DESARROLLO Y PRUEBAS DE POLÍTICAS DE SEGURIDAD SOBRE SERVIDORES WEB USANDO REFEREE

Castellanos Ariza, Francis

Fernández Segovia, Antonia

Zúñiga Galindo, Wilson

Asesor

Universidad Tecnológica de Bolívar

Maestría en Ciencias Computacionales

Cartagena de Indias

2001

INTRODUCCION

El estudio de la Herramienta REFEREE y su aplicación en el desarrollo y prueba de políticas de seguridad sobre servidores *WEB* es un proyecto propuesto por el Dr. Wilson Zuñiga director del Laboratorio de Cómputo Especializado de la Universidad Autónoma de Bucaramanga.

El proyecto pretende estudiar, y aplicar la herramienta REFEREE, que fue implementada por Yang-hua Chu como su tesis de Maestría en Ciencias de la Computación en el *Massachusetts Institute of Technology* en el año de 1997, en la implementación y prueba de las políticas de seguridad de un servidor *Web* mostrando todas sus ventajas y fortalezas en esta tarea.

1. MANEJO DE CONFIANZA

Hoy en día, la seguridad en el *Web* se asocia con la criptografía, protocolos y políticas públicas, obscureciendo de esta manera el gran reto de construir aplicaciones *Web* seguras. Como el objetivo del Web es ser un espacio donde la información refleje no solo el conocimiento humano sino también las relaciones humanas, en este pronto se reflejará la gran complejidad de las relaciones de confianza entre la gente, las computadoras y las organizaciones.

Dentro de la comunidad de seguridad de computadores, el **manejo de confianza** surge como una nueva filosofía, para codificar, analizar y manejar decisiones de confianza en sistemas abiertos (en contraste con las herramientas existentes de seguridad que son para sistemas cerrados).

1.1 ELEMENTOS DEL MANEJO DE CONFIANZA

Los elementos del manejo de confianza son :

- **Principios.** En el manejo de confianza se deben tener en cuenta los siguientes principios:

- **Ser específicos.** Se debe indicar quienes exactamente están en el equipo de confianza y que acciones pueden realizar.

- **Confiar en uno mismo.** Cada persona es dueña de su propio dominio, por tanto el principio de **confiar en uno mismo** indica que cualquier decisión de confianza debe ser lógicamente derivada de los axiomas que uno mismo crea.

- **Ser cuidadosos.** significa que se debe justificar de manera rigurosa cada decisión de confianza que se tome en la aplicación por muy simple que sea.

En conclusión, cuando se decide confiar en alguien superior para tomar alguna acción sobre un objeto se debe **ser especifico** acerca de los privilegios que se dan, se debe **confiar en uno mismo** para conceder los permisos y se debe **ser cuidadoso** antes y después de realizar este paso.

- **Participantes**. Los tres principios descritos anteriormente implican que todas las decisiones que otorgan confianza deben ser justificadas por una cadena de aserciones. Hay tres clases de participantes que permiten realizar estas aserciones: las personas, los

computadores y las organizaciones. Las personas son representadas por sus nombres, y pueden realizar aserciones de confianza utilizando firmas digitales. Las computadoras pueden mecánicamente verificar la integridad de la transmisión de los datos, ellas son representadas por su dirección y las organizaciones pueden representar un conjunto de personas o computadoras y pueden garantizar la membresía mediante credenciales y certificados firmados digitalmente.

Todo esto implica que para poder otorgar permisos para realizar una acción se debe tener en cuenta la persona que realiza el requerimiento, la computadora desde donde se realiza el requerimiento y que compañía avala el requerimiento.

- **Políticas**. Las políticas son las reglas acerca de cuales aserciones pueden ser combinadas para conceder permisos. Hablando ampliamente las políticas pueden conceder autorización basada en la identidad del que pregunta, en la capacidad de la computadora desde donde se pregunta, o en el objeto actualmente en uso. En otras palabras se puede confiar basándose en quien eres, que puedes hacer y que tienes. Para cada acción hay una política específica que indicará la justificación de la confianza.

Las políticas basadas en la identidad de la persona definen niveles de acreditación. Cada nivel indicará las acciones que pueden ejecutar aquellas identidades que pertenezcan a él.

1.2 HERRAMIENTAS PARA EL MANEJO DE CONFIANZA

Para determinar las políticas de seguridad en las aplicaciones Web se deben seguir los siguientes pasos:

- **Identificar los participantes**. El primer paso para construir un sistema seguro es identificar los usuarios autorizados y los que no lo están. El uso de passwords es la solución más común en los sistemas cerrados. La criptografía de llave pública es una manera más segura de manejar tales secretos, pero con el atenuante de que este sistema esta sobrecargado. El modelo de confianza de los *Webs* esta utilizando nuevas alternativas para la certificación de las personas, computadoras y organizaciones mediante el uso de llaves.

  Un certificado digital asegura el vínculo entre la llave criptográfica y su propietario como una aserción signada. El reto esta en decidir quien debería signar esta aserción y porque. Tradicionalmente una pirámide de autoridades certificadoras garantizan el vínculo. Ejem: "la llave de Antonia Fernández es 53, garantizado por Vc irvine, garantizado por Vc Agents, garantizado por el estado de California, garantizado por USA".

Los principios del manejo de confianza tienden a probar que están en contra de los certificados de identificación jerárquicos. Primero, una Autoridad Certificadora (AC) garantizada para identificaciones genéricas no puede ser específica acerca del grado de confianza involucrado, la capacidad verificada o el privilegio concedido sin conocer bien la aplicación que manejan. Segundo, confiando en Autoridades Certificadoras se atenúa el principio de confiar en uno mismo dado que estas requieren cubrir confianza en gran escala. Tercero, el reto lógico de listas de revocación centralizada hacen difícil ser cuidadosos usando estos certificados.

Las dos nuevas propuestas de infraestructuras de llave pública: Infraestructura de seguridad distribuida simple e infraestructura de llave pública simple son las que mejor se adaptan a los principios y al crecimiento descentralizado del *Web*. Primero son aplicaciones que especifican certificados que indican exactamente para lo que cada llave esta autorizada. Segundo ambos sistemas literalmente construyen una cadena de confianza que debe regresar al usuario. Finalmente ambos sistemas ofrecen una validación de certificados simple y en tiempo real.

- **Identificar Recursos**. El segundo paso en la construcción de sistemas seguros es asociar límites de acceso con cada elemento del sistema. La seguridad de metadatos debe incluir niveles de acreditación para los participantes del sistema, la capacidad requerida para cada acción o la llave para accesar un objeto.

Tradicionalmente los sistemas de computación seguros envían estos *bits* críticos dentro de la estructura de los datos y archivos. Dado que las herramientas para *Web* actualmente ofrecen solo una pequeña banda alrededor de las facilidades de seguridad, ellos usualmente ofrecen la misma clase de identificadores, tales como bits de permisos, de filesystem y límites de recursos en *script*.

- **Codificando y automatizando políticas.** El paso final para implementar un sistema seguro es especificar las decisiones de seguridad de acuerdo a alguna política. Como una plataforma de aplicación genérica el *Web* debería ser lo suficientemente flexible para acomodar un amplio rango de aplicaciones con una variedad de políticas de confianza centradas en los participantes, en los objetos o en las acciones.

REFEREE no usa un lenguaje de política simple, por esta razón. A pesar, de que los usuarios cargan dinámicamente los interpretadores para las políticas de seguridad, siempre mantiene una alto nivel de simplicidad en las decisiones de confianza. Dada una conjunto de hechos, una acción propuesta y su política, REFEREE puede determinar si la confianza es otorgada por siempre, algunas veces o nunca. *PolicyMaker* también provee un calculo de confianza simple.

Incorporar máquinas de confianza automatizadas retorna más control al usuario como un otorgador de confianza. La identificación descentralizada de participantes, la integración

de atributos de seguridad con metadatos del *Web*, y la flexibilidad de las políticas, todo lleva hacia el objetivo de manejo de confianza automatizable.

1.3 INTEGRACIÓN DEL MANEJO DE CONFIANZA EN EL WEB.

El Cambio de modelos de servicios cerrados (donde la comunidad de usuarios es conocida) a abiertos (accedidos públicamente) complica el análisis de seguridad en las aplicaciones *Web*. A continuación se muestran 2 ejemplos de aplicaciones *Web* donde es importante la seguridad.

- **Distribución segura de documentos**. Existen muchas aplicaciones *Web* donde existe un conjunto controlado de participantes con diferentes niveles de acceso, los cuales pueden observar un gran numero de documentos seguros. Anteriormente estas aplicaciones se montaban en mainframes donde la seguridad se manejaba a nivel del sistema operativo (con bases de datos de usuario/*password*) hoy en día estas aplicaciones se pueden colocar en servidores Web donde la seguridad se maneja a nivel del *Web* (con una infraestructura de certificación de llave pública).

- **Filtrar Contenido.** El control distribuido del *Web* y su rápido y continuo crecimiento hacen obsoletas las “listas negras” de proveedores de contenido censurable así como las

"listas blancas" de sitios conocidos como "buenos". Además estos mecanismos no permiten realizar juicios como: Si **.edu** es bueno y **sexo** es malo, que implica **sexo.edu**?.

El abordar este problema se esta expandiendo tan rápidamente como el *Web*. Por tanto el uso de PICS labels es lo mejor pues ellos también pueden escalar, porque los labels pueden ser asociados con los objetos por el autor o por una tercera parte.

Para representar propiamente las relaciones de confianza en estas aplicaciones, se debe tomar como base a las escuelas, librerías, oficinas y gobierno quienes tienen la voz en que constituye un contenido aceptable. Los filtros necesitan ser localizados dentro de la red bajo el control de cualquiera de estos entes.

1.4 DE SEGURIDAD EN WEB A MANEJO DE CONFIANZA

En medio de la acometida de nuevos protocolos, cifradores y todo tipo de herramientas relacionadas con industria de la seguridad en el *Web*, es fácil perder de vista el hecho de que las tecnologías convencionales de seguridad, aún implantadas perfectamente, no aseguran el manejo de confianza. Esto resulta de la incompleta o incorrecta especificación de las reglas de seguridad y del enfoque de los sistemas cerrados donde la seguridad en *Web* se traduce en seguridad en el servidor y en el cliente solamente. Para el manejo de confianza se

requiere que la seguridad sea manejada en varias capas de la conexión de red así como en los componentes de *software*.

El *Web* como un sistema de información no promulga artículos de política, ni corrupción de menores, ni reprogramación de computadores es simplemente un protocolo de Petición-Respuesta para enviar y/o recibir *bits* a través de la red. La seguridad en una transacción en el *Web* comprueba sólo la transferencia de *bits* de una máquina a otra sin intrusión.

Existen tres niveles en los que se puede proteger las transacciones *Web*. Ellos son: En la capa de transporte subyacente a HTTP, con los mensajes HTTP en si mismo o en el contexto de intercambio encima de HTTP.

La capa de transporte solo puede proveer seguridad a nivel de flujo de *bits*, por esta razón no puede utilizarse para asegurar documentos o acciones HTTP. Estas acciones son manejadas correctamente desde la capa de aplicación mediante el desarrollo de seguridad con mensajes HTTP. De igual forma los desarrolladores de aplicaciones pueden evadir completamente la seguridad en *Web* y construir sus propias soluciones usando el HTTP con una caja negra de intercambio de archivos.

Un servidor HTTP típico debería solicitar el nombre de usuario y la clave antes de conceder el acceso a algunos sitios URLs. Este método hace necesario que el administrador del servidor distribuya la información manejada en el sitio entre áreas protegidas y públicas

dependiendo de su importancia. Las políticas de seguridad en este contexto solo tratan la forma de una petición (correspondiente al método y el URL) pero no analizan la substancia o contenido que es el que puede resultar peligroso y atentar contra la integridad de los datos.

Al comparar los elementos del manejo de confianza(explicados en la sección 1) con el *Web server* se entiende lo difícil que resulta el construir sistemas de información con manejo de confianza sobre servidores *Web*. Entre los elementos analizados se tiene: pocos servidores *Web* pueden identificar claramente los tipos de participantes, pero aun los administradores a menudo confían en sistemas de seguridad muy débiles cuando en nuestros tiempos los servidores *Web* deben identificar claramente los usuarios que acceden a ellos debido a que existen muchas herramientas que realizan entre otras acciones la identificación de password, la modificación de direcciones IP o el cambio de las entradas en el DNS con el único fin de violar la seguridad y acceder de forma ilegal a servidor a realizar actos indebidos que van en detrimento del correcto funcionamiento del sitio afectado.

2. INFRAESTRUCTURA DE MANEJO DE CONFIANZA

2.1 INFRAESTRUCTURA IDEAL

La infraestructura del manejo de confianza representa un armazón conceptual del diseño de una solución coherente de varias decisiones de confianza que permitan a sus partes hacer aserciones de confianza sobre objetos que representan información y a las aplicaciones adquirir estas aserciones para que realicen decisiones de verdad con base en ellas.

La infraestructura planteada es independiente del criterio de confianza impuesto por una aplicación particular y del tipo de aserción realizada.

La infraestructura del manejo de confianza se divide en cuatro partes:

- **Formato de metadatos:** es un formato que permite describir información sobre un objeto y se conoce también con el nombre de aserción.

Los metadatos representan el medio por el cual fluye la confianza de la entidad que lo crea a la aplicación que es quien realiza la decisión de confianza por lo que se pueden entender los metadatos como "señas" de confianza.

Los metadatos en sí son objetos y pueden ser descritos por otros metadatos. La habilidad de formar cadenas de metadatos permite que la confianza se ramifique y descentralice dentro de una estructura *Web*.

- **Protocolo de confianza:** Es un método que le permite al sistema obtener aserciones de sus requerimientos, el único deber de un protocolo es el de obtener el conjunto apropiado de aserciones para que una petición dada cumpla con una política de confianza.

- **Lenguaje de políticas:** Es un lenguaje para especificar un conjunto de criterios confiables para ejecutar una acción dada.

- **Ambiente de ejecución.** Constituye un ambiente para interpretar políticas de seguridad y administrar protocolos de confianza. Un ambiente de ejecución toma peticiones de una aplicación *host* y le retorna una respuesta que cumpla con la política de seguridad.

A continuación se muestran gráficamente los componentes de la infraestructura del manejo de confianza y su dependencia funcional.

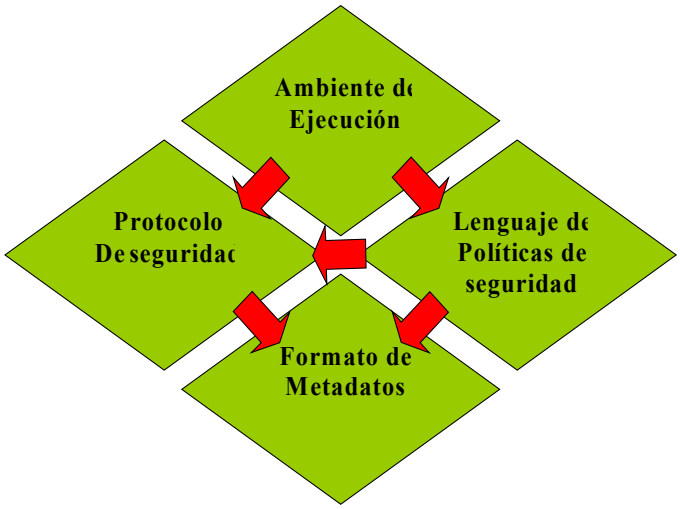

Figura 1. Gráfico de dependencia de los componentes de la infraestructura del manejo de confianza .

En la figura 1, las flechas indican las relaciones de dependencia con lo que podemos apreciar que el formato de metadatos es independiente de cualquier otro componente del sistema. Este puede describirse por múltiples protocolos y operar en múltiples lenguajes de políticas de seguridad.

El protocolo de confianza depende generalmente del formato de metadatos. Un protocolo típico contiene métodos para consultar metadatos específicos, una heurística para enlazarlos y formas de transportarlos. El lenguaje de políticas depende del protocolo de confianza y del formato de metadatos. Este debería entender la sintaxis y semántica de los metadatos para poder escribir una política.

El ambiente de ejecución depende del lenguaje de políticas de seguridad y del protocolo de confianza. El ambiente de ejecución debe ser capaz de interpretar las políticas y ejecutar los protocolos. Un ambiente de ejecución no entiende la sintaxis o la semántica del formato de datos directamente.

La mayoría de los sistema de manejo de confianza que existen pueden ser mapeados a esta estructura por lo que el trabajo de Yan Hua Chu se considera muy importante y definitivo en el ámbito de la seguridad en aplicaciones *Web*.

2.2 INFRAESTRUCTURA DE PROTOCOLOS DE SEGURIDAD EXISTENTES

Existen diversos sistemas y protocolos de manejo de confianza pero ninguno de ellos representa una solución completa y definitiva para aplicaciones *Web*.

A continuación se muestra como se enmarcan los protocolos existentes dentro de la infraestructura de manejo de confianza ideal que fue explicada en el ítem anterior.

- **PICS** (*Platform for Internet Content Selection*): es un estándar para la selección de contenido de internet y corresponde a un formato de metadatos y a un protocolo de confianza. Este sistema fue creado originalmente como un medio para proteger a los niños de la pornografía en Internet sin llegar a suprimir la libertad de expresión que se constituye en un derecho de todos. La figura 2, muestra la infraestructura de manejo de confianza de este protocolo:

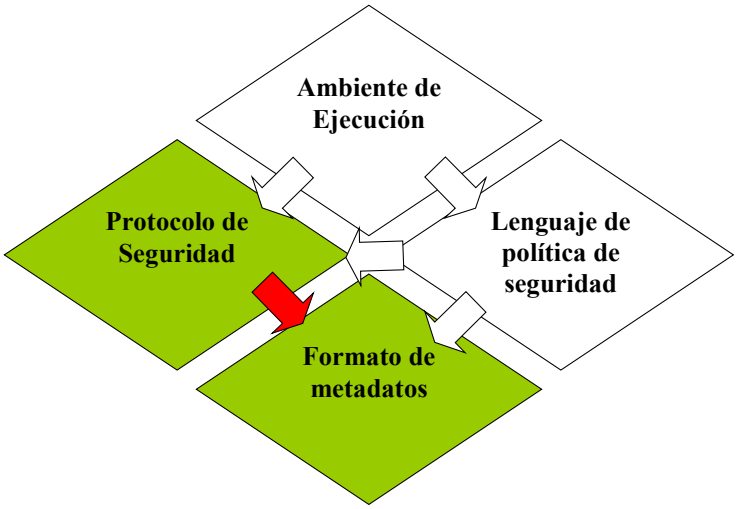

Figura 2. PICS en la infrestructura del manejo de confianza.

- **X509**: es un código para autenticación de usuarios en un servidor de directorio X.500. Este a menudo es referenciado como un esquema de certificación de identidad debido a que los certificados son instrucciones firmadas que trazan o representan una identidad en una llave pública. La siguiente figura muestra la infraestructura de confianza de este sistema.

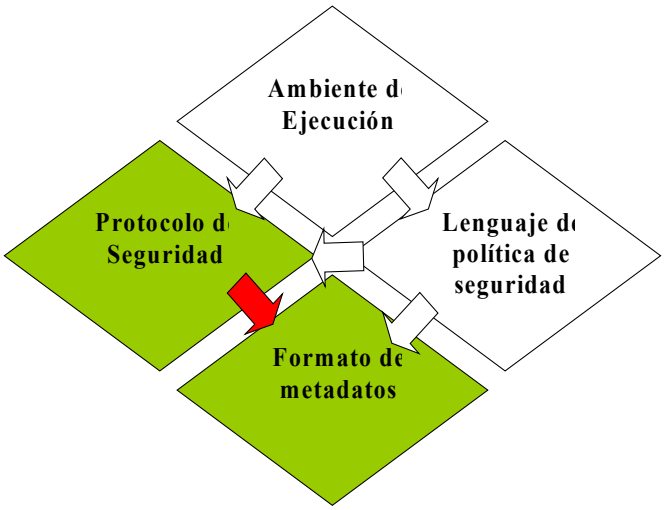

Figura 3. X.509 en la infraestructura de un sistema de manejo de confianza.

El protocolo de confianza en X.509 es simple y esta basado en la estructura de certificación jerárquica. X.509 no tiene un lenguaje de política de seguridad por lo que el camino de certificación es simplemente una estructura de datos para delegar confianza en la estructura de certificación. El X.509 es un buen estándar de autenticación de llave pública pero sólo, no basta para la mayoría de las aplicaciones *Web* existentes.

Desde el punto de vista de la infraestructura de manejo de confianza X.509 es similar a PICS. Ambos contienen y omiten los mismos componentes. Funcionalmente, PICS lleva información sobre un recurso de información y el X.509 lleva información sobre una entidad. Ambos son basados en el usuario y representan un paso importante hacia una infraestructura general de manejo de confianza.

- ***PolicyMaker***: Fue el primer sistema que trató de independizar las aplicaciones de los problemas de seguridad. El *PolicyMaker* esta formado por un formato general de metadatos (credenciales), un lenguaje de políticas de seguridad y un ambiente de ejecución. *PolicyMaker* no tiene el componente protocolo de seguridad que si tiene la infraestructura de confianza ideal. La figura 4 muestra como se sitúa al *PolicyMaker* dentro de la infraestructura de manejo de confianza ideal:

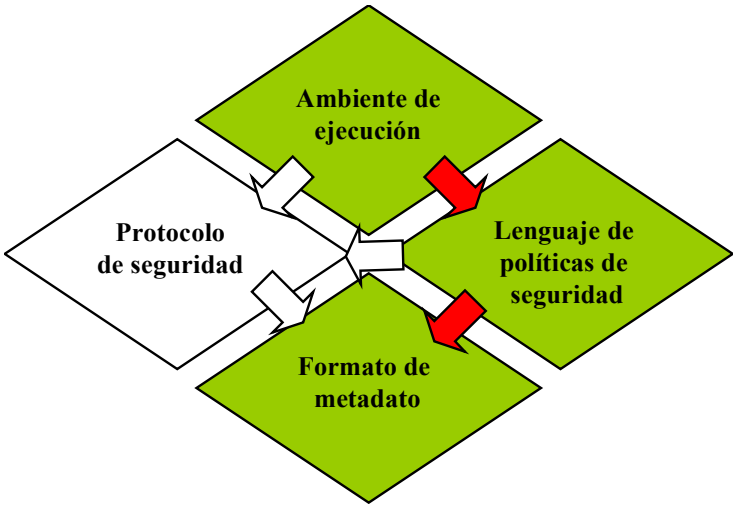

Figura 4. PolicyMaker en la infraestructura del manejo de confianza

3. AMBIENTE DE EJECUCION

El ambiente de ejecución es el corazón de un sistema de confianza porque es donde las políticas locales se encuentran con los protocolos de confianza completos y los formatos de metadatos para realizar las decisiones de confianza como una entidad interconectada y única de manejo de confianza.

El ambiente de ejecución tiene dos trabajos primarios que son: Interpretar las políticas de seguridad y administrar el protocolo de confianza. El ambiente de ejecución toma las peticiones de su aplicación *host* y retorna una respuesta que cumple con las políticas de seguridad.

REFEREE es un ambiente de ejecución propuesto por investigadores de Laboratorios AT&T y el grupo W3C. *Yan Hua Chu* como parte del grupo W3C participó en la definición y estandarización de REFEREE, colaboró en la definición general de la infraestructura de un sistema de confianza y realizó una implementación de REFEREE.

En REFEREE los protocolos de confianza y las políticas de seguridad son representadas como módulos de software que pueden ser invocados e instalados dinámicamente. Ellos

pueden compartir resultados intermedios completando un API determinado. Simultáneamente divide la tarea de manejo de confianza en pieza que tienen su tarea específica y la resuelven como un todo por la interacción de estas piezas. En el nivel de computo cada aspecto de REFEREE esta siempre sobre el control de una política.

3.1 OBJETIVO DEL DISEÑO

El objetivo de este ítem es mostrar una lista de propiedades que debería tener un ambiente de ejecución, aunque alguna de ellas puedan ser contradictorias es tarea del diseñador decidir que factores son más críticos en su aplicación e intentar usarlos.

- **Propósito general.** Un ambiente de ejecución fundamentalmente debería ser lo bastante poderoso para computar todas las decisiones de verdad que el usuario pueda realizar. Esto incluye varios grados de complejidad del requerimiento de las políticas del usuario y las credenciales.

- **Extensible:** Un ambiente de ejecución debería ser lo bastante extensible para acomodar dinámicamente nuevas piezas de los componentes en vez de retornar respuestas de “no se entiende la política” o “protocolo no puede actuar”.

- **Determinístico:** Si todas las entradas del ambiente son las mismas siempre debería retornarse las mismas respuestas aún cuando el orden de evaluación sea diferente.

- **Independencia de la plataforma:** El ambiente de ejecución debería ser independiente de las plataforma lo que significa que no debería basarse en atributos específicos del *host*. Lo que garantice que servirá para cualquiera de los sistemas operativos existentes como son: Windows, Unix, Mac, etc.

- **Eficiente**: El ambiente de ejecución y sus módulos de *software* deberían ser eficientes de tal forma que la aplicación no sienta una dramática caída en su rendimiento al realizar el manejo de confianza.

3.2 REFEREE

REFEREE no es una aplicación *standalone*, debe residir en una aplicación host. Con el propósito de que todos pudiesen entender REFEREE su desarrollador modeló la forma como interactúan una aplicación *Web* típica y REFEREE. (Ver figura 5).

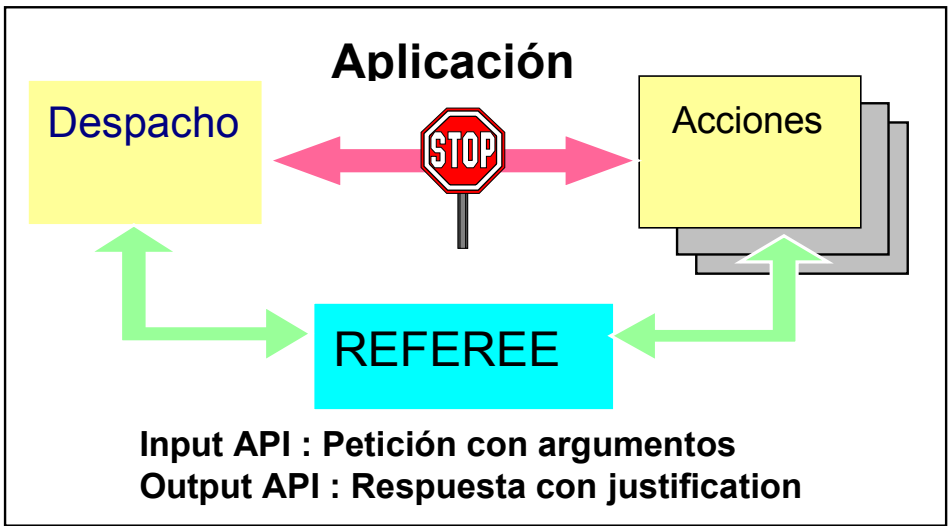

Figura 5. REFEREE External API.

El módulo de despacho es el responsable de generar peticiones cuyo tipo depende del contexto de la aplicación. Formalmente los requerimientos son despachados a los módulos de acción donde la acción toma lugar y sus resultados son retornados de vuelta al módulo de despacho.

REFEREE coloca una señal de pare entre el módulo de despacho y el módulo de acción cuando acciones potencialmente peligrosas requieren autorización. De esta forma el módulo de despacho consulta primero al REFEREE realizando toda una petición estándar API. REFEREE invoca las políticas de seguridad y los protocolos de confianza adecuados basándose en los requerimientos y sus argumentos asociados para retornar una respuesta que viene acompañada de alguna justificación.

El módulo de despacho esta entonces en la posición de decidir si continua el requerimiento al módulo de acción, modifica el requerimiento y lo reenvía a REFEREE o termina.

De esta arquitectura se puede ver a REFEREE como un sistema que realiza o da recomendaciones. Dado que el resultado retornado corresponde a una recomendación para el módulo de despacho quien puede o no ignorarla.

3.3 ARQUITECTURA INTERNA DE REFEREE

El ambiente de ejecución REFEREE es extensible y automodificable aunque aparentemente la aplicación *host* lo hace ver como un caja de decisiones de tres valores. El componente básico en REFEREE es el módulo. Un módulo REFEREE corresponde a un bloque de código ejecutable que procesa argumentos de entrada e instrucciones adicionales para encadenar otras tareas de diferentes módulos y hacer decisiones de confianza basado en las aserciones retornadas.

La interconectividad de los módulos de REFEREE hace que pueda procesar requerimientos de una aplicación *host* y producir una recomendación.

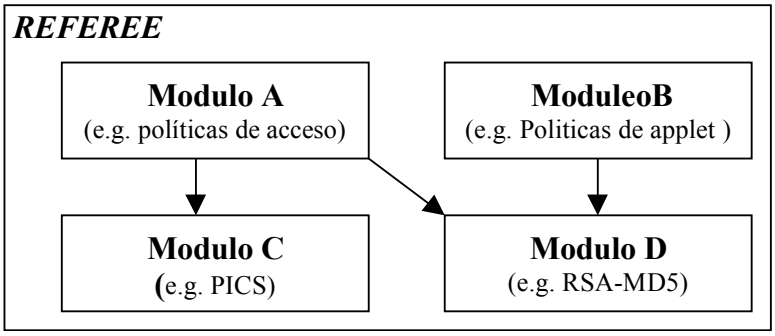

Figura 6. Diagrama simple de bloques de la estructura interna de REFEREE.

La figura 6, muestra un ejemplo del ambiente de ejecución con 4 módulos y sus flechas de dependencia. El módulo A contiene políticas de seguridad para ver páginas *Web*. El módulo B contiene políticas de seguridad para bajar *applets* JAVA. Los módulos A y B pueden llamar al módulo D para verificar firmas RSA-MD5. El módulo A puede llamar a C para recuperar *labels* PICS.

La separación de los servicios en los módulos de REFEREE tiene como ventaja que todos los módulos existentes pueden ser actualizados sin afectar a otros debido a que mientras se realiza la actualización el módulo guarda el API anterior compatible.

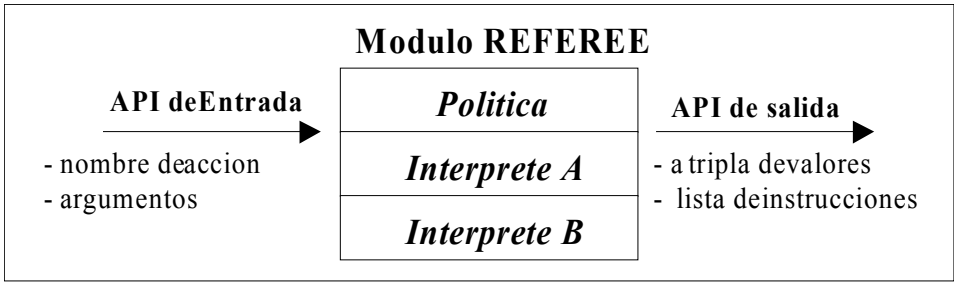

Figura 7. Interface requerida para cada módulo REFEREE.

La figura 7 muestra la interface para cada módulo de REFEREE que tienen el mismo API como el mismo REFEREE. La entrada es un nombre de acción con argumentos. Los argumentos proveen información adicional sobre la acción los cuales son probados incondicionalmente por los módulos. Si un módulo trata con selección de contenido los argumentos de entrada pueden ser el URL de interés o más bien la llave pública para hacer aserciones sobre el URL. La salida es una tripleta con una lista de instrucciones como justificación.

Todos los módulos de REFEREE deberían adherirse al mismo API para asegurar interoperabilidad entre ellos.

Internamente un módulo REFEREE consiste de una política y cero o más interpretes. Las políticas son fragmentos de código escritos en lenguaje de políticas de alto nivel y los interpretes son programas ejecutables para interpretar las políticas o a otros interpretes. El conjunto de interpretes en un módulo REFEREE es jerárquico, el módulo de políticas es interpretado por el interprete de más alto nivel, el cual a su vez es interpretado por uno de más bajo nivel y así. El interprete de más bajo nivel es interoperable por el subyacente ambiente de ejecución REFEREE.

La separación de políticas e interpretes tiene varias ventajas, la primera de todas es que el mismo interprete puede correr diferentes políticas de usuario mientras que ellas sean escritas en el mismo lenguaje de política. A la inversa, el mismo lenguaje de política puede

ser portado a diferentes aplicaciones habilitadas REFEREE usando diferentes interpretes. Sin embargo, desde el punto de vista de seguridad las políticas son generalmente más fáciles de probar que los interpretes debido a la complejidad de la construcción del lenguaje.

La arquitectura de REFEREE puede acomodar políticas escritas en lenguajes de alto nivel para usuarios promedio y en lenguajes de bajo nivel para usuarios sofisticados en el mismo ambiente de ejecución.

3.4 TIPOS DE DATOS PRIMITIVOS DE REFEREE

REFEREE tiene 3 tipos de datos primitivos y se explican a continuación.

3.4.1 Tri-Valores. Los Tri-valores son operandos lógicos, que puede tomar uno de los siguientes valores:

- Verdadero
- Falso
- Desconocido.

El concepto de verdadero y falso resulta familiar por la lógica Booleana. El valor adicional Desconocido refleja el hecho que algunas decisiones de confianza no son ni verdaderas ni falsas. Por ejemplo, cuando se pregunta acerca de la autorización de una acción particular, tal como "Debería aprobarse la siguiente orden?", típicamente hay tres posibles respuestas:

- Verdadera, significa "si, la acción puede ser realizada porque existen suficientes credenciales para que la acción sea aprobada".

- Falso, Significa "no, la acción no puede ser realizada porque existen suficientes credenciales para denegar la acción".

- Desconocido, significa "REFEREE es incapaz de encontrar suficientes credenciales para aprobar o negar la acción requerida".

En el tercer caso, el valor desconocido retornado por REFEREE obliga a la aplicación a decidir que acción tomar. Si la acción requerida es aprobar una orden, la aplicación *host* puede informar a las partes relevantes para realizar consideraciones adicionales acerca de aprobar o denegar la orden.

3.4.2 Declaraciones y lista de declaraciones. Las declaraciones son información adquirida durante la ejecución de los módulos. Son la información común que se intercambia entre los diferentes módulos. Todas las declaraciones son expresiones compuestas por dos elementos. El primer elemento lleva el contexto de la declaración y el segundo elemento provee el contenido de la declaración. Por ejemplo si un módulo de delegación REFEREE quiere hacer una declaración como "Bob no es confiable", esto se expresaría con la siguiente declaración:

(("*delegation-program*") ("Bob" (integridad 0)))

Si el contenido de una declaración expresa una autorización, el contexto indica la fuente de autoridad. La aplicación *host* u otro modulo REFEREE pueden hacer decisiones mas inteligentes de seguridad basados no solamente en lo que dice la declaración si no también en quien la hizo. El uso de declaraciones facilita un ambiente de ejecución REFEREE dinámico; los módulos REFEREE pueden delegar la evaluación de la confianza a otros módulos en REFEREE y conocer quien hace las declaraciones. Sólo las aplicaciones pueden delegar confianza a terceras partes y conocer quien hace la aserción.

Una lista de declaraciones es una lista ordenada de declaraciones. Estas generalmente actúan como una copia de declaraciones que un modulo REFEREE hace.

3.4.3 Módulos de base de datos. Un módulo de base de datos une nombres de acciones a módulos REFEREE, esto hace posible llamar un módulo por un nombre de acción, como es común en muchos lenguajes de programación. Un módulo de base de datos esta formado por entradas. Cada entrada es una tripleta: identificador, fragmento de código y nombre del lenguaje. El identificador es un *string* que únicamente identifica la entrada en su espacio local. El fragmento de código es la declaración de la política actual o el código del interpretador. El nombre del lenguaje es una cadena que identifica el lenguaje en el que el fragmento de código esta escrito y como interpretarlo. Un ejemplo de un módulo de base de datos es:

Identificador	Fragmento de código	Nombre del lenguaje
download-applets	>download-policy>	http://www.javasoft.com/jdk1.1/
view-URL	<view-URL-policy>	http://www.w3.org/PICS/profiles092/
http://www.w3.org/PICS/profiles092/	<profiles-0.92 interpreters>	http://www.javasoft.com/jdk1.1/

Para extraer un par, política e interpretador, de la base de datos el que llama suministra el nombre de una acción y una lista de nombres de lenguaje soportados por REFEREE. El modulo de base de datos primero encuentra una política en donde concuerde el nombre de acción con el identificador entrado. Si no lo encuentra la base de datos retorna un error. Si lo encuentra, chequea entonces si el nombre del lenguaje esta en la lista de lenguajes soportados por REFEREE. Si es así, la entrada es retornada como una política que no requiere interprete. Si no es así, la base de datos iterativamente busca interpretadores de

bajo nivel que puedan interpretar el lenguaje. Entonces la política y una lista de interpretes son retornados.

Por ejemplo en la tabla anterior si el nombre de la acción requerida es "*view-URL*" entonces la política es el fragmento de código identificado por "*view-URL*". El interprete asociado es el fragmento de código identificado por "*http://www.w3.org/PICS/profiles092*", asumiendo que REFEREE soporta Java JDK1.1.

Un módulo de base de datos puede instalar o no instalar ligaduras para controlar la disponibilidad de los módulos, adicionando o borrando ligaduras de la base de datos. No hay mecanismos de seguridad en los módulos de base de datos que por si mismos determinen que módulos puede instalar, desinstalar o consultar la base de datos.

3.5 CARGA INICIAL FUERTE (BOOTSTRAPPING) DE REFEREE

Hay dos etapas en el tiempo de vida de REFEREE: la etapa de carga inicial fuerte, seguida de la etapa de consulta. Durante la etapa de carga inicial fuerte la aplicación *host* provee a REFEREE de una cantidad de información que le permita estar informado por si mismo. Después de la etapa de carga inicial fuerte, REFEREE entra a la etapa de consulta donde los requerimientos son aceptados y procesados.

Hay dos piezas de información suministradas por la aplicación *host* durante la etapa de carga inicial fuerte:

- Aserciones de seguridad
- Módulos de base de datos.

Toda la información suministrada en la etapa de carga inicial fuerte es incondicionalmente confiable. Las aserciones de seguridad son aserciones claves que son utilizadas por REFEREE en sus operaciones, tales como credenciales y certificados. El modulo de base de datos contiene un conjunto mínimo de ligas que la aplicación *host* espera usar.

3.6 CONSULTAS REFEREE

Una vez que REFEREE finaliza la etapa de carga inicial fuerte, esta listo para procesar las consultas de la aplicación *host*. La figura 8 muestra los pasos que se siguen para realizar una consulta con tres módulos de software.

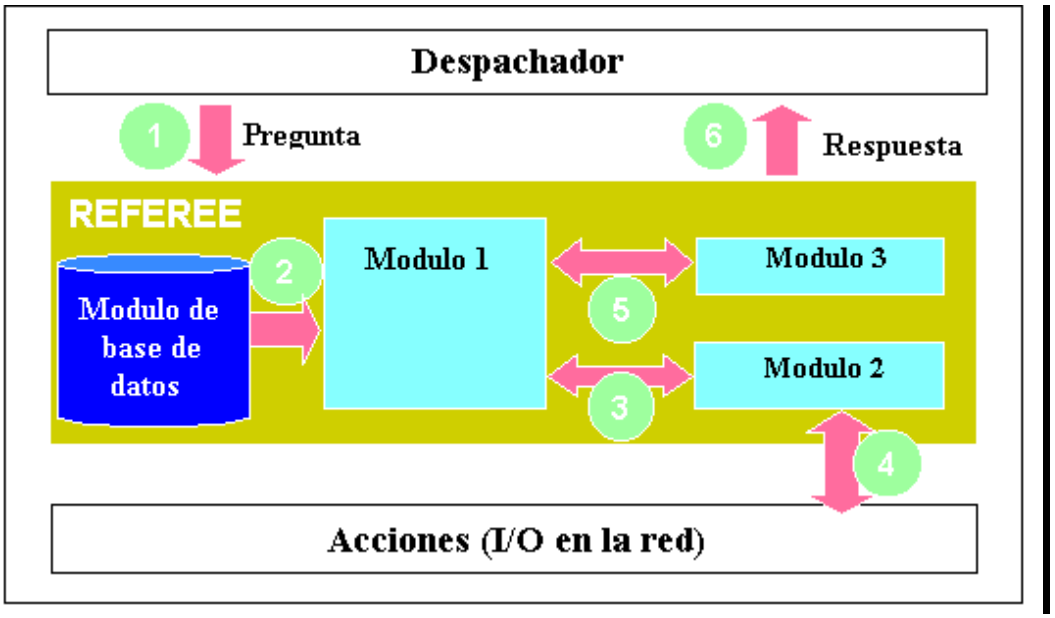

Figura 8. Pasos para realizar una consulta

Primero, el despachador en la aplicación *host* realiza una consulta a REFEREE, la cual consiste de una acción con una lista de argumentos (paso 1). Cuando REFEREE recibe la consulta, este arranca el módulo REFEREE apropiado (módulo 1) del módulo de base de datos (paso 2), el cual consiste de un par: política e interpretador. REFEREE invoca al interpretador con la política y la lista de argumentos de entrada. Durante la interpretación de la política, el módulo puede invocar otros módulos (paso 3 - 5), los cuales pueden a su turno llamar a las acciones específicas del *host* previstas por la aplicación *host* (paso 4). Cuando el módulo 1 finaliza la interpretación, REFEREE retorna al despachador los valores retornados por el módulo 1.

4. POLÍTICAS DE SEGURIDAD

4.1 NECESIDAD DE POLITICAS DE SEGURIDAD Y PROCEDIMIENTOS

El mundo de los computadores ha cambiado dramáticamente en los últimos tiempos. Hace 25 años, se manejaba el esquema centralizado en el procesamiento de la información. Existían equipos gigantescos guardados en los denominados centros de cómputo – cuartos cerrados, con normas y técnicas de seguridad definidas - y solo un grupo especializado de personas se encargaban del cuidado y seguridad de ellos. La conexión a sitios externos era inusual, Por lo que las amenazas a la seguridad de los computadores eran muy raras y se presentaba básicamente cuando personas de confianza (usuarios autorizados) de una forma conciente o inconsciente abusaban de las cuentas o cometían actos de robo y vandalismo. Estas amenazas estaban bien identificadas y se utilizaban técnicas como: acceso restringido a los computadores y contabilidad del uso de todos los recursos, para contrarrestarlas.

La computación en los 90s es radicalmente diferente. De hecho se paso del esquema de computo centralizado al distribuido con lo que surgen redes y subsistemas en oficinas privadas y laboratorios, muy distintos al concepto de centro de cómputo y manejados frecuentemente por individuos o personas sin especialidad en el área. Con el auge del

Internet se logró la globalización de la información permitiendo la interconexión de sistemas remotos y el uso y distribución de la información a toda partes del mundo.

Hoy en día las amenazas a la seguridad son muy diferentes a las que se tenían hace algunos años, pues se pasó de poder identificar completamente a los usuarios a tener cualquier tipo de usuarios potenciales mediante la conexión a Internet; de esta forma alguien podría entrar a un sistema desde el otro lado del mundo y robar *passwords,* extraer información, dañar equipos de cómputo en la mitad de la noche cuando el edificio este cerrado. Los virus y *worms* pueden ser pasados de una máquina a otra mediante la interconexión existente sin siquiera dejar huellas explicitas de su procedencia. El uso de Internet y el auge de las redes de comunicación han incrementando los problemas de seguridad inicialmente detectados, haciendo mas vulnerables los sistemas de computo y la información propiamente dicha. En estos momentos una persona puede chequear, modificar e incluso destruir miles de máquinas en unas pocas horas.

Las organizaciones y en ellas el persona encargado de administrar todo lo relacionado con los sistemas de información tienen que entender que estas amenazas a la seguridad existen y que en cualquier momento puede estar expuesto a la destrucción parcial o total de sus sistemas. Con este riesgo totalmente identificado deben tomarse medidas preventivas para evitar ataques y se deben tener planes de contingencia que permitan conocer en poco tiempo el costo de un ataque y las acciones que deben tomar para prevenir y responder a ellos.

Aunque la conectividad a Internet ofrece enormes beneficios en términos de incrementar el acceso a la información, es peligrosa para aquellos sitios con bajos niveles de seguridad. El Internet sufre de problemas muy evidentes de seguridad que, si son ignorados, podrían traer resultados desastrosos para sitios no preparados. Problemas inherentes con los servicios de TCP/IP, la complejidad de la configuración del host, vulnerabilidades introducidas en los procesos de desarrollo del software, y una variedad de otros factores han contribuido a hacer que los sitios no preparados estén abiertos a problemas y actividades relacionadas con intrusos.

Preguntas como : Perturbarán los intrusos los sistemas internos?, Los datos valiosos de la organización estarán comprometidos (cambiados o leídos) durante cualquier transmisión?, La organización se pondrá en aprietos y perderá toda su información vital?, son muy válidas en estos momentos y las organizaciones deben colocar todo su empeño y esfuerzo en resolverlas. Muchas soluciones técnicas están surgiendo dirigidas a la seguridad en Internet. Estas soluciones son muy diversas y su uso depende del grado o nivel de seguridad que requiera la organización, pueden ir desde limitaciones funcionales del sistema hasta la adquisición de recursos costosos y personal especializado; lo que si debe ser claro es la definición de las políticas de seguridad que se deben seguir para garantizar el éxito de cualquier plan implementado.

El propósito de una política de seguridad en Internet es decidir como una organización se va a proteger por si misma. Las políticas de seguridad usualmente requieren dos partes: una política general y reglas específicas (las cuales son equivalentes a los sistemas de políticas descritos anteriormente). La política general establece lo que abarcará la política, la seguridad de Internet. Las reglas definen lo que esta o no permitido. Las reglas pueden ser remplazadas con procedimientos y otras guías.

Para que una la política de seguridad en Internet sea efectiva, debe hacer sobrentender los cambios que hay que realizar. La política debe estar también sincronizada con los resultados de otras políticas y entre todas conllevan a ofrecer la seguridad al sitio o servicio que se esta protegiendo.

El Internet es un recurso vital que esta cambiando la manera en que muchas organizaciones e individuos se comunican y hacen negocios. Sin embargo, el Internet sufre de significativos y extensos problemas de seguridad. Muchas compañías han sido atacadas o investigadas por intrusos, con resultados de pérdida de productividad y reputación. En algunos casos, las organizaciones han sido desconectadas de Internet temporalmente y han invertido recursos significativos en corregir los problemas con configuración de los sistemas y de la red. Los sitios que son inconscientes o ignorantes de estos problemas afrontan el riesgo de que los intrusos de redes los ataquen. Aun los sitios que tienen buenas prácticas de seguridad afrontan problemas por las nuevas vulnerabilidades en software de red y por la persistencia de algunos intrusos.

El problema fundamental es que Internet no fue diseñada para ser muy segura. Algunos de los problemas de Internet son:

- Fácil de escuchar y engañar. La mayoría del tráfico en Internet no es encriptado. Los email, passwords y archivos transferidos pueden ser monitoreados y capturados usando software fácilmente disponible.

- Servicios TCP/IP vulnerables: Algunos servicios TCP/IP no son diseñados para ser seguros y pueden ser comprometidos por intrusos inteligentes; los servicios utilizados para comprobaciones son particularmente vulnerables.

- Complejidad de configuración: Los *Host* de seguridad de control de accesos son frecuentemente complejos de configurar y monitorear, los controles que son accidentalmente mal configurados pueden resultar en accesos no autorizados.

En nuestro trabajo nos centraremos en las políticas de seguridad que se deben definir en la empresa cuando esta conectada a Internet y específicamente las políticas para los usuarios que utilizan el *Web*.

4.2 COMO DEFINIR POLÍTICAS

Al definir políticas de seguridad, lo primero que debe preocupar es qué se intenta proteger y contra qué se intenta proteger, para de esta manera poder establecer las directrices que proporcionarán el nivel de seguridad esperado para la organización.

Cuando una organización trabaja en red pone en riesgo tres cosas:

- **Sus datos**: Corresponden a la información que se guarda en las computadoras y que es de vital información para el desarrollo de la empresa. Las características de los datos que deben protegerse son:

  - Confidencialidad: Se espera que cierta información sea conocida sólo por las personas autorizadas.

  - Integridad: Se espera que la información no sea cambiada indebidamente.

- Disponibilidad: Se espera que el acceso a la información esté disponible en el momento que se requiera.

- **Sus recursos**: Las computadoras en sí. Los recursos de cómputo (espacio en disco, tiempo de proceso, memoria) son elementos valiosos en la operación de la organización, por lo cual deben protegerse del uso inadecuado o de personas no autorizadas.

- **Su reputación**. La reputación, por otro lado, tiene un valor incalculable, el cual es expuesto si se vuelve de conocimiento público que a través de la red se ha afectado la confiabilidad, integridad o disponibilidad de los datos o si inclusive se han estado utilizando los recursos de cómputo de manera indebida.

Conociendo qué es lo que se intenta proteger, el paso siguiente es identificar contra qué se intenta proteger.

Una forma de clasificar los ataques contra los sistemas de información es agruparlos en tres categorías básicas: intrusión, negación del servicio y robo de información.

- **Las *intrusiones*** son uno de los ataques más comunes, pues con ellas las personas pueden utilizar los computadores como si fueran usuarios legítimos, para tener la posibilidad de acceder a la información allí almacenada o procesada.

- **Los ataques de *negación del servicio*** están dirigidos a evitar que se utilicen los recursos de cómputo de manera normal, o que queden inhabilitados totalmente para atender cualquier tipo de servicio.

- **El *robo de información*** consiste en la obtención ilícita de información, que puede ser desde claves de acceso a sistemas de información hasta documentos privados de la organización. Para realizar este tipo de ataques se pueden utilizar desde procedimientos técnicos para capturar la información en los sistemas o mientras están en tránsito por la red hasta la manipulación social de personas que tienen a cargo la confidencialidad de dicha información.

Existen diferentes tipos de atacantes que van desde los que buscan algún tipo de diversión, pasando por los vándalos que quieren causar daños por diversas razones, hasta llegar a los espías contratados por la competencia.

A las acciones de los atacantes propiamente dichos hay que incluir los accidentes de usuarios lícitos de los sistemas de información. Según un estudio en 1995 del Instituto de Seguridad en Computadores, de San Francisco, California, el 55% de todos los accidentes que involucran seguridad resultaron de usuarios ingenuos o sin entrenamiento que hacen las cosas que no debieran.

Es importante tener claro que el mayor porcentaje de los riesgos de seguridad se presentan con el personal interno de la organización, sólo que las acciones contra la seguridad realizadas por personas externas a la organización son por lo general más publicitadas. Según el reporte anual de estadísticas de seguridad de la CIA en 1997, el 85% de las pérdidas por seguridad informática reportadas en los EEUU se debieron por los usuarios propios o internos *"insiders"* y no por los usuarios externos *"outsiders"*.

Después de identificar lo que se quiere proteger y contra que se quiere proteger se debe definir el término en que se definirán las políticas para lo cual hay dos alternativas:

- *Aquello que no se permite en forma expresa, está prohibido*, es el primer paso a la seguridad. Esto significa que la organización que defina las políticas de esta manera ofrece un grupo de servicios preciso y documentado, y que todo lo demás esta prohibido. Por ejemplo, si se decide permitir transferencias FTP anónimas hacia y desde una

máquina particular, pero no acepta servicios telnet, entonces el soporte documental para FTP ilustra este planteamiento, y no el de telnet.

- *Aquello que no esté prohibido de manera expresa se permite*. Esto significa que amenos que se indique en forma expresa que un servicio no esta disponible, entonces todos los servicios estarán disponibles. Por ejemplo si no se dice con claridad que las sesiones de telnet a un anfitrión dado están prohibidas, entonces quiere decir que están permitidas.

Sin importar que alternativa se tome, la razón para definir una política de seguridad, es determinar que acción deberá tomarse en caso de que la seguridad de la organización se vea comprometida. La política también intenta describir que acciones serán toleradas y cuáles no.

4.3 CONSTRUCCION DE POLITICAS DE SEGURIDAD EN EL AMBIENTE REFEREE

Luego de presentar una visión global de la necesidad de la definición de las políticas de seguridad para los ambientes *Web*, nos centraremos en la forma como REFEREE permite la definición correcta de políticas de seguridad. En los capítulos anteriores se mostró de manera muy general la forma como el ambiente de ejecución REFEREE procesa consultas,

interpreta políticas de seguridad y corre protocolos de confianza de una manera genérica e independiente de la aplicación.

Este capitulo se centra en la manera de crear o modificar políticas de seguridad en el ambiente REFEREE mediante el uso de los lenguajes interpretados y de propósito específico PicsRULZ y Profiles 0.92, diseñados para esta función particular.

De manera general, un lenguaje para la construcción de políticas de seguridad describe una política de usuario en una forma clara y precisa cumpliendo con las siguientes propiedades:

- **Seguridad:** La escritura de políticas de seguridad en un lenguaje de políticas evita la producción de códigos dañinos que afecten a la aplicación *host* o a las políticas subyacentes.

- **Tranferible:** Las políticas escrita en un lenguaje de políticas deben ser transferible a diferentes aplicaciones e independiente de la plataforma. Esta propiedad permite tanto a los usuarios como a las compañías definir perfiles independientes y trabajar en otras locaciones sin reconfiguración.

- **Simple:** El diseño de un lenguaje de políticas debe ser simple y específico permitiendo definir las políticas de seguridad y sus futuras extensiones, pero sin llegar a la complejidad de un lenguaje de propósito general que sigue el sentido completo de la máquina de Turing.

- **Bien definido.** Un lenguaje de políticas debe permitir la escritura de políticas de una manera clara y precisa y sin ningún tipo de ambigüedad.

- **Expresivo**: El lenguaje de políticas debe ser capaz de expresar cantidad de políticas reales que diferentes usuarios quieran especificar en circunstancias diferentes. El nivel de expresividad puede depender de la habilidad en programación de la persona que crea la política o de la complejidad de la herramienta.

4.3.1 PicsRULZ. Es un lenguaje de políticas interpretado basado en reglas. Este lenguaje fue construido para la escritura de reglas basándose en peticiones URL o su atributo PICS asociado. El PICSRULZ es un leguaje de políticas de seguridad simple y conciso y en el que se puede trabajar con el protocolo PICS y formatos de metadatos.

PICSRULZ es el resultado del trabajo del grupo de investigadores del lenguaje de perfiles PICS del *World Wide Web Consorsium*. La descripción de este lenguaje fue presentada en el encuentro sobre PICS en Abril 10 de 1997 en Santa Clara California.

El Lenguaje esta organizado en cláusulas que se clasifican en 7 formas diferentes:

- ***FailURL***: Esta cláusula es un método para expresar una lista de prefijos de URL que van a ser bloqueados de manera explícita. Si un URL requerido se encuentra en la lista de *FailURL* la evaluación termina inmediatamente y se bloquea el acceso a dicho lugar. El FailURL tiene el valor más alto en el orden de precedencia y puede aparecer más de una vez en una regla PICSRULZ pero, se recomienda la existencia de un regla general de este tipo.

  Sintaxis: FailURL(Dirección 1, Dirección 2, ..., Dirección n)

- ***PassURL***: Tiene la misma sintaxis del FailURL pero su sentido es completamente contrario. Permite visitar aquellos lugares con prefijo definido en la lista de direcciones.

- ***ServiceInfo***: Especifica la información sobre la clasificación del servicio y contiene cinco atributos:

  Name: es el URL de la clasificación del servicio.

Shortname: Asocia una variable local a la clasificación del servicio.

BureauURL: Asocia la localización del *Label Bureau* para recuperar los labels PICS.

Ratfile: Contiene el URL actual.

BureauUnavailable: es un mecanismo de excepción para especificar que hacer cuando el *Label Bureau* no puede ser contactado.

Sintaxis: Serviceinfo (Name <Direccion>

Shortname <nombre>

BureauURL <Direccion del Bureau>

Ratfile <valor>

BureauUnavailable <valor>)

- ***Filter*:** Opera sobre los *Label PICS* obtenidos de la cláusula *ServiceInfo*, esta cláusula se divide en dos subexpresiones : *Pass-expression* y *Block-expression* unidas por un

AND lógico. Esto es, un URL pasa el filtro sólo si la *pass-expression* es verdadera y la *block-expression* es falsa.

Sintaxis: Filter (Pass-expression "Expresión", Block-Expression "Expresion")

Las expresiones son comparaciones de atributos de un *PICS label* y constantes.

- ***Extensión***: Provee la forma de extender la funcionalidad de PICSRULZ. Como en la extensión de PICS 1.1 existen extensiones obligatorias y opcionales. Si una extensión es requerida u obligatoria el evaluador de la regla debe entenderla y evaluarla, en cambio las extensiones opcionales no necesitan ser evaluadas.

- ***Name***: Esta cláusula provee información local sobre la regla para facilitar la construcción de interfaces.

Sintaxis: Name (rulename " nombre-regla"

Description " <descripción de la regla>")

- ***Source****:* Provee información sobre la procedencia de una regla y está compuesto de dos campos: *SourceURL:* Campo que identifica la regla y *CreationTool*: Campo que identifica como es construida la regla.

Sintaxis: Source (sourceURL <dirección>

CreationTool <nombre>

Autor <nombre>

LastModified <fecha>)

4.3.2 PROFILES - 0.92. El lenguaje profiles 0.92 fue desarrollado junto con el sistema de manejo de confianza REFEREE. Es un lenguaje de políticas flexible y modular cuyo objetivo es explotar y mostrar las características importantes de REFEREE en la construcción y definición correcta de las políticas de seguridad.

Una instancia del lenguaje profiles 0.92 es una política, que se puede interpretar como una secuencia ordenada de reglas. Cada una de las reglas es expresada como una *s-expresión*, en la cual el primer símbolo es un operador y el resto de los símbolos son operandos. La evaluación de una política es *top-down* por lo que el valor retornado por la última regla es el valor retornado por la política. La respuesta de una política tiene la misma estructura de las respuestas de REFEREE: un tri-valor con una lista de declaraciones como justificación.

Profiles 0.92 es mucho mas extenso y expresivo que PicsRULZ, aunque todavía no se pude considerar como un lenguaje de propósito general.

Para la escritura de políticas de seguridad en el lenguaje Profiles 0.92 se utilizan las siguientes funciones:

- **URL Coincidencia de prefijos**

(url-match URL (<URL-prefijo>+) [<prefijo-coincide?>])

Esta función provee un medio claro de retornar un tri-valor basado en la búsqueda de una cadena a través de un URL en particular. El primer argumento es el símbolo URL. El segundo argumento es una lista de cadenas que se desean buscar dentro del URL dado. El tercer argumento es un valor booleano que determina si se debe encontrar exactamente la cadena o solo un prefijo. Si este argumento es verdadero entonces se debe encontrar exactamente la cadena en el URL para poder retornar el valor verdadero. Si es falso, una de las cadenas encontradas debe tener de prefijo la cadena buscada para poder retornar el valor verdadero.

- **Coincidencia de patrones.**

(match <patrón> <lista de sentencias>)

La función de coincidencia de patrones busca la expresión <patrón> en las sentencias que están en <lista de sentencias>. Sucede una coincidencia cuando un patrón y una sentencia son sintáctica y estructuralmente equivalentes.

- **Formas de Combinar los *Tri-Value.***

el lenguaje Profile-092 provee seis operadores de tri-values:

(*and* <regla>+)

(*or* <regla>+)

(*threshold-and* <num> <regla>+)

(*not* <regla>)

(*true-if-unknow* <regla>)

(*false-if-unknow* <regla>)

El operador *and*, *or* y *threshold-and* son operadores de multi-argumentos y *not*, *true-if-unknow* y *false-if-unknow* son operadores unarios. Cada operador de multi-argumentos toma cero o mas reglas como entrada. El tri-valor de salida se basa en los cálculos realizados sobre los tri-valores de entrada y la lista de sentencias de salida es una concatenación de las listas de sentencias de entrada.

- **El operador *and***

Es la versión en tri-valores del operador booleano *and*. La tabla 1 describe la operación *and* cuando se dan dos argumentos. La primera fila representa el valor de verdad del primer argumento, la primera columna representa los valores de verdad del segundo argumento y el resto de las celdas representa el resultado de la operación *and*.

Regla1\Regla2	Verdadero	Desconocido	Falso
Verdadero	verdadero	desconocido	falso
Desconocido	desconocido	desconocido	falso
falso	falso	falso	falso

Tabla 1. Tabla de verdad del operador and

El operador *and* puede tomar cualquier número de argumentos. Para más de dos argumentos, el operador *and* reduce recursivamente la expresión para trabajar con dos argumentos al tiempo y esto lo hace de la siguiente forma:

(and arg1 arg2 ... argn) = (and (... (and arg1 arg2) ... argn)

El *and* de un solo argumento da como resultado el mismo argumento y el *and* sin argumentos es verdadero por definición. Si uno de los argumentos retorna falso, la regla *and* termina y retorna falso.

- **El operador *or***

Es la versión en tri-valores del operador *or* booleano. La tabla 2 muestra la operación *or* cuando se dan dos argumentos.

Regla1\Regla2	Verdadero	Desconocido	Falso
Verdadero	verdadero	verdadero	Verdadero
Desconocido	verdadero	desconocido	Desconocido
falso	verdadero	desconocido	Falso

Tabla 2. Tabla de verdad para el operador or.

Igual que el operador *and*, el operador *or* puede tomar cualquier número de argumentos, y ellos son recursivamente reducidos si hay más de dos. El *or* de un solo argumento es el mismo argumento y el *or* sin argumentos es falso por definición. Si uno de los argumentos evaluados es verdadero, entonces la regla termina y retorna el valor verdadero.

- **El operador *not***

Este operador es la versión en tri-valores del operador *not* booleano. Este operador toma solo un argumento. La tabla 3 muestra la tabla de verdad del operador *not*.

Entrada	Salida
Verdadero	falso
Desconocido	desconocido
Falso	verdadero

Tabla 3. Tabla de verdad del operador not.

- **El operador *true-if-unknow***

Es una función de proyección de la lógica de los tri-valores a la lógica booleana. Toma exactamente un argumento. La siguiente tabla muestra la tabla de verdad de este operador.

Entrada	Salida
Verdadero	verdadero
Desconocido	verdadero

falso	falso

Tabla 4. Tabla de verdad del operador true-if-unknown.

- **El operador *false-if-unknow***

Es también una función de proyección de la lógica de tri-valores a la lógica booleana. Toma exactamente un argumento.

Entrada	Salida
Verdadero	verdadero
Desconocido	Falso
falso	Falso

Tabla 5. Tabla de verdad del operador false-if-unknown.

- **El operador *threshold-and***

Este operador implementa la semántica "cualquier m de n" en una lista de argumentos tri-valores. El *threshold-and* toma al menos un argumento, el valor *threshold* es un valor entero no negativo. La sintaxis de este operador es:

(*threshold-and threshold arg1 arg2 arg3 ... argn*)

Sea n_v, n_f y n_d el número de argumentos verdaderos, falsos y desconocidos respectivamente que son evaluados por *threshold-and*. Se tiene que $0 <= n_v, n_f, n_d <= n$ y además $n_v + n_f + n_d = n$. El valor de *threshold-and* es calculado de la siguiente manera:

- Si $n_v >=$ *threshold* se retorna verdadero,
- Sino si $n_v <$ *threshold* y $n_v + n_d >=$ *threshold*, retorna desconocido,
- Sino si $n_v + n_d <$ *threshold*, retorna falso
- por definición, (*threshold-and* 0) retorna verdadero.

- **Invocación**

invoke <nombre política> <lista de sentencias> <argumentos adicionales>)

Invoke llama a la política que se indica en <nombre política> con una copia de <lista de sentencias> y posiblemente algunos argumentos adicionales. Cuando la política que se llama retorna, este valor retornado es un par que consiste de un *tri-value* y

una lista de sentencias. Por convención, el modulo que llama adiciona la lista de sentencias retornada por el modulo llamado a su propia lista de sentencias.

- **Instalación**

(install-policy <lista de sentencias>)

(install-interpreter <lista de sentencias>)

Install-policy crea políticas en el modulo de base de datos que se esta construyendo. *Install-interpreter* crea interpretadores en el modulo de base de datos que se esta construyendo.

En ambos casos la información necesaria para realizar la construcción es pasada adentro de una lista de sentencias, cuyas sentencias son de la forma:

((<contexto>) (<identificador> <fragmento de código> <nombre del lenguaje>)).

5. IMPLEMENTACIÓN DE POLÍTICAS DE SEGURIDAD CON REFEREE

Como una manera de convalidar la funcionalidad del software de manejo de confianza REFEREE, se desarrolló una aplicación prototipo en la que el control de seguridad esta definido por políticas implementadas en el *software* REFEREE.

5.1 REQUERIMIENTOS

Para que la aplicación desarrollada pueda ejecutarse correctamente y el manejo de sus políticas de seguridad sea otorgado por el *software* REFEREE se deben tener los siguientes elementos:

- Software REFEREE: Aplicación Java para la verificación y manejo de políticas.

- Applet Ref1: Applet que utiliza el *software* REFEREE para la verificación del cumplimiento de las políticas de seguridad.

- Archivos de configuración de políticas: Archivos de soporte a la verificación de políticas de seguridad que contienen las direcciones sobre las cuales se van a realizar controles de seguridad.

- Plug-in Java: Software que permite que el *browser* corra un applet Java.

- Archivo de Seguridad de Java: Archivo que otorga a la aplicación Java los permisos de acceso a la máquina del usuario.

- Archivo INS-REF.exe: Archivo de instalación que contiene la aplicación prototipo y el software REFEREE.

5.2 PROCEDIMIENTO DE INSTALACIÓN DE REFEREE

El proceso de instalación de REFEREE se puede realizar de dos formas dependiendo de las necesidades del entorno en que se desee trabajar.

Si se opta por la instalación de REFEREE o cualquier desarrollo que lo utilice se deben tener en cuenta los derechos de autor correspondientes a al Dr. Yang-hua Chu por haber implementado REFEREE como tesis en la Maestría de ciencias de la Computación del Massachussets Institute of Technology.

5.2.1 INSTALACION LOCAL

Para realizar la instalación local de la aplicación prototipo se debe seguir los siguientes pasos:

- Ejecutar el archivo INS-REF.exe: El resultado de la ejecución de este programa es la instalación de toda la estructura de directorios del *software* REFEREE y de la aplicación prototipo desarrollada, el programa INS-REF.exe corresponde a un programa ejecutable auto-extraible que descomprime la aplicación prototipo con toda la estructura de directorio tomando como raíz el C:\.

En la Figura 9 se puede observar la estructura de directorios definida luego de la descompresión del *software* y se destacan los siguientes elementos:

DOC: Este directorio corresponde a documentación relacionada con el diseño de REFEREE.

SRC: Directorio que contiene los programas REFEREE.bat y REFEREE.html correspondientes a las dos formas de correr la aplicación prototipo, los archivos SITCON y SITRES que proporcionan la configuración de la aplicación y el archivo .java.policy que es el archivo de políticas que le permitirá a REFEREE poder acceder la máquina desde donde se esta ejecutando.

REFEREE: Directorio que contiene todos los programas JAVA que componen el REFEREE y el applet desarrollado para la aplicación prototipo.

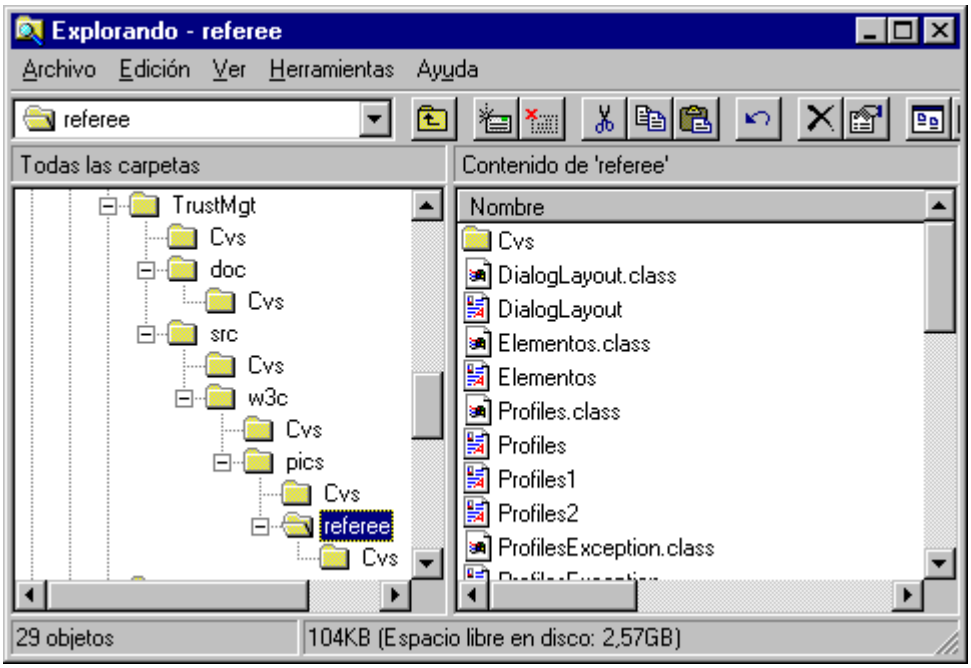

Figura 9. Estructura de Directorios luego de la descompresión

De aplicación prototipo

- Instalar el PLUG-IN de Java jdk-1_2_2_006-win.exe que se encuentra en el directorio recursos (del CD de instalación).

- Configurar los archivos SITCON y SITRES que corresponden a los archivos de control de seguridad de la aplicación prototipo. Estos son archivos ocultos que se encuentran en la ruta: *C:\TrustMgt\src*.

El archivo SITCON. Contiene la lista de direcciones a las que se permite acceder desde la aplicación prototipo. Para definir las direcciones se debe utilizar obligatoriamente la siguiente sintaxis:

(allow URL "direccion1" "direccion2" "direccionN")

Donde*: direccion1, direccion2, .., direccionN*; corresponde a las direcciones a los que se desea que la aplicación tenga permiso de acceder.

El archivo SITRES. Contiene la lista de direcciones desde las que no se permite bajar código y/o acceder desde la aplicación prototipo. Para definir las direcciones se debe utilizar obligatoriamente la siguiente sintaxis:

(block URL "direccion1" "direccion2" "direccionN")

Donde: *direccion1, direccion2, .., direccionN*; corresponde a las direcciones desde las que no se permite bajar código.

- Ejecutar la aplicación policytool.exe que se encuentra en el subdirectorio *bin* del directorio de trabajo donde instaló el Plug-in de Java. Al ejecutarlo este le dirá el directorio desde donde esta tomando el archivo .java.policy. Copie esta ruta, salga del programa y copie el archivo de seguridad .java.policy (que esta en C:\TrustMgt\src) en la ruta que anotó anteriormente. Abra el archivo y coloque /* antes de la siguiente sección de código:

```
grant codeBase "http://RUTASERVIDOR/TrustMgt/src-" {
  permission java.io.FilePermission "http://RUTASERVIDOR/TrustMgt/src/*", "read";
  permission java.net.SocketPermission "SERVIDOR", "connect";
};
```

y */ después de ella. Con esto estamos comentando la información referente a la instalación en un Servidor WEB. Reinicié el equipo para que se tomen todos los cambios.

5.2.2 INSTALACION EN UN SEVIDOR WEB

El proceso de instalación en un servidor Web es muy similar al procedimiento de instalación local, la gran diferencia radica en la forma de configurar el archivo de políticas de acceso para el applet de REFEREE y en el lugar donde se instalará el software de este.

Cuando se ejecute el archivo INS-REF.exe se le debe indicar la ruta del servidor donde se desea instalar, de esta forma se obtiene la misma estructura de directorios mostrada en la figura 9.

El archivo ***.java.policy*** se debe configurar para remplazar la palabra RUTASERVIDOR (por ejemplo: guayacan.uninorte.edu.co/afernand/) por la ruta de directorios bajo la cual quedo instalado el directorio ***TrustMgt*** y la palabra SERVIDOR debe ser remplazada por la dirección del servidor Web (por ejemplo: guayacan.uninorte.edu.co). De otro lado se debe Colocar /* antes del siguiente bloque de código:

```
grant codeBase "file:////c:/TrustMgt/src-" {
  permission java.util.PropertyPermission "user.home", "read";
  permission java.util.PropertyPermission "java.home", "read";
  permission java.io.FilePermission "C:\\TrustMgt\\src\\*", "read";
  permission java.io.FilePermission "C:\\TrustMgt\\src\\*", "write";
};
```

y */ después de este., con lo que se logra comentar la información referente a la instalación local de REFEREE.

5.3 APLICACIÓN PROTOTIPO

La aplicación desarrollada fue implementada en Java y funciona como un filtro que restringe o no el acceso a direcciones URL según los archivos de configuración definidos, utilizando a REFEREE como la herramienta para la toma de decisiones de confianza en los sitios definidos por el usuario con base en los archivos de configuración.

5.3.1 Funcionamiento. La aplicación desarrollada tiene como fin principal, probar el funcionamiento del *software* REFEREE como herramienta para la toma de decisiones de confianza.

En la Figura 10 se pueden apreciar los componentes de la aplicación prototipo desarrollada y a continuación se explican cada uno de ellos:

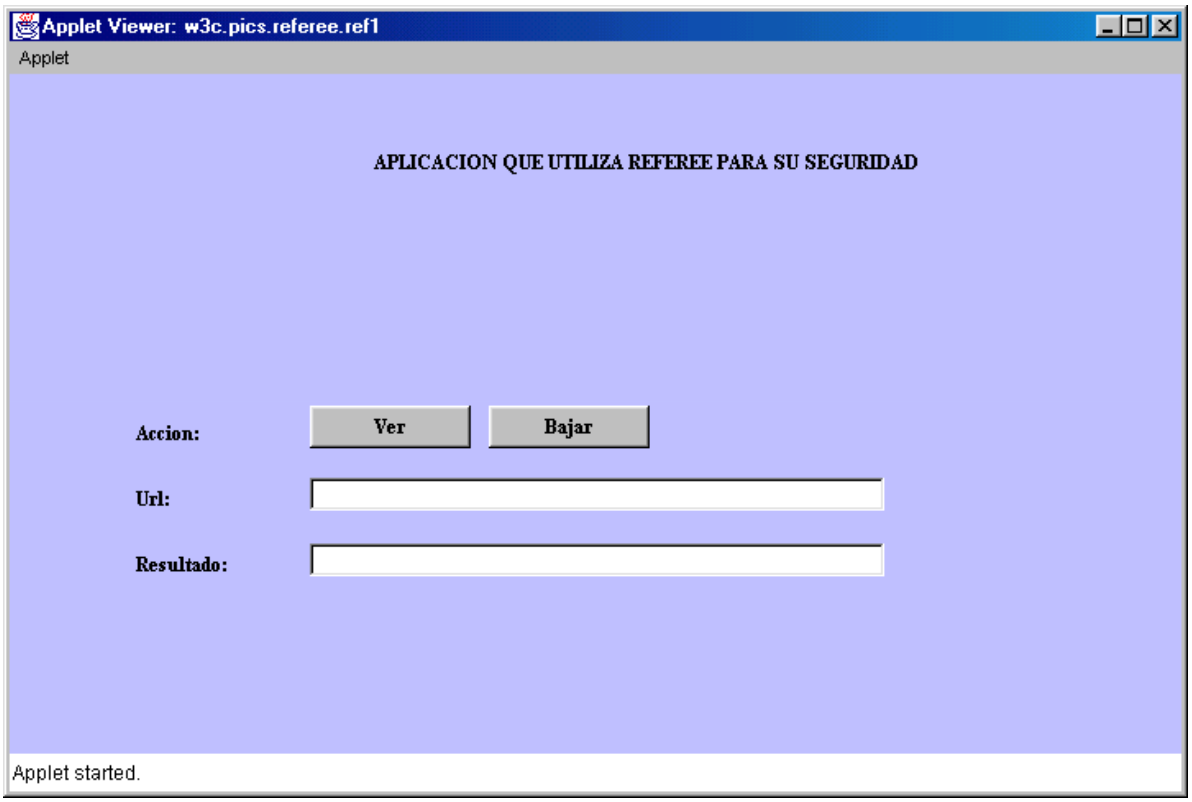

Figura 10. Aplicación prototipo desarrollada.

ACCION: La acción corresponde a las operaciones que se pueden realizar en la aplicación prototipo y están determinadas por los botones VER y BAJAR.

VER: Este botón permite verificar mediante el llamado a REFEREE y el *trivalue* retornado por este, que la dirección definida en el campo con label URL puede ser accedida desde la aplicación prototipo. Para realizar la comprobación de si se tiene o no permiso para acceder a la página indicada en el campo URL, la aplicación desarrollada se basa en el archivo de configuración SITCON; que es enviado a REFEREE junto con el URL determinado para comprobar si se niega o se permite el acceso a la página indicada.

BAJAR: Este botón permite verificar mediante el llamado a REFEREE y el *trivalue* retornado por este que la dirección definida en el campo con *label* URL puede ser accedida desde la aplicación prototipo para bajar código. Para realizar la comprobación de si se puede o no bajar código de la dirección indicada en el campo URL, la aplicación desarrollada se basa en el archivo de configuración SITRES; que es enviado a REFEREE junto con el URL determinado para comprobar si se puede o no bajar código de la página indicada.

URL: Corresponde a la dirección para la que se desea probar si la aplicación permite ejecutar las acciones definidas previamente. El URL debe ser ingresado completo iniciando con *http://*

RESULTADO: Este campo corresponde a la interpretación del *trivalue* de respuesta enviado por REFEREE. Luego del resultado la aplicación determina si accede o no a la página definida en el URL. Y según el resultado verdadero o falso del *trivalue* retornado por el *software* REFEREE, invoca o no al browser con la dirección determinada y ya validada.

5.3.2 Políticas utilizadas. Las políticas utilizadas en la aplicación prototipo son ***Block y Allow*** . Las cuales se basan en las reglas :

- ***FailURL***: Esta cláusula es un método para expresar una lista de prefijos de URL que van a ser bloqueados de manera explícita.

- ***PassURL***: Tiene la misma sintaxis del FailURL pero su sentido es completamente contrario. Permite visitar aquellos lugares con prefijo definido en la lista de direcciones.

Para modificar las políticas existentes solo basta con incluir o excluir direcciones de los archivos de configuración SITRES y SITCON.

Si se desea agregar una política se debe ir al código fuente y crear la función necesaria que la implemente basándose en las reglas existentes (explicadas en el capítulo 4: Políticas de seguridad)

5.3.3 Importancia de la aplicación Prototipo. Esta aplicación prototipo puede ser utilizada por otras aplicaciones similares, pues REFEREE funciona como un FILTRO y los desarrolladores de las nuevas aplicaciones solo tendrían que fijarse en la forma como es llamado y configurado el REFEREE desde el APPLET desarrollado.

De otro lado con la aplicación desarrollada estamos validando la funcionalidad de REFEREE y comprobando la relativa facilidad de definir e implementar las políticas de seguridad en él, luego de conocer la estructura y el funcionamiento del *software*.

5.4 PROCEDIMIENTO DE EJECUCION DE APLICACIÓN PROTOTIPO

La aplicación desarrollada se puede ejecutar de dos formas:

- Ejecución desde DOS:

  Para ejecutar la aplicación prototipo desde DOS se debe previamente realizar la configuración del archivo REFEREE.bat (su contenido se muestra en la Figura 11) que consiste en modificar el path para que apunte al directorio donde se encuentra el applettviewer.exe

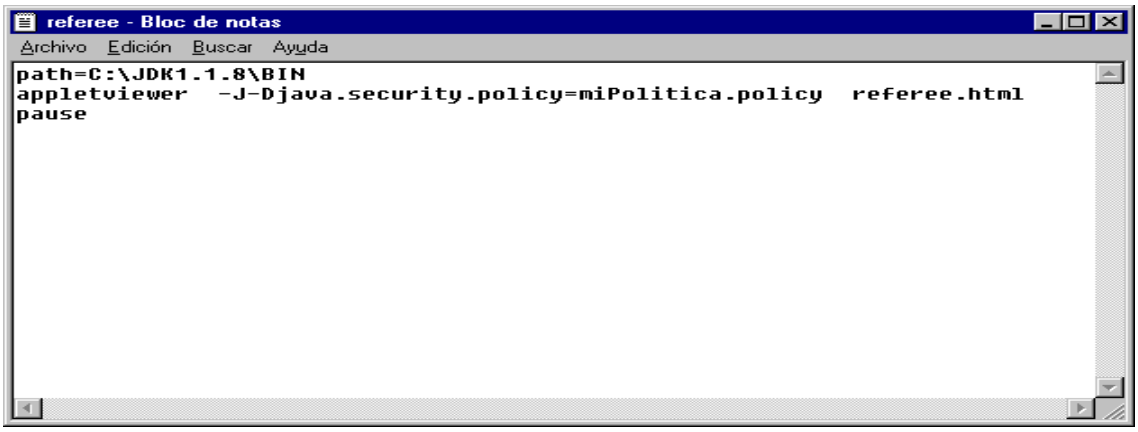

Figura 11. Archivo REFEREE.bat

La ejecución del REFEREE.bat nos lleva a correr la aplicación prototipo sin brindar la posibilidad de navegar a los sitios a los que no se tiene acceso restringido, pues esto es una limitante del *aplettviewer*.

- Ejecución desde el *browser (Netscape)*.

Para poder ejecutar la aplicación desde el *browser* se debe haber realizado el último paso de la instalación que es la de copiar el archivo .java.policy en el directorio indicado por el policytool.exe si esto no hace la aplicación no funcionará. Para ejecutar la aplicación desde el *browser* se corre el programa REFEREE.html. Este modo de ejecución permite ejecutar la aplicación con toda su funcionalidad, por lo que se tendría la posibilidad de navegar a aquellas direcciones sobre las cuales no se tengan ningún tipo de restricciones.

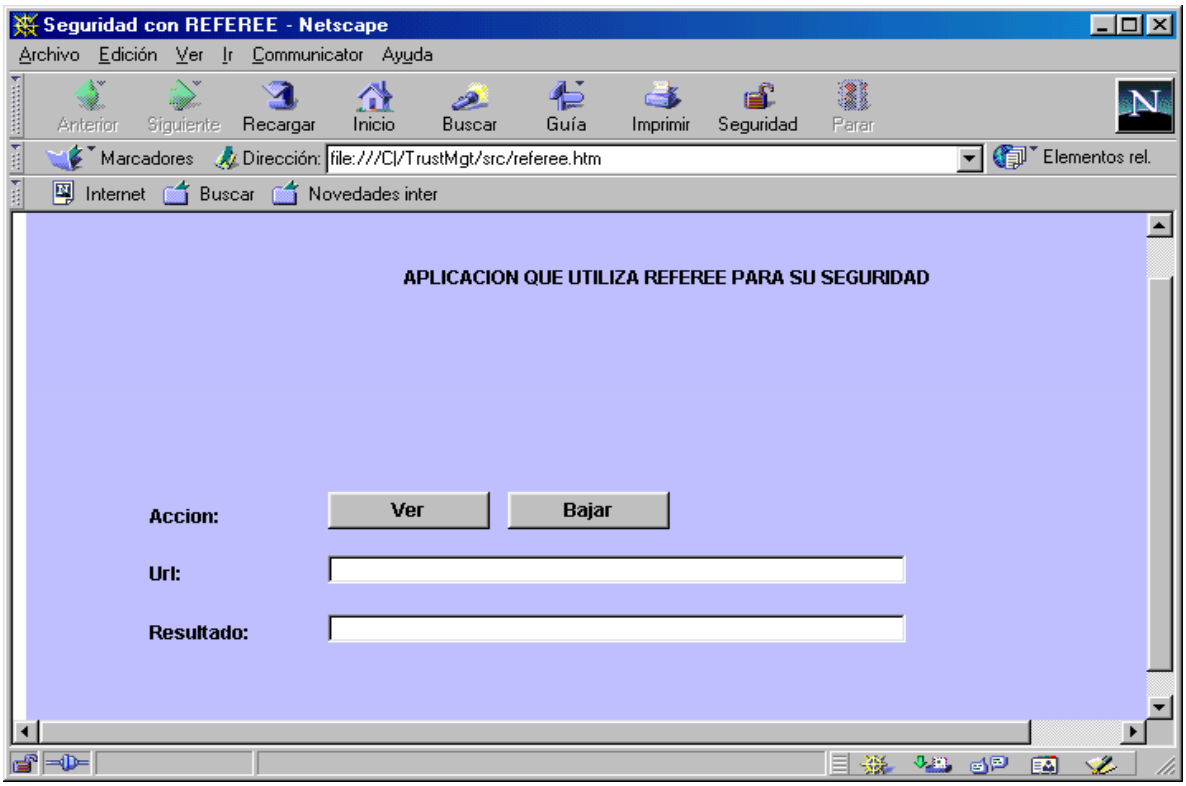

Figura 12. Aplicación REFEREE.html ejecutada desde Netscape

5.5 PROBLEMAS PRESENTADOS

El desarrollo de esta aplicación prototipo fue bastante costoso debido a que el software REFEREE por si mismo no funcionaba correctamente y se hizo necesario corregir ciertas partes del código para que cumpliera con el objetivo para el cual fue diseñado.

Las modificaciones realizadas para lograr un correcto funcionamiento de REFEREE fueron las siguientes:

- Se hizo necesario el desarrollo de una aplicación java tipo applet (ref1.java) para que la pagina html pudiera ejecutar REFEREE debido a que un a pagina html no puede ejecutar directamente un programa java normal. El applet ejecuta la rutina principal de REFEREE (startup, que antes se encontraba en el archivo Profiles.java).

- Se cambio el la rutina principal de REFEREE (startup, contenida ahora en ref1.java) para que recibiera dinámicamente la dirección web a evaluar.

- Se eliminaron todos los mensajes enviados a pantalla.

- Se modificó la manera como REFEREE retorna el resultado de la evaluación de las políticas. Dado que antes el resultado se escribía a un archivo y ahora el resultado es enviado directamente al applet y este es el que se encarga de mostrar el mensaje en la pantalla html y dependiendo de este resultado controla la acción a seguir.

Pensamos que todos estos problemas se debieron a que REFEREE corresponde a un trabajo de grado y la versión que esta disponible no corresponde al desarrollo final.

De otro lado no esta disponible en ningún lado la forma como utilizar REFEREE desde una aplicación, por lo que fue necesario desarrollar un applet que llamara a REFEREE y utilizara la potencialidad explicada en la teoría, la aplicación desarrollada fue probada en Windows 98 con el *browser Netscape*.

6. CONCLUSIONES

El trabajo de tesis realizado, corresponde al estudio y aplicación de la herramienta REFEREE en la definición y puesta en marcha de políticas de seguridad sobre aplicaciones *Web*., este trabajo está basado en una tesis de posgrado cuya finalidad fué desarrollar todas las bases teóricas del *software* de manejo de confianza REFEREE e implementar una aplicación que corroborara toda su funcionalidad

La tesis identifica el problema del manejo de confianza en el contexto del World Wide Web, mostrando la forma como se diseñan las politicas de seguridad,(Capítulo 4) y validando la efectividad de REFEREE mediante la implementación de una sencilla aplicación (Capitulo 5) que funciona como un filtro de confianza entre sitios restringidos o no.

La experiencia de estudiar, analizar y trabajar con REFEREE fue muy interesante porque nos enseñó la manera como definir e implantar politicas de seguridad y tuvimos la oportunidad de adentrarnos en el fascinante mundo del manejo de confianza, lo que nos ayuda a vislumbra el Internet como una gran comunidad en la que existen personas en las que podemos confiar plenamente y otras a las que debemos controlarles cualquiera de sus movimientos.

7. PROYECCIONES

A manera de futuros desarrollos con REFEREE consideramos que seria interesante y provechoso instalar REFEREE en un servidor WEB y evaluar los resultados desde el punto de vista técnico (funcionalidad, tiempo de respuesta, facilidad de uso, consumo de recursos, etc) y desde el punto de vista de los usuarios. Por ejemplo en un ambiente universitario prohibir el acceso a paginas inmorales, restricciones de acceso, etc. Con esto se puede evaluar los resultados de la aplicación y tener un efectivo resultado del *performance* de REFEREE en un ambiente real. Consideramos que esta proyección resultaría suficiente para una tesis de pregrado pues es una realización muy interesante y conveniente para el ambiente que se determine.

BIBLIOGRAFÍA

GARFINKEL, Simpson y SPARFORD, Gene. Seguridad y Comercio en el Web. Mexico. Mac Graw Hill, 1999.

KARANJIT, Siyan y CHRIS, Hare. Internet y seguridad en redes. México. Prentice-Hall, 1995.

YANG-HUA, Chu. Trust Management for the World Wide Web. 1997

URLs.

http://www.research.digital.com/SRC/personal/Martin_Abadi/Papers/tunnel/206.html

http://www.w3.org/PICS/TrustMgt/demo/Overview.html

http://www.w3.org/PICS/TrustMgt/presentation/96-12-4-presentation/index.htm

http://www.w3.org/pub/WWW/Security/DSig/.

ANEXOS

CONTENIDO

LISTA DE FIGURAS