PROPUESTA DE UNA ARQUITECTURA DE SEGURIDAD BASADA EN AUTENTICACIÓN Y AUTORIZACIÓN PARA SISTEMAS DISTRIBUIDOS TIPO GRID

ING. ERICK RAMÓN MENESES CUADROS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
MAESTRÍA EN INGENIERÍA ÁREA EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN
BUCARAMANGA.
2010.

PROPUESTA DE UNA ARQUITECTURA DE SEGURIDAD BASADA EN AUTENTICACIÓN Y AUTORIZACIÓN PARA SISTEMAS DISTRIBUIDOS TIPO GRID

PROYECTO DE INVESTIGACIÓN PARA OPTAR AL TÍTULO DE:
MAGISTER EN INGENIERÍA, ÁREA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

AUTOR
ING. ERICK RAMÓN MENESES CUADROS

DIRECTOR
Ph.D. ARTURO PLATA GOMEZ

CODIRECTOR
Ph.D. CARLOS BARRIOS HERNANDEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
MAESTRÍA EN INGENIERÍA ÁREA EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN
BUCARAMANGA.
2010.

DEDICATORIA

A Dios, por darme la oportunidad de vivir y hacerme dueño de un intervalo espacio – tiempo.

Y a mi familia por ser el soporte y el impulso en todos mis proyectos.

Erick Meneses

AGRADECIMIENTOS

A Dios porque bajo ese infinito orden propicia la vida, vida en la que tenemos libre albedrio y la posibilidad de llegar a ser todo lo que queramos.

A mi familia, simplemente por ser ellos, por **darme á**nimo y ser mi apoyo en los momentos difíciles.

A Carlos Jaime Barrios por ser amigo y guía en todo el andar por los sistemas distribuidos.

Al profesor Arturo Plata por brindarme su confianza al ser mi director y orientarme durante el pregrado y el postgrado.

Al profesor Yves Denneulin por su apoyo y orientación en la incursión al área de seguridad en sistemas distribuidos.

Al profesor Henry Lamos por sus valiosos consejos a nivel profesional y personal.

Al Centro Halley de Astronomía, porque allí empecé a deslumbrarme con el universo y conocí lo bonito que es la investigación.

A Cristian, Mario, Luis, Iván, Julián, Álvaro y Vanesa por darme la confianza para orientarlos en sus trabajos de grado, una de las mejores experiencias de mi maestría.

A mis amigos y a todas aquellas personas que de manera indirecta o directa hicieron posible la realización de este trabajo.

TABLA DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

LISTA DE ANEXOS

Pág.

RESUMEN

TITULO: PROPUESTA DE UNA ARQUITECTURA DE SEGURIDAD BASADA EN AUTENTICACIÓN Y AUTORIZACIÓN PARA SISTEMAS DISTRIBUIDOS TIPO GRID[*]

AUTOR: Erick Ramón Meneses Cuadros[**].

PALABRAS CLAVES: Computación Grid, Seguridad Informática, Autenticación, Autorización, Análisis de Riesgos.

DESCRIPCIÓN

Una grid computacional carga problemas de seguridad debido a su enfoque basado en compartir recursos de manera dinámica entre diversas organizaciones, dichos problemas no pueden ser completamente tratados por las tecnologías convencionales pues establecen nuevos conceptos y esto hace que se deban mejorar o generar nuevas propuestas en este sentido.

A partir de esto y con aras a contribuir a la iniciativa de grid académica nacional - Grid Colombia, este proyecto de investigación busca hacer un aporte mediante la propuesta de una arquitectura de seguridad basada en los procesos de autenticación y autorización por ser considerados básicos en la interacción entre usuarios y recursos.

Tal propuesta es desarrollada a partir de un estudio de requerimientos generales y de seguridad con los que debería contar cualquier sistema de autenticación/autorización para grid. Después de tener claro estos objetivos se realiza una evaluación del cumplimiento que da a los mismos el middleware globus (usado en grid Colombia) y sobre el se realiza un análisis de riesgos (NIST) a fin de determinar los problemas de seguridad inherentes al sistema. Finalmente, se genera una propuesta que busca cumplir con los objetivos planteados y evitar los problemas de seguridad encontrados generando así una arquitectura de seguridad respaldada teórica y tecnológicamente teniendo en cuenta siempre el contexto de desarrollo para favorecer la usabilidad, flexibilidad y disponibilidad sin descuidar la seguridad y rendimiento del sistema.

[*] Proyecto de Investigación.

[**] Facultad de Ingenierías Físico-Mecánicas, Maestría en Ingeniería – Área Informática y Ciencias de la Computación. Director: Ph.D. Arturo Plata Gómez. Codirector: Ph.D. Carlos Barrios Hernández.

SUMMARY

TITLE: PROPOSAL OF A SECURITY ARCHITECTURE BASED IN AUTHENTICATION AND AUTHORIZATION FOR DISTRIBUTED TYPE GRID SYSTEMS[*].

AUTHOR: Erick Ramón Meneses Cuadros[**].

KEY WORDS: Grid Computing, Information Security, Authentication, Authorization, Risk Analysis.

DESCRIPTION

A computational grid load security issues due to their approach based dynamically share resources between different organizations, these problems can not be fully addressed by conventional technologies because they establish new concepts and this makes it must improve or generate new proposals in this sense.

From this and with the interest to contribute to the national academic initiative - Grid Colombia, this research project seeks to make a contribution by proposing security architecture based on authentication and authorization processes to be considered basic in interaction between users and resources.

Such a proposal is developed from a study of the general and security requirements with which any authentication/authorization grid system should have. After these goals are clear, an assessment of compliance that gives them the globus middleware (used in grid Colombia) and an analysis of risks (NIST) to determine the security problems inherent in the system is made. Finally, is generated a proposal that seeks to meet the objectives and avoid security problems found, generating in this way a security architecture supported theoretical and technologically, taking into account the developmental context to promote usability, flexibility and availability without neglecting security and performance of the system.

[*] Research Project.

[**] Faculty of Physical-Mechanical Engineerings, Master in Engineering – Informatics and Computer Science Area. Director: Ph.D. Arturo Plata Gómez. Codirector: Ph.D. Carlos Barrios Hernández.

1 INTRODUCCIÓN

1.1 Presentación

De acuerdo a la ley de Moore, Kryder[1] y Butter[2] la capacidad de procesamiento, almacenamiento y ancho de banda se duplica cada 18, 12 y 9 meses respectivamente. No obstante, existen problemas computacionales cuya demanda en recursos es imposible cubrir con las arquitecturas de maquina convencionales.

En respuesta a esto y aprovechando el rápido desarrollo de las redes de comunicaciones aparece la computación grid, una infraestructura distribuida que propone un modelo de trabajo para compartir recursos en forma coordinada y resolver problemas de forma dinámica en organizaciones virtuales (multi-institucionales geográficamente distribuidas) dependiendo de su disponibilidad, capacidad, desempeño, costo y calidad de servicio requerida por sus usuarios.

Para lograr esta tarea, se han desarrollado tecnologías (protocolos, herramientas, aplicaciones) y metodologías (técnicas, servicios) que intentan respaldar las organizaciones virtuales, pero mientras la escalabilidad, rendimiento y heterogeneidad son los objetivos para cualquier sistema distribuido, las características de una grid computacional cargan problemas de seguridad debido a su enfoque basado en compartir recursos de manera dinámica entre diversas organizaciones [4]. Dichos problemas no pueden ser completamente tratados por las tecnologías convencionales pues establecen nuevos conceptos y esto hace que se deban mejorar o generar nuevas propuestas.

Debido a lo anterior y observando que la grid se basa en la interacción entre usuarios y recursos computacionales, ambos de carácter dinámico. Nace la necesidad de proponer y diseñar un esquema de seguridad basado en los procesos de autenticación y autorización, de modo que permita certificar la identidad de los usuarios y a su vez dar acceso efectivo a los recursos que les son permitidos, manteniendo siempre la seguridad y flexibilidad del sistema.

A continuación se explica de manera detallada que motiva la realización de este trabajo, la metodología usada para su desarrollo, los aportes generados en el proceso y una explicación general del contenido de este documento.

1.2 Motivación de la Investigación.

Actualmente, existe un especial interés por la temática en el país y es por eso que aparecen iniciativas en este sentido, la más importante de ellas, el proyecto Grid

[1] R. Schaller, "Moore's Law: past, present, and future", IEEE Spectrum, vol. 34, 1997.
[2] W. Webb, "Laying Down the Law", IEEE Communications Engineer, 2006.

Colombia [31], que busca unificar instituciones académicas y de investigación por medio de la red de alta velocidad RENATA[3], para de este modo compartir no solo recursos sino conocimiento y experiencias, y lograr de manera conjunta un avance significativo a nivel científico en el país.

El escenario a considerar entonces corresponde a un sistema distribuido altamente dinámico y heterogéneo, donde la seguridad es un aspecto crítico para un entorno de estas características ya que es necesario proporcionar autenticación a los usuarios y un control de acceso a los recursos del entorno. Lo anterior es la principal motivación del presente trabajo que busca hacer un aporte en esta línea, y tiene como objetivo proponer una arquitectura sencilla y flexible que mantenga la seguridad de la información sin impactar de manera negativa en el rendimiento o administración de la infraestructura.

1.3 Objetivos del Proyecto

- Identificar las características, servicios, procesos, componentes e interacción de los mismos dentro de una grid computacional, como paso inicial para comprender el objeto de estudio.

- Hacer un análisis de los requerimientos en autenticación y autorización de una arquitectura de seguridad distribuida tipo grid para el ambiente académico nacional y una revisión de las cualidades y debilidades que ofrecen las arquitecturas existentes, específicamente la Grid Security Architecture propuesta por el Global Grid Forum.

- Identificar las amenazas y vulnerabilidades inherentes a la naturaleza dinámica y colaborativa de una arquitectura grid en los procesos de autenticación y autorización, y a partir de ellas determinar los riesgos de seguridad, ponderarlos y proponer controles a fin de mitigar o eliminar cada uno de ellos.

- Proponer una arquitectura de seguridad para grid basada los procesos de autenticación y autorización, tomando en cuenta aspectos como usabilidad, rendimiento, interoperabilidad y el entorno para el cual se genera la solución.

- Realizar un prototipo de la arquitectura propuesta a fin de probar su funcionalidad e interacción entre sus componentes.

1.4 Metodología de Investigación

Este trabajo se desarrolló en base a los conceptos de diseño de investigación cualitativa del tipo investigación – acción [5], la cual establece las etapas del proyecto, planteando la realimentación correspondiente de manera que se pueda reformular y enriquecer lo obtenido en las etapas anteriores. A su vez presenta al investigador como instrumento de recolección de información, información que analiza y trata de forma adecuada teniendo en cuenta el fenómeno de interés.

[3] Red Nacional Académica de Tecnología Avanzada – http://www.renata.edu.co

En este sentido el proyecto abarca las siguientes etapas:

- Recopilación y estudio de información en las áreas de sistemas distribuidos, seguridad informática, y la relación entre ellas.
- Análisis de las vulnerabilidades en las arquitecturas de seguridad existentes y de las características que debería tener una grid computacional en el entorno académico nacional.
- Desarrollo de la propuesta de una arquitectura de seguridad basada en autenticación y autorización.
- Implementación y realización de pruebas de funcionalidad sobre un prototipo de acuerdo a la arquitectura de seguridad propuesta.
- Conclusiones y propuestas para la continuidad del trabajo, elaboración del informe final y divulgación de resultados.

1.5 Aportes del Proyecto

El trabajo presentado en este documento presenta dos tipos de aporte:

El primero es el producto directo del proyecto, el diseño de una arquitectura de seguridad sencilla, flexible, eficaz y eficiente como resultado de un estudio del estado del arte, un análisis de las vulnerabilidades y riesgos existentes en las soluciones actuales, y propuesta de mejora pensada para un entorno académico de grid nacional.

El segundo corresponde a todas aquellas actividades realizadas de manera paralela y que fueron fruto indirecto del desarrollo de este trabajo, estas son:

- Instrucción de la materia procesamiento paralelo durante el año 2008.
- Coordinación de 4 proyectos de pregrado en el área de sistemas distribuidos.
- Presentación de 4 ponencias internacionales y 1 nacional.
- Realización de 2 pasantías de investigación (España – Francia).

1.6 Organización del Documento

El documento está organizado como se describe a continuación:
El capitulo 2 plantea el problema de investigación detallando la arquitectura de seguridad propuesta por el Global Grid Fórum para a partir de esta mostrar los retos y requerimientos para mejorar su seguridad. El capitulo 3 presenta el marco teórico que describe las tecnologías utilizadas para la solución propuesta con el fin de brindar al lector los conocimientos básicos para el entendimiento de la misma. Después, se hace un recorrido por las tecnologías a nivel de autenticación y autorización a fin de identificar las necesidades de seguridad no cubiertas por las soluciones existentes. Posteriormente se desarrolla un análisis de seguridad para determinar los riesgos y su nivel de acuerdo a la probabilidad e impacto asociada a cada uno, y a partir de ellos proponer controles que los neutralicen o mitiguen. En el capítulo 6 y 7 se muestra el diseño y la implementación de un prototipo de la arquitectura de seguridad propuesta, incluyendo las decisiones de implementación y las ventajas funcionales. Finalmente se presentan las conclusiones y las recomendaciones para dar continuidad al proyecto con el ánimo de mejorarlo.

2 PLANTEAMIENTO DEL PROBLEMA

La seguridad se ha vuelto un aspecto vital en cuando sistemas distribuidos se refiere, y la computación grid no es la excepción, pues asegurar la interacción entre comunidades virtuales y recursos geográficamente distribuidos lo vuelve más complicado aun. En esta sección se pretende hacer una revisión del ambiente del problema y sus delimitaciones, así como los requerimientos generales y en materia de seguridad que la iniciativa de Grid en Colombia implica.

2.1 Contexto del Problema

Actualmente Colombia cuenta con un conjunto instituciones universitarias y centros de investigación, donde cada uno de ellos tiene sus propios recursos, servicios e incluso sus propias metodologías y formas para desarrollar los diferentes procesos que les atañen. Construir algo a partir de tanta heterogeneidad plantea un reto a nivel de infraestructura para comunicar los recursos y a nivel de aplicación para coordinarlos, sin embargo, la primera parte está cumplida gracias a la aparición de una red académica nacional de alta velocidad y para la segunda se hace necesario la utilización de un middleware, una capa de servicios capaz de integrar y ocultar la complejidad subyacente otorgando a los usuarios un acceso fácil y seguro a una gran máquina virtual de capacidad superior. Este es el objetivo de la iniciativa nacional Grid Colombia.

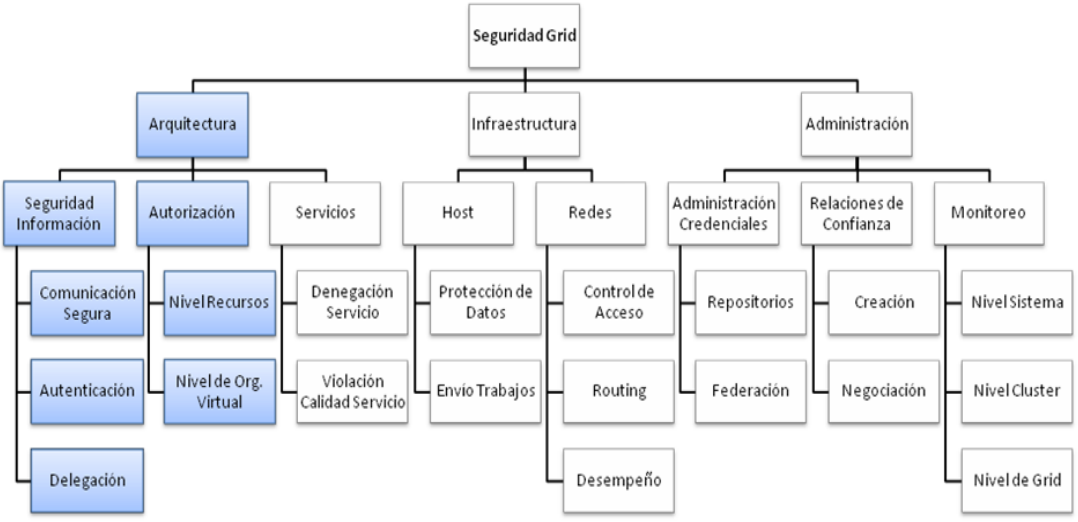

Figura 1. Taxonomía de la Seguridad en la Computación Grid

Con este objetivo claro se debe seleccionar entonces un middleware que soporte el proyecto, sin embargo, la seguridad es un factor crítico y por esta razón se debe evaluar de manera independiente la arquitectura que la soporta.

Este proyecto se enmarca entonces dentro del área de la arquitectura de seguridad en grid computing, considerando como prioritarias las sub áreas de Autorización, Autenticación y seguridad de la información de acuerdo a la taxonomía de seguridad propuesta por Chakrabarti [8]. Bajo este marco se plantean los requisitos que a nivel general y de seguridad debe cumplir la arquitectura intentando con ello esquematizar una solución adecuada para el ambiente de desarrollo de la iniciativa nacional.

2.2 Requerimientos Generales

Los requerimientos generales deseables y que definen el éxito de una arquitectura de seguridad son la usabilidad, escalabilidad e interoperabilidad, descritos a continuación.

- Usabilidad: La facilidad de uso y la no obstrucción al usuario son requisitos indispensables para un entorno distribuido, pues un sistema sencillo de utilizar por parte de los usuarios es más seguro que un sistema complejo donde tienden a usar incorrectamente las herramientas, ya sea por error o por desidia, y ponen en peligro la seguridad del sistema.

- Escalabilidad: es característica más deseada y está presente si no se perciben cambios en el servicio cuando se enfrenta a variaciones fuertes en el número de usuarios que acceden al sistema. La escalabilidad puede medirse en términos de rendimiento y administración de la arquitectura de seguridad.

- Inter-Operabilidad: una nueva arquitectura de seguridad por buena que sea podría no tener la acogida deseada si no toma en cuenta que la mayoría de organizaciones ya posee una infraestructura de seguridad que soporta sus procesos. Por lo tanto, una arquitectura de seguridad debe ser capaz de soportar tecnologías estándar e integrarse con los mecanismos existentes para hacer sencillo el proceso de integración con otras organizaciones.

2.3 Requerimientos de Seguridad

Los requerimientos a nivel de seguridad pueden generalizarse en términos de las características principales de la seguridad de la información (confidencialidad, integridad, disponibilidad) y aquellas que las complementan (autenticación, autorización y auditabilidad).

- Confidencialidad: los datos almacenados o transportados pueden ser leídos por usuarios no autorizados mediante técnicas como sniffing, por eso a fin de proteger la confidencialidad de la información se recomienda el uso de técnicas de cifrado y trasmisión punto a punto.

- Integridad: los datos pueden ser alterados por usuarios no autorizados, para esto es necesario hacer uso de funciones univocas (hash) a fin de verificar que el contenido ha sido o no modificado.

- Disponibilidad: se puede ver comprometida la disponibilidad o nivel del servicio del sistema si un atacante compromete un componente del sistema o realiza ataques de fuerza bruta. En este sentido es recomendable tener sistemas federados o tener una segmentación tal que no se vea afectado todo el sistema.

- Autenticación: todo el sistema puede verse comprometido si un atacante logra una suplantación la identidad y obtener los privilegios de un usuario, el sistema entonces deberá contemplar esquemas de autenticación de múltiple factor y definir los controles de acceso de forma escalable para cada participante en el sistema.

- Autorización: la incorrecta separación de recursos puede ocasionar que de manera casual o intencionada un usuario tenga acceso a recursos que no le son permitidos, para prevenir esto se deben establecer reglas de control de acceso e implementar sistemas de autorización basados en organizaciones o recursos.

- Auditabilidad: en caso de que un ataque sea exitoso el sistema debe ser auditable y proveer bitácoras o logs que permitan esclarecer lo ocurrido.

Otras características que son requerimientos directamente causados por la naturaleza distribuida del sistema son el single sing-on, la delegación y la revocación.

- Single Sign-On: un usuario en la grid puede interactuar con múltiples recursos y es dispendioso que el deba autenticarse con cada uno de ellos, por esto debe existir un mecanismo que mantengan su identidad a través de todo el sistema evitando que se autentique más de una vez.

- Delegación: frecuentemente la ejecución de un trabajo en la grid implica que este se deba tener acceso a múltiples recursos. En este caso es necesario contar un servicio que actúe en nombre del usuario realizando la delegación de la autorización de una entidad a otra.

- Revocación: generalmente se habla de ella cuando una credencial ha expirado porque ha cumplido su tiempo asignado, sin embargo si un usuario comete una acción indebida el sistema debe revocar sus permisos y sacarlo de manera inmediata del sistema.

3 MARCO TEORICO

3.1 Autenticación

3.1.1 Infraestructura de Clave Pública

Certificados Digitales

Los certificados digitales son un documento digital en el cual un tercero en confianza asegura la vinculación entre la entidad certificadora y el usuario. Este sirve principalmente para realizar validaciones de identidad y autenticación.

Los certificados digitales van acompañados por una Llave Pública del usuario al cual pertenece, la cual junto a la Llave Privada del usuario (la cual reside y es custodiada por el usuario) permiten realizar operaciones criptográficas de clave pública y que son comúnmente utilizadas en transacciones electrónicas y/o comercio electrónico. Además de tener asociada una Llave Pública, un certificado digital tiene asociada información del usuario.

PKI

La tecnología PKI permite la administración de llaves y certificados digitales mediante los cuales los usuarios pueden autenticarse entre ellos y contra los servicios manteniendo un ambiente de comunicación confiable [10] [11].

Tal infraestructura se basa en el uso de algoritmos criptográficos de llave pública para brindar facilidades como: cifrar y descifrar mensajes, firmar información digitalmente, garantizar el no repudio y asegurar comunicaciones, otros usos.

Para obtener un certificado digital se deben realizar los siguientes pasos:

a) Una *persona* debe realizar una solicitud formal de certificado digital y para esto debe presentar los documentos requeridos bien sea de asistiendo personalmente o realizando una solicitud vía web, dependiendo de los métodos soportados.
b) Una vez realizada la solicitud, la *Autoridad de Certificación* se encarga de verificar que haya sido bien diligenciada, se cuente con los documentos requeridos y estos sean validos, comprobado esto da su visto bueno
c) Por último, la *Autoridad Certificadora(AC)* siendo un tercero confiable y reconocido en la administración de identidad da su firma al certificado digital que indica que ese certificado es confiable y autentico y entrega al usuario dos cosas: la primera es un certificado digital X.509, el cual contiene datos del usuario, el nombre de la

entidad certificadora, un número de serie, una fecha de expiración y la llave pública del titular, y la segunda es la llave privada asociada al certificado digital.

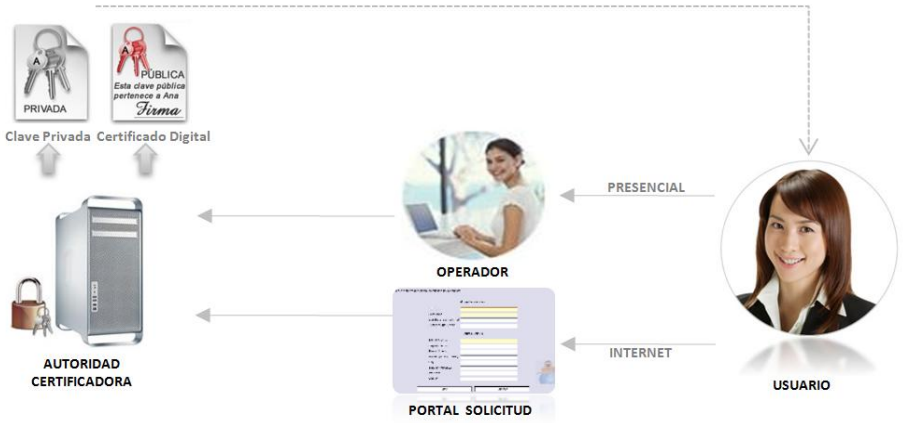

Figura 2. Proceso de Solicitud de un Certificado Digital.

Es de notar que en la mayoría de las veces la autoridad de registro y la autoridad de certificación son fusionadas en un solo servicio y que los documentos a presentar durante la solicitud de un certificado dependen enteramente de la autoridad certificadora.

Con el fin de flexibilizar la expedición de certificados, en PKI existen caminos de certificación:

a) Infraestructura Jerárquica: en esta existen niveles jerárquicos donde en el primero se encuentra la AC principal que está habilitada para certificar ACs en el segundo nivel, que a su vez están facultadas para certificar ACs de tercer nivel y así sucesivamente.
b) Infraestructura en Malla: en esta existen relaciones directas entre ACs para que los usuarios pueden verificar los certificados digitales de otros usuarios y así establecer un vínculo de confianza entre ellos y realizar autenticaciones mutuas.

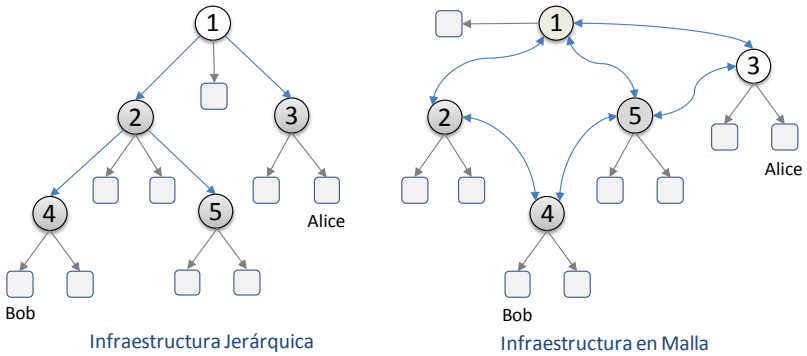

Figura 3. Arquitectura PKI

Finalmente, para que un sistema y/o usuario confíe en un certificado digital, se deben realizar las siguientes validaciones:

a) El certificado digital debe encontrarse en su periodo de validez, el cual es definido por la entidad certificadora que expidió el certificado.
b) La entidad certificadora que expidió el certificado debe encontrarse en la lista de autoridades de certificación de confianza del sistema.
c) El certificado digital no debe estar revocado por la entidad certificadora. Esto se valida realizando una petición a la entidad certificadora.

OSCP

Online Certificate Status Protocol (OCSP) es un método sobre HTTP para determinar el estado de un certificado digital X.509 usando otros medios que no sean el uso de CRL (Listas de Revocación de Certificados) [15].

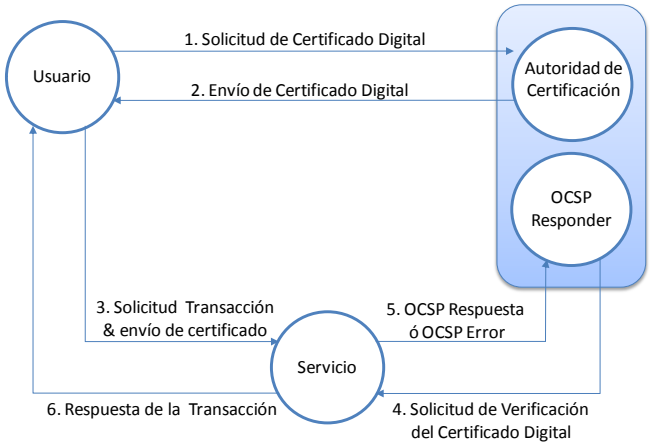

Figura 4. Funcionamiento de un Servidor OCSP.

Una petición de OCSP básicamente está compuesta por la versión del protocolo y los identificadores de los certificados que se quiere que sean validados. (Número de serie, el hash del Distinguished Name (DN) del emisor del certificado y el hash de la clave pública del mismo). En una petición se pueden solicitar la consulta del estado de varios certificados, incluso pertenecientes a diferentes CA. La firma de la petición es opcional y depende de lo que decida la Autoridad de Validación OCSP.

Un "responder" OCSP puede devolver una respuesta firmada, lo cual significaría que el certificado indicado en la petición es "bueno" (good), "revocado" (revoked) o "desconocido" (unknown). También puede devolver un código de error no firmado.

Es preferible la validación de los certificados mediante OCSP sobre el uso de CRL por:

- OCSP elimina la necesidad de que los clientes tengan que obtener y procesar las CRL, ahorrando de este modo tráfico de red y procesado por parte del cliente.
- El contenido de las CRL puede considerarse información sensible, análogamente a la lista de morosos de un banco.

- OCSP soporta el encadenamiento de confianza de las peticiones OCSP entre los "responders". Esto permite que los clientes puedan lanzar una petición a una autoridad de certificación alternativa dentro de la misma PKI.
- Una CRL debe recorrerse secuencialmente para decir si un certificado es válido o no. Un "OCSP responder" en el fondo, usa un motor de base de datos para consultar el estado del certificado, lo que lo hace mucho mas rápido.

3.1.2 LDAP

Descripción

Desarrollado en 1993 en la Universidad de Michigan, LDAP (Protocolo compacto de acceso a directorios) es un protocolo estándar que permite administrar directorios y acceder a bases de información de usuarios y recursos sobre una red mediante el protocolo TCP/IP [36].

Ldap es en sí una base de datos que contiene una jerarquía de directorios y dentro de ellos entradas con información basada en atributos. Dicha base de datos difiere de las bases de datos relacionales al estar optimizada para procesos de lectura permitiendo atender gran cantidad de operaciones de manera rápida, sin embargo es lenta cuando de escritura se trata bien sea debido a una inserción o actualización de datos. Como consecuencia con ldap no se implementan normalmente esquemas complicados como los transaccionales pues las actualizaciones deben ser cambios sencillos de «todo o nada» y no grandes volúmenes de datos.

Las características de un servidor LDAP son:

- Lecturas rápidas: por su naturaleza el proceso de lectura es rápido, por este motivo es recomendable para almacenar información relativamente estática vista en procesos como autenticación, localización de recursos, correo electrónico, etc.
- Estructura jerárquica: por su forma nativa almacena directorios de forma jerárquica donde cada uno contiene objetos que incorporan una colección de atributos.
- Esquema Estándar: implementado a partir del protocolo X500 y especificado por la revisión RFC2830 (LDAP v3), razones por las cuales es ampliamente usado en organizaciones y altamente compatible con otros esquemas de autenticación.
- Entorno distribuido: permite tener múltiples servidores y replicar información de forma amplia, con el fin de aumentar la disponibilidad y la fiabilidad, y a la vez reducir el tiempo de respuesta.

A continuación se presenta una relación comparativa entre una base de datos ldap y una base de datos relacional, una pregunta frecuente a la hora de usarlo.

Característica	LDAP	B.D. Relacional
Lectura / Escritura	Lectura optimizada	Lectura / Escritura Normal
Escalabilidad	Sencilla	Compleja
Distribución Tablas	Inherente	Posible en algunos casos
Replicación	Posible	Posible en algunos casos
Estándar	Si	No

Tabla 1. Comparación entre una B.D. LDAP y una B.D. Relacional.

Funcionamiento

El servicio de directorio LDAP se basa en un modelo cliente-servidor. Uno o más servidores LDAP contienen los datos que conforman el árbol del directorio LDAP o base de datos troncal. El cliente LDAP se conecta con el servidor y le hace una consulta, el servidor contesta con la respuesta correspondiente, o bien con una indicación de dónde puede el cliente hallar más información (normalmente otro servidor). No importa con qué servidor LDAP se conecte el cliente: siempre observará la misma vista del directorio.

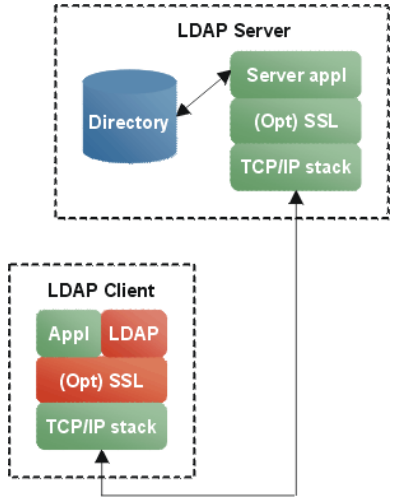

Figura 5. Arquitectura de LDAP.

LDAP le brinda al usuario métodos que le permiten: conectarse, desconectarse, buscar información, comparar información, insertar entradas, cambiar entradas y eliminar entradas del directorio. Asimismo, el protocolo LDAP ofrece mecanismos de cifrado (SSL, etc.) y autenticación para permitir el acceso seguro a la información almacenada en la base de datos.

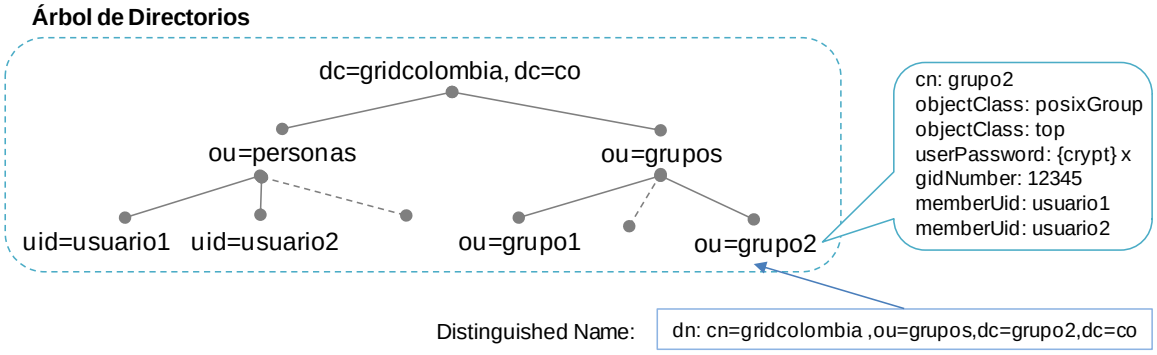

Figura 6. Ejemplo de Árbol de Directorios.

Cada entrada en el directorio LDAP corresponde a un objeto abstracto o real (por ejemplo, una persona, un objeto material, parámetros, etc.), que está conformada por un conjunto de pares clave/valor denominados atributos, algunos son: uid (id de usuario), cn (nombre

común), givenname (nombre de pila), sn (apellido), o (organización), u (unidad organizacional), mail (dirección de correo electrónico), entre otros.

Los servidores se pueden organizar en las siguientes configuraciones:

a) Servicio de Directorio Local: un servidor único es capaz de atender todas las peticiones de los clientes.
b) Servicio de Directorio Replicado: puede establecerse procedimientos de replicación entre servidores maestros y esclavos.
c) Servicio de Directorio Distribuido: la base de datos se divide en sub partes (eventualmente replicado) que son accesibles a través de un conjunto de referencias entre los servidores.

3.2 Autorización

3.2.1 XACML

Descripción

OASIS Extensible Access Control Markup Language (XACML) es un lenguaje que permite expresar políticas de autorización y/o control de acceso en formato XML. XACML nació como necesidad de realizar una estandarización ante la gran variedad de lenguajes propietarios que impedían la transferencia de políticas entre cada una de las aplicaciones. Algunas cualidades de XACML son permitir el uso de atributos arbitrarios en las políticas, control de acceso basado en roles, etiquetas de seguridad, políticas de seguridad basados en tiempos y fechas, políticas de denegación y políticas dinámicas sin la necesidad de modificar las aplicaciones que usan directamente a XAMCL [28].

Al momento de proponer y diseñar XACML fueron tenidos en cuenta ciertos requerimientos para un lenguaje de expresión de políticas:

a) Proveer un método para combinar reglas y políticas (locales y remotas) en un único conjunto de políticas que apliquen a una determinada decisión.
b) Facilitar un procedimiento para aplicar el control de acceso en términos de del sujeto, el recurso, los atributos de ambos y el ambiente involucrado.
c) Ofrecer la capacidad de utilizar operaciones matemáticas y lógicas sobre atributos de los recursos, sujetos y ambiente en el cual se toma una decisión.
d) Proveer un mecanismo para el manejo de múltiples sujetos actuando en diferentes contextos.

Arquitectura y Funcionamiento

El diseño de XACML está ideado para hacerlo completamente independiente de la aplicación que lo utilice. Para lograr esto XACML realiza el intercambio de mensajes a través de XML para recibir y responder peticiones de autorización, tal como se observa en la Figura 7. Los mensajes de petición deben contener una serie de atributos que pueden ser expresados en formato XPath. Tales atributos hacen referencia a una acción, un sujeto (el que realiza una acción) y un recurso (sobre el cual se realiza la acción); lo anterior implica que tanto como las peticiones como las políticas deben estar definidas en términos de triadas sujeto-recurso-acción.

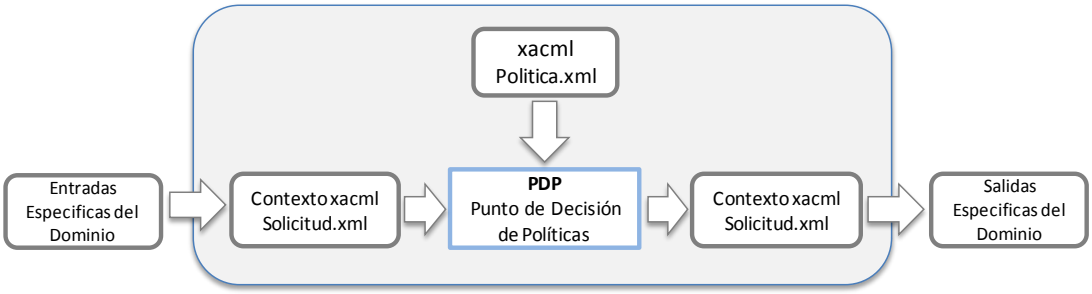

Figura 7. Contexto de XACML.

Por otra parte, al interior de XACML se ha considerado una serie de componentes que permiten ofrecer las funcionalidades anteriormente descritas. En la figura 8 se ilustra el intercambio de mensajes entre cada uno de los componentes y los componentes en sí. Estos están compuestos por un Punto de Administración de Políticas (*PAP*) que se encarga de la creación de políticas; un Punto Cumplimiento de Políticas (*PEP*) que se encarga de recibir y responder las peticiones; un Punto de Información de Políticas (*PIP*) que se encarga de dar información sobre los atributos de una política determinada; y un Punto de Decisión de Políticas (*PDP*) que se encarga de evaluar las políticas que aplican a una petición determinada y emite una decisión de autorización.

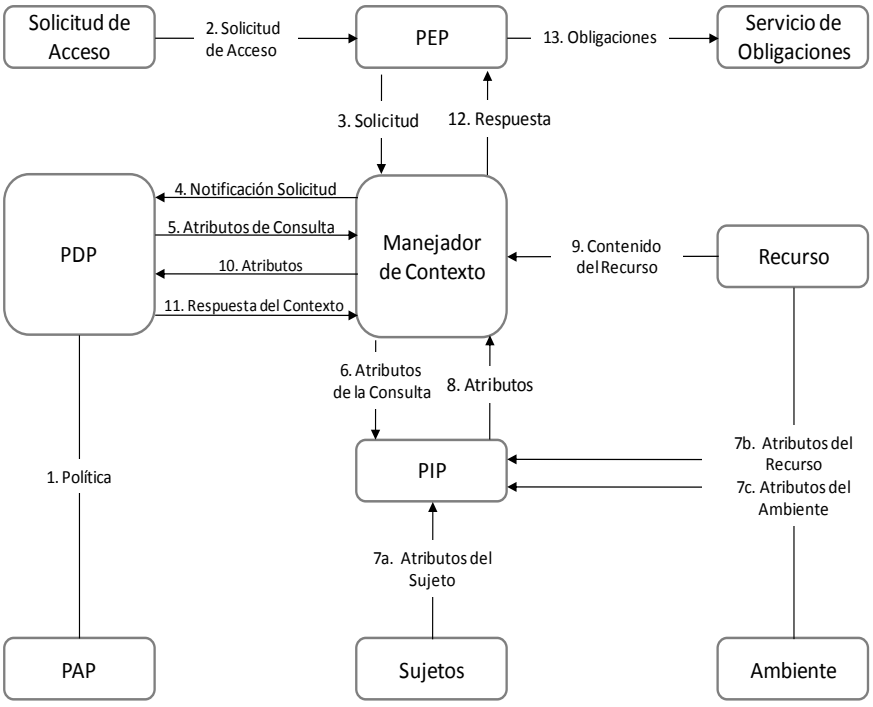

Figura 8. Componentes y Flujo de Mensajes de XACML.

Finalmente, es importante ver los elementos que componen una política XACML y a fin de explorar las propiedades estándar del lenguaje. Los elementos más importantes son mostrados a continuación:

a) Constructores de alto nivel: Policy y PolicySet

La raíz de todas las políticas XACML es un elemento *Policy* o *PolicySet*. Un elemento *PolicySet* es un contenedor que puede contener otros elementos *Policy* o *PolicySet*, así como referencias a políticas que se encuentran en ubicaciones remotas. Un elemento *Policy* representa una única política de control de acceso, expresada a través de un conjunto de reglas *Rules*.
Puesto que un elemento *Policy* o *PolicySet* puede contener múltiples *Rules*, cada una de ellas podrían evaluarse a diferentes decisiones mediante una colección de elementos que permiten reflejar una combinación de algoritmos *Combining Algorithms*.

b) Objetivos y Reglas

Parte de lo que un servicio PDP (Policy Decision Point) de XACML necesita hacer es buscar una política que sea aplicable a una petición dada. Para conseguir esto, XACML proporciona otra característica denominada *Target* que es un conjunto de condiciones simplificadas para el sujeto (elemento *Subject*), recurso (elemento *Resource*) o la acción (elemento *Action*) que deben ser expresadas por un *PolicySet*, *Policy* o *Rule* de forma que puedan ser aplicadas sobre una petición dada.

Es decir, para poder determinar si cierta política o conjunto de políticas, o si cierta regla aplica a la petición XACML recibida se deberá analizar el contenido del elemento *Target* que contiene el sujeto, el recurso y la acción sobre la que aplican. Una vez se encuentra una política que aplica a la petición, se procede a evaluar las reglas *Rule* que contiene. El resultado de combinar los resultados de evaluar cada una de las reglas contenidas en la política será el que refleje si, para esa política, la petición puede ser autorizada o no.

c) Atributos, Valores de Atributos y Funciones

Los atributos son valores con nombre de tipos conocidos que podrían ser cosas como un identificador de un emisor o una fecha y hora de emisión. Específicamente, los atributos son características del sujeto, del recurso, de la acción o del entorno para el cual la petición de acceso ha sido realizada. Un nombre de usuario, el fichero al que se quiere acceder y la hora del día son todos posibles valores de atributos. Cuando se envía una petición XACML desde un PEP a un PDP, ésta se compondrá básicamente de atributos que serán comparados con los valores de los atributos en una política, pudiéndose llevar a cabo la toma de decisión de acceso.

4 ESTADO DEL ARTE

4.1 Arquitectura de Seguridad en Globus

Para el contexto de grid nacional en Colombia se ha adoptado el uso del middleware Globus (a pesar de que algunas instituciones realizan trabajos en gLite de manera interna), por esta razón será su arquitectura de seguridad la que será estudiada en esta sección [32].

Globus implementa el estándar GSI[4] que especifica la comunicación segura, autenticación y autorización en un ambiente de computación en Grid. Los mecanismos de seguridad utilizados para la privacidad y confidencialidad de las comunicaciones son TLS/SSL[5] a nivel de transporte y WS-Security a nivel de mensajes entre servicios.

Por otra parte, la autenticación es realizada a través de certificados digitales permitiendo la delegación de privilegios para implementar un esquema de Single Sign On.

En cuanto a la Autorización, Globus ha considerado un mecanismo sencillo a nivel de organizaciones virtuales (VO) con los cuales se establecen reglas a nivel de grupos utilizando el lenguaje SAML que permite el intercambio de la identidad de los usuarios.

Las desventajas de esta arquitectura de seguridad residen en la autenticación, pues cuenta con un mecanismo centralizado que puede comprometer todo el sistema en caso de un fallo o ser sometido ataques de denegación de servicio, por otra parte el manejo de la autorización es algo pobre ya que se basa en organizaciones virtuales lo que implica una granularidad gruesa en la que no se pueden detallar casos específicos de usuarios y por último las reglas de control de acceso son estáticas sin tener en cuenta el contexto.

A continuación se muestra una tabla con una serie de características y las tecnologías usadas por Globus para cada una de ellas.

Característica	Globus	Característica	Globus
Confidencialidad	SSL/WS-Security	Control de Acceso	Reglas Estáticas / SAML
Integridad	SSL/WS-Security	Single Sign-On	Certificados Digitales
Disponibilidad	Media	Revocación	CRL[6]
Autenticación	Centralizado / CAs	Usabilidad	Media
Autorización	Centralizado / VOMS	Escalabilidad	Media
Auditabilidad	Baja	Interoperabilidad	Media

Figura 9. Ventajas y Desventajas de la Arquitectura de Seguridad en Globus.

[4] Grid Security Infrastructure
[5] Transport Layer Security/Secure Socket Layer
[6] Certificates Revocation List

4.2 Soluciones de Autenticación

La autenticación es un aspecto fundamental de la seguridad de un sistema. Confirmar la identidad de cualquier usuario que intenta iniciar la sesión en un dominio o tener acceso a los recursos de la red es una preocupación en las organizaciones.

A partir del trabajo de Haidar y Abdallah [12] se realiza la comparación de los mecanismos de autenticación usados en tres arquitecturas para organizaciones virtuales, las cuales son descritos a continuación:

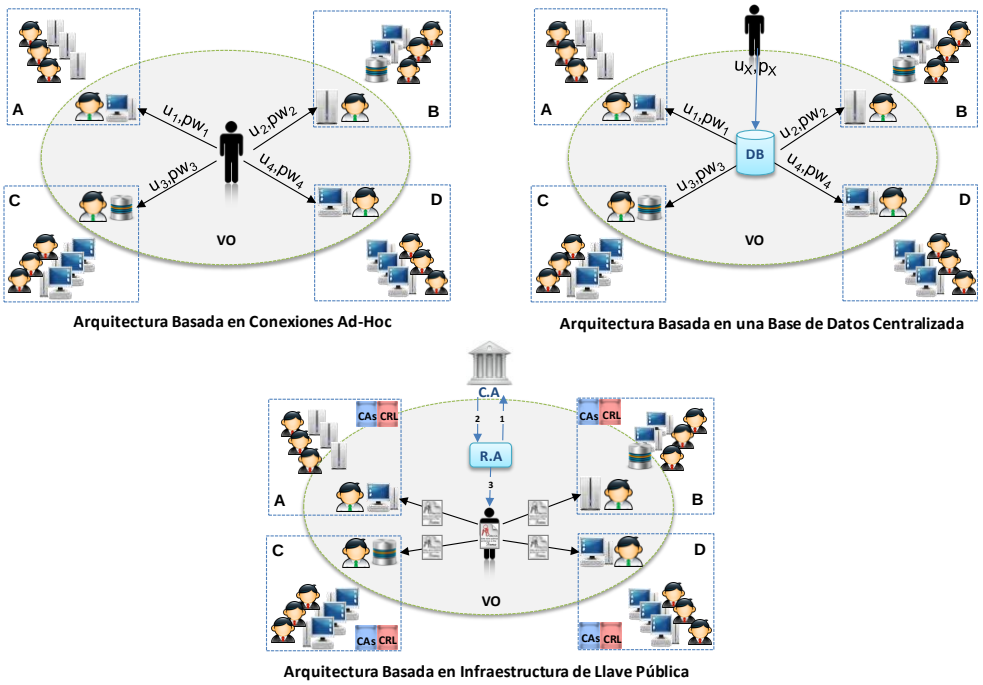

Figura 10. Administración de Identidad en Arquitecturas Distribuidas.

- Arquitectura Basada en Conexiones Ad-Hoc: esta es la organización más primitiva de una VO, es realmente virtual pues no cuenta con administración distribuida, recursos dedicados o confianza en terceros. Cada miembro de la VO cuenta con un par usuario/contraseña por cada uno de los sitios donde desea autenticarse, y cada sitio cuenta con usuarios propios pero atiende usuarios externos extendiendo su base organizacional y realizando un control de acceso discrecional.

- Arquitectura Basada en una Base de Datos Centralizada: en esta VO existe una administración por cada sitio y una base de datos dedicada para almacenar las cuentas y los permisos de los usuarios al accesar a los recursos. Cada miembro de la VO usa un par $usuario_X$/$contraseña_X$ para autenticarse contra la base de datos y una vez hecho el usuario puede solicitar acceso a recursos de un sitio con lo cual la base de datos le otorgara un par $usuario_{X\text{-}sitio}$/$contraseña_{X\text{-}sitio}$ para accesar allí y hacer uso de los recursos.

- Arquitectura Basada en Infraestructura de Llave Pública: esta es la organización más sofisticada de una VO. Cuenta con administración distribuida y confianza en un tercero denominado Autoridad Certificadora quien se encarga de proveer las credenciales (certificado digital) de cada usuario. En este caso la administración de la VO actúa como una Autoridad de Registro verificando la identidad y definiendo el rol del usuario, hecho esto realiza una solicitud a la autoridad de certificación para que emita un certificado digital con los datos asociados que le permitan autenticarse en los diferentes sitios bajo un rol especifico. La VO es responsable por proveer una lista de certificados raíz de las autoridades de certificación confiables y actualizar una lista de certificados revocados.

	Criterio de Evaluación			
Arquitectura	Usabilidad	Administración	Escalabilidad	Single Sign-On
Conexiones Ad-Hoc	El miembro debe recordar el par usuario/contraseña para cada sitio en la VO.	Se deben manejar usuarios de otros sitios agregando y eliminando cuentas	No es escalable pues involucra sobre carga manejo de usuarios de otros sitios.	No cuenta con single sign-on, el miembro-VO debería intervenir dando su usuario/ contraseña.
Base de Datos Centralizada	El usuario debe interactuar con la B.D para solicitar credenciales en cada sitio.	El administrador debe dar a cada usuario credenciales en cada sitio insertándolas en la B.D.	No es escalable pues implica la administración local de usuarios de otras instituciones.	Podría soportar single sign-on mediante el uso de tickets implementando Kerberos.
Infraestructura de Llave Pública	El usuario solo debe recordar la contraseña para instalar su certificado digital.	No se administran muchos usuarios sino certificados raíz de autoridades confianza y lista de certificados revocados.	Esta arquitectura es desacoplada al confiar en un tercero y realmente es escalable.	Soporta single sign-on a través de la implementación del servicio my-proxy de GGF.

Tabla 2. Comparación de Sistemas de Autenticación.

A continuación se muestran las soluciones tecnológicas más relevantes en cuanto a autenticación se refiere ya que cuentan con gran aceptación y son usadas en organizaciones de todo tipo (académico, empresarial y gubernamental).

- Linux: la autenticación local en ambientes Linux es basada en dos componentes: el primero es PAM (Pluggable Authentication Module) el cual permite la integración de tecnologías estándar (crypt, RSA, DCE, LDAP, etc.) para ser usadas por servicios del sistema (ssh, ftp, passwd, etc.) y por otro lado NSS (Name Service Switch) que permite que ya estando autenticado el usuario se busque información respecto a él a través de tablas locales (/etc/passwd, /etc/shadow, /etc/group, etc.).

- NIS (Network Information System): habilita la administración centralizada de información almacenando mapas bajo bases de datos indexadas accesibles por medio de RPC[7]. Usa un modelo maestro/esclavo pero no permite el tratamiento de grandes volúmenes de datos pues cada modificación involucra la transferencia

[7] Remote Procedure Call.

total de la base, además, es bastante complicado organizar la información de manera jerárquica, lo que genera una debilidad en el sistema.

- Kerberos: es un protocolo de autenticación creado por el Instituto Tecnológico de Massachusetts (MIT) que permite a dos computadores en una red insegura demostrar su identidad mutuamente de manera segura. Está basado en un modelo de cliente-servidor, y brinda autenticación mutua: tanto cliente como servidor verifican la identidad uno del otro. Es ampliamente usado en la industria por su característica de emisión de tickets con estampas de tiempo. Kerberos se basa en criptografía de clave simétrica, aunque extensiones para poder utilizar criptografía de clave asimétrica [19] [35].

- LDAP (Protocolo Ligero de Acceso a Directorios): es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.
  Un directorio con conjunto de objetos con atributos organizados en una manera lógica y jerárquica donde almacena información útil para la autenticación (login/password) así como otros datos del usuario (datos del usuario, ubicación de diversos recursos de la red, permisos, certificados, etc.) [18].

4.3 Soluciones de Autorización

Dentro del proceso de autorización es importante observar como hace parte del modelo de control de acceso el cual puede ser mandatorio, discrecional o basado en roles, y ver qué tipo de autorización se va a implementar pues esto tiene implicaciones en factores como la escalabilidad, usabilidad y revocación. A continuación se muestran los dos tipos de modelo de autorización y algunas de sus características.

- Modelo Push: en este mecanismo existe un generador de certificados que chequea la identidad del usuario y le genera un certificado, este es presentado a el controlador de acceso y una vez validado podrá accesar a todos los recursos sin solicitar permiso nuevamente.

- Modelo Pull: en este modelo el usuario provee el mínimo de credenciales como login/password y logra ingresar al sistema, luego de eso cada solicitud de acceso a recursos será consultada contra las políticas de acceso almacenadas en la base de datos.

Un análisis de estos modelos puede realizarse a través de sus características, por ejemplo: en cuanto a escalabilidad un modelo push es mejor debido a que la generación de certificados y el control de acceso están desacoplados y se ejecutan a diferentes tiempos, en cuanto a usabilidad es mejor un modelo pull pues es más sencillo para el usuario quien no debe solicitar credenciales y deja el trabajo al controlador de acceso para permitir el uso de los recursos.

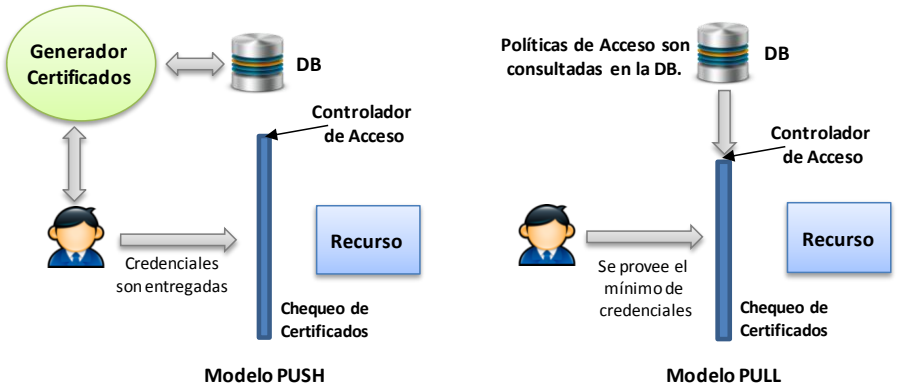

Figura 11. Tipos de Sistemas de Autorización.

Por otra parte al hablar de soluciones de autorización se tiene que hablar de dos grupos: aquellas basadas en organizaciones virtuales que se caracterizan por tener un sistema de autorización centralizada y proveer credenciales para que los usuario acceden a los recursos ó aquellas basadas en recursos que ponen requisitos mínimos para el acceso del usuario al sistema pero aplican las políticas de seguridad a la hora de autorizar el uso de un recurso. Entre las soluciones basadas en organizaciones virtuales se encuentran:

- CAS (Community Authorization Service): ha sido desarrollado como parte del proyecto Globus. En CAS la autorización está dada como: la autoridad otorgada por el proveedor de recursos a la comunidad (C) junto con la capacidad proporcionada por el servidor CAS al usuario (U) y las restricciones dadas por el recurso (R), lo que genera una capacidad efectiva igual a $C \cap U \cap R$ [20].

- VOMS (Virtual Organization Membership Service): ha sido desarrollado por European Data Grid como parte de los proyectos DataGrid and DataTag. En VOMS el usuario debe contar con un certificado digital emitido por tercero confiable (Autoridad Certificadora) y con este en su poder presentarlo para que con base en el grupo o grupos a los que pertenece se le asignen los permisos de acceso a los recursos [21].

- EALS (Enterprice Authorization and Licensing Service): ha sido desarrollado por Infosys Technologies como solución orientada a la empresa. EALS a diferencia de CAS y VOMS cuenta con un sistema de autenticación y autorización desacoplado que le da escalabilidad y permite reemplazar el sistema de autenticación por cualquier otro existente en la empresa, interoperabilidad que aumenta debido al uso de SAML[8] [22].

Y entre las soluciones basadas en recursos se están:

- Akenti: desarrollado por el Laboratorio Nacional Lawrence Berkeley. Aunque nació con orientación a recursos Web, el concepto se amplió posteriormente para incluir los recursos de una grid. El modelo consiste en la distribución de recursos a través

[8] Security Assertion Markup Language

de un punto de aplicación de políticas (PEP) para un conjunto de usuarios que son parte de una organización virtual. La identificación de usuarios se realiza a través de certificados digitales y la declaración de políticas a través de ficheros XML [23].

- PERMIS (Priviledge and Role Management Infrastructure): es un proyecto financiado por la comunidad europea, creado para solucionar tres problemas en diferentes áreas y países. PERMIS utiliza un modelo de acceso basado en roles (RBAC), donde un rol es asignado a cada usuario con sus funciones y las políticas y se le permite el acceso a determinados recursos en base a las políticas definidas [24].

- Grid-Map: Este es el primer sistema de autorización utilizado en Globus. Aunque existen sistemas más sofisticados GridMap sigue siendo uno de los sistemas de autorización más utilizado debido a su simplicidad. Cuenta con limitaciones debido a que usa políticas estáticas, no se comporta bien a gran escala y es necesario la actualización manual del archivo gridmap para conceder el acceso a un nuevo recurso.

A continuación se muestra una tabla comparativa respecto a las características que cada una de ellas tiene:

	Basadas en Organizaciones Virtuales			**Basadas en Recursos**		
Parámetros	CAS	VOMS	EALS	AKENTI	PERMIS	GRID-MAP
Tipo Autorización	Push	Push	Pull	Pull	Push/Pull	Pull
Escalabilidad	Alta	Alta	Media	Media	Alta	Media
Carga Administ.	Baja	Baja	Baja	Baja	Baja	Alta
Autenticación	GSI	GSI	Passwd/Certs.	Certificados	Certificados	GSI
Revocación	No	No	Si/Rápida	Si/Rápida	Si/Media	Si/Lenta
Interoperabilidad	SAML	SAML	SAML/XACML	Compleja	SAML	Mínima
Federación	No	Si	Si	Si	No	No

Tabla 3. Comparación Sistemas de Autorización.

5 ANALISIS DE RIESGOS EN UNA ARQUITECTURA DE SEGURIDAD GRID

Los sistemas tradicionales se encuentran protegidos detrás de firewalls, NATs, VPNs, y un conjunto de restricciones, tal que los atacantes deber realizar una exhaustiva labor de inteligencia para saber que ellos. Sin embargo, los servicios en una grid en cambio son altamente visibles y están agrupados, lo que si bien es atractivo para las organizaciones también es atractivo para un atacante pues lo convierte en un gran blanco en cuestión.

Por eso antes de empezar cualquier proceso que involucre cambios en la tecnología de una organización es recomendable valorar qué consecuencias tiene a nivel de seguridad. En esta sección se estudian las implicaciones que tiene la adopción o migración a la grid, brindando al lector una metodología para valorar y actuar de manera preventiva frente a este proceso.

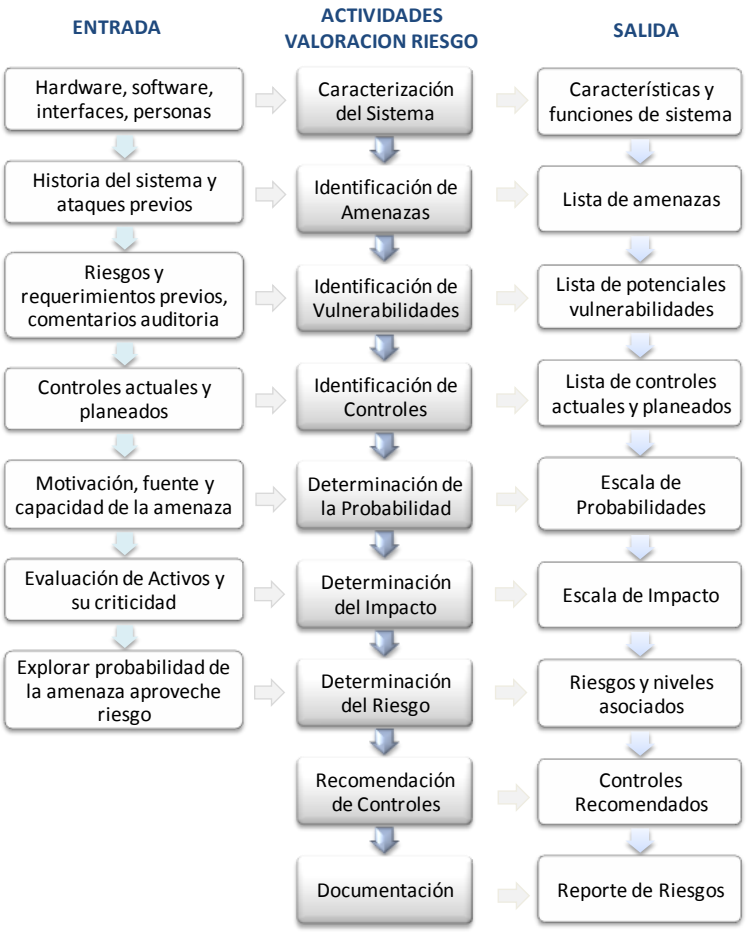

Figura 12. Metodología de Análisis de Riesgos – NIST.

5.1 Análisis de Riesgos.

El análisis de riesgos es el primer paso dentro de cualquier sistema de gestión de seguridad de la información. Para esta etapa existen muchas metodologías, todas con el objetivo de valorar los riesgos asociados al sistema en estudio, pero que varían en la forma de desarrollar tal proceso. Entre las más conocidas están: el análisis de riesgos probabilísticos de NASA[9], el análisis basado en el costo de activos de Magerit[10] y el análisis subjetivo - teórico de NIST, entre otras. Para este caso específico fue seleccionada la metodología NIST pues es cualitativa y con esto permite abordar sistemas generalizados, sin ser necesario contar con datos estadísticos, valores acerca de sus activos y procesos específicos, lo que sí es común en el estudio específico de una organización.

La determinación del riesgo aquí puede ser explicada a través de un ejemplo, como sigue:

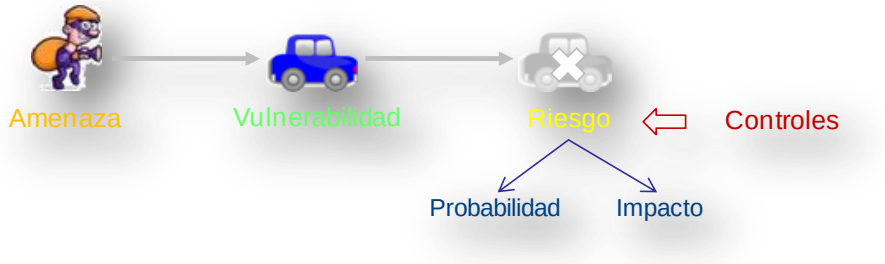

Figura 13. Análisis de un Ataque Informático.

Cualquier delito empieza por que existe una *amenaza*, un atacante (interno o externo a la organización) que cuenta con la motivación y conocimiento necesario para llevarlo a cabo, sin embargo esto no basta y por eso busca en el sistema (automóvil) una *vulnerabilidad* o falla de seguridad (puerta mal cerrada) y se aprovecha de ella materializando el *riesgo* (robo del automóvil).

A fin de evitar esto la metodología parte de las amenazas y vulnerabilidades presentes en el sistema para determinar los posibles riesgos y clasificarlos de acuerdo a la *probabilidad* e *impacto* que causaría la materialización de uno de ellos. Hecho esto se determina cuales riesgos serán tratados y se proponen *controles* para eliminarlos o mitigarlos.

5.2 Metodología NIST.

La metodología para análisis de riesgos propuesta por NIST presenta una serie de etapas donde parte caracterizando el sistema a estudiar, para luego identificar las amenazas y vulnerabilidades presentes, y en base a estas determinar los posibles riesgos y la probabilidad e impacto que causaría la materialización de uno de ellos. Finalmente, se proponen controles para mitigar los riesgos aceptados y se presenta un informe claro al respecto [6].

[9] Probabilistic Risk Assessment NASA - www.hq.nasa.gov/office/codeq/doctree/praguide.pdf
[10] Metodología Análisis y Gestión Riesgos Sistemas Información - www.csae.map.es/csi/pg5m20.htm

A continuación se dará paso al desarrollo de cada una de las etapas de la metodología. El esquema de colores utilizado en las tablas relaciona los niveles que pueden tener la probabilidad, el impacto o el riesgo; donde verde, amarillo y rojo corresponden a los niveles bajo, medio y alto respectivamente.

5.2.1 Caracterización del Sistema.

- Ambiente: está compuesto por universidades e institutos de investigación en Colombia, los cuales aportan recursos humanos y tecnológicos para tal propósito.

- Servicios: se deben prestar los servicios necesarios para hacer posible el acceso de usuarios a la infraestructura, así como la autorización, localización y asignación de los diferentes recursos que componen la infraestructura.

- Información: por ser un grid académico la información administrada no será tan crítica como la de una empresa o entidad financiera, sin embargo debe proveerse los mecanismos para garantizar su seguridad en todo momento.

- Tecnologías: para la implementación del grid computacional se utilizará el middleware gLite por poseer la mayor aceptación y uso debido a los proyectos de diseminación EELA y EELA2 desarrollados por la comunidad europea y dentro de los cuales se impartieron varias capacitaciones en Colombia.

5.2.2 Identificación de Amenazas.

Las amenazas que atentan contra la seguridad de un sistema se pueden generalizar respecto a las principales características de la seguridad de la información (confidencialidad, integridad y disponibilidad) y aquellas que las complementan (autenticación, autorización, audibilidad y no repudio), tal como se muestran en la Tabla 4.

Código	Fuente	Amenaza
A1	Atacante (Interno o Externo)	Espionaje de información sobre datos trasmitidos o almacenados.
A2		Modificación no autorizada de la información
A3		Generación de mal funcionamiento o denegación del servicio
A4		Suplantación de identidad o burla de sobre los sistemas de identificación.
A5		Acceso no autorizado al sistema y su información.
A6		Sabotaje del sistema sin dejar logs, registros o bitácoras que indiquen lo ocurrido.
A7		Negación de operaciones o procesos ejecutados por el usuario.

Tabla 4. Identificación de Amenazas.

5.2.3 Identificación de Vulnerabilidades.

En este punto es importante ver que pueden encontrarse vulnerabilidades a nivel general y directamente relacionadas con la grid, para el caso serán tomadas en cuenta estas últimas y acotadas por los procesos de autenticación y autorización.

Código	Vulnerabilidad
V1	Falta de Protección a Certificados y Llaves Privadas
V2	Falta de Protección a Reglas de Control de Acceso
V3	Centralización del Sistema de Autenticación
V4	Centralización del Sistema de Autorización
V5	Falta de Flexibilidad en Reglas de Control de Acceso
V6	Centralización de la Infraestructura Llave Pública
V7	Fallas en Proceso de Revocación de Identidad
V8	Poca Información para Auditoria y Trazabilidad
V9	Baja Interoperabilidad entre Sistemas (Uso Estándares)

Tabla 5. Identificación de Vulnerabilidades.

5.2.4 Identificación de Controles Existentes.

En el capítulo 4 se ha realizado un bosquejo de la tecnología con sus ventajas y desventajas, por tanto no será abordado de nuevo, Sin embargo se presenta una relación entre las amenazas y vulnerabilidades encontradas contra los riesgos que se generan.

Amenazas	Vulnerabilidades	Riesgos
A1. Confidencialidad, A4. Autenticación	V1. Falta de Protección a Certificados y Llaves Privadas	R1. Suplantación de identidad.
A3. Integridad, A5. Autorización	V2. Falta de Protección a Reglas de Control de Acceso	R2. Escalamiento de privilegios
A3. Disponibilidad	V3. Centralización del Sistema de Autenticación	R3. Indisponibilidad del servicio de autenticación
A3. Disponibilidad	V4. Centralización del Sistema de Autorización	R4. Indisponibilidad del servicio de autorización
A4. Autenticación, A5. Autorización	V5. Falta de Flexibilidad en Reglas de Control de Acceso	R5. Acceso a recursos no permitidos
A3. Disponibilidad A4. Autenticación.	V6. Centralización de la Infraestructura de llave pública	R6. Indisponibilidad infraestructura PKI.
A1. Confidencialidad, A2. Integridad	V7. Fallas en Proceso de Revocación de Identidad	R5. Acceso a recursos no permitidos
A6. Auditabilidad, A7. No Repudio.	V8. Poca Información para Auditoria y Trazabilidad	R7. Evasión de responsabilidades
A4. Autenticación, A5. Autorización.	V9. Baja Interoperabilidad entre Sistemas	R8. Perdida de integración entre organizaciones

Tabla 6. Relación entre Amenazas, Vulnerabilidades y Riesgos.

5.2.5 Determinación de Niveles de Probabilidad.

Para este análisis se utilizaran los criterios estándar para determinar los niveles de probabilidad de que una amenaza pueda aprovechar una vulnerabilidad ocasionando así de que se materialice un riesgo. En la tabla 7 se pueden observar estos criterios y en la tabla 8 la clasificación de cada riesgo respecto a los niveles de probabilidad definidos.

Nivel	Definición de Probabilidad
ALTO	La fuente de la amenaza está altamente motivada y es suficientemente capaz de realizar el ataque, los controles para prevenir la vulnerabilidad son totalmente inefectivos.
MEDIO	El origen de la amenaza está motivado y es capaz de realizar un ataque, y los controles para prevenir la vulnerabilidad están implementados pero poseen algunas debilidades.
BAJO	El origen de la amenaza está bajamente motivado o los controles para prevenir la vulnerabilidad están correctamente implementados.

Tabla 7. Definición de los Niveles de Probabilidad.

Riesgo	Nivel de Probabilidad
R1. Suplantación de identidad.	Medio
R2. Escalamiento de privilegios	Medio
R3. Indisponibilidad del servicio de autenticación	Alto
R4. Indisponibilidad del servicio de autorización	Alto
R5. Acceso a recursos no permitidos	Medio
R6. Indisponibilidad infraestructura PKI.	Medio
R7. Evasión de responsabilidades	Alto
R8. Perdida de integración entre sistemas	Alto

Tabla 8. Nivel de Probabilidad Correspondiente a cada Riesgo.

5.2.6 Determinación de Niveles de Impacto.

Al igual que la sección anterior, se utilizaran los criterios estándar para los niveles de impacto que tenga un riesgo, los cuales son observables en la tabla 9, así como la valoración del impacto que tiene cada riesgo puede observarse en la tabla 10.

Nivel	Definición de Impacto
ALTO	La materialización del riesgo puede resultar en altos costos para la organización; afecta el cumplimiento de sus objetivos y puede llegar a ocasionar severas lesiones e incluso la muerte de algunas personas.
MEDIO	La materialización del riesgo puede resultar en costos para la organización; afecta el cumplimiento de sus objetivos y poner en riesgo la integridad de las personas.
BAJO	La materialización del riesgo puede resultar en costos poco significativos para la organización; afectando levemente el cumplimiento de objetivos.

Tabla 9. Definición de los Niveles de Impacto.

Riesgo	Nivel de Impacto
R1. Suplantación de identidad.	Medio
R2. Escalamiento de privilegios	Alto
R3. Indisponibilidad del servicio de autenticación	Alto

R4. Indisponibilidad del servicio de autorización	Alto
R5. Acceso a recursos no permitidos	Medio
R6. Indisponibilidad infraestructura PKI.	Medio
R7. Evasión de responsabilidades	Medio
R8. Perdida de integración entre sistemas	Medio

Tabla 10. Nivel de Impacto Correspondiente a cada Riesgo.

5.2.7 Determinación de Niveles de Riesgo.

La determinación de los niveles de riesgo es la etapa en la cual se conocen los riesgos más importantes y por ende es posible planear controles de seguridad que mitiguen riesgos de acuerdo con su nivel, en otras palabras es posible priorizar que riesgos se pueden mitigar teniendo en cuenta la relación costo-beneficio.

Para conocer el nivel el riesgo, es necesario conocer tanto el nivel de probabilidad con la cual puede ocurrir junto el nivel de impacto que causaría el mismo. Para esto se utilizaran los valores estándar definidos en la tabla 11 y se mostraran en la tabla 12 y 13 su valoración y ubicación dentro del mapa de riesgos.

	Impacto		
Probabilidad	BAJO (10)	MEDIO (50)	ALTO (100)
ALTO (1.0)	10 x 1.0 = 10	50 x 1.0 = 50	100 x 1.0 = 100
MEDIO (0.5)	10 x 0.5 = 5	50 x 0.5 = 25	100 x 0.5 = 50
BAJO (0.1)	10 x 0.1 = 1	50 x 0.1 = 5	100 x 0.1 = 10

Tabla 11. Definición de los Niveles de Riesgo.

Riesgo	**Nivel de Probab.**	**Nivel de Impacto**	**Nivel de Riesgo**
R1. Suplantación de identidad.	Medio	Medio	Medio
R2. Escalamiento de privilegios	Medio	Alto	Medio
R3. Indisponibilidad del servicio de autenticación	Alto	Alto	Alto
R4. Indisponibilidad del servicio de autorización	Alto	Alto	Alto
R5. Acceso a recursos no permitidos	Medio	Medio	Medio
R6. Indisponibilidad infraestructura PKI.	Medio	Medio	Medio
R7. Evasión de responsabilidades	Alto	Medio	Medio
R8. Perdida de integración entre sistemas	Alto	Medio	Medio

Tabla 12. Clasificación de los Riesgos.

	Impacto		
Probabilidad	BAJO	MEDIO	ALTO
ALTO		R7, R8	R3, R4
MEDIO		R1, R5, R6	R2
BAJO			

Tabla 13. Matriz de Ubicación de Riesgos.

5.2.8 Controles Recomendados y Conclusiones del Análisis.

Como es observado la mayoría de los riesgos se encuentran en un rango medio – alto, por lo tanto es necesario tomar medidas para mitigarlos en el corto plazo dependiendo sensibilidad de la información que es tratada y su criticidad dentro de los procesos. Es importante entonces actuar sobre los riesgos mitigándolos, aceptándolos o cediéndolos a un tercero confiable.

Considerando lo anterior se proponen rediseñar los esquemas de Certificación, Autenticación y de Autorización de Globus considerando los riesgos identificados y características deseables en una arquitectura de seguridad distribuida.

Adicionalmente, estos esquemas deben ser altamente acoplables e independientes del sistema de información con el fin de que estos puedan ser utilizados para otro tipo de sistemas de información. Por ende, los nuevos esquemas deben cumplir los siguientes requerimientos:

Esquema de Certificación:

a) Esquema federado, altamente escalable y que permita relaciones de confianza entre sus componentes.
b) Establecer división de funciones entre la autoridad de registro y la autoridad certificadora.
c) Altamente flexible en el proceso de solicitud de certificación digital.
d) Debe proveer múltiples mecanismos de validación a fin de tener disponibilidad y realizar la tarea en el menor tiempo posible.

Esquema de Autenticación:

a) Sistema federado, que sea altamente escalable y que no afecte la disponibilidad total del sistema de cuando uno de sus componentes sufra inconvenientes.
b) Compatible con los esquemas de Autenticación de Globus.
c) Administración de perfiles y usuarios.
d) Dificultar la intercepción y/o robo de credenciales de autenticación.

Esquema de Autorización:

a) Sistema federado, que sea altamente escalable y que no afecte la disponibilidad total del sistema de cuando uno de sus componentes sufra inconvenientes.
b) Definición de reglas de control de acceso en términos de perfiles de usuario, usuarios puntuales, reglas dinámicas dependiendo de atributos de ambiente.
c) Protección de reglas de control de acceso.

Finalmente, a pesar de que este proyecto está centrado en arquitectura, debe tenerse en cuenta que el problema debe abordarse desde una visión holística, y en este sentido deben implementarse controles adicionales a nivel de infraestructura y administración para dar una solución completa al problema.

6 SOLUCIÓN PROPUESTA

6.1 Esquema de Autenticación.

6.1.1 Descripción de la Problemática

Actualmente uno de los mayores problemas en un sistema grid es la usabilidad, pues en la mayoría de los esquemas se exigen credenciales de autenticación y obtener estas credenciales puede convertirse en un dolor de cabeza para los usuarios quienes deben realizar un proceso largo y dispendioso ante una entidad (registro/autenticación) única y remota (en algunos casos), gastando el tiempo que podrían estar trabajando y produciendo resultados.

Además cuando se obtienen las credenciales de acceso nace otro problema, y es la revocación de las mismas. Como realizar la revocación ante una violación de las políticas y como proveer un servicio eficaz para que los sistemas puedan consultar el estado de las credenciales de un usuario y de acuerdo a esto determinar rápidamente si se da o no acceso a la plataforma.

Por otro lado se presenta el problema de obtener una relación costo/efectiva entre administración y disponibilidad, y esto depende directamente de si el sistema es centralizado o distribuido, pues un esquema de autenticación centralizado es fácil de administrar pero sacrifica la disponibilidad y por otro lado un sistema distribuido brinda alta disponibilidad pero complica la administración al ser necesario la sincronización de los diferentes subsistemas.

Por último, las soluciones a nivel de autenticación brindadas por middlewares para grid son soluciones propias que presentan poca interoperabilidad con los esquemas de autenticación existentes en instituciones de investigación o empresas, esto impide la interconexión de diferentes sistemas e implica migraciones y procesos complejos que pueden ser dispendiosos y llevar a la no adopción de la grid.

6.1.2 Descripción de la Solución

Como respuesta a los problemas antes planteados se propone:

La creación de una infraestructura de llave publica que emitirá certificados digitales como credenciales de autenticación, la arquitectura es distribuida en malla con confianza entre nodos lo que mejora el servicio de emisión y evita fallos por disponibilidad o sobrecarga de peticiones.

Mejorar el proceso de revocación implementando un servicio online de consulta denominado OCSP, con el cual se puede consultar el estado de un certificado digital de manera rápida por medio de un canal seguro y obtener una respuesta de la que se puede tener más fiabilidad que una consulta a una lista de revocación local que puede estar desactualizada.

Descentralizar el sistema de autenticación mediante la delegación de tal proceso a cada organización virtual, así cada VO estará encargada de sus usuarios y recursos y atenderá las solicitudes sobre estos, en caso de tener solicitudes sobre otros recursos las re direccionará a la VO encargada. Esto permite aumentar la disponibilidad ya que si se cae un nodo de autenticación solo se caen los recursos de esa VO y no los de toda la grid y por otro lado facilita enormemente la administración ya que cada VO se encarga de sus recursos agilizando procesos de adición/actualización/eliminación de recursos olvidando sincronizaciones entre nodos de las diferentes Vos que componen la grid.

Hacer uso de LDAP como tecnología de autenticación en cada uno de los nodos, este servicio ya es usado y cuenta con gran aceptación en instituciones de investigación y empresas, y es compatible con la autenticación a través de certificados y tokens kerberos, lo que lo hace compatible con infraestructuras de Grid como Globus Toolkit o OGSA-DAI.

6.1.3 Arquitectura Solución

Infraestructura de Llave Pública.

La propuesta entonces para un grid académico nacional es realizar una arquitectura de llave publica federada, donde por cada región geográfica se cuente con una AC que certifique las instituciones/usuarios dentro de su dominio, lo que distribuiría el trabajo y aumentaría la disponibilidad del servicio. También se propone mantener relaciones de confianza entre autoridades de certificación lo que implicaría una infraestructura en malla.

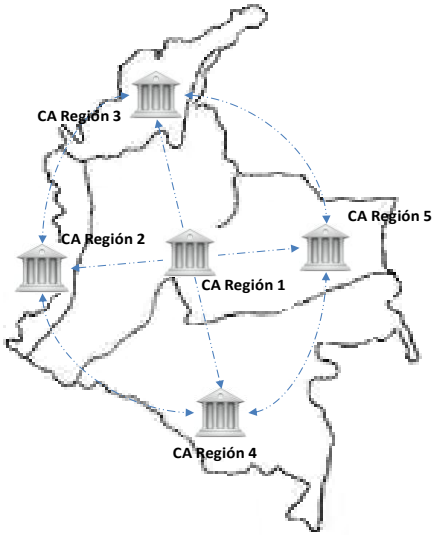

Figura 14. PKI con Infraestructura en Malla.

Ahora, dentro del proceso de certificación se recomienda una división funcional entre una autoridad de registro (AR) y una autoridad de certificación (AC) donde la primera se encargará del proceso de recepción de la solicitud y verificación de la información allí contenida, generando una respuesta de aprobación/reprobación y la con base en esta la segunda (AC) se encargará del proceso de generación del certificado digital como tal.

El proceso modificado empezaría entonces por la solicitud de un usuario a través de algún medio directo u on-line, después la autoridad de registro realiza la recepción de la solicitud, verifica la información y emite un juicio de aprobación o negación, luego un operario toma el resultado de esas solicitudes y las copia desde la autoridad de registro a la autoridad de certificación por medio de un dispositivo usb seguro[11], finalmente la autoridad de certificación firma y genera los certificados digitales.

Realizar esta división permite que solo la autoridad de registro se vea expuesta a la red pública y a la carga generada por las solicitudes, información que si bien es importante no causaría efectos graves en caso de indisponibilidad o pérdida, y por otra parte la autoridad de certificación se mantenga aislada e inaccesible manteniendo así el mayor cuidado de las llaves y certificados que en su base de datos residen, que bajo perdida generarían el caos de la infraestructura.

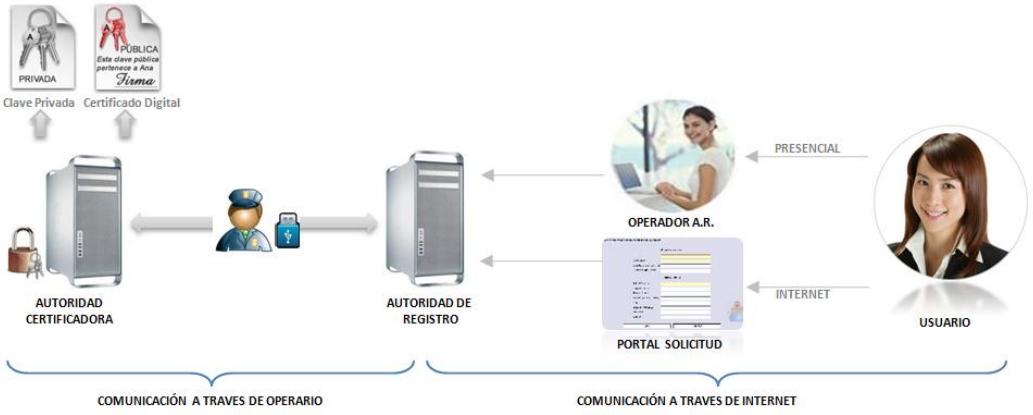

Figura 15. División Funcional de la PKI.

Solucionados los problemas de usabilidad, disponibilidad y seguridad en la infraestructura de llave pública para la generación de certificados se aborda el tema de revocación, donde se debe saber de manera exacta el estado de un certificado digital a fin de permitir o no el acceso a la grid. Para solucionar el problema de las listas de revocación no actualizadas se propone el uso de un servidor OCSP que atiende y resuelve consultas de estados a través del protocolo http sobre una comunicación segura.

Esto permitiría que se contará con dos fuentes de información: la lista de certificados revocados que se encuentra en local y el servidor OCSP que si bien está ubicado remotamente e implica un costo de comunicación es mínimo debido a que la grid opera sobre una red de alta velocidad con baja latencia.

[11] IronKey – http://www.ironkey.com

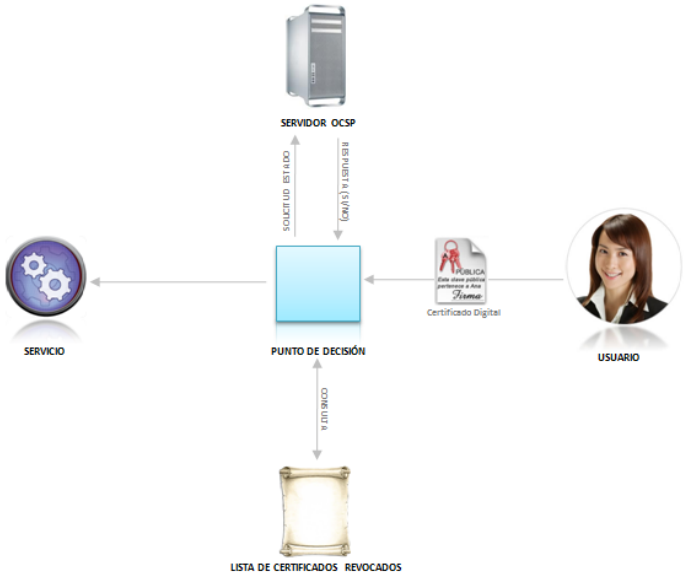

Figura 16. Comprobación del Estado de un Certificado Digital.

Infraestructura Autenticación.

El servicio de autenticación propuesto se basa en servidores LDAP, los cuales prestaran el servicio de autenticación de usuarios a través de certificados digitales. Cada servidor LDAP está determinado por una federación que contiene y autentica a los usuarios pertenecientes a ésta. Una federación es representada por un servidor LDAP, con lo que si se afecta la disponibilidad de un servidor se afectaría la disponibilidad de los recursos y usuarios concernientes a ésta.

La figura 17 muestra la arquitectura propuesta donde se plantea la federación y se muestra que cada servidor de región cuenta con el árbol de directorios y entradas de los recursos y usuarios que este contiene, de este modo está habilitado para atender las solicitudes que sobre sus recursos se hagan. Ahora, si se realiza una solicitud sobre una región o federación que no le se redirige tal solicitud al servidor LDAP encargado de la federación solicitada, dicha redirección se debe realizar a través de un modelo basado en DNS.

Concluyendo el modelo, se puede decir que cada servidor LDAP cuenta con el árbol de directorios que contiene entradas para los recursos de su federación y un link a otros servidores LDAP en las ramas del árbol donde se encuentran las otras federaciones.

Mediante este modelo se provee disponibilidad de la infraestructura pues no se compromete el sistema total sino de manera muy parcial (federación) y por otro lado en vez de sacrificar el rendimiento se mejora pues al distribuir la carga de solicitudes minimizando el trabajo, como es demostrado por Varrette y compañía en las pruebas realizadas sobre la infraestructura francesa grid5000 [13].

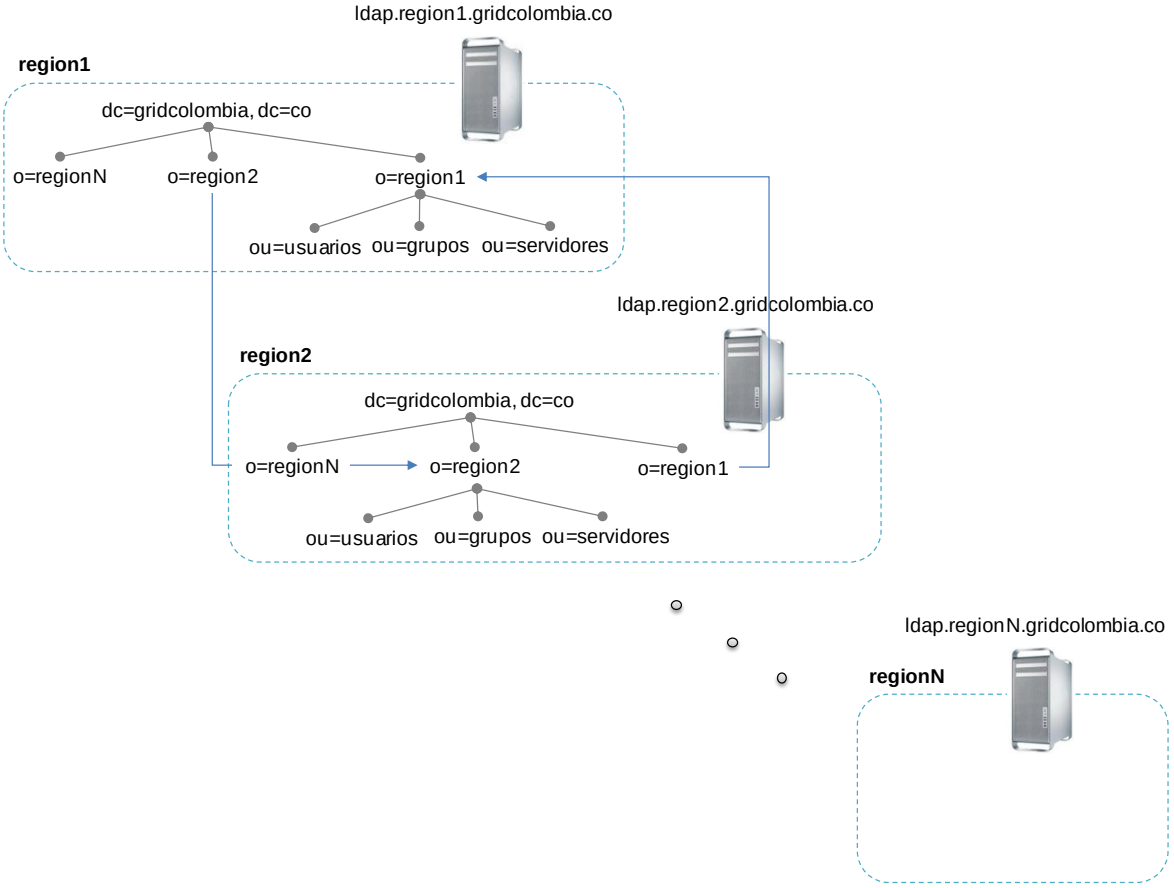

Figura 17. Arquitectura de Autenticación Propuesta.

Esta arquitectura ofrece servidores LDAP que autenticaran los usuarios para que accedan a la grid, la ventaja está en que siendo ldap tan flexible y usado, los usuarios pueden ser usuarios utilizados por otras aplicaciones de cada organización. De este modo se otorga una sola forma de autenticación para todos los recursos y se elimina la necesidad de tener diferentes nombres de usuario para diferentes aplicaciones y alcanzar un esquema de Single Sign On.

La información de un usuario está contenida en un certificado digital y compuesta por:

- Usuario: nombre único de usuario.
- Identificador de Usuario: número de identificación.
- Perfil: grupo o nivel de usuario.
- Dominio: organización al cual pertenece el usuario.

Esta información se encuentra almacenada en el campo Subject del certificado digital en codificación X.500 de la siguiente manera:

CN=(usuario), UID=(identificador), O=(dominio), OU=(perfil1) … OU=(perfiln)

La información que corresponde al nombre de usuario y contraseña, es manejada por un árbol, mientras que la información de los perfiles es almacenada en otro subárbol los cuales se relacionan y describen completamente al usuario.

6.2 Esquema de Autorización.

6.2.1 Descripción de la Problemática

Actualmente Globus y gLite poseen mecanismos de autorización que soportan la federación, lo cual ya no es un problema, sin embargo el acceso a recursos se da a través de directivas sencillas basadas en grupos que solo especifican si un usuario puede o no acceder a un nodo, pero no permiten definir reglas de control de acceso con mayor nivel de detalle, lo que genera un problema de granularidad.

La granularidad en el tema de autorización define el rango de especificación de permisos con el que se puede trabajar y de acuerdo a esto lo más deseado es una granularidad fina que bajo los esquemas y lenguaje actual usado para las reglas de control de acceso, no permite establecer condiciones dinámicas ni dependientes del contexto en el cual se realiza la petición de acceso tales como hora, dirección IP, usuario, etc.

6.2.2 Descripción de la Solución

Este servicio puede ser utilizado por otro servicio que requiera verificar el acceso a un recurso, por parte de un sujeto que desea realizar una acción. Dicho servicio posee un componente llamado Policy Decision Point (PDP), el cual se encarga de recibir peticiones de acceso conformadas por la triada: sujeto, recurso y acción, acompañada de atributos tales como hora de la petición y dirección IP.

El PDP debe localizarse en cada fuente de datos, con el fin de que cada una de estas pueda garantizar la integridad y validez de las reglas de control de acceso y también que cada fuente decida si debe dar el acceso o no.

6.2.3 Arquitectura Solución

El servicio de autenticación es proveído para el sistema Grid pero puede ser usado por cualquier servicio que lo requiera.

Este servicio requiere como entrada:

- Certificado del usuario o servicio que desea hacer la petición
- Consulta en formato MySQL.
- Localización del recurso.

Por otro lado el servicio tiene como salida:

- Respuesta de autorización, que relaciona el certificado del usuario, la consulta y una estampa de tiempo.

Punto de Decisión de Autorización

Para el diseño de la arquitectura se tuvo en cuenta que las reglas deben estar protegidos adecuadamente mediante mecanismos de protección de integridad y confidencialidad tal como se recomienda en el documento de especificación de XACML.
Con el fin de cumplir con este requerimiento, se decidió que el PDP y las reglas en XACML deben permanecer en la fuente de datos con mecanismos de protección adecuados y no deben salir de la fuente de datos. Lo anterior implica que la fuente de datos debe ser la responsable de tomar las decisiones sobre peticiones que se realicen sobre ésta.

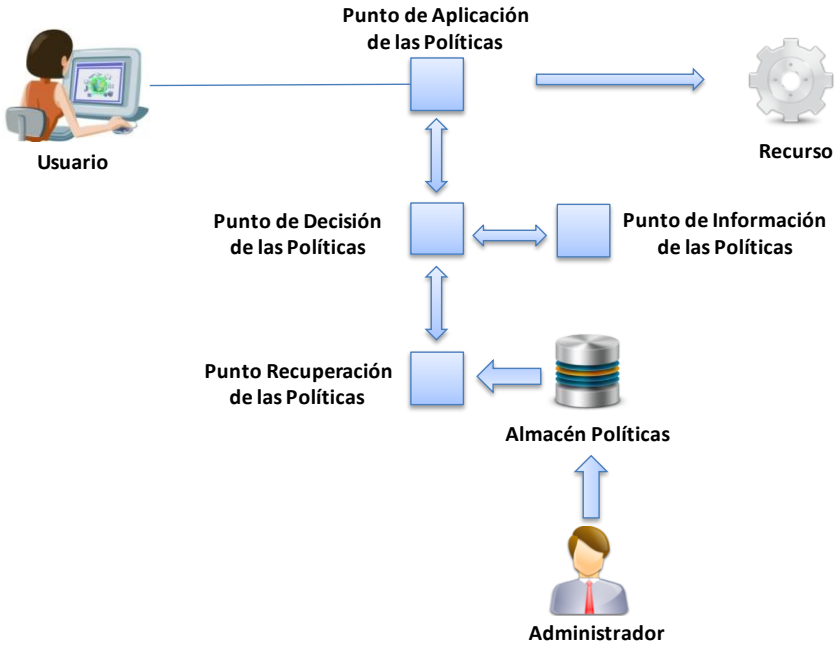

Figura 18. Arquitectura de Autorización

Definición de Atributos de Peticiones y Reglas

Las reglas deben ser especificadas en lenguaje XACML en términos de Subject-Resource-Action y se debe tener en cuenta que dependiendo como se especifique el Subject se obtenga una respuesta. Esto se debe a que el Subject se puede especificar en términos de perfil, nombre de usuario o funciones.
Para poder expresar estos atributos, es necesario definir cuáles serán los atributos del lenguaje de XACML van a ser usados para su representación. Para el esquema de autorización los atributos utilizados y la forma de expresarlos son los siguientes:

- Nombre de Usuario: Se utilizará el atributo urn:oasis:names:tc:xacml:1.0:subject:subject-id expresado de la forma <nombre usuario>@<dominio> para expresar a un usuario específico, o @<dominio> para expresar pertenencia a organizaciones.

- Perfil: Se utilizará el atributo urn:oasis:names:tc:xacml:2.0:subject:role del tipo String.
- Identificador de Usuario: Se usará el atributo urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier del tipo *String*.
- Recursos: Se usará el atributo urn:oasis:names:tc:xacml:1.0:resource:resource-id y se expresará en términos de XPath.

Vale la pena mencionar que la combinación de uno o más de cada uno de estos atributos se estará haciendo referencia a un *Subject* en particular.

Componentes

PDP
Este componente se encarga de recibir las peticiones y se encarga de tomar la decisión de aceptar o denegar la petición. Para esto el PDP debe preguntarle al contenedor de reglas las reglas relacionadas con la triada Subject- Resource-Action.

Para realizar esta comunicación entre estos dos componentes, es necesario establecer en cada conexión una llave privada simétrica (sesión) para poder tener una protección adicional a la ofrecida por Grid. Luego el contenedor debe pasar las reglas relacionadas acompañadas de un resultado de hash de cada una para garantizar la integridad de las mismas.

El procesamiento de la reglas se realiza de la forma del menor privilegio, es decir la respuesta por defecto es denegar y las reglas de control de acceso dan los accesos necesarios.

RulesContainer
Es el componente que se encarga de la protección y la custodia de las reglas proporcionadas por cada fuente de datos. Éste se encuentra localizado en cada fuente de datos y entrega las reglas relacionadas.
Éste posee una interfaz para el registro de nuevas reglas en la fuente, el cual somete a una función de hash a cada una de estas, luego las cifra y las almacena en el disco, con el fin de ofrecer integridad y confidencialidad de las mismas. Al momento de realizar de procesar una regla, éste componente verifica la integridad de la misma y en el caso que no sea integra la regla es descartada.
En el caso que una regla sea descarta no se verá afectada la seguridad del sistema dado a que las reglas solo dan permisos a ciertos usuarios, pero no los niega. En este caso sería propenso un caso de denegación de servicio en el caso que todas las reglas sean eliminadas.

TokenGeneration
Este componente se encarga de la generación del token de autorización que debe ser devuelto al servicio o usuario que llamo el servicio. El token está formado de la siguiente forma: *Token = Hash(<Query SQL>) + Certificado_Solicitante + Certificado_Fuente + Cprisolicitante(Timestamp_Solicitante))*

Es el ciframiento con la llave privada de la fuente de la concatenación del hash de la consulta en SQL, una estampa de tiempo en la cual se inició la solicitud del usuario cifrado con la llave privada del solicitante. Vale la pena mencionar que el token tiene una validez configurable, por lo que se exige que todos los relojes de los nodos participantes deban estar sincronizados.
Para evitar que el token sufra ataques de replicación, se incluye en éste una estampa de tiempo. Ésta va cifrada con la llave privada del solicitante, con la cual se garantizará que el solicitante es en realidad el que dice ser. Tal estampa de tiempo será validada, verificando hace cuanto fue generado. Además, las fuentes de datos almacenarán los tokens utilizados hasta el tiempo de su validez para evitar su doble uso.

Proceso de Autorización

Supóngase un usuario autenticado y posee un certificado que indica que ha realizado su autenticación en el sistema.

a) El usuario contacta el servicio de autorización y le provee: el certificado, la estampa de tiempo cifrada con la llave privada del usuario y la ubicación de la fuente.
b) El servicio de autorización contacta la fuente que entrega una petición en *XACML*.
c) El motor *PDP* le solicita al contenedor de reglas las reglas que apliquen a dicha petición.
d) *RulesContainer* carga las reglas del disco, verifica las firmas digitales y las descifra (Este proceso se realiza al inicio del arranque del servicio). Luego buscas las reglas aplicables.
e) El *PDP* con las reglas aplicables decide si se puede realizar la acción o no.
f) Si se puede realizar la acción, se solicita al *TokenGeneration* la generación del correspondiente *token* y lo retorna, de lo contrario retorna *null*.
g) La consulta llega a la fuente de datos, ésta verifica el *token* con los criterios de:
   - La consulta solicitada concuerde con el hash de la consulta en *SQL* del *token*.
   - El certificado del cliente solicitante concuerda con el del *token*.
   - El certificado de la fuente concuerda con el del *token*.
   - Descifra la estampa de tiempo con el certificado del usuario solicitante y verifica la vigencia de dicha estampa. Tal vigencia es configurable por la fuente de datos.

7 IMPLEMENTACIÓN

7.1 Prototipo de Identificación.

En este punto se realizo la implementación de la autoridad de certificación tal como está documentado en el Anexo 1.

La implementación se hizo mediante el software libre OpenCA, y se configuro de modo que hubiese división funcional entre la autoridad de registro (RA) y la autoridad certificadora (AC).

Figura 19. Solicitud Certificado Digital a Open CA

De acuerdo a la implementación realizada el procedimiento para solicitud/entrega de un certificado digital esta dado por los siguientes pasos:

1. El usuario interesado ingresa a la página http://ca.uis.edu.co/pub donde debe seleccionar la opción “crear una solicitud”
2. A continuación aparecerá una serie de campos los cuales el usuario debe llenar con su información y finalizar enviando la solicitud.
3. Dicha solicitud es recibida por el administrador del sistema y puede ser observada en el área de autoridad de registro con la url http://ca.uis.edu.co/ra
4. El operador AR revisara atentamente la solicitud y verificará los datos y si todo está de acuerdo a lo exigido dará el visto bueno a la solicitud.
5. Un funcionario autorizado se encarga de copiar las solicitudes aprobadas en la autoridad de registro y llevarlas por medio de un dispositivo a la autoridad certificadora.

6. El administrador de la autoridad certificadora puede entrar a su url http://ca.uis.edu.co/ca y dar por aprobada la solicitud emitiendo el certificado digital que será entregado al usuario.

7.2 Prototipo de Autenticación

En este nivel se implementa un esquema reducido donde se prueba la funcionalidad en autenticación a través de un LDAP federado. Para la instalación y configuración se utilizo el software Open LDAP, dicho procedimiento se puede detallar en el Anexo 2.

El ambiente de pruebas creado fue dos organizaciones virtuales cada una con sus usuarios y sus recursos computacionales representados en nodos de procesamiento.

Para lograr la interacción entre usuarios/recursos cada organización cuenta con un servidor LDAP que administra los arboles referentes a usuarios y nodos de cada sitio, de modo que el servidor LDAP1 se encargará de autenticar y dar acceso a los recursos de la organización 1 y el servidor LDAP2 se encargará de autenticar y dar acceso a los recursos de la organización 2.

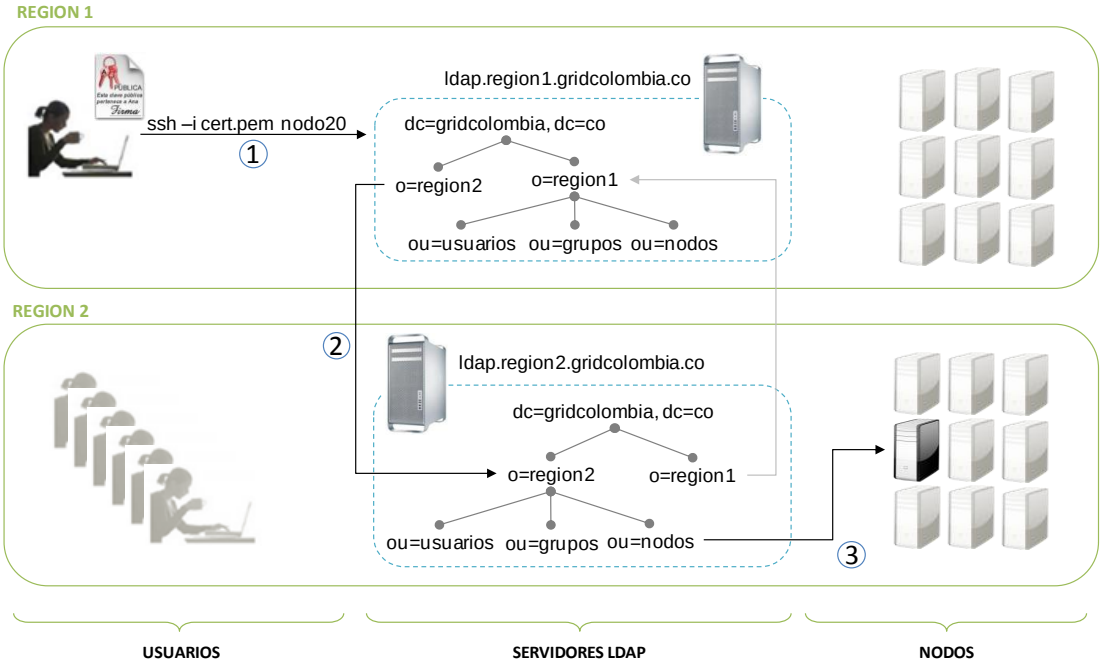

Figura 20. Autenticación a Través de LDAP Federado

La prueba se realizo a través de los siguientes pasos:

1. Un usuario perteneciente a la organización 1 hace una solicitud de un recurso en la organización 2 para autenticarse por medio de ssh con la instrucción:
   ssh –i certificado_usuario.pem recurso_organizacion2

2. La solicitud llega al servidor LDAP1 quien busca en su árbol LDAP, este al ver que se hace referencia a un recurso de la organización 2 va a esa rama y encuentra la redirección por DNS hacia el servidor LDAP2
3. El servidor LDAP2 toma la solicitud y hace la búsqueda bajo su árbol donde encuentra los recursos y así atiende la solicitud.

7.3 Prototipo de Control de Acceso.

El objetivo con esta prueba es observar la funcionalidad de acceso a una aplicación mediante el uso de certificados digitales.

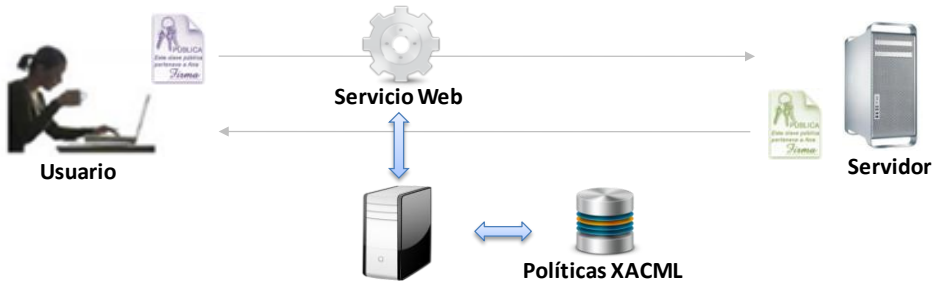

Figura 21. Acceso a un Recurso a Través de Certificados Digitales

La prueba consistió en habilitar un servidor web haciendo uso del software libre Apache el cual fue configurado de manera adecuada para que exigiera la presentación un certificado digital valido para permitir el acceso a la navegación de páginas web, este procedimiento puede ser detallado en el Anexo 3.

La prueba consiste de los siguientes pasos:

1. El usuario realiza una solicitud https a la dirección del servidor https://192.168.0.100
2. Una vez el servidor recibe la solicitud y responde mediante el protocolo https demostrando que está en servicio y que requiere la presentación de un certificado
3. En el browser del cliente se despliega una ventana que muestra los certificados locales y allí el cliente debe seleccionar el que desea presentar
4. El servidor apache recibe la nueva comunicación y carga las variables de ambiente normales y las variables SSL, en estas últimas se almacena temporalmente la información que está en el certificado digital del usuario.
5. Teniendo el servidor la información del certificado digital valida que el certificado sea expedido por una autoridad certificadora de confianza, que no haya caducado por tiempo y que no tenga estado revocado. Además de validar su origen, validez de tiempo y estado podría evaluarse más campos mediante la inclusión de instrucciones propias de apache o usando Perl.
6. Hecha la verificación del certificado y aprobando todos los ítems el servidor concede al usuario acceso a la aplicación web.

Debe recordarse que la comunicación se realiza mediante protocolo de cifrado simétrico, más específicamente SSL basado en el algoritmo de Diffie-Helman.

8 CONCLUSIONES Y TRABAJO FUTURO

8.1 Conclusiones.

La metodología de análisis de riesgos es una herramienta eficiente para realizar un proceso minucioso de identificación, valoración y mitigación de riesgos, específicamente la propuesta por NIST pues se acopla a entornos generalizados como es el caso.

La utilización de servidores LDAP permite federar el sistema proporcionando escalabilidad, facilita la administración de usuarios definiendo dominios de operaciones, permite integración de aplicaciones en la compañía y proporciona interoperabilidad con middlewares como Globus Toolkit a través del uso de certificados digitales.

El esquema de autorización utiliza XACML un sistema de expresión de reglas amplio y estandarizado. Tal decisión facilita la definición de reglas de control de acceso dinámicas basadas en condiciones de ambientes o atributos de pertenencia y/o de utilización mejorando la granularidad de los permisos, a su vez permite la integración o utilización de políticas de diferentes sitios.

Finalmente, la propuesta planteada no es la simple unión de controles propuestos al azar sino que es el producto de un proceso de análisis basado en el contexto académico nacional, los requerimientos planteados, el estado del arte y los riesgos no mitigados por las infraestructuras de seguridad existentes. En la Tabla 14 se muestra las características finales de la infraestructura de seguridad propuesta y su mejora respecto a la planteada inicialmente (Globus Toolkit).

Característica	Arquitectura Propuesta	Característica	Arquitectura Propuesta
Confidencialidad	SSL / WS-Security	Control de Acceso	Reglas Dinámicas/ XACML
Integridad	SSL / WS-Security	Single Sign-On	Certificados Digitales
Disponibilidad	Alta	Revocación	CRLs / Servidor OCSP
Autenticación	Federada / CAs	Usabilidad	Alta
Autorización	Federada / LDAP	Escalabilidad	Alta
Auditabilidad	Baja	Interoperabilidad	Alta

Tabla 14. Ventajas y Desventajas de la Arquitectura de Seguridad Propuesta.

8.2 Trabajo Futuro.

Profundización en el proceso de implementación de XACML y utilización de herramientas que hagan sencilla la labor de administración a fin de ofrecer un despliegue rápido y efectivo de las políticas locales e integración con políticas remotas.

Análisis y diseño de una arquitectura que incorpore auditabilidad, permitiendo la trazabilidad en las actividades desarrolladas por un usuario (acceso, transacciones, uso de recursos, interacción, etc.)

Dada la orientación del mercado a dispositivos móviles y la mejora de hardware de los mismos día a día, es factible que posteriormente las infraestructuras grid cuenten con este tipo de dispositivos para proveer poder computacional, en este sentido deben proveerse sistemas de autenticación/autorización acordes con estos sistemas ubicuos y dinámicos que tomarán importancia en el futuro.

9 BIBLIOGRAFÍA

[1] Anirban Chakrabarti, Grid Security. Springer, 2007.

[2] Abhijit Belapurkar et Al, Distributed Systems Security. Wiley, 2009.

[3] Xukai Zou et Al, Trust and Security in Collaborative Computing. World Scientific, 2008.

[4] I. Foster, y A. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure. Elsevier-Morgan Kaufman, 2004.

[5] R. Sampieri et al, Metodología de la Investigación. McGraw-Hill, 2006.

[6] G. Stoneburner, A. Goguen, A. Feringa, "Risk Management Guide for Information Technology Systems. SP 800-30. Nist Special Publications. 2002.

[7] Sécurité des Architectures de Calcul Distribué: Authentification et Certification de Résultats. Sébastien Varrette, Tesis Doctoral, 2007.

[8] A. Chakrabarti, A. Damoda, S. Sengupta, “Grid Computing Security: A Taxonomy”. IEEE, 2007.

[9] Bendahmane A. et Al, “Grid Computing Security Mechanisms: State-of-The-Art”. IEEE, 2009.

[10] Ellison C. y Schneier, “Ten Risks of PKI: What You are not Being Told about Public Key Infrastructure”. Computer Security Journal, Vol. XVI, 2000.

[11] Haidar N. y Abdallah A. “Comparison and Evaluation of Identity Management in Three Architectures for Virtual Organizations”. IEEE, 2008.

[12] Nasrat A y Abdallah, “Formal Modelling of PKI Based Authentication”. Elsevier, 2009.

[13] Shushan Zhao et Al, “PKI-Based Authentication Mechanisms in Grid Systems”. IEEE, 2008.

[14] S. Zhao, A. Aggarwal y R. Kent, “A Framework for Revocation of Proxy Certificates in a Grid”. IEEE, 2007.

[15] Shaomin Zhang, Baoyi Wang, “Research on an Extended OCSP Protocol for Grid”. IEEE 7th World Congress on Intelligent Control and Automation, 2008.

[16] S. Varrette, S. Georget, J. Montagnat, J.Roch y F. Leprevost, “Distributed Authentication in GRID5000” Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.8696&rep=rep1&type=pdf

[17] Sinnott R. et Al, " Advanced Security for Virtual Organizations: The Pros and Cons of Centralized vs Decentralized Security Models". IEEE Symposium CCGRID, 2008.

[18] Ciaschini V., "Domains Distributed Policy Framework Across Multiple Grid". IEEE, 2007.

[19] P. Liu, R. Zong y S. Liu, "Kerberos: A New Model for Authentication and Authorization across Heterogeneous Trust-Domain". IEEE, 2008.

[20] Sang M. Park y Soon M. Chung, "Enhanced CAS Certificate for Metadata-Based Access Control in Grid". En 20th IEEE Conference on Tools with Artificial Intelligence, 2008.

[21] R. Alfieri et Al. VOMS: an Authorization System for Virtual Organizations. 1st European Across Grids Conference, 2003.

[22] Chakrabarti, A. Damodaran. Enterprise Authorization and Licensing Service. Infosys Tech. Report, 2006.

[23] M. Thompson, A. Essiari, S. Mudumbai. ACM Transactions on Information and System Security, vol. 6, 2003.

[24] D. Chadwick, O. Otenko. The PERMIS X.509 Role Based Privilege Management Infrastructure. ACM, 2002.

[25] H. Khider y T.Osman, "Attribute-Based Authorization for Grid Computing". IEEE, 2010.

[26] D. Chadwick, A. Otenko, y E. Ball, "Role-Based Access Control with X.509 Attribute Certificates". IEEE, 2003.

[27] Y. Demchenco et Al, "XACML Policy Profile for Multidomain Network Resource Provisioning and Supporting Authorization Infrastructure". IEEE International Symposium on Policies for Distributed Systems, 2009.

[28] M. Laurch et Al, "First Experiences Using XACML for Access Control in Distributed Systems". En ACM Workshop on XML Security, 2003.

[29] L. Shengjian, "A study on the Mechanisms of Policy-based Grid Authorization". En International Conference on Multimedia Information Networking and Security, 2009.

[30] Open Grid Forum. Disponible en: http://www.ogf.org

[31] Grid Colombia. Disponible en: http://www.gridcolombia.org

[32] Globus Alliance: globus toolkit . Disponible en: http://www.globus.org

[33] gLite. Disponible en: http://glite.web.cern.ch/glite

[34] Open CA. Disponible en: http://www.openca.org

[35] Kerberos. Disponible en: http://web.mit.edu/kerberos

[36] Open LDAP. Disponible en: http://www.openldap.org

[37] XACML. Disponible en: www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

ANEXO 1. INSTALACIÓN DE OPEN CA

Prerrequisitos

Como OpenCA no es un sistema monolítico sino que usa otros productos de Open Source, se deben instalar primero unos prerrequisitos a través de la herramienta administradora de paquetes yum, estos son:

#yum install httpd mod_ssl openssl openldap perl

Además se deben instalar los siguientes módulos de perl a través de cpan:

#cpan –i Authen::SASL CGI::Session Convert::ASN1 Digest::HMAC Digest::MD5 URI Digest::SHA1 Encode::Unicode IO::Socket::SSL MIME::Base64 MIME::Lite MIME::tools MailTools Net::Server perl-ldap IO::stringy XML::Twig X500::DN Parse::RecDescent libintl-perl

Estos módulos se pueden instalar con la herramienta cpan la cual se conecta a un repositorio de módulos de perl y los instala remotamente. En caso de que no los encuentre, verifique que no esté buscando por un alias, sino también se pueden buscar y descargar de la página de cpan (http://search.cpan.org/), por orden se deberían descargar en el directorio home de perl (/usr/lib/perl5/) y una vez descargado el modulo en formato .tar.gz ejecutar: tar -zxvf modulo_perl.tar.gz, cd modulo_perl, perl Makefile, make, make test, make install.

Si su sistema tiene soporte de yum o apt-get actualizado, se puede hacer la instalación dando yum install seguido de perl-nombre_modulo, recuerde quitar los separadores :: y reemplazarlos por -.Ej: yum install perl-XML-Twig, en caso de usar apt-get deje la palabra perl al final del nombre del modulo.

A continuación se pone nombre a la maquina y para esto modifican algunos ficheros:

echo 'ca' > /etc/hostname
echo '192.168.0.101 ca.uis.edu.co ca' >> /etc/hosts
vi /etc/httpd/conf/httpd.conf copiar allí 'ServerName ca.uis.edu.co'

Para manejo de openca se crea un usuario y un grupo:

groupadd -g 1025 openca
useradd -g openca -s /bin/bash -u 1025 -c "OpenCA User" openca

Para el manejo de apache se pueden tomar prestados el usuario y el grupo apache que están creados por defecto en el sistema.

Instalación

En algunas ocasiones se puede generar error con ssl a pesar de que ya se haya instalado por medio de un paquete rpm, en este caso es recomendable instalarlo por medio de un paquete .tar.gz (teniendo en cuenta que sea la misma versión del que ya se encuentra en el sistema, que a su vez debe coincidir con la versión de mod_ssl) para esto se realizar los siguientes pasos:

```
#cd /usr/local
#wget http://www.openssl.org/source/openssl-0.9.7b.tar.gz
#tar xvzf openssl-0.9.7b.tar.gz
#cd openssl-0.9.7e
#./config
#make
#make install
```

Estando todos los prerrequisitos listos, se puede entrar propiamente en OpenCA, lo primero que se debe hacer es instalar OpenCA-Tools, ejecutando:

```
#cd /usr/local/src
#wget https://www.openca.org/alby/download?target=openca-tools-1.0.0.tar.gz
#tar -zxvf openca-tools-1.0.0.tar.gz
#cd openca-tools-1.0.0
#./configure \
  --with-openca-prefix=/usr/local/openca \
  --prefix=/usr/local/openca \
  --with-openca-user=openca \
  --with-openca-group=openca \
  --with-openssl-prefix=/usr/local/ssl

 #make
 #make install
```

Después la instalación de OpenCA llevando a cabo los siguientes pasos:

```
#cd /usr/local/src
#wget https://www.openca.org/alby/download?target=openca-0.9.3-rc1.tar.gz
#tar -zxvf openca-0.9.3-rc1.tar.gz
#cd openca-0.9.3-rc1
```

Ejecutar el configure especificando la mayor cantidad de parámetros posibles:

```
 #./configure \
  --with-openca-prefix=/usr/local/openca \
  --with-openca-user=openca \
  --with-openca-group=openca \
  --with-web-host=ca.uis.edu.co \
  --with-httpd-user=apache \
  --with-httpd-group=apache \
  --with-htdocs-fs-prefix=/var/www/html \
  --with-cgi-fs-prefix=/var/www/cgi-bin \
  --with-ext-prefix=/usr/local/src/openca-0.9.3-rc1 \
  --with-ca-organization="UIS" \
  --with-ca-locality="Bucaramanga" \
```

```
--with-ca-country="CO" \
--with-etc-prefix=/usr/local/openca/etc \
--with-var-prefix=/usr/local/openca/var \
--with-lib-prefix=/usr/local/openca/lib \
--with-module-prefix=/usr/local/openca/modules \
--enable-dbi \
--enable-rbac \
--with-service-mail-account="root@localhost" \
--with-db-type=mysql \
--with-db-name=openca \
--with-db-host=localhost \
--with-db-port=3306
```

Después se ejecuta make y make install para crear la ca y el nodo

```
#make
#make install-ca
#make install-node
```

Ahora hacemos otro configure pero cambiamos el parámetro de jerarquía a la ra:

```
#./configure \
  --with-openca-prefix=/usr/local/openca \
  --with-openca-user=openca \
  --with-openca-group=openca \
  --with-web-host=ca.uis.edu.co \
  --with-httpd-user=apache \
  --with-httpd-group=apache \
  --with-htdocs-fs-prefix=/var/www/html \
  --with-cgi-fs-prefix=/var/www/cgi-bin \
  --with-ext-prefix=/usr/local/src/openca-0.9.3-rc1 \
  --with-ca-organization="UIS" \
  --with-ca-locality="Bucaramanga" \
  --with-ca-country="CO" \
  --with-etc-prefix=/usr/local/openca/etc \
  --with-var-prefix=/usr/local/openca/var \
  --with-lib-prefix=/usr/local/openca/lib \
  --with-module-prefix=/usr/local/openca/modules \
  --enable-dbi \
  --enable-rbac \
  --with-service-mail-account="root@localhost" \
  --with-db-type=mysql \
  --with-db-name=openca \
  --with-db-host=localhost \
  --with-db-port=3306
```

Después se ejecuta make y make install para crear la ca y el nodo

```
#make
#make install-ra
#make install-pub
#make install-node
```

La instalación de OpenCA presenta un error pues los scripts no son puestos por defecto en $openca/bin, por lo tanto se deben copiar a mano desde el directorio fuente al de instalación y cambiar los permisos de estos

#cd /usr/local/src/openca-0.9.3-rc1/src/scripts/
#make install
#cp /usr/local/src/openca-0.9.3-rc1/src/scripts/ /usr/local/openca/bin/*
*#chmod 755 /usr/local/openca/bin/**

Se arreglan los permisos del directorio log:

#chown -R apache:apache /usr/local/openca/var/log/

Ahora se instala la base de datos donde se pueda guardar la información de la ca y la ra, esta b.d puede estar en postgresql, mysql u oracle. Esta instalación está basada en mysql, los pasos son:

Instalación de mysql vía yum:

yum install mysql-server

Se inicializa mysql:

mysql_install_db

Arranca el demonio:

service mysqld start

Configuración del password de root:

mysqladmin -u root password 'rootmysql'
mysqladmin -u root -h ca.uis.edu.co password 'rootmysql'

Creación de bases de datos y asignación de permisos a ca y ra respectivamente:

mysql -u root -p
<password>
create database openca;
create database openra;
grant all privileges on openca. to openca@localhost identified by "openca";*
grant all privileges on openra. to openra@localhost identified by "openra";*
exit;

Verificación de las bases de datos ca y ra:

mysql -u openca -p
Enter password: "openca"
use openca;
show tables;
(deberia retornar vacia (empty))
exit;
mysql -u openra -p
Enter password: "openra"
use openra;
show tables;
(deberia retornar vacia (empty))
exit;

Reiniciar servicio mysql

```
service mysqld restart
```

Una de las partes más importantes es la revisión y modificación del fichero config.xml:

```
cd /usr/local/openca/etc
cp config.xml config.xml.original
vi  config.xml
```

Se hace una revisión y se configuran las opciones generales, opciones configuración servidor web, opciones configuración ldap, opciones configuración base de datos, configuración de módulos, enlaces relativos y absolutos. La mayoría de los parámetros quedan por defecto basados en el configure, pero para ver los cambios hechos se ejecuta:

```
diff -Naur config.xml.original config.xml
--- config.xml.original 2007-09-08 12:35:19.000000000 -0500
+++ config.xml  2007-09-08 14:49:04.000000000 -0500
@@ -55,7 +55,7 @@
                strings in national languages here.
             -->
           <name>ca_organization</name>
-          <value></value>
+          <value>UIS</value>
         </option>
         <option>
           <!--
@@ -63,7 +63,7 @@
                strings in national languages here.
             -->
           <name>ca_locality</name>
-          <value></value>
+          <value>Bucaramanga</value>
         </option>
         <option>
           <!--
@@ -72,7 +72,7 @@
               this country code is ALWAYS two characters long
             -->
           <name>ca_country</name>
-          <value></value>
+          <value>CO</value>
         </option>
         <option>
           <name>sendmail</name>
@@ -84,7 +84,7 @@
         <option>
           <name>service_mail_account</name>
-          <value></value>
+          <value>erickmeneses@gmail.com</value>
         </option>
         <option>
           <name>policy_link</name>
```

```
@@ -393,7 +393,7 @@

        <option>
          <name>USE_LOAS</name>
-         <value>yes</value>
+         <value>no</value>
        </option>
        <option>
          <name>prefix</name>
```

Después se deben crear los ficheros de configuración reales, para esto se ejecuta el script configure_etc.sh el cual modifica los archivos, por ejemplo convierte index.html.templates en index.html.

```
#cd /usr/local/openca/etc
#./configure_etc.sh
```

Bug: en caso de que genere error por no encontrar el fichero openca-configure dentro del directorio /usr/local/bin/ copie este fichero y demás ejecutables desde la ruta /usr/local/openca/bin/

```
#cp /usr/local/openca/bin/*  /usr/local/bin/
```

Y después borre todos los ficheros que se pegaron pero no son ejecutables, estos tienen extensión .in

```
#rm -rvf /usr/local/bin/*.in
```

Ahora se debe copiar el fichero openca_rc en init.d para tener el servicio openca, después activarlo.

```
#cp /usr/local/openca/etc/openca_rc /etc/init.d/openca
#cd /etc/init.d/
#chkconfig --add openca
#service openca start
```

Después de esto se puede verificar en el browser entrando a la dirección https://ca.uis.edu.co/ca, esto mostrara una página web para el manejo de certificados reales, la entrada se realiza con el login root y password root los cuales están definidos por defecto.

Instalar los certificados ssl del apache

```
#mkdir /etc/httpd/conf.d/ssl
#cd /etc/httpd/conf.d/ssl/
#openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -days 3650 -out cacert.pem -nodes
```

Con este comando creamos un CA para certificados X509 con algoritmo de encriptación rsa de 2048 bytes. Con el -keyout le indicamos que la clave privada de nuestra CA se almacene en el fichero cakey.pem y la clave publica -out en el cacert.pem. Seguidamente nos pedirá un password para nuestra CA y se lo damos. También nos pedirá una serie de datos por ejemplo país, nombre de institución, que nos identifica como CA.

Finalmente se edita el fichero /etc/httpd/conf.d/ssl.conf y al final se agrega

```
Listen 443
<VirtualHost *:443>
```

```
        SSLEngine On
        SSLOptions +StdEnvVars
        SSLCertificateFile /etc/httpd/conf.d/ssl/cacert.pem
        SSLCertificateKeyFile /etc/httpd/conf.d/ssl/cakey.pem
    </VirtualHost>
```

Guardar y reiniciar apache

/etc/init.d/apache restart

Abrir un navegador y probar las urls: https://ca.uis.edu.co/ca, https://ca.uis.edu.co/ra https://ca.uis.edu.co/node, https://ca.uis.edu.co/pub

Configuración

Configuración Offline de la maquina:

General -> Initialization -> Initialize the Certification Authority

Show SQL statements for database initialization (Revisar si todo esta bien aquí)

Initialize Database (Debería verse el mensaje: "The database was successfully initialized.")

Generate new CA secret key -> choose your settings -> OK

Escoger y digitar la passphrase para la llave secreta de la CA, después la llave privada codificada PEM debe ser exhibida.

Generate new CA Certificate Request (use generated secret key) -> Edite los campos -> OK -> confirme la próxima pagina con OK -> digite la passphrase CA nuevamente.

Self Signed CA Certificate (de la petición ya generada)
-> choose the validity (Se muestran los detalles del certificado de la CA)

Rebuild CA Chain (Se muestra administración exitosa)

- -

General -> Initialization -> Create the initial administrator -> Create a new request (Se deben llenar toda su información y escoger role de CA Operator)

-> Continue -> Continue ->

(Todos tus datos son mostrados, podría editar la solicitud, pero no deberla ser necesario)

Issue the certificate -> click the "Issue certificate" button ->
(Después digite la clave secreta de administrador de la CA y los detalles seràn mostrados)

Handle the certificate -> Certificate and Keypair - PKCS#12 -> click "Download"

(Digite el passphrase que ingreso cuando hizo el requerimiento del certificado (PIN))
-> save

Una vez se descargue el certificado de administrador de la CA, debe subirse al browser del administrador para que con este se puedan firmar los certificados de los clientes, se debe también habilitar las opciones (servidores web, correo electrónico y aplicaciones) para ese certificado.

- -

General -> Initialization -> Create the initial RA certificate

-> Create a new request

(Llene toda la información y seleccione web server como role y hotsname como nombre)

-> Continue -> Continue (Los datos serán mostrados).

(Podría editar la información de la petición en Edit Request pero no debería ser necesario)

Issue the certificate -> click en el boton "Issue certificate"

(Digite la passphrase como administrador de la CA, después los datos serán mostrados)

Handle the certificate -> Certificate and Keypair - PKCS#12 -> click "Download" ->

((Digite el passphrase que ingreso cuando hizo el requerimiento del certificado (PIN))

-> save & import as racert.p12

- -

Ahora se exporta la configuración y los certificados a la maquina ra:

General -> Node management -> login (root/root otra vez) -> Administration -> Dataexchange -> Enroll data to a lower level of the hierarchy -> All -> click "OK"

(Revise los mensajes en los logs para ver los posibles errores pero debería esperarse que todo este correcto y el archivo exportado este localizado en el lugar especificado en el fichero config.xml).

Configuración Online de la Maquina:

Primero se intercambia el certificado uno auto-firmado (self-signed) del servidor que esta en la máquina en línea por el que se acabo de crear. Copie racert.p12 a la máquina en línea y conviertalo a un codificado PEM, par key&cert:

```
openssl pkcs12 -in racert.p12 -clcerts -nokeys -out hostcert.pem
openssl pkcs12 -in racert.p12 -nocerts -nodes -out hostkey.pem

cp /root/hostkey.pem /etc/httpd/conf/ssl.key/server.key
cp: overwrite `/etc/httpd/conf/ssl.key/server.key'? y

cp /root/hostcert.pem /etc/httpd/conf/ssl.crt/server.crt
cp: overwrite `/etc/httpd/conf/ssl.crt/server.crt'? y

service httpd restart
service openca start
```

Ahora ponga en su browser https://ca.uis.edu.co/ra con login y password root/root.

- -

General -> Server management -> login with root/root

Administration -> Server Init -> Show SQL statements for database initialization -> Check statements

Administration -> Server Init -> Initialize Database
Se debería observar: "The database was successfully initialized."

Para el siguiente paso usted debe copiar los datos exportados desde el dispositivo (/tmp/fd0 en nuestro caso, desde la maquina offline) a la maquina Online (otra vez en nuestro caso /tmp/fd0).

Administration -> Server Init -> Import Configuration -> click "OK" -> you should see a log file - check for errors

Solicitando el Primer Certificado:

Con el objetivo de hacer pruebas se solicita el primer certificado vía interfaz pública.

Entra a la dirección https://ca.uis.edu.co/pub.

User -> Request a certificate -> Request a certificate with automatic browserdetection
Llene todos los campos -> Continue -> Continue ->

Entonces se mostrara una página con todos sus datos, recuerde el número serial de su petición

Ahora entre a la RA con la dirección https://ca.uis.edu.co/ra y logeese de nuevo

Active CSRs -> New -> choose "All" on the Registry Authority combobox & click "Search"

De click sobre el numero del serial del certificado que usted ha solicitado y apruébelo

Click (serial number) -> click "Approve Request without Signing

Se selecciona la opción sin firma pues la opción firmada arroja un error, el resultado de la operación debería ser el siguiente mensaje:
Solicitud de certificado aprobada con éxito

Dentro de la RA dirijase a

General -> Server management -> con login y password root/root. -> Administration -> Dataexchange -> Upload data to a higher level of the hierarchy (Requests) -> click "OK (Revise la salida)

Ahora copie la data sobre su máquina offline y ponga la siguiente dirección en su browser: https://ca.uis.edu.co/ca

General -> Node management -> con login y password root/root. -> Administration -> Dataexchange -> Receive data from a lower level of the hierarchy (Requests) -> click "OK" (Revise la salida)

Dirijase a https://ca.uis.edu.co/ca -> con login y password root/root, entonces:
Usual Operations -> Approved Certificate Requests -> click en el Serial de solicitud certificado -> click "Issue Certificate" -> digite su llave secreta de la CA (revise la salida)

General -> Node management -> con login y password root/root.
-> Administration -> Dataexchange -> Enroll data to a lower level of the hierarchy (Certificates) -> click "OK" (revise la salida)

Copie los datos exportados a su máquina online y vaya a https://ca.uis.edu.co/ra, entonces:

General -> Server management -> con login y password root/root.-> Administration -> Dataexchange -> Download data from a higher level of the hierarchy (Certificates) -> click "OK" (revise la salida)

Vaya a la interfaz pública https://ca.uis.edu.co/pub:

User -> Get Requested Certificate -> provide the Request serial -> click "OK" -

Revise si su certificado está instalado en el browser. Felicitaciones!

ANEXO 2. INSTALACIÓN DE OPEN LDAP

Instalación Base en Servidor

yum -y install openldap openldap-clients openldap-servers authconfig authconfig-gtk

Con fines de organización se creará un directorio específico y se configurará con permisos de acceso exclusivamente al usuario y grupo ldap.

mkdir /var/lib/ldap/autenticar
chmod 700 /var/lib/ldap/autenticar
chown lda:ldap /var/lib/ldap/autenticar

Luego se asigna el password para el usuario administrador del directorio.

slappasswd

Se edita el fichero /etc/openldap/slapd.conf y se verifica que los ficheros de esquema mínimos requeridos estén presentes:

include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/nis.schema

Se edita el fichero /etc/openldap/slapd.conf y se define el nuevo directorio que se utilizará para autenticar los usuarios de la organización:

database bdb
suffix "dc=uis,dc=edu"
rootdn "cn=Administrador,dc=uis,dc=edu"
rootpw XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
directory /var/lib/ldap/autenticar
 # Indices to maintain for this database
index objectClass eq,pres
index ou,cn,mail,surname,givenname eq,pres,sub
index uidNumber,gidNumber,loginShell eq,pres
index uid,memberUid eq,pres,sub
index nisMapName,nisMapEntry eq,pres,sub

Se inicia el servicio LDAP y se agrega para que por defecto inicie con el arranque del sistema en el nivel por defecto.

service ldap start
chkconfig ldap on

Se edita el fichero /usr/share/openldap/migration/migrate_common.ph y se modifican los los valores de las variables $DEFAULT_MAIL_DOMAIN y $DEFAULT_BASE del siguiente modo:

 # Default DNS domain
$DEFAULT_MAIL_DOMAIN = "uis";

```
# Default base
$DEFAULT_BASE = "dc=uis,dc=edu";
```

A continuación hay que crear el objeto que contendrá el resto de los datos en el directorio.

```
/usr/share/openldap/migration/migrate_base.pl > base.ldif
```

Se utilizará ldapadd para insertar los datos necesarios en el directorio así:

```
ldapadd -x -W -D 'cn=Administrador, dc=uis, dc=edu' -h 127.0.0.1 -f base.ldif
```

Una vez hecho lo anterior, se podrá comenzar a poblar el directorio con datos. Una posibilidad es importar los grupos y usuarios existentes en el sistema del siguiente modo:

```
/usr/share/openldap/migration/migrate_group.pl /etc/group group.ldif
/usr/share/openldap/migration/migrate_passwd.pl /etc/passwd passwd.ldif
```

Lo anterior creará los ficheros group.ldif y passwd.ldif, los cuales incluirán la información de los grupos y cuentas en el sistema, incluyendo las claves de acceso. Los datos se podrán insertar en el directorio LDAP utilizando lo siguiente:

```
ldapadd -x -W -D 'cn=Administrador, dc=uis, dc=edu' -h 127.0.0.1 -f group.ldif
ldapadd -x -W -D 'cn=Administrador, dc=uis, dc=edu' -h 127.0.0.1 -f passwd.ldif
```

Después de tener datos de usuarios se deben almacenar los certificados digitales de cada uno de ellos, primero se convierte el certificado .pem a .der, luego se inserta en ldif y finalmente se modifica el usuario, así:

```
openssl x509 -outform DER -in usuario.pem  -out usuario.der
ldif -b "usercertificate;binary" < usuario.der   > cert.ldif
ldapmodify -x -W -D "cn=usuario,dc=uis,dc=edu" -f cert.ldif
```

Donde cert.ldif contiene la siguiente información:

```
dn: cn=usuario,ou=uis,dc=edu,dc=co
changetype: modify
add: usercertificate
usercertificate;binary::
MIIC2TCCAkKgAwIBAgIBADANBgkqhkiG9w0BAQQFADBGMQswCQYDVQQGEwJJVDENMAsGA1
UEChMESU5GTjESMBAGA1UECxMJQXV0aG9yaXR5MRQwEgYDVQQDEwtJTkZIENBICgyKTAeF
w05OTA2MjMxMTE2MDdaFw0wMzA4MDExMTE2MDdaMEYxCzAJBgNVBAYTAkIUMQ0wCwYDVQ
QKEwRJTkZOMRIwEAYDVQQLEwlBdXRob3JpdHkxFDASBgNVBAMTC0lORk4gQ0EgKDIpMIGfMA
0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCrHdRKJsobcjXz/OsGjyq8v73DbggG3JCGrQZ9f1Vm
9RrIWJPwggczqgxwWL6JLPKglxbUjAtUxiZm3fw2kX7FGMUq5JaN/Pk2PT4ExA7bYLnbLGZ9jKJsDh
4bNOKrGRIxRO9Ff+YwmH8EQdoVpSRFbBpNnoDIkHLc4DtzB+B4wwIDAQABo4HWMIHTMAwGA1
UdEwQFMAMBAf8wHQYDVR0OBBYEFK3QjOXGc4j9LqYEYTn9WvSRAcusMG4GA1UdIwRnMGW
AFK3QjOXGc4j9LqYEYTn9WvSRAcusoUqkSDBGMQswCQYDVQQGEwJJVDENMAsGA1UEChME
SU5GTjESMBAGA1UECxMJQXV0aG9yaXR5MRQwEgYDVQQDEwtJTkZOIENBICgyKYIBADALBgN
VHQ8EBAMCAQYwEQYJYIZIAYb4QgEBBAQDAgAHMAkGA1UdEQQCMAAwCQYDVR0SBAIwAD
ANBgkqhkiG9w0BAQQFAAOBgQCDs5b1jmbIYVq2epd5iDjQ109SJ/V7b6DFw2NIl8CWeDPOOjL1E5
M8dnlmCDeTR2TlBxqUZaBBJZPqzFdvxpxqsHC0HfkCXAnUe5MaefFNAH9WbxoB/A2pkXtT6WGWe
d+QsL5wyKJaO4oD9UD5T+x12aGsHcsDCy3EVEaGEOl+/A==
```

Para habilitar el uso de comunicación cifrada entre cliente/servidor es necesario obtener un certificado para el servicio ldap, esto mediante comandos ssl o una entidad certificadora y teniendo esto modificar el fichero openldap.conf para indicar donde se

encuentran el certificado servicio, la llave servicio, certificado de la ca y versión de protocolo SSL

TLSRandFile /dev/random
TLSCipherSuite HIGH:MEDIUM:+SSLv3
TLSCertificateFile /usr/share/openldap/cert/serverldapcrt.pem
TLSCertificateKeyFile /usr/share/openldap/cert/serverldapkey.pem
TLSCACertificateFile /usr/share/openldap/cert/cacert.pem

Antes de configurar el sistema para utilizar LDAP para autenticar, es conveniente verificar que todo funciona correctamente, para esto se listan los directorios disponibles mediante:

ldapsearch -h 127.0.0.1 -x -b '' -s base '(objectclass=)' namingContexts*

Configuración de clientes.

La configuración de LDAP a nivel de cliente es bastante sencilla y basta con usar la herramienta authconfig o authconfig-gtk y definir valores para los parámetros de host y de base a fin de establecer hacia que servidor y a que directorio debe conectarse

#authconfig-gtk
Habilitar con un clic el soporte para LDAP
Escoger la base dc=uis,dc=edu
Escoger el servidor ldap://192.168.0.100

Administración de LDAP.

Para la administración y uso fácil de LDAP es recomendable utilizar un administrador grafico como LDAP Administrator (http://www.ldapadministrator.com), LDAP Browser (http://www.ldapbrowser.com) o Apache Directory Studio (http://directory.apache.org/studio).

ANEXO 3. AUTENTICACIÓN MUTUA UTILIZANDO CERTIFICADOS DIGITALES

A continuación se describen la configuración que se debe realizar en el servidor apache para que funcione con el protocolo de comunicación segura SSL (Secure Socket Layer) y permita la autenticación mutua.

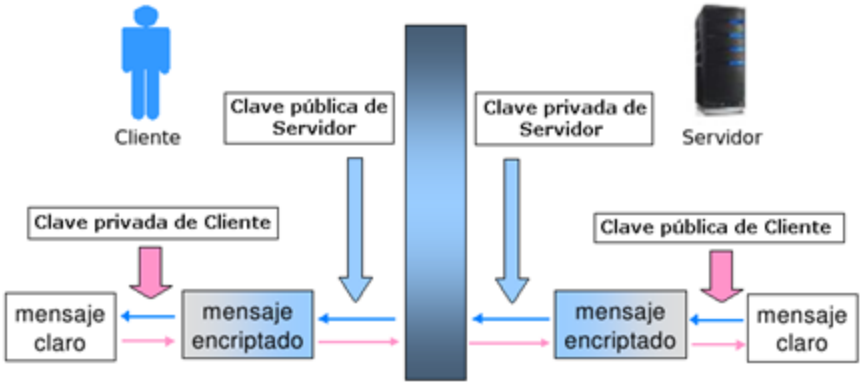

Figura 22. Comunicación Mediante Cifrado Asimétrico.

Los cambios se deben realizar sobre el fichero ssl.conf mediante la inclusión de directivas que indicaran a apache como debe comportarse en cada caso. Inicia el proceso así:

1. El cliente hace una petición al servidor web, por ejemplo: http://www.uis.edu.co, a la cual el servidor contesta enviando su certificado digital e identificándose con el mismo. Para lograr esto es necesario configurar las siguientes directivas:

   Directiva SSLCertificateFile

Descripción:	Especifica la ubicación del certificado de Servidor X.509 codificado en formato PEM
Sintaxis:	SSLCertificateFile *ruta_fichero*

   Esta directiva indica la ubicación del certificado de servidor y opcionalmente las claves privadas RSA o DSA (dentro del mismo archivo). Se puede utilizar hasta dos veces, referenciando a dos nombres de archivo diferentes cuando ambos certificados: RSA y DSA son usados en paralelo.
   Ejemplo: *SSLCertificateFile /etc/httpd/conf.d/certificado_servidor.crt*

   Directiva SSLCertificateKeyFile

Descripción:	Especifica la ubicación de la clave privada de servidor codificada en formato PEM
Sintaxis:	SSLCertificateKeyFile *ruta_fichero*

   Si la clave privada no está combinada con el certificado en el SSLCertificateFile, se usa esta directiva adicional para apuntar al fichero independiente que contiene la

clave privada del servidor. Sin embargo se recomienda tener separadamente el certificado y la clave privada.
Ejemplo: *SSLCertificateKeyFile /etc/httpd/conf/clave_privada_servidor.key*

2. Una vez el servidor envía su certificado y el cliente lo acepta, el servidor debe solicitarle al cliente que él también se identifique y envié su certificado digital, para esto se debe configurar la siguiente directiva:

Directiva SSLVerifyClient

Descripción:	Tipo de verificación del certificado de cliente
Sintaxis:	SSLVerifyClient *nivel*
Por defecto:	SSLVerifyClient none

Esta directiva establece el nivel de verificación para la autenticación del cliente y fuerza a SSL a una renegociación con el cliente y su configuración en la profundidad de verificación después que la petición http es leída pero antes de que la respuesta de http sea enviada.
Los niveles disponibles son los siguientes*:*

- None: no se requiere certificado de cliente.
- Optional: el cliente *puede* o no presentar un certificado válido
- Require: el cliente *tiene que* presentar un certificado válido para su acceso.
- Optional_no_ca: el cliente puede presentar un certificado válido pero no necesariamente ha de ser verificable con éxito.

Ejemplo: *SSLVerifyClient require*

3. El cliente ha de seleccionar de su browser el certificado digital que desea presentar al servidor y enviarlo, una vez lo reciba el servidor debe verificar que el certificado pertenezca a una autoridad de certificación reconocida y confiable. Para esto existen dos directivas de las cuales solo se hace necesario configurar una, la otra es opcional, ellas son:

Directiva SSLCACertificateFile

Descripción:	Fichero compuesto de certificados de AC concatenados para la autenticación del Cliente
Sintaxis:	SSLCACertificateFile *ruta_fichero*

Esta directiva es como el *"todo en uno",* donde, en un solo fichero se pueden concatenar todos los certificados (en formato PEM) de las Autoridades de Certificación (AC) en las cuales se confía, y con cuyos clientes se puede tratar.
Ejemplo: *SSLCACertificateFile /etc/httpd/conf.d/fichero_certificados_ACs.crt*

Directiva SSLCACertificatePath

Descripción:	Directorio que contiene certificados de AC, codificados en formato PEM para la autenticación del cliente.
Sintaxis:	SSLCACertificatePath *ruta_directorio*

Esta directiva establece el directorio donde se deben guardar los certificados de las Autoridades de Certificación en las cuales se confía y con cuyos clientes trata.
Ejemplo: *SSLCACertificatePath /etc/httpd/conf.d/directorio_certificados_ACs/*

4. Además de verificar que el certificado haya sido emitido por una autoridad certificadora confiable se debe verificar que sea válido, es decir que por uso indebido del certificado por parte del cliente no haya sido revocado por la autoridad de certificación, para esto se configura una o las dos, directivas a continuación:

Directiva SSLCARevocationFile

Descripción:	Fichero que tiene concatenadas las listas de certificados revocados de cada una de las AC en la que se confía.
Sintaxis:	SSLCARevocationFile *ruta-fichero*

Esta directiva establece el *"todo en uno"* donde en un fichero en formato PEM se pueden ensamblar las listas de certificados revocados (CRL) de las Autoridades de Certificación (CA) en las que se confía y con cuyos *clientes* se trata.
Ejemplo: *SSLCARevocationFile /etc/httpd/conf.d/fichero_lists_cert_revocados.crl*

Directiva SSLCARevocationPath

Descripción:	Directorio que contiene las Listas de Certificados Revocados codificadas en formato PEM para usarse como referencia en el proceso de autenticación Cliente
Sintaxis:	SSLCARevocationPath *ruta-directorio*

Esta directiva establece el directorio donde se deben guardar las listas de certificados revocados (CRL) de las Autoridades de Certificación (AC) en las cuales se confía y con cuyos clientes se trata. Estas se utilizan para revocar el certificado de cliente en su proceso de autenticación.
Ejemplo: *SSLCARevocationPath /etc/httpd/conf.d/direct_lists_certs_revocados/*

5. Otro factor que debe tenerse en cuenta es que el servidor debe verificar al nivel de profundidad que se le especifique y este está dado por el número de intermediarios que existen entre la autoridad de certificación y la emisión del cliente. Para esto se configura la directiva:

Directiva SSLVerifyDepth

Descripción:	Profundidad máxima que hay entre el nivel de la CA y el nivel donde se emiten los certificados de los clientes.
Sintaxis:	SSLVerifyDepth *número*
Por defecto:	SSLVerifyDepth 1

Esta directiva establece cuan profundamente mod_ssl debe verificar antes de decidir si un cliente tiene o no un certificado valido. Una profundidad de 0 significa que el certificado es auto-firmado y por tanto es un certificado de CA, una profundidad de 1 significa que el cliente puede fue certificado por una CA conocida, que está directamente en un nivel superior. Ejemplo: *SSLVerifyDepth 10*

6. Finalmente, se puede configurar una directiva que obligue a una conexión segura SSL y otra que solo permita el acceso a clientes que cumplan con características definidas dentro de una expresión booleana, estas son:

Directiva SSLRequireSSL

Descripción:	Permite denegar el acceso cuando no se usa SSL en la solicitud HTTP
Sintaxis:	SSLRequireSSL

Esta directiva prohíbe el acceso al cliente a menos de HTTP sobre SSL esté habilitado. Es útil para evitar errores de configuración. Ejemplo: *SSLRequireSSL*

Directiva SSLRequire

Descripción:	Solo permite el acceso a aquellos clientes cuyas características o variables cumplan con la evaluación de una expresión booleana
Sintaxis:	SSLRequire *expresión*

Esta directiva especifica los requerimientos generales para el acceso de un cliente, es muy poderosa ya que se pueden crear fuertes expresiones booleanas de acuerdo a las variables de entorno ssl que carga el certificado del cliente.

Ejemplo:
```
SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
and %{SSL_CLIENT_S_DN_O} eq "Maat Gknowledge" \
and %{SSL_CLIENT_S_DN_OU} in {"ES", "CO",  "AR", “FR”} \
and %{TIME_YEAR} >= 2005 and %{TIME_WDAY} <= 2015 \
or %{REMOTE_ADDR} =~ m/^192\.168\.0\.[0-9]+$/
```

7. Para aceptar los certificados de clientes que hayan sido emitidos por autoridades de certificación confiables lo primero que se debe hacer es obtener la llave pública de cada AC y almacenarla en un directorio o concatenarla en un fichero especifico para este propósito. A continuación se muestra el proceso:

   Se realiza la conversión de los certificados a formato .pem:

   #openssl x509 -inform der -in ca_confianza_2.crt -out ca_confianza_2.pem
   #openssl x509 -inform der -in ca_confianza_1.cer -out ca_confianza_1.pem

   Se deben copiar los ficheros en un directorio exclusivo creado para este propósito:

   #mkdir /etc/httpd/conf.d/CA_Certs
   #cp ca_confianza_2.pem /etc/httpd/conf.d/CA_Certs/
   #cp ca_confianza_1.pem /etc/httpd/conf.d/CA_Certs/

   Se deben concatenar estos ficheros un uno nuevo de formato PEM:

   #cat /etc/httpd/conf.d/CA_Certs/ ca_confianza_1.pem >> /etc/httpd/conf.d/CA_Certs.pem
   #cat /etc/httpd/conf.d/CA_Certs/ ca_confianza_1.pem >> /etc/httpd/conf.d/CA_Certs.pem

   El fichero de configuración ssl debe quedar de manera similar:

```
<VirtualHost _default_:443>
 ServerName www.uis.edu.co:443

 SSLCertificateFile    /etc/httpd/conf.d/Server_Certs/ServerCert.crt
 SSLCertificateKeyFile /etc/httpd/conf.d/Server_Certs/ServerKey.key

 SSLCACertificateFile  /etc/httpd/conf.d/CA_Certificates.pem
 SSLCARevocationFile   /etc/httpd/conf.d/CRL_Certificates.pem

  <Location /proyecto>
   SSLRequireSSL
```

```
    SSLVerifyClient require
    SSLVerifyDepth  10
    SSLRequire      ( %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
                      and %{SSL_CLIENT_S_DN_O} eq "Universidad Industrial Santander" \
                      and %{SSL_CLIENT_S_DN_OU} in {"AR", "CO",  "BR"} \
                      and %{TIME_YEAR} >= 2010 and %{TIME_WDAY} <= 2015  )\
                      or %{REMOTE_ADDR} =~ m/^192\.168\.0\.[0-9]+$ /
  </Location>

</VirtualHost>
```