INSTITUTO POLITÉCNICO NACIONAL.

Escuela Superior de Ingeniería Mecánica y Eléctrica.

Maestría en Ciencias en Ingeniería de Telecomunicaciones.

“MODELO E INFRAESTRUCTURA DE SEGURIDAD BASADO EN IDENTIFICACIÓN Y AUTENTIFICACIÓN PARA REDES”

T E S I S

Que para obtener el grado de:

Maestro en Ciencias en Ingeniería de Telecomunicaciones

P R E S E N T A:

Ing. Norma Alicia Cruz Guzmán.

DIRECTORES DE TESIS:

Dr. Salvador Álvarez Ballesteros

M. en C. Chadwick Carreto Arellano

México, DF. Diciembre 2011.

SIP-14

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de **México, D. F.** siendo las **15:00** horas del día **2** del mes de **DICIEMBRE** del **2011** se reunieron los miembros de la Comisión Revisora de Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de **ESIME-ZACATENCO** para examinar la tesis titulada:

"MODELO E INFRAESTRUCTURA DE SEGURIDAD BASADO EN IDENTIFICACIÓN Y AUTENTIFICACIÓN PARA REDES"

CRUZ	**GUZMÁN**	**NORMA ALICIA**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

A	1	0	0	4	7	7

aspirante de: **MAESTRO EN CIENCIAS EN INGENIERÍA DE TELECOMUNICACIONES**

Después de intercambiar opiniones, los miembros de la Comisión manifestaron ***APROBAR LA DEFENSA DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores(a) de tesis

DR. SALVADOR ALVAREZ BALLESTEROS

M. en C. CHADWICK CARRETO ARELLANO

Presidente

DR. VLADIMIR KAZAKOV

Segundo Vocal

M. en C. CHADWICK CARRETO ARELLANO

Tercer Vocal

DR. RAÚL CASTILLO PÉREZ

Secretario

DR. HÉCTOR OVIEDO GALDEANO

PRESIDENTE DEL COLEGIO DE PROFESORES

DR. JAIME ROBLES GARCÍA

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de **México D.F.** el día **6** del mes de **Diciembre** del año **2011**, el (la) que suscribe **Norma Alicia Cruz Guzmán** alumno (a) del Programa de **Maestría en Ciencias en Ingeniería de Telecomunicaciones** con número de registro **A100477**, adscrito a **la Sección de Estudios de Posgrado e Investigación de la Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Zacatenco**, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de **Dr. Salvador Álvarez Ballesteros y el M en C. Chadwick Carreto Arellano** y cede los derechos del trabajo intitulado **"Modelo e Infraestructura de Seguridad basado en Identificación y Autentificación para Redes"**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección **nor97301011@yahoo.com.mx**. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Ing. Norma Alicia Cruz Guzmán
Nombre y firma

AGRADECIMIENTOS

Son numerosas las personas a las que debo agradecer por ayudarme en el logro de concluir con la maestría, es demasiado poco, el decir gracias, pero en el fondo de mi ser eternamente les estaré agradecida. Sin embargo, resaltare solo algunas de estas personas sin las cuales no hubiese hecho realidad este sueño anhelado:

Ante todo, a DIOS por darme la vida para lograr esta meta, solo tú sabes el sacrificio que he pasado y mis días y noches de soledad me guiaste con tu luz divina por el camino correcto para no desmayar.

A mis padres por su constante amor inexplicable para mi superación personal, sin ningún interés material han vivido a mi lado cada sentimiento, que expresa mi corazón y sin importarles nuestras diferencias ni mis fallas me han apoyado y eso nunca lo olvidare, porque no todos tenemos la dicha de tener unos padres tan responsables como ustedes y por eso no me cansare nunca de expresarles hoy mañana y siempre pase lo que pase, que los amo con todo mi corazón.

A mis amigos por ofrecerme siempre esa mano amiga en los momentos más difíciles.

A mis amigos y compañeros de Maestría quienes me acompañarón en esta trayectoria de aprendizaje y conocimientos, y con los cuales compartí bellos momentos.

Al Instituto Politécnico Nacional, por ser mi máxima casa de estudio.

A mis asesores de tesis por confiar en mí Dr. Salvador Alvares y M en C. Chadwick Carreto, porque con su guía pude lograr la culminación de este proyecto.

Y no me puedo ir sin antes decirles, que sin ustedes a mi lado no lo hubiera logrado, tantas desveladas sirvieron de algo y aquí está el fruto. Les agradezco a todos ustedes con toda mi alma el haber llegado a mi vida y el compartir momentos agradables y momentos tristes, pero esos momentos son los que nos hacen crecer y valorar a las personas que nos rodean.

Es la hora de partir, la dura y fría hora que la noche sujeta a todo horario
(Pablo Neruda)

DEDICATORIA

Mi tesis la dedico con todo mi amor y cariño.

A ti DIOS

Por ser mí guía espiritual que me conduce siempre hacia el camino del bien y el éxito.

A mis padres

Que me dierón la vida y han estado conmigo en todo momento. Gracias por todo papá y mamá por creer en mí, aunque hemos pasado momentos difíciles siempre han estado apoyándome y brindándome todo su amor, por todo esto les agradezco de todo corazón el que esté a mi lado.

A mis hermanos

Jorge y Nancy para que siempre tengan en cuenta que todo lo que nos propongamos en la vida lo podemos lograr si trabajamos fuerte y continuamente con rectitud, sigan adelante y para que mis éxitos de hoy sean los suyos mañana y siempre. Gracias por ser mis hermanos.

A todos mis amigos

Por todo este tiempo donde he vivido momentos felices y tristes, gracias por ser mis amigos, recuerden que siempre los llevo en mi corazón. En especial a Fabián que nos ha dejado una gran enseñanza de vida y fortaleza.

Chicuelo

Gracias por este tiempo de conocernos y en los cuales hemos compartido tantas cosas, hemos pasado tanto que ahora estás conmigo en este día tan importante para mí, sólo quiero darte las gracias por todo el apoyo que me has dado para continuar y seguir con mí camino, gracias por estar conmigo y recuerda que eres muy importante para mí.

Por último a todas aquellas personas que siempre me apoyaron de forma silenciosa y que siempre confiaron en mí, a mis abuelitos que a pesar de no estar aquí físicamente, sé que su alma sí, tíos, primos, etc.

Norma

ÍNDICE GENERAL

RELACIÓN DE FIGURAS Y TABLAS

OBJETIVOS

OBJETIVO GENERAL

Analizar y diseñar un modelo e infraestructura que identifique y autentique usuarios en redes inalámbricas.

OBJETIVOS PARTICULARES

- Analizar un modelo e infraestructura de identificación y autenticación para redes.
- Diseñar un modelo e infraestructura de identificación y autenticación para redes.
- Probar el modelo e infraestructura de identificación y autenticación para redes.

RESUMEN

En el presente trabajo se describe el desarrollo de un modelo y arquitectura de seguridad para redes basado en identificación y autenticación, de acuerdo al modelo propuesto, los usuarios del entorno podrán acceder de forma transparente, en cualquier momento y en cualquier lugar, mediante diferentes dispositivos a la red.

La implementación del modelo propuesto se lleva a cabo en un entorno educativo dentro del laboratorio de la Maestría de Cómputo Móvil en la Escuela Superior de Cómputo (ESCOM) dentro del Instituto Politécnico Nacional (IPN), donde el modelo es evaluado por los usuarios del entorno, es decir estudiantes, profesores y algunos visitantes bajo una infraestructura de red convergente (con uso de las tecnologías Wi-Fi y Ethernet), esto a través de diferentes dispositivos móviles.

Finalmente se presentan las conclusiones en base a los resultados obtenidos de la evaluación del modelo para determinar si este cumplió con los objetivos que se trazaron desde el principio de la investigación.

ABSTRACT

The present work describes the development of a model and network security architecture based on identification and authentication, according to the proposed model, users can access to the Network environment seamlessly, at anytime and anywhere, using different devices.

The implementation of the proposed model takes place in an educational environment in the Masters of Mobile Computing laboratory in the Escuela Superior de Cómputo (ESCOM) within the Instituto Politécnico Nacional (IPN), where the model is evaluated by the students, professors and some visitors in a converged network infrastructure (using technologies such as Wi-Fi and Ethernet), this through various mobile devices.

Finally, the conclusions based in the results of evaluation of the proposed model are presented to determinate if the principal and specific objectives at the beginning of investigation were accomplished.

INTRODUCCIÓN

En México la credencial para votar expedida por el IFE y la Cédula de Identidad Ciudadana son documentos con validez oficial que acreditan la identificación del individuo, al igual que otros documentos como el pasaporte y la cartilla militar (en el caso masculino) entre otros, pero todos ellos tienen un grado de autenticación, lo cual no garantiza que el portador sea quien dice ser y hasta pueden ser falsificables, y por lo tanto, no certificar que los trámites que se realicen con ellos sean 100% confiables.

La identidad digital de un usuario, es la suma de datos únicos de su persona, como datos personales, académicos o financieros en un medio de transmisión digital.

El problema que se presenta en algunas de ellas, es el creado por la dispersión actual de las distintas partes que forman la identidad digital de un usuario, ya que actualmente el usuario es el responsable de mantener toda esa información de forma coherente y consistente, recordar el par usuario-contraseña para cada uno de los sistemas, y es el mismo quién debe administrarla para asegurarse de que cada sitio mantiene información actualizada.

Por todo lo anterior, la importancia de contar con un sistema de identidad digital confiable y robusto en cuanto a seguridad se refiere, se vuelve cada vez más necesario y sabiendo que la identificación de personas es y será un tema de investigación y desarrollo importante ligado a la constante evolución de las sociedades humanas, en este trabajo de tesis creará una Infraestructura de Identificación y Autentificación Digital de la que se obtendrá una identificación digital con la que se pretende lograr que los sistemas de seguridad sean más confiables.

PLANTEAMIENTO DEL PROBLEMA

Con la llegada de nuevas tecnologías, su utilización y adopción por gran cantidad de personas, surgen amenazas y es por ello que hoy en día es muy importante hablar de seguridad en cuanto a computación y redes se refiere. La seguridad de la información y de los sistemas, es un punto crítico en una sociedad en la que la información electrónica o digital cada vez obtiene más simpatizantes.

La identidad en la sociedad humana juega un rol muy importante y por ende, se entiende también la necesidad de contar con medios de control que brinden el acceso a recursos, instituciones, derechos, obligaciones, etc. de forma segura, tanto para el portador de la identificación, como del prestador de servicios o instancia que requiere de tal identificación.

Cuando se busca cubrir esta necesidad, se suele recurrir a la creación de identificaciones con la impresión de micas con ciertos elementos de seguridad como marca de agua, holograma bidimensional, firma digitalizada, entre otros. El inconveniente que existe en ellas, es que no te permiten identificarte en múltiples organizaciones y se tiene que portar todas las identificaciones con las que se cuenta y en caso de robo o extravió de cartera, tener que ir a cada organización para solicitar identificaciones nuevas.

La problemática entonces, no es el simple hecho de la identificación, si no la posibilidad de extender sus capacidades en cantidad de información, portabilidad, seguridad y generalización y hacer uso de los avances tecnológicos actuales, esto bajo una buena propuesta de infraestructura que permita identificar al usuario.

En cuanto a seguridad se refiere, es necesario verificar que el portador de la identificación es quien dice ser y para eso, existen diferentes tecnologías que apoyan el mecanismo de autenticación de usuarios tales como certificados digitales, claves de acceso/password, tarjetas inteligentes, tokens de seguridad, biométricos entre otros.

JUSTIFICACIÓN

La solución propuesta a tal problemática y como mejora de las soluciones existentes, es la creación de una Infraestructura que al integrar diversas tecnologías para la identificación y la autentificación antes mencionadas, permita verificar la identidad y la autenticidad de una persona de manera segura garantizando el no repudio.

Las principales características que se tienen son:

- Sencillez: Al eliminar la complejidad de uso de sus interfaces y sea apto para cualquier tipo de usuario.

- Robustez: Considerando el número de variables, contener suficiente información con la seguridad adecuada para evitar la violación de datos, garantizando seguridad recíproca entre el servidor y el usuario.

- Movilidad: Obtiene mayor portabilidad y sencillez.

Dado lo anterior, se vuelve importante la creación de un modelo e infraestructura que al integrar diversas tecnologías para la identificación y la autenticación antes mencionadas, permita verificar la identidad y la autenticidad de una persona de manera segura.

La propuesta del sistema nos brinda un alto grado de seguridad en comparación a otros comerciales ya que no solo tiene la seguridad mediante un usuario y contraseña sino que además cuenta con certificados que nos ofrecen una mayor seguridad lo cual hace que sea aún más confiable.

Utilizamos autenticación EAP-TLS por ser el más robusto de todos los métodos de autenticación que implementa EAP.

Una de las fortalezas con las que cuenta el sistema es que es difícil violar la seguridad por la doble seguridad, se podría intentar ingresar teniendo de alguna manera el nombre de usuario y la contraseña correctas pero no se completaría el acceso por que los certificados con únicos por usuario y esto es lo que hace que el sistema no sea vulnerable.

A continuación, en el Capítulo 1 se exhiben los antecedentes para el desarrollo del modelo planteado; en el 2 se describe el modelo propuesto de seguridad en el 3 se desarrolla el caso de estudio de un sistema de red, en el 4 el desarrollo del sistema, en el 5 se muestran las pruebas realizadas. Finalmente, en el Capítulo 6 se muestran las conclusiones de la propuesta y se establecen ideas para trabajos a futuro.

CAPÍTULO 1
SEGURIDAD Y AUTENTICACIÓN

1.1 ¿Que es la seguridad?

Definir el concepto de seguridad no es algo fácil, pero podemos empezar a partir de la frase citada por el Dr. Eugene Spafford: “El único sistema totalmente seguro es aquel que está apagado, desconectado, guardado en una caja fuerte de titanio, encerrado en un bunker de concreto y cuidado por guardias muy bien armados... y aún así tengo mis dudas” [1]. Esta cita parece no llevarnos a ningún lado ni mucho menos aclarar el concepto, pero nos sirve para percatarnos de tres cosas: no podemos hablar de sistemas totalmente seguros, lo complicado que es el tratar de asegurar un sistema y, que esto conlleva un proceso sumamente complejo.

Se puede decir que la seguridad consiste en que un sistema se comporte como el usuario espera que lo haga, y a su vez mantenerlo libre de amenazas y riesgos [2]. Por más de dos décadas se ha manejado que la seguridad se logra a partir de tres conceptos, conocidos como la triada de la seguridad: confidencialidad, integridad y disponibilidad (Figura 1.1) [3][4].

En la actualidad esta percepción ha sido sustituida por los seis elementos atómicos (más importantes) de la información: confidencialidad, posesión o control, integridad, autenticidad, disponibilidad y utilidad [5].

Figura 1.1 Triada de la seguridad

1.1.1. Confidencialidad

Consiste en mantener la información secreta a todos, excepto a aquellos que tienen autorización para verla [6].

Cuando la información de naturaleza confidencial ha sido accedida, usada, copiada o revelada a, o por una persona que no estaba autorizada, entonces se presenta una ruptura de confidencialidad. La confidencialidad es un requisito para mantener la privacidad de las personas [7].

1.1.2. Integridad

Significa que se debe asegurar que la información no ha sido alterada por medios no autorizados o desconocidos [6]. Un atacante no debe ser capaz de sustituir información legítima por falsa [8].

1.1.3. Autenticidad

Debe ser posible para un usuario establecer el origen de la información. Un atacante no debe tener la capacidad de hacerse pasar por otro usuario [8].

1.1.4. Posesión

Es el mantener, controlar y tener la habilidad de usar la información. La posesión es la habilidad de realmente poseer y controlar la información y cómo es usada [9]. Un ejemplo de pérdida de posesión es un empleado que envía información corporativa a una cuenta de correo electrónico que no pertenece a la compañía, en ese momento se pierde la posesión exclusiva.

1.1.5. Disponibilidad

Significa que todos aquellos elementos que sirven para el procesamiento de la información, así como los que sirven para facilitar la seguridad, estén activos y sean alcanzables siempre que se requiera. Dicha característica puede perderse a través de ataques de DoS (Denial of Service -Denegación de Servicio-). [1].

1.1.6. Utilidad

Utilidad de la información para un propósito. El valor de la información recae en su utilidad [9]. Por ejemplo, un archivo cifrado deja de ser útil si no se cuenta con las llaves necesarias para descifrarlo.

1.2. Elementos a proteger

Los elementos que conforman un sistema informático se dividen en tres distintas categorías: hardware, software y datos. Lo que más nos interesa proteger son los datos, ya que estos son los activos que poseen tanto empresas como usuarios, estos datos son los más amenazados, y normalmente es mucho más difícil recuperarlos en comparación con el hardware y el software [4].

Al asegurar los datos estamos asegurando la continuidad de operación de una organización, reduciendo al mínimo los daños causados por una contingencia y manteniendo a salvo nuestra persona, familia o patrimonio.

1.3. Amenazas a la seguridad

Existe un gran número de amenazas que pueden afectar la seguridad de un sistema en sus distintas categorías, las cuales se pueden clasificar en cuatro grandes grupos: interceptación, interrupción, modificación y fabricación [9]. Estos ataques afectan a uno o más de los elementos atómicos de la información.

1.3.1. Interceptación

Estos ataques propician que una entidad no autorizada gane acceso a un elemento del sistema (Figura 1.2).

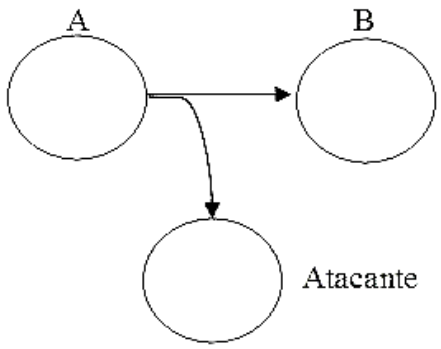

Figura 1.2 Ataque de Interceptación

1.3.2. Interrupción

Se dice que un ataque es una interrupción si se logra que un elemento se pierda, quede inutilizable o no disponible (Figura 1.3).

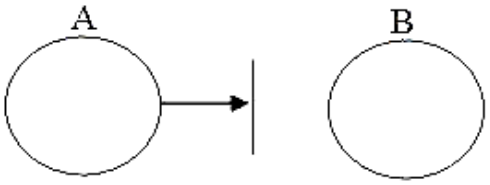

Figura 1.3 Ataque de interrupción

1.3.3. Modificación

Este ataque se da después de una interceptación, cuando el oponente logra modificar el elemento (Figura 1.4). Esto es lo que se envía de A al atacante es diferente a lo que recibe B.

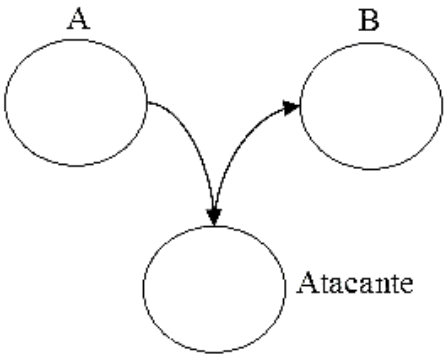

Figura 1.4 Ataque de modificación

1.3.4. Fabricación

Se habla de una fabricación cuando el atacante logra crear un objeto el cual es difícil de distinguir si es un elemento genuino (Figura 1.5).

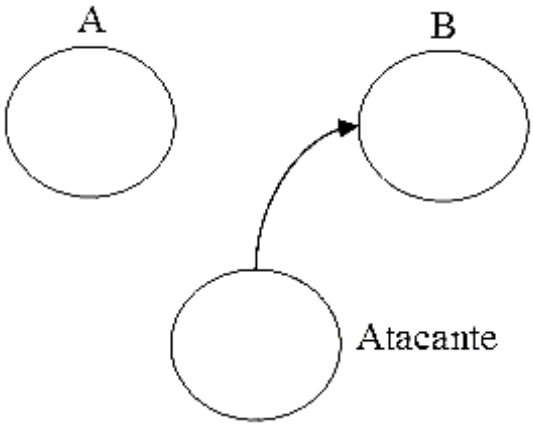

Figura 1.5 Ataque de fabricación

1.4. Autenticación de usuarios

La palabra autenticación proviene del griego '$\propto \nu\theta \in \gamma\tau\iota\kappa o\varsigma$'que significa verdadero o genuino, y authentes que significa el autor o dueño [10]. Es la acción de establecer o comprobar si una entidad es auténtica [11].

En el caso de este trabajo estaremos hablando de autenticar personas ante un sistema computacional, es decir, comprobar la identidad digital de un usuario para que pueda ganar acceso a un recurso del sistema.

La autenticación es la base para los controles de acceso, ya que los sistemas deben ser capaces de identificar y diferenciar entre diversos usuarios [12].

Conviene hacer la distinción entre los términos identificación y autenticación, ya que con frecuencia se suelen confundir y usar ambos para hacer referencia a la misma acción. Identificación es el medio que un usuario proporciona a un sistema para afirmar una identidad, mientras que autenticación es el medio para establecer la validez de esta afirmación [12]. Para aclarar un poco más esta diferencia pensemos en el esquema de control de acceso a través

de ID y contraseña, el usuario se identifica con su ID, el cual es conocido por el administrador del sistema e incluso por otros usuarios, pero la manera en que se autentifica es con su contraseña secreta. De esta forma podemos ver claramente cómo el sistema reconoce a las personas basándose en los datos de autenticación que recibe.

Existen tres métodos a través de los cuales un usuario puede autenticarse: algo que el usuario posee, algo que el usuario conoce y algo que el usuario es [12]. Cada método presenta sus propias ventajas y desventajas, por lo cual es aconsejable implementar controles de acceso multifactor, esto es combinando los grupos mencionados [13].

A pesar de las divisiones de mecanismos de autenticación que existen, todos ellos siguen un proceso bastante general:

1. El usuario pide acceso a un recurso.
2. El sistema le solicita al usuario su medio de autenticación.
3. El usuario entrega sus credenciales de autenticación.
4. El sistema verifica las credenciales del usuario.
5. El sistema niega o proporciona al usuario el acceso al recurso.

1.5. Métodos de autenticación de usuario

El objetivo de la autenticación de usuario es permitirle a una persona autorizada el acceso a un recurso físico, telemático o informático, previa verificación de que cumple con las condiciones exigidas para dicho acceso.

Los métodos de autenticación de usuario que existen hoy en día son muy variados. Una forma de clasificarlos es de acuerdo a su relación con el usuario y estos pueden ser:

- Aquellos que se basan en datos conocidos por el usuario,

- Los que requieren que el usuario lleve un dispositivo,
- Y finalmente los métodos biométricos que se basan en rasgos físicos o en patrones de comportamiento del usuario.

1.5.1. Datos conocidos por el usuario

En esta categoría están las contraseñas usadas para el acceso a recursos informáticos, generalmente con un nombre de usuario asociado, y los PINs o NIPs (Número de Identificación Personal).

Una contraseña o clave (en inglés *password*) es una forma de autentificación que utiliza información secreta para controlar el acceso hacia algún recurso. La contraseña normalmente debe mantenerse en secreto ante aquellos a quienes no se les permite el acceso. A aquellos que desean acceder a la información se les solicita una clave; si conocen o no conocen la contraseña, se concede o se niega el acceso a la información según sea el caso.

El uso de contraseñas se remonta a la antigüedad: los centinelas que vigilaban una posición solicitaban el «santo y seña» al que quisiera pasar. Solamente le permiten el acceso a aquella persona que conoce la seña. En la era tecnológica, las contraseñas son usadas comúnmente para controlar el acceso a sistemas operativos de computadoras protegidas, teléfonos celulares, decodificadores de TV por cable, cajeros automáticos de efectivo, etc. Un típico ordenador puede hacer uso de contraseñas para diferentes propósitos, incluyendo conexiones a cuentas de usuario, accediendo al correo electrónico (e-mail) de los servidores, accediendo a bases de datos, redes, y páginas Web, e incluso para leer noticias en los periódicos (diarios) electrónicos.

En la lengua inglesa se tienen dos denominaciones distintivas para las contraseñas: password (palabra de acceso) y pass code (código de acceso), donde la primera no implica necesariamente usar alguna palabra existente (sin embargo, es normal emplear alguna palabra familiar o de fácil memorización por parte del usuario), la primera suele asociarse también al uso de códigos alfanuméricos (también llamado PIT - *Personal Identification Text*), mientras que la segunda frecuentemente se liga a la utilización de algún código numérico

(asimismo llamado PIN - *Personal Identification Number*). Esto ocurre igualmente en el habla española, ya que en ocasiones clave y contraseña se usan indistintamente.

1.5.2. Balance entre seguridad y comodidad

Para el control de acceso total, se realiza una relación entre seguridad y comodidad para evitar que alguien extraño tenga acceso a ciertos recursos. Es decir, si algún recurso está protegido por una contraseña, entonces la seguridad se incrementa con el consecuente aumento de molestia para los usuarios. El nivel de seguridad es inherente dada una política de contraseñas en particular, que está influida por diversos factores. Sin embargo, no existe un método único que sea el mejor para definir un balance adecuado entre seguridad y comodidad de acceso.

Algunos sistemas protegidos por contraseñas plantean pocos o ningún riesgo a los usuarios si éstos se revelan, por ejemplo, una contraseña que permita el acceso a la información de un sitio Web gratuita. Otros plantean un modesto riesgo económico o de privacidad, por ejemplo, una contraseña utilizada para acceder al e-mail, o alguna contraseña para algún teléfono celular. Aun así, en otras situaciones, puede tener consecuencias severas si la contraseña es revelada. Por ejemplo, como las situaciones para limitar el acceso de expedientes sobre tratamientos del sida o el control de estaciones de energía.

1.5.3. Dispositivos

Los dispositivos incluyen las tarjetas plásticas de banda magnética, las tarjetas inteligentes y los tokens o módulos de seguridad, entre otros. Proporcionan una mayor seguridad que el método anterior, pero siempre y cuando sea el usuario autorizado quien las tenga en su poder.

a) **TOKENS:** Son dispositivos especiales denominados tokens de seguridad diseñados para facilitar la integridad y seguridad, así como evitar que ésta pueda ser exportada (Figura 1.6).

Fig.1.6 TOKENS

b) T**arjeta inteligente** (**smart card**), o tarjeta con circuito integrado (TCI), es cualquier tarjeta del tamaño de un bolsillo con circuitos integrados que permiten la ejecución de cierta lógica programada (Figura 1.7).

Aunque existe un diverso rango de aplicaciones, hay dos categorías principales de TCI.

- Las Tarjetas de memoria contienen sólo componentes de memoria no volátil y posiblemente alguna lógica de seguridad.
- Las tarjetas microprocesadoras contienen memoria y microprocesadores.

Figura 1.7. Tarjeta inteligente

1.5.4. Sistemas Biométricos

Los métodos anteriores verifican si un usuario está o no autorizado para tener acceso a un recurso pero no nos dicen nada relacionado con su identidad. Los métodos biométricos verifican la identidad del usuario, y con base en esta

identificación proporcionan acceso a los recursos autorizados para ese usuario en particular. Podemos dividir los métodos biométricos en:

Los que se basan en rasgos físicos del usuario según Tabla 1.1.

Y aquellos basados en patrones de comportamiento del usuario según Tabla 1.1.

Tabla 1.1. Métodos biométricos

Rasgos físicos	**Escaneo de retina** **Reconocimiento de huella digital** **Reconocimiento de iris** **Reconocimiento de la cara** **Geometría de la mano** **Patrón de venas** **Análisis de DNA**
Comportamiento	Análisis de firma Reconocimiento de voz Ritmo de uso del teclado

El escaneo de retina es de los métodos biométricos más antiguos. Se originó en investigaciones realizadas en los años 30. Hoy en día no es muy usado debido a que muchos consideran que es invasivo y que viola la privacidad: el usuario se debe situar el ojo a uno o dos centímetros del escáner y mirar una luz verde mientras se lee el patrón de vasos sanguíneos del fondo de su ojo. Además de lo molesto del procedimiento, este escaneo puede revelar datos adicionales a la identidad, como por ejemplo la existencia de un embarazo.

De los demás métodos, el sistema de reconocimiento de huellas digitales es el más popular en la actualidad, y le sigue el de reconocimiento de iris. Ambos métodos ofrecen un alto grado de seguridad.

Es posible que el análisis de DNA se use ampliamente en un futuro como método de autenticación, aunque hoy en día se emplea principalmente para investigaciones forenses y pruebas de paternidad.

Entre los métodos basados en patrones de comportamiento del usuario, el reconocimiento de voz es por ahora el menos confiable. El método de reconocimiento de firma no solo analiza la firma como tal sino la presión empleada en cada trazo de la misma. Los métodos biométricos tienden a generalizarse, usados en combinación con otros métodos de autenticación.

1.6. Estándar 802.11

Un estándar es una norma o requerimiento propuesto por una o varias entidades para promover una facilidad de diseño, construcción, homogeneidad y compatibilidad de los dispositivos [14]. Por esto, un estándar ha de ser ampliamente conocido, e incluso público, y estará sometido a un uso continuo y masivo. Debido a ello, es posible que se descubran debilidades de diseño en los estándares, aún disponiendo de sistemas de seguridad. Estas debilidades no suelen ser ni fáciles ni rápidas de solucionar, con lo que los atacantes, poco a poco, perfeccionan sus ataques y los hacen más potentes y certeros.

Todas las versiones activas del 802.11 tienen compatibilidad descendente, pero cada una de ellas añade funcionalidades que versiones anteriores no podrán utilizar. Aún así, las versiones tienen la base común del el estándar IEEE 802.11-1997, la mayoría de las cuales afectan a la capa física o capa PHY (Physical Layer) [14].

Las versiones más relevantes de cara al usuario son las mostradas en la Figura 1.8. Las versiones que modifican la capa física (PHY) suelen estar orientadas a la mejora de la velocidad de transmisión. Las versiones en la capa MAC (Control de Acceso al medio, Media Access Control) tienen que ver con la seguridad en la transmisión de datos [14].

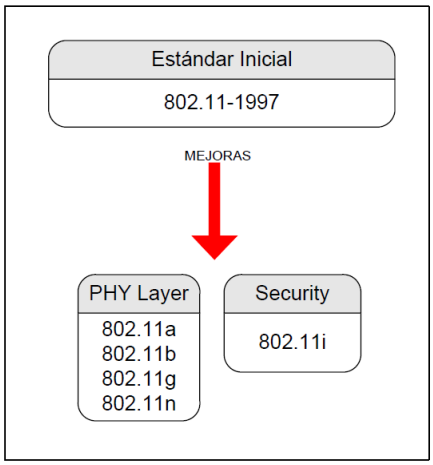

Figura 1.8. Actualizaciones del 802.11

1.6.1. Capas del protocolo

El estándar 802.11 sólo cubre las dos primeras capas de la arquitectura OSI. En concreto son la capa PHY y la subcapa MAC (parte de la capa Data Link Layer o DLL). En el estándar en su edición inicial de 1997, se propone una única capa MAC y tres posibles capas físicas. Estas posibles capas físicas son [15]:

- IR (Infrarrojos): que nunca ha sido implementado de manera comercial.
- FHSS (espectro ensanchado por salto de frecuencia, Frequency Hoping Spread Spectrum): Usa la frecuencia de 2,4 GHz.

Esta capa fue en la que se centraron los primeros estudios, ya que la electrónica que usaba era relativamente barata. Además este tipo de modulación permitía la coexistencia de varias redes en un mismo lugar manteniendo niveles relativamente altos de transmisión de datos. Aún así, en la versión de 1999 se introdujo DSSS (Direct Sequence Spread Spectrum), con lo que el desarrollo y producción de dispositivos usando FHSS propuesto en 1997 no fue llevado a término.

- DSSS: Usa la frecuencia 2,4 GHz. Mediante el uso de DSSS se consiguen velocidades de 1 ó 2 Mbps, operando en la misma

frecuencia que FHSS. La tecnología basada en DSSS fue mejorando y desplazó definitivamente a la FHSS (como se observa en la mejora 802.11b).

La definición de capas MAC y PHY específicas por parte del estándar 802.11, no sólo posibilita la implementación de dispositivos sencillos para realizar el encaminamiento o puente (bridging) entre redes, sino que también permite a las redes inalámbricas realizar control de errores optimizados para este tipo de comunicación. Se pretende así, que los errores producidos por la transmisión inalámbrica no lleguen a capas superiores del protocolo.

De no ser así, deberíamos llegar a niveles superiores del modelo OSI para corregir los errores, con lo que la retransmisión de los paquetes se haría de manera más costosa, tanto en ancho de banda como en tiempo de proceso. Esto podría causar una sobrecarga de retransmisiones influyendo de manera drástica en la eficiencia de estas redes [16].

Por lo tanto, como podemos ver en la Figura 1.9, el estándar 802.11 y las redes de la familia 802 tienen en común todas las capas del modelo OSI, a excepción de la capa MAC y la capa PHY, que en el caso de las redes 802.11 están optimizadas para la transmisión inalámbrica.

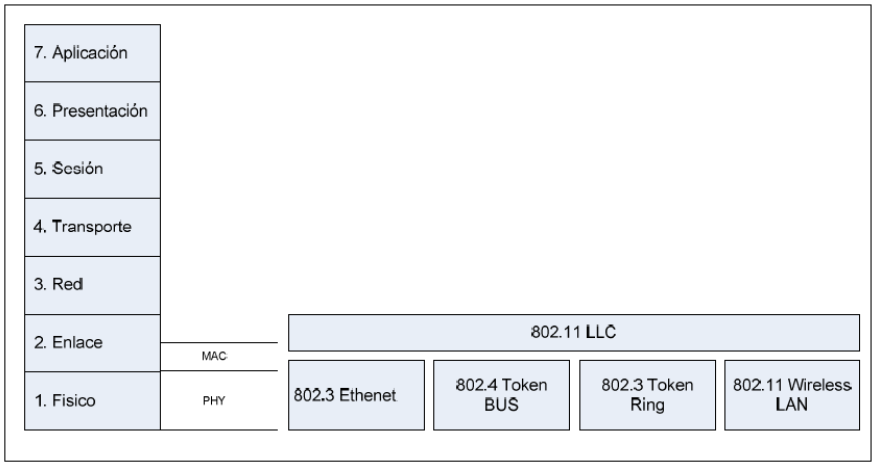

Figura 1.9. Diferencias entre protocolos de la familia 802 según el modelo OSI

1.7. Seguridad en el Estándar 802.11

La importancia de la seguridad ha ido creciendo a medida que estas redes se han ido popularizando. La preocupación por la seguridad en las comunicaciones inalámbricas se puede apreciar especialmente en las redes propiedades de empresas. Para una empresa, tener un punto de acceso con seguridad débil es lo mismo que tener un punto de red *Ethernet* fuera de los límites de su edificio.

El estándar *802.11* se diseñó con el sistema de seguridad *WEP*, donde los datos se cifran mediante el algoritmo *RC4*. Al observar diversas debilidades en dicho algoritmo se propuso la mejora *802.11i*. En la Figura 1.10 podemos ver la organización definitiva de los mecanismos de seguridad utilizados por el estándar en la actualidad.

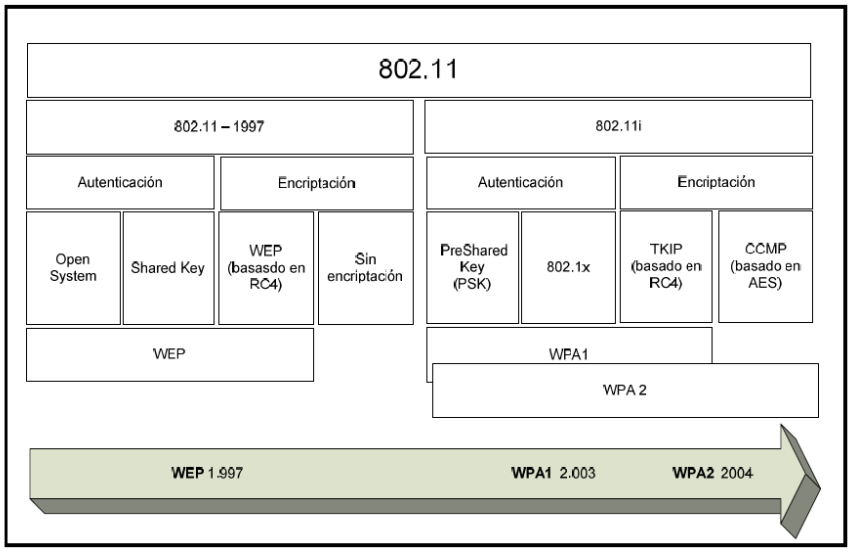

Figura 1.10. Mecanismos de seguridad de 802.11

Los mecanismos mostrados en la Figura 1.10 son los aceptados por el estándar, aunque hay soluciones propietarias ampliamente aceptadas. Por ejemplo, al reconocerse la debilidad en WEP por el uso masivo de la clave de cifrado, se desarrolló WEP dinámico, el cual cambia la clave WEP periódicamente [14]. Además, WEP dinámico posibilitaba la autenticación mediante 802.1x con lo

que permitía separar la parte de la autenticación y la parte del cifrado. Aún así WEP dinámico mantenía el RC4 para el cifrado de los datos.

Para resolver estos problemas, el IEEE propuso el 802.11i. Pero la aparición de esta mejora se fue aplazando ya que se quería usar el algoritmo de cifrado AES, el cual en ese momento no estaba completamente desarrollado. Por este motivo la organización y por querer que los dispositivos anteriores fueran compatibles, Wi-Fi Alliance decidió la certificación del mecanismo de seguridad llamado WPA (Wi-Fi Protected Access) basado en el tercer draft del 802.11i. WPA proporciona varios métodos de autenticación y mejora el cifrado usando claves temporales o TKIP (protocolo de integridad para una llave temporal, Temporal Key Integrity Protocol). Como TKIP utiliza el algoritmo RC4, le permite a WPA ser compatible con la mayoría de dispositivos anteriores simplemente con una actualización del firmware.

Finalmente, en 2004, una vez implementado el nuevo mecanismo llamado CCMP basado en el cifrado AES, apareció el estándar 802.11i. En este estándar se confirmó WPA (que usa el cifrado TKIP basado en RC4) y se incorporó WPA2 que utiliza CCMP (Counter-mode/CBC-MAC Protocol) basado en AES (Advanced Encrytion System). Con la inclusión de estos dos mecanismos de cifrado en el estándar final se promueve la compatibilidad (en el caso de CCMP es necesario que todos los dispositivos lo soporten, de no ser así, deberá ser usado TKIP).

1.7.1. Seguridad WEP

WEP (Wireless Equivalent Privacy) se diseñó para intentar ofrecer al usuario los servicios de autenticación, integridad y confidencialidad. Pero estos objetivos no se vieron cumplidos debido a fallos tanto en el concepto, como en el mecanismo usado para el cifrado. WEP es actualmente el método de seguridad más extendido, sobre todo debido a que los proveedores de Internet ofrecen routers inalámbricos configurados con este sistema.

1.7.2. Seguridad WPA

El mecanismo propuesto por el estándar, *WEP*, tiene debilidades muy importantes en el mecanismo de autenticación y de cifrado. La autenticación puede mejorarse ya que el estándar permite usar métodos robustos de autenticación como *EAP*. En el caso del cifrado, para mejorarlo sería necesario el rediseño de la capa de enlace (*LLC*) del estándar.

Dadas las necesidades del mercado, *Wi-Fi Alliance* propuso dos nuevas versiones que mejoran la seguridad del estándar basadas en *802.11i*: *WPA* y el *WPA2* respectivamente.

Recordamos que la diferencia entre estos dos protocolos está en el cifrado que usan:

WPA usa *TKIP* basado en *RC4* (compatible con *WEP*) y *WPA2* utiliza *CCMP* basado en *AES*.

Como curiosidad y para evitar confusiones, hemos de decir que *WPA* no es el nombre de un estándar oficial, sino que se estableció así para identificar y popularizar las mejoras en la seguridad basadas en *802.11i* [14]. Aún así esta división en pseudo-estándares nos es útil para comprender el funcionamiento y la seguridad ofrecida por cada uno de ellos. En la Figura 1.10 que anteriormente hemos mostrado, podíamos ver la división de estos nuevos mecanismos.

Seguidamente comentaremos los nuevos métodos de cifrado que *WPA* propone estableciendo las diferencias entre ellos y mostrando cómo mejoran al protocolo *WEP*.

También introduciremos la manera de funcionar de estos métodos y diferentes configuraciones de seguridad posibles usando *WPA*.

1.7.3. Protocolo TKIP

El protocolo *TKIP* fue la primera mejora a nivel de enlace. *TKIP* mejora la seguridad que ofrece *WEP*, pero sin necesidad de tener que cambiar el hardware (sólo es necesaria una actualización a nivel de *firmware*). La compatibilidad es

posible ya que el protocolo *TKIP* utiliza del algoritmo de cifrado usado por *WEP*, el *RC4*.

1.7.4. Modos de funcionamiento

WPA tiene 2 modos de funcionamiento de cara al usuario:

1. Personal: el usuario debe compartir una contraseña con el AP y con todos los demás usuarios de la red. A diferencia de WEP, donde el secreto compartido (shared secret) era de tamaño fijo, en WPA la contraseña de frase (passphrase) puede tener entre 8 y 63 caracteres.
2. Enterprise: en este caso se usa algún método externo (como puede ser alguno de los métodos EAP) para la autenticación. Este modo es más seguro que WPA personal porque este sistema puede autenticar por usuario, no por máquina (cada usuario tiene su identificador no compartido). Este caso necesita un servidor RADIUS de autenticación.

1.7.5. Protocolo CCMP

El protocolo CCMP (modo de contador con CBC-MAC, Counter Mode with CBC-MAC Protocol) es el protocolo diseñado para utilizar el algoritmo de cifrado AES en 802.11. CCMP usa el CCM (modo de contador con CBC-MAC, Counter Mode with CBC-MAC) que está definido en el RFC 3610.

1.8. Autenticación mediante 802.1x

El estándar inicial no incorporaba un mecanismo robusto de autenticación. En las siguientes mejoras se incluyó la posibilidad de hacer servir el estándar 802.1x el cual proporciona varios métodos que podemos considerar seguros para redes inalámbricas.

El estándar 802.1x no está relacionado directamente con el desarrollo del estándar 802.11, sino que es un conjunto de especificaciones totalmente independientes al estándar inalámbrico.

Este estándar nació de la necesidad de disponer de un método de autenticación seguro, flexible y optimizado para redes IP tanto LAN como WAN, el cual funcionara sobre la capa LLC [17]. El estándar 802.1x utiliza el protocolo EAP para la comunicación y el concepto de puertos de autenticación. Los dispositivos que quieren conectar a una red basada en este estándar realizan una primera conexión con el dispositivo que actúa como puente entre el cliente y el servidor de autenticación (en nuestro caso este puente es el AP). Los puertos de conexión entre la red y el cliente pueden estar autorizados o desautorizados, dependiendo de la validación del servidor de autenticación.

En la Figura 1.11 podemos observar la arquitectura de un sistema 802.1x y la nomenclatura usada en este sistema, donde:

- **Suplicante:** cliente que quiere ser autenticado.
- **Autenticador:** es el que provee al cliente del acceso a la red. Este acceso primero será sólo mediante el puerto de autenticación el cual sólo permitirá mensajes de autenticación. Si la autenticación es válida, se le asignará un puerto abierto exclusivo.
- **Servidor de autenticación:** la decisión de la concesión de acceso a un usuario la realiza este servidor, el cual indica al autenticador si debe asignar un puerto válido o debe denegar la petición. Este servidor puede estar en cualquier red y contendrá la información necesaria para dar validez al usuario según el tipo de EAP utilizado.

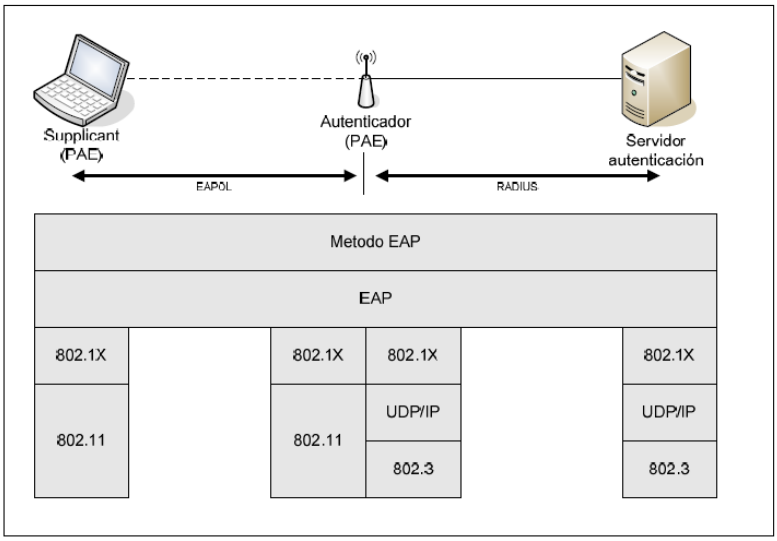

Figura 1.11. Comunicación EAP y pila de protocolos usados

El suplicante y el autenticador son los elementos que funcionan mediante el método de puertos, por eso son llamados elementos PAE (Puerto de Entidades de autenticación, Port Authentication Entities). Estas entidades se pueden comunicar mediante mensajes EAPOL (EAP sobre LAN) aún cuando el cliente no tenga ni dirección IP ni tenga un puerto autorizado (sólo se permitirán los mensajes de inicio de autenticación). El autenticador y el servidor de autenticación se comunican mediante mensajes RADIUS, ya que RADIUS proporciona una gran variedad de métodos de autenticación, flexibilidad y compatibilidad con muchas de las bases de datos de usuarios [14].

En definitiva, este sistema de autenticación se incorporó al estándar 802.11 el cual intenta eliminar una de las dos debilidades más importantes de WEP en este campo. El punto fuerte del 802.1x es que evita el paso de cualquier tipo de tráfico hacia la red si el usuario no ha sido previamente autorizado. El usuario no autenticado sólo puede conseguir enviar paquetes del tipo EAPOL al puente para que éste los envíe al servidor de autenticación.

1.8.1. EAP

EAP (Protocolo de autenticación Extensible, Extensible Authentication Protocol) nació de la necesidad de tener un mecanismo de autenticación más

robusto para las conexiones PPP (Punto a Punto, Point-to-Point). El EAP está definido en RFC 3748 y al ser “Extensible” (o tener estatus de framework) se le permite que pueda ser modificado para incluir nuevos métodos de autenticación (como en su momento hizo CISCO Systems con la autenticación LEAP).

En el caso de las transmisiones inalámbricas, el *EAP* es utilizado por el *802.1x* para realizar la comunicación entre los dispositivos que intervienen en la autenticación del cliente.

A continuación describiremos los métodos de autenticación más importantes para el desarrollo de este documento. Estos métodos han de cumplir con lo dicho en la introducción del proyecto: han de tener una autenticación veraz y han de proporcionar seguridad sobre sus mensajes. Es por eso que sólo trataremos los métodos de autenticación criptográficos dejando de lado los que usan una criptografía débil como pueden ser el EAP-MSCHAP (versión 1 y 2), EAP-AKA, CHAP, MD-5 entre otros.

1.8.2. LEAP

LEAP (*Lightweight Extensible Authentication Protocol*) fue diseñado por la compañía *CISCO Systems* con tal de proveer a sus sistemas de una autenticación robusta con *WEP* y a la vez sencilla de implementar.

Este sistema, dado a su carácter propietario, sólo puede hacerse servir en dispositivos *CISCO* y en algunos compatibles con este. Este sistema usa un servidor *RADIUS*, el cual también ha de ser de *CISCO*. Con esto se limita la agilidad del sistema dada la obligación de usar los dispositivos del fabricante.

Con el tiempo el sistema *LEAP* se ha visto debilitado a causa de que está basado en el sistema de autenticación *CHAP* (el cual es inseguro). Aprovechando ataques a *MSCHAP* se diseñaron diversos ataques de diccionario los cuales se demostró que *LEAP* era vulnerable [18].

LEAP actualmente es desaconsejado por el propio fabricante *CISCO* y es importante conocerlo debido a que fue el primer paso de los fabricantes para facilitar y mejorar la seguridad de las redes inalámbricas. *Cisco* desarrolló *EAP-*

Fast que mejoró la seguridad ofrecida por su predecesor, pero sigue siendo propietario y no estándar.

1.8.3. EAP-TLS

Este método es el más robusto de todos los métodos de autenticación que implementa EAP. Este método considera la red inalámbrica totalmente insegura, es decir, como si se tratara de la red de Internet, por lo tanto la autenticación debe ser mutua, tanto el cliente como el servidor deben ser autenticados. La autenticación se realiza mediante certificados digitales, con lo que es necesario disponer una estructura PKI (Infraestructura de llave Pública, Public Key Infrastructure).

Con esto podemos ver que este sistema proporciona los requerimientos para poder asegurar que la autenticación es completa y segura. Los certificados proporcionan autenticación entre usuarios y el servidor, con lo que se evitan los ataques de llamados rogue (atacante que usa un AP para suplantar a otro operativo). Además, dado a que los certificados digitales incluyen claves criptográficas, estos proporcionan un método para distribuir claves de cifrado de datos de manera segura.

En este sistema el problema llega en el momento de distribuir los certificados: la clave privada de un certificado digital puede quedar a la vista de un atacante que tenga acceso a los archivos personales de un usuario. No obstante, el mayor inconveniente de este sistema es que implica la necesidad de mantener una estructura PKI y un servidor de autenticación RADIUS. Por lo tanto, la dificultad de implementación de este sistema evita que muchas redes puedan disponer de esta seguridad.

1.8.4. EAP-TTLS

Este sistema es similar al EAP-TLS pero sólo se usa el certificado de servidor. Este método apareció dado a que se prefería mantener la autenticación de red nativa ya implementada, como puede ser la autenticación contra el

Directorio Activo (método usuario y contraseña usado en un dominio Windows), LDAP Directory o autenticación mediante Kerberos. Por lo tanto este método proporciona una mezcla entre el sistema inalámbrico y el sistema que ya funciona en la red cableada.

En un primer paso, se establece un túnel TLS mediante el certificado digital del servidor de autenticación (en el cual confiamos ya que está firmado por una CA). Una vez establecido el túnel seguro, se envía la autenticación de usuario original que ya no depende del método EAP.

Con este método podemos ver que eliminamos parte de la estructura de la PKI (ya no son necesarios los certificados de usuario).

1.9. Protocolo AAA

En seguridad informática, el acrónimo AAA corresponde a un tipo de protocolos que realizan tres funciones: Autenticación, Autorización y Arqueo (contabilización) (Authentication, Authorization and Accounting en inglés). La expresión protocolo AAA no se refiere pues a un protocolo en particular, sino a una familia de protocolos que ofrecen los tres servicios citados.

AAA se combina a veces con auditoria, convirtiéndose entonces en AAAA.

Autenticación: La Autenticación es el proceso por el que una entidad prueba su identidad ante otra. Normalmente la primera entidad es un cliente (usuario, ordenador, etc.) y la segunda un servidor (ordenador). La Autenticación se consigue mediante la presentación de una propuesta de identidad (un nombre de usuario) y la demostración de estar en posesión de las credenciales que permiten comprobarla. Ejemplos posibles de estas credenciales son las contraseñas, los testigos de un sólo uso (one-time tokens), los Certificados Digitales, o los números de teléfono en la identificación de llamadas. Viene al caso mencionar que los protocolos de autenticación digital modernos permiten

demostrar la posesión de las credenciales requeridas sin necesidad de transmitirlas por la red (véanse por ejemplo los protocolos de desafío-respuesta).

Autorización: Autorización se refiere a la concesión de privilegios específicos (incluyendo "ninguno") a una entidad o usuario basándose en su identidad (autenticada), los privilegios que solicita, y el estado actual del sistema. Las autorizaciones pueden también estar basadas en restricciones, tales como restricciones horarias, sobre la localización de la entidad solicitante, la prohibición de realizar logins múltiples simultáneos del mismo usuario, etc. La mayor parte de las veces el privilegio concedido consiste en el uso de un determinado tipo de servicio. Ejemplos de tipos de servicio son, pero sin estar limitados a: filtrado de direcciones IP, asignación de direcciones, asignación de rutas, asignación de parámetros de Calidad de Servicio, asignación de Ancho de banda y Cifrado.

Arqueo: Se refiere al seguimiento del consumo de los recursos de red por los usuarios. Esta información puede usarse posteriormente para la administración, planificación, facturación u otros propósitos. El arqueo en tiempo real es aquella en la que los datos generados se entregan al mismo tiempo que se produce el consumo de los recursos. En contraposición la contabilización por lotes (en inglés "batch accounting") consiste en la grabación de los datos de consumo para su entrega en algún momento posterior. La información típica que un proceso de contabilización registra es la identidad del usuario, el tipo de servicio que se le proporciona, cuando comenzó a usarlo, y cuando terminó.

1.10. RADIUS

RADIUS son las siglas de Remote Authentication Dial-Up Server, que significa Servidor de Autenticación de Autorización Remota para sistemas de Marcado Telefónico a Redes. Este nombre proviene de sus comienzos, donde su único uso era el acceso a redes a través de MÓDEM, pero actualmente su funcionalidad es mucho más amplia.

El motivo por el cual RADIUS es el protocolo AAA hegemónico en la actualidad no es solamente porque haya sido el primero, ni porque haya sido muy comercializado para alcanzar su globalización, sino porque ha ido creciendo y mejorando desde sus comienzos hasta el día de hoy. A pesar de algunas de sus limitaciones, ha ido adoptando una serie de mejoras que le han llegado a permitir gestionar desde pequeñas redes seguras y medianas empresas hasta redes de alto nivel [19].

1.10.1. Descripción del protocolo

RADIUS es un servicio o daemon que se ejecuta en una de las múltiples plataformas que permite (Unix, GNU/Linux, Windows, Solaris…) y que permanece de forma pasiva a la escucha de solicitudes de autenticación hasta que estas se producen. Para ello utiliza el protocolo UDP y permanece a la escucha en los puertos 1812 ó 1645 para la autenticación y 1813 ó 1646 para el arqueo. En un principio se utilizaban los puertos 1645 y 1646 para RADIUS, pero tras la publicación de la RFC 2865 se utilizan por acuerdo 1812 y 1813 debido a que el 1645 estaba siendo utilizado por otro servicio “datametrics”. Algunos servidores como Freeradius utilizan el puerto UDP 1814 para la escucha de respuestas Proxy RADIUS de otros servidores.

RADIUS está basado en un modelo cliente-servidor, ya que RADIUS escucha y espera de forma pasiva las solicitudes de sus clientes o NAS, a las que responderá de forma inmediata. En este modelo el cliente es el responsable del envío y de la correcta recepción de las solicitudes de acceso, y es el servidor RADIUS el responsable de verificar las credenciales del usuario y de ser correctas, de enviar al NAS los parámetros de conexión necesarios para presentar el servicio.

El motivo por el cual RADIUS justifica el uso de UDP sobre TCP en su RFC (Petición De Comentarios, Request for Comments) es por el aprovechamiento de las normativas del protocolo UDP, que mantiene una copia del paquete de solicitud sobre la capa de transporte a fin de poder recuperarlo para reenviarlo, si fuera necesario, a otro servidor RADIUS si el primero no estuviera disponible. De

esta manera se simplifica el diseño del protocolo, evitando tener que hacerse cargo del control de llegada de esos paquetes a su destino. Para aprovechar esta simplicidad se utiliza la característica de UDP de ser “sin cable”. Las retransmisiones se pueden hacer más rápidamente hacia otros servidores, ya que el puerto no quedará colapsado por el control de la conexión, evitándose las esperas necesarias en el protocolo TCP.

Dispone de una muy extensa variedad de módulos de autenticación, encargados de completar un proceso de autenticación con todo lo que ello conlleva. En una comunicación RADIUS nunca se enviarán las contraseñas en texto claro, incluso en sus versiones más antiguas se utilizaba un sistema de cifrado, aunque este sistema primitivo se ha quedado ya obsoleto. Estos módulos de autenticación se han ido desarrollando a medida que el mercado ha ido demandado nuevos sistemas más seguros y fiables para la autenticación. La idea predominante es la de sustituir los métodos que se van quedando obsoletos por vulnerabilidades o problemas de seguridad por otros más actuales que ofrezcan más confianza y más posibilidades de servicio.

Es un protocolo extensible, por lo que permite la introducción mediante su sistema de atributos o variables definibles (AVP) de cualquier adaptación a cualquier nuevo equipo de cualquier fabricante. Este sistema de funcionamiento mediante atributos es uno de los principales pilares de este protocolo, ya que toda la estructura modular se basa en ellos.

1.10.2. Especificaciones de RADIUS

Como cuando acudimos a comprar un vehículo, debemos conocer cuáles son las características a tener en cuenta y conocer la función e importancia de cada una de ellas. Para ello, comenzaríamos descargando de Internet o solicitando folletos y especificaciones de cada una de las opciones que vayamos a considerar [19].

Pero, básicamente, ¿qué especificaciones mínimas debe ofrecer un servidor RADIUS propicio para la aplicación a la que va a ir destinado?

- Cumplir la función para la cual se va a adquirir.
- Incluir todas las tecnologías necesarias para que pueda cubrir las necesidades del usuario final de la autenticación, a través de cualquier sistema operativo y/o plataforma segura.
- Soportar y ser soportado por todas las plataformas de hardware que utilicemos en la infraestructura interna de red, como equipos de electrónica de Red, NAS, Plataformas de PPP, ADSL, GSM, Wi-Fi, Wi-Max, enrutadores, etc.
- Soportar el sistema o sistema de base de datos o servicio de directorios que hayamos elegido para gestión de usuarios y de arqueo de cuentas.
- Disponer de la arquitectura adecuada para la instalación en la plataforma servidora elegida.
- Disponer de las librerías de programación y personalización adecuadas para la personalización de sistemas como PHP, Java, Perl, Python, etc.
- Ser fácilmente configurable y administrable.
- Fidelidad a los estándares y RFC, que los regulan.
- Ser transportable y migrable a otros entornos.

Todas estas características, dependiendo del tipo de implementación que necesitemos, son las que definen a RADIUS o a cualquier servidor AAA.

1.11. Autenticación contra base de datos

Se realiza la autenticación con una base de datos, normalmente del tipo SQL, como Oracle, Microsoft SQL Server, MySQL, etc. Los datos de credenciales de usuarios se almacenan en estas bases de datos, así como los atributos de autorización y la información de arqueo de cuentas. De esta manera es muy sencilla la administración de estos datos, así como la consulta, edición o

eliminación de las mismas. El rendimiento que ofrecen los sistemas SQL y las posibilidades de redundancia y balanceo de carga son espectaculares para implementaciones de gran tamaño.

La mayor parte de los servidores de RADIUS incluyen soporte para bases de datos, lo que simplifica la tarea de administración de las bases de datos necesarias. Todos los servidores importantes incorporan plantillas SQL para la creación de las instancias y tablas necesarias.

Para la gestión de autenticación de un elevado número de usuarios la base de datos es la solución más apropiada, pudiéndose crear scripts automáticos en lenguaje SQL para su administración. Además siempre podemos usar funciones como MD5, SHA 1 u otras personalizables para el almacenamiento cifrado de las contraseñas o información importante. Podemos modificar o crear cualquier secuencia o script SQL a ejecutar durante el proceso de Autenticación, Autorización o Arqueo, con lo que la potencia de este sistema es ilimitada. La autenticación de usuarios por base de datos va unida al uso de un método de autenticación y/o de transporte de la autenticación como CHAP, PAP, EAP-TLS, EAP-PEAP, etc. [19].

A la potencia de RADIUS le estaríamos sumando la potencia de las plataformas de bases de datos secuenciales más actuales, por lo que la suma de ambas merece de una importante consideración.

1.12. SHARED SECRET. El secreto mejor guardado

El SHARED SECRET o secreto compartido de RADIUS es lo que se conocía como RADIUS key o RADIUS secret. Es una contraseña con formato alfanumérico de hasta 128 bytes, que se define en los dos extremos de un canal RADIUS entre el cliente y el servidor, o sea, entre el NAS y el Servidor RADIUS. Se pueden y se deben usar contraseñas o secretos diferentes para cada uno de los equipos NAS que actúen contra un servidor.

Este secreto se utiliza para encriptar las comunicaciones entre el cliente y el servidor RADIUS, ya que de no hacerlo, se podría interceptar y examinar todo el tráfico en texto claro entre clientes y servidor RADIUS. Utilizando la autenticación simple PAP el servidor generará un mensaje de rechazo de acceso (Access-Reject) si el secreto compartido no es correcto debido a que el atributo clave de usuario (User-Password) se cifra utilizando el shared secret, mediante cualquier otro tipo de autenticación el servidor simplemente descarta la solicitud de forma silenciosa.

MD5 es un algoritmo de reducción del mensaje, no de cifrado de mensaje, y se utiliza para no tener que transportar una clave en texto plano. Esa clave se pasa por una función MD5 que la procesa en una frase generada o hash que es irreversible; no se debe poder obtener la clave inicial a través de ese hash.

En una autenticación mediante RADIUS, el cliente o NAS comparte el mismo secreto que el servidor, por eso se llama secreto compartido. Cuando se produce una solicitud de acceso (Access-Request) en NAS envía esta petición en texto plano, cifrando solamente el campo de password o contraseña de usuario. Para ello el NAS genera una frase de 16 bytes de código aleatorio que se denomina Request Autenticator, y que incluye en el paquete de solicitud. Esta frase se concatena con el secreto compartido y se pasa el resultado por una función MD5 que crea un hash de 16 bytes, al que se le aplica la función .XOR con los primeros 16 bytes de la contraseña del usuario. Si la contraseña midiera más de 16 bytes se repetiría la misma fórmula tantas veces como segmentos de 16 bytes tuviera la contraseña, excepto porque la concatenación de la función MD5 se realiza usando en vez del Request Authenticator el resultado de la primera frase cifrada.

$$fc[1] = p[16b] \,.XOR\, MD5\,(ss + ra)$$
$$fc[2] = p[16b] \,.XOR\, MD5\,(ss + fc[1])$$
$$fc = fc[1] + fc[2]$$

Donde:

fc= *Frase cifrada,* p [16b]= *segmento de 16 bytes de la contraseña de usuario,*

$ra = Request\ Authenticator, += concatenación$

Como medida de seguridad se recomienda usar shered secret de un mínimo de 22 bytes no basado en diccionario y respetando las normas de seguridad de cualquier contraseña aceptable, como utilizar símbolos, números, letras mayúsculas y minúsculas [19].

1.13. Referencias

[1] http://biblioteca.usac.edu.gt/tesis/08/08_6115.pdf

[2] http://biblioteca.usac.edu.gt/tesis/08/08_7235.pdf

[3] Menezes A., van Oorschot P., y Vanstone S. Handbook of Applied Cryptography. CRC Press, New York, quinta edición, 2001.

[4] Gollmann D. Computer Security. John Wiley & Sons, Chichester,New York, 1999.

[5] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press 1996

[6] B. Schneier, Applied Cryptography, John Wiley & Sons, Inc. 1996

[7] D.R. Stinson, Cryptography Theory and Practice, CRC Press Inc.

[8] ANSI X3.106 “American National Standard- Data Encryption Algorithm- Modes of Operation”, American National Standards Institute 1983

[9] FIPS 81 “DES modes of operation” 1980

[10] ISO 8372 “Information processing – Modes of operation for a 64-bit block cipher algorithm”, 1997

[11] P. van Oorschot, M. Wiener, A Known-plaintext attack on two-key triple encryption, Advances in Cryptology EUROCRYPT ’90, LNCS 473, pp 318-325, 1991

[12] ISO 10118-1,2,3,4 “Information technology- security techniques- hash functions” 1994, 1996

[13] ISO 9731-1,2 "Banking – Approved algorithms for message authentication" 1987,1992

[14] Mathew Gast, 802.11 Wireless Networks The Definitive Guide, Ed. O'Reilly, 2005

[15] ANSI/IEEE Std. 802.11-1997: Wireless LAN Medium Access Control (MAC) Sublayer and [ISO/IEDC 88021-11] Physical Layer Specifications.

[16] 802.11 WLAN Packets and Protocols. WildPackets. http://www.wildpackets.com/support/compendium/manual_appendices/nxA1_AP#wp1001277

[17] Jahanzeb Khan, Building secure wireless networks with 802.11, Wiley Publishing, 2003

[18] Cisco Response to Dictionary Attacks on Cisco LEAP. Cisco Systems http://www.cisco.com/en/US/products/hw/wireless/ps430/prod_bulletin09186a00801 cc901.html#wp1002197

[19] AAA /RADIUS/802.1x, Sistemas basados en la autenticación en Windows y Linux/GNU, Yago Fernández Hansen, Antonio Ramos Varón, Jean Paul García – Morán, Alfaomega Ra- Ma

CAPÍTULO 2 PROPUESTA Y ANÁLISIS DEL MODELO E INFRAESTRUCTURA DE IDENTIFICACIÓN Y AUTENTICACIÓN DE USUARIOS

2.1. Introducción

Cuando se habla de *Seguridad* de la información en cualquier tipo de red de datos, el objetivo es poder garantizar estos tres conceptos: confidencialidad, autenticación e integridad. Es por ello que nuestra propuesta de solución al problema de seguridad, nos lleva a la creación de una Infraestructura de Identificación y Autenticación de Usuarios.

El modelo se presenta en la Figura 2.1, pretende tener un carácter estándar para el manejo de diferentes tipos de red, diferente composición tecnológica de hardware y topología lógica.

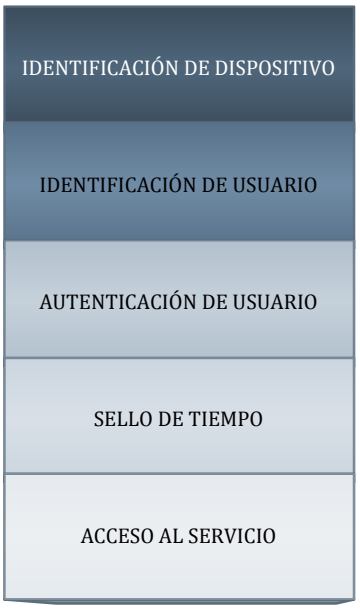

Figura 2.1 Infraestructura de sistema

2.2. Identificación de dispositivo

Este módulo permite la conexión entre el dispositivo móvil del usuario y el dominio de red mediante un dispositivo de interconexión que es el que le permitirá el acceso a la red y por ende a los servicios que en esta se ofrezcan (Figura 2.2). La Figura 2.3 muestra el diagrama del módulo.

El punto de acceso es el dispositivo de interconexión que existe entre el dominio de red y los dispositivos móviles de los usuarios. Cuando la conexión entre el dispositivo móvil y el dominio de red se da por primera vez tenemos que el punto de acceso de la red transmite periódicamente su identificador de conjunto de servicio (SSID). El equipo móvil de usuario escucha el SSID del punto de acceso y lo retransmite para lograr una asociación, haciendo una petición por parte del dispositivo móvil al servidor DHCP para que le proporcione la información necesaria para conectarse a la red.

El servidor DHCP envía al dispositivo móvil la configuración de red, que incluye una dirección IP, la dirección del Gateway y las direcciones de los servidores DNS's. Al mismo tiempo el Servidor DHCP actualiza su tabla de usuarios la cual contiene la asociación de la dirección IP con la dirección MAC de cada uno de los dispositivos móviles de los usuarios que están conectados.

Figura 2.2 Interconexión

Esta tabla de usuarios del servidor DHCP se le envía a un Servidor de Usuarios para que verifique contra una Base de Datos de Usuarios Registrados, si el dispositivo móvil ya está registrado en la red. En caso de ser afirmativa la búsqueda se le envía al usuario una petición de identificación a través de una interfaz en su dispositivo. De lo contrario se rechaza la petición de acceso.

En el diseño de la infraestructura de red de telecomunicaciones que soportará el modelo del sistema, deberá tenerse en cuenta:

- La cantidad de usuarios aproximada.
- Puntos de acceso suficientes para una carga balanceada de usuarios.
- Soporte de tecnologías inalámbricas diferentes (Wi-Fi, Bluetooth, Wimax, etc.).
- Interoperabilidad de estas tecnologías para acceso a la misma fuente de información.

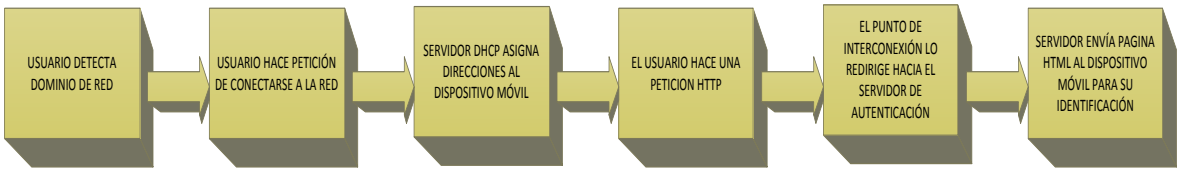

Figura 2.3 Módulo identificación de dispositivo

2.3. Identificación de usuario

La seguridad es un aspecto fundamental en el desarrollo del modelo. Pero, el que sea uno de los principales objetivos del modelo proporcionar información y/o servicios a todos los usuarios no significa que cualquier usuario pueda tener acceso a cualquier información. El acceso a los servicios debe ser controlado individualmente en función de la identidad del usuario.

Mediante la interfaz se le hace la petición de introducir su nombre de usuario lo cual permitirá saber si está registrado. Estos datos se envían al servidor de usuarios para que verifique contra una Base de Datos de Usuarios Registrados. En caso de ser afirmativa pasa a la etapa de autenticación. De lo contrario se niega el acceso.

El módulo de identificación de usuario se muestra en la Figura 2.4

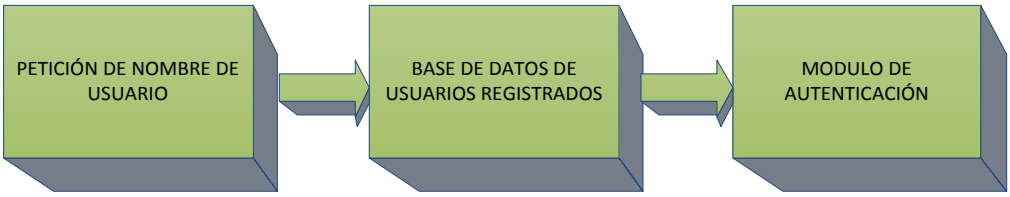

Figura 2.4 Módulo identificación de usuario

2.3.1. Diagrama de flujo del módulo de identificación

En este diagrama se muestra el proceso de identificación (Figura 2.5) por parte del usuario, deberá introducir sus datos los cuales serán verificados, si el usuario es válido entonces el sistema enviará una petición de Autenticación la cual se describe en el siguiente diagrama.

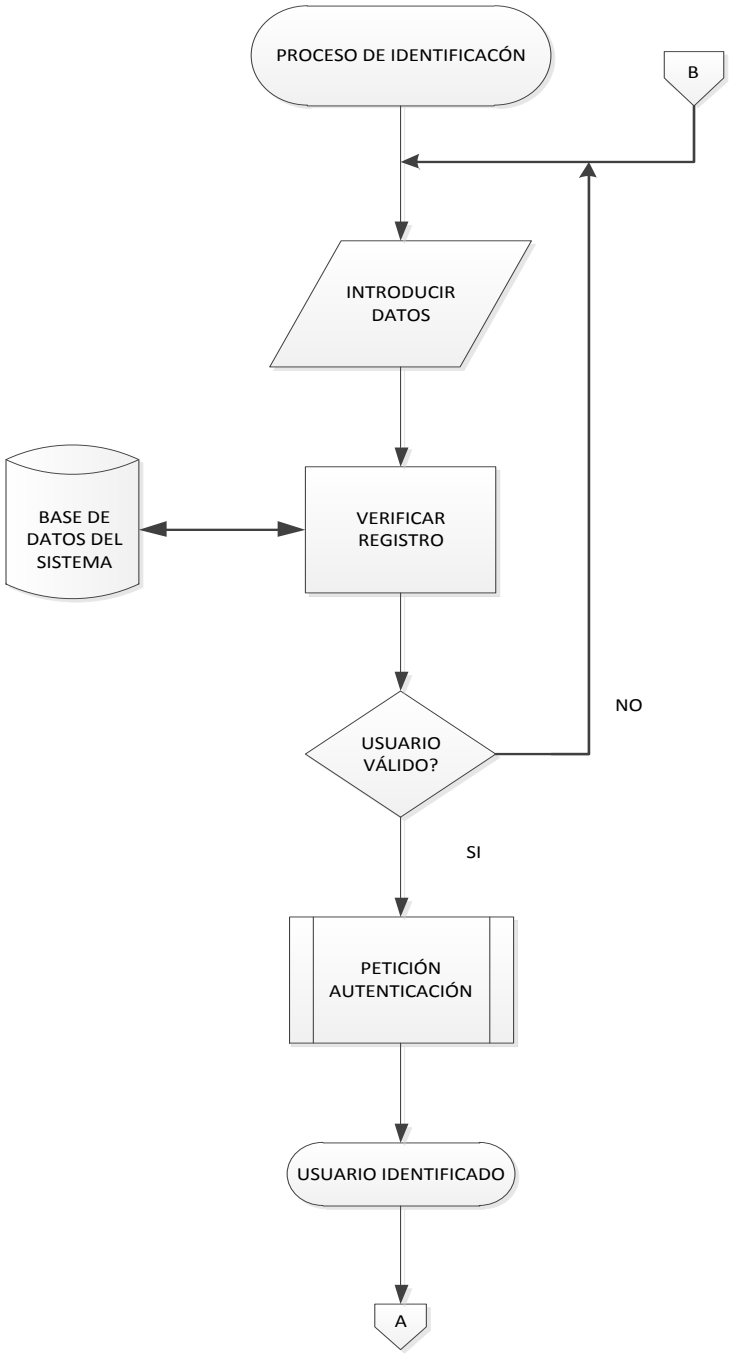

Figura 2.5 Diagrama de flujo del módulo de identificación

2.4. Autenticación

Si bien el estándar 802.1x de IEEE habla de los servidores de autenticación en términos genéricos, en la realidad son elementos que se diseñan en base a los siguientes pasos: Autenticación, Autorización y Auditoria (AAA).

Autenticación. La autenticación es un modo de asegurar que los usuarios son quién ellos dicen que ellos son y que el usuario que intenta el acceso a los servicios es de hecho el usuario que tiene la autorización para ello.

Autorización el proceso por el cual la red de datos autoriza al usuario identificado a acceder a determinados servicios.

Auditoría mediante la cual la red registra todos y cada uno de los accesos a los servicios que realiza el usuario, autorizados o no.

Muchos protocolos de seguridad extensamente adoptados están basados en esta asunción. Los métodos de autenticación están en función de lo que utilizan para la verificación y en este caso el método que será utilizado:

- Firma digital
- Implementaremos un Servidor de Autenticación que sea quien verifique la autenticidad del usuario utilizando los datos capturados en el formulario de una pequeña página que se le presenta al usuario para que introduzca su nombre de usuario y contraseña, como lo muestra la Figura 2.6 [1].

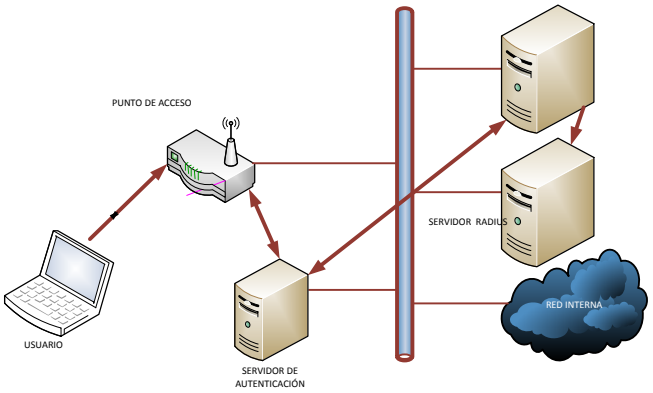

Figura 2.6 Conexión del módulo de autenticación

En el proceso de autenticación y autorización de un usuario para acceder a la red, tienen lugar varios pasos:

1. Previo al acceso, el usuario tiene que tener generado un certificado digital por la Autoridad de Certificación (CA).
2. El cliente *wireless* solicita al punto de acceso permiso para establecer una conexión y acceder a la red. Para ello le envía una solicitud firmada con su clave privada. El punto de acceso está configurado para reenviar la solicitud al servicio RADIUS.
3. El servicio RADIUS, consulta al directorio de autenticación para comprobar las credenciales del usuario y su validez. Además también consulta al Directorio de autenticación las políticas de acceso (horarios de conexión, requisitos de cifrado y autenticación, etc.).
4. El servicio RADIUS determina si el usuario tiene acceso a la red y envía la autorización al punto de acceso.
5. El punto de acceso inicia un intercambio de claves para establecer un cifrado de sesión con el cliente, permitiéndole así acceder a la red de forma segura.

2.4.1. Diagrama de flujo del módulo de autenticación

Este diagrama muestra el proceso de comparación de datos, una vez que el usuario es identificado se compararan los datos de la aplicación con los de la base de datos de la autoridad validadora, después hará una petición de firma para compararla con la base de datos del sistema, si la firma es correcta, el usuario será autenticado (Figura 2.7).

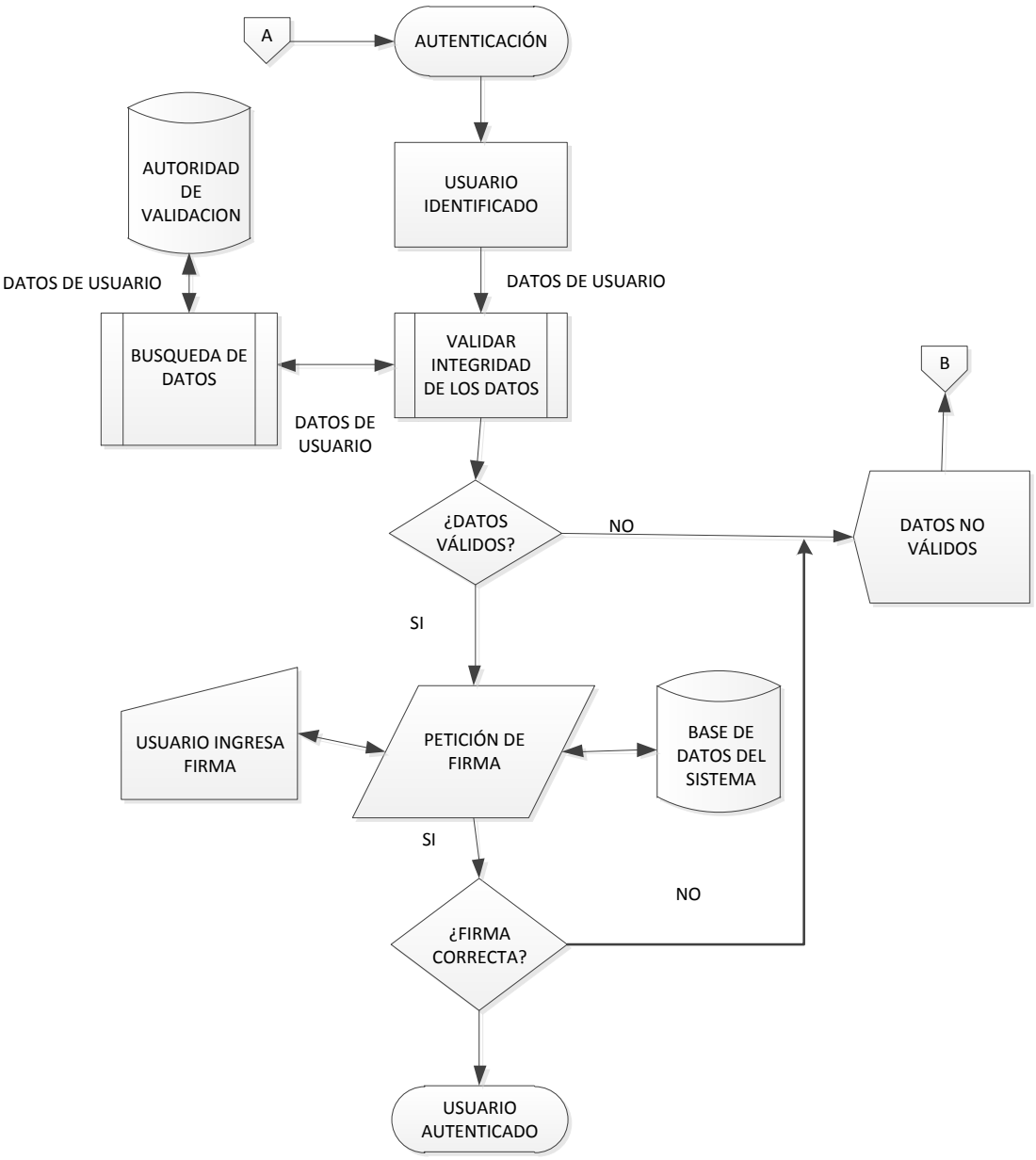

Figura 2.7 Diagrama de flujo del módulo de autenticación

2.4.2. Diagrama de flujo para verificar validez de los datos.

Este diagrama muestra el proceso de comparación de los datos del usuario del sistema con los datos almacenados en el servidor de la autoridad de validación, si los datos del usuario son iguales con los de la autoridad validadora, el sistema confirma la integridad de los datos del usuario (Figura 2.8).

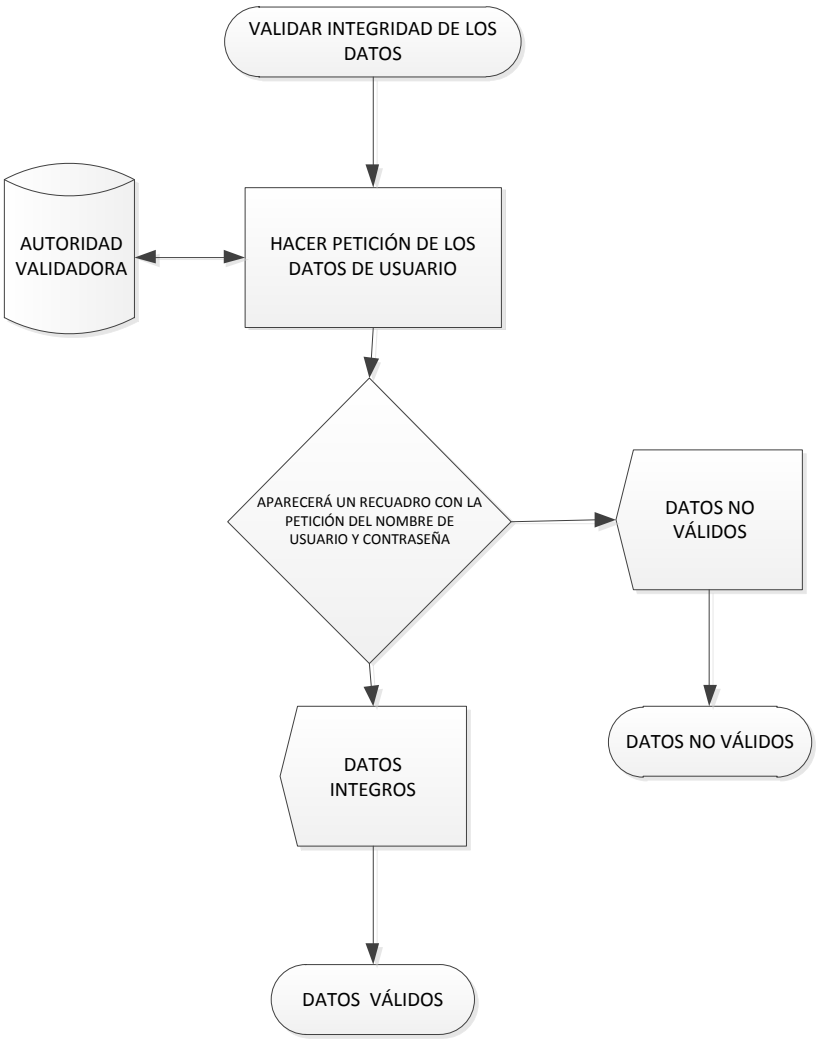

Figura 2.8 Diagrama de flujo para verificar la validez de los datos

2.5. Sello de tiempo

El sello de tiempo de un documento electrónico permite garantizar que la información contenida en el mismo no se ha modificado desde el momento de tiempo en el que se generó el sello. Se solicita a una Autoridad de Sellado de Tiempo (TSA) mediante un resumen (hash) de la información a sellar.

Este módulo genera un sello de tiempo con el resumen, la fecha y la hora. La TSA proporciona el sello al solicitante, que lo puede adjuntar a la información para garantizar su integridad en el tiempo. Asimismo, la TSA almacena los sellos emitidos para futuras verificaciones [2].

2.6. Acceso al servicio

Este módulo lleva a cabo la administración de servicios con los que cuenta cada usuario. Se encarga de presentar al usuario los servicios de forma organizada y disponible para cuando así los requiriera. Este módulo presenta dependencia en relación con el módulo de identificación ya que para poder establecer la administración de los servicios con los que cuenta el usuario requiere previamente su autenticación.

La administración del servicio se dará de acuerdo a la base de datos de usuarios registrados en el sistema Figura 2.9.

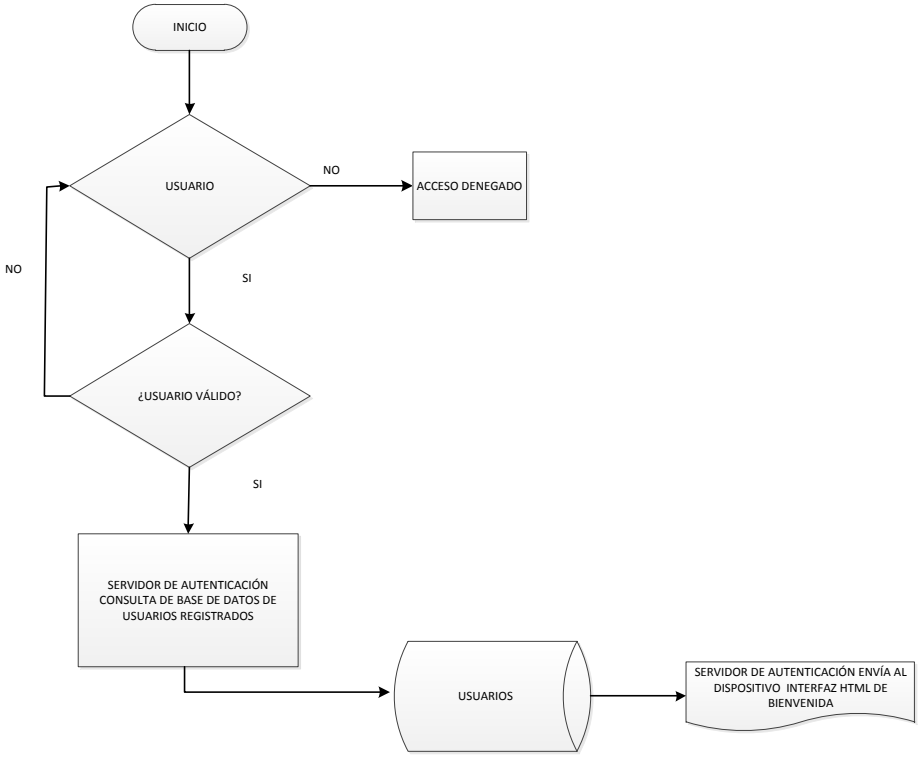

Figura 2.9 Diagrama de flujo de acceso al servicio

2.7. Referencias

[1] http://www.areino.com/alf/docs/ArquitecturaSeguraWireless.pdf
[2] http://www.accv.es/empresas/sellado-de-tiempo/

CAPÍTULO 3 CASO DE ESTUDIO DE UN SISTEMA DE RED PARA SERVICIOS

3.1. Introducción

El objetivo de la tercera parte es conseguir un sistema basado en el estándar 802.11 el cual podamos considerar seguro.

Como veremos en los siguientes puntos, seguiremos una metodología determinada para la construcción de este sistema, utilizando componentes ya existentes y desarrollando otros necesarios para cumplir con los requerimientos.

Así pues, los conceptos tratados en los diversos puntos de esta tercera parte pueden resumirse en:

- Definición de necesidades de nuestro sistema, la cual incluirá los requerimientos del sistema.
- Planteo de la solución y configuración que adoptaremos para solventar la problemática propuesta.

3.2. Requerimientos

La especificación del sistema debe ser basada en el conocimiento proporcionado en el Capítulo 1, donde se describen los conceptos básicos.

Los datos recogidos en los apartados anteriores nos hacen comprender cuál es la deficiencia real de la seguridad en los sistemas y qué podemos ofrecerle al usuario para mejorarla. Este conocimiento otorga la ventaja de poder hacer una especificación inicial más completa previa al diseño del sistema que ayude a conseguir los requerimientos propuestos.

Así pues en este apartado serán descritos los requerimientos de nuestro sistema. A partir de los requerimientos, se hará la especificación del sistema, la cual complementa a estos requerimientos y fija determinados parámetros, como la arquitectura o la seguridad utilizada. De esta manera propiciamos que la etapa de desarrollo se centre más en la implementación de las diferentes partes del sistema que en comprobar si se cumplen los requerimientos.

3.2.1. Especificación de requerimientos

La premisa básica del sistema será que proporcione robustez ante intrusiones. La seguridad debe ser el máximo exponente de nuestro sistema. Tampoco hemos de olvidar que este sistema ha de ser sencillo.

Seguidamente especificaremos los requerimientos u objetivos del sistema los cuales nos servirán de apoyo:

1. Garantizar la confidencialidad de datos, asegurando que la información transmitida sólo sea descifrable por el usuario al cual va dirigida.
2. Autenticación mutua de usuario y de servidor. Ambos han de asegurar que se comunican con quien realmente desean comunicarse. Comprobar la identidad del cliente por parte del servidor y del servidor por parte del cliente.
3. Control de acceso variable de los usuarios, para poder en todo momento permitir o denegar el servicio de manera individual.
4. De fácil manejo para usuarios o administradores poco experimentados.
5. Que no requiera de una instalación complicada ni de módulos externos que puedan limitar o impedir el uso del sistema. El sistema debe ser compacto.
6. Gestión automática de usuarios. El usuario dado de alta será único para todo el sistema, no debe haber varios pasos ni varios programas en los que definir el usuario. Con esto evitamos ambigüedad en la definición de usuarios así como posibles errores en el momento.
7. Permitir la autonomía del dispositivo, sin necesidad de actuaciones externas.
8. Posibilitar que el sistema pueda arrancar mediante la carga de algún tipo de archivo de configuración.
9. Ofrecer portabilidad al sistema para que pueda implantarse en diferentes ubicaciones.
10. De costo 0. La parte software ha de ser cuanto más barata mejor, teniendo como meta el costo 0.

3.2.2. Especificación del sistema

La especificación del sistema normalmente no es posible realizarla en el momento de plantear los requerimientos. En cambio, en este caso, dado que somos nosotros el ente que demanda los requerimientos y también conocemos posibles soluciones, podemos fijar algunos parámetros del sistema para ayudar tanto al cumplimiento de los requerimientos como a su correcto desarrollo.

Los parámetros fijados a priori para el desarrollo del proyecto serán tanto el nivel de seguridad, como una arquitectura que cumpla con los requerimientos 1, 2 y 3, que son los más críticos.

Seguidamente, realizaremos una descripción de las soluciones actuales que cumplan con los parámetros fijados y con los requerimientos del sistema y mostraremos cuál ha sido elegida.

3.2.2.1. Definición del nivel de seguridad

Para que una red inalámbrica sea segura no puede utilizar WEP. También podemos decir que un sistema fiable tendrá que tener una clave de cifrado dinámica, es decir, los paquetes no se cifrarán con el mismo Key Stream.

Además, vemos que es igual de importante tener la certeza de que, tanto el origen como el destino de la comunicación (AP y cliente), son realmente los que dicen ser, evitando así posibles ataques de falsas autenticaciones o ataques man-in-the-middle. Para ello deberemos implementar un sistema de autenticación que nos ayude distinguir entre cada elemento de la red. Podemos añadir también que el método de autenticación no puede ser del tipo clave compartida, donde todos los usuarios tienen la misma clave de red.

Afortunadamente, el estándar nos ofrece la posibilidad de usar diversos métodos de autenticación robustos como es el 802.1x. Pero no todos estos métodos son válidos, ya que algunos dejan parte de la seguridad de la red en una contraseña. Por ello la necesidad de un sistema que use contraseñas de conexión y algún otro método de autenticación.

La conclusión a estas premisas es que el método de autenticación que más encaja con nuestras necesidades será *EAP-TLS* (incluido en el estándar *802.1x*), el cual usa certificados digitales para la autenticación mutua en una comunicación.

En el caso del cifrado, para obtener un *Key Stream* dinámico, lo conseguimos también gracias al método de autenticación basado en *802.1x.* Gracias a este método conseguimos que cada conexión pueda negociar una clave propia.

Como veremos en la Definición de la arquitectura de red, la elección del cifrado quedará fuera de la frontera de nuestro sistema (estará definida en el *router*, el cual no estará dentro del sistema), por lo tanto para el cumplimiento del requerimiento 1 se aconsejará el uso de una configuración en el *router*, que se base en *TKIP* incluido en *WPA* (descartamos el uso de *AES* ya que necesita un hardware específico que no está disponible en tarjetas inalámbricas antiguas).

Así pues, podemos definir la configuración de la red como:

· Autenticación: *EAP-TLS*

· Método de cifrado: *TKIP.*

3.2.2.2. Definición de la arquitectura de red

La conclusión alcanzada en el punto anterior implica que, debido a que tenemos un conocimiento del estándar, podemos tener un primer esbozo de los elementos que deberán intervenir en nuestra red. Por lo tanto, aún sin definir el software que desempeñará cada función, el esquema inicial de la red quedará fijado como un requerimiento.

En el sistema descrito en la Figura 3.1 se pueden ver los componentes del sistema que se quiere desarrollar.

En el sistema a desarrollar intervienen una entidad certificadora (CA) y un servidor de autenticación (RADIUS) los cuales proporcionan los certificados para usar en EAP-TLS y el servicio de autenticación respectivamente. Para estos servicios será necesaria la creación de un servidor RADIUS o un sistema de gestión de certificados. Por lo tanto, estableciendo este sistema cumplimos con los

requisitos de establecer una red con autenticación EAP-TLS apta para trabajar con TKIP, resolviendo los tres primeros requerimientos, ofreciendo: confidencialidad, autenticación y control de acceso.

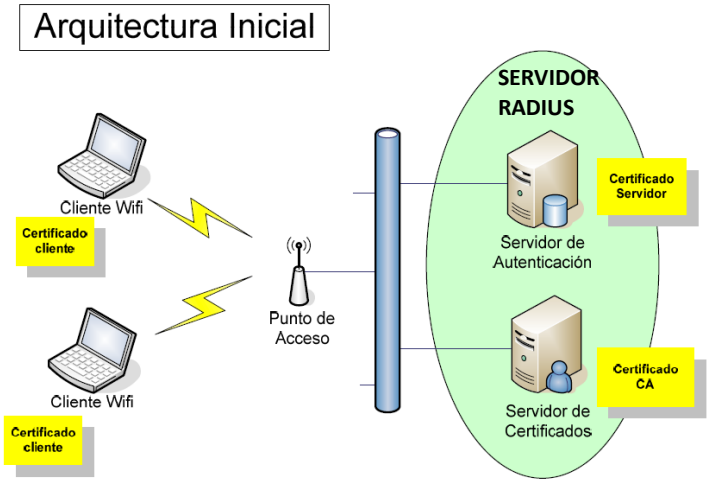

Figura 3.1 Arquitectura inicial del sistema

3.3. Proceso del desarrollo del sistema

El proceso de desarrollo de un sistema es el conjunto de actividades que conducen a la creación del sistema [1]. La elección del tipo de proceso de desarrollo se basará en el sistema a desarrollar y sus requerimientos. La definición de este proceso nos ayudará en la tarea de estructurar las diferentes partes genéricas de la creación de un sistema, que son: recogida de requerimientos, diseño e implementación.

3.3.1. Definición del proceso de desarrollo del sistema

De la arquitectura extraemos que los servicios de autenticación (RADIUS) y de generación de certificados deben ser implementados.

Elegimos el proceso de desarrollo basado en componentes [1]. Este proceso encaja con el sistema aquí expuesto ya que contempla la implementación de un sistema reutilizando otros sistemas existentes, los cuales proporcionarán algunos de los servicios necesarios para cubrir

determinados requerimientos. Dado que los componentes son totalmente independientes entre sí, es necesario el desarrollo de un sistema que haga de interfaz entre todos los sistemas y pueda cubrir los requerimientos que los otros sistemas no cubren.

3.3.2. Esquema del proceso de desarrollo del sistema

En la Figura 3.2 vemos las diferentes actividades a realizar en el proceso de creación de nuestro sistema, el cual seguirá el desarrollo basado en componentes. Podemos observar que se definen dos sistemas diferentes:

- Sistema de componentes: es el sistema conjunto el cual cumple los requerimientos. Este sistema está dividido en subsistemas o componentes, los cuales suelen ser software totalmente independientes.
- Sistema de apoyo: es un componente del sistema de componentes el cual consigue completar los requerimientos que los otros componentes no ofrecen. Este sistema de apoyo deberá actuar de interfaz entre los diferentes componentes.

El sistema de apoyo está incluido en el sistema de componentes, siendo un componente más pero que debe ser desarrollado.

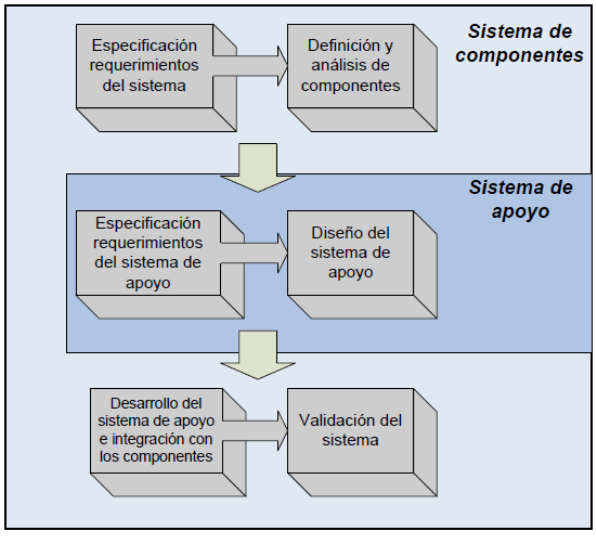

Figura 3.2 Proceso de desarrollo basado en componentes

Describiremos de manera ordenada cada una de las fases para poderlas identificar con el proyecto aquí realizado:

1. Especificación de requerimientos del sistema (de componentes): en esta fase se describen los requerimientos generales sin tener en cuenta división entre componentes. En el caso de este sistema esta fase ya ha sido descrita en el apartado 1 del CAPITULO 3.
2. Definición y análisis del sistema de componentes: en esta fase se propone de manera formal el sistema, se definen sus componentes, se describe su configuración y que requerimientos cumple cada componente. Especificación del sistema, pero no es una definición completa. En este apartado añadiremos un punto donde se mostrará la justificación de la elección de los componentes.
3. Especificación de requerimientos del sistema de apoyo: el sistema de apoyo necesitará de la definición de otros requerimientos ya que deberá cumplir con los requerimientos no cubiertos por los demás componentes. Además deberán añadirse otros requerimientos los cuales definan las interfaces para tratar con los componentes.
4. Diseño del sistema de apoyo: se especificarán a un nivel más bajo los procesos a implementar para cumplir con los requerimientos de la herramienta de apoyo.
5. Desarrollo del sistema de apoyo e integración: el desarrollo es la parte de escritura de los programas que confeccionarán el sistema de apoyo. Se realizará conjuntamente con la integración, dado que el sistema de apoyo ha de funcionar de manera conjunta con los otros componentes. Esta fase se basará en la fase de diseño y de especificación de requerimientos del sistema de apoyo.
6. Validación del sistema: finalmente se comprobará que los requerimientos del sistema se cumplen y que el sistema completo funciona.

3.4. Modelo de componentes

En este apartado definiremos los componentes del sistema, describiéndolos y estableciendo la configuración de aquellos que ya están implementados.

Antes, debido a que disponemos de la arquitectura del sistema, determinaremos el contexto del sistema el cual nos ayudará a distinguir las diferentes partes del mismo.

3.4.1. Contexto del sistema

Definimos como contexto el modelo de sistema donde pueden verse las interacciones entre los integrantes o componentes del sistema [1]. Con este esquema general pueden verse todos los componentes e incluso aparecer componentes nuevos necesarios por la integración.

En la Figura 3.3 vemos el contexto del sistema y también el ámbito, que es el contexto del sistema de apoyo. Mediante la definición ámbito podemos vislumbrar las interacciones directas (marcadas con línea continua) y las indirectas (marcadas con línea discontinua) que tendrá con los otros componentes. Las interacciones directas serán gestionadas mediante el sistema de apoyo que será el que se encargará de modificar, controlar y depositar en sus correspondientes carpetas los ficheros necesarios.

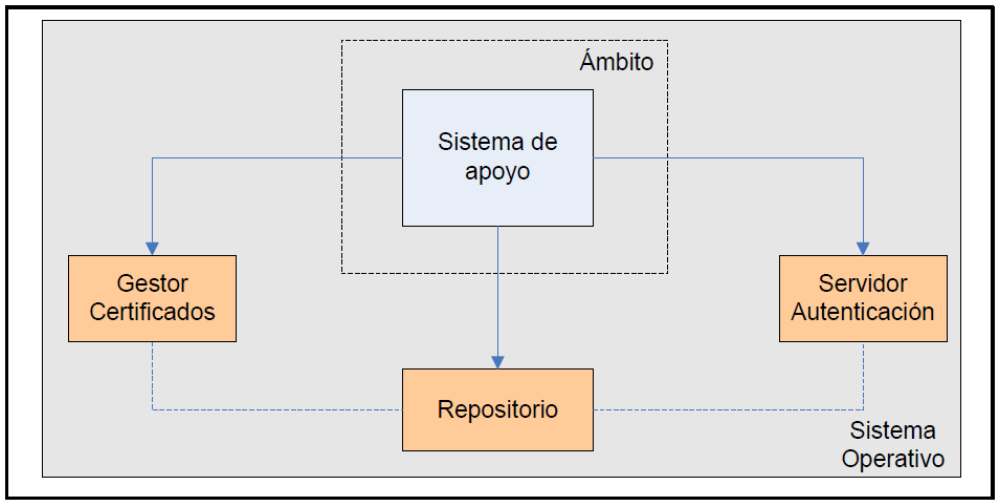

Figura 3.3 Contexto del sistema

Con la definición del contexto o diagrama de arquitectura obtenemos un nuevo componente, el repositorio común, el cual emerge de la necesidad de funcionamiento conjunto de los otros componentes. El repositorio común será donde se almacenarán las configuraciones de los componentes y la estructura necesaria para la entidad de certificados.

La inclusión del repositorio obliga a que el sistema de apoyo no sólo deba tener interfaces para trabajar con el gestor de certificados y el servidor de autenticación, sino que debe incluir métodos para gestionar el repositorio compartido. Todo este sistema debe interaccionar con un sistema operativo, el cual debe soportar todos los componentes.

Tomando como punto de partida este esquema, primero elegiremos los componentes y luego realizaremos la configuración para dejarlos listos para funcionar cumpliendo con lo requerido.

3.4.2. Elección de componentes

Manteniendo el esquema definido en los requerimientos, podemos analizar las diferentes posibilidades existentes en el mercado que nos permiten construir una red WPA con autenticación EAP-TLS.

Actualmente hay varios sistemas comerciales que ofrecen este tipo de seguridad y autenticación, como puede ser el servidor IAS (Internet Authetication Server) de Windows Server o el servidor ACS de Cisco Systems, los cuales nos permiten montar un servidor de certificados y de autenticación mediante EAP. El problema reside en que estos sistemas son caros y complicados de gestionar.

Las soluciones de Microsoft, las más extendidas dentro de las empresas, tienen el inconveniente de que solamente funcionan dentro de un dominio definido. Además, la entidad certificadora sólo puede expedir certificados a usuarios que se encuentren dentro del Directorio Activo o Active Directory del dominio. Debido a ello el servidor de autenticación de Microsoft, el IAS, y la entidad certificadora, deberán estar en una máquina que sea controladora de dominio o dependiente del domino. Después de probar esta solución usando la guía que puede verse en [2] y una máquina virtual mediante VMWare, concluimos que este sistema sólo sería

óptimo para una red dónde ya existiera un dominio basado en servidores Microsoft.

La solución de Cisco, el servidor ACS (Access Control Server) [3], encaja bastante en lo que buscamos, ya que se ejecuta en cualquier máquina y la gestión de usuarios es independiente. Aún así, esta solución no incluye entidad certificadora, por lo que deberíamos buscar otro software para la gestión y expedición de certificados, cosa que complica aún más el sistema. Aunque la característica que hace que esta solución sea inviable para nuestro sistema es que su precio puede sobrepasar los 3000€ por licencia, sin contar con los otros componentes como la CA o el servidor de certificados.

Así pues, estas dos soluciones, además de no cumplir nuestros requisitos de sencillez y costo 0, obligan a disponer de una infraestructura basada en Microsoft e incluso no disponen de la funcionalidad necesaria (como puede ser la expedición de certificados para clientes que estén fuera del dominio o que usen otros sistemas operativos). Por ello, para proporcionar agilidad en el sistema, sencillez y costo 0, sólo nos queda una solución de software abierto como Linux.

En el caso de Linux tiene la ventaja de que provee soluciones gratuitas bastante probadas. En contra, normalmente estas soluciones son difíciles de configurar y de comprender para usuarios poco habituados a sistemas Unix. En cuanto a nuestro sistema hay dos software que se ajustan a nuestras necesidades: el servidor de autenticación FreeRadius y la librería criptográfica OpenSSL. Ambos tienen licencia GNU, con lo cual pueden ser usados libremente sin fines lucrativos, y pueden funcionar en la misma máquina. Uniendo estos componentes en una misma máquina se puede generar el certificado y mantener el servicio de autenticación, sin necesidad de disponer de dos servidores.

OpenSSL y FreeRadius son software más avanzados y estables de su tipo, con lo cual el soporte que podremos encontrar será amplio.

Hasta aquí, el sistema sería viable cumpliendo la mayoría de requerimientos, excepto que FreeRadius y OpenSSL son servicios sin interfaz gráfica nativa (funcionan con ficheros de configuración en texto). Ello incumple claramente la premisa de facilidad de uso.

En conclusión es que para cumplir con los requerimientos de seguridad y de costo cero usaremos FreeRadius como autenticador y OpenSSL para la gestión de certificados. De esta manera evitamos tener que desarrollar una CA y un servidor RADIUS nativo. El sistema de apoyo se diseñará para cubrir los requerimientos no implementados por estos dos componentes ya existentes.

La elección de OpenSSL y FreeRadius nos obliga a trabajar bajo un sistema Unix. Esto en realidad será una ventaja ya que permitirá extender los requerimientos de portabilidad y agilidad. El sistema operativo usado será Debian squeeze 6.0 debido a que es caracterizado por su estabilidad por encima de todo, y su política de software libre, que ayudará al buen desarrollo del sistema.

3.4.3. Sistema operativo – Debian squeeze

El sistema operativo será Debian en su versión Debian squeeze 6.0, es la versión estabilizada de esta distribución. Cuenta con el apoyo del Equipo de Seguridad de Debian y es la recomendada para uso en producción. Su instalación se realizará a través del propio CD de Debian squeeze siguiendo los pasos indicados.

Para la configuración del sistema operativo sólo debemos tener en cuenta que disponga de entorno gráfico (para fácil manejo del usuario) y que no tenga habilitado el firewall (si lo tiene habilitado deberá incluir una regla para dejar pasar tráfico por los puertos usados por FreeRadius, normalmente el 1812 o el 1813).

En cuanto a los componentes que debe integrar el sistema operativo, Debian squeeze ya tiene una versión preinstalada de OpenSSL, la cual es recomendable actualizar con el gestor automático de instalaciones (apt-get o synaptic). En el caso de FreeRadius, este no está por defecto en la compilación Debian squeeze y por lo tanto deberá ser instalado.

La instalación de FreeRadius no debe ser realizada a través del gestor automático de paquetes ya que la versión que instala el gestor no compila varias librerías que el servidor necesitará cuando tenga que leer mensajes EAP-TLS (en concreto las librerías (libssl-dev y libpq-dev). Por lo tanto, se deben compilar

manualmente los diferentes archivos para asegurarnos la inclusión de dichas librerías.

3.4.4. Gestor de certificados – OpenSSL

Para la creación y administración de certificados usaremos OpenSSL [4]. Este componente no sólo proporciona la capacidad de gestionar certificados sino que proporciona scripts que facilitan la creación de una estructura de directorios dónde se almacenará la CA y la cual será parte de nuestro repositorio común.

OpenSSL funciona mediante órdenes de comando y usa un archivo de configuración donde se encuentran varios parámetros, entre ellos la definición de directorios de la estructura de la CA. La estructura de carpetas es, y debe ser, conocida y utilizada por todos los componentes del sistema. En este documento se usará un archivo de configuración propio (de esta manera se consigue mantener la configuración por defecto en el archivo original). En nuestro archivo de configuración, para evitar incompatibilidades, sólo estará modificada la ruta, o path, principal, apuntándola a nuestro repositorio, manteniendo intacta la estructura interna de subdirectorios.

En Configuración, archivos y comandos OpenSSL, podemos ver cómo ha sido realizada la configuración de OpenSSL así como todos los comandos que vamos a necesitar para gestionar nuestra estructura de certificados. Los archivos y comandos necesarios para la configuración y gestión de OpenSSL deberán ser tenidos en cuenta cuando se desarrolle el sistema de apoyo.

Debido a que los comandos para generar certificados debieron ser probados hasta dar con los atributos correctos, se han añadido los Certificados generados (formato P12), el cual proporciona capturas de los certificados generados en formato PKCS12 o P12. De esta forma, si en algún momento quisiéramos prescindir de OpenSSL, conoceríamos los atributos que estos deben tener y donde comprobar dichos parámetros.

3.4.5. Servidor de autenticación – FreeRadius

Como servidor de autenticación usaremos FreeRadius, el cual proporciona múltiples métodos de autenticación, entre ellos EAP-TLS. Como vimos en el

apartado del sistema operativo, FreeRadius deberá ser compilado de manera apropiada para gestionar los mensajes EAP-TLS correctamente.

Debido a que FreeRadius va a ser utilizado por el sistema de apoyo incluiremos información común en los ficheros de configuración de FreeRadius. Por ejemplo, en el caso del archivo users, además de indicar el nombre de usuario y su método de autenticación (necesarios para dar de alta a un usuario en FreeRadius), se almacenará otra información del usuario como: el nombre, grupo, nombre del PC, etc. Estas líneas de información estarán comentadas, con lo cual no afectarán al funcionamiento de FreeRadius. Así como el uso de una base de datos llamada radius realizada en Mysql.

En el siguiente Capítulo podemos ver cómo ha sido configurado FreeRadius y qué archivos han de modificarse para la gestión de usuarios y clientes. Además se incluyen explicaciones de cómo el sistema de apoyo debe gestionar estos archivos y cómo lanzar el servicio.

3.4.6. Repositorio común

Este componente aparece de la necesidad de mantener una estructura de directorios común, donde se concentrarán algunos de los ficheros de configuración modificados y toda la estructura de directorios de la CA. De esta manera se consigue mantener los archivos propios en un espacio distinto al espacio por defecto.

En el Capítulo 4 podemos ver cómo ha quedado determinado el repositorio para que puedan trabajar juntos tanto OpenSSL como FreeRadius.

3.4.7. Sistema de apoyo

Como ya hemos comentado, este sistema será el que se encargue de que los diferentes componentes funcionen de manera conjunta. Este sistema será presentado, diseñado e implementado en los puntos siguientes.

3.5. Requerimientos del sistema de apoyo

Los requerimientos aquí descritos serán los que definan al sistema de apoyo, no sólo por los requerimientos funcionales del sistema, sino que también por los requerimientos necesarios del sistema de componentes.

Así pues, aquí definiremos los requerimientos funcionales y no funcionales de nuestro sistema de apoyo, los cuales cumplirán los requerimientos aún no asumidos por el sistema de componentes.

3.5.1. Requerimientos funcionales

Los requerimientos funcionales son aquellos que declaran alguna funcionalidad concreta del sistema. Estos requerimientos determinan, a ojos del usuario, qué debe hacer el sistema [1].

Estas funcionalidades son las extraídas del funcionamiento conjunto del sistema de componentes y serán descritas seguidamente.

Gestión de certificado raíz (CA)

El sistema de apoyo debe ser capaz de crear una nueva CA y de eliminarla. Al crear una CA se requerirán los campos necesarios: nombre, país, región, ciudad y la duración en días de la validez de la entidad raíz.

Para ayudar al usuario se escribirán unos valores por defecto.

El certificado raíz debe ser generado con el componente OpenSSL. Debido a que esta funcionalidad deberá trabajar directamente con este componente de sistema, se recomienda utilizar una interfaz que ofrezca una librería de funciones que controle todas las operaciones descritas, teniendo en cuenta los archivos necesarios y su localización en el árbol de directorios.

Gestión del certificado de servidor

El sistema de apoyo debe ser capaz de crear un certificado de servidor y de eliminarlo. Al crear un servidor se requerirán los campos necesarios: nombre, país, región, ciudad y la duración en días de la validez.

Para ayudar al usuario/administrador se escribirán unos valores por defecto.

De la misma forma que en la creación de la CA, se recomienda el uso de la inclusión de las funciones en una interfaz con OpenSSL.

El sistema de apoyo deberá almacenar el certificado de servidor en la carpeta correspondiente. Además, deberá modificar el fichero eap.conf de FreeRadius.

Gestión de usuarios

El sistema debe ser capaz de ofrecer las funciones necesarias para realizar el alta, baja, consulta y modificación de objetos “usuario” (usuarios válidos en nuestro sistema, que deberán ser válidos en todos los componentes). Cada objeto “usuario” deberá ser creado con unos campos que lo determinen y se le asignará un certificado x509.

Los usuarios deben ser mostrados en una lista ordenada por el dato “Nombre Usuario” para su fácil visualización. Los objetos “usuario” no podrán repetir Alias, debido a que el CN será único. Desde esta lista y seleccionando sobre cada uno de ellos, debemos ser capaces de realizar las opciones de modificación y de borrado de los objetos “usuario”.

Debido a que al gestionar usuarios también se gestionarán certificados, el sistema de apoyo debe ofrecer las funciones necesarias para realizar el alta y la baja de certificados mediante OpenSSL. De igual manera, esta funcionalidad también deberá trabajar con el componente de autenticación FreeRadius, ya que deberemos modificar los ficheros de usuarios. Se deberá realizar una librería que proporcione las funciones capaces de modificar los ficheros necesarios al realizar alguna acción sobre los usuarios. Para evitar duplicidad de información, se recomienda usar el archivo de usuarios de FreeRadius como archivo de almacén de datos de usuario para nuestro sistema de apoyo.

Esta funcionalidad deberá trabajar con el componente de autenticación RADIUS, ya que deberemos modificar los archivos de clientes válidos. Se aconseja realizar una librería que proporcione las funciones capaces de modificar los archivos necesarios al realizar alguna acción sobre los clientes.

Gestión de Clientes

El sistema debe ser capaz de ofrecer las funciones necesarias para realizar el alta, baja, la consulta y la modificación de objetos “cliente” (puntos de acceso válidos en nuestro servidor RADIUS). Cada objeto “cliente” deberá ser creado con unos campos que los determinen.

Visualización de la lista de revocación

El sistema de apoyo debe ofrecer una opción de visualización de la CRL. Para poder visualizar correctamente la CRL deberemos usar el comando de visualización de la CRL de OpenSSL.

Modificación de la configuración EAP-TLS de FreeRadius

Debido a que el usuario puede querer cambiar algunos parámetros del sistema EAP-TLS de FreeRadius, el sistema de apoyo debe proporcionar una función que muestre y permita cambiar los parámetros del apartado TLS del fichero eap.conf de FreeRadius.

Gestión del servicio FreeRadius

Desde el sistema de apoyo se debe ser capaz de lanzar y parar el servicio de FreeRadius.

De la misma manera, el sistema de apoyo debe impedir la ejecución del servicio si no hay un certificado raíz y de servidor válidos.

Además el sistema debe incluir un modo de depuración de lanzamiento de FreeRadius.

Este modo es el que se realiza mediante el comando freeradius –X y permitirá al usuario observar si hay algún error inadvertido por el sistema al lanzar FreeRadius.

3.5.2. Requerimientos no funcionales

Los requerimientos no funcionales son los que restringen el sistema [1]. En nuestro caso algunos de estos pueden venir determinados por el ámbito en el cual se encuentra el sistema de apoyo.

Portabilidad

Como se vio en el apartado Requerimientos, el sistema debe ser portable, por eso se determinó usar Linux.

Además, la información y la estructura generada por nuestro sistema de apoyo también deben ser exportables e importables. Deben ser creadas unas funciones de exportación e importación que dispongan de las siguientes funcionalidades:

- Estructura completa: Donde se copiara toda la estructura del sistema (todo lo contenido en la carpeta raíz de nuestro sistema). La información deberá ser almacenada en un archivo comprimido y codificado. En el momento de la importación se requerirá la contraseña con la que fue exportada la estructura y será importada en otro sistema, el cual no contenga una CA. Esto puede servir tanto de copia de seguridad de la estructura como para generar una estructura de certificados para utilizarla en otro momento y lugar (en este caso los certificados serán copiados).
- Datos de usuario: Se exportarán sólo los datos de los objetos “usuario”. Se generará una base de datos de Mysql que tendrá el nombre los usuarios y su contraseña. La importación de datos en este caso no se copiará sino que generará nuevos objetos “usuario” (y por lo tanto nuevos certificados para éstos). Por lo tanto en este caso se necesitará disponer en el sistema de apoyo de una CA existente.

Seguridad en el sistema de apoyo

El sistema debe proporcionar un control de acceso y cifrar todos aquellos datos críticos (que no dependan de FreeRadius u OpenSSL, ya que si se cifran estos no podrán ser leídos por estos programas).

La contraseña de acceso y todas las demás contraseñas del sistema (de certificados de usuario, CA y servidor), deben ser almacenadas en el archivo de nuestro sistema.

Interoperabilidad con los componentes del sistema

Esta funcionalidad obliga a diseñar unas interfaces con OpenSSL, FreeRadius y con el repositorio común. Estas interfaces deben tener las funciones necesarias para la gestión de usuarios, clientes y ficheros de configuración que los requerimientos demandan.

Facilidad de uso

Para evitar que el usuario deba modificar ningún archivo de configuración o introducir comandos para generar los certificados, se creará una interfaz gráfica intuitiva donde el usuario sólo deberá introducir los datos requeridos y lanzar el proceso.

3.6. Arquitectura del sistema

La arquitectura del sistema final se muestra en la Figura 4.1 y el cual se explica en el siguiente Capítulo.

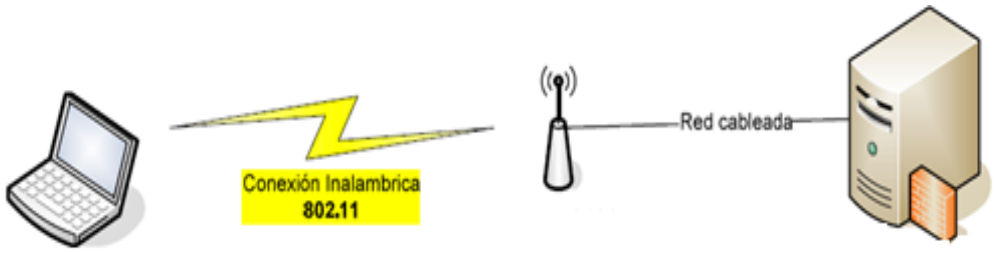

Figura 3.4 Arquitectura de red

3.7. Referencias

[1] Ian Sommerville. Ingeniería del Software. Pearson Educación, cop. 2005
[2] Step-by-Step Guide for Setting Up Secure Wireless Access in a Test Lab.Microsoft Corporation.
http://www.microsoft.com/downloads/details.aspx?familyid=0f7fa9a2-e113-415bb2a9-b6a3d64c48f5&displaylang=en
[3] EAP-TLS Deployment Guide for Wireless LAN Networks. Cisco Systems. http://www.cisco.com/en/US/products/sw/secursw/ps2086/products_white_paper09186a008009256b.shtml
[4] OpenSSL. http://www.openssl.org/

CAPÍTULO 4
DESARROLLO DEL SISTEMA DE APOYO

4.1. Introducción

Para poder desarrollar el sistema de apoyo, será necesario montar un escenario completo con todos los componentes del sistema, además de los componentes externos.

Utilizaremos dos entornos de trabajo. En primera instancia se ha definido un entorno basado en Windows. Una vez completado el aprendizaje, usaremos un sistema basado en Linux para facilitar la integración.

Seguidamente indicaremos los componentes usados en nuestro escenario. Una vez determinados estos componentes, mostraremos la arquitectura del sistema y su configuración básica sobre la cual se realizará el desarrollo.

Así como también se explica paso a paso la configuración que se llevó a cabo dentro del servidor Radius, esto es la instalación de RADIUS Y Openssl, creación de nuestra autoridad certificadora, los certificados de autoridad certificadora, servidor y usuarios, configuración del acces point y por último la configuración de apache como página de bienvenida.

4.2. Entorno inicial

El entorno inicial estará basado en Windows y en él se realizarán las primeras pruebas de manejo del sistema. También se confeccionarán las interfaces del sistema de apoyo:

Componentes del entorno:

- Sistema operativo: *Windows* XP, Windows 7.

4.3. Entorno de compilación

El desarrollo de la mayoría de clases se hará bajo *Linux* ya que el sistema de apoyo deberá funcionar en este entorno. Para comprobar todas sus

funcionalidades será necesaria la ejecución parcial del código. La compilación definitiva del sistema de apoyo también se hará en este entorno:

Componentes del entorno:

- Sistema operativo: Debian Squeeze 6.0

Debido a que este entorno será sobre el que se realizará la compilación y sobre el cual funcionará el sistema definitivo, añadiremos a este entorno los componentes del sistema y los clientes externos al sistema:

- OpenSSL: La versión usada será la 0.9.8o-4squeeze3, la cual ya está integrada en Debian. Debido a que OpenSSL es un proyecto bastante estable y ha cambiado poco en los últimos años [1].
- FreeRadius: Se usará la versión 2.1.10+dfsg-2 de FreeRadius la cual necesitará ser compilada con los módulos que le permitan soportar la autenticación EAP-TLS.
- Punto de acceso: Para montar nuestro sistema haremos servir un punto de acceso Linksys modelo WRT54GS con el firmware DD-WRT .v24 [2]. El firmware DDWRT (el cual no es propio de este dispositivo) no es estrictamente necesario, ya que el firmware original de Linksys soporta lo necesario para ser usado en el sistema. Aún así, el firmware DD-WRT tiene otras ventajas muy útiles que incrementarán sus posibilidades y su manejabilidad, como la capacidad de trabajar mediante comandos (por consola) o la de definir varias VLAN en sus interfaces de red. Incluso, mediante DD-WRT podemos definir varios SSID's con diferentes métodos de autenticación usando un solo punto de acceso, característica que puede ser útil para la implementación de nuestro sistema.
- Usuario: Como usuario usaremos un PC con sistema operativo Windows XP y Windows 7.

4.4. Arquitectura final del sistema

La arquitectura puede definirse en los apartados de requerimientos, diseño o desarrollo. Si se define en los apartados iniciales, ésta queda como un requerimiento difícil de cambiar.

En este documento se ha decidido que la arquitectura del sistema puede cambiar durante el desarrollo para conseguir los requerimientos. Por lo tanto, esta no ha podido ser determinada en una fase más temprana. Por esa razón la arquitectura ha sido incluida en este apartado de desarrollo, fase en la cual las modificaciones no serán tan complicadas de realizar (recordemos que modificar algún concepto en fases anteriores como en la de los requerimientos implicaría una revisión de las fases siguientes).

Describiremos la arquitectura del sistema final (véase Figura 4.1 Arquitectura de red).

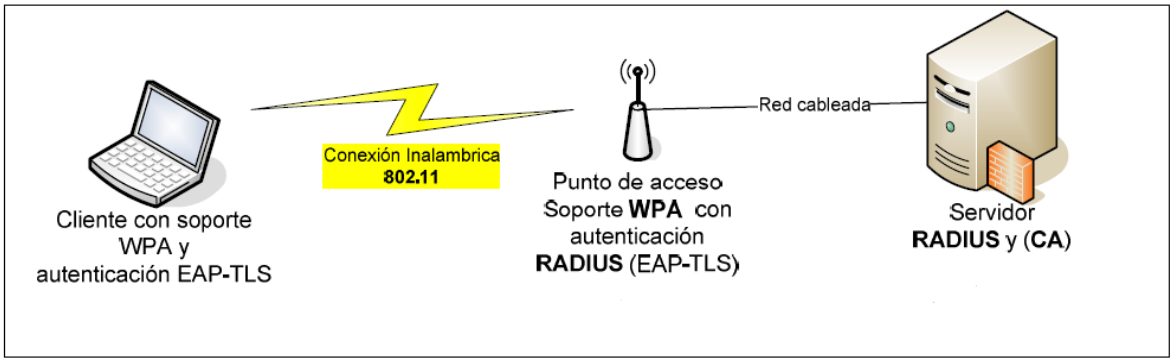

Figura 4.1 Arquitectura de red

Red

Usaremos un direccionamiento privado del tipo 192.168.x.x.

- Red: 192.168.2.0
- Máscara: 255.255.255.0

Servidor

Hardware

- Características: HP 2133 Mini-Note, procesador VIA C7-M a 1.6 GHz, memoria RAM de 2Gb, disco duro de 120Gb,

Sistema operativo servidor

- Debian Squeeze 6.0

Dirección IP

- 192.168.2.120

Gestor de certificados

- OpenSSL versión 0.9.8o-4squeeze3

Servidor de autenticación RADIUS

- FreeRadius versión 2.1.10+dfsg-2

Sistema de Apoyo

- Se encontrará en la carpeta CERTS dentro de FreeRadius.

Punto de Acceso

Hardware

- Router inalámbrico que soporte el estándar 802.11i (aquí se usará Linksys WRT54GS)

Dirección *IP*

- 192.168.2.1

Seguridad

- Configurado con seguridad *WPA* y autenticación *RADIUS*

Cliente

Hardware

- Tarjeta inalámbrica 802.11 con soporte *802.11g*

Software

- *Drivers* y herramienta de configuración para la tarjeta inalámbrica con
- soporte *802.11g*.

Dirección *IP*

- 192.168.2.150

4.5. Instalación de FREERADIUS

Después de haber instalado Debian procedemos a la instalación de la paquetería de LAMP (Linux, Apache, MySQL y PHP). De la siguiente manera:

root@debian:/# apt-get install apache2

root@debian:/# apt-get install php5 libapache2-mod-php5

root@debian:/# apt-get install mysql-server

1 - Primeramente hacemos la instalación de algunas librerías necesarias para que todo funcione.

root@debian:/# apt-get install debhelper libltdl3-dev libpam0g-dev libmysqlclient15-dev build-essential libgdbm-dev libldap2-dev libsasl2-dev libiodbc2-dev libkrb5-dev snmp autotools-dev dpatch libperl-dev libtool dpkg-dev libpq-dev libsnmp-dev libssl-dev

2 - Instalamos el paquete de FreeRadius:

root@debian:/# apt-get install freeradius freeradius-mysql

4.6. Configuración de FREERADIUS

FreeRadius tiene varios ficheros de configuración, los cuales tendrán que ser modificados para que el sistema funcione bajo autenticación EAP-TLS. Nos centraremos en los ficheros de configuración modificados describiendo su función y qué ha sido cambiado en ellos.

radius.conf

El principal archivo de configuración de RADIUS. En él se, encuentran la mayor parte de configuraciones y directivas importantes para hacer funcionar correctamente el servidor RADIUS en diferentes situaciones. La mayor parte de los otros archivos de configuración son leídos por FREERADIUS desde este fichero mediante ordenes *include* que parten de este fichero o de otros.

Es necesario, editar el archivo, que se encuentra en /etc/freeradius/radiusd.conf, descomentamos la línea "$INCLUDE sql.conf" y modificamos las siguientes líneas.

```
listen {
      ipaddr = *
#     ipv6addr = ::
      port = 1812
      type = auth
#     interface = eth1
#     clients = per_socket_clients
}

listen {
      ipaddr = *
#     ipv6addr = ::
      port = 1813
      type = acct
#     interface = eth0
#     clients = per_socket_clients
}
```

La configuración final se muestra en el Anexo 1.

Users

Editamos el archivo "users". Este fichero es donde se definen los nombres de los usuarios y el método de autenticación que estos utilizarán. En el archivo original se nos muestran otros comandos los cuales pueden definir diferentes políticas para los usuarios (por ejemplo, limitar a unas direcciones *IP*'s cada cliente o definir unos puertos accesibles). Nosotros usaremos los siguientes para cumplir nuestros requerimientos (Anexo 2):

root@debian:/# gedit /etc/freeradius/users

Quedando de la siguiente manera:

```
usuario Cleartext-Password := "123456"
Service-Type = Framed-User,
Framed-Protocol = PPP,
# Framed-IP-Address = 172.16.3.33,
# Framed-IP-Netmask = 255.255.255.0,
# Framed-Routing = Broadcast-Listen,
# Framed-Filter-Id = "std.ppp",
# Framed-MTU = 1500,
Framed-Compression = Van-Jacobsen-TCP-IP

usuario1 Auth-Type := EAP
```

```
usuario2 Auth-Type := reject
```

En el caso del usuario1, este tendrá la posibilidad de autenticarse con un certificado válido a nombre de usuario1. En el caso del usuario2, a este se le denegará el acceso aún teniendo un certificado válido. Añadiendo la posibilidad de marcar a un usuario con el parámetro reject logramos denegar el acceso temporalmente, sin la necesidad de revocar el certificado.

*** La primera prueba que hacemos es habilitar el programa en modo debug:

root@debian:/# /etc/init.d/freeradius stop

root@debian:/# freeradius –X

root@debian:/#radtest usuario 1234561 127.0.0.1 1812 testing123

Si todo está bien, veremos esta información:

```
Sending Access-Request of id 174 to 127.0.0.1 port 1812
User-Name = "usuario"
User-Password = "123456"
NAS-IP-Address = 127.0.1.1
NAS-Port = 1812
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812,
id=174, length=38
Service-Type = Framed-User
Framed-Protocol = PPP
Framed-Compression = Van-Jacobson-TCP-IP
```

Debemos asegurarnos de recibir un "Access-Accept" del request que se ha enviado.

clients.conf

En este fichero hemos de definir los clientes que realizará la petición de confirmación de autenticación, es decir, los puntos de acceso (no los clientes inalámbricos). La definición del cliente incluirá la IP habilitada para cada uno de estos puntos de acceso, el secreto compartido entre el punto de acceso y el servidor RADIUS y el nombre del punto de acceso.

En las siguientes líneas se muestra como debería ser definido un cliente con nombre “dd-wrt”, dirección IP 192.168.2.1 y secreto compartido la palabra “testing123” (ver Anexo 3).

```
client localhost {
secret = testing123
shortname = localhost
nastype = other
}

client 192.168.2.1 {
#     # secret and password are mapped through the "secrets" file.
      secret      = testing123
      shortname   = dd-wrt
#        # the following three fields are optional, but may be used
by
#        # checkrad.pl for simultaneous usage checks
      nastype     = other
#     login       = !root
#     password    = someadminpas ipaddr = 127.0.0.1

}
```

eap.conf

El fichero eap.conf, se utiliza para configurar los procesos de autenticación basados en los métodos EAP, tunelados como no-tunelados.

El primer campo que hemos de modificar lo encontramos en la sección EAP (ver Anexo 4):

default_eap_type = tls

El siguiente paso es dirigirse a la sección TLS. En esta sección indicaremos, entre otras cosas, donde se encuentra la clave pública y privada del certificado de servidor y la clave pública de la CA. En este apartado tendremos que descomentar la línea donde indica tls {y la línea final de este apartado,}. Además modificaremos las líneas aquí indicadas:

```
1: tls {
                #
                #  These is used to simplify later configurations.
                #
2:              certdir = /etc/certs/server
```

```
                cadir = ${confdir}/certs/masterCA

3:              private_key_password = misecreto
4:              private_key_file =
/etc/certs/server/radius1_keycert.pem

                #  If Private key & Certificate are located in
                #  the same file, then private_key_file &
                #  certificate_file must contain the same file
                #  name.
                #
                #  If CA_file (below) is not used, then the
                #  certificate_file below MUST include not
                #  only the server certificate, but ALSO all
                #  of the CA certificates used to sign the
                #  server certificate.

8:              certificate_file =
/etc/certs/server/radius1_keycert.pem
9:              CA_file = ${cadir}/cacert.pem

                #
                #  For DH cipher suites to work, you have to
                #  run OpenSSL to create the DH file first:
                #
                #    openssl dhparam -out certs/dh 1024
                #

12:             dh_file = /etc/certs/dh
13:             random_file = /etc/cets/random
                rsa_key_length = 1024
                dh_key_length = 1024
```

En la línea 2 indicamos el path donde encontrar la lista de certificados no válidos (CRL) que se crearan en el apartado de OpenSSL.

En la línea 4 indicamos donde encontrar la clave privada del certificado del servidor que hemos generado y en la línea 3 la contraseña para descifrar dicha clave privada. En la línea 8 indicamos donde encontrar el certificado con clave pública del servidor (en este caso el certificado ha sido generado en el formato PEM incluyendo la clave pública y clave privada, esta última cifrada mediante triple DES, con la contraseña “testing123”).

En la línea 9 indicamos donde encontrar el certificado público de la entidad que valida nuestro certificado de servidor y los certificados de los clientes, es decir

el certificado de CA. Hemos de decir que el certificado de servidor no puede llevar incrustado el certificado de la CA (como se hace en el proceso de generación de P12 para clientes).

Esto es debido a que FreeRadius necesita de la clave pública de la CA de manera individual.

En las líneas 12 y 13 se indican los ficheros que harán de semilla (o seed) es para inicializar la clave de cifrado (los ficheros han de ser de contenido aleatorio) y que generado con OpenSSL.

4.7 Configuración de RADIUS con MySQL

En esta sección vamos a configurar MySQL para que sea capaz de gestionar todos los requerimientos de almacenamiento de FreeRADIUS, entre los que incluyen:

- Registro de usuarios y credenciales.
- Registro de grupos de usuarios.
- Registro de clientes o NAS para almacenar la configuración de los clientes de autenticación o autenticadores.
- Los ámbitos de direcciones IP para la concesión de direcciones a los suplicantes.
- Registro de los datos de accouting de las sesiones de usuario.
- Registro del log de conexiones al servidor, tanto correctas como fallidas.

Para ello vamos a crear la base de datos de RADIUS, sus tablas y campos relacionados. Para evitar que tengamos que estar campo a campo creando la estructura de forma manual, FreeRADIUS incorpora los scripts SQL necesarios para poder automatizar esta tarea.

1.- Para accesar a la administración de MySQL vía CLI hacemos lo siguiente:

root@debian:/# mysql -u root –p

2.- Creamos la base de datos.

```
mysql> CREATE DATABASE radius;
```

3.- Declararamos un usuario para la base de datos:

```
mysql> GRANT ALL ON radius.* TO radius@localhost IDENTIFIED BY "labserver";
```

En este caso asumimos que la Base de datos se llama "radius" y creamos un usuario también "radius" con password "labserver"

4.- Salimos de la base de datos

```
mysql> exit;
```

5.- Ahora metemos algunas tablas en la base de Datos:

Las tablas de ejemplo están dentro del directorio: /etc/freeradius/sql/mysql/

root@debian:/# cd /etc/freeradius/sql/mysql/

root@debian:/etc/freeradius/sql/mysql# mysql -u root -p radius < admin.sql

root@debian:/etc/freeradius/sql/mysql# mysql -u root -p radius < ippool.sql

root@debian:/etc/freeradius/sql/mysql# mysql -u root -p radius < nas.sql

root@debian:/etc/freeradius/sql/mysql# mysql -u root -p radius < schema.sql

Para ver las tablas creadas podemos hacer lo siguiente:

root@debian:/# mysql -u root –p

```
mysql> use radius;

mysql> show tables;
```

Aparecerá algo como esto,

```
+------------------+
| Tables_in_radius |
+------------------+
| nas |
| radacct |
| radcheck |
| radgroupcheck |
| radgroupreply |
| radippool |
| radpostauth |
| radreply |
| radusergroup |
+------------------+
9 rows in set (0.00 sec)
```

Salimos de la base con la siguiente instrucción:

```
mysql> quit;
```

6.- Luego editamos el archivo **/etc/freeradius/sql.conf**

root@debian:/# gedit /etc/freeradius/sql.conf

Configuramos los setting para la conexión con el server de MySQL:

```
# Connection info:
server = "localhost"
login = "radius"
password = "labserver"
```

Y descomentamos la variable: readclients = yes

La configuración final se muestra en el Anexo 5

7.- Luego Editamos el Archivo: **/etc/freeradius/sites-available/default** (anexo 6) y agregamos la variable "sql" en las secciones de.: **authorize**{}, **accounting**{}, *session*{}, *post-auth*{} esto para traer los datos desde las tablas en la base de datos "radius".

root@debian:/# gedit /etc/freeradius/sites-available/default

```
 authorize {
       preprocess
       chap
       mschap
       suffix
       eap
       sql
       pap
}
 accounting {
       detail
       sql
 }
```

8.-Luego podemos insertar un usuario en la base de datos de la siguiente forma:

root@debian:/# mysql -u root –p

```
mysql> use radius;
mysql> INSERT INTO radcheck (UserName, Attribute, Value) VALUES
('usuario1', 'Password', 'password1');
mysql> select * from radcheck where UserName='usuario1';
```

En este caso debe de aparecer el usuario que hemos creado.

Para probar que todo está ok.

root@debian:/# freeradius –X

root@debian:/# radtest usuario1 password1 127.0.0.1 1812 testing123

Debe aparecer un resultado más o menos como este:

root@daloradius:~# radtest usuario1 password1 127.0.0.1 1812 testing123

```
Sending Access-Request of id 215 to 127.0.0.1 port 1812
User-Name = "usuario1"
User-Password = "password1"
NAS-IP-Address = 127.0.1.1
NAS-Port = 1812
```

```
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812,
id=215, length=20
```

4.8 Instalación de OpenSSL

En condiciones normales, en la mayor parte de las distribuciones basadas en Linux, debemos instalar OpenSSL para poder hacer uso de este programa y de sus librerías asociadas. Uno de los motivos por los que hemos decidido incluido Debian con LAMP es porque nos hemos ahorrado gran cantidad de tiempo en la instalación y configuración de muchas herramientas y servicios que vamos a necesitar. Es el caso de OpenSSL, que no vamos a tener que instalar, ya que durante la instalación del sistema y de las aplicaciones que hemos ido instalando se ha instalado automáticamente.

En esta sección configuramos los parámetros por defecto de OpenSSL para la creación de una infraestructura PKI.

4.9 Configuración de OpenSSL

Antes de utilizar OpenSSL para toda la gestión de certificados, conviene configurar los parámetros por defecto para la creación de los nuevos certificados que vamos a emitir. De esta manera nos evitamos de tener que introducir los datos fijos relacionados de nuestra empresa, cada vez que precisemos de una nueva operación relacionada con certificados. Además podemos configurar valores como la validez en días de los certificados emitidos, o las rutas donde se guardarán todos los archivos generados, así como los nombres de archivos que vamos a utilizar durante todo el proceso.

1. Editamos openssl.conf almacenado en el directorio /etc/ssl.

root@daloradius:~# gedit /etc/ssl/openssl.cnf

```
# ==========================================
# OpenSSL configuration file
# ==========================================

RANDFILE         = /path/to/ca/.rnd

[ ca ]
default_ca       = CA_default

[ CA_default ]
dir              =  ./masterCA   #  where everything is kept
certs            = $dir/certs
new_certs_dir    = $dir/newcerts
crl_dir          = $dir/crl
database         = $dir/index.txt
private_key      = $dir/private/cakey.pem
certificate      = $dir/cacert.pem
serial           = $dir/serial
crl              = $dir/crl.pem
RANDFILE         = $dir/private/.rand
default_days     = 730
default_crl_days = 30
default_md       = sha1
preserve         = no
policy           = policy_anything
name_opt         = ca_default
cert_opt         = ca_default

[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ req ]
default_bits            = 1024
default_md              = sha1
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
x509_extensions         = v3_ca
string_mask             = nombstr

[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default         = MX
countryName_min             = 2
countryName_max             = 2
stateOrProvinceName         = State or Province Name (full name)
```

```
stateOrProvinceName_default     = DISTRITO FEDERAL

localityName                = Locality Name (eg, city)

0.organizationName          = Organization Name (eg, company)
0.organizationName_default = INSTITUTO POLITECNICO NACIONAL

# we can do this but it is not needed normally :-)
#1.organizationName        = Second Organization Name (eg, company)
#1.organizationName_default     = World Wide Web Pty Ltd

organizationalUnitName          = Organizational Unit Name (eg,
section)
#organizationalUnitName_default =

commonName            = Common Name (eg, YOUR name)
commonName_max              = 64

emailAddress                = Email Address
emailAddress_max      = 64

# SET-ex3             = SET extension number 3

[ req_attributes ]
challengePassword           = A challenge password
challengePassword_min       = 4
challengePassword_max       = 20

unstructuredName      = An optional company name


[ usr_cert ]
basicConstraints        = CA:FALSE
# nsCaRevocationUrl       = https://url-to-exposed-clr-
list/crl.pem

[ ssl_server ]
basicConstraints        = CA:FALSE
nsCertType              = server
keyUsage                = digitalSignature, keyEncipherment
extendedKeyUsage        = serverAuth, nsSGC, msSGC
#nsComment               = "OpenSSL Certificate for SSL Web
Server"

[ ssl_client ]
basicConstraints        = CA:FALSE
nsCertType              = client
keyUsage                = digitalSignature, keyEncipherment
extendedKeyUsage        = clientAuth
#nsComment               = "OpenSSL Certificate for SSL Client"
```

```
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage         = nonRepudiation, digitalSignature,
keyEncipherment

[ v3_ca ]
basicConstraints        = CA:true
nsCertType              = sslCA
keyUsage                = cRLSign, keyCertSign
extendedKeyUsage        = serverAuth, clientAuth
#nsComment               = "OpenSSL CA Certificate"

[ crl_ext ]
basicConstraints        = CA:FALSE
keyUsage                = digitalSignature, keyEncipherment
#nsComment               = "OpenSSL generated CRL"
```

2. Copiamos un script que incluye OpenSSL para la creación de certificados.

root@daloradius:/etc/freeradius/certs# cp/usr/lib/ssl/CA.sh./

3. Se modifica el script para cambiar algunas opciones por defecto por otras más adecuadas a nuestra implantación (Anexo 7).

root@daloradius:/etc/freeradius/certs# gedit CA.sh

```
if [ -z "$DAYS" ] ; then DAYS="-days 730" ; fi   # 2 year
CADAYS="-days 1095"   # 3 years
REQ="$OPENSSL req $SSLEAY_CONFIG"
CA="$OPENSSL ca $SSLEAY_CONFIG"
VERIFY="$OPENSSL verify"
X509="$OPENSSL x509"
PKCS12="openssl pkcs12"

if [ -z "$CATOP" ] ; then CATOP=./masterCA ; fi
CAKEY=./cakey.pem
CAREQ=./careq.pem
CACERT=./cacert.pem
```

4.10 Creación de la autoridad certificadora raíz

En esta sección nos convertiremos en la Autoridad Certificadora raíz de nuestra organización, para poder emitir y gestionar todo el sistema de certificados que vamos a administrar. Este es el primer, y más importante, proceso del sistema

PKI. Vamos a ser capaces de gestionar nuestros propios certificados de servidores y de clientes para cualquier comunicación cifrada que utilice nuestra organización.

El principal problema de ser nuestra propia Autoridad Certificadora, y no utilizar Autoridades Certificadoras privadas, es que los sistemas operativos clientes no incluyen en sus listas de identidades emisoras de confianza a nuestra CA. Esto simplemente significa que debemos introducir manualmente los certificados de nuestra CA en cada uno de los equipos que deban confiar en nosotros.

La parte pública del certificado, o sea el propio certificado de CA raíz si su clave privada asociada, será posteriormente utilizada para su instalación en los repositorios de certificados de todos los clientes, que deseemos que confíen en nuestra CA. Copiaremos la clave pública del certificado de CA raíz en todos los equipos, en el almacén de “Entidades raíz de confianza”.

- A continuación creamos el certificado raíz de la Autoridad Certificadora como sigue:

root@debian:/etc/freeradius/certs# ./CA.sh -newca

```
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 1024 bit RSA private key
....++++++
.....................................++++++
writing new private key to './masterCA/private/./cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
Verify failure
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

```
-----
Country Name (2 letter code) [MX]:
State or Province Name (full name) [DISTRITO FEDERAL]:
Locality Name (eg, city) []:df
Organization Name (eg, company) [INSTITUTO POLITECNICO NACIONAL]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:norma
Email Address []:ncruzg0901@ipn.mx

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:misecreto
An optional company name []:
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./masterCA/private/./cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            ae:9a:ab:4a:b8:9d:c8:66
        Validity
            Not Before: Oct 22 19:28:07 2011 GMT
            Not After : Oct 21 19:28:07 2014 GMT
        Subject:
            countryName               = MX
            stateOrProvinceName       = DISTRITO FEDERAL
            organizationName          = INSTITUTO POLITECNICO
NACIONAL
            commonName                = norma
            emailAddress              = ncruzg0901@ipn.mx
        X509v3 extensions:
            X509v3 Subject Key Identifier:

03:41:99:6C:F5:53:2E:AF:CA:82:13:22:50:3C:73:AE:CC:00:7A:26
            X509v3 Authority Key Identifier:

keyid:03:41:99:6C:F5:53:2E:AF:CA:82:13:22:50:3C:73:AE:CC:00:7A:26
                DirName:/C=MX/ST=DISTRITO FEDERAL/O=INSTITUTO
POLITECNICO NACIONAL/CN=norma/emailAddress=ncruzg0901@ipn.mx
                serial:AE:9A:AB:4A:B8:9D:C8:66

            X509v3 Basic Constraints:
                CA:TRUE
Certificate is to be certified until Oct 21 19:28:07 2014 GMT
(1095 days)

Write out database with 1 new entries
Data Base Updated
```

4.11. Generación del certificado de servidor

Una vez que nos hemos convertido en una entidad emisora raíz de certificados, ahora podemos emitir certificados de todo tipo, entre los que destacamos los de cliente y los de servidor, que van a ser los que vamos a utilizar para la autenticación de usuarios y servidores. Además emitiremos certificados para nuestro servidor de página Web Apache.

☼ Solicitamos a la Autoridad Certificadora un nuevo certificado de servidor para nuestra máquina, y en especial para su uso para el servidor Radius. Siempre debemos generar una solicitud de certificado (CSR) antes de proceder a firmar el certificado definitivo. En nuestro script, -newreq significa: nueva solicitud.

root@debian:/etc/freeradius/certs# ./CA.sh –newreq

```
Generating a 1024 bit RSA private key
.....................................++++++
.....................++++++
writing new private key to 'newkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [MX]:
State or Province Name (full name) [DISTRITO FEDERAL]:
Locality Name (eg, city) []:df
Organization Name (eg, company) [INSTITUTO POLITECNICO NACIONAL]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:radius1.midominio.loc
Email Address []:ncruzg0901@ipn.mx

Please enter the following 'extra' attributes
```

```
to be sent with your certificate request
A challenge password []:misecreto
An optional company name []:
Request is in newreq.pem, private key is in newkey.pem
```

☼ Firmado por la Autoridad Certificadora la solicitud de certificado (CSR) para el servidor radius1. En nuestro script lo haremos mediante el parámetro –sign (firmar).

root@debian:/etc/freeradius/certs# ./CA.sh –sign

```
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./masterCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            ae:9a:ab:4a:b8:9d:c8:67
        Validity
            Not Before: Oct 22 19:37:06 2011 GMT
            Not After : Oct 21 19:37:06 2013 GMT
        Subject:
            countryName               = MX
            stateOrProvinceName       = DISTRITO FEDERAL
            localityName              = df
            organizationName          = INSTITUTO POLITECNICO
NACIONAL
            commonName                = radius1.midominio.loc
            emailAddress              = ncruzg0901@ipn.mx
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:

22:28:D1:58:DC:24:A7:C4:3F:96:52:00:C6:47:62:0C:96:7B:4E:03
            X509v3 Authority Key Identifier:

keyid:03:41:99:6C:F5:53:2E:AF:CA:82:13:22:50:3C:73:AE:CC:00:7A:26

Certificate is to be certified until Oct 21 19:37:06 2013 GMT (730
days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Certificate:
```

```
    Data:
        Version: 3 (0x2)
        Serial Number:
            ae:9a:ab:4a:b8:9d:c8:67
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=MX, ST=DISTRITO FEDERAL, O=INSTITUTO POLITECNICO
NACIONAL, CN=norma/emailAddress=ncruzg0901@ipn.mx
        Validity
            Not Before: Oct 22 19:37:06 2011 GMT
            Not After : Oct 21 19:37:06 2013 GMT
        Subject: C=MX, ST=DISTRITO FEDERAL, L=df, O=INSTITUTO
POLITECNICO NACIONAL,
CN=radius1.midominio.loc/emailAddress=ncruzg0901@ipn.mx
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:c4:c6:48:33:ab:0c:36:a6:be:d2:6d:41:fb:31:
                    ad:db:60:e1:7d:4f:e9:de:12:96:0f:e9:36:cc:44:
                    ad:a7:7e:7b:53:7d:19:cf:65:41:7e:4c:a8:b0:24:
                    3d:c2:9a:17:6c:60:d9:cd:69:8b:d1:fc:75:8d:35:
                    d6:f2:01:ce:0c:a9:1c:65:0d:29:c0:63:95:ab:14:
                    aa:39:26:f3:28:95:19:98:2b:d7:76:06:b2:8c:0c:
                    08:61:fa:dc:1b:d5:7c:3b:de:b5:54:d6:a0:73:95:
                    bc:9b:b1:66:ef:c7:42:0c:a5:6d:c1:af:fc:6d:9c:
                    b6:be:08:a1:03:28:ea:4e:d1
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:

22:28:D1:58:DC:24:A7:C4:3F:96:52:00:C6:47:62:0C:96:7B:4E:03
            X509v3 Authority Key Identifier:

keyid:03:41:99:6C:F5:53:2E:AF:CA:82:13:22:50:3C:73:AE:CC:00:7A:26

    Signature Algorithm: sha1WithRSAEncryption
        47:71:aa:48:cc:33:fc:c0:81:69:b9:51:0b:0d:d2:f4:02:37:
        af:73:14:e7:12:8f:4c:7f:7a:30:d9:25:c6:0b:62:03:87:dc:
        56:a5:c4:63:64:10:9b:ce:87:3f:33:c8:38:fd:1f:3c:ae:bf:
        ce:92:22:df:81:d2:55:f0:c2:95:34:d6:8f:c5:92:11:4a:89:
        9f:6f:7e:f6:95:c7:6a:5c:9b:3e:44:9d:9f:ef:fb:56:48:14:
        23:56:f9:dc:77:c4:cc:7f:7a:51:27:3e:53:99:5f:fa:f2:52:
        3d:0c:be:7e:f7:44:e7:ee:67:3a:2d:91:00:d2:79:e3:c3:25:
        5f:a9
-----BEGIN CERTIFICATE-----
MIIDHjCCAoegAwIBAgIJAK6aq0q4nchnMA0GCSqGSIb3DQEBBQUAMIGDMQswCQYD
VQQGEwJNWDEZMBcGA1UECBMQRElTVFJJVE8gRkVERVJBTDEnMCUGA1UEChMeSU5T
```

```
VElUVVRPIFBPTElURUNOSUNPIE5BQ0lPTkFMMQ4wDAYDVQQDEwVub3JtYTEgMB4G
CSqGSIb3DQEJARYRbmNydXpnMDkwMUBpcG4ubXgwHhcNMTExMDIyMTkzNzA2WhcN
MTMxMDIxMTkzNzA2WjCBoDELMAkGA1UEBhMCTVgxGTAXBgNVBAgTEERJU1RSSVRP
IEZFREVSQUwxCzAJBgNVBAcTAmRmMScwJQYDVQQKEx5JTlNUSVRVVE8gUE9MSVRF
Q05JQ08gTkFDSU9OQUwxHjAcBgNVBAMTFXJhZGl1czEubWlkb21pbmlvLmxvYzEg
MB4GCSqGSIb3DQEJARYRbmNydXpnMDkwMUBpcG4ubXgwgZ8wDQYJKoZIhvcNAQEB
BQADgY0AMIGJAoGBAMTGSDOrDDamvtJtQfsxrdtg4X1P6d4Slg/pNsxErad+e1N9
Gc9lQX5MqLAkPcKaF2xg2c1pi9H8dY011vIBzgypHGUNKcBjlasUqjkm8yiVGZgr
13YGsowMCGH63BvVfDvetVTWoHOVvJuxZu/HQgylbcGv/G2ctr4IoQMo6k7RAgMB
AAGjezB5MAkGA1UdEwQCMAAwLAYJYIZIAYb4QgENBB8WHU9wZW5TU0wgR2VuZXJh
dGVkIENlcnRpZmljYXRlMB0GA1UdDgQWBBQiKNFY3CSnxD+WUgDGR2IMlntOAzAf
BgNVHSMEGDAWgBQDQZls9VMur8qCEyJQPHOuzAB6JjANBgkqhkiG9w0BAQUFAAOB
gQBHcapIzDP8wIFpuVELDdL0AjevcxTnEo9Mf3ow2SXGC2IDh9xWpcRjZBCbzoc/
M8g4/R88rr/OkiLfgdJV8MKVNNaPxZIRSomfb372lcdqXJs+RJ2f7/tWSBQjVvnc
d8TMf3pRJz5TmV/68lI9DL5+90Tn7mc6LZEA0nnjwyVfqQ==
-----END CERTIFICATE-----
Signed certificate is in newcert.pem
```

☼ Para el caso de Windows se requiere del certificado raíz de nuestra CA en formato DER y no PEM, vamos a convertirlo en este formato y así disponer de las dos opciones.

root@debian:/etc/freeradius/certs# cd masterCA/

root@debian:/etc/freeradius/certs# openssl x509 – inform PEM –out form DER –in cacert.pem –out cacert.der

root@debian:/etc/freeradius/certs# cp cacert.der /etc/certs

☼ Generamos el fichero random y el fichero diffie-helman dh (estos dos ficheros son utilizados para el establecimiento de las sesiones TLS y para la encriptación):

root@debian:/etc/freeradius/certs# dd if=/dev/urandom of=random count=2

```
2+0 records in
2+0 records out
1024 bytes (1,0 kB) copied, 0,000771455 seconds, 1,3 MB/s
```

root@debian:/etc/freeradius/certs# openssl dhparam -out dh 1024

```
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
```

☼ Creamos un fichero de texto con las extensiones OID necesarias para la creación del certificado de cliente para Windows. Windows necesita que los certificados tengan unas extensiones específicas o si no, los rechazará.

root@debian:/etc/freeradius/certs# gedit xpextensions

```
[ xpclient_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2
nsCertType = client
[ xpserver_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1,clientAuth
nsCertType = server
```

☼ Concatenando en un solo fichero el certificado (con su clave pública) y su clave privada.

root@debian:/etc/freeradius/certs# cat newkey.pem newcert.pem>radius1_keycert.pem

root@debian:/etc/freeradius/certs# cat radius1_keycert.pem

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,78D3F3EADD9A8C8F

QisQ/EpiKkVolq9XiD+A5MZ2mpHL4w8HzweWGr+Ukgepy7MLLIh1OpUwo/qRQnN/
1dK4JRvGsUNo5yDhKHjPz/4qPUwYzqTVxJuptRXyaikPUAMrkJQ58/jhZwJsjQqc
IOt/rFaIg7VWIwVVEBedDM4D32VSFxA4trO1qwB+8vTjzwZAyGsVYQ8lVkoXq5lA
vuBHi3bgEUsMFuhPeBuNyHPqpdhcH6Jpn9E+MeLxZmZ2jRUvPKQAdq4ZG7OFaud4
Gf1rUmEHt1X77J7pyy2aPanwaJIYEH3hv8gOOZyUKH2nO58oWEwUjuFqV1mm9w8I
uKo2y2Dxi1rHaOJuC00fL9BSlkO4gVkPU45/Sio5iMvROifUnX+7e6rrJ8+8RwET
xjIhEGoqQhVae++VkT20+yvbZFvkY1AIN1KjlK9WltJ5HkIghuwGpThBDxkqZjqr
jsKNAxGU0T2TGQwhxBJx6oz5S9taHO50Fj458rdFhEBSsfTs6E4J6JPx7BzhKKvc
OTGzVXsrWSzdyQGksZhpUi4NeUrPoGQ002XfnIqtDxmsglDSOhQ5ObBejxWiUg+D
4JUOEwiha2MKDE2IO6v0cU6q9zrBV3N0YP8O+9hEHrg6+sqX9yws4FrJHcnNBb/S
X2pNxrxQ6ydaeVq7oQXNso/jElD7iEcPRPsCVrYcDrfLofIOIE4tPFH5+JYsZviX
gQVrZeUgY7FePaPWQsJwKV1cxdTo+UBsft2Izu72Dn5lZVHTOAmGWMjbQuHcayv3
BkFoRdquawU5Au5puiBuCNBxO0QAncJx2GNYN0SjGFZSccP2izdtVw==
-----END RSA PRIVATE KEY-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            e9:45:8f:aa:e2:95:be:40
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=MX, ST=DISTRITO FEDERAL, O=INSTITUTO POLITECNICO
NACIONAL, CN=norma/emailAddress=ncruzg0901@ipn.mx
        Validity
            Not Before: Oct 23 00:55:36 2011 GMT
```

```
            Not After : Oct 22 00:55:36 2013 GMT
        Subject:  C=MX,  ST=DISTRITO  FEDERAL,  L=df,  O=INSTITUTO
POLITECNICO                                                  NACIONAL,
CN=radius1.midominio.loc/emailAddress=ncruzg0901@ipn.mx
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:99:43:3c:34:ef:5d:a5:aa:de:9e:17:91:ff:b3:
                    65:7a:7b:71:e4:e8:2e:9a:11:49:db:be:1c:84:b0:
                    54:aa:01:a7:72:e0:41:4f:7f:cd:01:65:56:44:a3:
                    28:d2:1c:ee:10:1c:31:82:7a:41:0c:ea:c3:42:78:
                    59:4f:24:cf:e3:f4:aa:e1:4b:f8:bc:7c:aa:b4:d1:
                    61:7b:f0:9f:92:cf:db:3d:e6:7b:a7:a8:bb:1d:24:
                    3d:f8:e4:d0:a5:de:cd:0b:0b:5e:af:c8:0b:ac:e5:
                    dc:ca:8b:6f:3d:3e:fc:7f:bc:85:cf:6b:02:39:f1:
                    d2:55:81:43:02:61:1e:16:6d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:

BF:BF:09:DB:72:A9:4E:79:CC:86:50:75:9E:E5:0C:DE:37:D6:C1:E6
            X509v3 Authority Key Identifier:

keyid:61:6A:E2:90:9D:07:35:9C:65:35:0D:05:9D:91:85:A1:36:1E:BE:8F

    Signature Algorithm: sha1WithRSAEncryption
        a5:11:f2:67:5c:e5:2c:65:bc:50:8e:6d:8a:e5:38:21:dc:15:
        2e:da:1c:bb:09:1e:f5:f7:45:49:49:d1:90:16:f7:ed:27:77:
        2e:06:70:02:51:81:0e:dd:4c:46:f4:2c:bd:61:dd:1d:c4:53:
        bd:d0:37:08:fc:c8:7a:1d:0b:15:97:46:5a:52:5a:f2:31:a2:
        8a:87:f0:c6:c9:44:84:bb:87:e9:5b:d6:65:40:80:fe:47:c9:
        54:cb:04:03:07:bf:70:ce:02:c3:76:1c:db:02:77:87:c9:8e:
        aa:11:f1:f6:4f:35:ce:cd:bf:6c:cf:5c:0f:e4:dc:d3:d5:fc:
        ff:ef
-----BEGIN CERTIFICATE-----
MIIDHjCCAoegAwIBAgIJAOlFj6rilb5AMA0GCSqGSIb3DQEBBQUAMIGDMQswCQYD
VQQGEwJNWDEZMBcGA1UECBMQRElTVFJJVE8gRkVERVJBTDEnMCUGA1UEChMeSU5T
VElUVVRPIFBPTElURUNOSUNPIE5BQ0lPTkFMMQ4wDAYDVQQDEwVub3JtYTEgMB4G
CSqGSIb3DQEJARYRbmNydXpnMDkwMUBpcG4ubXgwHhcNMTExMDIzMDA1NTM2WhcN
MTMxMDIyMDA1NTM2WjCBoDELMAkGA1UEBhMCTVgxGTAXBgNVBAgTEERJU1RSSVRP
IEZFREVSQUwxCzAJBgNVBAcTAmRmMScwJQYDVQQKEx5JTlNUSVRVVE8gUE9MSVRF
Q05JQ08gTkFDSU9OQUwxHjAcBgNVBAMTFXJhZGl1czEubWlkb21pbmlvLmxvYzEg
MB4GCSqGSIb3DQEJARYRbmNydXpnMDkwMUBpcG4ubXgwgZ8wDQYJKoZIhvcNAQEB
BQADgY0AMIGJAoGBAJlDPDTvXaWq3p4Xkf+zZXp7ceToLpoRSdu+HISwVKoBp3Lg
QU9/zQFlVkSjKNIc7hAcMYJ6QQzqw0J4WU8kz+P0quFL+Lx8qrTRYXvwn5LP2z3m
```

```
e6eoux0kPfjk0KXezQsLXq/IC6zl3MqLbz0+/H+8hc9rAjnx0lWBQwJhHhZtAgMB
AAGjezB5MAkGA1UdEwQCMAAwLAYJYIZIAYb4QgENBB8WHU9wZW5TU0wgR2VuZXJh
dGVkIENlcnRpZmljYXRlMB0GA1UdDgQWBBS/vwnbcqlOecyGUHWe5QzeN9bB5jAf
BgNVHSMEGDAWgBRhauKQnQc1nGU1DQWdkYWhNh6+jzANBgkqhkiG9w0BAQUFAAOB
gQClEfJnXOUsZbxQjm2K5Tgh3BUu2hy7CR7190VJSdGQFvftJ3cuBnACUYEO3UxG
9Cy9Yd0dxFO90DcI/Mh6HQsVl0ZaUlryMaKKh/DGyUSEu4fpW9ZlQID+R8lUywQD
B79wzgLDdhzbAneHyY6qEfH2TzXOzb9sz1wP5NzT1fz/7w==
-----END CERTIFICATE-----
```

☼ Copiamos y organizamos los ficheros necesarios para nuestro servidor radius a una localización controlada, a fin de mantenerlos seguros y ordenados.

root@debian:/etc/freeradius/certs# mkdir –p /etc/certs/server

root@debian:/etc/freeradius/certs# mkdir –p /etc/certs/clients

root@debian:/etc/freeradius/certs# cp radius1_keycert.pem /etc/certs/server/

root@debian:/etc/freeradius/certs# cp dh random /etc/certs

root@debian:/etc/freeradius/certs# cp masterCA/cacert.pem /etc/certs

4.12. Generación del certificado del cliente

En esta sección, vamos a crear un certificado de cliente para la autenticación de un cliente Windows mediante el protocolo 802.x.

Debemos generar un certificado único para cada usuario o equipo que se deba conectar a nuestra red mediante autenticación por certificados de cliente, como es el caso de EAP-TLS. En este caso vamos a crear los certificados en el formato necesario, mediante el comando openssl.

✓ Generamos la CSR del certificado de cliente, mediante el argumento req (request). Durante la creación de una CSR se crea también la clave privada. Esta clave privada va cifrada mediante el uso de la contraseña misecreto.

root@debian:/etc/freeradius/certs# openssl req -new -keyout newreq.pem -out newreq.pem -passin pass:misecreto -passout pass:misecreto

```
Generating a 1024 bit RSA private key
.....................................................................
.....................................................................
.............................++++++
..++++++
writing new private key to 'newreq.pem'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [MX]:
State or Province Name (full name) [DISTRITO FEDERAL]:
Locality Name (eg, city) []:df
Organization Name (eg, company) [INSTITUTO POLITECNICO NACIONAL]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:cliente windows 1
Email Address []:ncruzg0901@ipn.mx
```

✓ Firmamos y generamos el certificado con las extensiones OID de cliente de Windows, mediante el fichero de entrada newreq.pem y generando el fichero de salida newcert.pem. Hemos indicado las claves de cifrado de la CSR (misecreto) para que la pueda cifrar. Al finalizar nos pregunta si deseamos firmar el certificado, a lo que contestamos que sí.

root@debian:/etc/freeradius/certs# openssl ca -policy policy_anything -passin pass:misecreto -infiles newreq.pem -extfile /etc/freeradius/certs/xpextensions -extensions xpclient_ext -out newcert.pem

```
Using configuration from /usr/lib/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            e9:45:8f:aa:e2:95:be:41
        Validity
            Not Before: Oct 23 02:21:15 2011 GMT
            Not After : Oct 22 02:21:15 2013 GMT
        Subject:
            countryName               = MX
            stateOrProvinceName       = DISTRITO FEDERAL
            localityName              = df
```

```
            organizationName          = INSTITUTO POLITECNICO
NACIONAL
            commonName                = cliente windows 1
            emailAddress              = ncruzg0901@ipn.mx
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:

4F:BE:C6:0F:93:85:EC:57:41:DB:53:09:8F:FD:CA:C5:41:94:20:D5
            X509v3 Authority Key Identifier:

keyid:61:6A:E2:90:9D:07:35:9C:65:35:0D:05:9D:91:85:A1:36:1E:BE:8F

Certificate is to be certified until Oct 22 02:21:15 2013 GMT (730
days)
Sign the certificate? [y/n]:y

-extfile: No such file or directory
3884:error:02001002:system library:fopen:No such file or
directory:bss_file.c:356:fopen('-extfile','r')
3884:error:20074002:BIO routines:FILE_CTRL:system
lib:bss_file.c:358:
```

- ✓ Utilizamos ahora el formato PKCS#12.

root@debian:/etc/freeradius/certs# openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -out clientcert.p12 -clcerts

```
Enter pass phrase for newreq.pem:
Enter Export Password:
Verifying - Enter Export Password:
```

root@debian:/etc/freeradius/certs# openssl pkcs12 -passin pass:misecreto -in clientcert.p12 -passout pass:misecreto -out clientcert.pem

```
MAC verified OK
```

- ✓ Exportamos el certificado codificado PEM a codificación DER:

root@debian:/etc/freeradius/certs# openssl x509 –inform PEM –outform DER –in clientcert.pem –out clientcert.der

4.13. Revocación de certificados

Después de una infraestructura PKI para el uso de AAA sobre Radius. Sin embargo, esta infraestructura precisa de un mantenimiento y de una seguridad

determinada, que a medio/largo plazo no sirve de mucho si no disponemos de la posibilidad de revocar certificados.

- Creamos el archivo crlnumber

root@debian:/etc/freeradius/certs# touch masterCA/crlnumber

- Generamos el certificado de entidad revocadora CRL mediante codificación PEM.

root@debian:/etc/freeradius/certs# openssl ca -gencrl -keyfile masterCA/private/cakey.pem -cert masterCA/cacert.pem -out crl.pem

```
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for masterCA/private/cakey.pem:

root@debian:/etc/freeradius/certs# cat crl.pem
-----BEGIN X509 CRL-----
MIIBSTCBszANBgkqhkiG9w0BAQUFADCBgzELMAkGA1UEBhMCTVgxGTAXBgNVBAgT
EERJU1RSSVRPIEZFREVSQUwxJzAlBgNVBAoTHklOU1RJVFVUTyBQT0xJVEVDTklD
TyBOQUNJT05BTDEOMAwGA1UEAxMFbm9ybWExIDAeBgkqhkiG9w0BCQEWEW5jcnV6
ZzA5MDFAaXBuLm14Fw0xMTEwMjQwMTM4NTRaFw0xMTExMjMwMTM4NTRaMA0GCSqG
SIb3DQEBBQUAA4GBAGP0AxOK6z/Mm5wU/77dufE+7rTk/Ij3n/d97dmACxg+K0d8
YM+2TPqshfr2LXR/gA/tt8MMC9mNrZR2AjU29xAP4JbJHzt/3SHTqExMQF8u2AXH
MVy6jXbb5HKxYhr95Qh+y9kzZGy7NA3p8BgmToDQDMpwa1WM4sYjMf9+wqZp
-----END X509 CRL-----
```

4.14. Configurando Apache2

Apache es el servidor Web más utilizado. Su potencia, ligereza, robustez y seguridad lo han convertido en la mejor alternativa.

Como este es un equipo crítico, en lo que a seguridad se refiere, debemos reforzar la seguridad del acceso Web, y obligar a que se haga de ssl para la conexión. Por tanto, podemos deshabilitar el acceso a través del puerto 80 mediante HTTP y obligar a utilizar otros puertos para HTTPS como el 443.

- Vamos a configurar Apache2 para utilizar únicamente seguridad SSL. Para ello editamos el fichero ports.conf del directorio de configuración de Apache2 que es /etc/apache2/.

root@debian:/# gedit /etc/apache2/ports.conf

```
# If you just change the port or add more ports here, you will
likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default
# This is also true if you have upgraded from before 2.2.9-3 (i.e.
from
# Debian etch). See /usr/share/doc/apache2.2-common/NEWS.Debian.gz
and
# README.Debian.gz

NameVirtualHost *:80
Listen 80

<IfModule mod_ssl.c>
    # If you add NameVirtualHost *:443 here, you will also have to
change
    #    the    VirtualHost    statement    in    /etc/apache2/sites-
available/default-ssl
    # to <VirtualHost *:443>
    #  Server  Name  Indication  for  SSL  named  virtual  hosts  is
currently not
    # supported by MSIE on Windows XP.
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

- Ahora habilitamos el módulo SSL para Apache2.

root@debian:/# ln -s /etc/apache2/mods-available/ssl. /etc/apache2/mods-enabled/*

- Se edita /etc/apache2/sites.enabled/000-default que es el fichero que representa el servidor.

```
NameVirtualHost *:443
<VirtualHost *:443>
ServerAdmin webmaster@midominio.loc
ServerName radius1.midominio.loc
ServerAlias radius1.midominio.loc www.midominio.loc
SSLEngine On
SSLOptions +FakeBasicAuth +ExportCertData
SSLCertificateFile /etc/certs/server/apache_cert.pem
SSLCertificatekeyFile /etc/certs/server/apache_key_insecure.pem

     DocumentRoot /var/www
     <Directory />
          Options FollowSymLinks
          AllowOverride None
```

```
      </Directory>
      <Directory /var/www/>
            Options Indexes FollowSymLinks MultiViews
            AllowOverride None
            Order allow,deny
            allow from all
      </Directory>

      ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
      <Directory "/usr/lib/cgi-bin">
            AllowOverride None
            Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
            Order allow,deny
            Allow from all
      </Directory>

      ErrorLog ${APACHE_LOG_DIR}/error.log

      # Possible values include: debug, info, notice, warn, error,
crit,
      # alert, emerg.
      LogLevel warn

      CustomLog ${APACHE_LOG_DIR}/access.log combined

    Alias /doc/ "/usr/share/doc/"
    <Directory "/usr/share/doc/">
        Options Indexes MultiViews FollowSymLinks
        AllowOverride None
        Order deny,allow
        Deny from all
        Allow from 127.0.0.0/255.0.0.0 ::1/128
    </Directory>

</VirtualHost>
```

♉ Después de configurar las rutas para que acceda al certificado de servidor. Ahora debemos crear el certificado de servidor Apache2.

root@debian:/# cd /etc/freeradius/certs

root@debian:/etc/freeradius/certs# ./CA.sh -newreq

```
Generating a 1024 bit RSA private key
.....................................++++++
.....++++++
writing new private key to 'newkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
```

```
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [MX]:
State or Province Name (full name) [DISTRITO FEDERAL]:
Locality Name (eg, city) []:df
Organization Name (eg, company) [INSTITUTO POLITECNICO NACIONAL]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:www.midominio.loc
Email Address []:ncruzg0901@ipn.mx

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:misecreto
An optional company name []:
Request is in newreq.pem, private key is in newkey.pem
```

- Firmamos la CSR y creamos el certificado:

root@debian:/etc/freeradius/certs# ./CA.sh -sign

```
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./masterCA/private/cakey.pem:
4088:error:28069065:lib(40):UI_set_result:result too
small:ui_lib.c:850:You must type in 4 to 8191 characters
Enter pass phrase for ./masterCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            f7:02:60:84:3f:35:73:0b
        Validity
            Not Before: Nov  8 06:02:59 2011 GMT
            Not After : Nov  7 06:02:59 2013 GMT
        Subject:
            countryName               = MX
            stateOrProvinceName       = DISTRITO FEDERAL
            localityName              = df
            organizationName          = INSTITUTO POLITECNICO
NACIONAL
            commonName                = www.midominio.loc
            emailAddress              = ncruzg0901@ipn.mx
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
```

```
            Netscape Cert Type:
                SSL Client, S/MIME
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:

D1:18:42:0C:61:A8:15:C8:5F:6F:A3:2A:A7:D4:E4:2B:78:6B:36:80
            X509v3 Authority Key Identifier:

keyid:5B:B2:04:7D:9B:0C:E0:C9:34:CD:B8:11:37:5B:A3:0F:87:3F:B1:92

Certificate is to be certified until Nov  7 06:02:59 2013 GMT (730
days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            f7:02:60:84:3f:35:73:0b
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=MX, ST=DISTRITO FEDERAL, O=INSTITUTO POLITECNICO
NACIONAL, CN=norma/emailAddress=ncruzg0901@ipn.mx
        Validity
            Not Before: Nov  8 06:02:59 2011 GMT
            Not After : Nov  7 06:02:59 2013 GMT
        Subject: C=MX, ST=DISTRITO FEDERAL, L=df, O=INSTITUTO
POLITECNICO NACIONAL,
CN=www.midominio.loc/emailAddress=ncruzg0901@ipn.mx
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d4:b1:f0:cc:80:0f:ad:62:b5:7a:21:ac:a7:a3:
                    ef:a8:a3:a6:0d:e9:61:ff:da:85:be:78:f2:69:39:
                    40:4c:0e:39:4c:fa:5e:b5:1d:12:68:c2:0a:ef:d5:
                    96:26:c4:f5:df:f8:b8:8f:ad:d5:69:cc:c6:64:8b:
                    01:12:87:2f:25:fe:b0:de:99:eb:e7:ea:47:f3:ef:
                    b3:a6:ae:82:c8:78:0f:07:c9:33:09:fd:84:07:e5:
                    46:72:1c:a0:be:9a:97:04:96:b8:e2:66:92:84:8d:
                    ca:14:96:84:51:f3:d6:10:3c:c0:f8:6a:6e:eb:f1:
                    88:a3:df:26:72:a8:c5:31:33
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
```

```
            SSL Client, S/MIME
        Netscape Comment:
            OpenSSL Generated Certificate
        X509v3 Subject Key Identifier:

D1:18:42:0C:61:A8:15:C8:5F:6F:A3:2A:A7:D4:E4:2B:78:6B:36:80
        X509v3 Authority Key Identifier:

keyid:5B:B2:04:7D:9B:0C:E0:C9:34:CD:B8:11:37:5B:A3:0F:87:3F:B1:92

    Signature Algorithm: sha1WithRSAEncryption
        40:a0:41:0d:0d:0f:ac:07:86:e1:b2:4f:7f:86:0b:90:17:01:
        51:d4:33:92:6a:fd:84:b1:45:48:68:a6:7e:86:23:14:81:09:
        c2:b4:a8:9b:55:0f:f9:c8:2b:9c:63:d7:5d:a7:3b:87:11:92:
        6d:9e:04:51:69:41:7c:fa:14:1b:96:7d:19:ef:a5:7c:cb:8c:
        43:45:d2:71:0d:7b:d8:70:09:87:2b:98:97:a1:7d:30:36:82:
        8c:e6:0f:31:25:7a:15:ef:97:b5:b7:e6:c7:6f:d8:38:4c:3d:
        81:b9:88:dd:46:c7:cf:d2:f0:be:0c:99:83:26:3c:8c:39:4e:
        b9:c9
-----BEGIN CERTIFICATE-----
MIIDLzCCApigAwIBAgIJAPcCYIQ/NXMLMA0GCSqGSIb3DQEBBQUAMIGDMQswCQYD
VQQGEwJNWDEZMBcGA1UECBMQRElTVFJJVE8gRkVERVJBTDEnMCUGA1UEChMeSU5T
VElUVVRPIFBPTElURUNOSUNPIE5BQ0lPTkFMMQ4wDAYDVQQDEwVub3JtYTEgMB4G
CSqGSIb3DQEJARYRbmNydXpnMDkwMUBpcG4ubXgwHhcNMTExMTA4MDYwMjU5WhcN
MTMxMTA3MDYwMjU5WjCBnDELMAkGA1UEBhMCTVgxGTAXBgNVBAgTEERJU1RSSVRP
IEZFREVSQUwxCzAJBgNVBAcTAmRmMScwJQYDVQQKEx5JTlNUSVRVVE8gUE9MSVRF
Q05JQ08gTkFDSU9OQUwxGjAYBgNVBAMTEXd3dy5taWRvbWluaW8ubG9jMSAwHgYJ
KoZIhvcNAQkBFhFuY3J1emcwOTAxQGlwbi5teDCBnzANBgkqhkiG9w0BAQEFAAOB
jQAwgYkCgYEA1LHwzIAPrWK1eiGsp6PvqKOmDelh/9qFvnjyaTlATA45TPpetR0S
aMIK79WWJsT13/i4j63VaczGZIsBEocvJf6w3pnr5+pH8++zpq6CyHgPB8kzCf2E
B+VGchygvpqXBJa44maShI3KFJaEUfPWEDzA+Gpu6/GIo98mcqjFMTMCAwEAAaOB
jzCBjDAJBgNVHRMEAjAAMBEGCWCGSAGG+EIBAQQEAwIFoDAsBglghkgBhvhCAQ0E
HxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFNEYQgxh
qBXIX2+jKqfU5Ct4azaAMB8GA1UdIwQYMBaAFFuyBH2bDODJNM24ETdbow+HP7GS
MA0GCSqGSIb3DQEBBQUAA4GBAECgQQ0ND6wHhuGyT3+GC5AXAVHUM5Jq/YSxRUho
pn6GIxSBCcK0qJtVD/nIK5xj112nO4cRkm2eBFFpQXz6FBuWfRnvpXzLjENF0nEN
e9hwCYcrmJehfTA2gozmDzElehXvl7W35sdv2DhMPYG5iN1Gx8/S8L4MmYMmPIw5
TrnJ
-----END CERTIFICATE-----
Signed certificate is in newcert.pem
```

ϒ Copiamos los certificados con su clave pública y de la clave privada a su ubicación definitiva, en nuestro repositorio.

ϒ

root@debian:/etc/freeradius/certs# cp newkey.pem /etc/certs/server/apache_key.pem

root@debian:/etc/freeradius/certs# cp newcert.pem /etc/certs/server/apache_cert.pem

4.15. Configuración del AP Linksys

En esta sección vamos a configurar el router AP Linksys WRT-54GS que utiliza un hardware compatible con Linux, por lo que permite cargar en su interior un firmware diferente al original, basado en un proyecto libre llamado DD-WRT que tiene una calidad excelente.

El aspecto del AP que vamos a utilizar se muestra en la Figura 4.2. Su aspecto no es demasiado profesional, pero su hardware y firmware son muy estables y completos.

Figura 4.2 Router WRT-54GS

La Figura 4.3 muestra la interfaz Web, con la configuración en nuestro caso:

Figura 4.3 Interfaz Web

En el menú RADIUS configuramos al equipo como cliente especifico de RADIUS para autenticación de clientes. Especificamos el formato en que se envía

la dirección MAC, la dirección IP y el puerto de escucha del servidor RADIUS además de su shared secret (Figura 4.4).

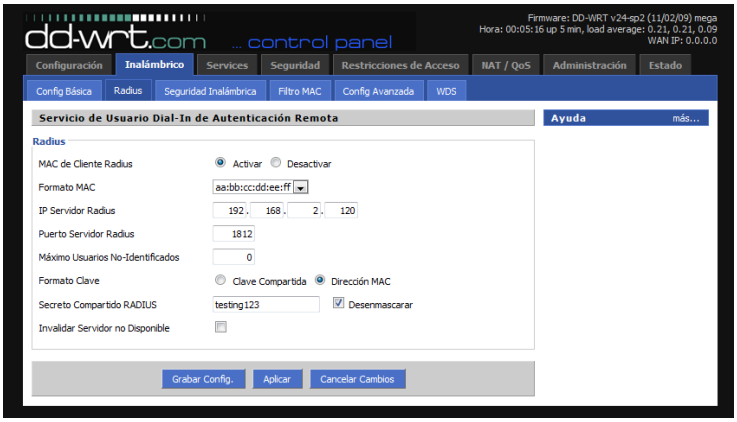

Figura 4.4 Menú RADIUS

En el menú Wireless Security (Figura 4.5) encontramos los parámetros que necesitamos. En nuestro caso configuramos WPA2 Enterprise con algoritmos TKIP y AES, para lograr mayor compatibilidad. Especificamos la dirección IP de nuestro servidor RADIUS y el puerto de autenticación a utilizar, además el secreto compartido.

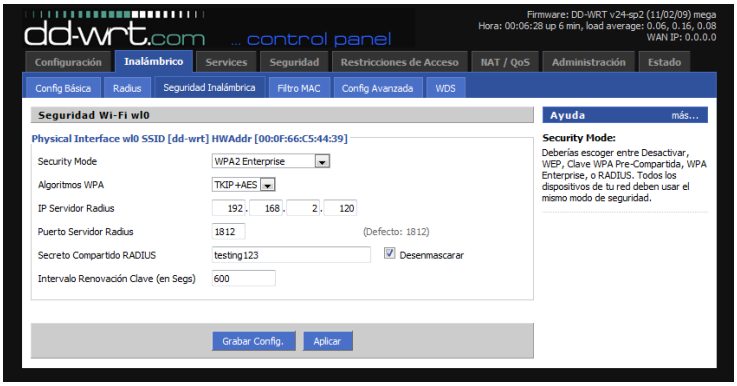

Figura 4.5 Menú Wireless Security

4.16 Referencias

[1] Project Newsflash. OpenSSL.org. http://www.openssl.org/news/

[2] DD-WRT Project. http://www.dd-wrt.com/

[3] Freeradius mod-eap isn't detected (Ticket 1837). FreeRadius.com. https://dev.openwrt.org/ticket/1837

CAPÍTULO 5
IMPLEMENTACIÓN Y PRUEBAS

5.1 Introducción

El entorno educativo es el medio donde se decidió probar el modelo debido a que es ahí donde se está desarrollando y en este momento existen las facilidades para poder hacer una pequeña implementación.

En los capítulos anteriores se describió el sistema, se diseñó su arquitectura. En este Capítulo se realizarán una serie de pruebas que ayudarán a determinar si el sistema cumple con su objetivo principal que es proporcionarle al usuario la identificación y autenticación a red mediante su dispositivo móvil.

Antes de realizar las pruebas se hace una descripción más a detalle del entorno educativo donde se implementó el caso de estudio. La Escuela Superior de Cómputo es una unidad académica del Instituto Politécnico Nacional que forma profesionales en sistemas computacionales a nivel licenciatura y posgrado, actualmente cuenta con un aproximado de 2000 alumnos en la licenciatura y 20 alumnos de posgrado. La oferta educativa en el área de posgrado es una Maestría en Ciencias en Sistemas Computacionales Móviles. Es en el laboratorio de Maestría donde se implementó el caso de estudio y donde se realizaron las pruebas sobre la implementación que se estarán presentando en el desarrollo del capítulo.

Es necesario mencionar que el espacio a cubrir sea dentro de este laboratorio (Figura 5.1).

El área del laboratorio es aproximadamente de 12 metros por 8 metros cuenta con 2 cubículos para profesores, además de un conjunto de mobiliario (mesas y sillas) y conexiones eléctricas para el apoyo a los estudiantes de un espacio donde puedan trabajar en sus diferentes proyectos y actividades académicas.

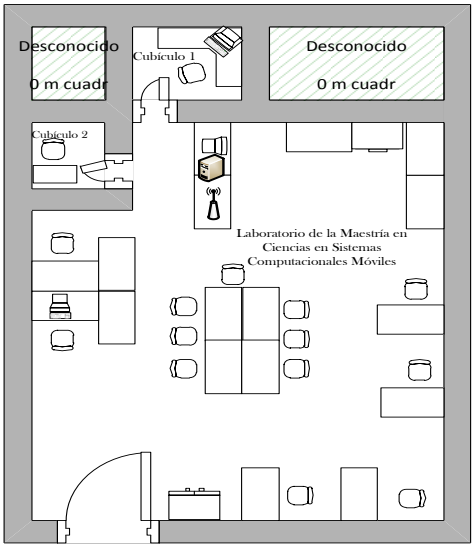

Figura 5. 1 Diagrama del Laboratorio de Posgrado dentro de ESCOM-IPN

5.2 Caso de estudio del Modelo

El caso de estudio que se propone para evaluar el sistema se centra sobre un entorno académico, en el cual se crea un dominio de red con servicios educativos. Este dominio ofrece los servicios educativos a usuarios de la comunidad educativa, mediante un dispositivo de interconexión inalámbrica, por lo que los usuarios deberán contar con un dispositivo móvil compatible con el dispositivo de interconexión de la red para poder acceder a los servicios. Cabe señalar que el modelo no está diseñado para trabajar con una tecnología de acceso específica, lo que se propone es solo un caso de estudio y se va a trabajar con una sola tecnología de acceso inalámbrico que es WiFi.

Previo a poder consultar o requerir los servicios, los usuarios deberán contactar al administrador del sistema para darse de alta en una base de datos que les dé acceso al dominio y así obtener los servicios.

5.3 Implementación del Caso de estudio del Modelo

El trabajo de implementación del caso de estudio se llevó a cabo en el laboratorio de la maestría de cómputo móvil de la Escuela Superior de Cómputo en el Instituto Politécnico Nacional. Es en esta área donde se dieron las facilidades para la implementación del modelo.

Un aspecto muy importante a considerar es que aunque se haya elegido el entorno educativo para probar el sistema se está situando solo dentro de un dominio pequeño dentro de este, como lo es el laboratorio de la maestría un espacio donde los alumnos realizan sus actividades académicas. Si se considera todo el entorno académico este se tendría que dividir en subdominios o subentornos como ejemplo: aulas, laboratorios, cafeterías, bibliotecas, etc.

Como primer paso se muestra el esquema de red (Figura 5.2) sobre el que se trabajará la implementación del caso de estudio, este esquema se basa puntualmente en la arquitectura del sistema que se propuso y definió.

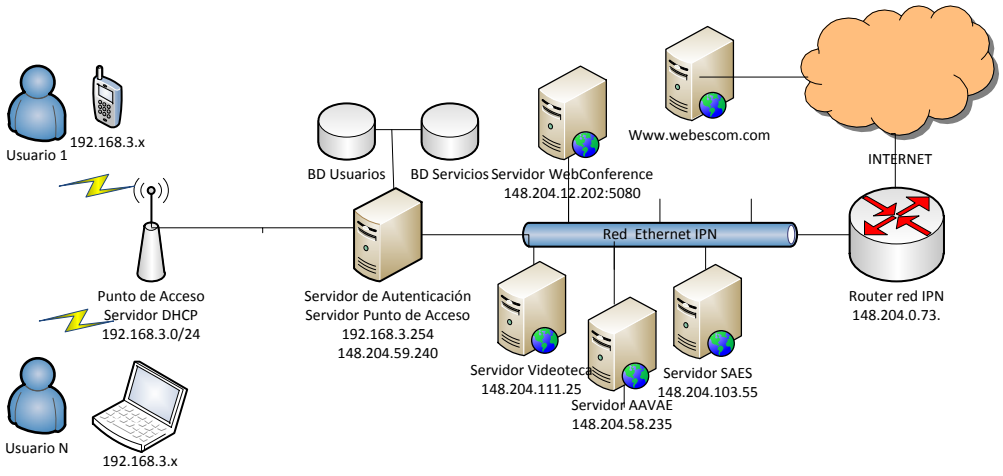

Figura 5. 2 Esquema de Red para caso de estudio

El esquema de red sobre el cual trabajará el sistema consta de una red convergente apoyada en el sistema de red institucional que está desplegado para

darle servicio a los diferentes usuarios dentro del Instituto Politécnico Nacional, tiene contemplados a más de 18,000 equipos de cómputo conectados al sistema de red, el backbone está integrado por tres nodos principales: Zacatenco, Santo Tomás y UPIICSA; por medio de enlaces de fibra óptica monomodo con ancho de banda de 12 Gbps entre cada uno de ellos y conexiones a Internet e Internet2 con un ancho de banda de E3 (34 Mbps).

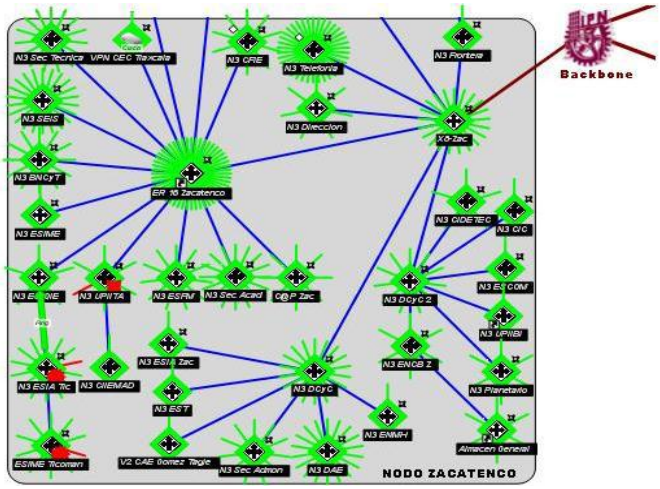

Figura 5.3 Nodo Zacatenco. Red Institucional IPN.[1]

El nodo de interés para este trabajo es el nodo Zacatenco (Figura 5.3) ya que es el que sirve de conexión a la Escuela Superior de Cómputo (ESCOM), dicho nodo es considerado el más importante del backbone, abarca la Unidad Profesional “Adolfo López Mateos” (UPALM) y la Unidad Profesional Ticomán (UPT), a través del nodo Zacatenco se logra la conexión a Internet e Internet 2, así como a los servidores de correo electrónico, los servidores DNS’s (Domain Name Service), los servidores de almacenaje (NAS), entre otros. Ahí se encuentran los enlaces satelitales, que se encargan de servirle conexión de red a los 16 centros foráneos que tiene el IPN; los enlaces de fibra óptica y cables de cobre a 40 diferentes centros cercanos a éste nodo; también se encuentran enlaces por microondas hacia UPIICSA, Santo Tomás y UPIITA.

La red de la Escuela Superior de Cómputo (ESCOM), depende directamente del centro de cómputo y comunicaciones del Instituto Politécnico Nacional. Cuenta con una conexión de ancho de banda T1. El enlace desde el

centro de cómputo y telecomunicaciones se hace mediante fibra óptica. El cuarto principal de telecomunicaciones que tiene ESCOM cuenta con el siguiente equipos un servidor unificado de mensajes Avaya modelo S8300 y un switch Enterasys N3. Además de ahí se distribuye hacia cuartos de telecomunicaciones instalados en los edificios en donde se tienen switches Enterasys serie A modelo 2H124 para distribuir verticalmente y horizontalmente la red hacia las aulas, laboratorios, biblioteca, oficinas, etc. por Ethernet. Dentro del Laboratorio se utilizó uno de esos nodos para conectarlo al servidor Principal el cual cuenta con dos tarjetas de red con soporte Ethernet una se conecta a la red institucional del IPN y la otra va hacia el punto de acceso inalámbrico con el cual se va a implementar una red inalámbrica que dará el acceso a los usuarios del entorno.

A continuación se hace una descripción general de las capacidades tecnológicas de los principales equipos que conforman el sistema de red alambrado de la red institucional del IPN que llega a ESCOM.

Switch Enterasys N3.(Tabla 5.1) Flexible para redes Voz, vídeo y datos (Redes Convergentes). Soporta servicios como Voz sobre IP (VoIP), alimentación eléctrica sobre Ethernet (Power-over-Ethernet), servicios broadcast y multicast de video. Proporciona seguridad avanzada mediante la prioridad de flujos y mecanismos de control de ancho de banda. Esto sin comprometer el rendimiento de la red. Tiene capacidad de manejar políticas avanzadas basada en el control de flujo y acceso.

Tabla 5. 1 Características de los Switches Enterasys N3

	N3
DFE Module Slots	3
Switching Throughput	40.5 Mpps
Total Backplane Capacity	240 Gbps
10/100 ports per system	216
100 Base-FX ports per system	162
10/100/1000 ports per system	216
10/100/1000 PoE ports per system	144
1000 Base-X ports per system	72
10 Gigabit ports per system	12

Switch Enterasys serie A. Los switches de la serie A con los que cuenta el centro, al ser de la misma compañía tecnológica son 100% compatibles con los equipos anteriormente descritos, estos switches proporcionan características similares y apoyo completo en el aseguramiento de calidad sobre la red.

Servidor de medios Avaya S8300. En complemento con los equipos anteriores, la red la compone también un servidor de medios Avaya S8300. Es el que da servicio a toda la comunicación de VoIP. Tiene capacidad para 450 host.

Una vez descrita la red alámbrica del IPN que servirá de apoyo para el sistema se describen los demás elementos de hardware y software utilizados, del mismo modo las configuraciones necesarias en las que se apoyó para realizar la implementación del caso de estudio.

5.4. Equipos de prueba

- **Pavilion dm4**

Especificaciones Técnicas [2]	
Sistema Operativo	Microsoft Windows 7 Home Premium
Procesador	Intel Core i5
Memoria RAM	3GB
Disco duro	640GB
Batería	Li-ion (estándar)
Conectividad	Bluetooth, Ethernet (RJ-45), HDMI, USB 2.0, Video VGA, Wi-Fi
Pantalla	14", LED-backlight
Unidad óptica	Grabador de CD, Grabador de DVD, Lector de CD, Lector de DVD

Resolución de pantalla	1366 x 768 (Wide XGA)
Lector de Tarjetas	Memory Stick PRO™ (MS PRO), Secure Digital™ (SD)
Periféricos	Altavoces integrados, Micrófono integrado, Webcam integrada
Tarjeta de video	Intel Graphics HD
Tarjeta de audio	Intel High Definition Audio

- **Toshiba T115D**

Características [3]:

- Pantalla LED TruBrite de 11.6″ pulgadas (1366×768)
- Procesador AMD Turion 1.5GHz Doble núcleo Neo XL325
- Disco duro de 320GB Serial ATA 5400 rpm con sensor de impactos
- Memoria RAM de 2GB DDR2 expandible hasta 4GB
- Tarjeta gráfica integrada ATI Radeon 3200 con memoria de 1919MB
- Cámara Web integrada de 1.3 mega pixeles con reconocimiento de cara
- Altavoz integrado
- Conectividad Wi-Fi (802.11b/g), Fast Ethernet (10/100)
- Lector de tarjetas 5-en-1
- Tres puertos USB con un puerto Cargue mientras duerma en lado izquierdo
- Conexión HDMI y una salida estándar para video análogo VGA
- PC Health Monitor que mantiene el buen funcionamiento del uso de energía, temperatura y el disco duro
- Batería de 6 celdas 5600 mAh con hasta 6 horas de funcionamiento
- Certificación Energy Star
- Peso: 1.6 Kg

- Dimensiones: 28.7 x 21 x 3.4 cm (Altura, Largo, Ancho)
- Un año de garantía: Con garantía internacional limitada.
- Viene con el siguiente software: Antivirus Norton Internet Security, Microsoft Works, Microsoft Office Home y versión Estudiante, Google Toolbar, Reproductor Windows Media Player, Consola de juegos WildTangent.

5.4. Resultados de la implementación del caso de estudio del sistema en el entorno educativo

La mayoría de las redes inalámbricas permiten a los usuarios conectarse simplemente proporcionando una contraseña correcta. Algunos son totalmente garantizados y no requieren verificación. Aunque es poco frecuente, también pueden asegurarse redes inalámbricas con la presencia de un certificado de firma instalada en el equipo. Este certificado debe estar correctamente instalado en su máquina antes de que se les conceda acceso a la red.

5.5. Instalación de certificado del servidor

❄ Hemos de instalar el certificado del servidor en primer lugar. Para comenzar, ejecutaremos el certificado si lo tenemos en un archivo (Figura 5.5):

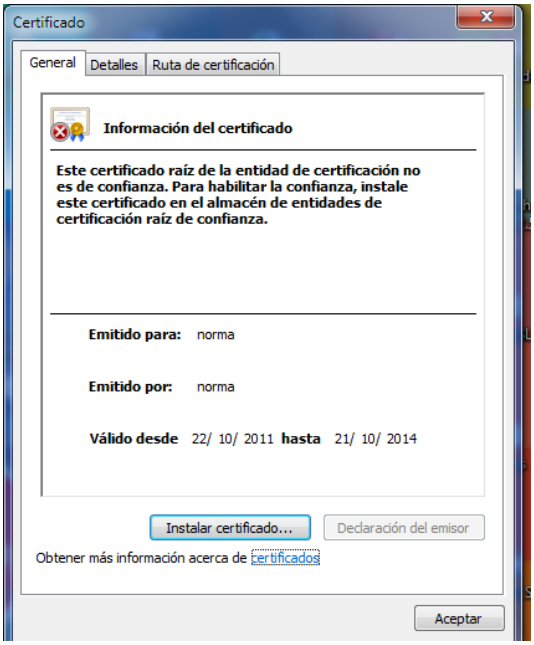

Figura 5.5 Certificado del servidor

- Hacemos click en Instalar certificado..., se nos abrirá la ventana del asistente de importación de certificados (Figura 5.6), pulsamos sobre siguiente.

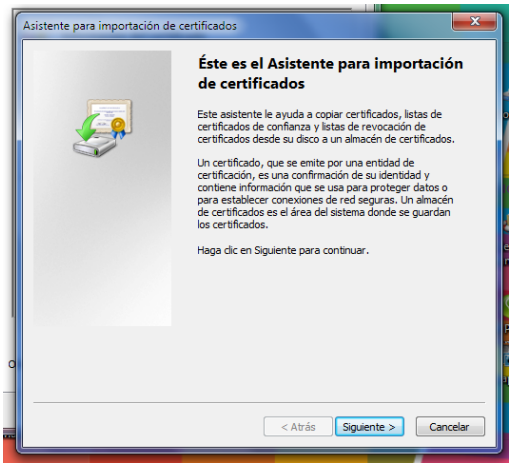

Figura 5.6 Asistente de importación de certificados

- Seleccionamos Colocar todos los certificados en el siguiente almacén y hacemos click en Examinar... para seleccionar: Entidades de certificación raíz de confianza (Figura 5.7)

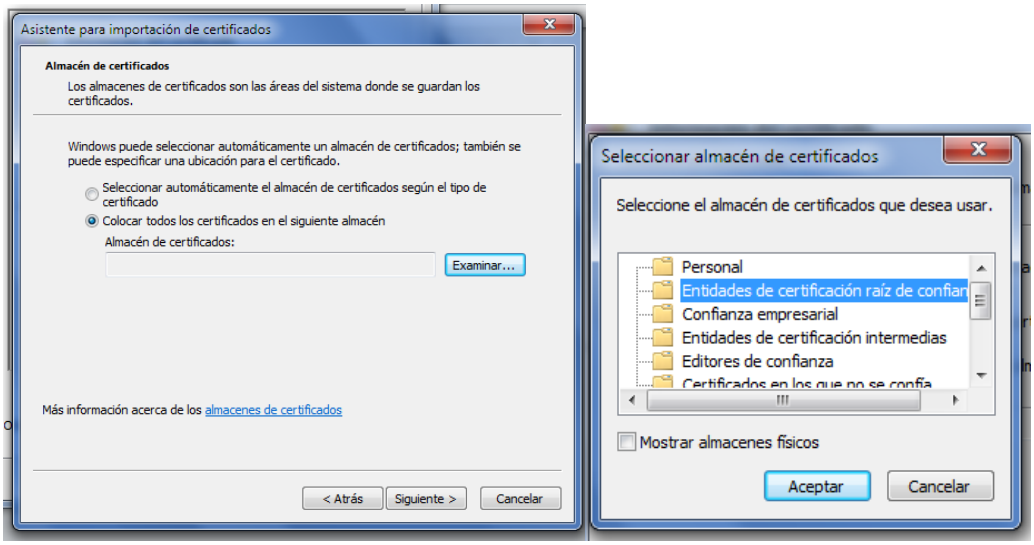

Figura 5.7 Almacén de certificados

❄ Damos clic en aceptar y nos muestra la finalización del asistente para importación de certificados (Figura 5.8).

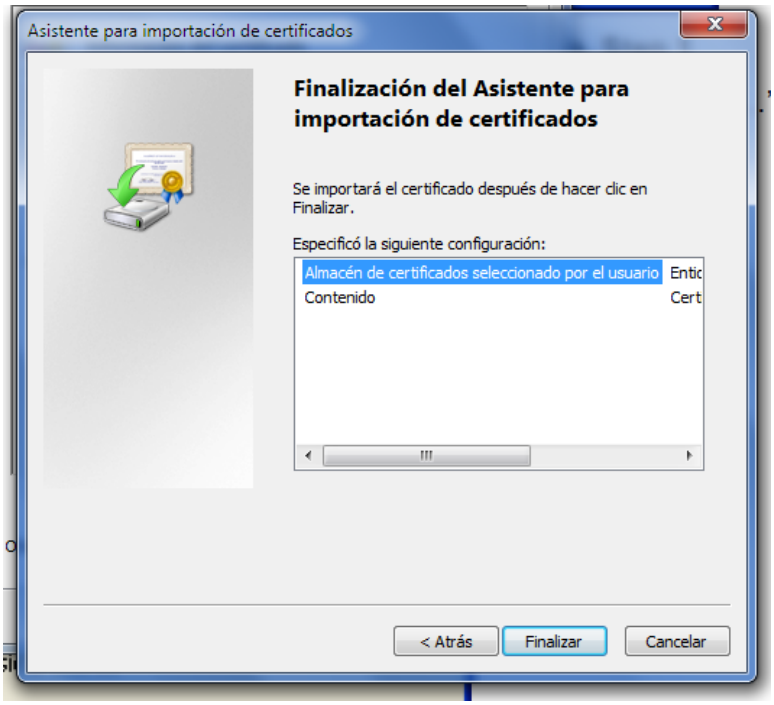

Figura 5.8 Finalización del asistente para importación de certificados

- Seleccionamos finalizar. Es posible que nos muestre una advertencia sobre la instalación del certificado. En caso que nos la muestre, le decimos que sí (Figura 5.9).

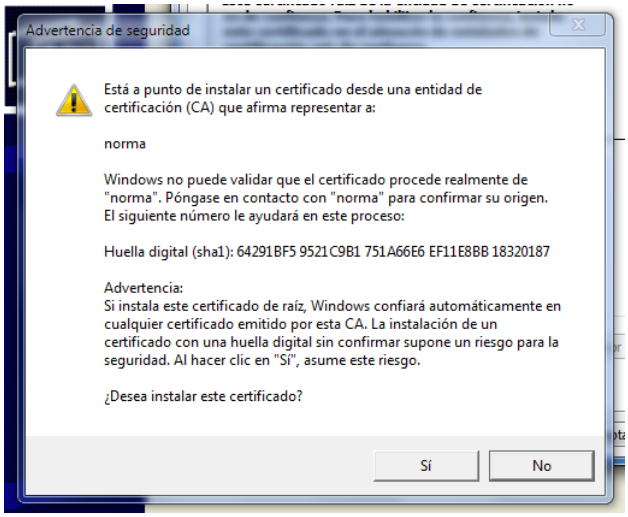

Figura 5.9 Advertencia sobre la instalación del certificado

- ✓ Por último nos mostrará que la importación se completó (Figura 5.10).

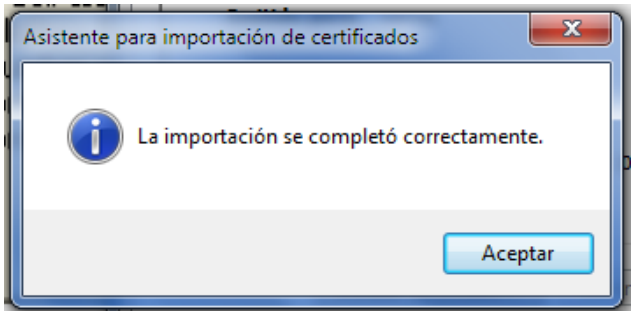

Figura 5.10 Importación completada

5.6. Instalación de certificado del cliente

Una vez instalado el certificado raíz del Laboratorio de Cálculo procederemos a instalar el que nos identifica como cliente de forma análoga.

- Se abrirá el asistente (Figura 5.11).

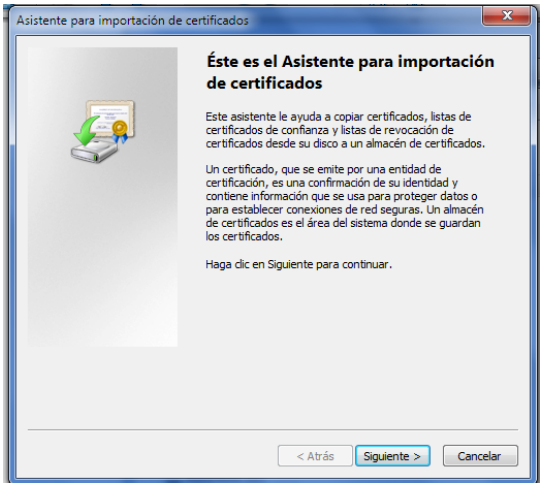

Figura 5.11 Asistente de importación de certificados

- Al instalar ese certificado personal nos pedirá un password, deberemos dejarlo en blanco (Figura 5.12).

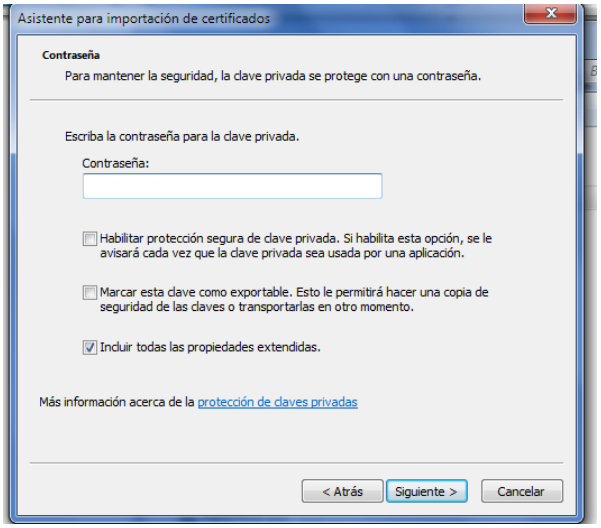

Figura 5.12 Password

- En la ventana donde seleccionamos el almacén de los certificados, seleccionaremos el almacén "Personal" (Figura 5.13).

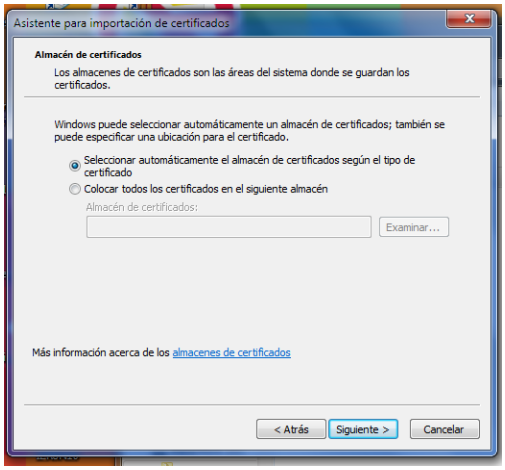

Figura 5.13 Almacén

✓ Seleccionamos siguiente y nos muestra la finalización de la importación del certificado (Figura 5.14).

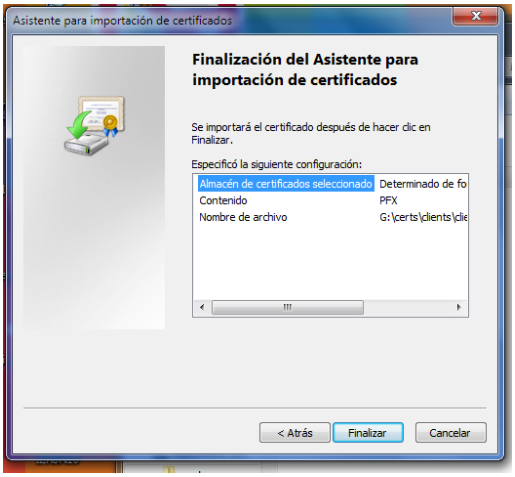

Figura 5.14 Finalización del asistente

✓ Por último nos mostrará que la importación se completó (Figura 5.15).

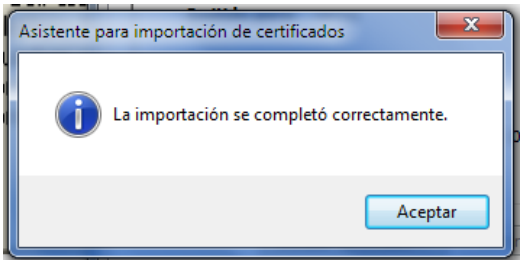

Figura 5.15 Importación completada

5.7. Configuración a la conexión Wireless.

Debemos asegurarnos que la red inalámbrica, tenga habilitada la configuración de red automática (*dhcp*).

Seguimos los siguientes pasos:

- ✧ Para ello, tenemos que ir a "Inicio -> Panel de Control -> Redes e Internet -> Centro de redes y recursos compartidos -> Cambiar configuración del adaptador" (Figura 5.16).

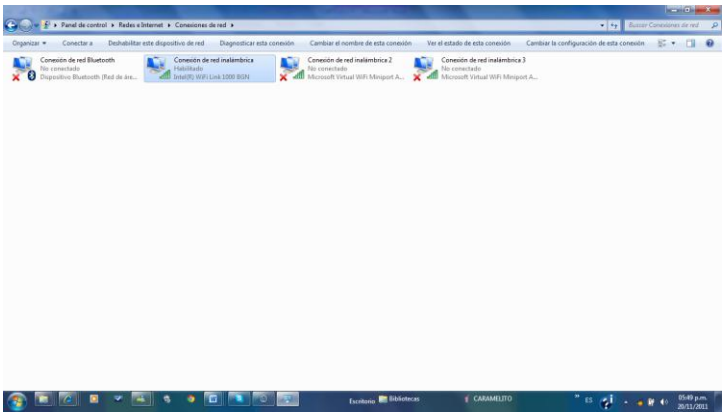

Figura 5.16 Conexión de redes

- ✧ Una vez allá, hacemos click con el botón derecho en "Conexión de red inalámbrica" (ver imagen inferior) y seleccionamos "Propiedades". Pulsamos "Continuar" en el caso de que nos pida permiso. Seleccionamos "Protocolo de Internet versión 4 (TCP/IPv4)" y damos clic en "Propiedades" donde habilitamos las opciones (Figura 5.17):

  - ✓ Obtener una dirección IP automáticamente
  - ✓ Obtener la dirección del servidor DNS automáticamente Pulsamos "Aceptar".

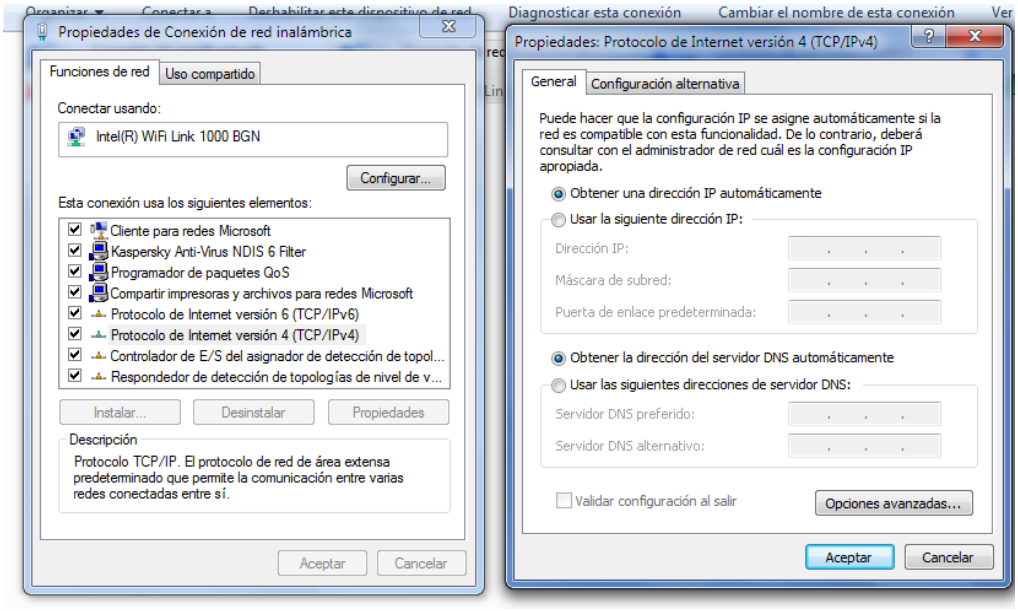

Figura 5.17 Protocolo de Internet

✧ Seguidamente, hacemos click en: "Inicio -> Panel de Control -> Redes e Internet -> Centro de redes y recursos compartidos -> Administrar redes inalámbricas". Pulsamos "Agregar" y nos encontramos con esta ventana de la Figura 5.18:

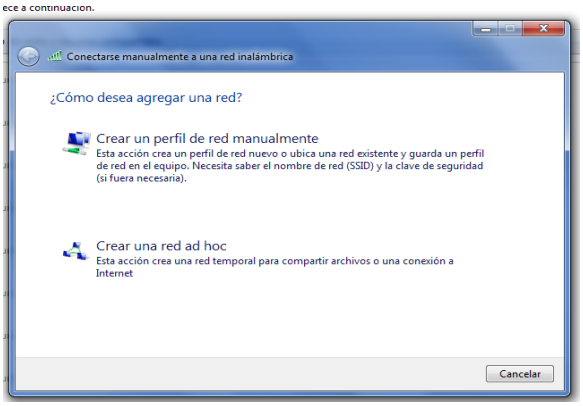

Figura 5.18 Agregar red

✧ Seleccionamos Crear un perfil de red manualmente y seguidamente veremos:

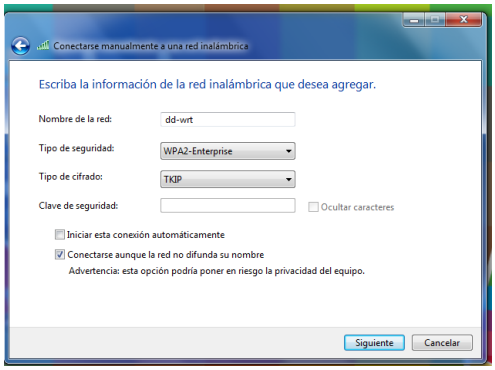

Figura 5.19 Información de la red inalámbrica

Tenemos que rellenarlo tal y como se ve en la Figura 5.15. El Nombre de la Red es dd-wrt, el tipo de seguridad es WPA2-Enterprise y marcamos la casilla Conectarse aunque la red no difunda su nombre (Figura 5.20).

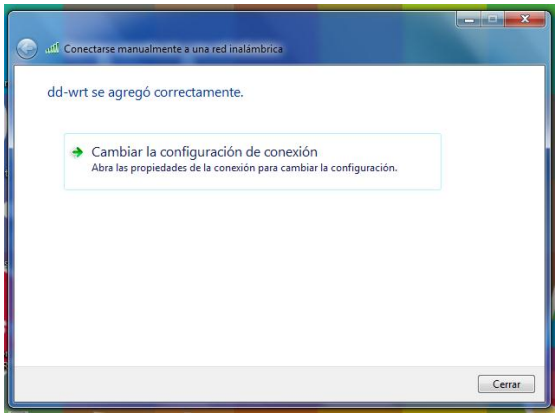

Figura 5.20 Red agregada correctamente

- ✧ Es muy importante que seleccionemos Cambiar la configuración de la conexión para que verifique los certificados (Figura 5.21).

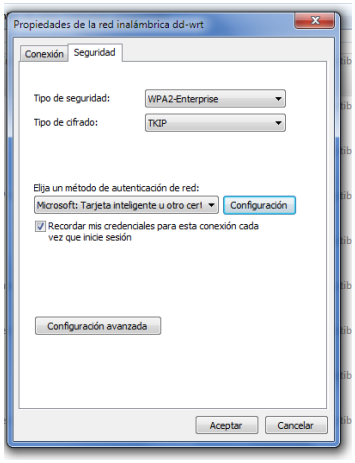

Figura 5.21 Configuración de conexión

- En la pestaña de "Seguridad" seleccionamos como método la Microsoft: Tarjeta inteligente u otro certificado y hacemos click en "Configuración" (Figura 5.22).

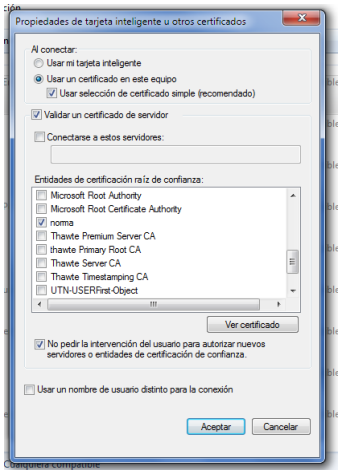

Figura 5.22 Propiedades de tarjeta inteligente

- Activamos Norma que es el nombre del certificado raíz y No pedir la intervención del usuario.... Pulsamos "Aceptar" y volveremos a la ventana que nos dice que dd-wrt ha sido agregada. Seguidamente cerramos esa ventana.

- Ahora ya podemos conectarnos a dd-wrt. Seleccionándola tal y como se muestra en la Figura 5.23:

Figura 5.23 Conexión a la red dd-wrt

- Seleccionamos la red dd-wrt y nos aparecerá la imagen de la Figura 5.24 la cual nos pedirá nuestro nombre y contraseña para tener finalmente la conexión.

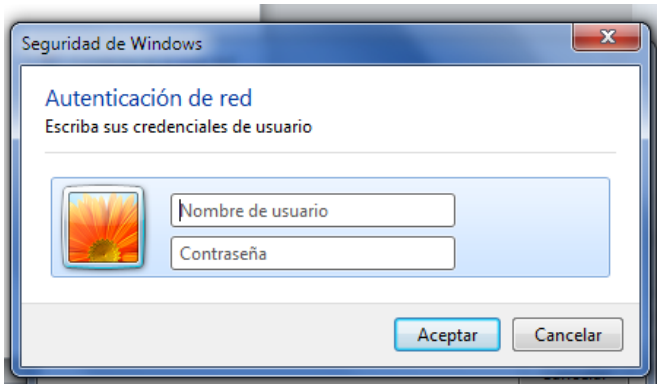

Figura 5.24 Petición de datos de usuario

- En este momento ya estamos conectados a la red.

5.8. Prueba del sistema

Después de haber instalado los certificados y ya estando conectados a la red procedemos a ingresar a la página de bienvenida como prueba de que el sistema está funcionando correctamente

- Abrimos el navegador y escribimos https://192.168.2.120, usamos "https" en lugar de "http" para indicar que queremos conectarnos a través de SSL.
- Nos aparece una página de error que nos advierte que un certificado creado por nosotros es sospechoso. Hacemos clic en el link "Vaya a este sitio web" para saltarnos este aviso de seguridad e ir al sitio (Figura 5.25).

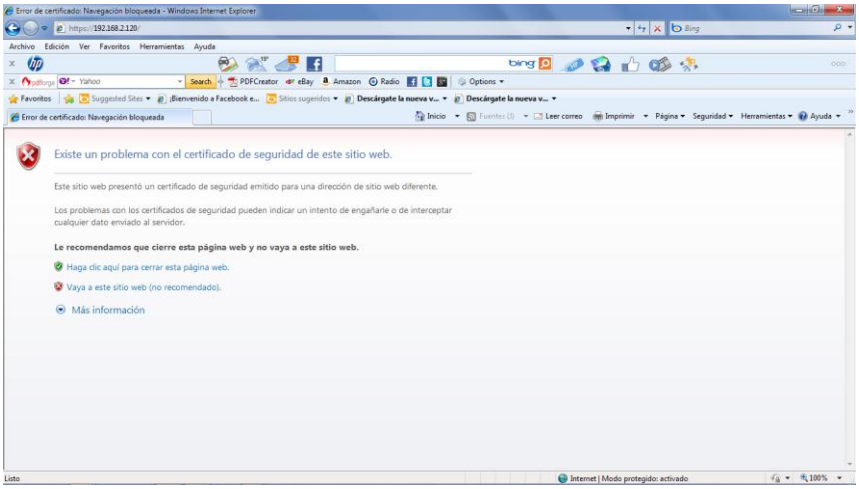

Figura 5.25 Página de error

- Encontramos nuestra página de bienvenida corriendo sobre SSL(Figura 5.26):

Figura 5.26 Página de bienvenida

Esta prueba consistió en que usuarios (alumnos, profesores y visitantes) se conectarón al dominio de red, recibieran la bienvenida al dominio, se autenticaran e hicieran uso de los servicios de Internet. La prueba se desarrolló durante un periodo de dos semanas y se realizó con los usuarios que tienen acceso al laboratorio que es aproximadamente de 80 usuarios entre alumnos de licenciatura o maestría, profesores y algunos visitantes, pero la prueba se realizó con 70 de ellos, cabe mencionar que el acceso de estos no fue simultáneo sino conforme se

hacía uso del laboratorio. Inicialmente cuando llegarón se les dio una introducción del trabajo que se realizó y algunas indicaciones previas al uso del sistema.

Esta prueba se realizó sobre una variedad de dispositivos móviles y de diferentes marcas como: Laptops (de las marcas Dell, Acer, Apple, Compaq y Hp), No solo se probó sobre distintos dispositivos, sino también sobre varios sistemas operativos y diferentes navegadores de red.

Los resultados obtenidos de esta prueba de funcionalidad fuerón los presentados en la gráfica de la Figura 5.22 la interpretación de la misma indica que de una muestra de 70 usuarios el 95.7% se conectó a la red de manera satisfactoria, mientras que el 4.3% tuvo problemas de conexión estos debido a un inconveniente que se tuvo con el punto de acceso inalámbrico donde se bloqueó por algunas aplicaciones extras que se realizaron en ese momento, por lo cual se tuvo que reiniciar el punto de acceso y reconfigurar algunos parámetros, por lo que lamentablemente los usuarios ya no pudieron probar el sistema durante ese tiempo que el dominio de red estuvo caído.

La disponibilidad de un acceso de red debe garantizar una disponibilidad arriba del 99.9% dependiendo del sistema, pero el cálculo que se hace no se está basado en el tiempo total que duro la prueba y el tiempo que estuvo disponible el dominio de red. Haciéndolo de esta manera se tiene que la prueba se realizó por espacio de 15 días lo que equivale a 360 horas, el problema que se suscitó provoco detener el servicio de red por 15 minutos aproximadamente en lo que se reinició el equipo y se configuró nuevamente por lo que el tiempo en que la red estuvo disponible fue un total de las 359.75 horas. Apoyados en la siguiente formula:

Disponibilidad (%) = 100*(Tiempo de uso efectivo / Tiempo de uso total)

Disponibilidad = 100* (359.75/360)

Disponibilidad del Dominio de Red = 99.93%

Con lo que se puede garantizar una excelente disponibilidad del dominio de red.

El siguiente punto que se valida es la autenticación, donde el máximo de los usuarios no tuvierón problema alguno, es decir se pudieron conectar y accesar

a la página de bienvenida o al servicio de Internet como lo muestran las imágenes (Figura 5.3), cabe mencionar que hubo ciertos errores de dedo por llamarle así, cuando introducían su usuario y/o contraseña ya que el sistema es sensitivo en ambos datos a las mayúsculas/minúsculas, pero una vez que los ingresaban correctamente el sistema los autenticaba de manera correcta.

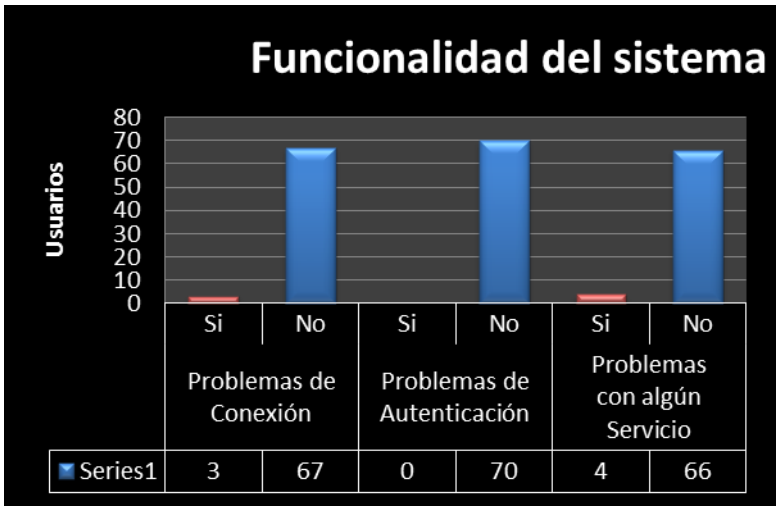

Figura 5. 27 Gráfica de funcionalidad del sistema

Por último se verificó que los usuarios pudieran hacer uso de los servicios que se les ofrecían el 94.2% logro establecer contacto con los servicios manera satisfactoria mientras que el restante 5.8% tuvo problemas con el servicio de Webescom (Contiene videos sobre conferencias y ponencias de congresos) que es un servicio multimedia que se ofrece fuera de la red del IPN y debido a fallos aún desconocidos el servicio estuvo detenido.

Los resultados obtenidos muestran que el dominio de red está disponible, que el método de autenticación es confiable y la respuesta a los servicios es satisfactoria.

A continuación se muestran algunas de las pantallas del sistema en algunos de los diferentes dispositivos donde se realizaron las pruebas. (Figura 5.28)

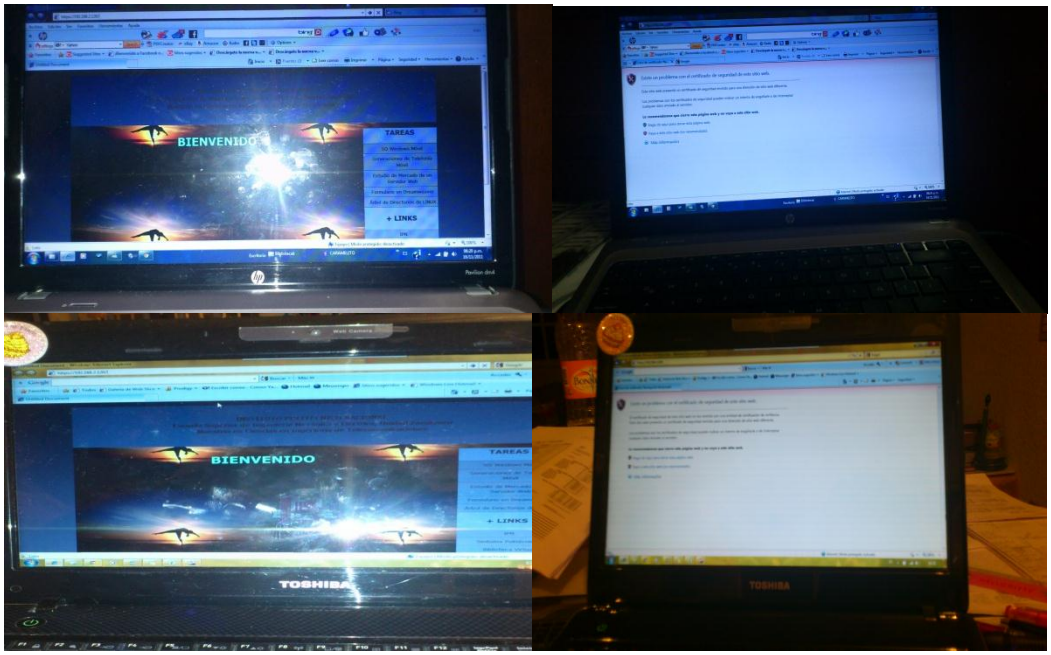

Figura 5. 28 Pantallas del sistema en Algunos Dispositivos Móviles

5.9. Comparativa de métodos de autenticación

En la Tabla 5.2 se presenta una comparación de las características que ofrecen dos métodos de autenticación los cuales se mencionaron en el Capítulo 3.

Analizando cada una de las características listadas, se observa que la solución planteada en nuestro sistema, proporciona gran parte de las funcionalidades de dos métodos diferentes que son soluciones comerciales actuales [5]. Nuestro sistema es una alternativa bastante interesante de control de acceso a redes inalámbricas.

Tabla 5. 2 Comparativa de métodos de autenticación

	CISCO	Microsoft	Nuestro sistema
Certificado-Cliente	N/A	SI	SI
Certificado-Servidor	N/A	SI	SI
Seguridad Credenciales	Débil	Fuerte	Fuerte
Bases de datos soportadas	Active Directory Dominios NT	Active Directory	SQL LDAP
Intercambio de claves dinámicas	SI	SI	SI
Autenticación mutua	SI	SI	SI
Costo	SI	SI	NO

Como conclusión a esta Tabla se puede mencionar que nuestro sistema presenta mejores características que los productos comerciales en seguridad en credenciales, bases de datos con licencia libre y bajo costo.

5.10. Referencias

[1] Tesis de Maestría MODELO DE LA INFRAESTRUCTURA PARA QoS APLICADO EN AMBIENTES EDUCATIVOS. Néstor Guillermo Martínez Alvarado. ESIME-ZAC. 2009
[2] http://www.idirecto.com/le743laabm-laptop-pavilion-1280la-core-640gb-p-53008.html
[3] http://www.generacionportatil.com/caracteristicas-del-ultra-portable-toshiba-satellite-t115d/
[4] DD-WRT http://www.dd-wrt.com/site/index
[5]http://www.virusprot.com/Whitepap3.html

CAPÍTULO 6
CONCLUSIONES Y TRABAJO A FUTURO

6.1. Conclusiones

La identificación y autenticación de los seres humanos como un individuo único dentro de una sociedad a lo largo de los años desde los inicios de nuestra existencia ha sido fundamental para el desarrollo de la vida cotidiana, el tener una identidad que te diferencie de los demás y gracias a ella poder participar o tener voz como persona única es algo que siempre deberá de ser tomado en cuenta, ya que en las sociedades modernas es necesario estar identificado para realizar algún trámite o participar en algún evento.

Todo esto en el mundo físico, pero en el mundo virtual aún hay muy poco avance en la manera en que las personas se pueden identificar en un entorno de red, y en una red tan inmensa como el internet, la cual en la actualidad domina gran parte de nuestras vidas y actividades cotidianas, era necesario que tarde o temprano nos hiciéramos la pregunta de cómo nos podríamos identificar en el internet, esto es lo que nos llevó a crear nuestro MODELO E INFRAESTRUCTURA DE SEGURIDAD BASADO EN IDENTIFICACIÓN Y AUTENTIFICACIÓN PARA REDES.

Podemos darnos cuenta de medios alternos existentes para implementar la seguridad en redes a través de una autenticación controlada a los usuarios.

Linux, junto con FreeRADIUS, son herramientas que permiten esta implementación de una manera efectiva y segura para clientes que utilizan cualquier sistema operativo.

Contando con un manual eficiente, es posible instalar y configurar RADIUS de una manera relativamente sencilla.

El manejo de cuentas también es muy sencillo, sobre todo, si se utiliza algún servidor externo como es el caso de MySQL. Puesto que permite realizar modificaciones y actualizaciones a la base de datos de usuarios, es decir, realizar la administración de ésta.

El aplicar políticas de seguridad no es tarea sencilla; sin embargo, actualmente, se cuenta con herramientas que ayudan a la realización de tan importante tarea.

Este sistema fue hecho pensando en esa necesidad, aunque tal vez a un nivel muy básico, pero que pueda ser utilizado como base para trabajos futuros e investigaciones futuras que puedan hacer de esto un estándar de identificación internacional, es una arquitectura muy maleable que puede tomar distintos rumbos.

Finalmente podemos decir que es un pequeño paso en una gran carrera de desarrollo e implementaciones futuras que solo el tiempo dirá que forma tomara y hacia donde se dirigirá, todo esto con el único fin de llevar la identificación de personas a un nuevo nivel y a un nuevo mundo, como es el caso de la internet la cual es como un mundo virtual en la cual salimos a pasear todos los días, a hacer compras, a buscar un libro, a escuchar buena música, o simplemente a aprender nuevas cosas día a día.

6.2. Trabajos futuros

Al ser necesarios mundialmente hoy en día, nuevos e innovadores métodos que tengan como objetivo garantizar al 100% la identidad y autenticidad de las personas, esta tesis, pretende ser una base de conocimientos para nuevos trabajos y personas o grupos que se interesen en darle continuidad al tema.

Pues sabiendo que las tecnologías con las que se logra dicho objetivo avanzan continuamente, sería una buena opción añadir nuevas tecnologías y en conjunto lograr un gran impacto dentro del hábito de seguridad de información e identificación y autentificación de personas.

Actualmente nos han hecho propuestas para mejorar el sistema pero por motivos de tiempo no se han podido implementar es por ello que con el objeto de ampliar la funcionalidad del modelo propuesto se plantea considerar los siguientes desarrollos en trabajos futuros:

- Implementar el sistema en un entorno educativo más amplio.

- Hacer pruebas con otro tipo de equipos como Smartphone y tablets.
- Hacer mediciones y pruebas más robustas que ayuden a comprobar la seguridad del sistema.

6.3. Trabajos derivados de la tesis

Finalmente se presentan algunas de las publicaciones que se han realizado como resultado de la investigación de la cual forma parte esta tesis, y que han sido presentados en congresos internacionales y nacionales (anexo 8):

- Autores: Ing. Norma Alicia Cruz Guzmán, Dr. Felipe Rolando Menchaca García

  Seguridad en Sistemas Móviles Mediante Autentificación de Firma Digital, IV Congreso de Ingeniería en Comunicaciones y electrónica 2010 (CICE), México D.F. Noviembre 2010.

- Autores: Ing. Norma Alicia Cruz Guzmán, M. en C. Chadwick Carreto Arellano, Dr. Salvador Álvarez Ballesteros

  Modelo e Infraestructura de Seguridad basado en Identificación y Autenticación para Redes, 4to Congreso Internacional de Ingenierías Mecánica, Eléctrica, Electrónica, Mecatrónica y Computacional (CIMEEM), Querétaro. Septiembre 2011.

- Autores: Ing. Norma Alicia Cruz Guzmán, M. en C. Chadwick Carreto Arellano, Dr. Salvador Álvarez Ballesteros

  Poster: Modelo e Infraestructura de Seguridad basado en identificación y Autenticación para Redes, 4to Congreso Internacional de Ingenierías Mecánica, Eléctrica, Electrónica, Mecatrónica y Computacional (CIMEEM), Querétaro. Septiembre 2011.

- Autores: Ing. Norma Alicia Cruz Guzmán, M. en C. Chadwick Carreto Arellano, Dr. Salvador Álvarez Ballesteros

  Conferencia: Prototipo de una Arquitectura de Seguridad Basada en Identificación y Autentificado de Red, 18ª Semana Nacional de Ciencia y Tecnología, CECYT 1 Gonzalo Vázquez Vela, México D.F. Octubre 2011.

- Autores: Ing. Norma Alicia Cruz Guzmán, M. en C. Chadwick Carreto Arellano, Dr. Salvador Álvarez Ballesteros

  Diseño de un Modelo e Infraestructura de Seguridad basado en Identificación y Autenticación para Redes, Congreso Internacional sobre Investigación y Desarrollo (CIIDE), Mérida, Yucatán. Noviembre 2011.

- Autores: Ing. Norma Alicia Cruz Guzmán, M. en C. Chadwick Carreto Arellano, Dr. Salvador Álvarez Ballesteros

  Desarrollo de una Arquitectura de Seguridad para Identificación y Autenticación para Redes, Vigésima Reunión de otoño de Comunicaciones, Computación, Electrónica, Automatización, Robótica y Exposición Industrial IEEE Sección México (ROC&C), Acapulco, Guerrero. Diciembre 2011.

ANEXO 1. Radius.conf

```
# -*- text -*-
##
## radiusd.conf  -- FreeRADIUS server configuration file.
##
##        http://www.freeradius.org/
##        $Id$
##

#######################################################################
#
#        Read "man radiusd" before editing this file.  See the section
#        titled DEBUGGING.  It outlines a method where you can quickly
#        obtain the configuration you want, without running into
#        trouble.
#
#        Run the server in debugging mode, and READ the output.
#
#                $ radiusd -X
#
#        We cannot emphasize this point strongly enough.  The vast
#        majority of problems can be solved by carefully reading the
#        debugging output, which includes warnings about common issues,
#        and suggestions for how they may be fixed.
#
#        There may be a lot of output, but look carefully for words like:
#        "warning", "error", "reject", or "failure".  The messages there
#        will usually be enough to guide you to a solution.
#
#        If you are going to ask a question on the mailing list, then
#        explain what you are trying to do, and include the output from
#        debugging mode (radiusd -X).  Failure to do so means that all
#        of the responses to your question will be people telling you
#        to "post the output of radiusd -X".

#######################################################################
#
#        The location of other config files and logfiles are declared
#        in this file.
#
#        Also general configuration for modules can be done in this
#        file, it is exported through the API to modules that ask for
#        it.
#
#        See "man radiusd.conf" for documentation on the format of this
#        file.  Note that the individual configuration items are NOT
#        documented in that "man" page.  They are only documented here,
#        in the comments.
#
```

```
#           As of 2.0.0, FreeRADIUS supports a simple processing language
#           in the "authorize", "authenticate", "accounting", etc. sections.
#           See "man unlang" for details.
#

prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
logdir = /var/log/freeradius
raddbdir = /etc/freeradius
radacctdir = ${logdir}/radacct

#
#  name of the running server.  See also the "-n" command-line option.
name = freeradius

#  Location of config and logfiles.
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/${name}

# Should likely be ${localstatedir}/lib/radiusd
db_dir = ${raddbdir}

#
# libdir: Where to find the rlm_* modules.
#
#   This should be automatically set at configuration time.
#
#   If the server builds and installs, but fails at execution time
#   with an 'undefined symbol' error, then you can use the libdir
#   directive to work around the problem.
#
#   The cause is usually that a library has been installed on your
#   system in a place where the dynamic linker CANNOT find it.  When
#   executing as root (or another user), your personal environment MAY
#   be set up to allow the dynamic linker to find the library.  When
#   executing as a daemon, FreeRADIUS MAY NOT have the same
#   personalized configuration.
#
#   To work around the problem, find out which library contains that symbol,
#   and add the directory containing that library to the end of 'libdir',
#   with a colon separating the directory names.  NO spaces are allowed.
#
#   e.g. libdir = /usr/local/lib:/opt/package/lib
#
#   You can also try setting the LD_LIBRARY_PATH environment variable
#   in a script which starts the server.
#
#   If that does not work, then you can re-configure and re-build the
#   server to NOT use shared libraries, via:
#
```

```
#           ./configure --disable-shared
#           make
#           make install
#
libdir = /usr/lib/freeradius

#  pidfile: Where to place the PID of the RADIUS server.
#
#  The server may be signalled while it's running by using this
#  file.
#
#  This file is written when ONLY running in daemon mode.
#
#  e.g.:  kill -HUP `cat /var/run/radiusd/radiusd.pid`
#
pidfile = ${run_dir}/${name}.pid

#  chroot: directory where the server does "chroot".
#
#  The chroot is done very early in the process of starting the server.
#  After the chroot has been performed it switches to the "user" listed
#  below (which MUST be specified).  If "group" is specified, it switchs
#  to that group, too.  Any other groups listed for the specified "user"
#  in "/etc/group" are also added as part of this process.
#
#  The current working directory (chdir / cd) is left *outside* of the
#  chroot until all of the modules have been initialized.  This allows
#  the "raddb" directory to be left outside of the chroot.  Once the
#  modules have been initialized, it does a "chdir" to ${logdir}.  This
#  means that it should be impossible to break out of the chroot.
#
#  If you are worried about security issues related to this use of chdir,
#  then simply ensure that the "raddb" directory is inside of the chroot,
#  end be sure to do "cd raddb" BEFORE starting the server.
#
#  If the server is statically linked, then the only files that have
#  to exist in the chroot are ${run_dir} and ${logdir}.  If you do the
#  "cd raddb" as discussed above, then the "raddb" directory has to be
#  inside of the chroot directory, too.
#
#chroot = /path/to/chroot/directory

# user/group: The name (or #number) of the user/group to run radiusd as.
#
#   If these are commented out, the server will run as the user/group
#   that started it.  In order to change to a different user/group, you
#   MUST be root ( or have root privleges ) to start the server.
#
#   We STRONGLY recommend that you run the server with as few permissions
#   as possible.  That is, if you're not using shadow passwords, the
#   user and group items below should be set to radius'.
#
#  NOTE that some kernels refuse to setgid(group) when the value of
```

```
#  (unsigned)group is above 60000; don't use group nobody on these systems!
#
#  On systems with shadow passwords, you might have to set 'group = shadow'
#  for the server to be able to read the shadow password file.  If you can
#  authenticate users while in debug mode, but not in daemon mode, it may be
#  that the debugging mode server is running as a user that can read the
#  shadow info, and the user listed below can not.
#
#  The server will also try to use "initgroups" to read /etc/groups.
#  It will join all groups where "user" is a member.  This can allow
#  for some finer-grained access controls.
#
user = freerad
group = freerad

#  max_request_time: The maximum time (in seconds) to handle a request.
#
#  Requests which take more time than this to process may be killed, and
#  a REJECT message is returned.
#
#  WARNING: If you notice that requests take a long time to be handled,
#  then this MAY INDICATE a bug in the server, in one of the modules
#  used to handle a request, OR in your local configuration.
#
#  This problem is most often seen when using an SQL database.  If it takes
#  more than a second or two to receive an answer from the SQL database,
#  then it probably means that you haven't indexed the database.  See your
#  SQL server documentation for more information.
#
#  Useful range of values: 5 to 120
#
max_request_time = 30

#  cleanup_delay: The time to wait (in seconds) before cleaning up
#  a reply which was sent to the NAS.
#
#  The RADIUS request is normally cached internally for a short period
#  of time, after the reply is sent to the NAS.  The reply packet may be
#  lost in the network, and the NAS will not see it.  The NAS will then
#  re-send the request, and the server will respond quickly with the
#  cached reply.
#
#  If this value is set too low, then duplicate requests from the NAS
#  MAY NOT be detected, and will instead be handled as seperate requests.
#
#  If this value is set too high, then the server will cache too many
#  requests, and some new requests may get blocked.  (See 'max_requests'.)
#
#  Useful range of values: 2 to 10
#
cleanup_delay = 5

#  max_requests: The maximum number of requests which the server keeps
```

```
#  track of.  This should be 256 multiplied by the number of clients.
#  e.g. With 4 clients, this number should be 1024.
#
#  If this number is too low, then when the server becomes busy,
#  it will not respond to any new requests, until the 'cleanup_delay'
#  time has passed, and it has removed the old requests.
#
#  If this number is set too high, then the server will use a bit more
#  memory for no real benefit.
#
#  If you aren't sure what it should be set to, it's better to set it
#  too high than too low.  Setting it to 1000 per client is probably
#  the highest it should be.
#
#  Useful range of values: 256 to infinity
#
max_requests = 1024

#  listen: Make the server listen on a particular IP address, and send
#  replies out from that address. This directive is most useful for
#  hosts with multiple IP addresses on one interface.
#
#  If you want the server to listen on additional addresses, or on
#  additionnal ports, you can use multiple "listen" sections.
#
#  Each section make the server listen for only one type of packet,
#  therefore authentication and accounting have to be configured in
#  different sections.
#
#  The server ignore all "listen" section if you are using '-i' and '-p'
#  on the command line.
#
listen {
        #  Type of packets to listen for.
        #  Allowed values are:
        #       auth      listen for authentication packets
        #       acct      listen for accounting packets
        #       proxy   IP to use for sending proxied packets
        #       detail  Read from the detail file.  For examples, see
        #         raddb/sites-available/copy-acct-to-home-server
        #       status  listen for Status-Server packets.  For examples,
        #                 see raddb/sites-available/status
        #       coa     listen for CoA-Request and Disconnect-Request
        #                 packets.  For examples, see the file
        #                 raddb/sites-available/coa-server
        #
        type = auth

        #  Note: "type = proxy" lets you control the source IP used for
        #       proxying packets, with some limitations:
        #
        #    * A proxy listener CANNOT be used in a virtual server section.
        #    * You should probably set "port = 0".
```

```
        #    * Any "clients" configuration will be ignored.
        #
        #  See also proxy.conf, and the "src_ipaddr" configuration entry
        #  in the sample "home_server" section.  When you specify the
        #  source IP address for packets sent to a home server, the
        #  proxy listeners are automatically created.

        #  IP address on which to listen.
        #  Allowed values are:
        #          dotted quad (1.2.3.4)
        #      hostname    (radius.example.com)
        #      wildcard    (*)
        ipaddr = *

        #  OR, you can use an IPv6 address, but not both
        #  at the same time.
#       ipv6addr = ::      # any.  ::1 == localhost

        #  Port on which to listen.
        #  Allowed values are:
        #          integer port number (1812)
        #          0 means "use /etc/services for the proper port"
        port = 1812

        #  Some systems support binding to an interface, in addition
        #  to the IP address.  This feature isn't strictly necessary,
        #  but for sites with many IP addresses on one interface,
        #  it's useful to say "listen on all addresses for eth0".
        #
        #  If your system does not support this feature, you will
        #  get an error if you try to use it.
        #
#       interface = eth0

        #  Per-socket lists of clients.  This is a very useful feature.
        #
        #  The name here is a reference to a section elsewhere in
        #  radiusd.conf, or clients.conf.  Having the name as
        #  a reference allows multiple sockets to use the same
        #  set of clients.
        #
        #  If this configuration is used, then the global list of clients
        #  is IGNORED for this "listen" section.  Take care configuring
        #  this feature, to ensure you don't accidentally disable a
        #  client you need.
        #
        #  See clients.conf for the configuration of "per_socket_clients".
        #
#       clients = per_socket_clients
}

#  This second "listen" section is for listening on the accounting
#  port, too.
```

```
#
listen {
        ipaddr = *
#       ipv6addr = ::
        port = 1813
        type = acct
#       interface = eth0
#       clients = per_socket_clients
}

#  hostname_lookups: Log the names of clients or just their IP addresses
#  e.g., www.freeradius.org (on) or 206.47.27.232 (off).
#
#  The default is 'off' because it would be overall better for the net
#  if people had to knowingly turn this feature on, since enabling it
#  means that each client request will result in AT LEAST one lookup
#  request to the nameserver.   Enabling hostname_lookups will also
#  mean that your server may stop randomly for 30 seconds from time
#  to time, if the DNS requests take too long.
#
#  Turning hostname lookups off also means that the server won't block
#  for 30 seconds, if it sees an IP address which has no name associated
#  with it.
#
#  allowed values: {no, yes}
#
hostname_lookups = no

#  Core dumps are a bad thing.  This should only be set to 'yes'
#  if you're debugging a problem with the server.
#
#  allowed values: {no, yes}
#
allow_core_dumps = no

#  Regular expressions
#
#  These items are set at configure time.  If they're set to "yes",
#  then setting them to "no" turns off regular expression support.
#
#  If they're set to "no" at configure time, then setting them to "yes"
#  WILL NOT WORK.  It will give you an error.
#
regular_expressions     = yes
extended_expressions    = yes

#
#  Logging section.  The various "log_*" configuration items
#  will eventually be moved here.
#
log {
        #
        #  Destination for log messages.  This can be one of:
```

```
#
#       files - log to "file", as defined below.
#       syslog - to syslog (see also the "syslog_facility", below.
#       stdout - standard output
#       stderr - standard error.
#
#  The command-line option "-X" over-rides this option, and forces
#  logging to go to stdout.
#
destination = files

#
#  The logging messages for the server are appended to the
#  tail of this file if destination == "files"
#
#  If the server is running in debugging mode, this file is
#  NOT used.
#
file = ${logdir}/radius.log

#
#  If this configuration parameter is set, then log messages for
#  a *request* go to this file, rather than to radius.log.
#
#  i.e. This is a log file per request, once the server has accepted
#  the request as being from a valid client.  Messages that are
#  not associated with a request still go to radius.log.
#
#  Not all log messages in the server core have been updated to use
#  this new internal API.  As a result, some messages will still
#  go to radius.log.  Please submit patches to fix this behavior.
#
#  The file name is expanded dynamically.  You should ONLY user
#  server-side attributes for the filename (e.g. things you control).
#  Using this feature MAY also slow down the server substantially,
#  especially if you do thinks like SQL calls as part of the
#  expansion of the filename.
#
#  The name of the log file should use attributes that don't change
#  over the lifetime of a request, such as User-Name,
#  Virtual-Server or Packet-Src-IP-Address.  Otherwise, the log
#  messages will be distributed over multiple files.
#
#  Logging can be enabled for an individual request by a special
#  dynamic expansion macro:  %{debug: 1}, where the debug level
#  for this request is set to '1' (or 2, 3, etc.).  e.g.
#
#       ...
#       update control {
#            Tmp-String-0 = "%{debug:1}"
#       }
#       ...
#
```

```
        #  The attribute that the value is assigned to is unimportant,
        #  and should be a "throw-away" attribute with no side effects.
        #
        #requests = ${logdir}/radiusd-%{%{Virtual-Server}:-DEFAULT}-%Y%m%d.log

        #
        #  Which syslog facility to use, if ${destination} == "syslog"
        #
        #  The exact values permitted here are OS-dependent.  You probably
        #  don't want to change this.
        #
        syslog_facility = daemon

        #  Log the full User-Name attribute, as it was found in the request.
        #
        # allowed values: {no, yes}
        #
        stripped_names = yes

        #  Log authentication requests to the log file.
        #
        #  allowed values: {no, yes}
        #
        auth = yes

        #  Log passwords with the authentication requests.
        #  auth_badpass  - logs password if it's rejected
        #  auth_goodpass - logs password if it's correct
        #
        #  allowed values: {no, yes}
        #
        auth_badpass = yes
        auth_goodpass = yes

        #  Log additional text at the end of the "Login OK" messages.
        #  for these to work, the "auth" and "auth_goopass" or "auth_badpass"
        #  configurations above have to be set to "yes".
        #
        #  The strings below are dynamically expanded, which means that
        #  you can put anything you want in them.  However, note that
        #  this expansion can be slow, and can negatively impact server
        #  performance.
        #
#       msg_goodpass = ""
#       msg_badpass = ""
}

#  The program to execute to do concurrency checks.
checkrad = ${sbindir}/checkrad

# SECURITY CONFIGURATION
#
#  There may be multiple methods of attacking on the server.  This
```

```
#  section holds the configuration items which minimize the impact
#  of those attacks
#
security {
        #
        #  max_attributes: The maximum number of attributes
        #  permitted in a RADIUS packet.  Packets which have MORE
        #  than this number of attributes in them will be dropped.
        #
        #  If this number is set too low, then no RADIUS packets
        #  will be accepted.
        #
        #  If this number is set too high, then an attacker may be
        #  able to send a small number of packets which will cause
        #  the server to use all available memory on the machine.
        #
        #  Setting this number to 0 means "allow any number of attributes"
        max_attributes = 200

        #
        #  reject_delay: When sending an Access-Reject, it can be
        #  delayed for a few seconds.  This may help slow down a DoS
        #  attack.  It also helps to slow down people trying to brute-force
        #  crack a users password.
        #
        #  Setting this number to 0 means "send rejects immediately"
        #
        #  If this number is set higher than 'cleanup_delay', then the
        #  rejects will be sent at 'cleanup_delay' time, when the request
        #  is deleted from the internal cache of requests.
        #
        #  Useful ranges: 1 to 5
        reject_delay = 1

        #
        #  status_server: Whether or not the server will respond
        #  to Status-Server requests.
        #
        #  When sent a Status-Server message, the server responds with
        #  an Access-Accept or Accounting-Response packet.
        #
        #  This is mainly useful for administrators who want to "ping"
        #  the server, without adding test users, or creating fake
        #  accounting packets.
        #
        #  It's also useful when a NAS marks a RADIUS server "dead".
        #  The NAS can periodically "ping" the server with a Status-Server
        #  packet.  If the server responds, it must be alive, and the
        #  NAS can start using it for real requests.
        #
        #  See also raddb/sites-available/status
        #
        status_server = no
```

```
}

# PROXY CONFIGURATION
#
#  proxy_requests: Turns proxying of RADIUS requests on or off.
#
#  The server has proxying turned on by default.  If your system is NOT
#  set up to proxy requests to another server, then you can turn proxying
#  off here.  This will save a small amount of resources on the server.
#
#  If you have proxying turned off, and your configuration files say
#  to proxy a request, then an error message will be logged.
#
#  To disable proxying, change the "yes" to "no", and comment the
#  $INCLUDE line.
#
#  allowed values: {no, yes}
#
proxy_requests  = yes
$INCLUDE proxy.conf


# CLIENTS CONFIGURATION
#
#  Client configuration is defined in "clients.conf".
#

#  The 'clients.conf' file contains all of the information from the old
#  'clients' and 'naslist' configuration files.  We recommend that you
#  do NOT use 'client's or 'naslist', although they are still
#  supported.
#
#  Anything listed in 'clients.conf' will take precedence over the
#  information from the old-style configuration files.
#
$INCLUDE clients.conf


# THREAD POOL CONFIGURATION
#
#  The thread pool is a long-lived group of threads which
#  take turns (round-robin) handling any incoming requests.
#
#  You probably want to have a few spare threads around,
#  so that high-load situations can be handled immediately.  If you
#  don't have any spare threads, then the request handling will
#  be delayed while a new thread is created, and added to the pool.
#
#  You probably don't want too many spare threads around,
#  otherwise they'll be sitting there taking up resources, and
#  not doing anything productive.
#
#  The numbers given below should be adequate for most situations.
```

```
#
thread pool {
        #  Number of servers to start initially --- should be a reasonable
        #  ballpark figure.
        start_servers = 5

        #  Limit on the total number of servers running.
        #
        #  If this limit is ever reached, clients will be LOCKED OUT, so it
        #  should NOT BE SET TOO LOW.  It is intended mainly as a brake to
        #  keep a runaway server from taking the system with it as it spirals
        #  down...
        #
        #  You may find that the server is regularly reaching the
        #  'max_servers' number of threads, and that increasing
        #  'max_servers' doesn't seem to make much difference.
        #
        #  If this is the case, then the problem is MOST LIKELY that
        #  your back-end databases are taking too long to respond, and
        #  are preventing the server from responding in a timely manner.
        #
        #  The solution is NOT do keep increasing the 'max_servers'
        #  value, but instead to fix the underlying cause of the
        #  problem: slow database, or 'hostname_lookups=yes'.
        #
        #  For more information, see 'max_request_time', above.
        #
        max_servers = 32

        #  Server-pool size regulation.  Rather than making you guess
        #  how many servers you need, FreeRADIUS dynamically adapts to
        #  the load it sees, that is, it tries to maintain enough
        #  servers to handle the current load, plus a few spare
        #  servers to handle transient load spikes.
        #
        #  It does this by periodically checking how many servers are
        #  waiting for a request.  If there are fewer than
        #  min_spare_servers, it creates a new spare.  If there are
        #  more than max_spare_servers, some of the spares die off.
        #  The default values are probably OK for most sites.
        #
        min_spare_servers = 3
        max_spare_servers = 10

        #  There may be memory leaks or resource allocation problems with
        #  the server.  If so, set this value to 300 or so, so that the
        #  resources will be cleaned up periodically.
        #
        #  This should only be necessary if there are serious bugs in the
        #  server which have not yet been fixed.
        #
        #  '0' is a special value meaning 'infinity', or 'the servers never
        #  exit'
```

```
        max_requests_per_server = 0
}

# MODULE CONFIGURATION
#
#  The names and configuration of each module is located in this section.
#
#  After the modules are defined here, they may be referred to by name,
#  in other sections of this configuration file.
#
modules {
        #
        #  Each module has a configuration as follows:
        #
        #       name [ instance ] {
        #               config_item = value
        #               ...
        #       }
        #
        #  The 'name' is used to load the 'rlm_name' library
        #  which implements the functionality of the module.
        #
        #  The 'instance' is optional.  To have two different instances
        #  of a module, it first must be referred to by 'name'.
        #  The different copies of the module are then created by
        #  inventing two 'instance' names, e.g. 'instance1' and 'instance2'
        #
        #  The instance names can then be used in later configuration
        #  INSTEAD of the original 'name'.  See the 'radutmp' configuration
        #  for an example.
        #

        #
        #  As of 2.0.5, most of the module configurations are in a
        #  sub-directory.  Files matching the regex /[a-zA-Z0-9_.]+/
        #  are loaded.  The modules are initialized ONLY if they are
        #  referenced in a processing section, such as authorize,
        #  authenticate, accounting, pre/post-proxy, etc.
        #
        $INCLUDE ${confdir}/modules/

        #  Extensible Authentication Protocol
        #
        #  For all EAP related authentications.
        #  Now in another file, because it is very large.
        #
        $INCLUDE eap.conf

        #  Include another file that has the SQL-related configuration.
        #  This is another file only because it tends to be big.
        #
        $INCLUDE sql.conf
```

```
        #
        #  This module is an SQL enabled version of the counter module.
        #
        #  Rather than maintaining seperate (GDBM) databases of
        #  accounting info for each counter, this module uses the data
        #  stored in the raddacct table by the sql modules. This
        #  module NEVER does any database INSERTs or UPDATEs.  It is
        #  totally dependent on the SQL module to process Accounting
        #  packets.
        #
#       $INCLUDE sql/mysql/counter.conf

        #
        #  IP addresses managed in an SQL table.
        #
#       $INCLUDE sqlippool.conf
}

# Instantiation
#
#  This section orders the loading of the modules.  Modules
#  listed here will get loaded BEFORE the later sections like
#  authorize, authenticate, etc. get examined.
#
#  This section is not strictly needed.  When a section like
#  authorize refers to a module, it's automatically loaded and
#  initialized.  However, some modules may not be listed in any
#  of the following sections, so they can be listed here.
#
#  Also, listing modules here ensures that you have control over
#  the order in which they are initalized.  If one module needs
#  something defined by another module, you can list them in order
#  here, and ensure that the configuration will be OK.
#
instantiate {
        #
        #  Allows the execution of external scripts.
        #  The entire command line (and output) must fit into 253 bytes.
        #
        #  e.g. Framed-Pool = `%{exec:/bin/echo foo}`
        exec

        #
        #  The expression module doesn't do authorization,
        #  authentication, or accounting.  It only does dynamic
        #  translation, of the form:
        #
        #       Session-Timeout = `%{expr:2 + 3}`
        #
        #  So the module needs to be instantiated, but CANNOT be
        #  listed in any other section.  See 'doc/rlm_expr' for
        #  more information.
        #
```

```
        expr

        #
        # We add the counter module here so that it registers
        # the check-name attribute before any module which sets
        # it
#       daily
        expiration
        logintime

        # subsections here can be thought of as "virtual" modules.
        #
        # e.g. If you have two redundant SQL servers, and you want to
        # use them in the authorize and accounting sections, you could
        # place a "redundant" block in each section, containing the
        # exact same text.  Or, you could uncomment the following
        # lines, and list "redundant_sql" in the authorize and
        # accounting sections.
        #
        #redundant redundant_sql {
        #       sql1
        #       sql2
        #}
}

######################################################################
#
#       Policies that can be applied in multiple places are listed
#       globally.  That way, they can be defined once, and referred
#       to multiple times.
#
######################################################################
$INCLUDE policy.conf

######################################################################
#
#       Load virtual servers.
#
#       This next $INCLUDE line loads files in the directory that
#       match the regular expression: /[a-zA-Z0-9_.]+/
#
#       It allows you to define new virtual servers simply by placing
#       a file into the raddb/sites-enabled/ directory.
#
$INCLUDE sites-enabled/

######################################################################
#
#       All of the other configuration sections like "authorize {}",
#       "authenticate {}", "accounting {}", have been moved to the
#       the file:
#
#               raddb/sites-available/default
```

```
#
#       This is the "default" virtual server that has the same
#       configuration as in version 1.0.x and 1.1.x.  The default
#       installation enables this virtual server.  You should
#       edit it to create policies for your local site.
#
#       For more documentation on virtual servers, see:
#
#                raddb/sites-available/README
#
######################################################################
```

ANEXO 2. users

```
#
#       Please read the documentation file ../doc/processing_users_file,
#       or 'man 5 users' (after installing the server) for more information.
#
#       This file contains authentication security and configuration
#       information for each user.  Accounting requests are NOT processed
#       through this file.  Instead, see 'acct_users', in this directory.
#
#       The first field is the user's name and can be up to
#       253 characters in length.  This is followed (on the same line) with
#       the list of authentication requirements for that user.  This can
#       include password, comm server name, comm server port number, protocol
#       type (perhaps set by the "hints" file), and huntgroup name (set by
#       the "huntgroups" file).
#
#       If you are not sure why a particular reply is being sent by the
#       server, then run the server in debugging mode (radiusd -X), and
#       you will see which entries in this file are matched.
#
#       When an authentication request is received from the comm server,
#       these values are tested. Only the first match is used unless the
#       "Fall-Through" variable is set to "Yes".
#
#       A special user named "DEFAULT" matches on all usernames.
#       You can have several DEFAULT entries. All entries are processed
#       in the order they appear in this file. The first entry that
#       matches the login-request will stop processing unless you use
#       the Fall-Through variable.
#
#       If you use the database support to turn this file into a .db or .dbm
#       file, the DEFAULT entries _have_ to be at the end of this file and
#       you can't have multiple entries for one username.
#
#       Indented (with the tab character) lines following the first
#       line indicate the configuration values to be passed back to
#       the comm server to allow the initi#steve    Cleartext-Password := "testing"
#       Service-Type = Framed-User,
#       Framed-Protocol = PPP,
#       Framed-IP-Address = 172.16.3.33,
#       Framed-IP-Netmask = 255.255.255.0,
#       Framed-Routing = Broadcast-Listen,
#       Framed-Filter-Id = "std.ppp",
#       Framed-MTU = 1500,
#       Framed-Compression = Van-Jacobsen-TCP-IPation of a user session.
#       This can include things like the PPP configuration values
#       or the host to log the user onto.
#
#       You can include another `users' file with `$INCLUDE users.other'
```

```
#


#
#       For a list of RADIUS attributes, and links to their definitions,
#       see:
#
#       http://www.freeradius.org/rfc/attributes.html
#


#
# Deny access for a specific user.  Note that this entry MUST
# be before any other 'Auth-Type' attribute which results in the user
# being authenticated.
#
# Note that there is NO 'Fall-Through' attribute, so the user will not
# be given any additional resources.
#
#lameuser       Auth-Type := Reject
#               Reply-Message = "Your account has been disabled."

#
# Deny access for a group of users.
#
# Note that there is NO 'Fall-Through' attribute, so the user will not
# be given any additional resources.
#
#DEFAULT        Group == "disabled", Auth-Type := Reject
#               Reply-Message = "Your account has been disabled."
#

#
# This is a complete entry for "steve". Note that there is no Fall-Through
# entry so that no DEFAULT entry will be used, and the user will NOT
# get any attributes in addition to the ones listed here.
#

usuario  Cleartext-Password := "123456"
         Service-Type = Framed-User,
         Framed-Protocol = PPP,
#        Framed-IP-Address = 172.16.3.33,
#        Framed-IP-Netmask = 255.255.255.0,
#        Framed-Routing = Broadcast-Listen,
#        Framed-Filter-Id = "std.ppp",
#        Framed-MTU = 1500,
         Framed-Compression = Van-Jacobsen-TCP-IP

usuario1 Auth-Type := EAP
usuario2 Auth-Type := reject


#steve   Cleartext-Password := "testing"
#        Service-Type = Framed-User,
#        Framed-Protocol = PPP,
```

```
#       Framed-IP-Address = 172.16.3.33,
#       Framed-IP-Netmask = 255.255.255.0,
#       Framed-Routing = Broadcast-Listen,
#       Framed-Filter-Id = "std.ppp",
#       Framed-MTU = 1500,
#       Framed-Compression = Van-Jacobsen-TCP-IP

#
# This is an entry for a user with a space in their name.
# Note the double quotes surrounding the name.
#
#"John Doe"     Cleartext-Password := "hello"
#               Reply-Message = "Hello, %{User-Name}"

#
# Dial user back and telnet to the default host for that port
#
#Deg    Cleartext-Password := "ge55ged"
#       Service-Type = Callback-Login-User,
#       Login-IP-Host = 0.0.0.0,
#       Callback-Number = "9,5551212",
#       Login-Service = Telnet,
#       Login-TCP-Port = Telnet

#
# Another complete entry. After the user "dialbk" has logged in, the
# connection will be broken and the user will be dialed back after which
# he will get a connection to the host "timeshare1".
#
#dialbk Cleartext-Password := "callme"
#       Service-Type = Callback-Login-User,
#       Login-IP-Host = timeshare1,
#       Login-Service = PortMaster,
#       Callback-Number = "9,1-800-555-1212"

#
# user "swilson" will only get a static IP number if he logs in with
# a framed protocol on a terminal server in Alphen (see the huntgroups file).
#
# Note that by setting "Fall-Through", other attributes will be added from
# the following DEFAULT entries
#
#swilson        Service-Type == Framed-User, Huntgroup-Name == "alphen"
#               Framed-IP-Address = 192.168.1.65,
#               Fall-Through = Yes

#
# If the user logs in as 'username.shell', then authenticate them
# using the default method, give them shell access, and stop processing
# the rest of the file.
#
#DEFAULT        Suffix == ".shell"
#               Service-Type = Login-User,
```

```
#                       Login-Service = Telnet,
#                       Login-IP-Host = your.shell.machine


#
# The rest of this file contains the several DEFAULT entries.
# DEFAULT entries match with all login names.
# Note that DEFAULT entries can also Fall-Through (see first entry).
# A name-value pair from a DEFAULT entry will _NEVER_ override
# an already existing name-value pair.
#

#
# Set up different IP address pools for the terminal servers.
# Note that the "+" behind the IP address means that this is the "base"
# IP address. The Port-Id (S0, S1 etc) will be added to it.
#
#DEFAULT        Service-Type == Framed-User, Huntgroup-Name == "alphen"
#                       Framed-IP-Address = 192.168.1.32+,
#                       Fall-Through = Yes

#DEFAULT        Service-Type == Framed-User, Huntgroup-Name == "delft"
#                       Framed-IP-Address = 192.168.2.32+,
#                       Fall-Through = Yes

#
# Sample defaults for all framed connections.
#
#DEFAULT        Service-Type == Framed-User
#       Framed-IP-Address = 255.255.255.254,
#       Framed-MTU = 576,
#       Service-Type = Framed-User,
#       Fall-Through = Yes

#
# Default for PPP: dynamic IP address, PPP mode, VJ-compression.
# NOTE: we do not use Hint = "PPP", since PPP might also be auto-detected
#       by the terminal server in which case there may not be a "P" suffix.
#       The terminal server sends "Framed-Protocol = PPP" for auto PPP.
#
DEFAULT         Framed-Protocol == PPP
        Framed-Protocol = PPP,
        Framed-Compression = Van-Jacobson-TCP-IP

#
# Default for CSLIP: dynamic IP address, SLIP mode, VJ-compression.
#
DEFAULT         Hint == "CSLIP"
        Framed-Protocol = SLIP,
        Framed-Compression = Van-Jacobson-TCP-IP

#
# Default for SLIP: dynamic IP address, SLIP mode.
```

```
#
DEFAULT         Hint == "SLIP"
        Framed-Protocol = SLIP

#
# Last default: rlogin to our main server.
#
#DEFAULT
#       Service-Type = Login-User,
#       Login-Service = Rlogin,
#       Login-IP-Host = shellbox.ispdomain.com

# #
# # Last default: shell on the local terminal server.
# #
# DEFAULT
#       Service-Type = Administrative-User

# On no match, the user is denied access.
```

ANEXO 3. clients.conf

```
# -*- text -*-
##
## clients.conf -- client configuration directives
##
##	$Id$

#######################################################################
#
#  Define RADIUS clients (usually a NAS, Access Point, etc.).

#
#  Defines a RADIUS client.
#
#  '127.0.0.1' is another name for 'localhost'.  It is enabled by default,
#  to allow testing of the server after an initial installation.  If you
#  are not going to be permitting RADIUS queries from localhost, we suggest
#  that you delete, or comment out, this entry.
#
#


#
#  Each client has a "short name" that is used to distinguish it from
#  other clients.
#
#  In version 1.x, the string after the word "client" was the IP
#  address of the client.  In 2.0, the IP address is configured via
#  the "ipaddr" or "ipv6addr" fields.  For compatibility, the 1.x
#  format is still accepted.
#
client localhost {
	#  Allowed values are:
	#	dotted quad (1.2.3.4)
	#	hostname    (radius.example.com)
	ipaddr = 127.0.0.1

	#  OR, you can use an IPv6 address, but not both
	#  at the same time.
#	ipv6addr = ::	# any.  ::1 == localhost

	#
	#  A note on DNS:  We STRONGLY recommend using IP addresses
	#  rather than host names.  Using host names means that the
	#  server will do DNS lookups when it starts, making it
	#  dependent on DNS.  i.e. If anything goes wrong with DNS,
	#  the server won't start!
	#
	#  The server also looks up the IP address from DNS once, and
```

```
        #  only once, when it starts.  If the DNS record is later
        #  updated, the server WILL NOT see that update.
        #

        #  One client definition can be applied to an entire network.
        #  e.g. 127/8 should be defined with "ipaddr = 127.0.0.0" and
        #  "netmask = 8"
        #
        #  If not specified, the default netmask is 32 (i.e. /32)
        #
        #  We do NOT recommend using anything other than 32.  There
        #  are usually other, better ways to achieve the same goal.
        #  Using netmasks of other than 32 can cause security issues.
        #
        #  You can specify overlapping networks (127/8 and 127.0/16)
        #  In that case, the smallest possible network will be used
        #  as the "best match" for the client.
        #
        #  Clients can also be defined dynamically at run time, based
        #  on any criteria.  e.g. SQL lookups, keying off of NAS-Identifier,
        #  etc.
        #  See raddb/sites-available/dynamic-clients for details.
        #

#       netmask = 32

        #
        #  The shared secret use to "encrypt" and "sign" packets between
        #  the NAS and FreeRADIUS.  You MUST change this secret from the
        #  default, otherwise it's not a secret any more!
        #
        #  The secret can be any string, up to 8k characters in length.
        #
        #  Control codes can be entered vi octal encoding,
        #       e.g. "\101\102" == "AB"
        #  Quotation marks can be entered by escaping them,
        #       e.g. "foo\"bar"
        #
        #  A note on security:  The security of the RADIUS protocol
        #  depends COMPLETELY on this secret!  We recommend using a
        #  shared secret that is composed of:
        #
        #       upper case letters
        #       lower case letters
        #       numbers
        #
        #  And is at LEAST 8 characters long, preferably 16 characters in
        #  length.  The secret MUST be random, and should not be words,
        #  phrase, or anything else that is recognizable.
        #
        #  The default secret below is only for testing, and should
        #  not be used in any real environment.
        #
```

```
        secret          = testing123

        #
        #  Old-style clients do not send a Message-Authenticator
        #  in an Access-Request.  RFC 5080 suggests that all clients
        #  SHOULD include it in an Access-Request.  The configuration
        #  item below allows the server to require it.  If a client
        #  is required to include a Message-Authenticator and it does
        #  not, then the packet will be silently discarded.
        #
        #  allowed values: yes, no
        require_message_authenticator = no

        #
        #  The short name is used as an alias for the fully qualified
        #  domain name, or the IP address.
        #
        #  It is accepted for compatibility with 1.x, but it is no
        #  longer necessary in 2.0
        #
        shortname       = localhost

        #
        # the following three fields are optional, but may be used by
        # checkrad.pl for simultaneous use checks
        #

        #
        # The nastype tells 'checkrad.pl' which NAS-specific method to
        #  use to query the NAS for simultaneous use.
        #
        #  Permitted NAS types are:
        #
        #       cisco
        #       computone
        #       livingston
        #       max40xx
        #       multitech
        #       netserver
        #       pathras
        #       patton
        #client 10.10.10.10 {
#       # secret and password are mapped through the "secrets" file.
#       secret      = testing123
#       shortname   = liv1
#     # the following three fields are optional, but may be used by
#     # checkrad.pl for simultaneous usage checks
#       nastype     = livingston
#       login       = !root
#       password    = someadminpas
#}#     portslave
        #       tc
        #       usrhiper
```

```
        #       other           # for all other types

        #
        nastype     = other     # localhost isn't usually a NAS...

        #
        #  The following two configurations are for future use.
        #  The 'naspasswd' file is currently used to store the NAS
        #  login name and password, which is used by checkrad.pl
        #  when querying the NAS for simultaneous use.
        #
#       login       = !root
#       password    = someadminpas

        #
        #  As of 2.0, clients can also be tied to a virtual server.
        #  This is done by setting the "virtual_server" configuration
        #  item, as in the example below.
        #
#       virtual_server = home1

        #
        #  A pointer to the "home_server_pool" OR a "home_server"
        #  section that contains the CoA configuration for this
        #  client.  For an example of a coa home server or pool,
        #  see raddb/sites-available/originate-coa
#       coa_server = coa
}

# IPv6 Client
#client ::1 {
#       secret          = testing123
#       shortname       = localhost
#}
#
# All IPv6 Site-local clients
#client fe80::/16 {
#       secret          = testing123
#       shortname       = localhost
#}

#client some.host.org {
#       secret          = testing123
#       shortname       = localhost
#}

#
#  You can now specify one secret for a network of clients.
#  When a client request comes in, the BEST match is chosen.
#  i.e. The entry from the smallest possible network.
#
#client 192.168.0.0/24 {
#       secret          = testing123-1
```

```
#       shortname       = private-network-1
#}
#
#client 192.168.0.0/16 {
#       secret          = testing123-2
#       shortname       = private-network-2
#}

client 192.168.2.1 {
#       # secret and password are mapped through the "secrets" file.
        secret      = testing123
        shortname   = dd-wrt
#     # the following three fields are optional, but may be used by
#     # checkrad.pl for simultaneous usage checks
        nastype     = other
#       login       = !root
#       password    = someadminpas      ipaddr = 127.0.0.1

}

#client 10.10.10.10 {
#       # secret and password are mapped through the "secrets" file.
#       secret      = testing123
#       shortname   = liv1
#     # the following three fields are optional, but may be used by
#     # checkrad.pl for simultaneous usage checks
#       nastype     = livingston
#       login       = !root ipaddr = 127.0.0.1

#       password    = someadminpas
#}

#############################################################################
#
#  Per-socket client lists.  The configuration entries are exactly
#  the same as above, but they are nested inside of a section.
#
#  You can have as many per-socket client lists as you have "listen"
#  sections, or you can re-use a list among multiple "listen" sections.          ipaddr = 127.0.0.1

#
#  Un-comment this section, and edit a "listen" section to add:
#  "clients = per_socket_clients".  That IP address/port combination
#  will then accept ONLY the clients listed in this section.  ipaddr = 127.0.0.1

#
#clients per_socket_clients {
#       client 192.168.3.4 {
#               secret = testing123
#     }
#}
```

ANEXO 4. eap.conf

```
# -*- text -*-
##
##  eap.conf -- Configuration for EAP types (PEAP, TTLS, etc.)
##
##      $Id$

######################################################################
#
#  Whatever you do, do NOT set 'Auth-Type := EAP'.  The server
#  is smart enough to figure this out on its own.  The most
#  common side effect of setting 'Auth-Type := EAP' is that the
#  users then cannot use ANY other authentication method.
#
#  EAP types NOT listed here may be supported via the "eap2" module.
#  See experimental.conf for documentation.
#
        eap {
                #  Invoke the default supported EAP type when
                #  EAP-Identity response is received.
                #
                #  The incoming EAP messages DO NOT specify which EAP
                #  type they will be using, so it MUST be set here.
                #
                #  For now, only one default EAP type may be used at a time.
                #
                #  If the EAP-Type attribute is set by another module,
                #  then that EAP type takes precedence over the
                #  default type configured here.
                #
        default_eap_type = tls
        #default_eap_type = mschapv2

                #default_eap_type = md5

                #  A list is maintained to correlate EAP-Response
                #  packets with EAP-Request packets.  After a
                #  configurable length of time, entries in the list
                #  expire, and are deleted.
                #
                timer_expire     = 60

                #  There are many EAP types, but the server has support
                #  for only a limited subset.  If the server receives
                #  a request for an EAP type it does not support, then
                #  it normally rejects the request.  By setting this
                #  configuration to "yes", you can tell the server to
                #  instead keep processing the request.  Another module
                #  MUST then be configured to proxy the request to
```

```
        #  another RADIUS server which supports that EAP type.
        #
        #  If another module is NOT configured to handle the
        #  request, then the request will still end up being
        #  rejected.
        ignore_unknown_eap_types = no

        # Cisco AP1230B firmware 12.2(13)JA1 has a bug.  When given
        # a User-Name attribute in an Access-Accept, it copies one
        # more byte than it should.
        #
        # We can work around it by configurably adding an extra
        # zero byte.
        cisco_accounting_username_bug = no

        #
        #  Help prevent DoS attacks by limiting the number of
        #  sessions that the server is tracking.  Most systems
        #  can handle ~30 EAP sessions/s, so the default limit
        #  of 4096 should be OK.
        max_sessions = 4096

        # Supported EAP-types

        #
        #  We do NOT recommend using EAP-MD5 authentication
        #  for wireless connections.  It is insecure, and does
        #  not provide for dynamic WEP keys.
        #
md5 {
        }

        # Cisco LEAP
        #
        #  We do not recommend using LEAP in new deployments.  See:
        #  http://www.securiteam.com/tools/5TP012ACKE.html
        #
        #  Cisco LEAP uses the MS-CHAP algorithm (but not
        #  the MS-CHAP attributes) to perform it's authentication.
        #
        #  As a result, LEAP *requires* access to the plain-text
        #  User-Password, or the NT-Password attributes.
        #  'System' authentication is impossible with LEAP.
        #
        leap {
        }

        #  Generic Token Card.
        #
        #  Currently, this is only permitted inside of EAP-TTLS,
        #  or EAP-PEAP.  The module "challenges" the user with
        #  text, and the response from the user is taken to be
        #  the User-Password.
```

```
#
#  Proxying the tunneled EAP-GTC session is a bad idea,
#  the users password will go over the wire in plain-text,
#  for anyone to see.
#
gtc {
        #  The default challenge, which many clients
        #  ignore..
        #challenge = "Password: "

        #  The plain-text response which comes back
        #  is put into a User-Password attribute,
        #  and passed to another module for
        #  authentication.  This allows the EAP-GTC
        #  response to be checked against plain-text,
        #  or crypt'd passwords.
        #
        #  If you say "Local" instead of "PAP", then
        #  the module will look for a User-Password
        #  configured for the request, and do the
        #  authentication itself.
        #
        auth_type = PAP
}

## EAP-TLS
#
#  See raddb/certs/README for additional comments
#  on certificates.
#
#  If OpenSSL was not found at the time the server was
#  built, the "tls", "ttls", and "peap" sections will
#  be ignored.
#
#  Otherwise, when the server first starts in debugging
#  mode, test certificates will be created.  See the
#  "make_cert_command" below for details, and the README
#  file in raddb/certs
#
#  These test certificates SHOULD NOT be used in a normal
#  deployment.  They are created only to make it easier
#  to install the server, and to perform some simple
#  tests with EAP-TLS, TTLS, or PEAP.
#
#  See also:
#
#  http://www.dslreports.com/forum/remark,9286052~mode=flat
#
#  Note that you should NOT use a globally known CA here!
#  e.g. using a Verisign cert as a "known CA" means that
#  ANYONE who has a certificate signed by them can
#  authenticate via EAP-TLS!  This is likey not what you want.
tls {
```

```
#
#  These is used to simplify later configurations.
#
certdir = /etc/certs/server
cadir = ${confdir}/certs/masterCA



private_key_password = misecreto
private_key_file = /etc/certs/server/radius1_keycert.pem



#  If Private key & Certificate are located in
#  the same file, then private_key_file &
#  certificate_file must contain the same file
#  name.
#
#  If CA_file (below) is not used, then the
#  certificate_file below MUST include not
#  only the server certificate, but ALSO all
#  of the CA certificates used to sign the
#  server certificate.

certificate_file = /etc/certs/server/radius1_keycert.pem

#  Trusted Root CA list
#
#  ALL of the CA's in this list will be trusted
#  to issue client certificates for authentication.
#
#  In general, you should use self-signed
#  certificates for 802.1x (EAP) authentication.
#  In that case, this CA file should contain
#  *one* CA certificate.
#
#  This parameter is used only for EAP-TLS,
#  when you issue client certificates.  If you do
#  not use client certificates, and you do not want
#  to permit EAP-TLS authentication, then delete
#  this configuration item.
CA_file = ${cadir}/cacert.pem
#
#  For DH cipher suites to work, you have to
#  run OpenSSL to create the DH file first:
#
#       openssl dhparam -out certs/dh 1024
#

dh_file = /etc/certs/dh
random_file = /etc/cets/random
rsa_key_length = 1024
dh_key_length = 1024
```

```
        #
        #  This can never exceed the size of a RADIUS
        #  packet (4096 bytes), and is preferably half
        #  that, to accomodate other attributes in
        #  RADIUS packet.  On most APs the MAX packet
        #  length is configured between 1500 - 1600
        #  In these cases, fragment size should be
        #  1024 or less.
        #
        fragment_size = 1024

        #  include_length is a flag which is
        #  by default set to yes If set to
        #  yes, Total Length of the message is
        #  included in EVERY packet we send.
        #  If set to no, Total Length of the
        #  message is included ONLY in the
        #  First packet of a fragment series.
        #
#       include_length = yes

        #  Check the Certificate Revocation List
        #
        #  1) Copy CA certificates and CRLs to same directory.
        #  2) Execute 'c_rehash <CA certs&CRLs Directory>'.
        #    'c_rehash' is OpenSSL's command.
        #  3) uncomment the line below.
        #  5) Restart radiusd
#       check_crl = yes
        CA_path = ${cadir}

    #
    #  If check_cert_issuer is set, the value will
    #  be checked against the DN of the issuer in
    #  the client certificate.  If the values do not
    #  match, the cerficate verification will fail,
    #  rejecting the user.
    #
    #  In 2.1.10 and later, this check can be done
    #  more generally by checking the value of the
    #  TLS-Client-Cert-Issuer attribute.  This check
    #  can be done via any mechanism you choose.
    #
#    check_cert_issuer = "/C=GB/ST=Berkshire/L=Newbury/O=My Company Ltd"

    #
    #  If check_cert_cn is set, the value will
    #  be xlat'ed and checked against the CN
    #  in the client certificate.  If the values
    #  do not match, the certificate verification
    #  will fail rejecting the user.
    #
    #  This check is done only if the previous
```

```
        #  "check_cert_issuer" is not set, or if
        #  the check succeeds.
        #
        #  In 2.1.10 and later, this check can be done
        #  more generally by checking the value of the
        #  TLS-Client-Cert-CN attribute.  This check
        #  can be done via any mechanism you choose.
        #
#       check_cert_cn = %{User-Name}
#
        # Set this option to specify the allowed
        # TLS cipher suites.  The format is listed
        # in "man 1 ciphers".
        cipher_list = "DEFAULT"

        #

        #  This configuration entry should be deleted
        #  once the server is running in a normal
        #  configuration.  It is here ONLY to make
        #  initial deployments easier.
        #
        make_cert_command = "${certdir}/bootstrap"

        #
        #  Session resumption / fast reauthentication
        #  cache.
        #
        #  The cache contains the following information:
        #
        #  session Id - unique identifier, managed by SSL
        #  User-Name  - from the Access-Accept
        #  Stripped-User-Name - from the Access-Request
        #  Cached-Session-Policy - from the Access-Accept
        #
        #  The "Cached-Session-Policy" is the name of a
        #  policy which should be applied to the cached
        #  session.  This policy can be used to assign
        #  VLANs, IP addresses, etc.  It serves as a useful
        #  way to re-apply the policy from the original
        #  Access-Accept to the subsequent Access-Accept
        #  for the cached session.
        #
        #  On session resumption, these attributes are
        #  copied from the cache, and placed into the
        #  reply list.
        #
        #  You probably also want "use_tunneled_reply = yes"
        #  when using fast session resumption.
        #
        cache {
            #
            #  Enable it.  The default is "no".
```

```
    # Deleting the entire "cache" subsection
    # Also disables caching.
    #
    # You can disallow resumption for a
    # particular user by adding the following
    # attribute to the control item list:
    #
    #               Allow-Session-Resumption = No
    #
    # If "enable = no" below, you CANNOT
    # enable resumption for just one user
    # by setting the above attribute to "yes".
    #
    enable = no

    #
    # Lifetime of the cached entries, in hours.
    # The sessions will be deleted after this
    # time.
    #
    lifetime = 24 # hours

    #
    # The maximum number of entries in the
    # cache.  Set to "0" for "infinite".
    #
    # This could be set to the number of users
    # who are logged in... which can be a LOT.
    #
    max_entries = 255
}

#
# As of version 2.1.10, client certificates can be
# validated via an external command.  This allows
# dynamic CRLs or OCSP to be used.
#
# This configuration is commented out in the
# default configuration.  Uncomment it, and configure
# the correct paths below to enable it.
#
verify {
        # A temporary directory where the client
        # certificates are stored.  This directory
        # MUST be owned by the UID of the server,
        # and MUST not be accessible by any other
        # users.  When the server starts, it will do
        # "chmod go-rwx" on the directory, for
        # security reasons.  The directory MUST
        # exist when the server starts.
        #
        # You should also delete all of the files
        # in the directory when the server starts.
```

```
		#		tmpdir = /tmp/radiusd

				#  The command used to verify the client cert.
				#  We recommend using the OpenSSL command-line
				#  tool.
				#
				#  The ${..CA_path} text is a reference to
				#  the CA_path variable defined above.
				#
				#  The %{TLS-Client-Cert-Filename} is the name
				#  of the temporary file containing the cert
				#  in PEM format.  This file is automatically
				#  deleted by the server when the command
				#  returns.
		#		client = "/path/to/openssl verify -CApath ${..CA_path} %{TLS-
Client-Cert-Filename}"
		}
	}

	# The TTLS module implements the EAP-TTLS protocol,
	# which can be described as EAP inside of Diameter,
	# inside of TLS, inside of EAP, inside of RADIUS...
	#
	# Surprisingly, it works quite well.
	#
	# The TTLS module needs the TLS module to be installed
	# and configured, in order to use the TLS tunnel
	# inside of the EAP packet.  You will still need to
	# configure the TLS module, even if you do not want
	# to deploy EAP-TLS in your network.  Users will not
	# be able to request EAP-TLS, as it requires them to
	# have a client certificate.  EAP-TTLS does not
	# require a client certificate.
	#
	# You can make TTLS require a client cert by setting
	#
	#	EAP-TLS-Require-Client-Cert = Yes
	#
	# in the control items for a request.
	#
	ttls {
		# The tunneled EAP session needs a default
		# EAP type which is separate from the one for
		# the non-tunneled EAP module.  Inside of the
		# TTLS tunnel, we recommend using EAP-MD5.
		# If the request does not contain an EAP
		# conversation, then this configuration entry
		# is ignored.
		default_eap_type = md5

		# The tunneled authentication request does
		# not usually contain useful attributes
		# like 'Calling-Station-Id', etc.  These
```

```
		#  attributes are outside of the tunnel,
		#  and normally unavailable to the tunneled
		#  authentication request.
		#
		#  By setting this configuration entry to
		#  'yes', any attribute which NOT in the
		#  tunneled authentication request, but
		#  which IS available outside of the tunnel,
		#  is copied to the tunneled request.
		#
		# allowed values: {no, yes}
		copy_request_to_tunnel = no

		#  The reply attributes sent to the NAS are
		#  usually based on the name of the user
		#  'outside' of the tunnel (usually
		#  'anonymous').  If you want to send the
		#  reply attributes based on the user name
		#  inside of the tunnel, then set this
		#  configuration entry to 'yes', and the reply
		#  to the NAS will be taken from the reply to
		#  the tunneled request.
		#
		# allowed values: {no, yes}
		use_tunneled_reply = no

		#
		#  The inner tunneled request can be sent
		#  through a virtual server constructed
		#  specifically for this purpose.
		#
		#  If this entry is commented out, the inner
		#  tunneled request will be sent through
		#  the virtual server that processed the
		#  outer requests.
		#
		virtual_server = "inner-tunnel"

		#  This has the same meaning as the
		#  same field in the "tls" module, above.
		#  The default value here is "yes".
#		include_length = yes
}

######################################################
#
#  !!!!! WARNINGS for Windows compatibility  !!!!!
#
######################################################
#
#  If you see the server send an Access-Challenge,
#  and the client never sends another Access-Request,
#  then
```

```
#
#                       STOP!
#
#  The server certificate has to have special OID's
#  in it, or else the Microsoft clients will silently
#  fail.  See the "scripts/xpextensions" file for
#  details, and the following page:
#
#       http://support.microsoft.com/kb/814394/en-us
#
#  For additional Windows XP SP2 issues, see:
#
#       http://support.microsoft.com/kb/885453/en-us
#
#
#  If is still doesn't work, and you're using Samba,
#  you may be encountering a Samba bug.  See:
#
#       https://bugzilla.samba.org/show_bug.cgi?id=6563
#
#  Note that we do not necessarily agree with their
#  explanation... but the fix does appear to work.
#
##################################################

#
#  The tunneled EAP session needs a default EAP type
#  which is separate from the one for the non-tunneled
#  EAP module.  Inside of the TLS/PEAP tunnel, we
#  recommend using EAP-MS-CHAPv2.
#
#  The PEAP module needs the TLS module to be installed
#  and configured, in order to use the TLS tunnel
#  inside of the EAP packet.  You will still need to
#  configure the TLS module, even if you do not want
#  to deploy EAP-TLS in your network.  Users will not
#  be able to request EAP-TLS, as it requires them to
#  have a client certificate.  EAP-PEAP does not
#  require a client certificate.
#
#
#  You can make PEAP require a client cert by setting
#
#       EAP-TLS-Require-Client-Cert = Yes
#
#  in the control items for a request.
#
peap {
        #  The tunneled EAP session needs a default
        #  EAP type which is separate from the one for
        #  the non-tunneled EAP module.  Inside of the
        #  PEAP tunnel, we recommend using MS-CHAPv2,
        #  as that is the default type supported by
```

```
                # Windows clients.
                default_eap_type = mschapv2

                # the PEAP module also has these configuration
                # items, which are the same as for TTLS.
                copy_request_to_tunnel = no
                use_tunneled_reply = no

                # When the tunneled session is proxied, the
                # home server may not understand EAP-MSCHAP-V2.
                # Set this entry to "no" to proxy the tunneled
                # EAP-MSCHAP-V2 as normal MSCHAPv2.
        #       proxy_tunneled_request_as_eap = yes

                #
                # The inner tunneled request can be sent
                # through a virtual server constructed
                # specifically for this purpose.
                #
                # If this entry is commented out, the inner
                # tunneled request will be sent through
                # the virtual server that processed the
                # outer requests.
                #
                virtual_server = "inner-tunnel"
        }

        #
        # This takes no configuration.
        #
        # Note that it is the EAP MS-CHAPv2 sub-module, not
        # the main 'mschap' module.
        #
        # Note also that in order for this sub-module to work,
        # the main 'mschap' module MUST ALSO be configured.
        #
        # This module is the *Microsoft* implementation of MS-CHAPv2
        # in EAP.  There is another (incompatible) implementation
        # of MS-CHAPv2 in EAP by Cisco, which FreeRADIUS does not
        # currently support.
        #
        mschapv2 {
        }
}
```

ANEXO 5. sql.conf

```
# -*- text -*-
##
## sql.conf -- SQL modules
##
##	$Id$

#####################################################################
#
#  Configuration for the SQL module
#
#  The database schemas and queries are located in subdirectories:
#
#		sql/DB/schema.sql	Schema
#		sql/DB/dialup.conf	Basic dialup (including policy) queries
#		sql/DB/counter.conf	counter
#		sql/DB/ippool.conf	IP Pools in SQL
#		sql/DB/ippool.sql	schema for IP pools.
#
#  Where "DB" is mysql, mssql, oracle, or postgresql.
#

sql {
	#
	#  Set the database to one of:
	#
	#		mysql, mssql, oracle, postgresql
	#
	database = "mysql"

	#
	#  Which FreeRADIUS driver to use.
	#
	driver = "rlm_sql_${database}"

# Connection info:
	server = "localhost"
	#port = 3306
	login = "radius"
	password = "labserver"


	# Connection info:
	#server = "localhost"
	#port = 3306
	#login = "radius"
	#password = "radpass"

	# Database table configuration for everything except Oracle
```

```
        radius_db = "radius"
        # If you are using Oracle then use this instead
    # radius_db =
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT_DA
TA=(SID=your_sid)))"

        # If you want both stop and start records logged to the
        # same SQL table, leave this as is.  If you want them in
        # different tables, put the start table in acct_table1
        # and stop table in acct_table2
        acct_table1 = "radacct"
        acct_table2 = "radacct"

        # Allow for storing data after authentication
        postauth_table = "radpostauth"

        authcheck_table = "radcheck"
        authreply_table = "radreply"

        groupcheck_table = "radgroupcheck"
        groupreply_table = "radgroupreply"

        # Table to keep group info
        usergroup_table = "radusergroup"

        # If set to 'yes' (default) we read the group tables
        # If set to 'no' the user MUST have Fall-Through = Yes in the radreply table
        # read_groups = yes

        # Remove stale session if checkrad does not see a double login
        deletestalesessions = yes

        # Print all SQL statements when in debug mode (-x)
        sqltrace = no
        sqltracefile = ${logdir}/sqltrace.sql

        # number of sql connections to make to server
        num_sql_socks = 5

        # number of seconds to dely retrying on a failed database
        # connection (per_socket)
        connect_failure_retry_delay = 60

        # lifetime of an SQL socket.  If you are having network issues
        # such as TCP sessions expiring, you may need to set the socket
        # lifetime.  If set to non-zero, any open connections will be
        # closed "lifetime" seconds after they were first opened.
        lifetime = 0

        # Maximum number of queries used by an SQL socket.  If you are
        # having issues with SQL sockets lasting "too long", you can
        # limit the number of queries performed over one socket.  After
        # "max_qeuries", the socket will be closed.  Use 0 for "no limit".
```

```
        max_queries = 0

        # Set to 'yes' to read radius clients from the database ('nas' table)
        # Clients will ONLY be read on server startup.  For performance
        # and security reasons, finding clients via SQL queries CANNOT
        # be done "live" while the server is running.
        #
        #readclients = yes

        # Table to keep radius client info
        nas_table = "nas"

        # Read driver-specific configuration
        $INCLUDE sql/${database}/dialup.conf
}
```

ANEXO 6. default

```
##########################################################################
#
#	As of 2.0.0, FreeRADIUS supports virtual hosts using the
#	"server" section, and configuration directives.
#
#	Virtual hosts should be put into the "sites-available"
#	directory.  Soft links should be created in the "sites-enabled"
#	directory to these files.  This is done in a normal installation.
#
#	$Id$
#
##########################################################################
#
#	Read "man radiusd" before editing this file.  See the section
#	titled DEBUGGING.  It outlines a method where you can quickly
#	obtain the configuration you want, without running into
#	trouble.  See also "man unlang", which documents the format
#	of this file.
#
#	This configuration is designed to work in the widest possible
#	set of circumstances, with the widest possible number of
#	authentication methods.  This means that in general, you should
#	need to make very few changes to this file.
#
#	The best way to configure the server for your local system
#	is to CAREFULLY edit this file.  Most attempts to make large
#	edits to this file will BREAK THE SERVER.  Any edits should
#	be small, and tested by running the server with "radiusd -X".
#	Once the edits have been verified to work, save a copy of these
#	configuration files somewhere.  (e.g. as a "tar" file).  Then,
#	make more edits, and test, as above.
#
#	There are many "commented out" references to modules such
#	as ldap, sql, etc.  These references serve as place-holders.
#	If you need the functionality of that module, then configure
#	it in radiusd.conf, and un-comment the references to it in
#	this file.  In most cases, those small changes will result
#	in the server being able to connect to the DB, and to
#	authenticate users.
#
##########################################################################

#
#	In 1.x, the "authorize", etc. sections were global in
#	radiusd.conf.  As of 2.0, they SHOULD be in a server section.
#
#	The server section with no virtual server name is the "default"
```

```
#       section.  It is used when no server name is specified.
#
#       We don't indent the rest of this file, because doing so
#       would make it harder to read.
#

# Authorization. First preprocess (hints and huntgroups files),
# then realms, and finally look in the "users" file.
#
# The order of the realm modules will determine the order that
# we try to find a matching realm.
#
# Make *sure* that 'preprocess' comes before any realm if you
# need to setup hints for the remote radius server
authorize {
        #
        # The preprocess module takes care of sanitizing some bizarre
        # attributes in the request, and turning them into attributes
        # which are more standard.
        #
        # It takes care of processing the 'raddb/hints' and the
        # 'raddb/huntgroups' files.
        preprocess

        #
        # If you want to have a log of authentication requests,
        # un-comment the following line, and the 'detail auth_log'
        # section, above.
#       auth_log

        #
        # The chap module will set 'Auth-Type := CHAP' if we are
        # handling a CHAP request and Auth-Type has not already been set
        chap

        #
        # If the users are logging in with an MS-CHAP-Challenge
        # attribute for authentication, the mschap module will find
        # the MS-CHAP-Challenge attribute, and add 'Auth-Type := MS-CHAP'
        # to the request, which will cause the server to then use
        # the mschap module for authentication.
        mschap

        #
        # If you have a Cisco SIP server authenticating against
        # FreeRADIUS, uncomment the following line, and the 'digest'
        # line in the 'authenticate' section.
        digest

        #
        # The WiMAX specification says that the Calling-Station-Id
        # is 6 octets of the MAC.  This definition conflicts with
        # RFC 3580, and all common RADIUS practices.  Un-commenting
```

```
        #  the "wimax" module here means that it will fix the
        #  Calling-Station-Id attribute to the normal format as
        #  specified in RFC 3580 Section 3.21
#       wimax

        #
        #  Look for IPASS style 'realm/', and if not found, look for
        #  '@realm', and decide whether or not to proxy, based on
        #  that.
#       IPASS

        #
        #  If you are using multiple kinds of realms, you probably
        #  want to set "ignore_null = yes" for all of them.
        #  Otherwise, when the first style of realm doesn't match,
        #  the other styles won't be checked.
        #
        suffix
#       ntdomain

        #
        #  This module takes care of EAP-MD5, EAP-TLS, and EAP-LEAP
        #  authentication.
        #
        #  It also sets the EAP-Type attribute in the request
        #  attribute list to the EAP type from the packet.
        #
        #  As of 2.0, the EAP module returns "ok" in the authorize stage
        #  for TTLS and PEAP.  In 1.x, it never returned "ok" here, so
        #  this change is compatible with older configurations.
        #
        #  The example below uses module failover to avoid querying all
        #  of the following modules if the EAP module returns "ok".
        #  Therefore, your LDAP and/or SQL servers will not be queried
        #  for the many packets that go back and forth to set up TTLS
        #  or PEAP.  The load on those servers will therefore be reduced.
        #
        eap {
                ok = return
        }

        #
        #  Pull crypt'd passwords from /etc/passwd or /etc/shadow,
        #  using the system API's to get the password.  If you want
        #  to read /etc/passwd or /etc/shadow directly, see the
        #  passwd module in radiusd.conf.
        #
#       unix

        #
        #  Read the 'users' file
        files
```

```
        #
        #  Look in an SQL database.  The schema of the database
        #  is meant to mirror the "users" file.
        #
        #  See "Authorization Queries" in sql.conf
        sql

        #
        #  If you are using /etc/smbpasswd, and are also doing
        #  mschap authentication, the un-comment this line, and
        #  configure the 'etc_smbpasswd' module, above.
#       etc_smbpasswd

        #
        #  The ldap module will set Auth-Type to LDAP if it has not
        #  already been set
#       ldap

        #
        #  Enforce daily limits on time spent logged in.
#       daily

        #
        # Use the checkval module
#       checkval

        expiration
        logintime

        #
        #  If no other module has claimed responsibility for
        #  authentication, then try to use PAP.  This allows the
        #  other modules listed above to add a "known good" password
        #  to the request, and to do nothing else.  The PAP module
        #  will then see that password, and use it to do PAP
        #  authentication.
        #
        #  This module should be listed last, so that the other modules
        #  get a chance to set Auth-Type for themselves.
        #
        pap

        #
        #  If "status_server = yes", then Status-Server messages are passed
        #  through the following section, and ONLY the following section.
        #  This permits you to do DB queries, for example.  If the modules
        #  listed here return "fail", then NO response is sent.
        #
#       Autz-Type Status-Server {
#
#       }
}
```

```
# Authentication.
#
#
# This section lists which modules are available for authentication.
# Note that it does NOT mean 'try each module in order'. It means
# that a module from the 'authorize' section adds a configuration
# attribute 'Auth-Type := FOO'. That authentication type is then
# used to pick the apropriate module from the list below.
#

# In general, you SHOULD NOT set the Auth-Type attribute. The server
# will figure it out on its own, and will do the right thing. The
# most common side effect of erroneously setting the Auth-Type
# attribute is that one authentication method will work, but the
# others will not.
#
# The common reasons to set the Auth-Type attribute by hand
# is to either forcibly reject the user (Auth-Type := Reject),
# or to or forcibly accept the user (Auth-Type := Accept).
#
# Note that Auth-Type := Accept will NOT work with EAP.
#
# Please do not put "unlang" configurations into the "authenticate"
# section. Put them in the "post-auth" section instead. That's what
# the post-auth section is for.
#
authenticate {
        #
        # PAP authentication, when a back-end database listed
        # in the 'authorize' section supplies a password. The
        # password can be clear-text, or encrypted.
        Auth-Type PAP {
                pap
        }

        #
        # Most people want CHAP authentication
        # A back-end database listed in the 'authorize' section
        # MUST supply a CLEAR TEXT password. Encrypted passwords
        # won't work.
        Auth-Type CHAP {
                chap
        }

        #
        # MSCHAP authentication.
        Auth-Type MS-CHAP {
                mschap
        }

        #
        # If you have a Cisco SIP server authenticating against
```

```
        #  FreeRADIUS, uncomment the following line, and the 'digest'
        #  line in the 'authorize' section.
        digest

        #
        #  Pluggable Authentication Modules.
#       pam

        #
        #  See 'man getpwent' for information on how the 'unix'
        #  module checks the users password.  Note that packets
        #  containing CHAP-Password attributes CANNOT be authenticated
        #  against /etc/passwd!  See the FAQ for details.
        #
        #  For normal "crypt" authentication, the "pap" module should
        #  be used instead of the "unix" module.  The "unix" module should
        #  be used for authentication ONLY for compatibility with legacy
        #  FreeRADIUS configurations.
        #
        unix

        # Uncomment it if you want to use ldap for authentication
        #
        # Note that this means "check plain-text password against
        # the ldap database", which means that EAP won't work,
        # as it does not supply a plain-text password.
#       Auth-Type LDAP {
#               ldap
#       }

        #
        #  Allow EAP authentication.
        eap

        #
        #  The older configurations sent a number of attributes in
        #  Access-Challenge packets, which wasn't strictly correct.
        #  If you want to filter out these attributes, uncomment
        #  the following lines.
        #
#       Auth-Type eap {
#               eap {
#                       handled = 1
#               }
#               if (handled && (Response-Packet-Type == Access-Challenge)) {
#                       attr_filter.access_challenge.post-auth
#                       handled  # override the "updated" code from attr_filter
#               }
#       }
}


#
```

```
# Pre-accounting. Decide which accounting type to use.
#
preacct {
        preprocess

        #
        # Session start times are *implied* in RADIUS.
        # The NAS never sends a "start time". Instead, it sends
        # a start packet, *possibly* with an Acct-Delay-Time.
        # The server is supposed to conclude that the start time
        # was "Acct-Delay-Time" seconds in the past.
        #
        # The code below creates an explicit start time, which can
        # then be used in other modules.
        #
        # The start time is: NOW - delay - session_length
        #

#         update request {
#                 FreeRADIUS-Acct-Session-Start-Time = "%{expr: %l - %{%{Acct-Session-Time}:-0}
- %{%{Acct-Delay-Time}:-0}}"
#         }


        #
        # Ensure that we have a semi-unique identifier for every
        # request, and many NAS boxes are broken.
        acct_unique

        #
        # Look for IPASS-style 'realm/', and if not found, look for
        # '@realm', and decide whether or not to proxy, based on
        # that.
        #
        # Accounting requests are generally proxied to the same
        # home server as authentication requests.
#       IPASS
        suffix
#       ntdomain

        #
        # Read the 'acct_users' file
        files
}

#
# Accounting. Log the accounting data.
#
accounting {
        #
        # Create a 'detail'ed log of the packets.
        # Note that accounting requests which are proxied
        # are also logged in the detail file.
```

```
        detail
#       daily

        #  Update the wtmp file
        #
        #  If you don't use "radlast", you can delete this line.
        unix

        #
        #  For Simultaneous-Use tracking.
        #
        #  Due to packet losses in the network, the data here
        #  may be incorrect.  There is little we can do about it.
        radutmp
#       sradutmp

        #  Return an address to the IP Pool when we see a stop record.
#       main_pool

        #
        #  Log traffic to an SQL database.
        #
        #  See "Accounting queries" in sql.conf
        sql

        #
        #  If you receive stop packets with zero session length,
        #  they will NOT be logged in the database.  The SQL module
        #  will print a message (only in debugging mode), and will
        #  return "noop".
        #
        #  You can ignore these packets by uncommenting the following
        #  three lines.  Otherwise, the server will not respond to the
        #  accounting request, and the NAS will retransmit.
        #
#       if (noop) {
#               ok
#       }

        #
        #  Instead of sending the query to the SQL server,
        #  write it into a log file.
        #
#       sql_log

        #  Cisco VoIP specific bulk accounting
#       pgsql-voip

        # For Exec-Program and Exec-Program-Wait
        exec

        #  Filter attributes from the accounting response.
        attr_filter.accounting_response
```

```
        #
        #  See "Autz-Type Status-Server" for how this works.
        #
#       Acct-Type Status-Server {
#
#       }
}


#  Session database, used for checking Simultaneous-Use. Either the radutmp
#  or rlm_sql module can handle this.
#  The rlm_sql module is *much* faster
session {
        radutmp

        #
        #  See "Simultaneous Use Checking Queries" in sql.conf
        sql
}


#  Post-Authentication
#  Once we KNOW that the user has been authenticated, there are
#  additional steps we can take.
post-auth {
        #  Get an address from the IP Pool.
#       main_pool

        #
        #  If you want to have a log of authentication replies,
        #  un-comment the following line, and the 'detail reply_log'
        #  section, above.
#       reply_log

        #
        #  After authenticating the user, do another SQL query.
        #
        #  See "Authentication Logging Queries" in sql.conf
        sql

        #
        #  Instead of sending the query to the SQL server,
        #  write it into a log file.
        #
#       sql_log

        #
        #  Un-comment the following if you have set
        #  'edir_account_policy_check = yes' in the ldap module sub-section of
        #  the 'modules' section.
        #
#       ldap
```

```
        # For Exec-Program and Exec-Program-Wait
        exec

        #
        #  Calculate the various WiMAX keys.  In order for this to work,
        #  you will need to define the WiMAX NAI, usually via
        #
        #       update request {
        #            WiMAX-MN-NAI = "%{User-Name}"
        #       }
        #
        #  If you want various keys to be calculated, you will need to
        #  update the reply with "template" values.  The module will see
        #  this, and replace the template values with the correct ones
        #  taken from the cryptographic calculations.  e.g.
        #
        #       update reply {
        #               WiMAX-FA-RK-Key = 0x00
        #               WiMAX-MSK = "%{EAP-MSK}"
        #       }
        #
        #  You may want to delete the MS-MPPE-*-Keys from the reply,
        #  as some WiMAX clients behave badly when those attributes
        #  are included.  See "raddb/modules/wimax", configuration
        #  entry "delete_mppe_keys" for more information.
        #
#       wimax

        #  If there is a client certificate (EAP-TLS, sometimes PEAP
        #  and TTLS), then some attributes are filled out after the
        #  certificate verification has been performed.  These fields
        #  MAY be available during the authentication, or they may be
        #  available only in the "post-auth" section.
        #
        #  The first set of attributes contains information about the
        #  issuing certificate which is being used.  The second
        #  contains information about the client certificate (if
        #  available).
#
#       update reply {
#            Reply-Message += "%{TLS-Cert-Serial}"
#            Reply-Message += "%{TLS-Cert-Expiration}"
#            Reply-Message += "%{TLS-Cert-Subject}"
#            Reply-Message += "%{TLS-Cert-Issuer}"
#            Reply-Message += "%{TLS-Cert-Common-Name}"
#
#            Reply-Message += "%{TLS-Client-Cert-Serial}"
#            Reply-Message += "%{TLS-Client-Cert-Expiration}"
#            Reply-Message += "%{TLS-Client-Cert-Subject}"
#            Reply-Message += "%{TLS-Client-Cert-Issuer}"
#            Reply-Message += "%{TLS-Client-Cert-Common-Name}"
#       }
```

```
        #  If the WiMAX module did it's work, you may want to do more
        #  things here, like delete the MS-MPPE-*-Key attributes.
        #
        #       if (updated) {
        #               update reply {
        #                       MS-MPPE-Recv-Key !* 0x00
        #                       MS-MPPE-Send-Key !* 0x00
        #               }
        #       }

        #
        #  Access-Reject packets are sent through the REJECT sub-section of the
        #  post-auth section.
        #
        #  Add the ldap module name (or instance) if you have set
        #  'edir_account_policy_check = yes' in the ldap module configuration
        #
        Post-Auth-Type REJECT {
                # log failed authentications in SQL, too.
                sql
                attr_filter.access_reject
        }
}

#
#  When the server decides to proxy a request to a home server,
#  the proxied request is first passed through the pre-proxy
#  stage.  This stage can re-write the request, or decide to
#  cancel the proxy.
#
#  Only a few modules currently have this method.
#
pre-proxy {
#       attr_rewrite

        #  Uncomment the following line if you want to change attributes
        #  as defined in the preproxy_users file.
#       files

        #  Uncomment the following line if you want to filter requests
        #  sent to remote servers based on the rules defined in the
        #  'attrs.pre-proxy' file.
#       attr_filter.pre-proxy

        #  If you want to have a log of packets proxied to a home
        #  server, un-comment the following line, and the
        #  'detail pre_proxy_log' section, above.
#       pre_proxy_log
}

#
#  When the server receives a reply to a request it proxied
#  to a home server, the request may be massaged here, in the
```

```
# post-proxy stage.
#
post-proxy {

        # If you want to have a log of replies from a home server,
        # un-comment the following line, and the 'detail post_proxy_log'
        # section, above.
#       post_proxy_log

#       attr_rewrite

        # Uncomment the following line if you want to filter replies from
        # remote proxies based on the rules defined in the 'attrs' file.
#       attr_filter.post-proxy

        #
        # If you are proxying LEAP, you MUST configure the EAP
        # module, and you MUST list it here, in the post-proxy
        # stage.
        #
        # You MUST also use the 'nostrip' option in the 'realm'
        # configuration.  Otherwise, the User-Name attribute
        # in the proxied request will not match the user name
        # hidden inside of the EAP packet, and the end server will
        # reject the EAP request.
        #
        eap

        #
        # If the server tries to proxy a request and fails, then the
        # request is processed through the modules in this section.
        #
        # The main use of this section is to permit robust proxying
        # of accounting packets.  The server can be configured to
        # proxy accounting packets as part of normal processing.
        # Then, if the home server goes down, accounting packets can
        # be logged to a local "detail" file, for processing with
        # radrelay.  When the home server comes back up, radrelay
        # will read the detail file, and send the packets to the
        # home server.
        #
        # With this configuration, the server always responds to
        # Accounting-Requests from the NAS, but only writes
        # accounting packets to disk if the home server is down.
        #
#       Post-Proxy-Type Fail {
#                       detail
#       }
}
```

ANEXO 7. CA.sh

```
#!/bin/sh
#
# CA - wrapper around ca to make it easier to use ... basically ca requires
#      some setup stuff to be done before you can use it and this makes
#      things easier between now and when Eric is convinced to fix it :-)
#
# CA -newca ... will setup the right stuff
# CA -newreq ... will generate a certificate request
# CA -sign ... will sign the generated request and output
#
# At the end of that grab newreq.pem and newcert.pem (one has the key
# and the other the certificate) and cat them together and that is what
# you want/need ... I'll make even this a little cleaner later.
#
#
# 12-Jan-96 tjh    Added more things ... including CA -signcert which
#                  converts a certificate to a request and then signs it.
# 10-Jan-96 eay    Fixed a few more bugs and added the SSLEAY_CONFIG
#                  environment variable so this can be driven from
#                  a script.
# 25-Jul-96 eay    Cleaned up filenames some more.
# 11-Jun-96 eay    Fixed a few filename missmatches.
# 03-May-96 eay    Modified to use 'ssleay cmd' instead of 'cmd'.
# 18-Apr-96 tjh    Original hacking
#
# Tim Hudson
# tjh@cryptsoft.com
#

# default openssl.cnf file has setup as per the following
# demoCA ... where everything is stored
cp_pem() {
    infile=$1
    outfile=$2
    bound=$3
    flag=0
    exec <$infile;
    while read line; do
            if [ $flag -eq 1 ]; then
                    echo $line|grep "^-----END.*$bound"  2>/dev/null 1>/dev/null
                    if [ $? -eq 0 ] ; then
                            echo $line >>$outfile
                            break
                    else
                            echo $line >>$outfile
                    fi
            fi
```

```
        echo $line|grep "^-----BEGIN.*$bound"  2>/dev/null 1>/dev/null
        if [ $? -eq 0 ]; then
                echo $line >$outfile
                flag=1
        fi
   done
}

usage() {
 echo "usage: $0 -newcert|-newreq|-newreq-nodes|-newca|-sign|-verify" >&2
}

if [ -z "$OPENSSL" ]; then OPENSSL=openssl; fi

if [ -z "$DAYS" ] ; then DAYS="-days 730" ; fi     # 2 year
CADAYS="-days 1095" # 3 years
REQ="$OPENSSL req $SSLEAY_CONFIG"
CA="$OPENSSL ca $SSLEAY_CONFIG"
VERIFY="$OPENSSL verify"
X509="$OPENSSL x509"
PKCS12="openssl pkcs12"

if [ -z "$CATOP" ] ; then CATOP=./masterCA ; fi
CAKEY=./cakey.pem
CAREQ=./careq.pem
CACERT=./cacert.pem

RET=0

while [ "$1" != "" ] ; do
case $1 in
-\?|-h|-help)
   usage
   exit 0
   ;;
-newcert)
   # create a certificate
   $REQ -new -x509 -keyout newkey.pem -out newcert.pem $DAYS
   RET=$?
   echo "Certificate is in newcert.pem, private key is in newkey.pem"
   ;;
-newreq)
   # create a certificate request
   $REQ -new -keyout newkey.pem -out newreq.pem $DAYS
   RET=$?
   echo "Request is in newreq.pem, private key is in newkey.pem"
   ;;
-newreq-nodes)
   # create a certificate request
   $REQ -new -nodes -keyout newreq.pem -out newreq.pem $DAYS
   RET=$?
   echo "Request (and private key) is in newreq.pem"
   ;;
```

```
-newca)
   # if explicitly asked for or it doesn't exist then setup the directory
   # structure that Eric likes to manage things
   NEW="1"
   if [ "$NEW" -o ! -f ${CATOP}/serial ]; then
         # create the directory hierarchy
         mkdir -p ${CATOP}
         mkdir -p ${CATOP}/certs
         mkdir -p ${CATOP}/crl
         mkdir -p ${CATOP}/newcerts
         mkdir -p ${CATOP}/private
         touch ${CATOP}/index.txt
   fi
   if [ ! -f ${CATOP}/private/$CAKEY ]; then
         echo "CA certificate filename (or enter to create)"
         read FILE

         # ask user for existing CA certificate
         if [ "$FILE" ]; then
            cp_pem $FILE ${CATOP}/private/$CAKEY PRIVATE
            cp_pem $FILE ${CATOP}/$CACERT CERTIFICATE
            RET=$?
            if [ ! -f "${CATOP}/serial" ]; then
                  $X509 -in ${CATOP}/$CACERT -noout -next_serial \
                      -out ${CATOP}/serial
            fi
         else
            echo "Making CA certificate ..."
            $REQ -new -keyout ${CATOP}/private/$CAKEY \
                              -out ${CATOP}/$CAREQ
            $CA -create_serial -out ${CATOP}/$CACERT $CADAYS -batch \
                              -keyfile ${CATOP}/private/$CAKEY -selfsign \
                              -extensions v3_ca \
                              -infiles ${CATOP}/$CAREQ
            RET=$?
         fi
   fi
   ;;
-xsign)
   $CA -policy policy_anything -infiles newreq.pem
   RET=$?
   ;;
-pkcs12)
   if [ -z "$2" ] ; then
         CNAME="My Certificate"
   else
         CNAME="$2"
   fi
   $PKCS12 -in newcert.pem -inkey newreq.pem -certfile ${CATOP}/$CACERT \
            -out newcert.p12 -export -name "$CNAME"
   RET=$?
   exit $RET
   ;;
```

```
-sign|-signreq)
   $CA -policy policy_anything -out newcert.pem -infiles newreq.pem
   RET=$?
   cat newcert.pem
   echo "Signed certificate is in newcert.pem"
   ;;
-signCA)
   $CA -policy policy_anything -out newcert.pem -extensions v3_ca -infiles newreq.pem
   RET=$?
   echo "Signed CA certificate is in newcert.pem"
   ;;
-signcert)
   echo "Cert passphrase will be requested twice - bug?"
   $X509 -x509toreq -in newreq.pem -signkey newreq.pem -out tmp.pem
   $CA -policy policy_anything -out newcert.pem -infiles tmp.pem
   RET=$?
   cat newcert.pem
   echo "Signed certificate is in newcert.pem"
   ;;
-verify)
   shift
   if [ -z "$1" ]; then
            $VERIFY -CAfile $CATOP/$CACERT newcert.pem
            RET=$?
   else
         for j
         do
            $VERIFY -CAfile $CATOP/$CACERT $j
            if [ $? != 0 ]; then
                  RET=$?
            fi
         done
   fi
   exit $RET
   ;;
*)
   echo "Unknown arg $i" >&2
   usage
   exit 1
   ;;
esac
shift
done
exit $RET
```

ANEXO 8. Trabajos derivados de la tesis

Seguridad en Sistemas Móviles Mediante Autentificación de Firma Digital

Ing. Norma Alicia Cruz Guzmán, Dr. Felipe Rolando Menchaca García
Instituto Politécnico Nacional
Escuela Superior de Ingeniería Mecánica y Eléctrica, Campus Zacatenco
U.P.A.L.M. Edif. Z, Acc 4, 3er. Piso, Col. Lindavista, C. P. 07738, México, D. F.
ncruzg0901@ipn.mx, fmenchac@ipn.mx

RESUMÉN

En esencia, este documento describe los elementos principales de seguridad en espacios inteligentes cerrados mediante la autentificación por firma digital. Y de esta manera, poder explotar o emplear a su máxima capacidad los servicios compartidos dentro de la red, resguardando los recursos e información de los diversos riesgos que asedian a esta tecnología siguiendo las recomendaciones pertinentes.

PALABRAS CLAVE

Seguridad, autentificación y firma digital.

INTRODUCCIÓN

La globalización de las comunicaciones inalámbricas ha permitido el desarrollo de nuevos estándares y productos que están produciendo cambios en la vida diaria. La movilidad se ha vuelto un requerimiento cada vez mayor dentro de los ambientes de trabajo y gracias a las redes inalámbricas se ha obtenido una movilidad *real* en los dispositivos *móviles*.

La navegación por Internet a través de los dispositivos inalámbricos, hace que el intercambio de información por este medio, incluyendo datos de alto valor, sea una práctica común para los usuarios de las redes inalámbricas, por lo que actualmente se ha puesto un especial ´énfasis a la seguridad en tales medios de comunicación.

Un sistema posee la propiedad de *confidencialidad* si los recursos manipulados por éste no son puestos al descubierto por usuarios, entidades o procesos no autorizados. Un sistema posee la propiedad de *integridad* si los recursos manipulados por este no son alterados o destruidos por usuarios, entidades o procesos no autorizados. Un sistema posee la propiedad de *disponibilidad* si los recursos brindan servicio en el momento en que así lo deseen los usuarios, entidades o procesos autorizados. La *autenticación* es el proceso de verificar y asegurar la identidad de las partes involucradas en una transacción. Si este servicio no se llevara a cabo cabe la posibilidad de que una entidad es conocida asuma una identidad falsa, comprometiendo de esta manera la privacidad y la integridad de la información.

La seguridad es muy importante para que el desarrollo e implementación de las redes inalámbricas sean explotados de una manera eficaz y confiable; desafortunadamente, las características inherentes de las redes inalámbricas pueden ser un punto en contra de tal seguridad.

ESTADO DEL ARTE

Criptografía de Llave Pública

Hoy en día el intercambio de información valiosa por Internet, como números de tarjetas de crédito, es una práctica común. En general, en todo tipo de redes la información que se comunica

de una entidad a otra está expuesta a la posibilidad de un ataque de un adversario, por lo que el proteger los datos intercambiados se ha vuelto una tarea prioritaria en todo ámbito. Es por ello que la premisa básica de cualquier comunicación y administración de datos confiable es cumplir con los principios de la seguridad computacional, que se resumen en los siguientes servicios [2,1]:

1. Confidencialidad. Los componentes del sistema serán accesibles sólo por aquellos usuarios autorizados.
2. Integridad. Los componentes del sistema sólo pueden ser creados y modificados por los usuarios autorizados.
3. Disponibilidad. Los usuarios deben tener disponibles todos los componentes del sistema cuando así lo deseen.
4. Autentificación. La autentificación es el proceso de verificar y asegurar la identidad de las partes involucradas en una transacción.
5. No Repudio. El no repudio está asociado a la aceptación de un protocolo de comunicación entre emisor y receptor (cliente y servidor) normalmente este proceso se lleva a cabo a través de la autentificación.

Si se cumplen estos 5 servicios se considera que los datos están protegidos y seguros [2]. La criptografía es la responsable directa de brindar sistemas criptográficos a los servicios de seguridad para llevar a cabo su tarea. Como se muestra en la figura 1, las aplicaciones de los algoritmos criptográficos son los servicios de seguridad mencionados.

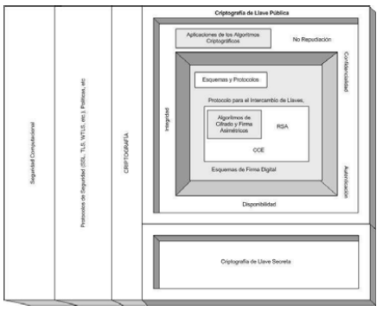

Figura 1. Relación de la Criptografía con la Seguridad Computacional

La criptografía es el proceso de diseñar sistemas basados en técnicas matemáticas para obtener una comunicación segura en canales que no lo son [3].

Esquema de Firma Digital

Una primitiva criptográfica que es fundamental en la autentificación, autorización y no repudio, es la *firma digital*. El propósito de una firma digital es proveer a una entidad un medio para enlazar su identidad a una pieza de información. El proceso de *firma* dentro de los esquemas de llave pública se puede ver como el proceso de cifrado con la llave privada y el proceso de *verificación* se puede ver como el proceso de descifrado con la llave pública. El esquema general de firma digital se muestra en la figura 2.

Como se observa, al mensaje se le aplica una función *hash* (toma un mensaje como entrada de longitud arbitraria, lo digiere y produce una salida conocida como *digestión* de longitud fija) cuyo resultado será firmado con la llave privada del signatario y anexado al mensaje para ser enviados al destinatario. El destinatario separa los dos componentes:

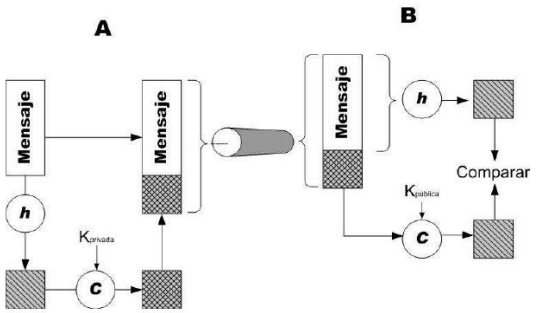

Figura 2. Esquema de firma digital con apéndice

DESRIPCIÓN DE METODOLOGÍA

Autentificación

La principal meta de la criptografía es garantizar que se cumplan los cinco servicios de la seguridad computacional. La autentificación es el proceso mediante el cual se verifica y asegura la identidad de las partes involucradas en una transacción. Si este proceso no se llevara a cabo cabría la posibilidad de que una entidad desconocida asuma una identidad falsa, comprometiendo de esta manera la privacidad y la integridad de la información.

La autentificación es necesaria en los sistemas de llave pública. El ataque conocido como "intruso en medio" muestra como un adversario activo puede burlar el modelo sin romper el criptosistema. Esto resalta la necesidad de autenticar a las llaves públicas para lograr una certificación del origen de datos de las llaves públicas en sí.

La autentificación es cualquier proceso a través de cuál se demuestra y se verifica cierta información referente a un objeto, como el origen de un documento, la identidad del remitente, momento en que un documento fue enviado y/o firmado, la identidad de una computadora o usuario, etc.

Firmas Digitales para Autentificación

Las firmas digitales han sido la herramienta criptográfica que ha resuelto varios aspectos de la autentificación e integridad de datos. A través de estos esquemas ha sido posible sustituir documentos legales en papel por documentos digitales de manera confiable y eficaz.

La firma digital, a diferencia de la autógrafa, está ligada tanto al signatario como al documento o mensaje que está siendo acreditado. Los esquemas de firma digital consisten de dos pasos: el proceso de *firma* y el proceso de *verificación*. Cualquier alteración en el documento firmado digitalmente hará que el proceso de verificación falle y la firma no sea aceptada.

La firma digital, es un proceso criptográfico que permite asegurar la identidad del autor de un documento, y la inalterabilidad del contenido del documento firmado; para que ello sea posible, la firma digital debe ser:

• única
• infalsificable
• fácil de verificar

Para hacer más eficientes los esquemas, los mensajes son reprocesados por una función de digestión mejor conocida como funciones *hash*.

Partiendo de una función *hash h* pública, se toma el mensaje *m* a procesar y se calcula *h*(*m*), la salida de *h*(*m*) es significativamente más pequeña que *m* y por lo tanto firmar el valor *hash* puede hacerse más rápido que procesar el mensaje completo.

El resultado de la operación *hash* se cifra con la llave privada del remitente, *sig*(*h*(*m*)) y se usa como la firma digital del mensaje. Para verificar la firma, se envía el par (*m; sig*(*h*(*m*))), tal como se ilustra en la figura 2. Al mensaje *m* recibido se le realiza el *hash*, h'(*m*), se descifra *sig*(*h*(*m*)) con la llave publica del remitente y el resultado debe concordar con el resultado de la operación *hash*, h'(*m*) = *h*(*m*).

CONCLUSIONES Y TRABAJOS FUTUROS

La movilidad ha originado nuevos desafíos a la seguridad de los sistemas de información tradicional. El intercambio de datos confidenciales es común en las redes inalámbricas donde el canal de comunicación es inherentemente inseguro y puede ser atacado de manera pasiva comprometiendo la confidencialidad de los datos. Se ha destacado que la mejor protección contra el acceso no autorizado es la implementación de mecanismos de autenticación que aseguren el acceso de usuarios identificados a la red.

Brindar servicios de seguridad en especial de autenticación de mala calidad, ya sea por un mal diseño o al hacer elecciones desfavorables en su implementación puede traer consecuencias perjudiciales a los usuarios de las redes inalámbricas.

Este trabajo deja entrever diferentes aspectos que pueden explotarse como trabajo a futuro, entre ellos se puede mencionar:

¿Cuál es la relación de los servicios de seguridad con la calidad de servicio?

REFERENCIAS

[1] Menezes A., van Oorschot P., and Vanstone S. *Handbook of Applied Cryptography*. CRC Press, New York, 5ª edition, 2001.
[2] Gollmann D. *Computer Security*. John Wiley & Sons, Chichester, New York, 1999.
[3] Trappe W. and Washington L. *Introduction to Cryptography with Coding Theory*. Prentice-Hall, Upper Saddle River, New Jersey, 2002.

4° CONGRESO INTERNACIONAL DE INGENIERÍAS MECÁNICA, ELÉCTRICA, ELECTRÓNICA, MECATRÓNICA Y COMPUTACIONAL (CIMEEM2011). Web:cimeem.com

27 al 29 de septiembre de 2011. Querétaro, Estado de Querétaro, México

MEMORIAS TÉCNICAS

(No. ISBN en trámite)

Trabajo arbitrado No. 9. Volumen 4(5), 6 páginas.

MODELO E INFRAESTRUCTURA DE SEGURIDAD BASADO EN IDENTIFICACIÓN Y AUTENTICACIÓN PARA REDES

Norma Alicia Cruz Guzmán[1], Salvador Álvarez Ballesteros[2], Chadwick Carreto Arellano[3]

Resumen

En el presente documento se describe el diseño de un Modelo e Infraestructura de seguridad para redes basado en identificación y autenticación, de acuerdo al modelo propuesto, los usuarios del entorno podrán acceder de forma transparente, en cualquier momento y en cualquier lugar, mediante diferentes dispositivos a la red y servicios disponibles dentro de los diferentes dominios de trabajo que maneje este entorno con la garantía de disponer de los servicios de acuerdo a su perfil.

Palabras clave: Autenticación, firma digital, identificación, seguridad.

Summary

In essence, this paper describes the design of a model and network security infrastructure based on identification and authentication, according to the proposed model, users can access the environment seamlessly, anytime, anywhere, using different devices network and services available within the different domains of work, to handle this environment.

Keyswords: Autentication, Digital Signature, Identification, Security.

Introducción

La identidad en la sociedad humana juega un rol muy importante y por ende, se entiende también la necesidad de contar con medios de control que brinden el acceso a recursos, instituciones, derechos, obligaciones, etc. de forma segura, tanto para el portador de la identificación, como del prestador de servicios o instancia que requiere de tal identificación.

Cuando se busca cubrir esta necesidad, se suele recurrir a la creación de identificaciones con la impresión de micas con ciertos elementos de seguridad como marca de agua, holograma bidimensional, firma digitalizada, entre otros. El inconveniente que existe en ellas, es que no te permiten identificarte en múltiples organizaciones y se tiene que portar todas las identificaciones con las que se cuenta y en caso de robo o extravió de cartera, tener que ir a cada organización para solicitar identificaciones nuevas.

La problemática entonces, no es el simple hecho de la identificación, si no la posibilidad de extender sus capacidades en cantidad de información, portabilidad, seguridad y generalización y hacer uso de los avances tecnológicos actuales, esto bajo una buena propuesta de infraestructura que permita identificar al usuario.

En cuanto a seguridad se refiere, es necesario verificar que el portador de la identificación es quien dice ser y para eso, existen diferentes tecnologías que apoyan el mecanismo de autenticación de usuarios tales

como certificados digitales, claves de acceso/password, tarjetas inteligentes, tokens de seguridad, biométricos entre otros.

Dado lo anterior, se vuelve importante la creación de un modelo e infraestructura que al integrar diversas tecnologías para la identificación y la autenticación antes mencionadas, permita verificar la identidad y la autenticidad de una persona de manera segura.

A continuación, en la sección 2 se exhiben los antecedentes para el desarrollo del modelo planteado; en la 3 se describe el modelo propuesto de seguridad para en el 4 definir la infraestructura que se utilizaría para aplicar el modelo, finalmente, en la sección 5 se muestran las conclusiones de la propuesta y se establecen ideas para trabajos a futuro.

[1] Escuela Superior de Ingeniería Mecánica y Eléctrica, IPN, Unidad Profesional "Adolfo López Mateos", Zacatenco, Del. Gustavo A. Madero, C.P. 07738, México D.F., Tels.: 5729-6000 Exts. 54755 y 54756. Emails: ncruzg0901@ipn.mx
[2] Idem. E-mail: salvarez@ipn.mx
[3] Escuela Superior de Cómputo, IPN, Av. Juan de Dios Bátiz s/n esquina Miguel Othón de Mendizabal. Unidad Profesional " Adolfo López Mateos", Col. Lindavista C.P. 07738, México, D. F. Teléfono 57296000 ext. 46188.

Antecedentes

En el mundo actual, donde carecemos completamente de la necesaria confianza para poder realizar cualquier operación sin preceder de algún de identificación, donde ya no podemos dejar abiertas las puertas de casa o del coche, la identificación de las personas se ha convertido en una importante necesidad en cualquier ámbito. Para poder identificarnos ante otra persona utilizamos los documentos de identidad aceptados (pasaporte, tarjetas, etc.), presentándolos ante quien los solicita. Sin embargo, en muchas situaciones esto ya no es suficiente, ya que no existe una persona física ante nosotros, encargada de comprobar la autenticidad de nuestro documento, sino un sistema (computadora, red, cajero …) que precisa saber quiénes somos sin posibilidad de duda, para ofrecernos un servicio concreto. La cantidad de situaciones donde se produce esto en la vida real es muy elevada y aumenta cada día. Muchas veces no percibimos el gran alcance de estas situaciones, ya que lo hacemos de forma automática, sin plantearnos la importancia de esta seguridad.

Cada día aumenta el número de proveedores de servicios que nos solicita identificación para todo tipo de funciones que, teóricamente, nos facilitan la vida: acceso público o privado a Internet, telefonía móvil, acceso al banco, compras por Internet, servicios y trámites legales, etc. El problema del manejo de tantas credenciales es tan grave que se venden incluso programas para el archivado seguro o cifrado d todos los nombres de usuario, contraseñas, secretos compartidos, etc. Con lo que tuvimos que lidiar día a día.

Desde el punto de vista del usuario que se identifica ante un sistema, también es de vital importancia que el sistema que nos identifica sea el que suponemos que tiene que ser.

Lo que hay detrás de todos y cada uno de estos servicios es una infraestructura de sistemas basados en la autenticación, que gestiona todos nuestros datos de identificación de forma segura y eficiente.

Seguridad Informática.

Es la primera línea de defensa para la mayoría de los sistemas computarizados, permitiendo prevenir el ingreso de personas no autorizadas. Es la base para la mayor parte de los controles de acceso.

Se denomina “Identificación” al momento en que el usuario se da a conocer en el sistema; y “Autenticación” a la verificación que realiza el sistema sobre esta identificación.

Existen cuatro tipos de técnicas que permiten realizar la autenticación de la identidad del usuario, las cuales pueden ser utilizadas individualmente o combinadas, algo que:

1. Solamente el individuo conoce: por ejemplo una clave secreta de acceso o password, una clave criptográfica, un número de identificación personal o PIN, etc.
2. La persona posee: por ejemplo una tarjeta magnética.
3. El individuo es y que lo identifica unívocamente: por ejemplo las huellas digitales o la voz.
4. El individuo es capaz de hacer: por ejemplo los patrones de escritura.

Para cada una de estas técnicas vale lo mencionado en el caso de la seguridad física en cuanto a sus ventajas y desventajas. Se destaca que en los dos primeros casos enunciados, es frecuente que las claves sean olvidadas o que las tarjetas o dispositivos se pierdan, mientras que por otro lado, los controles de autenticación biométricos serían los más apropiados y fáciles de administrar, resultando ser también, los más costosos por lo dificultosos de su implementación eficiente.

Desde el punto de vista de la eficiencia, es conveniente que los usuarios sean identificados y autenticados solamente una vez, pudiendo acceder a partir de allí, a todas las aplicaciones y datos a los que su perfil les permita, tanto en sistemas locales como en sistemas a los que deba acceder en forma remota. Esto se denomina "single login" o sincronización de passwords.

Una de las técnicas para implementar esta única identificación de usuarios sería la utilización de un servidor de autenticaciones sobre el cual los usuarios se identifican, y que se encarga luego de autenticar al usuario sobre los restantes equipos a los que éste pueda acceder. Este servidor de autenticaciones no debe ser necesariamente un equipo independiente y puede tener sus funciones distribuidas tanto geográfica como lógicamente, de acuerdo con los requerimientos de carga de tareas.

La Seguridad Informática se basa, en gran medida, en la efectiva administración de los permisos de acceso a los recursos informáticos, basados en la identificación, autenticación y autorización de accesos.

Riesgos Seguridad

Intrusión. Alguien entra ilegalmente a un sistema y es capaz de utilizarlo y modificarlo como si fuera un usuario legítimo.

Rechazo de Servicios. Alguien logra que no puedan prestarse los servicios a los usuarios legítimos, puede ser dañando al sistema o sobrecargando el sistema o la red.

Robo de Información. Alguien tiene acceso a información confidencial, secreta, reservada o restringida. Es común en espionaje industrial, piratería, etc.

Firma digital

Una firma digital es un conjunto de datos asociados a un mensaje que permite asegurar la identidad del firmante y la integridad del mensaje. La firma digital no implica que el mensaje esté encriptado, es decir, que este no pueda ser leído por otras personas; al igual que cuando se firma un documento holográficamente este sí puede ser visualizado por otras personas.

El procedimiento utilizado para firmar digitalmente un mensaje es el siguiente: el firmante genera mediante una función matemática una huella digital del mensaje. Esta huella digital se encripta con la clave privada del firmante, y el resultado es lo que se denomina firma digital la cual se enviará adjunta al mensaje original. De esta manera el firmante va a estar adjuntando al documento una marca que es única para ese documento y que sólo él es capaz de producir.

El receptor del mensaje podrá comprobar que el mensaje no fue modificado desde su creación y que el firmante es quién dice serlo a través del siguiente procedimiento: en primer término generará la huella digital
del mensaje recibido, luego desencriptará la firma digital del mensaje utilizando la clave pública del firmante y obtendrá de esa forma la huella digital del mensaje original; si ambas huellas digitales coinciden, significa que el mensaje no fue alterado y que el firmante es quien dice serlo [1].

Radius

RADIUS (acrónimo en inglés de Remote Authentication Dial-In User Server). Es un protocolo de autenticación y autorización para aplicaciones de acceso a la red o movilidad IP. Utiliza el puerto 1812 UDP para establecer sus conexiones.

Cuando se realiza la conexión con un ISP mediante módem, DSL, cablemódem, Ethernet o Wi-Fi, se envía una información que generalmente es un nombre de usuario y una contraseña. Esta información se transfiere a un dispositivo Network Access Server (NAS) sobre el protocolo PPP, quien redirige la petición a un servidor RADIUS sobre el protocolo RADIUS. El servidor RADIUS comprueba que la información es correcta utilizando esquemas de autenticación como PAP, CHAP o EAP. Si es aceptado, el servidor autorizará el acceso al sistema del ISP y le asigna los recursos de red como una dirección IP, y otros parámetros como L2TP, etc.

Una de las características más importantes del protocolo RADIUS es su capacidad de manejar sesiones, notificando cuando comienza y termina una conexión, así que al usuario se le podrá determinar su consumo y facturar en consecuencia; los datos se pueden utilizar con propósitos estadísticos.

Autoridad de certificación.

Una autoridad de certificación es un sistema informático dedicado a la emisión y gestión posterior de certificados digitales, incluyendo la renovación, expiración, suspensión, la habilitación y la revocación de certificados, a petición de la autoridad de registro. La emisión de certificados se hace de una forma automatizada y no sin la previa confirmación de la autoridad local de registro. La función básica de las autoridades de certificación: es verificar la identidad de los solicitantes de certificados [2].

Certificados Digitales.

Son documentos electrónicos que asocian una clave pública con la identidad de su propietario. Tienen como objetivo principal permitir verificar la identidad del usuario, garantizando que únicamente él puede acceder a su información personal, evitando suplantaciones. También es el elemento usado para firmar electrónicamente solicitudes o documentos, ayudando a prevenir que alguien utilice una clave para hacerse pasar por otra persona.

Adicionalmente, además de la clave pública y la identidad de su propietario, un certificado digital puede contener otros atributos para, por ejemplo, concretar el ámbito de utilización de la clave pública, las fechas de inicio y fin de la validez del certificado, etc. El usuario que haga uso del certificado podrá, gracias a los distintos atributos que posee, conocer más detalles sobre las características del mismo y es válido únicamente dentro del período de vigencia, que comienza en la fecha de inicio indicada en el certificado y finaliza en su fecha de vencimiento, o con su revocación si fuere revocado [3].

Modelo propuesto.

Una vez definidas todas los antecedentes sobre los esquemas de identificación y autenticación en la Fig. 1 se muestra el modelo propuesto que ofrece servicios y/o información a la comunidad del entorno. Cuando se habla de Seguridad de la información en redes de datos, el objetivo es poder garantizar tres conceptos: confidencialidad, autenticación e integridad.

Fig. 1. Modelo Propuesto.

- *Confidencialidad:* Es una propiedad de la información mediante la cual se garantizará el acceso a la misma solo por parte de las personas que estén autorizadas.
- *Autenticación*: Verificación de la identidad de un usuario para así acceder a determinados recursos o poder realizar determinadas tareas.
- *Integridad:* Hace referencia a que todas las características de la información deben ser correctos para que los datos estén correctos.

La propuesta de solución que se presenta conlleva la creación de una Infraestructura de Identificación y Autenticación de Usuarios. En la Fig. 2 se presenta un diagrama a bloques del modelo propuesto, este tendrá un carácter estándar para un manejo en diferentes redes.

Módulo de Identificación de dispositivo:

Permite la conexión entre el dispositivo móvil del usuario y el dominio de red mediante un dispositivo de interconexión que será el que permita el acceso a la red y por ende a los servicios que se ofrecen. La Fig. 2 muestra el proceso que se realiza en este módulo para la identificación del dispositivo.

El punto de acceso es el dispositivo de interconexión que existe entre el dominio de red y los dispositivos móviles de los usuarios. Cuando la conexión entre el dispositivo móvil y el dominio de red se da por primera vez tenemos que el punto de acceso de la red transmite periódicamente su identificador de conjunto de servicio (SSID), El equipo móvil de usuario escucha el SSID del punto de acceso y lo retransmite para lograr una asociación, haciendo una petición por parte del dispositivo móvil al servidor DHCP para que le proporcione la información necesaria para conectarse a la red.

El servidor DHCP envía al dispositivo móvil la configuración de red, que incluye una dirección IP, la dirección del Gateway y las direcciones de los servidores DNS's. Al mismo tiempo el Servidor DHCP actualiza su tabla de usuarios la cual contiene la asociación de la dirección IP con la dirección MAC de cada uno de los dispositivos móviles de los usuarios que están conectados.

Fig. 2..Proceso de Identificación y Autenticación Modelo e infraestructura

…

Esta tabla de usuarios del servidor DHCP se le envía a un Servidor de Usuarios para que verifique contra una Base de Datos de Usuarios Registrados, si el dispositivo móvil ya está registrado en la red. En caso de ser afirmativa la búsqueda se le envía al usuario una petición de identificación a través de una interfaz HTML en su dispositivo móvil. De lo contrario se rechaza la petición de acceso.

Identificación de usuario: La seguridad es un aspecto fundamental en el desarrollo del modelo. Pero, el que sea uno de los principales objetivos del modelo proporcionar información y/o servicios a todos los usuarios no significa que cualquier usuario pueda tener acceso a cualquier información. El acceso a los servicios debe ser controlado individualmente en función de la identidad del usuario.

Mediante la interfaz HTML se le hace la solicitud de introducir nombre de usuario, esto permitirá saber si está registrado. Estos datos se envían al servidor de usuarios para que verifique contra una Base de Datos de Usuarios Registrados. En caso de ser afirmativa pasa a la etapa de autenticación. De lo contrario, se niega el acceso. El proceso del módulo de identificación de usuario se muestra en la Fig. 3.

Fig. 3. Módulo identificación de dispositivo

Autenticación:

Si bien el estándar 802.1x de IEEE habla de los servidores de autenticación en términos genéricos, en la realidad los elementos se diseñan en base a: Autenticación, Autorización y Auditoria (AAA).
Autenticación. La autenticación es un modo de asegurar que los usuarios son quién dicen ser y que el usuario que intenta el acceso a los servicios es de hecho el usuario que tiene la autorización para ello.
Autorización. El proceso por el cual la red de datos autoriza al usuario identificado a acceder a determinados servicios.

Auditoría.

Mediante la cual la red registran todos y cada uno de los accesos a los servicios que realiza el usuario, autorizados o no.

Infraestructura de Seguridad

Los métodos de autenticación están en función de lo que utilizan para la verificación, para nuestro caso se utilizará la Firma digital. Se implementará un Servidor de Autenticación para verificar la autenticidad del usuario utilizando los datos capturados en el formulario de la página HTML que se le presenta al usuario para que introduzca su nombre de usuario y firma digital. Fig. 4. En el proceso de autenticación y autorización de un usuario para acceder a la red, tienen lugar varios pasos:

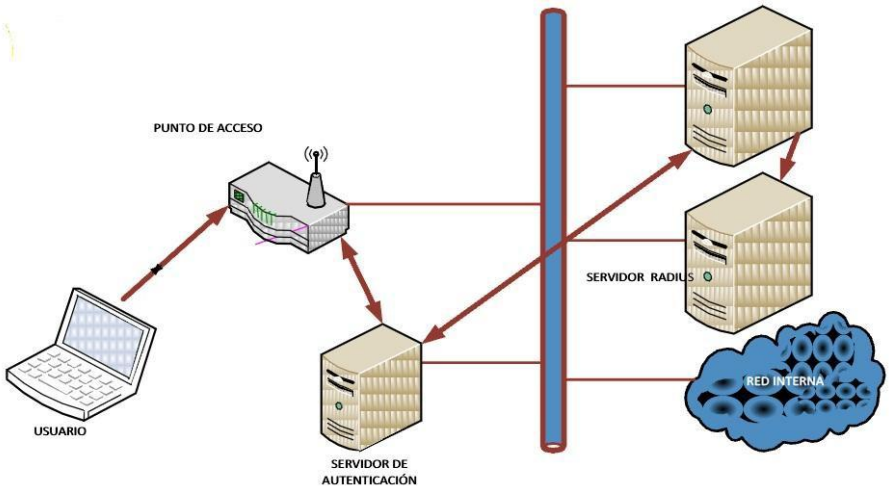

Fig. 4. Conexión del módulo de autenticación

1. Previo al acceso, el usuario tiene que tener generado un certificado digital por la Autoridad de Certificación (CA).
2. El cliente wireless solicita al punto de acceso permiso para establecer una conexión y acceder a la red. Para ello le envía una solicitud firmada con su clave privada. El punto de acceso está configurado para reenviar la solicitud al servicio RADIUS.
3. El servicio RADIUS, consulta al directorio de autenticación para comprobar las credenciales del usuario y su validez. Además también consulta al Directorio de autenticación las políticas de acceso (horarios de conexión, requisitos de cifrado y autenticación, etc.).
4. El servicio RADIUS determina si el usuario tiene acceso a la red y envía la autorización al punto de acceso.
5. El punto de acceso inicia un intercambio de claves para establecer un cifrado de sesión con el cliente, permitiéndole así acceder a la red de forma segura.

Sello de tiempo: El sello de tiempo de un documento electrónico permite garantizar que la información contenida en el mismo no se ha modificado desde el momento de tiempo en el que se generó el sello. Se solicita a una Autoridad de Sellado de Tiempo (TSA) mediante un resumen (hash) de la información a sellar. Este módulo genera un sello de tiempo con el resumen, la fecha y la hora. La TSA proporciona el sello al solicitante, que lo puede adjuntar a la información para garantizar su integridad en el tiempo. Asimismo, la TSA almacena los sellos emitidos para futuras verificaciones.

Acceso al servicio: Este módulo lleva a cabo la administración de servicios con los que cuenta cada usuario. Se encarga de presentar al usuario los servicios de forma organizada y disponible para cuando así los requiriera. Este módulo presenta dependencia en relación con el módulo de identificación ya que para poder establecer la administración de los servicios con los que cuenta el usuario requiere previamente su autenticación. La administración de los servicios se dará de acuerdo a perfiles específicos que serán creados a partir de la base de datos de usuarios registrados en el sistema y de la base de datos de los servicios disponibles, así el usuario tendrá acceso a su lista de servicios disponibles.

El servidor de autenticación será el encargado de realizar la consulta de cada perfil enviando una interfaz HTML al usuario a través de su dispositivo móvil de la relación de servicios a los cuales tiene acceso.

Conclusiones y trabajo a futuro

La movilidad ha originado nuevos desafíos a la seguridad de los sistemas de información tradicional. El intercambio de datos confidenciales es común en las redes inalámbricas donde el canal de comunicación es inherentemente inseguro y puede ser atacado de manera pasiva comprometiendo la confidencialidad de los datos. Otro de los puntos débiles de la seguridad en las redes inalámbricas es el acceso no autorizado a sus recursos donde un intruso puede violar la confidencialidad y la integridad del tráfico de la red al enviar, recibir, alterar o falsificar mensajes. Este último ataque, que se considera activo, puede llevarse a cabo utilizando un dispositivo inalámbrico comprometido con acceso libre a la red. Se ha destacado que la mejor protección contra el acceso no autorizado es la implementación de mecanismos de autenticación que aseguren el acceso de usuarios identificados a la red.

Actualmente el modelo se encuentra en fase de desarrollo; sin embargo, se considera que es una propuesta factible debido a que proporcionará comodidad a los usuarios. Este modelo tiene como meta: brindar seguridad y eficiencia, esto se comprobara con pruebas exhaustivas y de diferentes tipos para definir el nivel de seguridad que se brinda. En lo que respecta a trabajo a futuro, se pretende realizar la implantación del modelo de seguridad en un ambiente educativo real.

Agradecimientos:

Los autores agradecen al Instituto Politécnico Nacional, en particular a la ESIME, ESCOM, CIC, SIP, COFAA, el apoyo para la realización de este trabajo.

Referencias

[1] Yago Fernández Hansen, Antonio Ramos Varón, AAA / RADIUS/ 802.1X. Alfaomega. Extraido el 8 de septiembre [2] Certificado de la ACC. Extraído el 29 de abril de 2011 de http://www.certificadoscopitec.org.ar/info_conocer.htm [3] Cifrado digital-Certificados digitales. Extraído el 25 de mayo de 2011 de http://www.cert.fnmt.es/popup.php?o=faq

4to Congreso Internacional de Ingenierías
Mecánica, Eléctrica, Electrónica, Mecatrónica
y Sistemas Computacionales

MODELO E INFRAESTRUCTURA DE SEGURIDAD BASADO EN IDENTIFICACIÓN Y AUTENTICACIÓN PARA REDES

Norma Alicia Cruz Guzmán[1], Salvador Álvarez Ballesteros[2], Chadwick Carreto Arellano[3]

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA, IPN
Unidad Profesional "Adolfo López Mateos", Zacatenco, Del. Gustavo A. Madero, C.P. 07738, México D.F.
Tels.: 57296000 ext. 54755 Y 54756 E-mail: ncruzg0901@ipn.mx

Área temática: Comunicaciones

INTRODUCCIÓN

La identidad en la sociedad humana juega un rol muy importante y por ende, se entiende también la necesidad de contar con medios de control que brinden el acceso a recursos, instituciones, derechos, obligaciones, etc. de forma segura, tanto para el portador de la identificación, como del prestador de servicios o instancia que requiere de tal identificación.

Cuando se busca cubrir esta necesidad, se suele recurrir a la creación de identificaciones con la impresión de micas con ciertos elementos de seguridad como marca de agua, holograma bidimensional, firma digitalizada, entre otros. El inconveniente que existe en ellas, es que no te permiten identificarte en múltiples organizaciones y se tiene que portar todas las identificaciones con las que se cuenta y en caso de robo o extravió de cartera, tener que ir a cada organización para solicitar identificaciones nuevas.

La problemática entonces, no es el simple hecho de la identificación, si no la posibilidad de extender sus capacidades en cantidad de información, portabilidad, seguridad y generalización y hacer uso de los avances tecnológicos actuales, esto bajo una buena propuesta de infraestructura que permita identificar al usuario.

En cuanto a seguridad se refiere, es necesario verificar que el portador de la identificación es quien dice ser y para eso, existen diferentes tecnologías que apoyan el mecanismo de autenticación de usuarios tales como certificados digitales, claves de acceso/password, tarjetas inteligentes, tokens de seguridad, biométricos entre otros.

Dado lo anterior, se vuelve importante la creación de un módelo e infraestructura que al integrar diversas tecnologías para la identificación y la autenticación antes mencionadas, permita verificar la identidad y la autenticidad de una persona de manera segura.

MODELO PROPUESTO

En la Fig. 1 se muestra el modelo propuesto que ofrece servicios y/o información a la comunidad del entorno. Cuando se habla de Seguridad de la información en redes de datos, el objetivo es poder garantizar tres conceptos: confidencialidad, autenticación e integridad.

- Confidencialidad: Es una propiedad de la información mediante la cual se garantizará el acceso a la misma solo por parte de las personas que estén autorizadas.
- Autenticación: Verificación de la identidad de un usuario para así acceder a determinados recursos o poder realizar determinadas tareas.
- Integridad: Hace referencia a que todas las características de la información deben ser correctos para que los datos estén correctos.

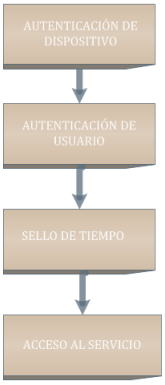

Fig. 1. Modelo Propuesto.

Módulo de Identificación de dispositivo: permite la conexión entre el dispositivo móvil del usuario y el dominio de red mediante un dispositivo de interconexión que será el que permita el acceso a la red y por ende a los servicios que se ofrecen.

INFRAESTRUCTURA DE SEGURIDAD

Los métodos de autenticación están en función de lo que utilizan para la verificación, para nuestro caso se utilizará la Firma digital. Se implementará un Servidor de Autenticación para verificar la autenticidad del usuario utilizando los datos capturados en el formulario de la página HTML que se le presenta al usuario para que introduzca su nombre de usuario y firma digital. Fig. 2.

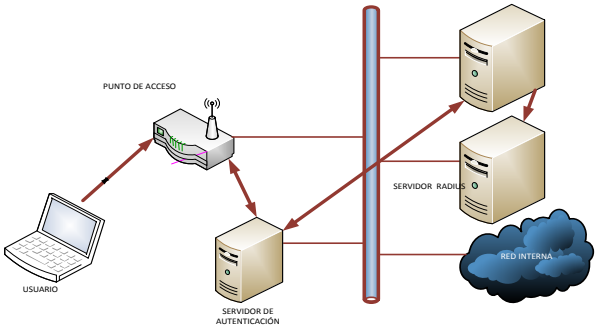

Fig. 2. Conexión del módulo de autenticación

En el proceso de autenticación y autorización de un usuario para acceder a la red, tienen lugar varios pasos:

1. Previo al acceso, el usuario tiene que tener generado un certificado digital por la Autoridad de Certificación (CA).
2. El cliente wireless solicita al punto de acceso permiso para establecer una conexión y acceder a la red. Para ello le envía una solicitud firmada con su clave privada. El punto de acceso está configurado para reenviar la solicitud al servicio RADIUS.
3. El servicio RADIUS, consulta al directorio de autenticación para comprobar las credenciales del usuario y su validez. Además también consulta al Directorio de autenticación las políticas de acceso (horarios de conexión, requisitos de cifrado y autenticación, etc.).
4. El servicio RADIUS determina si el usuario tiene acceso a la red y envía la autorización al punto de acceso.
5. El punto de acceso inicia un intercambio de claves para establecer un cifrado de sesión con el cliente, permitiéndole así acceder a la red de forma segura.

Sello de tiempo: El sello de tiempo de un documento electrónico permite garantizar que la información contenida en el mismo no se ha modificado desde el momento de tiempo en el que se generó el sello.

Acceso al servicio: Este módulo lleva a cabo la administración de servicios con los que cuenta cada usuario. Se encarga de presentar al usuario los servicios de forma organizada y disponible para cuando así los requiriera.

CONCLUSIONE Y TRABAJO FUTURO

Actualmente el modelo se encuentra en fase de desarrollo; sin embargo, se considera que es una propuesta factible debido a que proporcionará comodidad a los usuarios.

Este modelo tiene como meta: brindar seguridad y eficiencia, esto se comprobara con pruebas exhaustivas y de diferentes tipos para definir el nivel de seguridad que se brinda.

En lo que respecta a trabajo a futuro, se pretende realizar la implantación del modelo de seguridad en un ambiente educativo real.

REFERENCIAS

1. Certificado de la ACC, Extraído el 29 de abril de 2011 desde http://www.certificadoscopitec.org.ar/info_conocer.htm

2. Cifrado digital-Certificados digitales, Extraído el 25 de mayo de 2011 desde http://www.cert.fnmt.es/popup.php?o=faq

PROTOTIPO DE UNA ARQUITECTURA DE SEGURIDAD EN IDENTIFICACIÓN BASADO EN IDENTIFICACIÓN Y AUTENTICACIÓN PARA REDES

Autores:

Ing. Norma Alicia Cruz Guzmán

M. en C. Chadwick Carreto Arellano

Dr. Salvador Álvarez Ballesteros

Resumen:

En el presente documento se describe el Prototipo de un Modelo e Infraestructura de seguridad para redes basado en identificación y autenticación, de acuerdo al modelo propuesto, los usuarios del entorno podrán acceder de forma transparente, en cualquier momento y en cualquier lugar, mediante diferentes dispositivos a la red y servicios disponibles dentro de los diferentes dominios de trabajo que maneje este entorno con la garantía de disponer de los servicios de acuerdo a su perfil.

La identidad en la sociedad humana juega un rol muy importante y por ende, se entiende también la necesidad de contar con medios de control que brinden el acceso a recursos, instituciones, derechos, obligaciones, etc. de forma segura, tanto para el portador de la identificación, como del prestador de servicios o instancia que requiere de tal identificación.

Cuando se habla de *Seguridad* de la información en cualquier tipo de red de datos, el objetivo es poder garantizar estos tres conceptos: confidencialidad, autenticación e integridad. Es por ello que nuestra propuesta de solución al problema de seguridad, nos lleva a la creación de un Prototipo de Identificación y Autenticación de Usuarios.

El modelo se presenta en la Figura 1, pretende tener un carácter estándar para el manejo de diferentes tipos de red, diferente composición tecnológica de hardware y topología lógica.

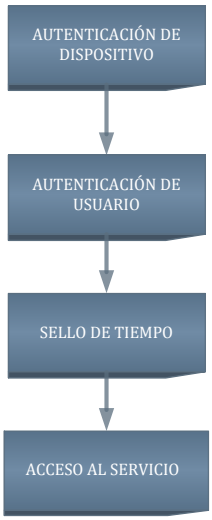

Fig. 1. Modelo Propuesto.

Extracto curricular

Norma Alicia Cruz Guzmán es Ingeniero en Comunicaciones y Electrónica, egresado de la Escuela Superior de Ingeniería Mecánica y Eléctrica, ESIME Unidad Zacatenco del IPN (2003). Actualmente estudia la Maestría en Ciencias de Ingeniería de Telecomunicaciones en la Sección de Estudios de Posgrado e Investigación de la ESIME del IPN.

DISEÑO DE UN MODELO E INFRAESTRUCTURA DE SEGURIDADBASADO EN IDENTIFICACIÓN Y AUTENTICACIÓN PARA REDES

Ing. Norma Alicia Cruz Guzmán[1], M. en C. Chadwick Carreto Arellano[2], Dr. Salvador Álvarez Ballesteros[3]

[1,3] Escuela Superior de Ingeniería Mecánica y Eléctrica - Sección de Estudios de Posgrado e Investigación, Instituto Politécnico Nacional Unidad Profesional Adolfo López Mateos Av. Instituto Politécnico s/n, Edificio Z-4, 3er. Piso. Col. Lindavista C. P. 07738, México, D. F. Tel: +52 55 5729 6000 Ext. 54755,54756. [2] Escuela Superior de Cómputo, Instituto Politécnico Nacional Unidad Profesional Adolfo López Mateos. Av. Juan de Dios Batíz, esquina con Miguel Otón de Mendizábal, Col. Lindavista C. P. 07738, México, D. F. Tel: +52 55 5729 6000

E-mail: ncruzg0901@ipn.mx, salvarez@ipn.mx, ccarretoa@ipn.mx

Resumen

En el presente documento se describe el Desarrollo de una Arquitectura de Seguridad para Redes Basado en Identificación y Autenticación, de acuerdo al modelo propuesto, los usuarios del entorno podrán acceder de forma transparente, en cualquier momento y en cualquier lugar, mediante diferentes dispositivos a la red y servicios disponibles dentro de los diferentes dominios de trabajo que maneje este entorno con la garantía de disponer de los servicios de acuerdo a su perfil.

Palabras clave: Autenticación, firma digital, identificación, seguridad.

Summary

In essence, this paper describes the design of a model and network security infrastructure based on identification and authentication, according to the proposed model, users can access the environment seamlessly, anytime, anywhere, using different devices network and services available within the different domains of work, to handle this environment.

Keys words: Autentication, digital signature, identification, security.

1. Introducción

Con la llegada de nuevas tecnologías, su utilización y adopción por gran cantidad de personas, surgen amenazas y es por ello que hoy en día es muy importante hablar de seguridad en cuanto a computación y redes se refiere. La seguridad de la información y de los sistemas, es un punto crítico en una sociedad en la que la información electrónica o digital cada vez obtiene más simpatizantes.

La identidad en la sociedad humana juega un rol muy importante y por ende, se entiende también la necesidad de contar con medios de control que brinden el acceso a recursos, instituciones, derechos, obligaciones, etc. de forma segura, tanto para el portador de la identificación, como del prestador de servicios o instancia que requiere de tal identificación.

Cuando se busca cubrir esta necesidad, se suele recurrir a la creación de identificaciones con la impresión de micas con ciertos elementos de seguridad como marca de agua, holograma bidimensional, firma digitalizada, entre otros. El inconveniente que existe en ellas, es que no te permiten identificarte en múltiples organizaciones y se tiene

que portar todas las identificaciones con las que se cuenta y en caso de robo o extravió de cartera, tener que ir a cada organización para solicitar identificaciones nuevas.

La problemática entonces, no es el simple hecho de la identificación, si no la posibilidad de extender sus capacidades en cantidad de información, portabilidad, seguridad y generalización y hacer uso de los avances tecnológicos actuales, esto bajo una buena propuesta de infraestructura que permita identificar al usuario.

En cuanto a seguridad se refiere, es necesario verificar que el portador de la identificación es quien dice ser y para eso, existen diferentes tecnologías que apoyan el mecanismo de autenticación de usuarios tales como certificados digitales, claves de acceso/password, tarjetas inteligentes, tokens de seguridad, biométricos entre otros.

Dado lo anterior, se vuelve importante la creación de un modelo e infraestructura que al integrar diversas tecnologías para la identificación y la autenticación antes mencionadas, permita verificar la identidad y la autenticidad de una persona de manera segura.

A continuación, en la sección 2 se exhiben los antecedentes para el desarrollo del modelo planteado; en la 3 se describe el modelo propuesto de seguridad para en el 4 definir el protocolo de seguridad, en la 5 el desarrollo, finalmente, en la sección 6 se muestran las conclusiones de la propuesta y se establecen ideas para trabajos a futuro.

2. Antecedentes

En el mundo actual, donde carecemos completamente de la necesaria confianza para poder realizar cualquier operación sin preceder de algún de identificación, donde ya no podemos dejar abiertas las puertas de casa o del coche, la identificación de las personas se ha convertido en una importante necesidad en cualquier ámbito. Para poder identificarnos ante otra persona utilizamos los documentos de identidad aceptados (pasaporte, tarjetas, etc.), presentándolos ante quien los solicita. Sin embargo, en muchas situaciones esto ya no es suficiente, ya que no existe una persona física ante nosotros, encargada de comprobar la autenticidad de nuestro documento, sino un sistema (computadora, red, cajero …) que precisa saber quiénes somos sin posibilidad de duda, para ofrecernos un servicio concreto. La cantidad de situaciones donde se produce esto en la vida real es muy elevada y aumenta cada día. Muchas veces no percibimos el gran alcance de estas situaciones, ya que lo hacemos de forma automática, sin plantearnos la importancia de esta seguridad.

Cada día aumenta el número de proveedores de servicios que nos solicita identificación para todo tipo de funciones que, teóricamente, nos facilitan la vida: acceso público o privado a Internet, telefonía móvil, acceso al banco, compras por Internet, servicios y trámites legales, etc. El problema del manejo de tantas credenciales es tan grave que se venden incluso programas para el archivado seguro o cifrado d todos los nombres de usuario, contraseñas, secretos compartidos, etc. Con lo que tuvimos que lidiar día a día.

Desde el punto de vista del usuario que se identifica ante un sistema, también es de vital importancia que el sistema que nos identifica sea el que suponemos que tiene que ser.

Lo que hay detrás de todos y cada uno de estos servicios es una infraestructura de sistemas basados en la autenticación, que gestiona todos nuestros datos de identificación de forma segura y eficiente.

Seguridad Informática.

Es la primera línea de defensa para la mayoría de los sistemas computarizados, permitiendo prevenir el ingreso de personas no autorizadas. Es la base para la mayor parte de los controles de acceso.

Se denomina “Identificación” al momento en que el usuario se da a conocer en el sistema; y “Autenticación” a la verificación que realiza el sistema sobre esta identificación.

Existen cuatro tipos de técnicas que permiten realizar la autenticación de la identidad del usuario, las cuales pueden ser utilizadas individualmente o combinadas:

•Algo que solamente el individuo conoce: por ejemplo una clave secreta de acceso o password, una clave criptográfica, un número de identificación personal o PIN, etc.

•Algo que la persona posee: por ejemplo una tarjeta magnética.

•Algo que el individuo es y que lo identifica unívocamente: por ejemplo las huellas digitales o la voz.

•Algo que el individuo es capaz de hacer: por ejemplo los patrones de escritura.

Para cada una de estas técnicas vale lo mencionado en el caso de la seguridad física en cuanto a sus ventajas y desventajas. Se destaca que en los dos primeros casos enunciados, es frecuente que las claves sean olvidadas o que las tarjetas o dispositivos se pierdan, mientras que por otro lado, los controles de autenticación biométricos serían los

más apropiados y fáciles de administrar, resultando ser también, los más costosos por lo dificultosos de su implementación eficiente.

Desde el punto de vista de la eficiencia, es conveniente que los usuarios sean identificados y autenticados solamente una vez, pudiendo acceder a partir de allí, a todas las aplicaciones y datos a los que su perfil les permita, tanto en sistemas locales como en sistemas a los que deba acceder en forma remota. Esto se denomina "single login" o sincronización de passwords.

Una de las técnicas para implementar esta única identificación de usuarios sería la utilización de un servidor de autenticaciones sobre el cual los usuarios se identifican, y que se encarga luego de autenticar al usuario sobre los restantes equipos a los que éste pueda acceder. Este servidor de autenticaciones no debe ser necesariamente un equipo independiente y puede tener sus funciones distribuidas tanto geográfica como lógicamente, de acuerdo con los requerimientos de carga de tareas.

La Seguridad Informática se basa, en gran medida, en la efectiva administración de los permisos de acceso a los recursos informáticos, basados en la identificación, autenticación y autorización de accesos.

Riesgos Seguridad

Intrusión.- alguien entra ilegalmente a un sistema y es capaz de utilizarlo y modificarlo como si fuera un usuario legítimo.

Rechazo de Servicios.- alguien logra que no puedan prestarse los servicios a los usuarios legítimos, puede ser dañando al sistema o sobrecargando el sistema o la red.

Robo de Información.- alguien tiene acceso a información confidencial, secreta, reservada o restringida. Es común en espionaje industrial, piratería, etc.

Firma digital

Una firma digital es un conjunto de datos asociados a un mensaje que permite asegurar la identidad del firmante y la integridad del mensaje. La firma digital no implica que el mensaje esté encriptado, es decir, que este no pueda ser leído por otras personas; al igual que cuando se firma un documento holográficamente este sí puede ser visualizado por otras personas.

El procedimiento utilizado para firmar digitalmente un mensaje es el siguiente: el firmante genera mediante una función matemática una huella digital del mensaje. Esta huella digital se encripta con la clave privada del firmante, y el resultado es lo que se denomina firma digital la cual se enviará adjunta al mensaje original. De esta manera el firmante va a estar adjuntando al documento una marca que es única para ese documento y que sólo él es capaz de producir.

El receptor del mensaje podrá comprobar que el mensaje no fue modificado desde su creación y que el firmante es quién dice serlo a través del siguiente procedimiento: en primer término generará la huella digital del mensaje recibido, luego desencriptará la firma digital del mensaje utilizando la clave pública del firmante y obtendrá de esa forma la huella digital del mensaje original; si ambas huellas digitales coinciden, significa que el mensaje no fue alterado y que el firmante es quien dice serlo [1].

Radius.

RADIUS (acrónimo en inglés de Remote Authentication Dial-In User Server). Es un protocolo de autenticación y autorización para aplicaciones de acceso a la red o movilidad IP. Utiliza el puerto 1812 UDP para establecer sus conexiones.

Cuando se realiza la conexión con un ISP mediante módem, DSL, cablemódem, Ethernet o Wi-Fi, se envía una información que generalmente es un nombre de usuario y una contraseña. Esta información se transfiere a un dispositivo Network Access Server (NAS) sobre el protocolo PPP, quien redirige la petición a un servidor RADIUS sobre el protocolo RADIUS. El servidor RADIUS comprueba que la información es correcta utilizando esquemas de autenticación como PAP, CHAP o EAP. Si es aceptado, el servidor autorizará el acceso al sistema del ISP y le asigna los recursos de red como una dirección IP, y otros parámetros como L2TP, etc.

Una de las características más importantes del protocolo RADIUS es su capacidad de manejar sesiones, notificando cuando comienza y termina una conexión, así que al usuario se le podrá determinar su consumo y facturar en consecuencia; los datos se pueden utilizar con propósitos estadísticos.

Autoridad de certificación.

Una autoridad de certificación es un sistema informático dedicado a la emisión y gestión posterior de certificados digitales, incluyendo la renovación, expiración, suspensión, la habilitación y la revocación de certificados, a petición de la autoridad de registro. La emisión de certificados se hace de una forma automatizada y no sin la previa confirmación de la autoridad local de registro. La función básica de las autoridades de certificación: es verificar la identidad de los solicitantes de certificados [2].

Certificados Digitales.

Son documentos electrónicos que asocian una clave pública con la identidad de su propietario. Tienen como objetivo principal permitir verificar la identidad del usuario, garantizando que únicamente él puede acceder a su información personal, evitando suplantaciones. También es el elemento usado para firmar electrónicamente solicitudes o documentos, ayudando a prevenir que alguien utilice una clave para hacerse pasar por otra persona.

Adicionalmente, además de la clave pública y la identidad de su propietario, un certificado digital puede contener otros atributos para, por ejemplo, concretar el ámbito de utilización de la clave pública, las fechas de inicio y fin de la validez del certificado, etc. El usuario que haga uso del certificado podrá, gracias a los distintos atributos que posee, conocer más detalles sobre las características del mismo y es válido únicamente dentro del período de vigencia, que comienza en la fecha de inicio indicada en el certificado y finaliza en su fecha de vencimiento, o con su revocación si fuere revocado [3].

3. Modelo propuesto

Una vez definidas todas los antecedentes sobre los esquemas de identificación y autenticación en la Fig. 1 se muestra el modelo propuesto que ofrece servicios y/o información a la comunidad del entorno. Cuando se habla de Seguridad de la información en redes de datos, el objetivo es poder garantizar tres conceptos: confidencialidad, autenticación e integridad.

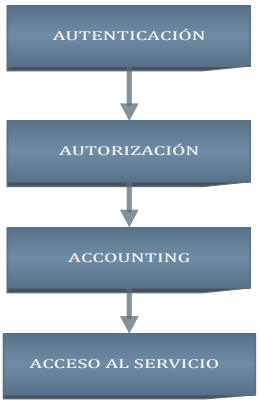

Fig. 1. Modelo Propuesto.

•**Confidencialidad:** Es una propiedad de la información mediante la cual se garantizará el acceso a la misma solo por parte de las personas que estén autorizadas.

•**Autenticación:** Verificación de la identidad de un usuario para así acceder a determinados recursos o poder realizar determinadas tareas.

•**Integridad:** Hace referencia a que todas las características de la información deben ser correctos para que los datos estén correctos.

La propuesta de solución que se presenta conlleva la creación de una Infraestructura de Identificación y Autenticación de Usuarios. En la Fig. 2 se presenta un diagrama a bloques del modelo propuesto, este tendrá un carácter estándar para un manejo en diferentes redes.

4. Protocolo de Seguridad

La figura 2 muestra la secuencia seguida cuando un cliente accede a la red y se desconecta de la misma.

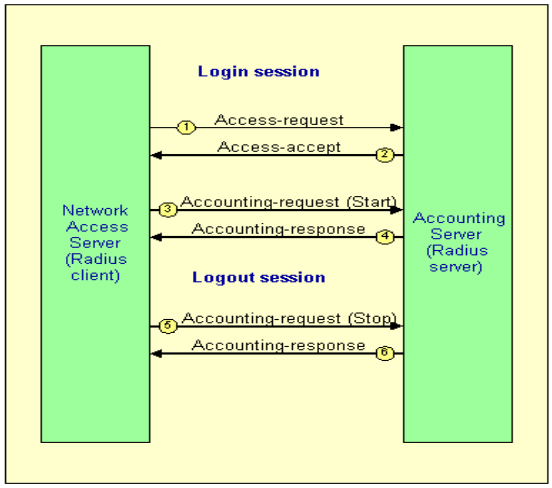

Fig. 2. Diagrama de Secuencia.

1. El cliente envía su usuario/contraseña, esta información es encriptada con una llave secreta y enviada en un Access-Request al servidor RADIUS (Fase de Autenticación).
2. Cuando la relación usuario/contraseña es correcta, entonces el servidor envía un mensaje de aceptación, Access-Accept, con información extra (Por ejemplo: dirección IP, máscara de red, tiempo de sesión permitido, etc.) (Fase de Autorización).
3. El cliente ahora envía un mensaje de Accounting-Request (Start) con la información correspondiente a su cuenta y para indicar que el usuario está reconocido dentro de la red (Fase de Accounting).
   La cual a su vez se comunica con el servidor de certificados para verificar la información y permitir el acceso a la red.
4. El servidor RADIUS responde con un mensaje Accounting-Response, cuando la información de la cuenta es almacenada.
5. Cuando el usuario ha sido identificado, éste puede acceder a los servicios proporcionados. Finalmente, cuando desee desconectarse, enviará un mensaje de Accounting-Request (Stop) con la siguiente información:
   - ♣ Delay Time. Tiempo que el cliente lleva tratando de enviar el mensaje.
   - ♣ Input Octets. Número de octetos recibido por el usuario.
   - ♣ Output Octets. Número de octetos enviados por el usuario.
   - ♣ Session Time. Número de segundos que el usuario ha estado conectado.
   - ♣ Input Packets. Cantidad de paquetes recibidos por el usuario.
   - ♣ Output Packets. Cantidad de paquetes enviados por el usuario.
   - ♣ Reason. Razón por la que el usuario se desconecta de la red.
6. El servidor RADIUS responde con un mensaje de Accounting-Response cuando la información de cuenta es almacenada.

Arquitectura de red

En el sistema descrito en la figura 3 se pueden ver los componentes del sistema que se quiere desarrollar.

En el sistema a desarrollar intervienen una entidad certificadora (CA) y un servidor de autenticación (RADIUS) los cuales proporcionan los certificados para usar en EAP-TLS y el servicio de autenticación respectivamente. Para estos servicios será necesaria la creación de un servidor RADIUS o uno sistema de gestión de certificados. Por lo tanto, estableciendo este sistema cumplimos con los requisitos de establecer una

red con autenticación EAP-TLS apta para trabajar con TKIP ofreciendo: confidencialidad, autenticación, y control de acceso.

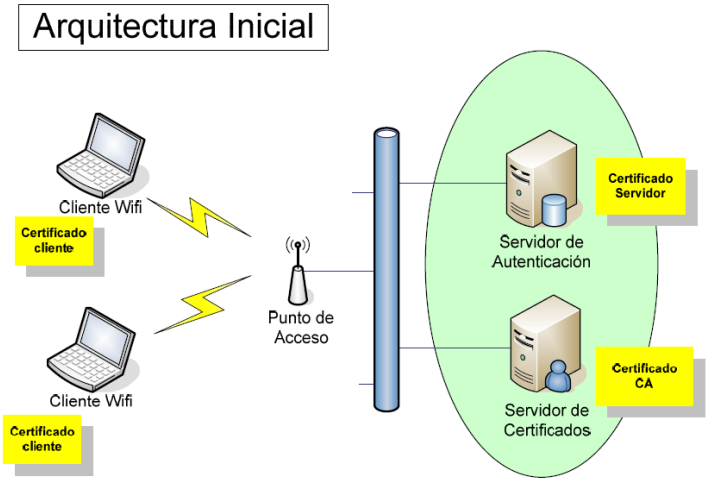

Fig. 3. Arquitectura inicial del sistema

5. Desarrollo

Para la implementación del Modelo de Seguridad y la infraestructura de Seguridad utilizamos las siguientes herramientas de hardware y software:

- Un Servidor de identificación con Linux que fungirá como servidor RADIUS (en nuestro caso, Debian).
- Un Access Point-Router Linksys WRT54G Wi-Fi 802.11b/g (S.O. OpenWRT) [4].
- Equipos con tarjetas inalámbricas usadas como clientes.
- FreeRADIUS como servidor de Autentificación.
- MySQL para el almacenamiento de datos de usuarios.

El entorno educativo implementado en la Escuela Superior de Cómputo del IPN es el medio donde se decidió probar el modelo debido a que es ahí donde se está desarrollando y en este momento existen las facilidades para realizar una implementación de la infraestructura.

El trabajo de implementación del caso de estudio se llevó a cabo en el laboratorio de la maestría de cómputo móvil de la Escuela Superior de Computo en el Instituto Politécnico Nacional. Es en esta área donde se dieron las facilidades para la implementación del modelo.

Como primer paso se muestra el esquema de red (figura 4) sobre el que se trabajará la implementación del caso de estudio, este esquema se basa puntualmente en la arquitectura se propuso y definió en la figura 2.

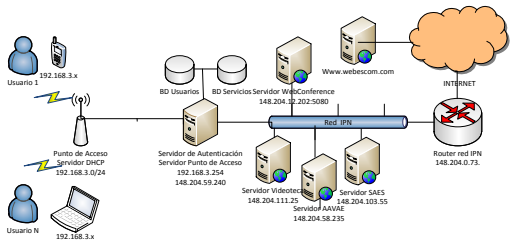

Fig. 4 Esquema de Red para caso de estudio del Modelo

El esquema de red sobre el cual trabajará consta de una red convergente apoyada en el sistema de red institucional que está desplegado para darle servicio a los diferentes usuarios dentro del Instituto Politécnico Nacional, tiene contemplados a más de 18,000 equipos de cómputo conectados al sistema de red, el backbone está

integrado por tres nodos principales: Zacatenco, Santo Tomás y UPIICSA; por medio de enlaces de fibra óptica monomodo con ancho de banda de 12 Gbps entre cada uno de ellos y conexiones a Internet e Internet2 con un ancho de banda de E3 (34 Mbps).

El nodo de interés para este trabajo es el nodo Zacatenco (Figura 5) ya que es el que sirve de conexión a la Escuela Superior de cómputo (ESCOM), dicho nodo es considerado el más importante del backbone, abarca la Unidad Profesional "Adolfo López Mateos" (UPALM) y la Unidad Profesional Ticomán (UPT), a través del nodo Zacatenco se logra la conexión a Internet e Internet 2, así como a los servidores de correo electrónico, los servidores DNS's (Domain Name Service), los servidores de almacenaje (NAS), entre otros. Ahí se encuentran los enlaces satelitales, que se encargan de servirle conexión de red a los 16 centros foráneos que tiene el IPN; los enlaces de fibra óptica y cables de cobre a 40 diferentes centros cercanos a éste nodo; también se encuentran enlaces por microondas hacia UPIICSA, Santo Tomás y UPIITA.

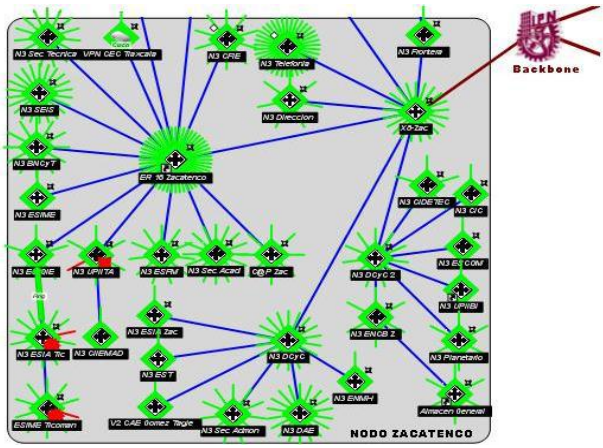

Fig. 5 Nodo Zacatenco. Red Institucional IPN [4].

Continuando con la definición de la arquitectura y siguiendo el orden del diseño, se implementara el módulo de autenticación y autorización que es donde se envía la identificación y se solicita la autenticación por parte del usuario al servidor de autenticación. Para llevar a cabo esto es necesario instalar y configurar un servidor de autenticación lo cual se hace dentro de un servidor principal.

Una vez instalado y configurado el sistema operativo Debian se instala el siguiente software: Freeradius y OpenSSL, Este software es el que permite la implementación del servidor de autenticación.

El siguiente paso es configurar el Servidor de Autenticación [5].

En el cual se maneja la configuración del servidor de autenticación, que es a donde se redirigirán a los usuarios una vez dado el mensaje de bienvenida al dominio y este les presentara al usuario un formulario de identificación el cual se define en este archivo de configuración. Como lo muestra la siguiente figura 6:

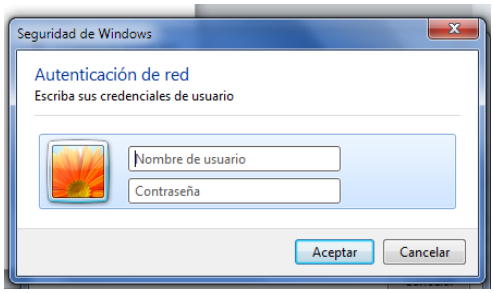

Fig. 6 Pantallas Predeterminadas del Servidor de Autenticación

Para definir la clave privada y pública es necesario crearlas con Openssl utilizando los siguientes comandos en modo consola y ubicarlos en el directorio de trabajo que se define previamente

certdir = /etc/certs/server

```
cadir = ${confdir}/certs/masterCA
private_key_password = misecreto
private_key_file=/etc/certs/server/radiusI_keycert.pem
certificate_file = /etc/certs/server/radiusI_keycert.pem
CA_file = ${cadir}/cacert.pem
dh_file = /etc/certs/dh
random_file = /etc/cets/random
```

Creación de autoridad certificadora:

```
root@debian:/etc/freeradius/certs# ./CA.sh –newca
root@debian:/etc/freeradius/certs# ./CA.sh -newreq
root@debian:/etc/freeradius/certs# ./CA.sh -sign
```

Creación de certificados de clientes:

```
root@debian:/etc/freeradius/certs# openssl dhparam -out dh 1024
root@debian:/etc/freeradius/certs# openssl ca -policy policy_anything -out newcert.pem -infiles newreq.pem
root@debian:/etc/freeradius/certs# openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -out clientcert.p12 -clcerts
root@debian:/etc/freeradius/certs# openssl pkcs12 -passin pass:misecreto -in clientcert.p12 -passout pass:misecreto -out clientcert.pem
```

Se establecen los parámetros en el archivo de configuración del servidor de autenticación para definir el uso de dirección de la BD de usuarios .

Se procese a la instalación del certificado de cliente para poder accesar a la red figura 7.

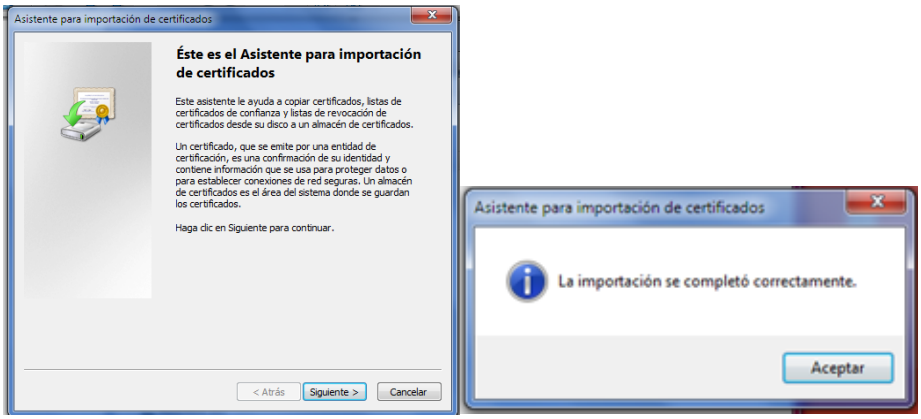

Fig. 7 instalación de certificado

Después de la instalación de certificados ingresamos a una página de bienvenida la cual se muestra en la figura 8.

Fig. 8 Página de bienvenida

6. Conclusiones y trabajos futuros

Podemos darnos cuenta de medios alternos existentes para implementar la seguridad en redes a través de una autenticación controlada a los usuarios.

Linux, junto con FreeRADIUS, son herramientas que permiten esta implementación de una manera efectiva y segura para clientes que utilizan cualquier sistema operativo.

Contando con un manual eficiente, es posible instalar y configurar RADIUS de una manera relativamente sencilla.

El manejo de cuentas también es muy sencillo, sobre todo, si se utiliza algún servidor externo como es el caso de MySQL. Puesto que permite realizar modificaciones y actualizaciones a la base de datos de usuarios, es decir, realizar la administración de ésta.

El aplicar políticas de seguridad no es tarea sencilla; sin embargo, actualmente, se cuenta con herramientas que ayudan a la realización de tan importante tarea.

Actualmente el modelo se encuentra en fase de pruebas; sin embargo, se considera que es una propuesta factible debido a que proporcionará comodidad a los usuarios.

Este modelo tiene como meta: brindar seguridad y eficiencia.

En lo que respecta a trabajo a futuro, se pretende realizar la implantación del modelo de seguridad en un ambiente educativo real.

Agradecimientos: Los autores agradecen al Instituto Politécnico Nacional, en particular a la ESIME, ESCOM, CIC, SIP, COFAA, CONACYT por el apoyo para la realización de este trabajo.

7. Referencias

[1] Yago Fernández Hansen, Antonio Ramos Varón, AAA / RADIUS/ 802.1X, Alfaomega Extraido el 8 de septiembre
[2] Certificado de la ACC, Extraído el 29 de abril de 2011 desde http://www.certificadoscopitec.org.ar/info_conocer.htm
[3]Cifrado digital-Certificados digitales, Extraído el 25 de mayo de 2011 desde http://www.cert.fnmt.es/popup.php?o=faq
[4] Wi-Fi Access Point with hostap + hostapd + freeradius + mysql backend Part 1, Extraído el 20 de September desde http://ubuntuforums.org/showthread.php?t=151781
[5] FreeRADIUS+EAP/PEAP, Extraído el 28 de septiembre de 2011 desde http://ubuntuforums.org/showthread.php?t=478804&highlight=freeradius%2BEAP%2FPEAP

8. Extracto curricular

Norma Alicia Cruz Guzmán es Ingeniero en Comunicaciones y Electrónica, egresado de la Escuela Superior de Ingeniería Mecánica y Eléctrica, ESIME Unidad Zacatenco del IPN (2003). Actualmente estudia la Maestría en Ciencias de Ingeniería de Telecomunicaciones en la Sección de Estudios de Posgrado e Investigación de la ESIME del IPN.

Desarrollo de una Arquitectura de Seguridad para Identificación y Autenticación para Redes

Ing. Norma Alicia Cruz Guzmán[1], M. en C. Chadwick Carreto Arellano[2], Dr. Salvador Álvarez Ballesteros[3]

[1,3] Escuela Superior de Ingeniería Mecánica y Eléctrica - Sección de Estudios de Posgrado e Investigación, Instituto Politécnico Nacional Unidad Profesional Adolfo López Mateos Av. Instituto Politécnico s/n, Edificio Z-4, 3er. Piso. Col. Lindavista C. P. 07738, México, D. F. Tel: +52 55 5729 6000 Ext. 54755,54756. [2] Escuela Superior de Cómputo, Instituto Politécnico Nacional Unidad Profesional Adolfo López Mateos. Av. Juan de Dios Batíz, esquina con Miguel Otón de Mendizábal, Col. Lindavista C. P. 07738, México, D. F. Tel: +52 55 5729 6000 Ext.

E-mail: ncruzg0901@ipn.mx, salvarez@ipn.mx, ccarretoa@ipn.mx

Resumen

En el presente documento se describe el Desarrollo de una Arquitectura de Seguridad para Redes Basado en Identificación y Autenticación, de acuerdo al modelo propuesto, los usuarios del entorno podrán acceder de forma transparente, en cualquier momento y en cualquier lugar, mediante diferentes dispositivos a la red y servicios disponibles dentro de los diferentes dominios de trabajo que maneje este entorno con la garantía de disponer de los servicios de acuerdo a su perfil.

Palabras clave: Autenticación, firma digital, identificación, seguridad.

Summary

In essence, this paper describes the design of a model and network security infrastructure based on identification and authentication, according to the proposed model, users can access the environment seamlessly, anytime, anywhere, using different devices network and services available within the different domains of work, to handle this environment.

Keys words: Authentication, digital signature, identification, security.

1. Introducción

Con la llegada de nuevas tecnologías, su utilización y adopción por gran cantidad de personas, surgen amenazas y es por ello que hoy en día es muy importante hablar de seguridad en cuanto a computación y redes se refiere. La seguridad de la información y de los sistemas, es un punto crítico en una sociedad en la que la información electrónica o digital cada vez obtiene más simpatizantes.

La identidad en la sociedad humana juega un rol muy importante y por ende, se entiende también la necesidad de contar con medios de control que brinden el acceso a recursos, instituciones, derechos, obligaciones, etc. de forma segura, tanto para el portador de la identificación, como del prestador de servicios o instancia que requiere de tal identificación.

Cuando se busca cubrir esta necesidad, se suele recurrir a la creación de identificaciones con la impresión de micas con ciertos elementos de seguridad como marca de agua, holograma bidimensional, firma digitalizada, entre otros. El inconveniente que existe en ellas, es que no te permiten identificarte en múltiples organizaciones y se tiene que portar todas las identificaciones con las que se cuenta y en caso de robo o extravió de cartera, tener que ir a cada organización para solicitar identificaciones nuevas.

La problemática entonces, no es el simple hecho de la identificación, si no la posibilidad de extender sus capacidades en cantidad de

información, portabilidad, seguridad y generalización y hacer uso de los avances tecnológicos actuales, esto bajo una buena propuesta de infraestructura que permita identificar al usuario.

En cuanto a seguridad se refiere, es necesario verificar que el portador de la identificación es quien dice ser y para eso, existen diferentes tecnologías que apoyan el mecanismo de autenticación de usuarios tales como certificados digitales, claves de acceso/password, tarjetas inteligentes, tokens de seguridad, biométricos entre otros.

Dado lo anterior, se vuelve importante la creación de un modelo e infraestructura que al integrar diversas tecnologías para la identificación y la autenticación antes mencionadas, permita verificar la identidad y la autenticidad de una persona de manera segura.

A continuación, en la sección 2 se exhiben los antecedentes para el desarrollo del modelo planteado; en la 3 se describe el modelo propuesto de seguridad para en el 4 definir el protocolo de seguridad, en la 5 el desarrollo, finalmente, en la sección 6 se muestran las conclusiones de la propuesta y se establecen ideas para trabajos a futuro.

2. Antecedentes

En el mundo actual, donde carecemos completamente de la necesaria confianza para poder realizar cualquier operación sin preceder de algún de identificación, donde ya no podemos dejar abiertas las puertas de casa o del coche, la identificación de las personas se ha convertido en una importante necesidad en cualquier ámbito. Para poder identificarnos ante otra persona utilizamos los documentos de identidad aceptados (pasaporte, tarjetas, etc.), presentándolos ante quien los solicita. Sin embargo, en muchas situaciones esto ya no es suficiente, ya que no existe una persona física ante nosotros, encargada de comprobar la autenticidad de nuestro documento, sino un sistema (computadora, red, cajero …) que precisa saber quiénes somos sin posibilidad de duda, para ofrecernos un servicio concreto. La cantidad de situaciones donde se produce esto en la vida real es muy elevada y aumenta cada día. Muchas veces no percibimos el gran alcance de estas situaciones, ya que lo hacemos de forma automática, sin plantearnos la importancia de esta seguridad.

Cada día aumenta el número de proveedores de servicios que nos solicita identificación para todo tipo de funciones que, teóricamente, nos facilitan la vida: acceso público o privado a Internet, telefonía móvil, acceso al banco, compras por Internet, servicios y trámites legales, etc. El problema del manejo de tantas credenciales es tan grave que se venden incluso programas para el archivado seguro o cifrado d todos los nombres de usuario, contraseñas, secretos compartidos, etc. Con lo que tuvimos que lidiar día a día.

Desde el punto de vista del usuario que se identifica ante un sistema, también es de vital importancia que el sistema que nos identifica sea el que suponemos que tiene que ser.

Lo que hay detrás de todos y cada uno de estos servicios es una infraestructura de sistemas basados en la autenticación, que gestiona todos nuestros datos de identificación de forma segura y eficiente.

Seguridad Informática.

Es la primera línea de defensa para la mayoría de los sistemas computarizados, permitiendo prevenir el ingreso de personas no autorizadas. Es la base para la mayor parte de los controles de acceso.

Se denomina “Identificación” al momento en que el usuario se da a conocer en el sistema; y “Autenticación” a la verificación que realiza el sistema sobre esta identificación.

Existen cuatro tipos de técnicas que permiten realizar la autenticación de la identidad del usuario, las cuales pueden ser utilizadas individualmente o combinadas:

•Algo que solamente el individuo conoce: por ejemplo una clave secreta de acceso o password, una clave criptográfica, un número de identificación personal o PIN, etc.

•Algo que la persona posee: por ejemplo una tarjeta magnética.

•Algo que el individuo es y que lo identifica unívocamente: por ejemplo las huellas digitales o la voz.

•Algo que el individuo es capaz de hacer: por ejemplo los patrones de escritura.

Para cada una de estas técnicas vale lo mencionado en el caso de la seguridad física en cuanto a sus ventajas y desventajas. Se destaca que en los dos primeros casos enunciados, es frecuente que las claves sean olvidadas o que las tarjetas o dispositivos se pierdan, mientras que por otro lado, los controles de autenticación biométricos serían los más apropiados y fáciles de administrar, resultando ser también, los más

costosos por lo dificultosos de su implementación eficiente.

Desde el punto de vista de la eficiencia, es conveniente que los usuarios sean identificados y autenticados solamente una vez, pudiendo acceder a partir de allí, a todas las aplicaciones y datos a los que su perfil les permita, tanto en sistemas locales como en sistemas a los que deba acceder en forma remota. Esto se denomina "single login" o sincronización de passwords.

Una de las técnicas para implementar esta única identificación de usuarios sería la utilización de un servidor de autenticaciones sobre el cual los usuarios se identifican, y que se encarga luego de autenticar al usuario sobre los restantes equipos a los que éste pueda acceder. Este servidor de autenticaciones no debe ser necesariamente un equipo independiente y puede tener sus funciones distribuidas tanto geográfica como lógicamente, de acuerdo con los requerimientos de carga de tareas.

La Seguridad Informática se basa, en gran medida, en la efectiva administración de los permisos de acceso a los recursos informáticos, basados en la identificación, autenticación y autorización de accesos.

Riesgos Seguridad

Intrusión.- alguien entra ilegalmente a un sistema y es capaz de utilizarlo y modificarlo como si fuera un usuario legítimo.

Rechazo de Servicios.- alguien logra que no puedan prestarse los servicios a los usuarios legítimos, puede ser dañando al sistema o sobrecargando el sistema o la red.

Robo de Información.- alguien tiene acceso a información confidencial, secreta, reservada o restringida. Es común en espionaje industrial, piratería, etc.

Firma digital

Una firma digital es un conjunto de datos asociados a un mensaje que permite asegurar la identidad del firmante y la integridad del mensaje. La firma digital no implica que el mensaje esté encriptado, es decir, que este no pueda ser leído por otras personas; al igual que cuando se firma un documento holográficamente este sí puede ser visualizado por otras personas.

El procedimiento utilizado para firmar digitalmente un mensaje es el siguiente: el firmante genera mediante una función matemática una huella digital del mensaje. Esta huella digital se encripta con la clave privada del firmante, y el resultado es lo que se denomina firma digital la cual se enviará adjunta al mensaje original. De esta manera el firmante va a estar adjuntando al documento una marca que es única para ese documento y que sólo él es capaz de producir.

El receptor del mensaje podrá comprobar que el mensaje no fue modificado desde su creación y que el firmante es quién dice serlo a través del siguiente procedimiento: en primer término generará la huella digital del mensaje recibido, luego desencriptará la firma digital del mensaje utilizando la clave pública del firmante y obtendrá de esa forma la huella digital del mensaje original; si ambas huellas digitales coinciden, significa que el mensaje no fue alterado y que el firmante es quien dice serlo [1].

Radius.

RADIUS (acrónimo en inglés de Remote Authentication Dial-In User Server). Es un protocolo de autenticación y autorización para aplicaciones de acceso a la red o movilidad IP. Utiliza el puerto 1812 UDP para establecer sus conexiones.

Cuando se realiza la conexión con un ISP mediante módem, DSL, cablemódem, Ethernet o Wi-Fi, se envía una información que generalmente es un nombre de usuario y una contraseña. Esta información se transfiere a un dispositivo Network Access Server (NAS) sobre el protocolo PPP, quien redirige la petición a un servidor RADIUS sobre el protocolo RADIUS. El servidor RADIUS comprueba que la información es correcta utilizando esquemas de autenticación como PAP, CHAP o EAP. Si es aceptado, el servidor autorizará el acceso al sistema del ISP y le asigna los recursos de red como una dirección IP, y otros parámetros como L2TP, etc.

Una de las características más importantes del protocolo RADIUS es su capacidad de manejar sesiones, notificando cuando comienza y termina una conexión, así que al usuario se le podrá determinar su consumo y facturar en consecuencia; los datos se pueden utilizar con propósitos estadísticos.

Autoridad de certificación.

Una autoridad de certificación es un sistema informático dedicado a la emisión y gestión posterior de certificados digitales, incluyendo la

renovación, expiración, suspensión, la habilitación y la revocación de certificados, a petición de la autoridad de registro. La emisión de certificados se hace de una forma automatizada y no sin la previa confirmación de la autoridad local de registro. La función básica de las autoridades de certificación: es verificar la identidad de los solicitantes de certificados [2].

Certificados Digitales.

Son documentos electrónicos que asocian una clave pública con la identidad de su propietario. Tienen como objetivo principal permitir verificar la identidad del usuario, garantizando que únicamente él puede acceder a su información personal, evitando suplantaciones. También es el elemento usado para firmar electrónicamente solicitudes o documentos, ayudando a prevenir que alguien utilice una clave para hacerse pasar por otra persona.

Adicionalmente, además de la clave pública y la identidad de su propietario, un certificado digital puede contener otros atributos para, por ejemplo, concretar el ámbito de utilización de la clave pública, las fechas de inicio y fin de la validez del certificado, etc. El usuario que haga uso del certificado podrá, gracias a los distintos atributos que posee, conocer más detalles sobre las características del mismo y es válido únicamente dentro del período de vigencia, que comienza en la fecha de inicio indicada en el certificado y finaliza en su fecha de vencimiento, o con su revocación si fuere revocado [3].

3. Modelo propuesto

Una vez definidas todas los antecedentes sobre los esquemas de identificación y autenticación en la Fig. 1 se muestra el modelo propuesto que ofrece servicios y/o información a la comunidad del entorno. Cuando se habla de Seguridad de la información en redes de datos, el objetivo es poder garantizar tres conceptos: confidencialidad, autenticación e integridad.

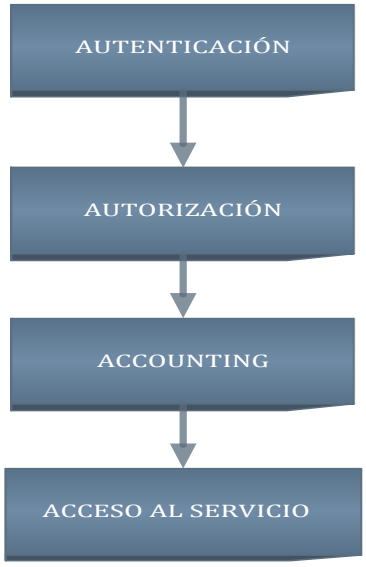

Fig. 1. Modelo Propuesto.

•**Confidencialidad:** Es una propiedad de la información mediante la cual se garantizará el acceso a la misma solo por parte de las personas que estén autorizadas.

•**Autenticación:** Verificación de la identidad de un usuario para así acceder a determinados recursos o poder realizar determinadas tareas.

•**Integridad:** Hace referencia a que todas las características de la información deben ser correctos para que los datos estén correctos.

La propuesta de solución que se presenta conlleva la creación de una Infraestructura de Identificación y Autenticación de Usuarios. En la Fig. 2 se presenta un diagrama a bloques del modelo propuesto, este tendrá un carácter estándar para un manejo en diferentes redes.

4. Protocolo de Seguridad

La figura 2 muestra la secuencia seguida cuando un cliente accede a la red y se desconecta de la misma.

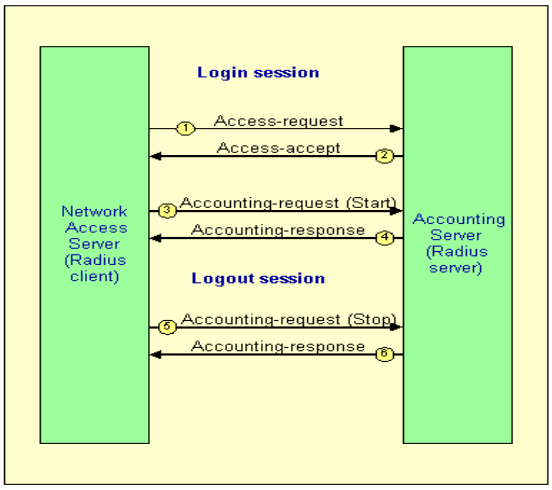

Fig. 2. Diagrama de Secuencia.

6. El cliente envía su usuario/contraseña, esta información es encriptada con una llave secreta y enviada en un Access-Request al servidor RADIUS (Fase de Autenticación).
7. Cuando la relación usuario/contraseña es correcta, entonces el servidor envía un mensaje de aceptación, Access-Accept, con información extra (Por ejemplo: dirección IP, máscara de red, tiempo de sesión permitido, etc.) (Fase de Autorización).
8. El cliente ahora envía un mensaje de Accounting-Request (Start) con la información correspondiente a su cuenta y para indicar que el usuario está reconocido dentro de la red (Fase de Accounting).
   La cual a su vez se comunica con el servidor de certificados para verificar la información y permitir el acceso a la red.
9. El servidor RADIUS responde con un mensaje Accounting-Response, cuando la información de la cuenta es almacenada.
10. Cuando el usuario ha sido identificado, éste puede acceder a los servicios proporcionados. Finalmente, cuando desee desconectarse, enviará un mensaje de Accounting-Request (Stop) con la siguiente información:
    - ♣ Delay Time. Tiempo que el cliente lleva tratando de enviar el mensaje.
    - ♣ Input Octets. Número de octetos recibido por el usuario.
    - ♣ Output Octets. Número de octetos enviados por el usuario.
    - ♣ Session Time. Número de segundos que el usuario ha estado conectado.
    - ♣ Input Packets. Cantidad de paquetes recibidos por el usuario.
    - ♣ Output Packets. Cantidad de paquetes enviados por el usuario.
    - ♣ Reason. Razón por la que el usuario se desconecta de la red.
6. El servidor RADIUS responde con un mensaje de Accounting-Response cuando la información de cuenta es almacenada.

5. Desarrollo

Para la implementación del Modelo de Seguridad y la infraestructura de Seguridad utilizamos las siguientes herramientas de hardware y software:

- ॐ Un Servidor de identificación con Linux que fungirá como servidor RADIUS (en nuestro caso, Debian).
- ॐ Un Access Point-Router Linksys WRT54G Wi-Fi 802.11b/g (S.O. OpenWRT) [4].
- ॐ Equipos con tarjetas inalámbricas usadas como clientes.
- ॐ FreeRADIUS como servidor de Autentificación .
- ॐ MySQL para el almacenamiento de datos de usuarios.

El entorno educativo implementado en la Escuela Superior de Cómputo del IPN es el medio donde se decidió probar el modelo debido a que es ahí donde se está desarrollando y en este momento existen las facilidades para realizar una implementación de la infraestructura.

El trabajo de implementación del caso de estudio se llevó a cabo en el laboratorio de la maestría de cómputo móvil de la Escuela Superior de Computo en el Instituto Politécnico Nacional. Es en esta área donde se dieron las facilidades para la implementación del modelo.

Como primer paso se muestra el esquema de red (figura 3) sobre el que se trabajará la implementación del caso de estudio, este esquema se basa puntualmente en la arquitectura se propuso y definió en la figura 1.

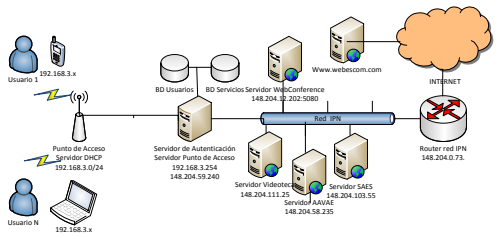

Fig. 3 Esquema de Red para caso de estudio del Modelo

El esquema de red sobre el cual trabajará consta de una red convergente apoyada en el sistema de red institucional que está desplegado para darle servicio a los diferentes usuarios dentro del Instituto Politécnico Nacional, tiene contemplados a más de 18,000 equipos de cómputo conectados al sistema de red, el backbone está integrado por tres nodos principales: Zacatenco, Santo Tomás y UPIICSA; por medio de enlaces de fibra óptica monomodo con ancho de banda de 12 Gbps entre cada uno de ellos y conexiones a Internet e Internet2 con un ancho de banda de E3 (34 Mbps).

El nodo de interés para este trabajo es el nodo Zacatenco (Figura 4) ya que es el que sirve de conexión a la Escuela Superior de cómputo (ESCOM), dicho nodo es considerado el más importante del backbone, abarca la Unidad Profesional "Adolfo López Mateos" (UPALM) y la Unidad Profesional Ticomán (UPT), a través del nodo Zacatenco se logra la conexión a Internet e Internet 2, así como a los servidores de correo electrónico, los servidores DNS's (Domain Name Service), los servidores de almacenaje (NAS), entre otros. Ahí se encuentran los enlaces satelitales, que se encargan de servirle conexión de red a los 16 centros foráneos que tiene el IPN; los enlaces de fibra óptica y cables de cobre a 40 diferentes centros cercanos a éste nodo; también se encuentran enlaces por microondas hacia UPIICSA, Santo Tomás y UPIITA.

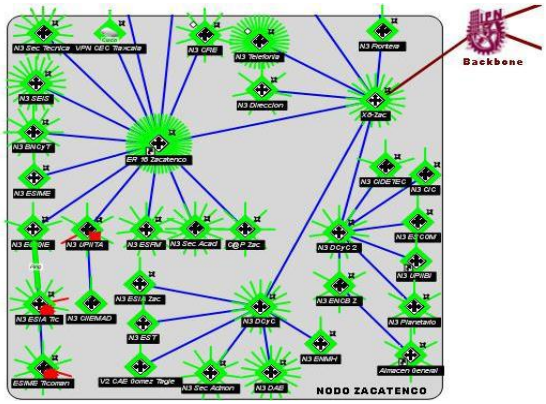

Fig. 4 Nodo Zacatenco. Red Institucional IPN.[4]

Continuando con la definición de la arquitectura y siguiendo el orden del diseño, se implementara el módulo de autenticación y autorización que es donde se envía la identificación y se solicita la autenticación por parte del usuario al servidor de autenticación. Para llevar a cabo esto es necesario instalar y configurar un servidor de autenticación lo cual se hace dentro de un servidor principal.

Una vez instalado y configurado el sistema operativo Debian se instala el siguiente software: Freeradius y OpenSSL, Este software es el que permite la implementación del servidor de autenticación.

El siguiente paso es configurar el Servidor de Autenticación [5].

En el cual se maneja la configuración del servidor de autenticación, que es a donde se redirigirán a los usuarios una vez dado el mensaje de bienvenida al dominio y este les presentara al usuario un formulario de identificación el cual se define en este archivo de configuración. Como lo muestra la siguiente figura 5:

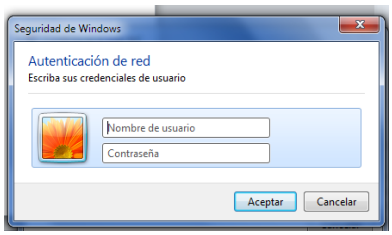

Fig. 5 Pantallas Predeterminadas del Servidor de Autenticación

Para definir la clave privada y pública es necesario crearlas con Openssl utilizando los siguientes comandos en modo consola y ubicarlos en el directorio de trabajo que se define previamente

certdir = /etc/certs/server
cadir = ${confdir}/certs/masterCA
private_key_password = misecreto
private_key_file=/etc/certs/server/radius1_keycert
.pem
certificate_file =
/etc/certs/server/radius1_keycert.pem
CA_file = ${cadir}/cacert.pem
dh_file = /etc/certs/dh
random_file = /etc/cets/random

Creación de autoridad certificadora:

root@debian:/etc/freeradius/certs# ./CA.sh –
newca
root@debian:/etc/freeradius/certs# ./CA.sh -
newreq
root@debian:/etc/freeradius/certs# ./CA.sh -sign

Creación de certificados de clientes:

root@debian:/etc/freeradius/certs# openssl
dhparam -out dh 1024
root@debian:/etc/freeradius/certs# openssl ca -
policy policy_anything -out newcert.pem -infiles
newreq.pem
root@debian:/etc/freeradius/certs# openssl pkcs12

-export -in newcert.pem -inkey newreq.pem -out clientcert.p12 -clcerts
root@debian:/etc/freeradius/certs# openssl pkcs12 -passin pass:misecreto -in clientcert.p12 -passout pass:misecreto -out clientcert.pem

Se establecen los parámetros en el archivo de configuración del servidor de autenticación para definir el uso de dirección de la BD de usuarios .

Se procese a la instalación del certificado de cliente para poder accesar a la red figura 6.

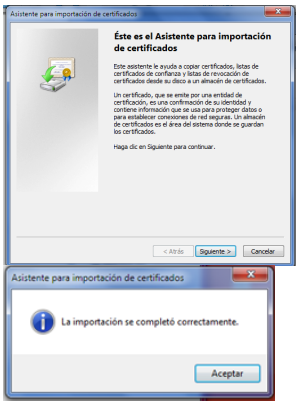

Fig. 6 Instalación de certificado

6. Conclusiones y trabajos futuros

Podemos darnos cuenta de medios alternos existentes para implementar la seguridad en redes a través de una autenticación controlada a los usuarios.

Linux, junto con FreeRADIUS, son herramientas que permiten esta implementación de una manera efectiva y segura para clientes que utilizan cualquier sistema operativo.

Contando con un manual eficiente, es posible instalar y configurar RADIUS de una manera relativamente sencilla.

El manejo de cuentas también es muy sencillo, sobre todo, si se utiliza algún servidor externo como es el caso de MySQL. Puesto que permite realizar modificaciones y actualizaciones a la base de datos de usuarios, es decir, realizar la administración de ésta.

El aplicar políticas de seguridad no es tarea sencilla; sin embargo, actualmente, se cuenta con herramientas que ayudan a la realización de tan importante tarea.

Actualmente el modelo se encuentra en fase de pruebas; sin embargo, se considera que es una propuesta factible debido a que proporcionará comodidad a los usuarios.

Este modelo tiene como meta: brindar seguridad y eficiencia.

En lo que respecta a trabajo a futuro, se pretende realizar la implantación del modelo de seguridad en un ambiente educativo real.

Agradecimientos: Los autores agradecen al Instituto Politécnico Nacional, en particular a la ESIME, ESCOM, CIC, SIP, COFAA, CONACYT por el apoyo para la realización de este trabajo.

7. Referencias

[1] Yago Fernández Hansen, Antonio Ramos Varón, AAA / RADIUS/ 802.1X, Alfaomega Extraido el 8 de septiembre
[2] Certificado de la ACC, Extraído el 29 de abril de 2011 desde http://www.certificadoscopitec.org.ar/info_conocer.htm
[3]Cifrado digital-Certificados digitales, Extraído el 25 de mayo de 2011 desde http://www.cert.fnmt.es/popup.php?o=faq
[4] Wifi Access Point with hostap + hostapd + freeradius + mysql backend Part 1, Extraído el 20 de September desde http://ubuntuforums.org/showthread.php?t=151781
[5] FreeRADIUS+EAP/PEAP, Extraído el 28 de septiembre de 2011 desde http://ubuntuforums.org/showthread.php?t=478804&highlight=freeradius%2BEAP%2FPEAP

8. Extracto curricular

Norma Alicia Cruz Guzmán es Ingeniero en Comunicaciones y Electrónica, egresado de la Escuela Superior de Ingeniería Mecánica y Eléctrica, ESIME Unidad Zacatenco del IPN (2003). Actualmente estudia la Maestría en Ciencias de Ingeniería de Telecomunicaciones en la Sección de Estudios de Posgrado e Investigación de la ESIME del IPN.

GLOSARIO DE TÉRMINOS

Ancho de Banda: Para señales analógicas, el ancho de banda es la anchura, medida en Hertz, del rango de frecuencias en el que se concentra la mayor parte de la potencia de la señal. Puede ser calculado a partir de una señal temporal mediante el análisis de Fourier.

Conmutador: Es un dispositivo electrónico de interconexión de redes de ordenadores que opera en la capa 2 (nivel de enlace de datos) del modelo OSI (Open Systems Interconection). Un conmutador interconecta dos o más segmentos de red, funcionando de manera similar a los puentes (bridges), pasando datos de un segmento a otro, de acuerdo con la dirección MAC de destino de los datagramas en la red.

DDWRT: Es un firmware libre para diversos routers inalámbricos o WIFI, es muy común observarlo en equipos Linksys WRT54G. Ejecuta un reducido sistema operativo basado en Linux. Está licenciado bajo la GNU General Public License versión 2.

DHCP: Protocolo de Configuración Dinámica de Servidor (Dynamic Host Configuration Protocol) es un protocolo de red en el que un servidor provee los parámetros de configuración a las computadoras conectadas a la red informática que los requieran y también incluye un mecanismo de asignación de direcciones IP.

DNS: Sistema de Nombre de Dominio (Domain Name System) Es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado al internet o a una red privada. Este sistema asocia información variada con nombres de dominios asignado a cada uno de los participantes. Su función más importante, es traducir (resolver) nombres inteligibles para los humanos en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos.

Encriptación: Es el proceso mediante el cual cierta información o texto sin formato es cifrado de forma que el resultado sea ilegible a menos que se conozcan los datos necesarios para su interpretación. Es una medida de seguridad utilizada para que al momento de almacenar o transmitir información sensible ésta no pueda ser obtenida con facilidad por terceros.

ESCOM: Escuela Superior de Cómputo

Ethernet: Es el nombre de una tecnología de redes de computadoras de área local (LANs) basada en tramas de datos. Ethernet define las características de

cableado y señalización de nivel físico y los formatos de trama del nivel de enlace de datos del modelo OSI.

Firmware: Programación en Firme, es un bloque de instrucciones de programa para propósitos específicos, grabado en una memoria tipo ROM, que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Al estar integrado en la electrónica del dispositivo es en parte hardware, pero también es software, ya que proporciona lógica y se dispone en algún tipo de lenguaje de programación. Funcionalmente, el firmware es el intermediario (interfaz) entre las órdenes externas que recibe el dispositivo y su electrónica, ya que es el encargado de controlar a ésta última para ejecutar correctamente dichas órdenes externas

Gateway: Véase Puerta de Enlace

HTML: Acrónimo de Lenguaje de Formato de Documentos de Hipertexto (HyperText Markup Language), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

HTTP: Protocolo de Transferencia de Hipertexto (HyperText Transfer Protocol). Define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

IEEE: Institute of Electrical and Electronics Engineers. Instituto de Ingenieros Electricistas y Electrónicos. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros electricistas, ingenieros en electrónica, científicos de la computación, ingenieros en informática, ingenieros en biomédica, ingenieros en telecomunicación e Ingenieros en Mecatrónica. Dedicada principalmente a la difusión de los avances de las tecnologías recientes y estandarización de las mismas.

Internet: Es un método de interconexión descentralizada de redes de computadoras implementado en un conjunto de protocolos denominado TCP/IP y garantiza que redes físicas heterogéneas funcionen como una red lógica única, de alcance mundial.

IP (Internet Protocol): Protocolo de Internet. Es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados. Los datos en una red basada en IP son enviados en bloques conocidos como paquetes o datagramas.

IPN: Instituto Politécnico Nacional.

Kernel: Es la parte fundamental de un sistema operativo. Es el software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

LAN: Local Area Network (Red de Área Local o simplemente Red Local). Una red local es la interconexión de varios ordenadores y periféricos. Su extensión está limitada físicamente a un edificio o a un entorno de unos pocos kilómetros. Su aplicación más extendida es la interconexión de ordenadores personales y estaciones de trabajo en oficinas, fábricas, etc; para compartir recursos e intercambiar datos y aplicaciones.

LDAP: Son las siglas de Lightweight Directory Access Protocol (en español Protocolo Ligero de Acceso a Directorios) que hacen referencia a un protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también es considerado una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.
Habitualmente, almacena la información de autenticación (usuario y contraseña) y es utilizado para autenticarse aunque es posible almacenar otra información (datos de contacto del usuario, ubicación de diversos recursos de la red, permisos, certificados, etc).

Linux: Es una distribución de software basada en el núcleo Linux que incluye determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones domésticas, empresariales y para servidores. Por lo general están compuestas, total o mayoritariamente, de software libre, aunque a menudo incorporan aplicaciones o controladores propietarios.

MAC: Dirección Media Access Control address (Dirección de Control de Acceso al Medio) es un identificador hexadecimal de 48 bits que se corresponde de forma única con una tarjeta o interfaz de red. Es individual, cada dispositivo tiene su propia dirección MAC determinada y configurada por el IEEE (los primeros 24 bits) y el fabricante (los últimos 24 bits).

OSI: Open Systems Interconnection (Interconexión de sistemas Abiertos). Es un estándar del ISO para las comunicaciones mundiales que define un marco de trabajo para implementar protocolos en siete capas

Perl: Lenguaje Práctico de Extracción y Reporte (Practical Extraction and Report Language) es un lenguaje de programación desarrollado por Larry Wall inspirado en otras herramientas de UNIX.

Puerta de Enlace: Son equipos para interconectar redes con protocolos y arquitecturas completamente diferentes a todos los niveles de comunicación. Es normalmente un equipo informático configurado para dotar a las máquinas de una red local (LAN) conectadas a él de un acceso hacia una red exterior, generalmente realizando para ello operaciones de traducción de direcciones IP. Esta capacidad de traducción de direcciones permite aplicar una técnica llamada IP Masquerading (enmascaramiento de IP), usada muy a menudo para dar acceso a Internet a los equipos de una red de área local compartiendo una única conexión a Internet, y por tanto, una única dirección IP externa. Se podría decir que un Gateway, o puerta de enlace, es un router que conecta dos redes.

RFC: Request For Comment (Petición de comentarios). Es un documento cuyo contenido es una propuesta oficial para un nuevo protocolo de la red Internet, que se explica con todo detalle para que en caso de ser aceptado pueda ser implementado sin ambigüedades.

Router: Véase Ruteador.

Ruteador: (o encaminador) Es un dispositivo de hardware para interconexión de redes de las computadoras que opera en la capa tres (nivel de red)

Servidor: Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de un ordenador y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final.

SNMP: Simple Network Management Protocol (Protocolo Simple de administración de red) es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. Es parte de la suite de protocolos TCP/IP. SNMP permite a los administradores supervisar el desempeño de la red, buscar y resolver sus problemas, y planear su crecimiento.

Switch: Véase Conmutador.

Wi-Fi: Acrónimo de Wireless Fidelity, es un conjunto de estándares para redes inalámbricas basado en las especificaciones IEEE 802.11. Wi-Fi se creó para ser utilizada en redes locales inalámbricas, pero es frecuente que en la actualidad también se utilice para acceder a Internet. Wi-Fi es una marca de la Wi-Fi Alliance, la organización comercial que prueba y certifica que los equipos cumplen los estándares IEEE 802.11.

WiMAX: Acrónimo de Worldwide Interoperability for Microwave Access (Interoperabilidad Mundial para Acceso por Microondas), es una norma de transmisión por ondas de radio de última generación orientada a la última milla

que permite la recepción de datos por microondas y retransmisión por ondas de radio (protocolo basado en la especificación de IEEE 802.16 MAN - Metropolitan Area NetWork, Red de Área Metropolitana) proporcionando acceso concurrente con varios repetidores de señal superpuestos, ofreciendo total cobertura en áreas de hasta 48 km de radio y a velocidades de hasta 70 Mbps, utilizando tecnología que no requiere visión directa con las estaciones base (a diferencia de las microondas).