Universidad de Mendoza

Facultad de Ingeniería

Tesis de Maestría en Teleinformática

Arquitectura Unificada para Control de Acceso en Redes Inalámbricas Seguras

Ing. Pablo Fidel Gómez Vergara

Directores de Tesis:

Magíster en Teleinformática Ing. Juan José Ciarlante

Ing. Osvaldo Rosso

Mendoza, Agosto de 2007

Índice de contenido

I. RESUMEN

La popularización de las redes inalámbricas y, en consecuencia, la disminución del costo del hardware inalámbrico han planteado nuevos desafíos para los administradores de red. Actualmente es inconcebible un dispositivo móvil sin comunicación inalámbrica.

La implementación de redes inalámbricas en ámbitos semi públicos constituye uno de los desafíos no resueltos en la actualidad. Al contrario de los ámbitos puramente corporativos, en escenarios de este tipo, los administradores no pueden definir las características de conectividad de los dispositivos inalámbricos que poseen los usuarios. Adicionalmente es muy frecuente la necesidad de diferenciar el tipo de servicios que se brindan de acuerdo a las necesidades de distintos grupos de usuarios.

Estas necesidades plantean la conveniencia de solucionar una problemática que no es cubierta por ninguno de los protocolos inalámbricos existentes en la actualidad. En efecto, en general cada protocolo o método de acceso resuelve una parte del problema, y generalmente de manera excluyente; por lo tanto no permite la integración de soluciones diferentes sobre los mismos equipos en simultáneo.

En este trabajo se aborda la problemática planteada de acceso inalámbrico en un ámbito semi público, y se presenta un modelo de infraestructura de base, cuya implementación permite:

- brindar puntos de acceso a redes inalámbricas mediante diferentes métodos de acceso,

- tipificar los servicios que se brindarán en la red de acuerdo a los tipos de usuario que se conectarán a dicha red,

- implementar el funcionamiento de diversos dispositivos clientes con distintos sistemas operativos,

De esta forma satisface las necesidades planteadas anteriormente.

En el desarrollo de este trabajo se abordarán los diferentes estándares de conexión para redes inalámbricas de área local, se analizarán los distintos métodos de acceso y se presentarán las herramientas con que cuenta GNU/Linux para administrar usuarios de estas conexiones inalámbricas.

II. INTRODUCCIÓN

La popularización de las tecnologías de red inalámbricas han planteado nuevos desafíos de conectividad, en las que los administradores de red requieren de nuevas herramientas que permitan modelar soluciones para estas nuevas necesidades.

Particularmente en ámbitos donde es necesario implementar un acceso del tipo semi público, se generan escenarios de mayor complejidad. Dado que en este tipo de ámbitos en general resulta necesaria una amplia cobertura de la red inalámbrica, no es posible definir las características de conectividad de los dispositivos que se conecten y se requiere que la red brinde servicios diferenciados a diferentes grupos de usuarios.

Generalmente, al realizar la implementación de puntos de acceso inalámbrico en la red de una organización de tipo corporativo, se opta por un protocolo y un método de acceso que cumpla con las políticas definidas por dicha organización. Se definen también, en esta elección, las características que deben cumplir los dispositivos de los usuarios, excluyendo a todos los equipos que no cumplen con dichas características.

Este tipo de elecciones se torna difícil en los escenarios con acceso del tipo semi público, ya que esta última definición, sobre los equipos de los usuarios, no puede llevarse a cabo. Y por lo tanto la elección de un único método de acceso puede traer aparejada la imposibilidad de conectarse a la red por parte de los usuarios cuyo equipamiento no cumple con estas características.

Desde el punto de vista de la seguridad global de la red de una organización, una red inalámbrica en un ámbito semi público debería tratarse como una red separada de la red privada. En efecto, es potencialmente insegura, por las características del medio de transmisión y por el diferente grado de confiabilidad que pueden presentar los distintos grupos de usuarios de dicha red. Esto último obliga a restringir los servicios de acceso de acuerdo al grado de confiabilidad que presenta cada grupo de usuarios.

De esto surge la necesidad de tipificar a los usuarios mediante diferentes perfiles de conectividad dentro de la red inalámbrica y de almacenar información que permita auditar algunos parámetros de comportamiento de cada usuario dentro de la red inalámbrica.

Por otro lado la proliferación de dispositivos con capacidades inalámbricas como las computadoras portátiles, los PDA[1] y los teléfonos móviles, marcan una fuerte tendencia al crecimiento en el uso de estos dispositivos. Ante esto el modelo a plantear debe ser capaz de soportar un crecimiento en cantidad de dispositivos, y por lo tanto en puntos de acceso y en cantidad de usuarios, sin presentar mayores complejidades en la implementación.

1. OBJETIVOS

El presente desarrollo propone:

- Plantear un “modelo de infraestructura de base” para la implementación del acceso a redes inalámbricas de tipo semi públicas que permita:

1 Personal Digital Assistant, Asistente personal digital.

- Identificar el acceso de cada usuario a la red, independientemente del dispositivo que utilice para realizar dicho acceso.

- Definir una política de uso del recurso inalámbrico, implementada a través de perfiles de usuario y permisos sobre cada usuario, que permita regular el acceso de los mismos a los diferentes servicios de conectividad.

- Brindar diferentes métodos de acceso que permitan la conexión de un amplio espectro de equipamiento de diferentes plataformas a través de protocolos abiertos, con mínimas necesidades de intervención por parte de los administradores de red.

- Implementar un “modelo de infraestructura de base” que provea herramientas amigables para la administración de usuarios, la definición de los perfiles de los mismos y los servicios que se brinden a dichos usuarios.

- Utilizar herramientas de software libre y GNU Linux en la implementación de dicho modelo.

III. MARCO TEÓRICO

En este capítulo, se realiza una revisión de los conceptos necesarios para la presentación del modelo propuesto. Se comienza por los conceptos de redes inalámbricas y sus estándares. Se continúa por los métodos de acceso, sus protocolos de autenticación, y protocolos genéricos para realizar *AAA*[1]. Y se concluye con los conceptos de servicios de directorio donde se almacenan los datos de los usuarios. Luego se presenta el planteo del modelo y su implementación.

1. REDES DE ÁREA LOCAL INALÁMBRICAS

Las redes inalámbricas [GAST00] están definidas en el estándar 802.11 [IEEE80211] de la *IEEE*[2]. El estándar 802.11 es un miembro de la familia 802 [IEEE802], y consiste en una serie de especificaciones para las tecnologías de redes de área local (*LAN*[3]). Dichas especificaciones están enfocadas en las dos capas inferiores del modelo *OSI [ISOOSI]* debido a que incorporan elementos tanto de la capa física (*PHY*[4]) como de la capa de enlace.

Todas la redes del tipo 802 poseen ambas capas, la capa de enlace tiene por objeto determinar cómo se accede al medio de transmisión para enviar o recibir los datos, pero los detalles de cómo esos datos son transmitidos o recibidos es el objeto de la capa física.

---

1 del inglés *Authentication, Authorization and Accounting*, autenticación, autorización y contabilidad. Es un protocolo o sistema que permite a los usuarios proveer una identidad, obtener acceso a recursos y recolectar estadísticas de utilización. El protocolo más común para AAA es RADIUS.

2 I*nstitute of Electronics and Electrical Engineers*, Instituto de Ingenieros en Electrónica y Electricidad.

3 *Local Area Networks*, redes de área local.

4 Abreviatura del inglés de la palabra *Physical*, física.

Las especificaciones individuales en la serie 802 se identifican por segundo número. Por ejemplo, 802.3 [IEEE8023]. es la especificación para las redes denominados genéricamente *Ethernet CSMA/CD*[1]. 802.5 es la especificación para *Token Ring*. 802.2 especifica una capa de enlace común, la *LLC*[2] que puede ser utilizada por cualquier tecnología de *LAN* de una capa inferior. En la especificaciones 802.1 se definen las características de administración como 802.1D (*Bridging*[3]) o 802.1Q (*VLANs*[4]).

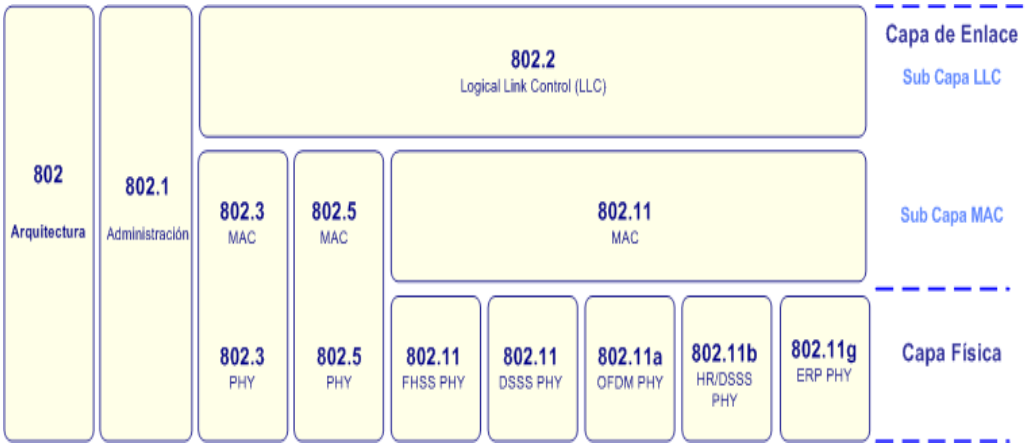

Ilustración III.1.: Familia de Protocolos IEEE802.

1 del inglés *Carrier Sense Multiple Access with Collision Detection*, sensado de portadora con acceso múltiple y detección de colisiones.

2 del inglés *Logical Link Control,* control del enlace lógico.

3 del inglés *Bridge*, especifica las conexiones de puente.

4 del inglés *Virtual LANs*, redes virtuales.

1. LICENCIAS Y REGULACIONES

En las transmisiones de radio, evitar las interferencias es un problema que debe ser contemplado en la legislación. Es por eso que una autoridad debe imponer las reglas de utilización del espectro de radiofrecuencia (*RF*). Por ejemplo en los EE.UU. esta tarea recae sobre la *Comisión Federal de Comunicaciones* (*FCC*). En Europa la asignación de *RF* es responsabilidad de la *Oficina Europea de Radiocomunicaciones* (*ERO*). La compatibilización de las normas de cada país o región es realizada por la *Unión Internacional de Telecomunicaciones* (*ITU*).

Debido a esto, quien desee transmitir en el espectro *RF*, deberá poseer una licencia otorgada por los organismos reguladores. Las licencias pueden restringir la frecuencia asignada y la potencia de transmisión, así como el área geográfica en que las señales de *RF* pueden ser utilizadas. De esta forma el licenciatario de una frecuencia tiene utilización exclusiva de la frecuencia asignada, evitando la interferencia en la uso de dicha frecuencia *de* RF.

Para eso, el radio espectro se divide en bandas dedicadas a un propósito particular. Una banda es definida como el conjunto de frecuencias de *RF* que una aplicación particular puede utilizar. Existen varias bandas de frecuencia destinadas a una utilización de dispositivos hogareños, por ejemplo, para los hornos de microondas, los teléfonos inalámbricos y las LAN inalámbricas. Estas frecuencias generalmente se denominan *Bandas de Frecuencia ISM*[1], y la principal característica es que son de *operación no licenciada*, y solamente presentan restricciones en la potencia de transmisión (ver Tabla III.1).

1 del inglés *Industrial Scientific and Medical use*, uso industrial, científico y médico.

Bandas de Frecuencia	Rango de Frecuencias
UHF ISM	902-928 MHz
Banda S	2 – 4 GHz
Banda S ISM	2.4 – 2.5 GHz
Banda C	4 -8 GHz
Banda C para Bajada de Satélite	3.7 – 4.2 GHz
Banda C de Radar	5.25 -5.925 GHz
Banda C ISM	5.725 – 5.875 GHz
Banda C para subida a satélite	5.925 – 6.425 GHz
Banda X	8 – 12 GHz
Banda X (policía y radar)	8.5 – 10.55 GHz
Banda Ku	12 – 18 GHz
Banda Ku Radar (policía)	13.4 – 14 GHz 15.7 – 17.7 GHz

Tabla III.1.: Bandas de frecuencia comunes en EEUU[1].

2. REDES 802.11

La especificación original 802.11, que data de 1997, incluye la capa de enlace y dos capas físicas posibles por transmisión de radio, *FHSS*[2] y DSSS[3]. Las revisiones posteriores agregaron nuevas capas físicas como por ejemplo, 802.11b que especifica *HR/DSSS*[4], 802.11a que agrega *OFDM*[5] y 802.11g que hace lo propio con *ERP*[6].

---

1 Se puede descargar el mapa completo del uso del espectro electromagnético en EEUU de la *National Telecommunications and Information Administration* en http://www.ntia.doc.gov/osmhome/allochrt.pdf .

2 *Frecuency Hopping Spread Spectrum*, espectro ensanchado por salto de frecuencia

3 *Direct Sequence Spread Spectrum*, espectro ensanchado por secuencia directa

4 *High Rate DSSS,*

5 *Orthogonal Frequency Division Multiplexing,* multiplexación por division de frecuencias ortogonales.

6 *Extended Rate PHY,* interfaz física de rango extendido, utiliza varias especificaciones de intefaz física en una sola definición.

Estas capas físicas utilizan tecnología *Spread Spectrum* (Espectro Ensanchado) la cual es la base para aprovechar las bandas *ISM* para transmitir datos. Esta tecnología utiliza funciones matemáticas para distribuir la potencia de la señal a transmitir a lo largo de un amplio rango de frecuencias. Cuando el receptor realiza la operación inversa, la señal esparcida en el rango de frecuencias se reconstituye como una señal normal de banda angosta.

El esparcir la transmisión en un amplio rango de frecuencias hace que estas transmisiones sean vistas como ruido por los receptores de banda angosta. Si bien esta tecnología es robusta frente a las interferencias, si existen una gran cantidad de dispositivos se interferirán indefectiblemente. A fin de minimizar esta interferencia la *FCC* limita la potencia de los transmisores *SS* a 1 Watt y a 4 Watts de potencia efectiva irradiada *(ERP*). La utilización de tecnologías *Spread Spectrum* es una obligación para los equipos que operan en las bandas no licenciadas.

Las capas físicas basadas en *RF* en 802.11 utilizan tres técnicas de *Spread Spectrum* diferentes:

- *Frequency Hopping o Salto de Frecuencia (FH o FHSS):* En esta técnica los sistemas saltan de una frecuencia a otra en un patrón aleatorio, transmitiendo una corta ráfaga de datos en cada una de estas frecuencias o canales. Esta capa física esta definida en la clausula 14 del estándar.

- *Direct Sequence o Secuencia Directa (DS o DSSS*): En esta técnica los sistemas esparcen la potencia de salida sobre una amplia banda de frecuencias utilizando funciones de codificación

matemática. Esta capa física tiene dos definiciones en el estándar: en la clausula 15 está definida la *DSSS* a 2 Mbps original y en la clausula 18 está definida la *HR/DSSS* utilizada en 802.11b.

- *Orthogonal Frequency Division Multiplexing o Multiplexación por División de Frecuencias Ortogonales (OFDM*): Esta técnica divide cada canal disponible en varios subcanales y codifica una porción de la señal en cada subcanal en paralelo. La clausula 17 especifica *OFDM* para 802.11a a 5 GHz y la clausula 18 especifica *ERP* para 802.11g que es básicamente lo mismo pero para frecuencias de 2.4 Ghz.

Las *LAN* 802.11 no sólo se diferencian por las frecuencias de transmisión y técnicas de codificación, sino también por las velocidades de transmisión que pueden alcanzar como se puede observar en la Tabla III.2.

Estándar IEEE	Velocidad de Transmisión	Banda de Frecuencia	Comentario
802.11	1 Mbps 2 Mbps	2.4 GHz	Primer estándar de capa física (1997) definido para las técnicas de modulación DSSS y FHSS
802.11a	hasta 54 Mbps	5 GHz	Segundo estándar de capa física (1999). Sin productos en el mercado hasta finales del 2000.
802.11b	5.5 Mbps 11 Mbps	2.4 GHz	Tercer estándar de capa física. Posee la mayor base instalada actualmente (2006).
802.11g	hasta 54 Mbps	2.4 GHz	Cuarto estándar de capa física (2003). Aplica técnicas de codificación usadas en 802.11a para obtener mayor velocidad de transmisión en la banda de 2.4 GHz y provee compatibilidad con las redes 802.11b. Es la tecnología más común en notebooks desde 2005.

Tabla III.2.: Comparación de las capas físicas de 802.11

2. COMPONENTES DE UNA RED INALÁMBRICA

1. ELEMENTOS DE LA INFRAESTRUCTURA DE RED

Los elementos que componen la infraestructura de una red inalámbrica son cuatro:

- *Estaciones:* Las estaciones son dispositivos computacionales que poseen una interfaz de red inalámbrica. Típicamente estos dispositivos son *Notebooks, PDA,* etc. pero pueden ser computadoras normales en lugares en que se ha optado no realizar un cableado de red y utilizar tecnologías inalámbricas solamente. Adicionalmente varios fabricantes de dispositivos electrónicos están utilizando 802.11 para comunicar dispositivos no computacionales.

- *Access Points o Puntos de Acceso (AP):* Los *AP* fundamentalmente cumplen la función de *bridge* entre una red inalámbrica y una red cableada, transformando los marcos 802.11 a otro tipos de marcos para poder enviarlos al resto de la red. Esta no es la única función que cumplen los *AP*, pero es la más importante.

- *Medio de Transmisión Inalámbrico:* Es el medio de transmisión utilizado por las estaciones para enviar y recibir marcos. Si bien 802.11 define varias capas físicas diferentes, las capas basadas en *RF* han sido mucho más populares que las capas basadas en transmisión Infrarroja (*IR*). El hecho de que las señale no están circunscriptas a un medio físico, como por ejemplo un cable, tiene

como consecuencia que los límites geográficos de la red son difusos.

- *Sistema de Distribución:* El sistema de distribución es el componente lógico de 802.11 que se utiliza para reenviar los marcos a su destino. Si bien 802.11 no especifica ninguna tecnología en particular para implementar el sistema de distribución, generalmente solo se denomina *Red de Backbone*, y está formado por las conexiones *Ethernet* que unen los distintos *AP.*

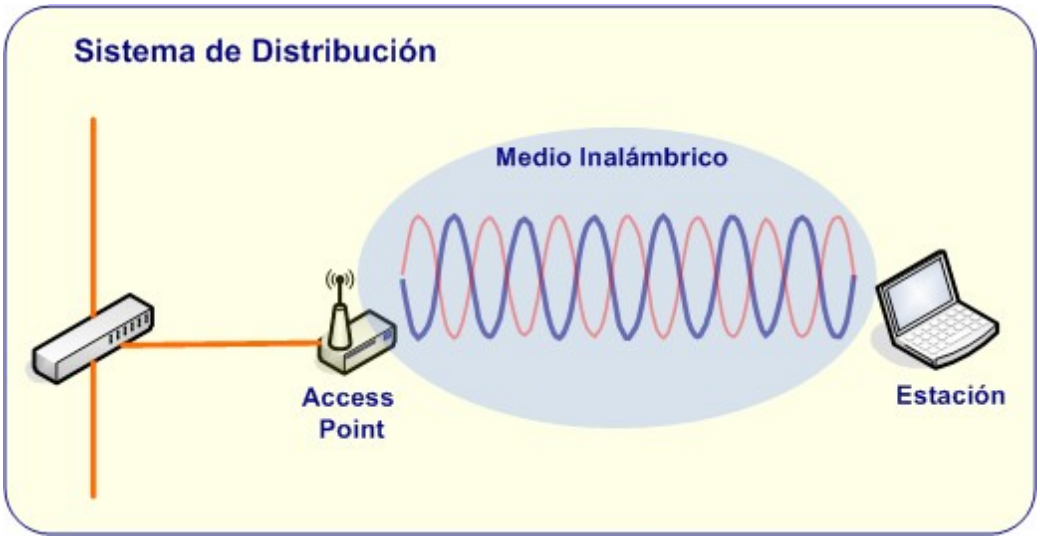

Ilustración III.2.: Componentes de la Infraestructura Inalámbrica.

2. BLOQUES LÓGICOS DE UNA RED INALÁMBRICA

1. CONJUNTO DE SERVICIOS BÁSICOS (BSS)

El bloque de construcción básico en una red 802.11 es el *Basic Service Set o Conjunto de Servicios Básicos (BSS)* que simplemente es

un grupo de estaciones que se comunican entre sí. La comunicación entre estas estaciones tiene lugar en un área difusa denominada *Basic Service Area o Área de Servicio Básico (BSA)* definida por las características de propagación del medio inalámbrico. Cuando una estación entra en la *BSA,* puede comunicarse con los otros miembros *del BSS.* Como se ve en la Ilustración III.3, los *BSS* pueden ser de dos tipos diferentes: Independientes o de Infraestructura.

En los *BSS* Independientes (*IBSS),* las estaciones se comunican directamente entre ellas y para ello es necesario que se encuentren dentro del alcance de radio. La red más pequeña posible es de dos estaciones. Típicamente este tipo de *BSS* se componen de un número pequeño de estaciones, que se configuran para un evento especial por un corto periodo de tiempo, es por ello que usualmente se denominan *ad hoc BSS* o *Redes ad hoc*.

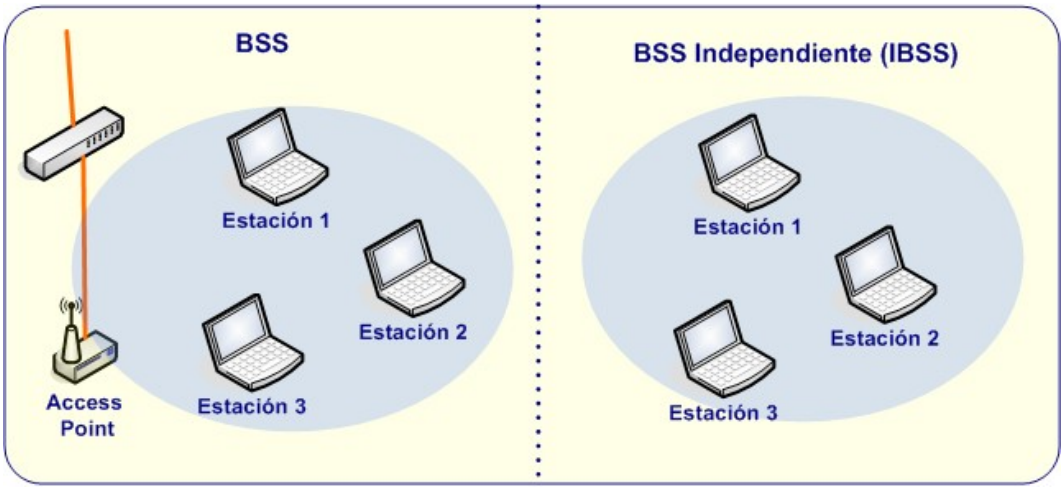

Ilustración III.3.: Comparación entre BSS de Infraestructura y BSS Independiente

En contraposición, los *BSS de Infraestructura*, también llamados *Redes de Infraestructura*, se distinguen por el uso de un *Access Point*

(*AP*) que es utilizado para realizar todas las comunicaciones dentro de una *BSA.* Cuando una estación envía un marco a otra estación, se la envía al *AP* y este reenvía el marco a la estación destino. De esta forma una *BSA* que corresponde a un *BSS de Infraestructura*, se define por todos los puntos en que las transmisiones del *AP* pueden recibirse. A pesar de que este tipo de transmisión consume más capacidad del medio que una transmisión directa a la estación destino, posee dos ventajas importantes:

- Un *BSS de infraestructura* se define por la distancia al *AP*. Si bien todas las estaciones deben estar en el rango de alcance del *AP*, no existen restricciones para la distancia entre las estaciones. El costo de usar un *IBSS* se determina por el incremento de la complejidad de la capa física, porque las estaciones deben mantener alcance con las estaciones vecinas dentro del *BSA.*

- Los *AP* están en posición de ayudar al ahorro de energía de las estaciones móviles, ya que pueden actuar como *buffers* de los marcos. Dichas estaciones, en modo de ahorro de energía, pueden apagar el transmisor y prenderlo sólo para transmitir y recibir los marcos almacenados en el *AP*.

En una red de infraestructura, las estaciones deben *asociarse* a un *AP* para obtener servicios de red. Esta operación es equivalente a enchufar un cable de red en una red cableada y es una acción exclusiva de las estaciones, que sólo pueden asociarse a un *AP .* No existe un límite de estaciones que pueden asociarse a un *AP.* Sin embargo, en la práctica, el bajo ancho de banda que presenta el medio de transmisión inalámbrico limita la cantidad de estaciones para una red inalámbrica.

2. CONJUNTOS DE SERVICIO EXTENDIDO (ESS)

Si bien los *BSS* pueden cubrir áreas de servicio pequeñas, como oficinas o casas, no pueden proveer de cobertura a áreas más grandes. Es por esto que el estándar 802.11 permite, a fin de tener cobertura en áreas extensas, definir un bloque mayor, denominado *Extended Service Set o Conjunto de Servicios Extendidos (ESS)*. Dicho bloque se crea al conectar varios *BSS* mediante una red troncal y al configurar todos los *AP* en el *ESS* con el mismo *Service Set Identifier o Identificador de Conjunto de Servicios (SSID)*, que sirve como "nombre de red" para los usuarios del servicio.

Las estaciones dentro del mismo *ESS* pueden comunicarse entre ellas, aún con las estaciones que están en un *BSS* diferente. Tanto el medio inalámbrico como el troncal actúan como un solo medio de capa de enlace.

Las *Áreas de Servicio Extendido o Extended Service Areas (ESA*) son el nivel de abstracción más alto soportado por las redes 802.11, los *AP* de un *ESS* interoperan para permitir al resto de la red ubicar a una estación por su *dirección física o MAC,* independientemente del *BSS* en el que esta estación esté ubicada. De este modo, el router de la Ilustración III.4 permanece ignorante de la localización física de la estación y depende de los *AP* para entregar el marco.

3. SERVICIOS DE RED

El estándar 802.11 no especifica ninguna tecnología en el troncal o *backbone* y sólo requiere que el mismo provea un grupo especificado de servicios. 802.11 provee nueve servicios. Sólo tres de estos se utilizan

para transportar datos. Los seis restantes son operaciones de administración que permiten a la red trazar a las estaciones móviles y entregar los marcos de red efectivamente.

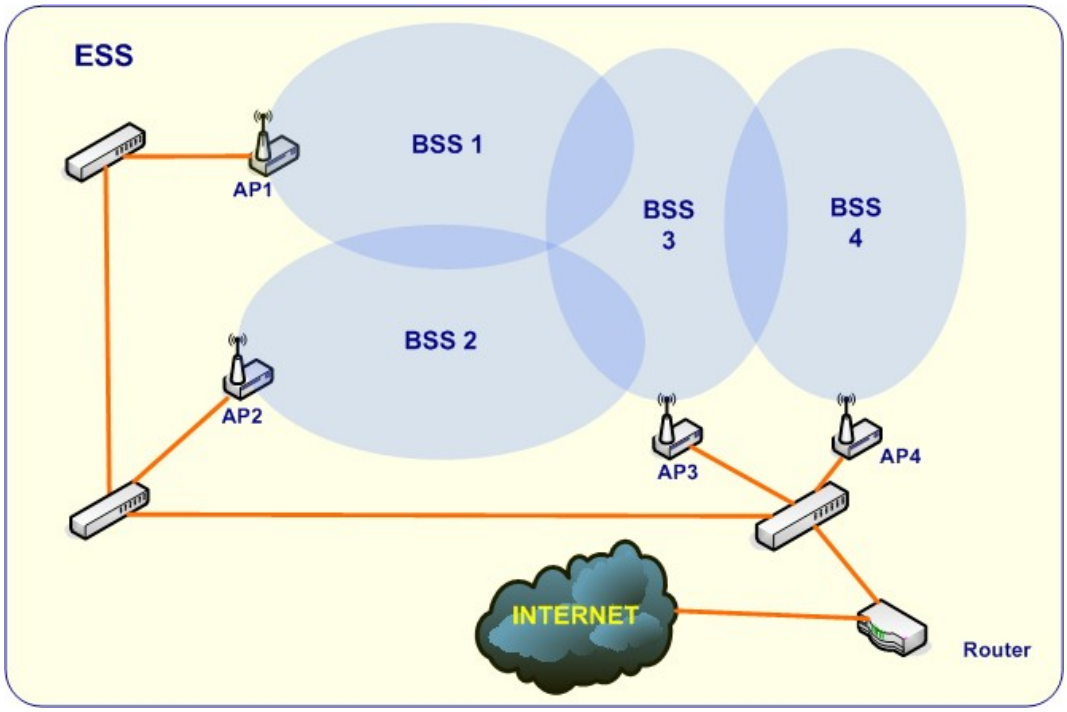

Ilustración III.4.: Esquema de un Extended Service Set (ESS).

- *Distribución:* Este servicio es utilizado por las estaciones móviles en una red de infraestructura cada vez que envían datos. Una vez que un marco es aceptado por un *AP*, este utiliza el servicio de distribución para reenviar este marco a su destino. Todas las comunicaciones que utilizan a un *AP* viajan a través del sistema de distribución, aun la comunicación entre dos estaciones asociadas al mismo *AP.*

- *Integración:* Este es un servicio provisto por el sistema de distribución que permite la conexión de el sistema de distribución a una red no IEEE802.11. Esta función es específica del sistema de distribución utilizado y no está especificada en el estándar 802.11, salvo en términos de descripción del servicio que se ofrece.

- *Asociación:* Este servicio requiere que las estaciones se registren o *asocien* al *AP* que provee un *BSS*. Gracias a esta asociación, es posible la comunicación. Y a su vez, el sistema de distribución puede determinar a cúal *AP* está asociada una estación. Las estaciones no asociadas, no pertenecen a la red. El estándar 802.11 especifica que esta función debe ser provista por el sistema de distribución utilizando los datos de asociación; pero no obliga a una implementación particular. Cuando se utilizan los protocolos *RSN*[1], la asociación es un precursor a la autenticación, es decir antes de completar la autenticación, un *AP*, descarta todo el tráfico de red de una estación.

- *Reasociación:* Este servicio permite que, cuando una estación se mueve entre dos *BSA* dentro de una *ESA*, pueda evaluar el nivel de la señal recibido y quizás cambiar la asociación de un *AP* a otro. La *Reasociación* es iniciada por las estaciones cuando las condiciones de la señal le indican que este cambio puede ser beneficioso. Nunca es iniciado por un *AP*, si bien existen *AP* que desconectan a las estaciones para obligarlas a reasociarse. Una vez que la reasociación está completada, el

1 *Robust Security Network*, Redes con seguridad robusta, ver 802.11i.

sistema de distribución actualiza los registros de localización para reflejar la trazabilidad de la estación en un *AP* diferente.

- *Desasociación:* Este servicio permite terminar con una asociación existente y es típicamente un servicio que utilizan las estaciones. Cuando una estación invoca al servicio, todos los datos de movilidad almacenados por el sistema de distribución son removidos. El uso de este servicio es “educado”, pero de todas maneras, la capa de enlace está diseñada para acomodarse a las estaciones que dejan la red sin desasociarse formalmente.

- *Autenticación:* Debido a que las redes inalámbricas no pueden ofrecer la seguridad física que ofrecen las redes cableadas, dependen de rutinas de autenticación adicionales que aseguran que los usuarios que acceden a la red están autorizados para hacerlo. La autenticación es un prerrequisito necesario para la asociación porque sólo los usuarios autenticados están autorizados a utilizar la red. La autenticación puede ocurrir varias veces durante el tiempo que una estación está conectada a una red inalámbrica. Antes de asociarse a un *AP*, una estación realiza un intercambio básico de identidades con el *AP* que consiste en su dirección *MAC.* Dicho intercambio es frecuentemente denominado “autenticación 802.11”, y es distinto a la autenticación de usuarios con criptografía robusta que se ejecuta después.

- *Desautenticación:* Este servicio termina con una relación de autenticación y, como efecto, termina con la asociación de la estación. En una red *RSN*, la desautenticación también borra la información referida a las claves.

- *Confidencialidad:* Este servicio tiene como fin el prevenir los ataques contra la privacidad de los datos en las comunicaciones. En las redes inalámbricas, es mucho más frecuente y difícil de prevenir una ataque de este tipo que en las redes cableadas. En efecto, el acceso no autorizado es dependiente de la seguridad física. En las revisiones iniciales de 802.11, el servicio de *confidencialidad* se denominó *privacidad*, y fue provisto por el protocolo *Privacidad Equivalente al Cableado o Wired Equivalent Privacy (WEP)*, ahora desacreditado. El estándar 802.11i, además de nuevos esquemas de cifrado, aumenta la confidencialidad dotando al servicio de autenticación basada en usuarios y servicios de administración de claves, dos factores críticos de los que *WEP* carecía.

- *Entrega de MSDU (MAC Service Data Unit):* Este servicio provee una de las capacidades vitales para una red: la de entregar los datos al destinatario. Las estaciones proveen el servicio de entrega *Unidades de Datos de Servicio MAC (MSDU*) el cual es responsable de entregar los datos al destinatario.

- *Control de Potencia de Transmisión*: Este servicio fue definido por 802.11h, ya que los estándares europeos para la banda de 5 GHz requieren que las estaciones controlen la potencia de las transmisiones de radio para evitar las interferencias con otros usuarios de esta banda de *RF.* La distancia máxima al *AP* está en función de la potencia transmitida. Controlando esta potencia, para emitir el mínimo necesario para transmitir, se evita la interferencia a otros usuarios o dispositivos.

- *Selección de Frecuencia Dinámica:* Debido a que algunos radares operan en la banda de *RF* de 5 Ghz, algunos marcos regulatorios han definido que las redes inalámbricas deben detectar a los sistemas de radar y cambiar a las frecuencias no utilizadas por el mismo. Otras marcos regulatorios requieren de una utilización uniforme de la banda de 5 GHz para redes, por lo que los dispositivos deben tener la capacidad de reubicar los canales de forma que su uso es ecualizado.

Servicio	Tipo de Servicio	Descripción
Distribución	Distribución	Servicio utilizado en el envío de marcos para determinar la dirección de destino en redes de infraestructura.
Integración	Distribución	Servicio utilizado al reenviar marcos a una red IEEE 802.3 fuera de la red inalámbrica.
Asociación	Distribución	Servicio utilizado para establecer el *AP* que servirá de *gateway* a una estación móvil en particular. Acción que obtiene como resultado la conexión de la estación a la red inalámbrica.
Reasociación	Distribución	Servicio usado para cambiar el *AP* que oficia de *gateway* a una estación móvil en particular.
Desasociación	Distribución	Servicio utilizado para terminar una conexión entre una red inalámbrica y una estación.
Autenticación	Estación	Servicio utilizado para establecer la identidad de la estación *(MAC)* antes de asociarse a la red. Llamada también "*Autenticación 802.11*". No confundir con la autenticación del servicio de confidencialidad.
Desautenticación	Estación	Servicio utilizado para finalizar una autenticación realizada y por lo tanto la asociación de una estación a la red inalámbrica.
Confidencialidad	Estación	Servicio cuyo objetivo es proveer protección

Servicio	Tipo de Servicio	Descripción
		contra la violación de la privacidad de las transmisiones.
Envío de MSDU	Estación	Servicio utilizado para enviar los marcos de datos al destinatario final.
Control de Potencia de Transmisión (TPC)	Estación / Administración de Espectro *RF*	Servicio usado para reducir la interferencia, minimizando la potencia de transmisión. (802.11h).
Selección de Frecuencia Dinámica (DFS)	Estación / Administración de Espectro *RF*	Servicio utilizado para evitar la interferencia con radares que operan en la banda de *RF* de 5 GHz. (802.11h).

Tabla III.3.: Resumen de los Servicios de Red Inalámbrica.

Los servicios de estación deben formar parte de cualquier estación que cumpla con el estándar 802.11. Dichos servicios son provistos tanto por estaciones móviles como por las interfaces inalámbricas de los *AP*. Para brindar algunos de estos servicios en posible que se requieran otros. Por ejemplo, las estaciones proveen el servicio de *entrega de MSDU* y pueden necesitar utilizar el servicio de *autenticación* para realizar una asociación.

Los servicios del sistema de distribución conectan a los *AP* con el sistema de distribución. El rol principal de los *AP* es extender los servicios de la red cableada a la red inalámbrica. Esto se logra a través de los servicios *distribución e integración*. Además para mantener los datos de asociación y la información de localización de la estaciones, el sistema de distribución brinda los servicios *de asociación, reasociación, y desasociación.*

4. SERVICIOS DE CONFIDENCIALIDAD Y CONTROL DE ACCESO

Confidencialidad, a veces denominada privacidad, es el término utilizado para describir que los datos están protegidos contra la intercepción de personas no *autorizadas. Integridad* significa que esos datos no han sido modificados en tránsito. *Autenticación* es el proceso mediante el cual se asegura la identidad del origen de los datos. La *autorización y control de acceso,* que gestionan el acceso a los recursos y las operaciones permitidas por parte de los usuarios autenticados, se implementan montados encima de la *autenticación.*

Ambos, privacidad e integridad, dependen de las claves de cifrado compartidas (*shared cryptographic keying).* Por lo tanto el servicio de *confidencialidad* depende necesariamente de otros servicios para proveer de *autenticación y administración de claves.*

Se aprecian a continuación los diferentes bloques que entran en el juego del servicio de confidencialidad.

- *Administración de claves y autenticación (AKM):* La integridad del cifrado no tiene valor alguno si no evita que los usuarios no autorizados tengan acceso a la red. El servicio de confidencialidad depende de la administración de claves y de la autenticación. Su función consiste en establecer la identidad de cada usuario y las claves de cifrado. La autenticación puede ser llevada a cabo por un protocolo externo, por ejemplo, 802.1X [IEEE8021X] o claves pre-compartidas (*pre-shared keys, PSK).*

- *Algoritmos de Cifrado:* La protección de los marcos puede ser realizada por el algoritmo *WEP,* utilizando claves de 40 o

104 bits, por el Protocolo de Integridad de Claves Temporales *o Temporal Key Integrity Protocol (*TKIP), o por el Protocolo CBC-MAC Modo Contador *o Counter Mode CBC-MAC Protocol (CCMP).*

- *Autenticidad del Origen:* A fin de evitar ataques de *suplantación de identidad*, TKIP y CCMP permiten al receptor validar la dirección *MAC* del origen. Dicha protección sólo es factible para datos tipo *Unicast*[1].

- *Detección de Repetición:* Tanto TKIP como CCMP, protegen contra los *ataques de repetición* incorporando un contador de secuencia que se valida una vez que se ha recibido el marco. Los marcos que son demasiado "viejos" para ser validados se descartarán.

- *Sistemas y Protocolos Externos:* El servicio de confidencialidad tiene una fuerte dependencia de protocolos externos para cumplir su objetivo. La administración de claves es proporcionada por el protocolo 802.1X que, junto con el protocolo EAP, transporta los datos de autenticación. Si bien 802.11 no posee restricciones en los protocolos que pueden utilizarse, las elecciones más comunes en las implementaciones son los protocolos EAP para la autenticación y RADIUS para comunicarse con el servidor de autenticación.

Ahora se describirá el funcionamiento de las distintas implementaciones del *Servicio de Confidencialidad* sobre redes 802.11.

1. WEP (WIRED EQUIVALENT PRIVACY)

1 *Unicast*, forma de transmitir en la cual el marco enviado sólo tiene un destinatario.

WEP fue inicialmente el servicio de confidencialidad de la especificación 802.11 original. Utiliza un conjunto preestablecido de claves (*PSK*) para cifrar cada marco 802.11 con el algoritmo para flujo de datos (*streams cipher) RC4.*

En líneas generales *WEP* requiere de los siguientes datos de entrada para realizar el cifrado:

- Los datos que el marco deberá transportar como carga o *payload.* Estos son provistos por la capa superior del stack de protocolos.

- Una clave secreta, utilizada para cifrar el marco. Dependiendo de la implementación, las claves pueden ser especificadas como una cadena de bits o por el número de clave, ya que *WEP* permite almacenar cuatro claves de forma simultánea.

- Un *Vector de Inicialización (IV),* utilizado junto con la clave secreta en la transmisión del marco.

Luego de realizar el proceso de cifrado, *WEP* ofrece como resultado:

Un marco cifrado, listo para transmitir sobre redes no confiables, con suficiente información en la cabecera para permitir su descifrado en el receptor del marco.

Como se ve en la Ilustración III.5, el driver y la interfaz de hardware son responsables de procesar los datos y de enviar el marco cifrado utilizando la siguiente secuencia:

1. El marco 802.11 entra en la cola de transmisión. Este marco consiste en una cabecera y en el *payload. WEP* protege solamente el *payload* y deja la cabecera intacta.

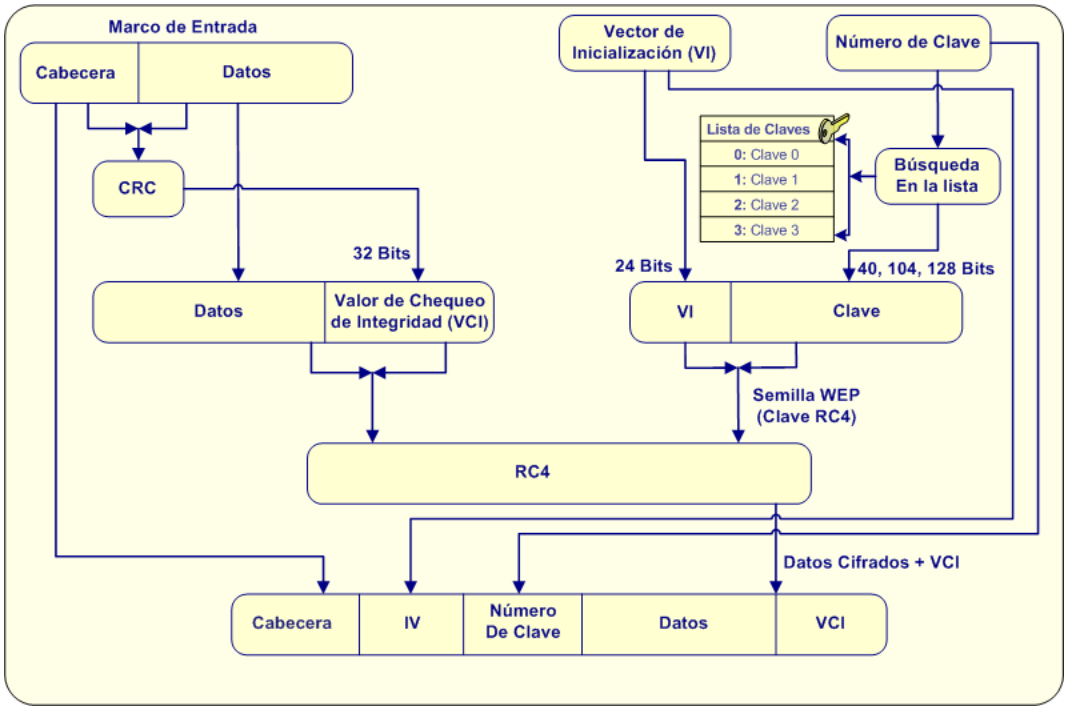

Ilustración III.5.: Operaciones que lleva a cabo WEP

2. Se calcula un V*alor de Chequeo de Integridad (ICV)* sobre el *payload* original. EL *Chequeo de Secuencia del Marco (FCS)* no ha sido calculado en este punto, por lo que no se incluye en el cálculo del *ICV.* El *ICV* es un *CRC*[1], y por lo tanto es predecible y *criptográficamente inseguro* para chequeos de integridad.

1 *Cyclic Redundancy Check,* Chequeo de Redundancia Cíclico.

3. La *Clave de Cifrado del Marco (FEK) o WEP seed* se ensambla en este punto, esta consta de dos partes: la *clave secreta* y el v*ector de inicialización (IV)*. Los *stream ciphers* producen el mismo flujo de salida para la misma clave, por lo que el *IV* se utiliza para producir flujos de salida distintos para cada marco transmitido y se coloca como prefijo de la clave secreta. El estándar 802.11 no establece ninguna restricción para el algoritmo usado para elegir los *IV.* Algunas implementaciones asignan los *IV* de forma secuencial, otras lo asignan a través de un algoritmo pseudoaleatorio. La selección del *IV* tiene algunas implicaciones de seguridad, ya que una "pobre" selección en el *IV* puede comprometer la clave.

4. La *Clave de Cifrado del Marco* se usa como clave *RC4* para cifrar el *payload* del marco original del paso 1 y el *ICV* del paso 2. El proceso de cifrado casi siempre es realizado con hardware especializado para el algoritmo *RC4* en la placa inalámbrica.

5. Como último paso, se ensambla el marco a transmitir formado por el *payload cifrado,* la cabecera original, y una cabecera *WEP,* formada por el *IV y número de clave*. Una vez ensamblado el nuevo marco se calcula el *FCS* y se realiza la transmisión. El descifrado del marco se realiza en orden inverso.

WEP ha demostrado tener graves falencias que lo hacen poco confiable como *servicio de* confidencialidad, algunas de estas falencias se describen continuación [CHAN00]:

1. No provee ningún mecanismo para establecer claves sobre un medio inseguro, por lo que las claves son compartidas por todas las estaciones en un *BBS* y a veces entre varios *BBS.*

2. Utiliza un algoritmo de cifrado sincrónico sobre un medio en el que es muy difícil asegurar la sincronización durante una sesión completa.

3. Usa una clave por marco transmitido, concatenado el *IV* directamente con la *PSK* para producir una clave para *RC4* para resolver el problema anteriormente planteado. Esto expone la clave maestra a ataques del tipo *FMS (Fluhrer-Mantin-Shamir).*

4. Es muy limitado el espacio de claves debido a que la clave maestra es configurada de forma manual y es estática, y a que el *IV* sólo tiene 24 bits de largo. A causa de esto en una red con tráfico medio se produce repetición de claves cada aproximadamente 5 horas.

5. La reutilización de claves es altamente probable, debido a que 802.11 especifica que el recambio del *IV* es opcional.

6. El *ICV* se genera con un algoritmo CRC-32 que es lineal.

7. El *ICV* no protege la integridad de la cabecera 802.11, permitiendo ataques de redirección.

8. No tiene protección contra ataques de repetición.

9. No posee soporte para que una estación autentique a la red.

En general la principal deficiencia de *WEP*, es que su diseño intenta adaptar *RC4* a un ambiente para el cual no fue diseñado.

2. 802.11i Y WPA (WIRELESS PROTECTED ACCESS)

Cuando los problemas en la seguridad de *WEP* fueron expuestos, la IEEE formó un grupo de tareas: 802.11i con la tarea de mejorar la seguridad en las redes 802.11. Este grupo propuso las *Redes con Seguridad Robusta o Robust Security Network (RSN),* esto es una red que implementa las propuestas de seguridad de 802.11i y permite sólo dispositivos capaces de cumplir con estas propuestas.

Debido a que la transición a *RSN* no siempre es factible como un proceso inmediato, 802.11i permite la implementación de *Redes con Seguridad de Transición o Transitional Security Networks (TSN)*, que pueden aceptar la existencia de estaciones *RSN y WEP* en una red 802.11i. Como su nombre lo indica estas redes deberían moverse finalmente, a un esquema de *RSN*.

La propuesta de seguridad especificada por el estándar, usa el *Estándar Avanzado de Cifrado o Advanced Encryption Standard (AES)* como algoritmo de cifrado. El problema que se plantea es que *AES* no es compatible con el hardware existente, debido que requiere de un poderoso soporte computacional en las placas de red. Ante a la presión que esto generó en los fabricantes de equipos, la *Wi-Fi Alliance*, entró en el juego.

La *Wi-Fi Alliance*, está formada por fabricantes de 802.11 y tiene por propósito asegurar la interoperabilidad entre los equipos de distintas marcas. A fin de mejorar la seguridad en las redes inalámbricas sin obligar a un cambio en el hardware inalámbrico, esta asociación pidió como requisito para certificar los productos, que estos adoptaran como estándar el protocolo *TKIP*. Estas especificaciones, pre 802.11i, se conocen como *Wi-Fi Protected Access (WPA).*

WPA es básicamente un subconjunto de las normas 802.11i, que incluye las arquitecturas de autenticación y administración de claves (802.1X) especificadas en 802.11i, también llamado *WPA2*, para evitar confusiones. A diferencia de 802.11i, que utiliza *AES* para proveer confidencialidad e integridad, *WPA* utiliza *TKIP y MICHAEL*, respectivamente, para realizar estas funciones.

Uno de los logros de 802.11i en la posibilidad de implementarlo en dos ambientes totalmente diferentes: una red *SOHO*[1] o una red empresarial. Estos ambientes difieren tanto en los requerimientos de seguridad, como en las capacidades de la infraestructura que poseen para poder proveer la seguridad.

Para las redes empresariales, se especifica el uso de *IEEE802.1X* para intercambio de claves y autenticación. Esto requiere un servidor de autenticación separado. Para los ambientes *SOHO*, en los cuales no siempre es posible contar con este tipo de infraestructura, 802.11i permite el uso de *PSK.*

1. 802.1X Y EAP

1 *Small Office or Home Office,* una oficina pequeña o una oficina hogareña.

802.1X [IEEE8021X] está basado en el *Protocolo de Autenticación Extensible (EAP) [EAP]. EAP* es una estructura de protocolo. Que, en vez de especificar cómo autenticar a los usuarios, permite a los diseñadores de protocolos construir sus *propios métodos EAP,* que son subprotocolos que ejecutan las transacciones de autenticación. Los *métodos EAP* pueden tener diferentes objetivos, y por lo tanto, pueden utilizarse métodos diferentes dependiendo de las necesidades del caso. *EAP* es una simple encapsulación que puede montarse sobre cualquier capa de enlace, si bien ha sido montada usualmente sobre vínculos *PPP [PPP]* [1].

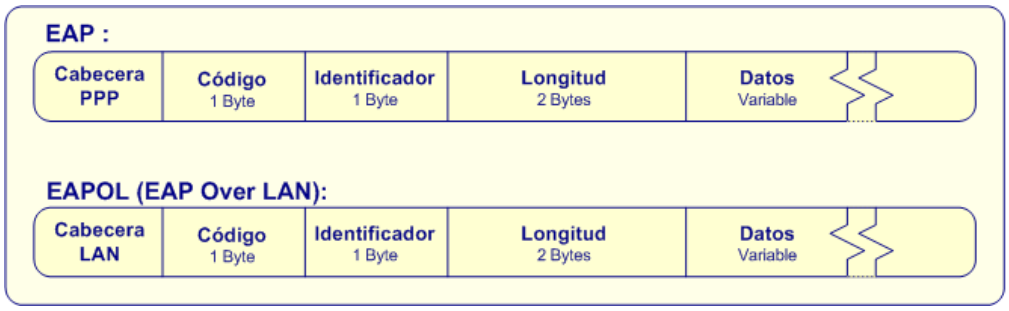

Ilustración III.6.: Formato genérico de los paquetes EAP.

La Ilustración III.6 muestra los formatos genéricos de los paquetes *EAP* transportados en vínculos *PPP* o en otro tipo de marco. Cuando se transporta *EAP* en un vínculo *PPP,* se utiliza el número de protocolo *0x*C227[2]. Los campos en cada paquete son los que figuran en la Tabla III.4.

Nombre del Campo	Significado del Campo
Código	Este campo tiene una longitud de 1 byte e identifica el tipo de paquete *EAP*, se usa para interpretar el campo *Datos* del paquete. Los valores asignados son: 1 (Solicitud), 2

1 *Point to Point Protocol, p*rotocolo punto a punto.

2 Cada vez que se haga referencia a un número en base hexadecimal se usará el prefijo *0x*.

Nombre del Campo	Significado del Campo
	(Respuesta), 3 (Éxito), 4 (Falla), otro valor produce que el paquete sea descartado.
Identificador	Este campo tiene una longitud de 1 byte y contiene un dato de tipo entero sin signo. Se usa para realizar las correspondencias entre las peticiones y las respuestas. Las retransmisiones utilizan el mismo *identificador*, y las transmisiones nuevas usan un *nuevo identificador.*
Longitud	Este campo posee una longitud de 2 bytes, y es el número de bytes de longitud de todo el paquete, incluyendo la cabecera. Si bien en algunos protocolos de enlace se puede requerir el relleno, *EAP* asume que cualquier exceso a la longitud declarada por este campo es relleno y puede ignorarse.
Datos	Este campo posee longitud variable, y dependiendo del tipo de paquete, puede tener una longitud nula. La interpretación de este campo se basa en el valor del campo *Código.*

Tabla III.4.: Descripción de los campos del paquete EAP.

Las operaciones que se llevan a cabo con *EAP* están compuestas de paquetes con solicitudes y respuestas, y en estos casos, el valor del campo *Código* tendrá valores 1 o 2 respectivamente. En esos paquetes el campo *Datos* se descompone en dos subcampos: *Tipo y Datos del Tipo,* como se ve en la Ilustración III.7 y en la Tabla III.5.

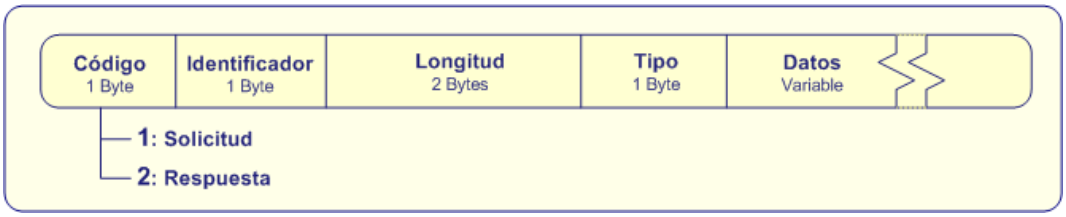

Ilustración III.7.: Formato de Paquetes EAP: Solicitud y Respuesta.

Nombre del Subcampo	Significado del Subcampo
Tipo	Este campo tiene una longitud de 1 byte que indica el tipo de solicitud o respuesta. Sólo un valor de tipo se usa en cada paquete, y el tipo de una respuesta es el mismo que el de una solicitud, a excepción del caso en que una solicitud es inaceptable y la respuesta puede tener valor 3 (*NAK)* para sugerir otro tipo alternativo. Los valores superiores a 3 indican métodos de autenticación.
Datos del Tipo	Este campo tiene longitud variable y debe ser interpretado de acuerdo a las reglas para cada valor de tipo.

Tabla III.5.: Descripción de los sub campos

Dentro de los intercambios *Solicitud/Respuesta*, el campo *Tipo* define la estructura que tendrá un paquete *EAP.* Los primeros tres valores definidos para este campo se consideran *códigos de tipos especiales*. El resto de los códigos definidos definen el intercambio de autenticación, es decir, definen *el método EAP* que se utilizará en el proceso de autenticación.

Todas la implementaciones de *EAP deben* soportar los códigos de 1 a 4, y deberían soportar el código 254. En la Tabla III.6 se puede observar un listado de códigos de uso común para el campo tipo.

Código	Tipo	Descripción
1	*Identidad*	Este código se usa para consultar la identidad del par a autenticar. Usualmente el autenticador envía este código en la solicitud inicial. Se recomienda usar este código con el propósito de seleccionar el *método EAP.* Por ello, los métodos deberían incluir un mecanismo para determinar la identidad del par y no confiar en la respuesta a este código.
2	*Notificación*	Este código se usa para enviar un mensaje de naturaleza imperativa al usuario. El par a autenticar, debería mostrar este mensaje al usuario. No cambia el estado de la autenticación y no es un indicador de

Código	Tipo	Descripción
		error. Por ejemplo puede usarse para notificar que una clave está a punto de expirar o la razón del bloqueo de una cuenta. Comúnmente no son usados en 802.1X.
3	*NAK*	Este código sólo puede usarse para *Respuestas.* Se envía en el caso de que el tipo de autenticación deseada sea inaceptable. La respuesta contiene uno o más tipos de autenticación deseados por el par. En caso de que el tipo sea 0 , se indica de que no hay alternativas viables para realizar la autenticación.
4	Desafío MD5	Autenticación al estilo CHAP, pero con algoritmo MD5.
5	OTP	Está diseñado para brindar soporte a contraseñas de un sólo uso a sistemas solamente. (RFC2289 y RFC2243).
6	GTC	Tarjeta de símbolo genérico. Método originalmente desarrollado para usar con tarjetas de símbolos como la RSA SecureID.
13	EAP-TLS	Autenticación mutua con certificados digitales. [EAP-TLS]
18	EAP-SIM	Autenticación por teléfono celular a través del módulo de Identidad del suscriptor (SIM)
21	EAP-TTLS	TLS por túnel; protege métodos de autenticación más débiles con cifrado TLS.
25	PEAP	*EAP Protegido;* protege métodos *EAP* más débiles con cifrado TLS.
29	MS-CHAP-V2	Autenticación con contraseña cifrada de Microsoft; compatible con dominios Windows. Generalmente se usa como método de autenticación interno a un túnel protegido protegido por otro método, como TTLS, TLS o PEAP.
254	Tipos Expandidos	Este código se usa para permitir a los fabricantes la utilización de sus propios códigos de tipo que no son adecuados para uso general. También permite expandir los métodos a más de 255 códigos. Esto se debe a que muchas de las implementaciones de *EAP* son específicas de cada fabricante.
255	Uso Experimental	Este código no tiene formato o contenido específico y se debería usar para experimentar nuevos tipos de *EAP*.

Tabla III.6.: Códigos de tipo de uso común en EAP.

Como resultado de los intercambios *EAP*, el usuario ha sido autenticado con éxito o ha fallado la autenticación. Una vez que el autenticador determina que el intercambio está completo, envía un paquete con el código 3 (operación exitosa), o con el código 4 (operación fallida). En ambos casos la longitud del paquete es de 4 bytes y tiene el formato que ve en la Ilustración III.8

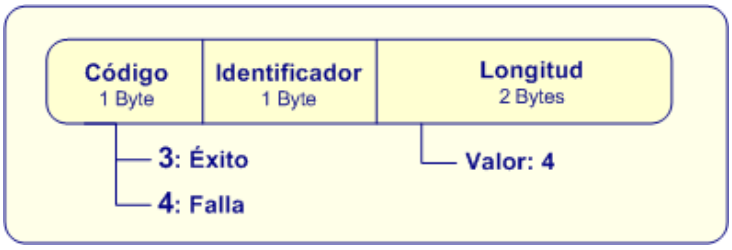

Ilustración III.8.: Formato del paquete EAP Éxito-Falla.

La capacidad de *extender* los métodos del protocolo *EAP*, son los una de las grandes fortalezas de este protocolo, ya que permite diseñar nuevos métodos para cumplir con nuevas necesidades.

Al seleccionar un método *EAP*, se deben tener en cuenta las capacidades del sistema que realizará la autenticación. Si bien los primeros métodos diseñados tenían el foco sobre todo en proporcionar un canal de comunicaciones para contactar al servidor de autenticación, los métodos diseñados para autenticar a los usuarios de redes inalámbricas deben, además, cumplir con tres importantes objetivos, como se expresa en la Tabla III.7.

Objetivos	Descripción
Protección de Cifrado Fuerte de las Credenciales de Usuario	Por definición, las redes inalámbricas deberían considerarse redes abiertas. El tráfico de datos sobre estas redes debe protegerse mediante cifrado para poder considerarlas "redes seguras". La mayoría de los métodos *EAP* diseñados para redes inalámbricas usan *TLS* para proveer protección de cifrado de las

Objetivos	Descripción
	credenciales de usuario.
Autenticación Mutua	El primer protocolo 802.11 consideraba *autenticación*, a la realizada por el *AP* para asociar a las estaciones. Debido a la posibilidad de sufrir un ataque de *Falsificación de AP*[1], adicionalmente a la autenticación de los usuarios, los mismos deberían tener la capacidad de autenticar o validar la identidad de la red a la que se conectan.
Derivación de Claves	Las funciones de *derivación de* claves, se usan frecuentemente junto a parámetros no secretos para derivar una o más claves desde un valor secreto. Esto permite que los protocolos que utilizan cifrado fuerte, usen claves dinámicas derivadas, que pueden distribuirse sin distribuir la clave principal.

Tabla III.7.: Objetivos de los métodos de autenticación en redes inalámbricas.

A continuación se presentan algunos de los métodos EAP que se utilizan comúnmente en ambientes inalámbricos.

1. EAP-TLS

EAP-TLS [EAP-TLS] usa *Seguridad de la Capa de Transporte o Transport Layer Security (TLS)* como método de cifrado. *TLS* es considerado el sucesor de *SSL*[2] y fue diseñado para ser usado enlaces que pueden ser susceptibles de interceptación. *TLS* provee *autenticación mutua* a través de intercambio de certificados digitales. Los clientes deben enviar un certificado digital al servidor de autenticación para validarse y el servidor de autenticación debe enviar un certificado al cliente.

---

1 *Falsificación de AP:* Ataque realizado por un AP externo a la infraestructura propia, que ha configurado el mismo *ESSID*, para engañar a los usuarios y robar sus identidades cuando se conectan a él pensando que se conecta a la red propia. También son llamados *Rogue AP.*

2 *Secure Socket Layer,* Capa de "Conexión" Segura.

Validando el certificado enviado por el servidor de autenticación contra una lista de autoridades de certificación, el cliente asegura que su conexión es a una red autorizada por la autoridad certificante y viceversa.

El método *EAP-TLS*, fue el primer método de autenticación que cumplió con los tres objetivos mencionados anteriormente en la Tabla III.7. La *autenticación mutua* evita los ataques de *falsificación de AP* y *TLS* usa una clave maestra que puede ser utilizada para *derivar claves* que usan los protocolos de seguridad de capa de enlace.

EAP-TLS no ha sido utilizado en forma masiva debido a la gran complicación que significa el montar una *Infraestructura de Claves Públicas (PKI)*, para las organizaciones que no la poseen. Esto obliga a la generación y distribución de certificados a todos los usuarios, es por ello que muchas organizaciones utilizan métodos alternativos.

La mayoría de las organizaciones prefiere utilizar sistemas de autenticación existentes, por ejemplo directorios *LDAP*, *Active Directory*, *Dominios Windows,* etc. De esta forma se evita crear un sistema de autenticación paralelo.

2. EAP-TTLS Y EAP-PEAP

Tanto *EAP-TTLS [EAP-TTLS]* como *EAP-PEAP [PEAP],* usan *certificados digitales* para autenticar la red[1] hacia al cliente; pero, a diferencia de *EAP-TLS,* no usan *certificados* para autenticar al cliente hacia la red. En lugar de *certificados digitales,* los clientes se autentican al servidor con métodos basados en contraseñas. Esto permite utilizar

1 La red o el servidor en este contexto se consideran sinónimos.

sistemas de autenticación existentes en la organización y reduce notablemente la cantidad de certificados que se necesita emitir.

Ambos métodos trabajan de de forma similar. Primero establecen un *túnel* TLS entre el cliente y el servidor de forma similar a *EAP-TLS*. Para ello utilizan el *certificado digital* que autentica al servidor, antes de proceder a la segunda fase.

Una vez completada satisfactoriamente la primera fase, usan el *túnel TLS* como canal de comunicaciones cifrado para realizar el proceso de autenticación del usuario con un método de autenticación basado en contraseñas. Frecuentemente se llama *autenticación externa* a la primera fase y *autenticación interna* a la segunda fase.

La diferencia entre *EAP-TTLS* y *EAP-PEAP* es la forma en que se realiza el proceso de *autenticación interna*. *EAP-TTLS* usa *el túnel* para intercambiar pares de atributo-valor, mientras que *EAP-PEAP* utiliza el *túnel* para iniciar otra sesión *EAP*. *EAP-TTLS* es más flexible porque al usar pares atributo-valor puede ejecutar métodos de autenticación que no están implementados con *EAP.*

Una de las ventajas de utilizar *EAP-TTLS o EAP-PEAP* es que las *autenticaciones internas y externas* pueden usar diferentes nombres de usuario. Esto evita revelar el nombre de usuario en los marcos enviados sin cifrado si se utiliza un nombre de usuario anónimo para el proceso de *autenticación externa*. No todos los clientes de software soportan este comportamiento.

3. FUNCIONAMIENTO DE 802.1X

802.1X provee un marco para realizar autenticación de usuarios sobre cualquier tipo de LAN. Controla el acceso a la red basándose en habilitar y deshabilitar un puerto físico de red[1], este tipo de autenticación se denomina *port based*. Define tres componentes, el *supplicant*, el *autenticador y el servidor de autenticación,* como se observa en la Ilustración III.9.

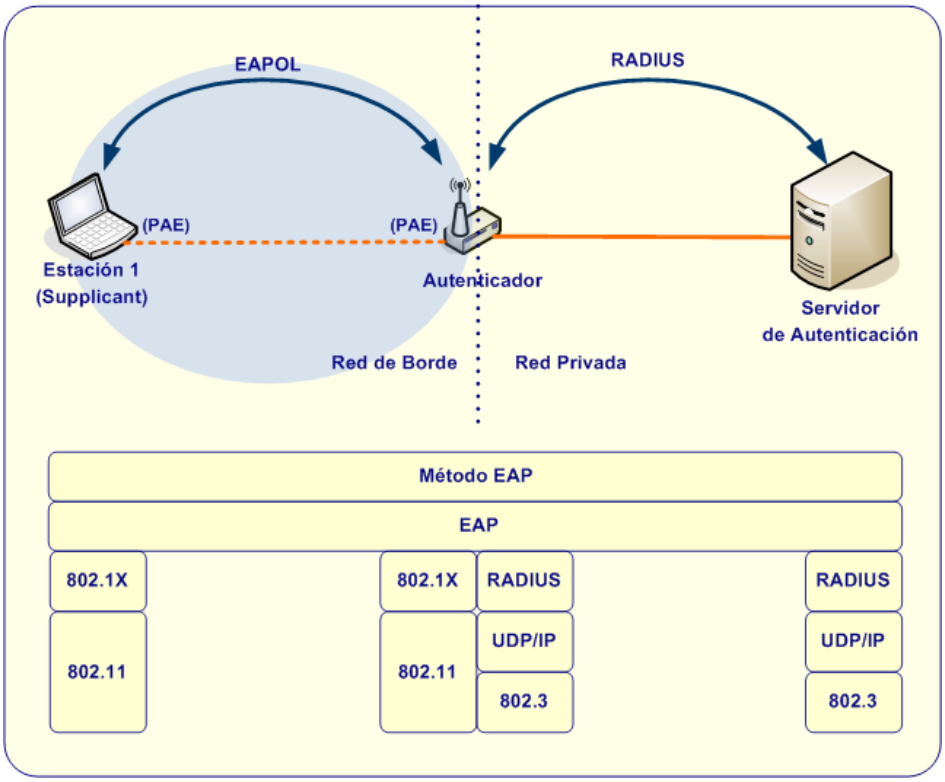

Ilustración III.9.: Componentes de 802.1X

El *supplicant* y el *autenticador* se consideran *Entidades de Autenticación por Puerto (PAEs)* dentro de la especificación. *802.1X* evita el acceso a la red solicitando al usuario iniciar el proceso de autenticación *antes* de permitir paso de tráfico de datos. Los puertos no autorizados

1 Típicamente se refiere a un puerto de un switch.

están restringidos a enviar sólo marcos de autenticación *EAPOL (EAP over LAN).*

Se considera "puerto*" 802.1X* en redes inalámbricas a la asociación entre un dispositivo inalámbrico y el *AP*. Una vez que se completa el proceso de asociación, se reporta a la máquina de estados de *802.1X* y la estación puede iniciar el proceso de autenticación. Una vez que este proceso se lleva a cabo de forma exitosa, incluido el intercambio de claves, la estación puede iniciar el tráfico de datos.

La principal diferencia de un intercambio EAPOL con un intercambio EAP, es que los *supplicants* pueden enviar un marco *EAPOL-Start* para disparar el proceso de intercambio y pueden enviar un marco *EAPOL-Logoff* para desautorizar un puerto.

En la Ilustración III.10 se puede observar el proceso de de un intercambio *EAPOL* en *IEEE802.11.* En el mismo se lleva a cabo una autenticación exitosa a través de los siguientes pasos:

1. El *supplicant* se asocia a la red 802.11, este proceso es un simple intercambio de dos marcos.

2. El *supplicant* inicia el intercambio *802.1X* con un mensaje *EAPOL-Start*. Este paso es opcional.

3. Se inicia un intercambio *EAP.* El *Autenticador* envía un mensaje *EAP-Request/Identity*. Este mensaje se puede enviar sin necesidad de recibir primero un mensaje *EAPOL-Start* si el *AP* sólo permite reenvío de marcos autenticados.

4. El *supplicant* responde con un mensaje *EAP-Response/Identity,* que el *Autenticador* reenvía al *servidor RADIUS* como solicitud de acceso.

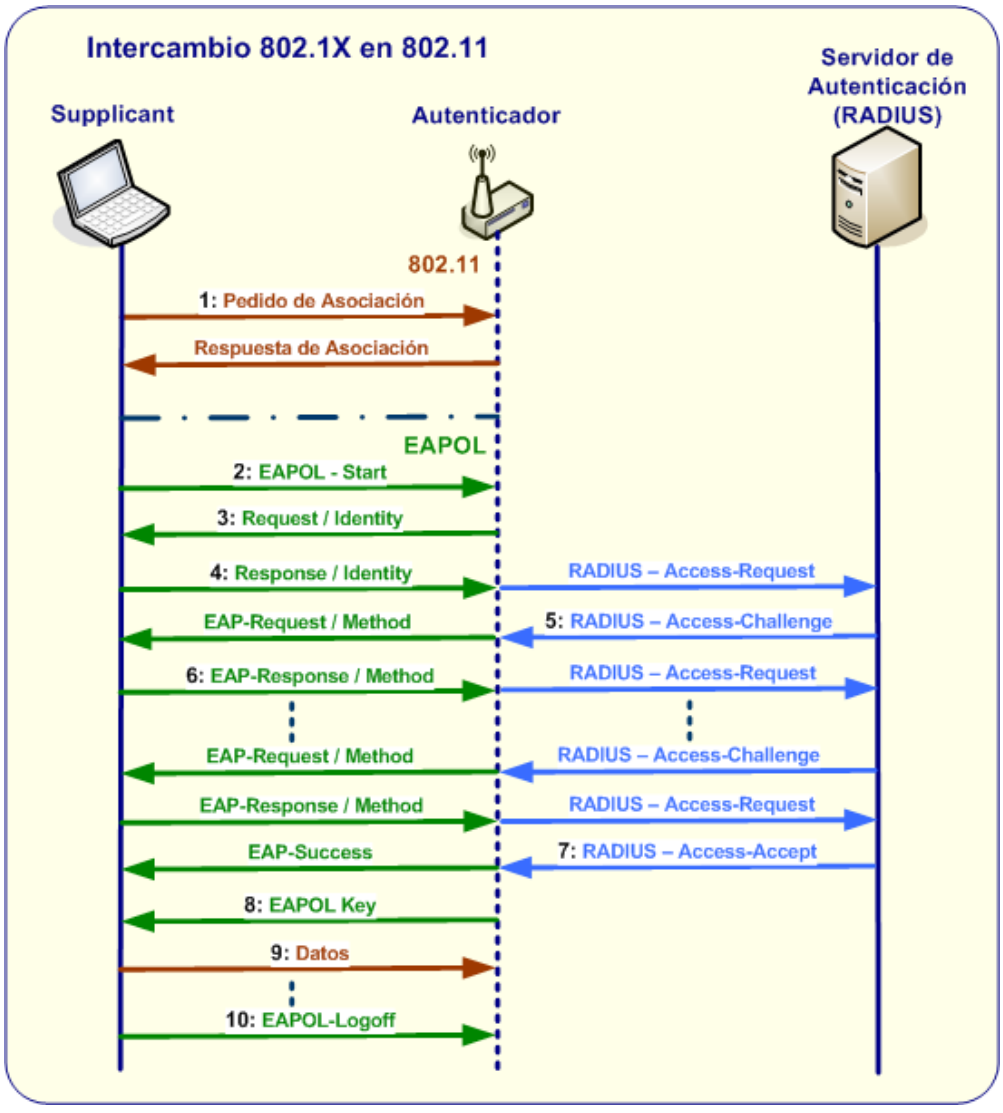

Ilustración III.10.: Intercambio 802.1X en 802.11. Proceso de Autenticación.

5. El servidor RADIUS determina el tipo de autenticación que se requiere y envía un mensaje *EAP-Request* para el método requerido. Por ejemplo si el método requerido fuese *EAP-PEAP*, el

mensaje sería *EAP-Request/PEAP.* Dicho mensaje se encapsula en un mensaje *RADIUS-Access-Challenge* y se envía al *AP,* que a su vez lo reenvía al *supplicant.*

6. El *supplicant* toma las credenciales suministradas por el usuario y devuelve un mensaje *EAP-Response*. El *Autenticador* traduce esta respuesta en un mensaje *RADIUS-Access-Request* con la respuesta al desafío del paso 5 en el campo de datos. Los pasos 5 y 6 pueden repetirse tantas veces como sea necesario para completar el proceso de autenticación. Si el *método EAP* utiliza intercambio de certificados es posible que se repitan entre 10 y 20 veces.

7. El S*ervidor de Autenticación* concede acceso con un mensaje *RADIUS-Access-Accept*, que el *Autenticador* traduce como mensaje *EAP-Success.* En este punto del proceso, el *Autenticador autoriza* el "puerto".

8. Inmediatamente después de recibir el mensaje *RADIUS-Access-Accept* el *AP* distribuye las claves al *supplicant* utilizando mensajes *EAPOL-Key.*

9. El *supplicant* instala las claves y comienza a enviar marcos de datos a la red. Es en esta etapa donde se lleva a cabo la configuración *DHCP.*

10. Cuando el *supplicant* finaliza el acceso a la red, envía un mensaje *EAP-Logoff* y el *Autenticador* cambia el estado del puerto a *no-autorizado.*

Para evitar el compromiso de las claves, los marcos de intercambio de las mismas se envían sólo si el proceso de autenticación tiene éxito. Los mensajes *EAPOL-Key* se utilizan para actualizar las claves de forma periódica.

2. CONFIDENCIALIDAD E INTEGRIDAD EN 802.11I

La mayor diferencia entre *WPA* y *802.11i (WPA2)* es que mientras *WPA2* utiliza *AES* para proveer confidencialidad e integridad, *WPA* utiliza *TKIP* y *MICHAEL* respectivamente.

WPA extiende los dos niveles de jerarquía de las claves utilizadas por WEP. En efecto*, WEP* utiliza una *clave maestra* (*MK*) para autenticación y para calcular las claves de cada paquete, al concatenar la *MK* con el *vector de inicialización (IV).*

802.11i utiliza la *MK,* también denominada *Pair-wise Master Key (PMK),* proporcionada por el proceso de autenticación realizado por *802.1X o* por la configuración manual de una *Preshared Secret Key (PSK).* A partir de la *PMK*, se derivan un grupo de claves transitorias o de *sesión*, denominadas *Pair-wise Transient Keys (PTK)* y a partir de las *PTK,* a través de una función de mezcla de claves, se generan las claves para cada paquete.

Además, se utiliza un grupo de claves diferentes para las comunicaciones tipo *broadcast y multicast*. El *autenticador* mantiene una *Group Master Key (GMK)* para derivar las claces transitorias de grupo o *Group Transient Keys (GTK).* No se puede derivar *GMK* a partir de las claves que utiliza cada estación para el tráfico *unicast.* Las jerarquías de claves se pueden apreciar en las Ilustraciones III.11 y III.12.

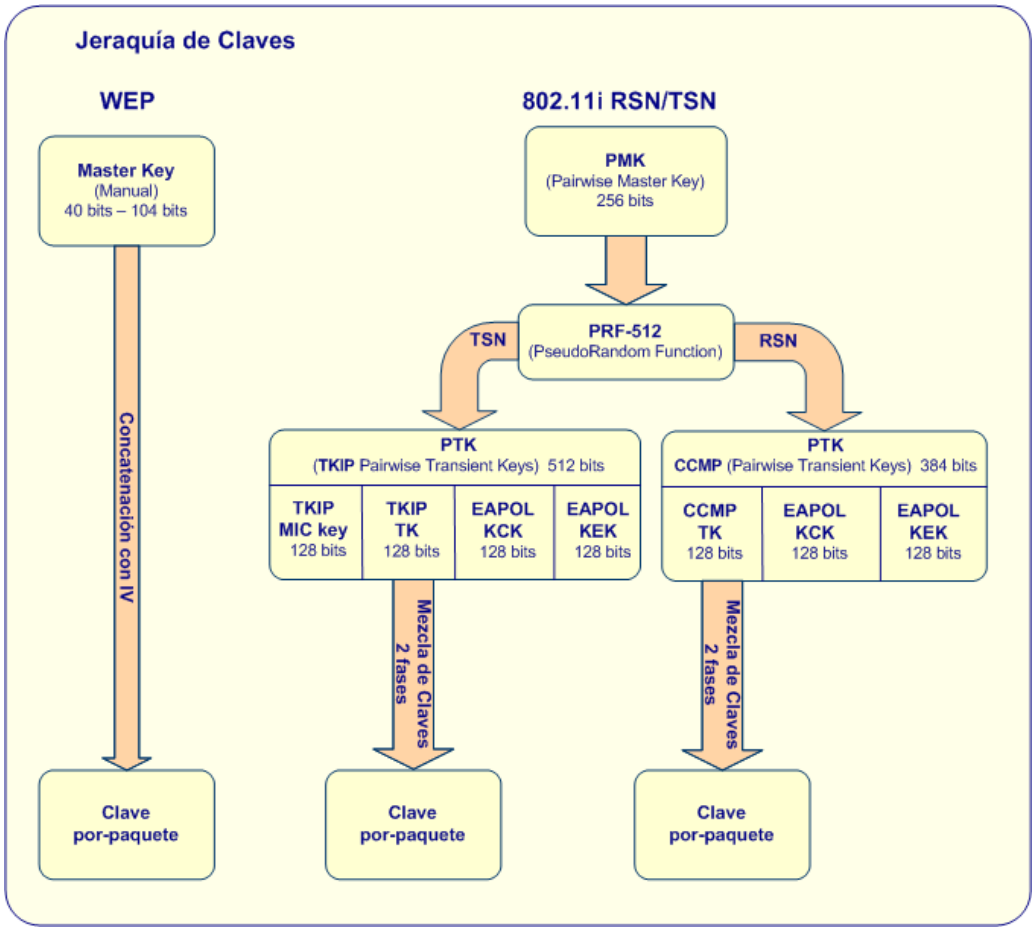

Ilustración III.11.: Jerarquía de claves 802.11i comparada con WEP.

El algoritmo *Pseudo-aleatorio(PRF)* que se utiliza para realizar la derivación de las claves transitorias desde la *PMK* necesita de cuatro argumentos, las direcciones *MAC* tanto de la estación, como del *autenticador* y dos valores denominados *SNonce y ANonce,* uno suministrado por la estación y otro suministrado por el *autenticador.* Nonce, es un acrónimo de *Number-Once (Número una vez*), es un valor

arbitrario que no puede repetirse nunca, y permite diferenciar dos sesiones diferentes.

Como la *PMK* y los argumentos necesarios del *PRF* son compartidos por la estación y el *AP,* ambos realizar la derivación de claves y obtener la *PTK*.

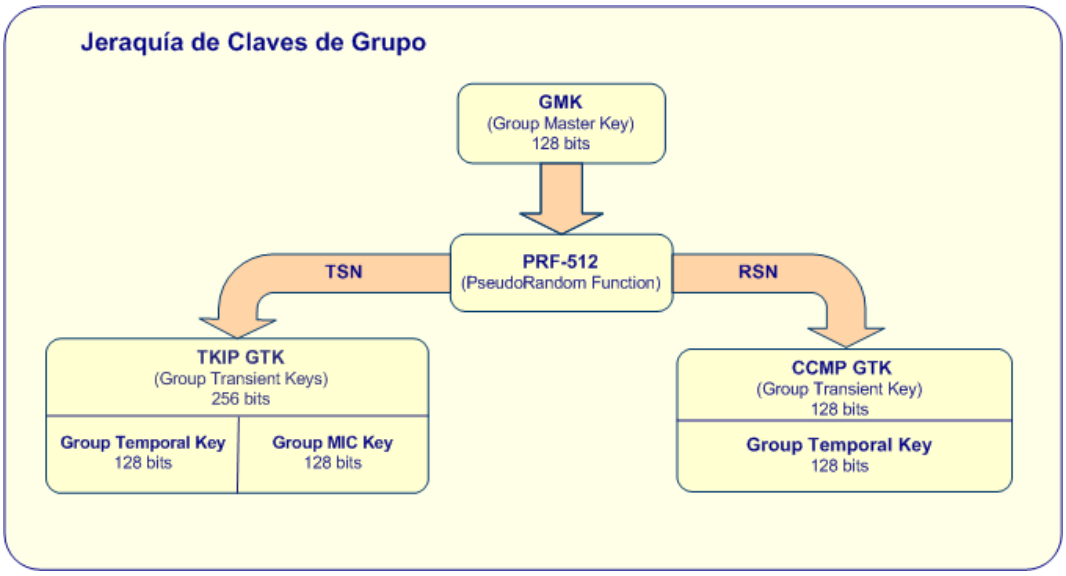

Ilustración III.12.: Jerarquía de claves de grupo de 802.11i.

El proceso de *TKIP y MICHAEL* para derivar las claves de cifrado de cada paquete se puede observar en la Ilustración III.13. Los detalles importantes de este proceso son:

1. Se utiliza un *contador de secuencia o IV* de 48 bits.

2. El tamaño de la clave de cifrado es de 104 bits para mantener compatibilidad con el hardware con soporte *WEP.*

3. Para disminuir la carga de proceso en la placa de red, se divide el proceso de generación de claves en dos fases. La primera fase requiere de mayor potencia de cálculo que la segunda.

4. Al involucrar los 32 bits de mayor orden del *contador de secuencia*, la primera fase sólo debe ejecutarse cuando cambian estos bits, cada 65.536 paquetes.

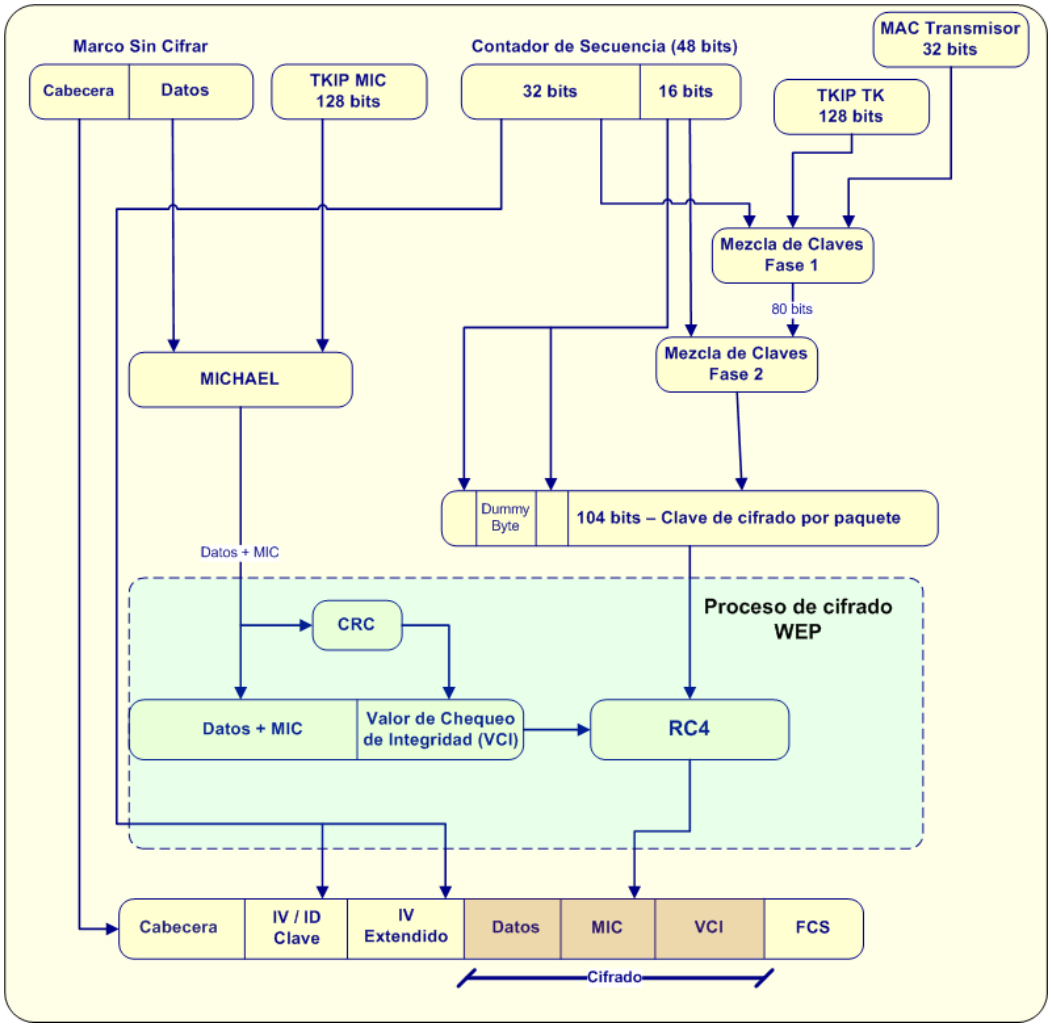

Ilustración III.13.: Proceso de cifrado de cada paquete en WAP.

5. La función de *mezcla de claves* hace muy difícil para quien intercepta el tráfico encontrar un correlato entre el número de secuencia y la clave de cada paquete.

El proceso de *chequeo de integridad del mensaje (MIC)* en general utiliza cálculos intensivos que requieren el uso de hardware acorde (por ejemplo MD5 o SHA-1). La solución de compromiso que se usa en *WPA,* debido a la restricción de cálculo que existe en el hardware, es *MICHAEL*, un nuevo protocolo *MIC*, que utiliza operaciones de desplazamiento y suma.

MICHAEL es una mejora sobre el algoritmo lineal *CRC32* que utiliza *WEP.* Y como es una solución de compromiso, *TKIP* implementa contramedidas en caso de que *MICHAEL* sea comprometido, por ejemplo, si *TKIP* detecta dos paquetes en que el *MIC* calculado no coincide con el *MIC* del paquete en el lapso de un segundo, infiere que la estación está bajo un ataque y borra sus claves, desasocia la estación y espera un minuto para volver a asociarla nuevamente.

802.11i(WPA2) aborda el servicio de confidencialidad e implementa un algoritmo para cifrado por bloques en lugar de un algoritmo de cifrado de *streams.* Para reemplazar a *RC4* utiliza AES en modo contador, combinando la seguridad de un algoritmo de cifrado de bloques con la facilidad de uso de un algoritmo de cifrado de *streams.*

Para proveer el servicio de integridad se utiliza *CCMP (Counter-mode CBC-MAC Protocol).* Este opera realizando *XOR* de un bloque en “texto plano” con el bloque previo que usa el algoritmo de cifrado *antes* de que

el mismo sea cifrado. El proceso se puede observar en la Ilustración III.14.

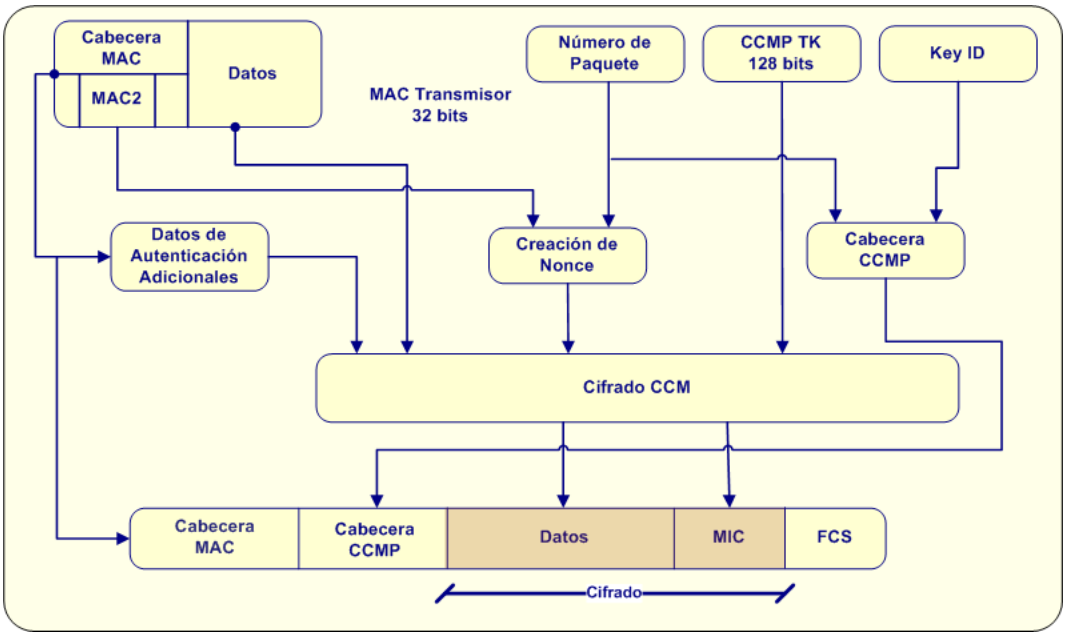

Ilustración III.14.: Proceso de cifrado de cada paquete en 802.11i.

En la Tabla III.8 se puede apreciar la comparación de *WPA, WPA2 y WEP*. Esta comparación tiene el foco en las debilidades de los servicios de confidencialidad e integridad de *WEP* listados en la página 30.

WEP	WPA	WPA2
No provee mecanismos para establecimiento de claves sobre medios inseguros. Las claves son compartidas en todo el *BSS* y frecuentemente en el *ESS*.	Recomienda el uso de 802.1X para autenticación y establecimiento de claves. También puede utilizar *preshared keys* como *WEP*.	Idem *WPA*.
Utiliza un algoritmo de	Idem *WEP*.	Utiliza un algoritmo fuerte de

WEP	WPA	WPA2
cifrado sincrónico para *streams*, sobre un medio en el que es muy difícil asegurar la sincrónización sobre una sesión completa.		cifrado por bloques (*AES*).
Utiliza una clave por-paquete que se construye concatenando la *MK con el IV* directamente. Esto expone la *MK* a ataques del tipo *FMS*.	Incorpora el concepto de *PTK* en la jerarquía de claves y utiliza una función de mezcla de claves, en vez de la concatenación directa para generar las claves por-paquete. Reduce la exposición de la *MK*.	Idem *WPA*.
Espacio de claves limitado debido a la *MK* configurada manualmente y al *IV* de 24 bits.	Aumenta el tamaño del *IV* a 56 bits, usando sólo 48 bits y reservando 8 bits para descartar claves débiles. Regenera las *PTK* en cada nueva sesión, aumentando el espacio de claves	Idem *WPA*.
El recambio de *IV* es opcional, por lo que la reutilización de claves es altamente probable.	Explícitamente especifica que se debe inicializar el *IV* a 0 cada vez que se establece una nueva *PTK*.	Idem *WPA*.
El algoritmo de *MIC* usado *es* CRC32. Es lineal y provee una protección de integridad muy débil.	Utiliza *MICHAEL* como algoritmo de *MIC*, que no es lineal y también especifica contramedidas de protección en caso de que el mismo sea comprometido.	Provee una fuerte protección de integridad al utilizar *CCMP, basado en AES*.
El *ICV* no protege la cabecera del paquete, por lo que existe el peligro de un ataque de redirección.	Extiende la protección del *ICV* para incluir las direcciones de origen y destino. Protege de esta forma contra los ataques de redirección.	Idem *WPA*.
No tiene protección contra los ataques de repetición.	Utiliza *IV* como número de secuencia para proteger contra los ataques de repetición.	Idem *WPA*.
No tiene soporte para que las estaciones autentiquen a la red.	No provee de forma explícita una solución a este problema, pero, al recomendar el uso de *802.1X,* puede utilizarse el	Idem *WPA*.

WEP	WPA	WPA2
	mismo para que las estaciones autentiquen a la red (*EAP-TLS,EAP-TTLS y EAP-PEAP).*	

Tabla III.8.: Comparación entre las falencias de WEP y las soluciones de WPA/WPA2.

5. PPPOE

PPP over Ethernet (PPPoE) [PPPoE] brinda la posibilidad de encapsular una sesión *PPP [PPP]* en marcos *Ethernet*. El protocolo *PPP* requiere de una conexión punto a punto entre el concentrador de acceso (*AC*) y el cliente, por lo tanto no se ha diseñado para las comunicaciones tipo multipunto que existen en una red *Ethernet.*

El principal uso de *PPPoE* son las tecnologías de acceso remoto a través de banda ancha, donde brinda la capacidad de de conectar un cliente a un *AC* a través de un dispositivo de *bridging*.

El protocolo hereda de *PPP* la posibilidad de mantener estadísticas de tráfico, efectuar controles de integridad de los datos del marco, asignar direcciones de red y, al ser un protocolo de capa de enlace, puede transportar cualquier protocolo.

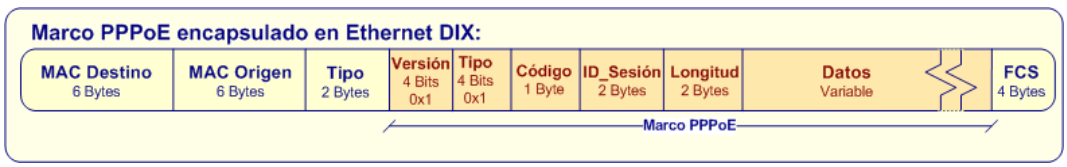

Ilustración III.15.: Marco PPPoE.

El principal problema que plantea la implementación de *PPPoE* está relacionado con el tamaño máximo de la *MTU*[1], el cual es inferior al de un marco *ethernet* debido a la encapsulación. Esto causa la fragmentación del paquete con la consecuente reducción del rendimiento.

En la se puede apreciar la disposición de los campos de un marco *PPPoE* y en la Tabla III.9 el significado de los mismos.

Campo	Significado
Versión	Este valor tiene 4 bits de longitud y debe ser 0x1 para la versión especificada en la *RFC2516.*
Tipo	Este valor tiene 4 bits de longitud y debe ser 0x1 para la versión especificada en la *RFC2516.*
Código	Este valor se usa para especificar la fase del estado de descubrimiento y de sesión *PPP.*
ID_Sesión	El valor *es 0x0000* durante las fase de descubrimiento para marcos de tipo *Discovery (excepto para marcos PADT y PADS)*, y una vez establecida la sesión *PPP,* se usa para identificar la conexión junto con las direcciones de *origen y destino*. El valor 0xFFFF se ha reservado para uso futuro y no debe utilizarse.
Longitud	Este valor indica la longitud de la carga del marco *PPPoE*, no incluye las cabeceras *Ethernet o PPPoE*.
Datos	El campo *Datos* contiene las cabeceras del protocolo *PPP* y los datos a transmitir.

Tabla III.9.: Significado de los campos del marco PPPoE.

El proceso de *Discovery o Descubrimiento* que se lleva a cabo al iniciar una sesión *PPPoE* consta de 4 fases diferentes. Durante este proceso el campo *Type* del marco *Ethernet* tiene el valor *0x8863.* La síntesis del proceso se puede observar en la Tabla III.10.

1 *MTU, Maximum Transmition Unit.* Unidad máxima de transmisión que especifica el máximo tamaño de paquete que se puede transmitir.

Fase	Sentido	Tipo de Marco
Fase 1	*Cliente --> Servidor*	**PADI:** *PPPoE Active Discovery Initiation* *Valor campo Código: 0x09* *Valor campo ID_Sesion: 0x0000*
Fase 2	*Cliente <-- Servidores*	**PADO:** *PPPoE Active Discovery Offer* *Valor Campo Código: 0x07* *Valor campo ID_Sesion: 0x0000*
Fase 3	*Cliente --> Servidor*	**PADR:** *PPPoE Active Discovery Request* *Valor Campo Código: 0x19* *Valor campo ID_Sesion: 0x0000*
Fase 4	*Cliente <-- Servidor*	**PADS:** *PPPoE Active Discovery Session-Confirmation.* *Valor Campo Código: 0x65* *Valor campo ID_Sesion: Número de sesión.* **PADT:** *PPPoE Active Discovery Terminated* *Valor Campo Código: 0xA7* *Valor campo ID_Sesion: Número de sesión.*

Tabla III.10.: Fases de Descubrimiento de PPPoE.

1. El cliente envía un marco *PADI* al *broadcast Ethernet* para descubrir los AC disponibles.

2. El *AC* que recibe el marco responde con un marco *unicast PADO* para informar de su existencia. Puede responder más de un *AC.*

3. El cliente selecciona el *AC* entre los marcos *PADO* recibidos y envía un marco *unicast PADR* al *AC* seleccionado.

4. El *AC* concede una sesión al cliente y envía un marco *unicast PADS* que incluye el número de sesión concedido. También puede enviar un marco *unicast PADT* para terminar la comunicación en cualquier momento.

Una vez que la sesión *PPPoE* se establece, se inicia la sesión *PPP* normalmente. Los marcos *Ethernet* de esta fase tienen el valor *0x8864* en

el campo *tipo*, el valor del campo *código* de *PPPoE* debe tener el valor 0x00 y el campo *ID_Sesión* no debe cambiar durante toda la sesión.

6. LDAP

Lightweight Directory Access Protocol (LDAP) [LDAP1] [LDAP2] es un protocolo de *Internet* para acceder a servicios de directorio distribuidos que actúan en concordancia con los modelos de datos y servicios *X.500* definidos por la *ISO*[1].

Según X.501 “El propósito del Directorio es contener y proveer acceso a objetos almacenados en él ... Un objeto puede ser cualquier entidad que sea identificable. Una clase de objeto es una familia de objetos identificados que comparten alguna característica.”

La unidad básica de información que contiene el Directorio es una *entrada de directorio.* Existen múltiples clases de entradas de directorio.

El conjunto de entradas de directorio que representan la *Base de Información del Directorio (DIB) y* se organizan jerárquicamente en una estructura de árbol llamada *Árbol de Información del Directorio (DIT)*. Las ramas entre dos entradas de directorio definen las relaciones entre ellas, y entre los objetos que dichas entradas representan.

Una *entrada de directorio* consiste en un conjunto de *atributos* que contienen información acerca del *objeto* que la entrada representa. Los *atributos* que representan información operativa o administrativa se

1 *ISO, International Organization for Standarization.* Organización Internacional para la Estandarización.

llaman *atributos operativos.* En las definiciones de cada clase de objeto se definen los atributos que están relacionados con un objeto de dicha clase.

Un atributo está compuesto por una descripción del atributo (un tipo y ninguna o alguna opción) y por uno o más valores asociados.

Cada entrada de directorio se nombra de forma relativa a su inmediato superior. Este nombre relativo se denomina *Relative Distinguished Name (RDN),* y debe ser único entre todos los subordinados de la entrada superior inmediata. Cada RDN se compone de uno o más *pares atributo-valor.* Por ejemplo: *uid=12345; ou=ventas, etc.*

```
......
attributetype ( 2.16.840.1.113730.3.1.4
        NAME 'employeeType'
        DESC 'RFC2798: type of employment for a person'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
.......
objectclass     ( 2.16.840.1.113730.3.2.2
    NAME 'inetOrgPerson'
    DESC 'RFC2798: Internet Organizational Person'
    SUP organizationalPerson
    STRUCTURAL
        MAY (audio $ businessCategory $ carLicense $ departmentNumber
          $ displayName $ employeeNumber $ employeeType $ givenName
          $ homePhone $ homePostalAddress $ initials $ jpegPhoto
          $ labeledURI $ mail $ manager $ mobile $ o $ pager
          $ photo $ roomNumber $ secretary $ uid $ userCertificate
          $ x500uniqueIdentifier $ preferredLanguage $
           userSMIMECertificate $ userPKCS12 )
```

Texto III.1.: Ejemplo de schema en LDAP.

El *nombre totalmente calificado* de una entrada es el *Distinguished Name (DN)* y es la concatenación de su *RDN* con el *DN* de su superior inmediato. El *DN* identifica unívocamente una entrada en el *DIT*. Por ejemplo, un *DN* sería: *CN=Juan Perez,OU=Ventas,O=ACME SRL, L=Godoy Cruz, ST=Mendoza,C=Argentina*.

Todos los elementos que componen el *DIT* están definidos en el *esquema o schema*, usando la sintaxis que se puede observar en el cuadro de Texto III.1.

```
dn: uid=00-admins,ou=profiles,dc=um,dc=edu
objectClass: radiusObjectProfile
objectClass: radiusprofile
objectClass: top
radiusFramedProtocol: PPP
radiusServiceType: Login
structuralObjectClass: radiusObjectProfile
entryUUID: 4da0c070-7653-102b-818d-53cd3ee9b86b
creatorsName: cn=admin,dc=um,dc=edu
createTimestamp: 20070403172003Z
dialupAccess: yes
radiusFilterId: 50
uid: 00-admins
cn: 00-admins
description: -N marca-admins -t mangle
description: -A marca-admins -t mangle -j MARK --set-mark 50
description: -A PREROUTING -t mangle -s 192.168.176.0/26 -m mark --
mark 0 -j marca-admins
description: -N filtra-admins
description: -A FORWARD -m mark --mark 50 -j filtra-admins
```

Texto III.2.: Ejemplo archivo LDIF.

En la definición del cuadro de Texto III.1, también se puede observar un número resaltado, este número identifica a cada tipo de elemento y se denomina *OID* (*object Identifier*). *El OID* es global y único en el esquema.

Si bien el *DIT* almacena los datos en forma binaria, se puede observar en el cuadro de Texto III.2 una entrada de directorio en formato *LDIF*[1], donde aparece el *DN, las clases a las que pertenece y los atributos* de un objeto.

7. RADIUS

RADIUS[2] *[RADIUS]*[3] [RADIUSACC][4] es un protocolo de control de acceso que autentica usuarios a través de un método muy común denominado genéricamente *desafío/respuesta.* El proceso que lleva a cabo se denomina *AAA*[5]*.*

El proceso de *AAA* se puede sintetizar en las siguientes preguntas que realiza el servidor de acceso al cliente:

- ¿Quién eres?
- ¿Qué servicios estoy autorizado a darte?
- ¿Qué hiciste con los servicios mientras los usabas?

---

1 *LDAP Data Interchange Format,* definido en las RFC2849 y modificado en la RFC 4525.

2 *Remote Authentication Dial In User Service.* Servicio de Autenticación de Usuarios Remotos.

3 La RFC2865 ha sido actualizada por las RFC2868 y RFC3575. Y deja obsoleta a la RFC2138.

4 La RFC2866 ha sido actualizada por la RFC2867 y deja obsoleta a la RFC2139.

5 *Autherntication, Authorization and Accounting*. Autenticación, Autorización y Registro.

La *Autenticación* es el proceso de verificar la identidad que ha declarado un cliente (máquina o persona). Uno de los métodos más comunes para realizar esta tarea es utilizar un nombre de usuario y una contraseña. El objetivo principal de este proceso es formar una *relación de confianza* entre el cliente y el servidor.

El proceso de *Autorización* consta de un conjunto de reglas para decidir qué puede hacer un usuario autenticado en el sistema. Las reglas son definidas por los administradores del sistema.

El *Registro* es el proceso que documenta el uso de los recursos hecho por los usuarios que acceden al sistema. Los datos recabados por el proceso de *Registro* se pueden utilizar para controlar autorizaciones realizadas, cobrar la utilización de recursos, medir tendencias en el uso de recursos, y para realizar actividades relacionadas con la planificación del crecimiento.

Las características generales del protocolo, definidas en la *RFC2865*, son:

- Usa un modelo *Cliente/*Servidor, donde el modelo de seguridad que se utiliza se denomina *hop-by-hop,* y se basa en un secreto compartido entre el *NAS*[1] *y* el *Servidor* que no se transmite en la red.

- Utiliza *UDP/IP* para las conexiones.

- Es un protocolo *stateless*, es decir sin estado.

1 *Network Access Server*. Servidor de acceso a la red, usualmente es la entidad que recibe la petición de acceso del usuario y solicita al servidor *RADIUS* la autenticación del mismo.

- Soporta una gran variedad de métodos de autenticación, que otorgan una gran flexibilidad. Soporta *PAP, CHAP, EAP, Unix,* entre otros.

- El protocolo es *extensible*. Todas las transacciones utilizan un trío de valores *Atributo-Longitud-Valor*, y permite agregar definiciones de valores sin comprometer el funcionamiento de las implementaciones existentes.

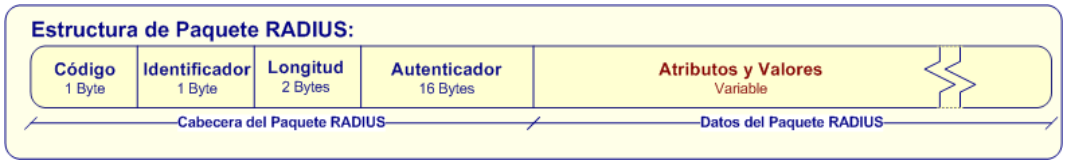

Ilustración III.16.: Estructura del paquete RADIUS.

El protocolo se comunica a través de los puertos UDP/1812 y UDP/1813. En la Ilustración III.16 se puede apreciar la estructura del paquete *RADIUS.* En la Tabla III.11 se puede observar el significado de los campos del paquete *RADIUS.*

Campo	Descripción
Código	Este campo es el que distingue el tipo de mensaje *RADIUS.* Los paquetes con código inválido se descartan. Los códigos permitidos son: *Access-Request (1), Access-Accept (2), Access-Reject (3), Accounting-Request (4), Accounting-Response (5), Access-Challenge (11), Status-Server (12), Status-Client (13), Reservado (255).*
Identificador	Este valor se utiliza para relacionar una respuesta a una solicitud. También permite que el servidor *RADIUS* detecte las solicitudes duplicadas que se realizan en un corto período de tiempo, comparando la dirección *IP* y el puerto *UDP* origen, junto con el identificador.
Longitud	Indica la longitud (incluye *Código, Identificador, Longitud, Autenticador y Datos)* y si el paquete es más largo se descarta el excedente y si el paquete es más corto se descarta. Las longitudes válidas oscilan entre 20 *bytes y 4096 bytes*.

Campo	Descripción
Autenticador	Con este valor se verifica la integridad de los datos. Existen dos tipos diferentes de valores: *Solicitud y Respuesta.* El primer valor se calcula de forma aleatoria. El segundo valor se calcula con el algoritmo *MD5* en función de los siguientes parámetros: *Código, Identificador, Longitud, Autenticador de la solicitud, Datos, Secreto).*
Atributos y Valores	Este campo de longitud variable es el contiene los pares *atributo-valor* que intercambia el servidor con el *NAS.*

Tabla III.11.: Descripción de los campos de un paquete RADIUS.

En los procesos de *Autenticación y Autorización* son relevantes cuatro tipos de paquetes: *Access-Request, Access-Accept, Access-Reject, y Access-Challenge.* El primer paso del proceso de autenticación lo ejecuta el usuario.

El cliente solicita acceso a un servicio particular al enviar al *Servidor RADIUS* un paquete del tipo *Access-Request*. El paquete debe contener ciertos atributos que son significativos para el proceso de autenticación. Por ejemplo, debe contener atributos que identifiquen al *NAS (NAS-IP-Address y/o NAS-Identifier*), la contraseña del usuario *(User-Password o CHAP-Password)* y debería contener nombre de usuario *(User-Name)* y el puerto de conexión *(NAS-Port y/o NAS-Port-Type*). El paquete *Access-Request* puede contener atributos adicionales. Si el *atributo User-Password* está presente, no debe transmitirse en texto plano, sino utilizando un *hash MD5.* Al recibir el paquete, el servidor debe determinar y comunicar si el usuario tiene permitido el acceso a través del *NAS* y al tipo de servicio que ha solicitado. Todos los paquetes válidos de este tipo deben ser contestados por el servidor, ya sea con un paquete del tipo *Access-Accept,* del tipo *Access*-Reject o del tipo *Access-Challenge.*

Una vez que el servidor ha procesado un paquete tipo *Access-Request* y ha determinado que todos los atributos son aceptables y se

puede conceder acceso al usuario, envía un paquete del tipo *Access-Accept*. Este paquete debe proveer la información necesaria para realizar configuración específica a fin de comenzar con la utilización del servicio. El campo *Autenticador* debe contener la respuesta correcta para el campo *Autenticador* de paquete tipo *Access-Request* que ha recibido previamente.

El servidor puede enviar al usuario un desafío que requiera de una respuesta para disminuir el riesgo de una autenticación fraudulenta. Si este es el caso se envía al cliente un paquete del tipo *Access-Challenge* al que el cliente debe responder con un paquete del tipo *Access-Request* que incluya los atributos apropiados. En el paquete *Access-Challenge* se pueden incluir uno o más atributos *Reply-Message* y un atributo *State*. Adicionalmente sólo se puede incluir alguno de los siguientes atributos: *Específicos-del-Fabricante*, *Idle-Timeout, Session-Timeout o Proxy-State.*

Si el servidor determina, al procesar el paquete *Access-Request*, que debe negar el acceso porque alguno de los atributos recibidos no es aceptable (por política de sistema, privilegios insuficientes u otro criterio), debe enviar un paquete del tipo *Access-Reject*. Este tipo de paquete puede incluir uno o más atributos *Reply-Message* con el texto que el *NAS* debe mostrar al usuario y uno o más atributos *Proxy-State.* No se permiten otro tipo de atributos. Los paquetes del tipo *Access-Reject* pueden ser enviados en cualquier instancia de una sesión, por lo que son muy útiles para aplicar límites de tiempo en la duración de la sesión.

Una vez que se lleva a cabo el proceso de autenticación puede empezar el proceso de *Accounting o* Registro, para esto usa el puerto 1813/*UDP*. El cliente (*NAS o Proxy Server)* envía un paquete del tipo

Accounting-Request al *Servidor de Accounting RADIUS.* Este servidor puede ser el mismo servidor RADIUS utilizado para los procesos de *Autenticación y Autorización* u otro servidor configurado para tal propósito.

Este paquete de tipo *Accounting*-Request, *denominado Accounting-Start,* transmite la información que se registrará como utilización del servicio por parte del usuario. Al recibir este paquete el servidor debe transmitir un paquete del tipo *Accounting-Response* si puede almacenar satisfactoriamente el contenido del paquete y no debe transmitir ninguna respuesta si existe algún tipo de falla en el proceso de almacenamiento. A excepción de los atributos *User-Password, CHAP-Password, Reply-Message y State,* se pueden enviar todos los atributos que pueden estar en un paquete *Access-Request o Access-Accept.* El paquete *Accounting-Request* debe contener el atributo *NAS-IP-Address y/o NAS-Identifier.* Si está presente en el paquete o si estuvo presente en el paquete *Access-Accept*, el atributo *Framed-IP-Address* debe contener la dirección *IP realmente* asignada al usuario.

Cuando el cliente finaliza una sesión envía un paquete *Accounting*-Request, denominado *Accounting-Stop*, que incluye las estadísticas de uso (duración de la sesión, volumen de datos transferidos, etc.). El servidor debe confirmar que pudo almacenar esta información con éxito enviando un paquete tipo *Accounting-Response*. El cliente debería reenviar el paquete *Accounting-Respose* hasta que el servidor confirme la operación de almacenamiento.

Como se puede apreciar en los procesos *AAA*, *RADIUS* transporta toda la información necesaria para llevar a cabo estos procesos en

Atributos. Los atributos se transmiten con el formato que se observa en la Ilustración III.17 y en la Tabla III.12.

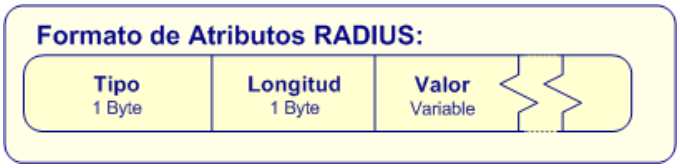

Ilustración III.17.: Formato de Transmisión de Atributo RADIUS.

Campo	Descripción
Tipo	Este valor especifica el "número de atributo" que lo define, estos números están definidos en las RFC2865 y RFC2866. El número 26 está designado para *Vendor Specific Attributes (VSA).* Los valores 192-223 están reservados para uso experimental, los valores 224-240 están reservados para uso de cada implementación específica y los valores 241-255 están reservados para uso futuro.
Longitud	Este valor indica la longitud del atributo transmitido, incluyendo los campos tipo, longitud y valor.
Valor	Este valor puede ser de longitud cero y puede tener los siguientes tipos de dato: *texto, cadena, dirección IP, número entero y tiempo.*

Tabla III.12.: Descripción del formato de transmisión de los atributos RADIUS.

En la descripción del campo *valor* se especifican cinco tipos de dato soportados cuyo formato es:

- *Texto*: de 1 a 253 *bytes* que deben contener caracteres con codificación *UTF-8.* En lugar de enviarse con longitud cero debe omitirse el atributo.

- *Cadena:* de 1 a 253 *bytes* que deben contener *datos binarios*. En lugar de enviarse con longitud cero debe omitirse el atributo.

- *Dirección IP:* valor de 32 bits con el *byte* más significativo en primer lugar.

- *Número Entero:* valor de 32 bits sin signo con el *byte* más significativo en primer lugar.

- *Tiempo:* valor de 32 bits sin signo con el *byte* más significativo en primer lugar. Se debe codificar como el número de segundos desde el 01/01/1970 a las 00:00:00 UTC.

Muchas de las definiciones de atributos establecen un conjunto de valores posibles por *enumeración* para dicho atributo, el tipo de dato utilizado en esos casos es *número entero*.

El *Tipo 26* está reservado para uso de *Vendor Specific Attributes (VSA,* Atributos específicos del fabricante), y debe usar un formato especial para encapsular las definiciones de los atributos *VSA* dentro del campo *valor*.

Ilustración III.18.: Formato de atributos VSA.

Los sub-campos son similares a los campos de un atributo normal, salvo el sub-campo *ID Fabricante*. Este campo es un código de 32 bits que representa al fabricante y que está definido en el documento *RFC1700* como *"Números Asignados".* Este código se denomina *Network Management Private Enterprise Code (NMPEC).*

Para relacionar los atributos con su código y tipo o para definir los atributos *VSA* es necesario el uso de *Diccionarios*. En ellos se especifican los atributos estándar y sus propiedades, así como los atributos *VSA* necesarios, como se puede observar en el cuadro de Texto III.3.

```
.......
#       Service types
VALUE   Service-Type       Login-User             1
VALUE   Service-Type       Framed-User            2
VALUE   Service-Type       Callback-Login-User    3
VALUE   Service-Type       Callback-Framed-User   4
VALUE   Service-Type       Outbound-User          5
VALUE   Service-Type       Administrative-User    6
VALUE   Service-Type       NAS-Prompt-User        7
VALUE   Service-Type       Authenticate-Only      8
VALUE   Service-Type       Callback-NAS-Prompt    9
VALUE   Service-Type       Call-Check             10
VALUE   Service-Type       Callback-Administrative 11
```

Texto III.3.: Ejemplo de diccionario RADIUS.

IV. DESARROLLO

1. PLANTEO DE RESTRICCIONES Y NECESIDADES

La problemática de diseño de una red inalámbrica en un ambiente semi público presenta necesidades y restricciones que no están presentes en otro tipo de implementaciones. Precisamente por tratarse de un ambiente semi público, se plantea la necesidad brindar servicios diferentes a grupos de usuarios distintos. Por lo menos, surge la necesidad de diferenciar dos grandes grupos, un grupo de usuarios como si fuesen usuarios *privados o internos,* y a otro grupo como si fuesen usuarios de una red pública, es decir con ciertas restricciones en el acceso.

En general siempre se cumple que, *A mayor seguridad, menor facilidad de uso*. De esta premisa se puede inferir que, si los administradores de red pudieran elegir *sin restricciones*, es evidente que elegirían el modelo que brindara la mayor seguridad posible en los accesos de red.

El problema parte de la existencia de restricciones importantes, principalmente de compatibilidad, en la implementación de los modelos de seguridad existentes. En efecto, estas restricciones permiten imaginar un modelo en el que la *política de conexión* requiere de diferentes niveles de seguridad en la conexión inalámbrica, en función a los permisos de acceso del perfil cada usuario.

El planteo de un modelo de infraestructura para el acceso inalámbrico en un ámbito semi-público presenta, por lo tanto, la necesidad de llegar a una solución de compromiso entre la seguridad de los puntos de acceso

brindados, la compatibilidad de los dispositivos cliente y la necesidad de brindar diferentes servicios o niveles de servicio a diferentes grupos de usuarios.

Se plantea a continuación una lista de elementos que componen la problemática general de los accesos semi-públicos. Cada uno de los elementos plantea objetivos a resolver, algunos de ellos contrapuestos:

1. **Compatibilidad del equipamiento:** Esta es una fuerte restricción a la hora de definir la política de seguridad de acceso para los dispositivos inalámbricos. Como se vio, el modelo *RSN* que plantea *802.11i* no es compatible con los dispositivos *WEP.* Y no puede ser salvado por una actualización del firmware de los dispositivos inalámbricos, porque el soporte de cifrado está en el hardware y el algoritmo *AES* es mucho más intensivo en el uso de capacidad de cálculo que el algoritmo *RC4.* Como consecuencia, definir una política de seguridad *RSN* implica desechar los dispositivos existentes, tanto a nivel de infraestructura de acceso, como en los dispositivos cliente. La decisión de desechar los dispositivos cliente en un ámbito semi-público, implica negar conectividad a una porción de usuarios. La opción por un modelo de seguridad basado en *TSN*, disminuye significativamente la porción de usuarios a los que se le negaría conectividad inalámbrica, pero todavía no contempla todo el universo posible de dispositivos. Además de tener en cuenta la compatibilidad del equipamiento a nivel de seguridad, en un segundo plano, hay que tener en cuenta la compatibilidad del equipamiento en cuanto a modo de transmisión, *802.11 a, b, o* g. El estándar *802.11g* es compatible con *802.11b*, y alcanza mayores velocidades de

transmisión, por lo que es recomendable para las implementaciones nuevas. Muchos de los dispositivos *PDA* tienen capacidad sólo para transmitir en *802.11b,* lo que refuerza la elección de *802.11g* como estándar para la infraestructura.

2. Compatibilidad de los sistemas operativos: No todos los sistemas operativos soportan *RSN o TSN*, aún si el hardware del cliente lo soporta. Por ejemplo, en la familia de sistemas operativos Microsoft Windows el soporte nativo se puede ver en la Tabla IV.1:

Sistema Operativo	Soporte Nativo de *WPA*	Soporte Nativo de *WPA2*
MS Windows Vista	Sí	Sí
MS Windows XP	Con Service Pack 2 o Service Pack 1 y Wireless Update Rollup Package for Windows XP	Con Service Pack 2 y Wireless Client Update for Windows XP with Service Pack 2
MS Windows 2000	No*[1]	No*[1]
MS Windows Me	No*	No*
MS Windows 98	No*	No*
MS Windows NT 4.0	No*	No*
MS Windows Server Longhorn	Sí	Sí
MS Windows Server 2003	Con Service Pack 1 o Service Pack 2	Con Service Pack 2

Tabla IV.1.: Soporte Nativo de WPA y WPA2 de los Sistemas Operativos Microsoft

** el fabricante del dispositivo puede proveer software para dar soporte a WPA o WPA2. [1] Sólo provee soporte para 802.1X (fuente: http://www.microsoft.com/technet/network/wifi/wififaq.mspx#EGAAE Actualizado a Abril 2007.)*

De este limitado ejemplo, que no contempla dispositivos PDA u otros sistemas operativos (Linux, OSX, etc.), se desprende que al definir una política de seguridad, las restricciones no sólo son por el hardware de los dispositivos inalámbricos. Como consecuencia, el definir una política de seguridad rígida basada en *802.11i* implica negar conectividad a una porción de usuarios o brindar una alternativa adicional de conectividad fuera de los modelos *RSN o TSN.*

3. Permisos para Usuarios y Perfiles de Usuarios: La implementación de validación de usuarios y la asignación de permisos a los mismos sobre lo que pueden hacer o no, una vez conectados a la red, presenta una gran complejidad. En necesario definir varios requisitos para que este objetivo sea completado:

1. Existencia de un listado o base de datos de usuarios que contenga los requisitos necesarios para *autenticar* a los usuarios y para almacenar sus vinculaciones con los permisos del grupo al que pertenecen, además de permisos específicos del usuario. Dicha base de datos deberá ser *compatible* con los métodos de acceso seleccionados para autenticar a los usuarios y con los sistemas operativos de los nodos cliente.

2. Definición de un protocolo común a todos los métodos de acceso y sistemas operativos que permita realizar las tareas de *AAA* necesarias para validar el acceso a los recursos por parte de los usuarios de la red, accediendo a la base de datos de usuarios y permisos. Este protocolo debe

actuar de nexo entre los mecanismos de autenticación de los nodos cliente y la base de datos.

3. Definición de la forma en que se implementarán las restricciones que sufrirán los usuarios una vez que estén autenticados y conectados a la red inalámbrica. Estas restricciones deberán ejecutarse en el punto de conexión con la red interna, y no en los nodos clientes de la red, a fin de presentar un único punto de control y de evitar el costo de desarrollar una implementación para cada combinación de hardware/sistema operativo.

4. Definición de la forma en que se generarán los registros de las operaciones que llevan a cabo los usuarios. La existencia de estos registros permitirá auditar tanto las operaciones llevadas a cabo por los usuarios, como los patrones de tráfico generado por los mismos y validar que el comportamiento de la red inalámbrica es el deseado.

4. **Herramientas de Administración de Usuarios y Perfiles:** Es necesario brindar herramientas amigables que permitan una fácil administración tanto de los permisos de los perfiles de usuario, como de los usuarios. De forma que el crecimiento en el número de usuarios o perfiles de usuario, no represente un problema para los administradores de la red.

2. MODELO Y ENTIDADES AAA

El modelo de infraestructura que se plantea en este trabajo brinda un camino de solución a cada uno de los problemas planteados. Se puede observar el modelo que se propone en el diagrama de bloques de la Ilustración IV.1. Cada bloque representa una función específica que debe ser cubierta por la implementación.

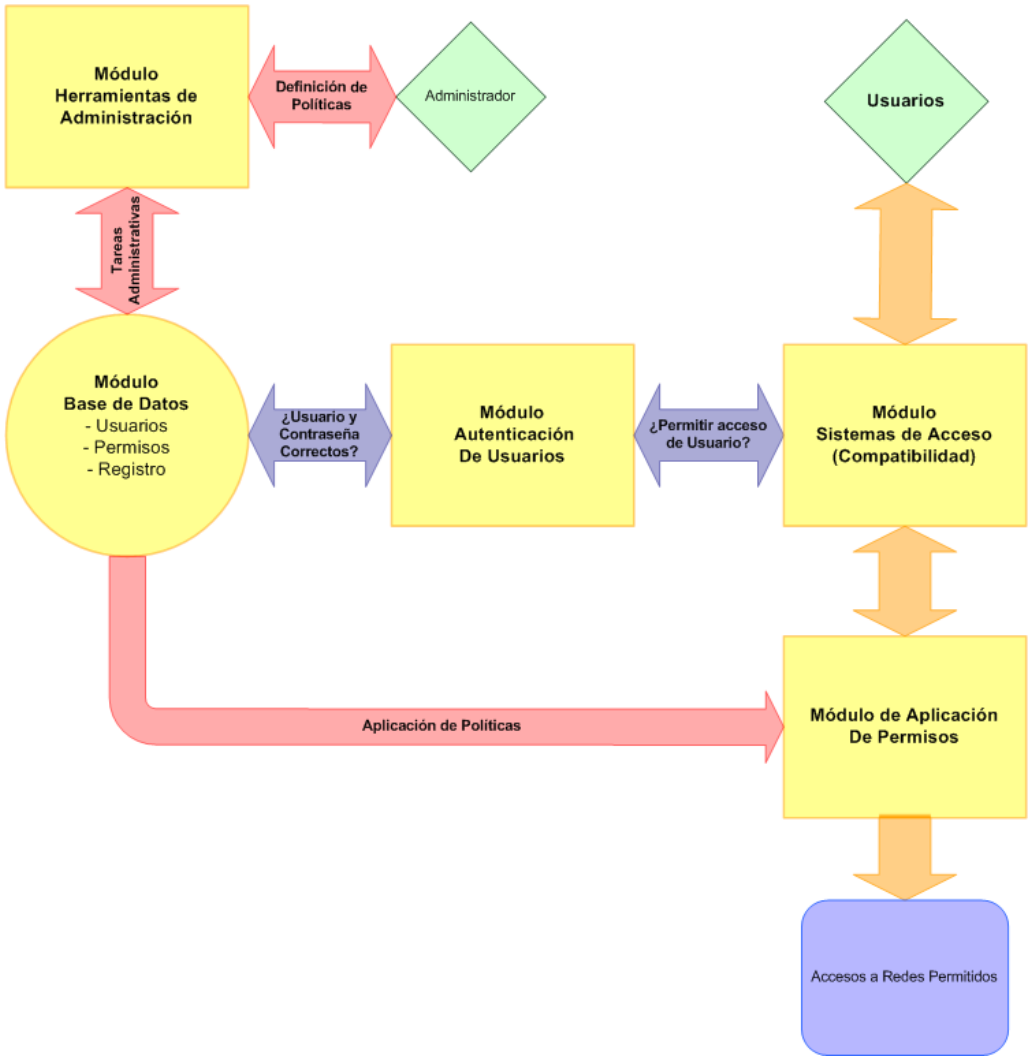

Ilustración IV.1.: Modelo funcional propuesto.

Se puede visualizar un esquema general de la topología física del modelo planteado en la Ilustración IV.2. En la ilustración se puede apreciar el tratamiento de la red inalámbrica como una red hostil, separada de la red interna o privada de la organización a través de un *firewall.*

La red semi pública está compuesta por dos redes diferentes, una es la *red inalámbrica propiamente* dicha, y la otra es la *red de distribución de servicios inalámbricos*, esta última es una red generalmente cableada que interconecta todos los dispositivos que brindan servicios a la red inalámbrica. Como medio de transmisión de esta última red se puede utilizar un cableado dedicado, una *VLAN* específica dentro de la red cableada privada o *WDS* [1].

1 Wireless Distribution System , Sistema de Distribución Inalámbrica.

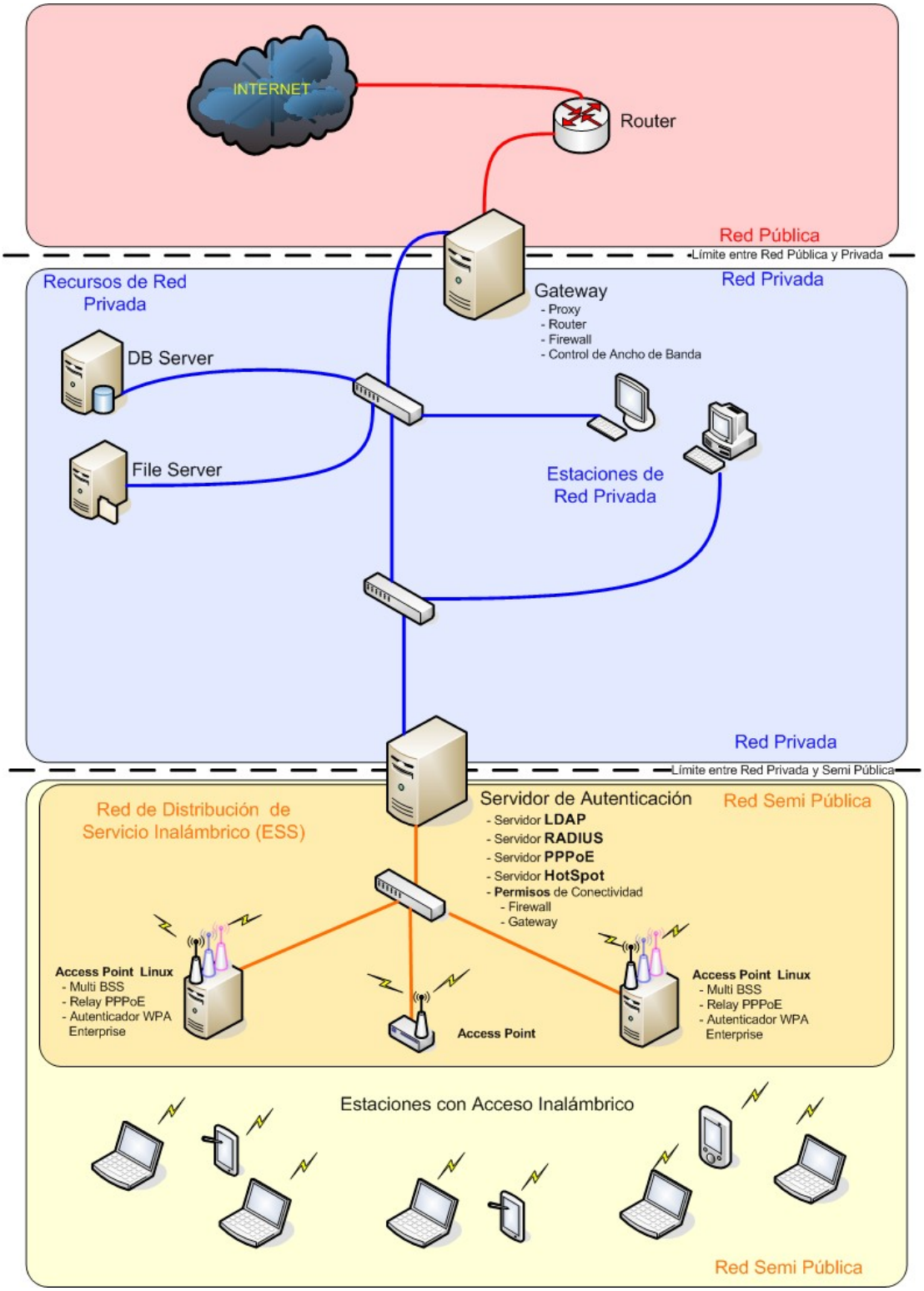

Ilustración IV.2.: Modelo de Infraestructura general propuesto.

Como puede verse en la Ilustración IV.2, todo el tráfico de la red inalámbrica confluye en el *servidor de autenticación.* Dicho servidor funciona como un punto de control por el que obligatoriamente debe pasar todo el tráfico de la red inalámbrica. Es por ello que este punto es donde se realizan los controles de los permisos que poseen los usuarios dentro de su perfil de conectividad, y además se realiza el *routing* de los paquetes.

La red de distribución, desde el punto de vista del usuario inalámbrico, es totalmente transparente, ya que sólo “ve” el *AP* que le brinda acceso al *ESS.* Los dispositivos que conforman la infraestructura de esta red tienen un direccionamiento *IP* totalmente diferente al de las redes privadas e inalámbrica.

En la ilustración también figuran dispositivos denominados *AP Linux*, estos dispositivos son computadores con sistema operativo *Linux*, y dos interfaces de red, una interfaz *ethernet cableada y una interfaz inalámbrica 802.11 a/b/g* .

A fin de poder presentar de forma ordenada cada una de las funciones que cumplen los dispositivos que componen este modelo, se procede a analizar cada uno de los bloques funcionales por separado, haciendo referencia a su aporte funcional a la solución completa.

En la Ilustración IV.3 se puede apreciar el *sistema de distribución inalámbrico*, con los bloques funcionales ampliados que muestran las funcionalidades de cada componente de infraestructura.

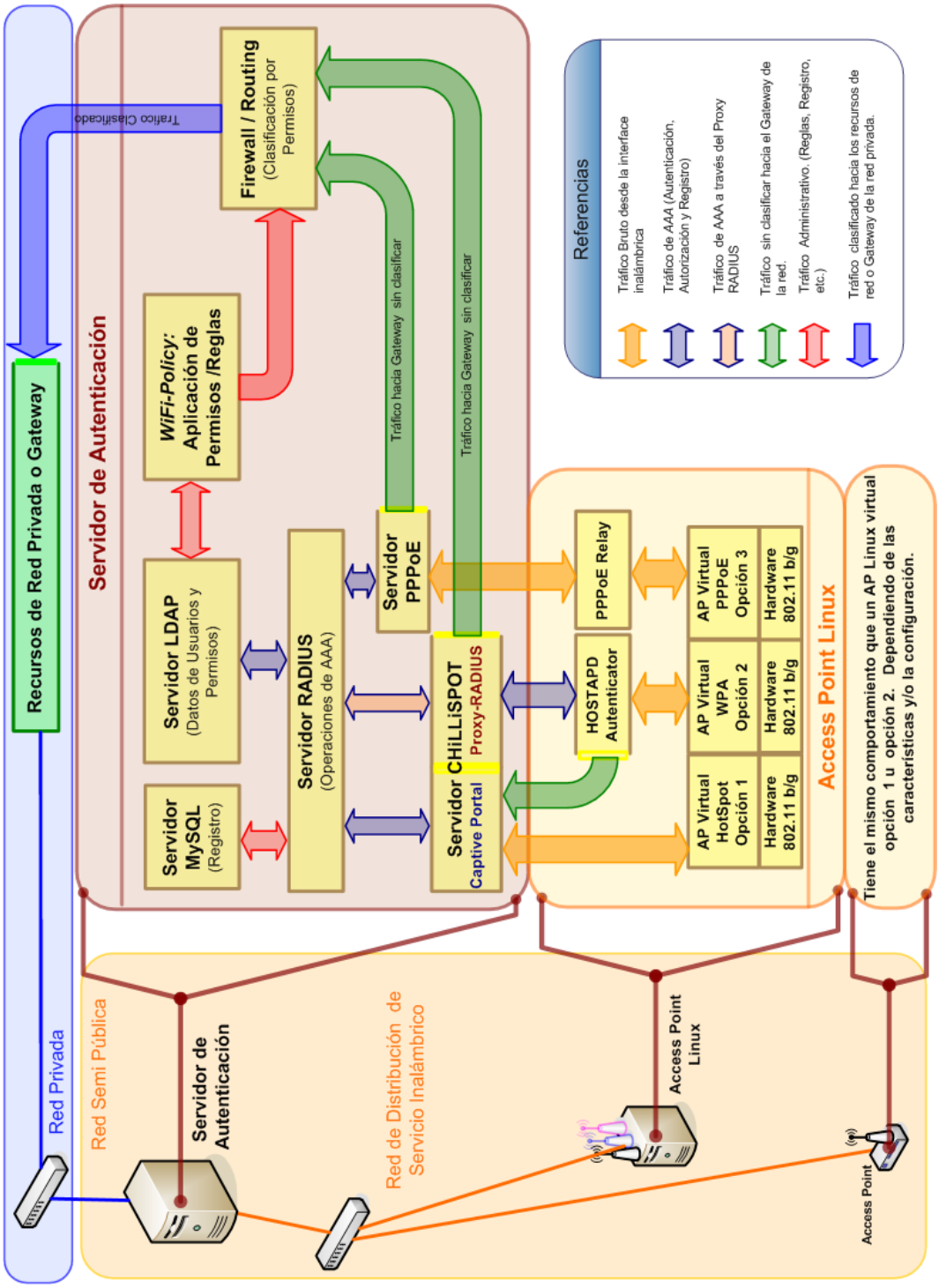

Ilustración IV.3.: Bloques Funcionales Implementados del Modelo de Infraestructura.

3. BLOQUES FUNCIONALES

El modelo de infraestructura se puede dividir en tres grandes bloques de infraestructura. Dos de estos son el *Servidor de Autenticación* y los *AP.* Estos bloques son atravesados transversalmente por los *Métodos de Acceso,* que definen el comportamiento de estos bloques funcionales. La implementación de los bloques de este modelo se llevó a cabo con sistema operativo GNU Linux, distribución Debian 3.1 Sarge[1].

1. MÉTODOS DE ACCESO

Los *Métodos de Acceso*, definen la forma en que se conectarán los nodos cliente, y por lo tanto los requerimientos tanto en el hardware como en el sistema operativo de los mismos. Todos los métodos de acceso propuestos permiten realizar el proceso de autenticación por usuario, para asignar los permisos de conectividad por usuario o por perfiles de usuario.

Si bien pueden implementarse sólo algunos de los *Métodos de Acceso* que se va a describir, la fortaleza del modelo radica en soportar varios *Métodos de Acceso* de forma simultánea. Esto tiende a poder brindar al menos una forma de conectarse a los nodos cliente, independientemente de las capacidades de conexión de los mismos, salvando de este modo las incompatibilidades existentes, expresadas al principio de este capítulo.

1. MÉTODO PPPOE

Este método permite, como es tradicional para la familia de protocolos PPP, la autenticación por usuario. Requiere que en el *Servidor de Autenticación* se ejecute un servidor PPPoE y en el *AP Linux* se ejecute

1 http://www.debian.org.

un servicio de *Relay de PPPoE*, que se encarga de reenviar las tramas PPPoE que se reciban a través de la interfaz inalámbrica al servidor de PPPoE que se ejecuta en el *Servidor de Autenticación.*

La elección de este método se debe a la gran popularidad alcanzada por PPPoE, debido principalmente a la utilización por parte de los Proveedores de Servicios de Internet como método para brindar acceso por usuario. Como consecuencia, existen clientes PPPoE para la mayoría de las plataformas, incluidas las *PDA*. Inclusive existe un borrador de la IETF denominado "*Using PPP-over-Ethernet (PPPoE) in Wireless LANs*" [DRAFT-GPATERNO], que sugiere la implementación de PPPoE para realizar accesos a redes inalámbricas.

Algunas de las ventajas brinda *PPPoE* son;

- A través de los *plugins* de *radius.so y radattr.so*, el servidor *PPPoE* puede autenticar a los usuarios contra un *Servidor RADIUS*, y configurar la sesión con los parámetros que envíe dicho servidor.

- *PPPoE* aporta el concepto de *sesión sobre Ethernet,* por lo que se puede realizar un seguimiento de los tiempos de conexión de un usuario en particular, que son registrados en *el Servidor RADIUS.*

- Dado que las sesiones *PPPoE* son sólo sesiones *PPP* encapsuladas sobre *Ethernet,* el servidor se ocupa de asignar la dirección *IP* al nodo cliente, por lo que no se requiere la utilización de un servidor *DHCP* separado. Por otro lado, esta asignación de dirección puede controlarse a través de los parámetros de

configuración, siendo posible asignar un *pool* de direcciones dinámicas, o asignando una dirección específica para cada usuario, a través de los parámetros que envía el *Servidor RADIUS.*

- El protocolo *PPPoE*, al ser un protocolo de capa de enlace, puede encapsular protocolos distintos al protocolo *IP*.

El servidor *PPPoE*, soporta la autenticación de usuarios a través de:

- PAP (Password Autentication Protocol)
- CHAP (Challenge Handshake Authentication Protocol)
- MS-CHAP (Challenge Handshake Authentication Protocol)
- MS-CHAP V2 (Challenge Handshake Authentication Protocol Version 2)

Se elige realizar la autenticación de los usuarios a través de MS-CHAP-V2, debido a que es la opción más segura de las soportadas. Si bien *PPPD*, soporta un desafío *EAP-MD5* como método de autenticación, la implementación actual no permite usar el mismo con un servidor *RADIUS*, sólo permite utilizar el desafío contenido en un archivo local por lo que se descarta su utilización en este modelo.

Si bien el protocolo soporta cifrado *MPPE*[1] *[MPPE]* para el flujo de datos, la implementación del servidor sobre sistema operativo *GNU/*Linux presenta fallas al utilizarlo con clientes con sistema operativo *MS-Windows.* Las pruebas realizadas se pueden encontrar en el Apéndice F.

1 *Microsoft Point-to-Point Encryption Protocol*

Es por ello que, a pesar de que este método de acceso permite la utilización de varios tipos de nodo cliente y está dotado de una relativa seguridad en el momento de la autenticación al utilizar *MS-CHAP-V2,* posee una gran desventaja al no poder cifrar el tráfico entre el nodo y el servidor de autenticación, por ello no se recomienda su uso en estaciones que realicen operaciones de administración en la red.

Este método se implementa a través de dos componentes, en el *Servidor de Autenticación* se ejecuta el servidor *PPPoE*, y en los *AP Linux*, se ejecuta el *Relay de PPPoE*. Los nodos cliente que acceden por este método deben configurarse con un cliente de *PPPoE.*

2. WPA/RSN – HOSTAPD

Este método puede implementarse tanto con un *AP Linux* como con *AP* convencionales que soporten *WPA/WPA2-Enterprise*. Requiere de un servidor RADIUS, para realizar las tareas de *AAA*, y de un servidor *DHCP* para entregar las direcciones *IP* a los nodos cliente.

Tanto en los *AP* convencionales, como en los *AP* Linux, el *AP* cumple con los roles de *AP* y de *Autenticador,* a través del uso del protocolo *IEEE802.1X.* En los *AP Linux*, el *Autenticador*, está implementado con el software libre *hostapd-0.5.5*[1]. Dicha implementación de software soporta la autenticación *IEEE802.1X* a través de la suite de protocolos *EAP, y* soporta las siguientes variantes del protocolo:

- EAP-TLS
- EAP-TTLS (EAP-MD5, EAP-GTC, EAP-MSCHAPV2, MSCHAPv2, MSCHAP, PAP, CHAP).
- EAP-PEAP (MSCHAPv2, GTC, MD5)

1 http://hostap.epitest.fi/hostapd/

- ESP-SIM
- EAP-AKA
- EAP-PAX
- EAP-PSK
- EAP-SAKE
- EAP-FAST
- EAP-MD5
- EAP-MSCHAPv2
- EAP-GTC

Debido a la gran cantidad de métodos que proporciona el protocolo *EAP*, al elegir los métodos que se usarían, se tuvo en cuenta los siguientes puntos:

- Nivel de seguridad proporcionada.
- Compatibilidad con los dispositivos cliente.
- Viabilidad de implementación (Soporte en *Servidor de Autenticación* y *AP convencionales*)
- Facilidad de administración.

Por estas razones, la elección recayó en dos implementaciones *EAP-TTLS y EAP-PEAP.* Estos métodos proporcionan un nivel de seguridad aceptable, son ampliamente soportados en los clientes, tanto nativamente en los sistemas operativos, como a través de implementaciones de software libre; y son soportados en los *AP* convencionales. Si bien también es factible y puede ser deseable usar *EAP-TLS,* el mismo requiere una compleja administración de *Infraestructura de Claves Públicas* (*PKI).* En efecto, requiere la emisión de certificados para cada cliente, no sólo para el servidor, de forma de que los clientes validen la red a la que se conectan y el servidor valide los cliente que se conectan a

a la red. Es por ello que se deja la decisión de utilizarlo a los administradores de red que implementen este modelo.

Este *Método de Acceso* aparte de la infraestructura para autenticar a los usuarios y cifrar la transmisión de datos, requiere de ciertos componentes adicionales, como mínimo un servidor *DHCP,* para establecer la conectividad de los clientes. Si bien se puede recurrir a la utilización de un servidor DHCP estándar implementado en software libre o de un servidor DHCP embebido en los *AP* convencionales, en este modelo se recurre a la integración con una implementación específica de un *Portal Cautivo* debido a ciertas ventajas que se plantean en la sección siguiente.

3. PORTAL CAUTIVO (*CAPTIVE PORTAL*)

Un *Portal Cautivo* es una técnica que fuerza a un cliente *HTTP*[1] a ver una página web en especial, que usualmente se usa para autenticar al cliente. Una vez que el cliente se ha autenticado, puede usar el servicio de forma normal.

Este comportamiento se logra interceptando todos los paquetes, independientemente del destino de los mismos (dirección IP, puerto TCP/UDP), negando de esta forma conectividad al cliente. Una vez que el cliente abre el navegador, se redirige el mismo a una página web que puede presentar diferentes mensajes, desde una página que pide la aceptación de las políticas de uso, hasta una página que requiera la autenticación o el pago para usar el servicio.

1 *Hyper Text Transfer Protocol (HTTP),* es un protocolo de comunicaciones usado para transferir información en la *World Wide Web.*

La utilización de un *Portal Cautivo* permite la identificación del usuario y posibilita el brindar conectividad a un rango amplio de dispositivos que pueden ejecutar un navegador; sin embargo, no son capaces de conectarse vía *WPA o WPA2.*

Para este método se eligió la implementación de *Chillispot-v1.1*[1]. Este software libre presenta las siguientes características:

- Permite la autenticación a través de un *Servidor RADIUS*, usando dos métodos, *UAM (Universal Access Method)* que es el método tradicional de los *Portales Cautivos,* y permite el uso de *WPA* a través del *AP.*

- Permite el uso de los *atributos RADIUS WISPr* definidos por la *WiFi Alliance.*

- Posee servicio de *DHCP.*

- Puede actuar como *Proxy-RADIUS* para otros métodos de autenticación y, si se combina con el método *WPA-HOSTAPD*, permite al servidor *DHCP* entregar una dirección IP predefinida para un cliente en los *atributos RADIUS*.

- Posee soporte de atributos RADIUS específicos que permiten control de Ancho de Banda y la definición de una cantidad máxima de tráfico utilizable por cada cliente.

- No posee requerimientos a nivel de *AP,* salvo el soporte de *WPA o WPA2,* si desea usarse.

1 http://www.chillispot.org/

- Puede utilizarse para autenticar usuarios en redes cableadas.

La implementación de este método se realiza en el *Servidor de Autenticación* de la red inalámbrica y adicionalmente sólo requiere de un servidor web con soporte de *HTTPS*[1] para realizar la autenticación de usuarios al *Servidor RADIUS*.

Chillispot tiene dos componentes principales,

- una aplicación de en espacio de usuario denominada *chilli* que es el *Portal Cautivo* en sí mismo, y cumple las funciones de *Servidor DHCP, Cliente RADIUS, Proxy-RADIUS, y Redirector*;

- un archivo *CGI*[2] en el servidor web, denominado *hotspotlogin.cgi,* el cual es un script, programado en *perl,* que se encarga de enviar los datos de autenticación a *chilli.* Este script, genera un *desafío-CHAP* para validar el usuario y la clave de acceso, que ha recibido del cliente, a través del servidor web cifrado con HTTPS, y envía el mismo a *chilli*.

Las funciones que cumple dentro del modelo propuesto esta aplicación, están ligadas a dos de los métodos de autenticación, *Portal Cautivo y WPA/RSN-Hostapd.* Esto se debe principalmente a las características que que posee ya que además de *Portal Cautivo* cumple las siguientes funciones:

---

1 *Hyper Text Transfer Protocol Secure,* Protocolo de Transferencia de Hyper Texto Seguro, usado para transportar el servicio usualmente denominado *WEB* Segura, cifra el tráfico a través de *SSL Secure Socket Layer.*

2 *Common Gateway Interface,* Interface de Gateway Común, uno de las primeras formas de ejecutar un script en un servidor web.

- *Proxy-RADIUS* para el *método WPA/RSN-Hostapd* que, al interceptar los paquetes RADIUS, puede modificar el comportamiento de algunos servicios que *cumple Chillispot.*

- *Servidor DHCP* para el *método WPA/RSN-Hostapd,* con entrega dinámica y fija a través de un *atributo RADIUS*.

- *Generador de Registro (Accounting).* Si bien *Hostapd* puede generar la información de registro que debe enviarse al servidor RADIUS, *Chillispot* lo hace automáticamente tanto para clientes del *Portal Cautivo* como para clientes *WPA/RSN.*

- *Controlador de Ancho de Banda* a través de *atributos RADIUS,* tanto para clientes del *Portal Cautivo* como para clientes *WPA/RSN.*

- *Controlador de Cantidad de Tráfico* a través de *atributos RADIUS.,* tanto para clientes del *Portal Cautivo* como para clientes *WPA/RSN.*

2. DIRECCIONAMIENTO GENERAL DE RED[1]

Uno de los puntos a definir, dentro de la implementación del modelo de infraestructura, es el direccionamiento IP de la red inalámbrica.

Si bien este tema puede adaptarse a las necesidades particulares de una implementación, se propone lo siguiente:

1 Esta breve sección tiene por objeto presentar un ejemplo del direccionamiento de red, por motivos didácticos, a fin de mejorar la compresión de las secciones subsiguientes.

- Definir un direccionamiento de red distinto al direccionamiento de la red privada para la red inalámbrica. Esto permite mantener separados los mecanismos de asignación de direcciones IP entre las dos redes y por lo tanto mantener la independencia casi completa entre las redes.

- Mantener en el *gateway* de la red privada visibilidad completa de capa de red de los paquetes que atraviesan dicho *gateway.* Esto significa que el gateway de la red inalámbrica enruta los paquetes a su destino sin modificarlos, es decir sin realizar *NAT*[1]. Como consecuencia de esto, en el *gateway* a *Internet* de la red privada se puede filtrar el tráfico, controlar ancho de banda, etc. Esto requiere modificar las tablas de *routing* del *gateway* de la red privada para extender el alcance a la red inalámbrica.

- Definir un direccionamiento IP totalmente diferente para los dispositivos que componen la red de distribución. Esta separación tiene sentido dado que dicha red es una red con *funciones administrativas*. No debe debería tener alcance desde la red inalámbrica ni desde la red privada, salvo para los administradores de red.

- Dividir el direccionamiento IP de la red inalámbrica en dos partes, una para asignación de direcciones IP dinámicas y otra para asignación de direcciones IP estáticas por usuario a través del atributo *Framed-IP-Address* de *RADIUS.*

1 *Network Address Translation,* Traducción de direcciones de red.

Como resultado en el prototipo se propondrá el siguiente direccionamiento IP:

<table>
<tr><th>Redes</th><th colspan="3">Direccionamiento IP</th></tr>
<tr><td>Red Administrativa
Sistema de Distribución</td><td colspan="3">10.10.10.0/24</td></tr>
<tr><td rowspan="4">Red Inalámbrica</td><td colspan="3">192.168.176.0/21</td></tr>
<tr><td>Direcciones Asignadas por Usuario</td><td colspan="2">Direcciones Asignadas Dinámicamente</td></tr>
<tr><td rowspan="3">192.168.176.0/22</td><td colspan="2">192.168.180.0/22</td></tr>
<tr><td>Pool PPPoE</td><td>Pool Portal Cautivo</td></tr>
<tr><td>192.168.180.0/23</td><td>192.168.182.0/23</td></tr>
</table>

Tabla IV.2.: Esquema de direccionamiento IP propuesto para prototipo.

3. SERVIDOR DE AUTENTICACIÓN

Este es el componente principal del modelo propuesto, ya que independientemente de los métodos de acceso que se implementen, se encarga de aplicar los permisos de conectividad a los usuarios.

1. SERVIDOR LDAP

El *Servidor LDAP* será el soporte de almacenamiento de la base de datos de usuarios, permisos definidos por política de acceso y atributos o datos del *servidor RADIUS*. En este prototipo se utilizó la implementación de software libre *OpenLDAP-v2.3.25*[1]. En el árbol *LDAP* no se almacenarán los datos de *Registro o Accounting* del *Servidor RADIUS*, debido a que representan un gran volumen de datos a almacenar, por lo que estos datos se almacenarán en otro soporte.

1 http://www.openldap.org

La elección de almacenar todos los datos en un árbol *LDAP*, permite que todos los datos necesarios para la operación de este modelo estén auto-contenidos e inclusive puedan utilizarse para validar a los mismos usuarios en otras aplicaciones.

Es por ello que en este prototipo se han implementado los *esquemas* necesarios para que también un servidor de correo (*Qmail-LDAP*[1]) pueda obtener los datos necesarios para operar, demostrando que se puede simplificar tanto la administración de la base de datos de usuarios, como la operación para los usuarios, que deberán recordar sólo un nombre de usuario y una clave, para operar en varios ámbitos diferentes.

```
......
# Schema and objectClass definitions
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema
include         /etc/ldap/schema/samba.schema
include         /etc/ldap/schema/radius.schema
include         /etc/ldap/schema/qmail.schema
.......
```

Texto IV.1.: Esquemas utilizados en Servidor LDAP

Los esquemas implementados son los siguientes, como se puede ver en el archivo de configuración */etc/ldap/slapd.conf:*

Junto a los *esquemas básicos,* necesarios para el funcionamiento, se pueden observar tres *esquemas de uso específico:*

1 http://www.qmail.org

- *Samba.schema:* Provee del atributo *SambaNTPassword* necesario para poder realizar la autenticación vía *MSCHAPv2*.

- *Radius.schema:* Provee de los atributos donde se almacenarán todos los datos necesarios para que funcione el *Servidor RADIUS.* Este esquema se provee a través del paquete *freeradius-1.1.3-3.*

- *Qmail.schema (Opcional):* Provee los atributos necesarios para el funcionamiento de un servidor de correo electrónico *Qmail-LDAP.* Sólo se muestra como ejemplo.

Para contener los datos necesarios para el funcionamiento del modelo, es necesario crear la unidad organizativa *Profiles (ou=profiles)*, donde se almacenarán los objetos que representan un perfil de conectividad. Cada uno de los objetos dentro de la *ou=profiles,* representa la aplicación de una *política de conectividad* específica para un grupo de usuarios.

Los objetos que representan un perfil de usuario pertenecen a las siguientes clases:

- **RadiusObjectProfile**
  - OID: 1.3.6.1.4.1.3317.4.3.2.2
  - Tipo: Estructural
  - Descripción: Es una clase de objeto contenedor que se utiliza para crear objetos radiusprofile.
- **radiusprofile**
  - OID: 1.3.6.1.4.1.3317.4.3.2.1
  - Tipo: Auxiliar.

- Descripción: Es una clase de objeto que posee los atributos necesarios para definir un perfil estándar de *RADIUS.*

- **top**
  - OID: 2.5.6.0
  - Tipo: Abstracto

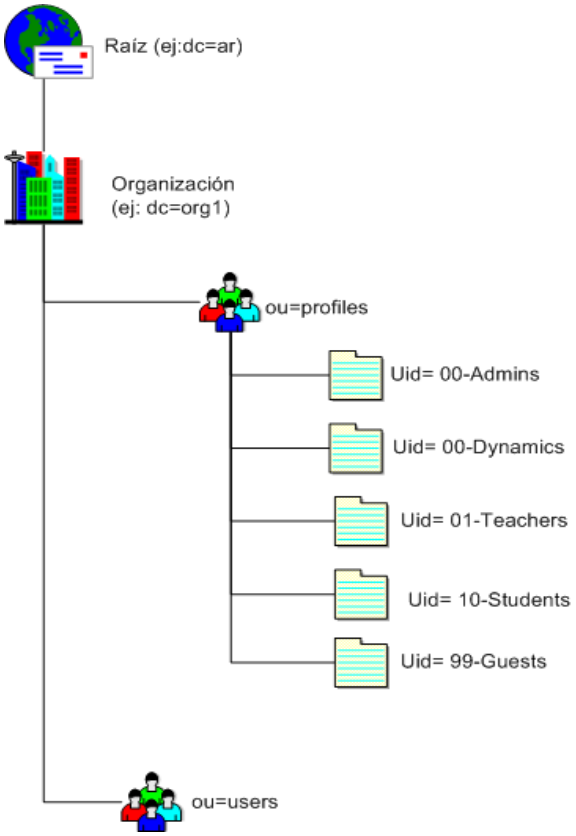

Ilustración IV.4.: Árbol LDAP Parcial, Unidad Organizativa Profiles.

Estos objetos poseen los atributos para almacenar un perfil *RADIUS* básico.

Para almacenar los datos referentes a los permisos de cada perfil de usuarios se sobrecargó el atributo *Description (OID: 2.5.4.13*). Este atributo soporta valores múltiples y soporta una longitud máxima de 1024 caracteres. Los valores de los permisos se almacenan en forma de reglas, definiendo una regla para cada valor del atributo. Dichas reglas deben usar el formato de la sintaxis del comando *iptables de netfilter*[1].

```
........
dn: uid=00-admins,ou=profiles,dc=um,dc=edu
objectClass: radiusObjectProfile
objectClass: radiusprofile
objectClass: top
radiusFramedMTU: 1460
radiusFramedProtocol: PPP
radiusServiceType: Login
dialupAccess: yes
radiusFilterId: 50
uid: 00-admins
cn: 00-admins
description: -N marca-admins -t mangle
description: -A marca-admins -t mangle -j MARK --set-mark 50
description: -A PREROUTING -t mangle -s 192.168.176.0/26 -m mark --mark
0 -j marca-admins
description: -N filtra-admins
description: -A FORWARD -m mark --mark 50 -j filtra-admins
description: -A filtra-admins -j ACCEPT
```

Texto IV.2.: *Ejemplo de definición de Perfil de Usuario*

Los nombres de los perfiles de usuarios tienen la forma *nn-nompolítica*, dónde *nn* es un número que definirá el orden de aplicación del conjunto de reglas y *nompolítica* hace referencia al nombre del grupo

1 http://www.netfilter.org

de usuarios al que se le aplicarán las politicas de acceso. El significado de los atributos *RADIUS*, se verá en la sección *Servidor RADIUS.*

En el cuadro de Texto IV.2 se puede observar un ejemplo de la definición de uno de los *Perfiles de Usuario,* en donde se ve un ejemplo claro de la sobrecarga del atributo *Description.*

La vinculación de un usuario con un perfil de usuario se realiza a través del atributo *RadiusProfileDn* en el objeto del usuario, que debe contener el *dn* del *perfil de usuario*.

```
dn: cn=test user2,ou=um.edu.ar,dc=um,dc=edu
cn: test user2
uid: test.user2
sambaSID: 22309883
userPassword:: e0NSWVBUfVdxeTRTeU9zR2ptR1E=
radiusProfileDn: uid=01-profesores,ou=profiles,dc=um,dc=edu
dialupAccess: yes
objectClass: person
objectClass: radiusprofile
objectClass: sambaSamAccount
objectClass: top
radiusServiceType: Framed-User
sambaNTPassword: 0CB6948805F797BF2A82807973B89537
```

Texto IV.3.: *Ejemplo de vinculación del Perfil de Usuario a un usuario.*

Los usuarios deben poseer el atributo *sambaNTPassword* para poder autenticarse a través de *MSCHAPv2*, que es utilizado por el método de acceso *PPPoE.*

El *atributo dialupAccess* es un atributo especial específicamente definido en la clase radiusprofile para usarse con soporte LDAP. No existe

como atributo RADIUS cuando se utiliza otro soporte de almacenamiento. La existencia de este atributo en un usuario con valor *yes*[1], significa que dicho usuario tiene autorización para conectarse. En caso de que el mismo no exista, o exista y posea el valor *FALSE*, el servidor RADIUS no continúa con el proceso de autenticación y se deniega el acceso remoto.

2. SERVIDOR RADIUS

La operación de este servidor se lleva a cabo a través de *Freeradius-1.1.3-3*[2] el cual es la implementación de un servidor *RADIUS* en software libre, que posee un amplio soporte del protocolo *RADIUS* y que permite usar diferentes soportes de almacenamiento, entre ellos *LDAP y MySQL.*

A fin de usar como soporte de almacenamiento el *directorio LDAP, freeradius* provee un esquema para *LDAP versión 3* denominado *openldap.schema* y un módulo denominado *rlm_ldap* que se encarga de realizar las tareas de a*utorización y autenticación* contra el directorio *LDAP.*

Los atributos del diccionario *RADIUS* son extensibles a través de los *diccionarios de atributos de los fabricantes* y los atributos definidos en el esquema *LDAP* provisto por *freeradius* contempla sólo los atributos estándar del protocolo. Por estas razones, *freeradius* proporciona una manera de almacenar los valores de diccionarios adicionales en un árbol *LDAP,* a través del archivo */etc/freeradius/ldap.attrmap.*

Este archivo es un “mapa” de las correspondencias entre los atributos del diccionario *RADIUS* y los atributos del directorio *LDAP,* permitiendo de

1 En base a pruebas realizadas, si el atributo posee cualquier valor, *excepto FALSE, el usuario puede autenticarse correctamente.*

2 http://www.freeradius.org/.

esta forma “redefinir” el significado de algunos atributos del directorio *LDAP.*

```
.....
ITEM-TYPE    RADIUS-Attribute-Name      LDAP-Attribute-Name
.....
replyItem     Port-Limit                radiusPortLimit
replyItem     Login-LAT-Port            radiusLoginLATPort
replyItem     Reply-Message             radiusReplyMessage
#Atributos Redefinidos
#      para WPA (No estaban definidos en el ldap.attrmap)
replyItem     Tunnel-Type               radiusTunnelType
replyItem     Tunnel-Medium-Type        radiusTunnelMediumType
replyItem     Tunnel-Private-Group-Id   radiusTunnelPrivateGroupId
#       para Chillispot y WPA (Definidos sobre atributos que no se #
usan en esta implementacion en particular).
#      Intervalo de tiempo de de accounting en segundos
replyItem     Acct-Interim-Interval     radiusArapFeatures

#Atributos de WISPr Dictionary para control de Ancho de Banda
replyItem     WISPr-Bandwidth-Max-Up    radiusArapSecurity
replyItem     WISPr-Bandwidth-Max-Down  radiusArapZoneAccess
```

Texto IV.4.*: Atributos LDAP redefinidos para uso con RADIUS.*

Un ejemplo del uso de esta característica se puede ver en el cuadro de Texto IV.4. En él aparece un ejemplo de algunos atributos redefinidos, que son necesarios para la implementación de este modelo, junto con las definiciones estándar para el esquema provisto *por freeradius.* El archivo *ldap.attrmap*, es usado por el módulo rlm_ldap.

La configuración del servidor se encuentra en el *directorio /etc/freeradius*, y consta de diferentes archivos de configuración. Las directivas principales de configuración se encuentran en el archivo

/etc/freeradius/radiusd.conf. La estructura general de este archivo se puede en ver en el cuadro de Texto IV.5.

```
[Opciones Globales de Configuración]
modules           {  ...
}
authorize         {  ...
}
authenticate      {  ...
}
accounting        {  ...
}
```

Texto IV.5.*: Estructura General del archivo /etc/freeradius/radiusd.conf.*

En la sección *modules* se definen cada uno de los módulos que se utilizarán indistintamente para realizar las tareas de *AAA.* Las secciones a*uthorize, authenticate y accounting*, se completan con el nombre de los módulos que se utilizarán para realizar esa tarea en particular y que se han definido previamente .

Para que el servidor pueda realizar las tareas de *AAA* con los métodos de acceso que forman parte del modelo de infraestructura propuesto, es necesario configurar los siguientes módulos:

- LDAP
- MSCHAP
- PAP
- EAP (TLS, TTLS y PEAP)

La configuración del módulo *LDAP* es la que permite realizar la autenticación contra el directorio LDAP. Se puede observar la configuración de este módulo en el cuadro de Texto IV.6:

```
.....
ldap {
      server = "localhost"
      identity = "cn=admin,dc=um,dc=edu"
      password = OP4h8C982p170
      basedn = "ou=um.edu.ar,dc=um,dc=edu"
      filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"
      base_filter = "(objectclass=radiusprofile)"
      start_tls = no
      profile_attribute = "radiusProfileDn"
      access_attr = "dialupAccess"
      dictionary_mapping = ${raddbdir}/ldap.attrmap
      ldap_connections_number = 10
      password_header = "{CRYPT}"
      password_attribute = userPassword
      timeout = 4
      timelimit = 3
      net_timeout = 1
      compare_check_items = no
}
.....
```

Texto IV.6.*: Configuración del módulo LDAP en el archivo /etc/freeradius/radiusd.conf.*

En la configuración de este módulo se especifican las siguientes directivas de configuración:

Nombre	Significado
Server, identity, y password	Estos son los datos de conexión necesarios para acceder al directorio *LDAP.*
basedn	Ámbito a partir del cual se realizarán las búsquedas en el directorio *LDAP*.
filter	Filtro de búsqueda LDAP, para localizar un objeto de usuario utilizando el nombre suministrado por el cliente durante la autenticación *RADIUS. (*por defecto = "(uid=%u)")
base_filter	Filtro de búsqueda *LDAP* usado como ámbito de búsqueda básico. (por defecto = "(objectclass=radiusprofile)")
start_tls	Cuando el valor de esta opción es *yes*, se inicia una conexión cifrada con *TLS* al directorio *LDAP.* Requiere de la configuración adicional de parámetros de claves y certificados.
profile_attribute y default_profile	El valor de *profile_attribute* define el nombre del atributo en el objeto de usuario que contiene el DN de un objeto radiusprofile que contiene el perfil al que pertenece dicho usuario. El valor *default_profile* contiene DN del perfil por defecto para los usuarios. (por defecto= NULL).
access_attr y access_attr_used_for_allow	El valor *access_attr_used_for_allow* define si el valor *access_attr* es usado para permitir (*yes)* o denegar (*no)* el acceso, por defecto el valor es *yes.* Si el valor *acces_attr* es usado para permitir el acceso: ● Si no existe o el valor es *FALSE,* no permite el acceso remoto. ● Si el valor es *yes (o cualquier otro distinto de FALSE),* permite el acceso remoto. Si el valor *acces_attr* es usado para denegar el acceso: ● Si existe el usuario tiene denegado el acceso. ● Si no existe el usuario tiene permitido el acceso.
dictionary_mapping	Referencia al archivo que contiene las correspondencias entre los atributos *LDAP* y los atributos *RADIUS.* Por defecto el valor es *${raddbdir}/ldap.attrmap*
password_header y password_attribute	Estos valores definen el atributo *LDAP* que contiene la contraseña del usuario por defecto y cual es el cifrado del mismo, esto se utiliza para validar por *CHAP .*
compare_check_items	Este valor determina si se deben comparar los valores extraídos del directorio *LDAP* con los que envía el cliente en el pedido entrante. Por defecto el valor es *no.*
timeout, timelimit, net_timeout y ldap_connections_number	Estos valores determinan los tiempos de espera para el servidor *LDAP*, los tiempos límite que se puede demorar en procesar un pedido y el número de conexiones abiertas

Nombre	Significado
	contra el directorio *LDAP*. Estos parámetros han de establecerse en función de la carga de proceso que tiene una implementación particular.

Tabla IV.3.: Parámetros de configuración del módulo LDAP.

Los parámetros antes mencionados definen el comportamiento del servidor *RADIUS,* con almacenamiento en un directorio *LDAP.*

El módulo *MSCHAP* soporta la autenticación MS-CHAP y MS-CHAP-v2, y es utilizado por los métodos de acceso *PPPoE y WPA con EAP-PEAP.* El cuadro de Texto IV.7 muestra la configuración requerida.

```
.....
mschap {
        authtype = MS-CHAP
        use_mppe = yes
        require_encryption = yes
        require_strong = yes
        with_ntdomain_hack = no
}
.....
```

Texto IV.7.*: Configuración del módulo MSCHAP en el archivo /etc/freeradius/radiusd.conf.*

Las directivas de configuración de este módulo son las siguientes:

Nombre	Significado
use_mppe	Permite la utilzación de *MPPE*, en la autenticación
require_encryption	Si *use_mppe* tiene por valor *yes*, realiza un cifrado moderado.
require_strong	Si *use_mppe* tiene por valor *yes*, siempre requiere claves de 128 bits.
with_ntdomain_hack	Si este parámetro tiene valor yes, en caso

Nombre	Significado
	de que un cliente MS Windows se conecte envíando el usuario en la forma *Dominio/nombre_de_usuario* y el *desafío* tomando la porción del nombre del usuario solamente, corrige el comportamiento incorrecto.

Tabla IV.4.: Directivas de configuración de módulo MSChap en freeradius.

El módulo PAP requiere una configuración muy simple y solamente es necesario definir el esquema de cifrado que se utilizará, como se puede observar en el cuadro de Texto IV.8.

```
.....
        #  clear: Clear text
        #  crypt: Unix crypt
        #    md5: MD5 ecnryption
        #   sha1: SHA1 encryption.
        #  DEFAULT: crypt
pap {
                encryption_scheme = crypt
    }
.....
```

Texto IV.8.*: Configuración del módulo PAP en el archivo /etc/freeradius/radiusd.conf.*

La sección de configuración *EAP* del archivo *radiusd.conf*, se colocarán todas las directivas de configuración para las variantes soportadas de dicho protocolo. En este caso, en que se ha elegido usar *EAP-TTLS y EAP-PEAP*, será necesario configurar también la sección de *EAP-TLS,* ya que ambos protocolos elegidos realizan el intercambio de

paquetes dentro de un túnel TLS y requieren certificados sólo por parte del servidor.

Ambos protocolos se usan en el método de acceso WPA para autenticar los usuarios. El protocolo *EAP-TTLS* se utiliza junto con *PAP* y el protocolo *EAP-PEAP* se usa junto con *MSCHAPv2.*

Los módulos de *freeradius* necesarios para utilizar estas variantes del protocolo *EAP* son *rlm_eap_peap, rlm_eap_ttls y rlm_eap_tls*. Estos

```
.....
eap { default_eap_type = peap
      timer_expire     = 60
      ignore_unknown_eap_types = no
      cisco_accounting_username_bug = no
# Tipos soportados de EAP
#EAP-MD5
md5 {       }
#EAP-TLS
tls { default_eap_type = tls
      private_key_password = Adzp10YH39ksk902
      private_key_file = ${raddbdir}/certs/cert-srv.pem
      certificate_file = ${raddbdir}/certs/cert-srv.pem
      CA_file = ${raddbdir}/certs/root.pem
      dh_file = ${raddbdir}/certs/dh
      random_file = ${raddbdir}/certs/random
      fragment_size = 1024
      include_length = yes    }
#Continuan los tipos soportados de EAP
}
.....
```

Texto IV.9.*: Configuración parcial del módulo EAP en el archivo /etc/freeradius/radiusd.conf.*

módulos no están compilados con el paquete *freeradius-1.1.3* para la distribución *Debian Sarge 3.1*, lo que requiere descargar el archivo fuente y compilarlo [Apéndice A] .

La sección de directivas de configuración del módulo EAP se divide en una sección de configuración común y diferentes subsecciones para cada una de las configuraciones específicas del protocolo *EAP.* Para realizar la configuración de *EAP-TLS,* es necesario crear previamente una *Autoridad de Certificación (CA)* y generar las claves y certificados para el servidor [Apéndice B].

En el cuadro de Texto IV.9 se puede apreciar un fragmento de la sección del archivo de configuración *radiusd.conf,* relativa a *EAP,* en el que se pueden observar los parámetros generales para este protocolo y las subsecciones para *EAP-MD5* y *EAP-TLS*.

El significado de las directivas que se observan es expresado en la siguiente tabla:

Nombre	Significado
Subsección General EAP	
default_eap_type	Como los pedidos *EAP* pueden no contener el tipo de *EAP* utilizado, este valor se utiliza para determinar qué tipo de *EAP* elegir a fin de realizar la autenticación. Si otro módulo *EAP* fija el atributo *EAP-Type*, éste tendrá precedencia sobre el valor por defecto.
timer_expire	El servidor mantiene una lista que relaciona los paquetes *EAP-Request/Response*. Este valor determina el tiempo que se debe esperar para borrar una entrada en la lista.
ignore_unknown_eap_types	Debido a que el servidor tiene soporte para una cantidad limitada de variantes del protocolo *EAP*, si recibe un tipo que no soporta, deniega el acceso. Si esta directiva

Nombre	Significado
	tiene por valor *yes* y se ha definido otro módulo con un proxy que soporte el tipo, se reenvía la petición al proxy.
cisco_accounting_username_bug	Soluciona el bug de Cisco AP1230B fw.12.2(13)JA1, que agrega un byte más al atributo *user-name* en la aceptación a la petición de acceso.
Subsección EAP-MD5	
MD5	Sólo con colocar el nombre, sin parámetros, se configura el uso de este módulo *EAP.*
Subsección EAP-TLS	
default_eap_type	Este valor fija el tipo *EAP* a *EAP-TLS.*
private_key_password	Este valor es la contraseña de la clave privada del servidor.
private_key_file	Este valor es la ubicación del archivo de clave privada del servidor.
certificate_file	Este valor es la ubicación del archivo del certificado del servidor.
CA_file	Este valor es la ubicación del archivo del certificado de la Autoridad Certificante, que firma los certificados emitidos.
dh_file	Este valor es la ubicación del archivo de clave Diffie-Hellman, denominado *dh.*
random_file	Este valor es la ubicación de un archivo con valores aleatorios denominado *random.*
fragment_size	Este valor no debe exceder los 4096 bytes de un paquete *RADIUS,* en la mayoría de los AP la máxima longitud de un paquete es de 1500-1600 bytes, por lo que el tamaño de un fragmento debería ser de 1024 bytes o menor.
include_length	Si esta directiva tiene por valor *yes,* se incluye en cada paquete el tamaño total del mensaje (valor por defecto). Si el valor es *no*, sólo se incluye el tamaño total del mensaje en el primer paquete.

Tabla IV.5.: Directivas de configuración del módulo EAP, EAP-MD5 y EAP-TLS.

Una vez definidas las directivas de *EAP-TLS*, se deben configurar las directivas para *EAP-TTLS y EAP-PEAP,* como se observa en el cuadro de Texto IV.10, el significado de cada directiva puede observarse en la tabla subsiguiente.

```
.....
eap {
.....
      ttls { default_eap_type = md5
             copy_request_to_tunnel = yes
             use_tunneled_reply = yes       }

       peap { default_eap_type = mschapv2
              copy_request_to_tunnel = yes
              use_tunneled_replay = yes
              proxy_tunneled_request_as_eap = yes     }
      mschapv2 {     }
   }
...
```

Texto IV.10.*: Configuración parcial del módulo EAP de freeradius.*

Nombre	Significado
Subsección *EAP-TTLS*	
default_eap_type	La sesión *EAP* dentro del túnel TLS necesita tener definido un tipo de *EAP* de forma separada a la sesión externa al túnel. (Recordar que EAP-TTLS puede describirse como una sesión *EAP,* dentro de una sesión Diameter, dentro de un túnel TLS, dentro de una sesión *EAP*, dentro de una solicitud *RADIUS).*
copy_request_to_tunnel	La solicitud de *autenticación* dentro del túnel TLS no contiene todos los atributos *RADIUS*, sólo los necesarios para realizar

Nombre	Significado
	la *autenticación*, si este parámetro tiene por valor *yes*, los atributos que no están presentes en la solicitud *interna* al túnel y si lo están fuera de él, se copian en la solicitud *interna*.
use_tunneled_reply	Los atributos enviados en la respuesta al *NAS* están basados en el nombre de usuario externo al túnel. Generalmente el valor es *anonymous.* Para que se envíen al *NAS* los atributos basados en la sesión *EAP* interna al túnel, este parámetro debe tener valor *yes.*
Subsección *EAP-PEAP*	
default_eap_type	*Idem EAP-TTLS*. Por defecto debe tener el valor *mschapv2* soportado por los clientes con sistema operativo MS Windows.
copy_request_to_tunnel	Idem *EAP-TTLS*.
use_tunneled_reply	Idem *EAP-TTLS*.
proxy_tunneled_request_as_eap	Si la sesión dentro del túnel, utiliza un proxy de *RADIUS*, el servidor que realiza la autenticación, puede no soportar *EAP-MSCHAPV2*. Si el parámetro tiene por valor *no*, el proxy de *RADIUS* reenvía los paquetes como si fuese una sesión normal de MSCHAPv2.
Subsección *EAP-MSCHAPV2*	
MSCHAPv2	Sólo con colocar el nombre, sin parámetros, se configura el uso de este módulo *EAP-MSCHAPv2.* Requiere que se haya configurado el módulo *mschap.*

Tabla IV.6.: Directivas de configuración del módulo EAP-TTLS, EAP-PEAP y EAP-MSCHAPv2

El registro *o accounting* de uso de los recursos utiliza como soporte de almacenamiento un servidor de bases de *datos* MySQL. Por esto es necesario completar las directivas de configuración para el módulo *rlm_sql_mysql* que utiliza el servidor *RADIUS* para acceder al motor de bases de datos.

La sección de configuración de este módulo se encuentra en el archivo */etc/freeradius/sql.conf*, que se incluye en el *archivo* radiusd.conf. En dicha sección hay que modificar los parámetros de conexión al motor de bases de datos. Por defecto están definidas las consultas que realizan las tareas de registro.

También se realiza la configuración de un parámetro que esta relacionado con la política de uso definida. Este parámetro utiliza los registros de utilización, para chequear en el proceso de *autenticación* si existe una sesión activa para la cuenta de usuario.

```
.....
sql {
        driver = "rlm_sql_mysql"
        server = "localhost"
        login = "radius"
        password = "radius"
      ......
      # Eliminar comentario de la siguiente línea
      simul_count_query = "SELECT COUNT(*) FROM ${acct_table1}
WHERE UserName='%{SQL-User-Name}' AND AcctStopTime = 0"
   }
...
```

Texto IV.11.*: Configuración parcial del módulo rlm_sql_mysql de freeradius.*

Además se realiza un chequeo de la existencia y el valor del atributo *Simultaneous-Use*, y dependiendo del valor del mismo, se permite o no un número predefinido de sesiones simultáneas.

Otra posibilidad de realizar el chequeo para utilización simultánea es utilizar un módulo denominado *radutmp*, que realiza el mismo chequeo

pero utilizando un archivo tipo *utmp* donde registra los inicios de sesión y las terminaciones de sesión. De esta forma obtiene la información de que usuarios tienen una sesión iniciada en el sistema. La configuración del mismo es bastante sencilla, como se puede observar en el cuadro de Texto IV.12. Se recomienda realizar esta tarea a través del motor *Mysql* por razones de rendimiento.

```
...
radutmp {    filename = ${logdir}/radutmp
             username = %{User-Name}
             case_sensitive = yes
             check_with_nas = yes
             perm = 0600
             callerid = "yes"
          }.......
```

Texto IV.12*.: Configuración opcional del módulo radutmp, para chequeo uso simultáneo.*

La sección *authorize* del archivo *radiusd.conf* se configura colocando los nombres de los módulos definidos en las secciones anteriores. Es importante el orden en que se colocan los módulos.

El módulo *eap* debe ser el primer módulo de autorización, luego de los módulos *preprocess y auth_log* que realizan los procesos para evitar las ambigüedades en los nombre de usuario y el registro de los pedidos de autorización. Puede observarse la configuración de esta sección en el cuadro de Texto IV.13.

```
...
authorize {
            preprocess
            auth_log
            eap
            mschap
            ldap
          }.......
```

Texto IV.13.*: Sección de autorización.*

La sección *autheticate* se configura listando los módulos disponibles para realizar el proceso de autenticación. Dicho proceso no intenta autenticar probando con cada módulo listado, sino que el módulo usado en la sección *authorize* agrega un atributo de configuración denominado *Auth-Type.* Ese tipo de autenticación se utiliza para elegir el módulo apropiado de la lista definida. En un caso particular se puede definir en el archivo de configuración el atributo *Auth-Type*, a fin de forzar un tipo de autenticación predefinida, pero es posible que dejen de funcionar algunos de los módulos de autenticación.

```
...
authenticate {
            pap
            Auth-Type MS-CHAP {
                                        mschap }
            Auth-Type LDAP    {
                                        ldap  }
            eap
          }.......
```

Texto IV.14.*: Sección de autorización.*

Puede observarse la configuración de esta sección requerida para el funcionamiento del modelo de infraestructura planteado. En dicha configuración se han fijado dos valores de *Auth-Type* predefinidos, ya que para que se complete el proceso de autenticación del método de acceso *PPPoE* resultó necesario definir al módulo MS-CHAP como método de autenticación por defecto.

Las secciones a continuación están relacionadas en la instanciación de algunos módulos. Dichas secciones son *accounting y session.* La sección *accounting* se encarga de instanciar los módulos que realizarán el almacenamiento de los registros de utilización, y la sección *session* se encarga de instanciar los módulos que realizarán el control sobre el uso simultáneo de una cuenta de usuario, descripto anteriormente.

```
...
accounting {      sql
            }
session     { sql
              #radutmp
            }
.......
```

Texto IV.15.*: Secciones de registro y control de sesión.*

Terminada la configuración principal del servidor *RADIUS*, se deben modificar dos archivos adicionales que definen los *NAS* que pueden comunicarse al servidor y los diccionarios de atributos a utilizar por el servidor.

El archivo que define los *NAS* es */etc/freeradius/clients.conf*, en dicho archivo se deberá consignar desde donde se podrá acceder al servidor y con que contraseña, como se puede observar en el cuadro de Texto IV.16,

```
client 10.10.10.0/24 {  secret = Sqa439fG
                        shortname = Red-Sistema-Distribucion
                        nastype = other}
client 127.0.0.1     {  secret = Sqa439fG
                        shortname = localhost
                        nastype = other   }
.......
```

Texto IV.16.*: Configuración del archivo clients.conf de freeradius.*

se puede consignar una red completa. En este caso se deberá configurar la red del *sistema de distribución* y al propio *servidor de autenticación*.

El archivo */etc/freeradius/dictionary* es el diccionario de atributos principal. En él deben realizarse todas las modificaciones de los atributos y las inclusiones de los archivos de diccionarios externos. Para un correcto funcionamiento del modelo de infraestructura, deben estar presentes los diccionarios que figuran en el cuadro de Texto IV.17.

```
....
$INCLUDE        /usr/share/freeradius/dictionary
$INCLUDE        /usr/share/freeradius/dictionary.chillispot
$INCLUDE        /usr/share/freeradius/dictionary.wispr
$INCLUDE        /usr/share/freeradius/dictionary.merit
$INCLUDE        /usr/share/freeradius/dictionary.microsoft
....
```

Texto IV.17.*: Configuración del diccionario de atributos principal.*

Cada uno de los diccionarios tiene las definiciones de los atributos y el tipo de los mismos, para atributos de un fabricante dado. El diccionario ubicado en */usr/share/freeradius/dictionary* incluye la mayoría de los diccionarios de los fabricantes.

Una vez realizada la configuración del servidor *RADIUS*, se deberá configurar un cliente *RADIUS*, cuya implementación en Linux se llama *radiusclient1.* Este cliente *RADIUS* permite que los diferentes servidores de los métodos de acceso, por ejemplo el servidor *PPPoE*, puedan realizar la autenticación, de los usuarios usando el servidor *RADIUS.*

Los archivos de configuración de dicho cliente se encuentran en el directorio */etc/radiusclient*. Hay que editar al menos dos de ellos, *servers y dictionary.*

El primero de ellos define la dirección del servidor *RADIUS* y la contraseña para acceder al mismo, como se puede observar en el cuadro de Texto IV.18.

```
#Server Name or Client/Server pair              Key
#----------------                            ---------------
localhost/localhost                             Sqa439fG
```

Texto IV.18.*: Configuración del archivo servers de radiusclient.*

El segundo es el diccionario de atributos que utiliza el cliente *RADIUS.* Éste es similar al archivo de diccionario principal del servidor y la principal diferencia con el mismo es la sintaxis para incluir un diccionario externo.

En el diccionario de *radiusclient* debe utilizarse la directiva *INCLUDE* en vez de *$*INCLUDE, tal como se observa en el cuadro de Texto IV.19.

En dicho cuadro también pueden observarse los diccionarios necesarios para la operación del servidor *PPPoE* en el modelo de infraestructura propuesto.

```
.......
INCLUDE /etc/radiusclient/dictionary.merit
INCLUDE /etc/radiusclient/dictionary.microsoft
INCLUDE /usr/share/freeradius/dictionary.chillispot
```

Texto IV.19*.: Configuración del archivo dictionary de radiusclient.*

3. PORTAL CAUTIVO (CAPTIVE PORTAL)

La implementación del portal cautivo bajo *GPL*[1] que se eligió se llama *Chillispot*[2]. *Chillispot* posee una implementación sencilla y auto-contenida, es decir, depende de muy pocos servicios externos para funcionar. Las principales características de *Chillispot* son las siguientes:

- Soporta dos métodos de autenticación distintos *UAM (Universal Access Method) y WPA/RSN (Wireless Protected Access).*

- Soporta Autenticación, Autorización y Registro (AAA) contra un *servidor RADIUS.*

- Posee un servicio de *DHCP* propio para ambos métodos de autenticación. Con *WPA/RSN* soporta asignación de direcciones *IP* a través de atributos RADIUS, funcionando como proxy.

1 *GNU GPL, General Public License.* ver http://www.gnu.org.

2 http://www.chillispot.org

- La autenticación con *UAM* soporta *SSL.*

- Soporta cualquier *AP.*

- Soporta *scripts* de inicio y final de conexión para cada cliente.

- Funciona utilizando NAT o Routing.

- Realiza diferentes controles a través de los valores de los atributos *RADIUS* de los diccionarios *wispr y chillispot.* Esto permite controlar los volúmenes de tráfico entrante/saliente de un usuario en particular, realizar controles de ancho de banda, etc.

Chillispot necesita de un servidor *WEB con soporte SSL* y de un servidor *RADIUS.*

Cuando utiliza el método de autenticación *UAM*, entrega una dirección IP a través del protocolo *DHCP,* pero cuando el cliente intenta conectarse, redirecciona el tráfico hacia el servidor *WEB.*

Esto sucede hasta que el cliente inicia un navegador, y al intentar realizar un acceso se redirecciona a una página segura específica del servidor *WEB* que posee un *link* hacia la página de autenticación. Una vez que el cliente ha ingresado su usuario y contraseña, un *script* en el servidor *WEB*, cifra la contraseña y la envía a programa *Chilli*, que realiza la autenticación contra el servidor *RADIUS.* Una vez que el usuario se ha autenticado correctamente se deja de redireccionar el tráfico del cliente, que funciona normalmente.

En caso de usar el método de autenticación *WPA/RSN*, el *AP* oficia de *autenticador WPA/RSN,* el cliente entabla una sesión a través de *EAPOL* con el *AP,* que tiene configurado como servidor *RADIUS* a *Chillispot*.

Chillispot en este caso oficia de *Proxy RADIUS,* reenviando el pedido de acceso al servidor *RADIUS*, quien ejecuta el proceso de autenticación. En caso de que la autenticación sea satisfactoria, cuando *Chillispot* recibe el mensaje de acceso aceptado, si dicho mensaje contiene el atributo *Framed-IP-Address*, asigna al cliente dicha dirección IP a través del servidor *DHCP interno.*

Chillispot actualmente tiene dos versiones, la versión estable 1.0 y un a versión de desarrollo 1.1. Una de las diferencias entre las versiones son los parámetros que permiten instanciar un *script* cada vez que un cliente inicia o finaliza una conexión. Esta es una característica que se utiliza junto con el *script wifi-policy*, para aplicar permisos, como se ve en dicha sección. La distribución para *DEBIAN* solo brinda la versión estable

```
.......
#Red a la que se da servicio
net 192.168.176.0/21
#Red que se utiliza para direccionamiento dinámico.
dynip 192.168.182.0/23
#Red que se utiliza para direccionamiento estático
statip 192.168.176.0/22
#Interfaz DHCP
dhcpif eth1
.......
```

Texto IV.20.*: Fragmento de chilli.conf -Configuración de redes.*

Chillispot-1.0.0, por lo que se debe compilar una versión posterior,

proceso en el que se han presentado algunas particularidades y se desarrolla en el Apéndice C.

Una vez que se ha instalado *chillispot* en la versión requerida, se debe proceder a configurar la aplicación para que funcione con los parámetros que se usan en el modelo de infraestructura, editando el archivo */etc/chilli.conf.*

Los primeros parámetros a definir son los parámetros de la red, incluyendo cuál es la red a la que *chillispot* brinda servicio *(net),* y qué porción de la misma utiliza direccionamiento dinámico (*dynip*) y fijo (*statip*) respectivamente. Se debe definir la interfaz en que escucha el servidor *DHCP interno (dhcpif).* Adicionalmente se pueden definir los servidores *DNS,* si fuesen diferentes a los que tiene configurada el propio servidor.

```
.......
# Parámetros del Servidor RADIUS
radiuslisten 192.168.0.100
radiusserver1 127.0.0.1
#radiusserver2 127.0.0.1
radiusauthport 1812
radiusacctport 1813
radiussecret Sqa439fG
radiusnasid mti-server
#Parámetros para Proxy RADIUS
proxylisten 10.10.10.1
proxyport 1645
proxyclient 10.10.10.0/24
proxysecret Sqa439fG
......
```

Texto IV.21.*: Fragmento de chilli.conf -Configuración de RADIUS*

En el archivo de configuración también se definen los parámetros relacionados con el servidor *RADIUS* al que se debe consultar, como se observa en el cuadro de Texto IV.21, en la tabla subsiguiente se encuentran las definiciones de cada parámetro usado.

Nombre	Significado
Parámetros para comunicarse con el *servidor RADIUS*	
radiuslisten	Dirección IP por la cual se comunica el servidor *RADIUS.* En este caso es la interfaz que pertenece a la red privada del Servidor de Autenticación.
radiusserver1 *radiusserver2*	Dirección IP del servidor *RADIUS,* primario y secundario, en este caso ese valor es *localhost* 127.0.0.1
radiusauthport	Número de puerto UDP donde el servidor *RADIUS* recibe los paquetes destinados a procesos de autenticación. Por defecto es el puerto 1812.
radiusacctport	Número de puerto UDP donde el servidor *RADIUS* recibe los paquetes destinados a procesos de registro. Por defecto es el puerto 1813.
radiussecret	Contraseña para acceder al servidor *RADIUS.* Debe ser la misma que se ha configurado en el archivo *clients.conf* del servidor.
radiusnasid	Es el nombre que se ha elegido para que el servidor *chillispot* se identifique frente al servidor *RADIUS.*
Parámetros para actuar como *Proxy RADIUS*	
proxylisten	Dirección IP en la que se espera recibir los pedidos de autenticación cuando el servidor *chillispot* oficia como *Proxy RADIUS.*
proxyport	Número de puerto UDP donde se espera recibir los pedidos de autenticación. Debe ser diferente al puerto definido en *radiusauthport.*
proxyclient	Debe ser la red desde la que se permite el acceso para pedidos de autenticación. En este caso el la red del sistema de distribución. 10.10.10.0/24.
proxysecret	Contraseña que se compartirá con los clientes que contactarán a *chillispot* como si fuera un servidor *RADIUS.*

Tabla IV.7.: Directivas de configuración RADIUS en Chillispot.

Para configurar el método de acceso *UAM*, es necesario definir previamente la configuración del servidor *WEB* donde se aloja la página inicial de bienvenida y es necesario copiar al servidor web el script *CGI* que se encuentra en */usr/lib/cgi-bin/hotspotlogin.cgi.*

```
.......
uamserver https://192.168.176.1/cgi-bin/hotspotlogin.cgi
uamhomepage https://192.168.176.1/chilli/index.html
uamsecret ASDFG12340987
#uamlisten 192.168.182.1
#uamport 3990
#uamallowed 192.168.0.100
#uamanydns
.......
```

Texto IV.22.*: Fragmento de chilli.conf -Configuración UAM.*

Nombre	Significado
uamserver	URL del script del servidor *WEB* que realiza el manejo de la autenticación de usuarios.
uamhomepage	URL de la página de inicio o bienvenida a la que se redireccionarán los clientes no autenticados.
uamsecret	Este parámetro tiene que definirse también en el script *hotspotlogin.cgi* del servidor *WEB*. Se utiliza para cifrar la contraseña del usuario cuando se envía desde el servidor *WEB* hacia *chillispot.*
uamlisten	Este parámetro permite definir la dirección IP donde se escucharán las solicitudes de autenticación. Por defecto es la primera dirección IP del segmento definido en el parámetro *net.* En la configuración propuesta no debe estar activado.
uamport	Este parámetro establece el puerto TCP donde se escucharán las solicitudes de autenticación. Por defecto es el puerto 3990. En la configuración propuesta no debe estar activado.

Nombre	Significado
uamallowed	Esta directiva permite múltiples valores, y los parámetros pueden ser direcciones IP, segmentos de red o nombres de dominio. Si se ha definido los usuarios no autenticados podrán acceder libremente a las direcciones establecidas. En la configuración propuesta debe estar desactivado.
uamanydns	Si este parámetro no está comentado, todos los clientes, aún los no autenticados podrán acceder a cualquier servidor *DNS* externo. No es recomendable activarlo, ya que puede ser una vulnerabilidad de seguridad.

Tabla IV.8.: Directivas de configuración UAM en Chillispot

Además se deberá configurar dos parámetros adicionales, *conup y condown*, cada uno de estos parámetros apunta a un *script* diferente. Cuando un usuario sea autenticado satisfactoriamente se ejecuta el *script* del parámetro *conup,* y cuando el usuario se desconecte se ejecuta el *script* del parámetro *condown.* Las funciones que cumplen los *scripts* se ven más adelante en la sección Herramientas para Aplicación de Políticas**.**

Por último se debe editar el archivo *hotspotlogin.cgi*, dicho script está programado en *Perl*, y su función principal es solicitar el nombre de usuario y la contraseña al cliente, cifrar la contraseña y enviar estos datos al servidor *chilli.*

```
#!/usr/bin/perl
.......
$uamsecret = "ASDFG12340987";
$userpassword = 1;
.......
```

Texto IV.23.*: Fragmento de hotspotlogin.cgi -Configuración chillispot.*

El parámetro *$uamsecret* debe ser el mismo que se configuró en *chilli.conf.* El parámetro *$userpassword* debe configurarse con el valor 1, y esto significa que se realiza la autenticación usando el atributo *User-Password* de un servidor *RADIUS.*

Este *script* puede personalizase o sustituirse por otro que cumpla las mismas funciones.

4. SERVIDOR WEB

La instalación de un servidor *WEB* en el *Servidor de Autenticación*, obedece a que es necesario tanto para la implementación de un *Portal* Cautivo, como para la implementación de *herramientas de administración.*

Los requerimientos básicos para la instalación de este servidor son el soporte de cifrado *SSL* y el soporte de *PHP*. Lo primero es requisito de *chillispot* y lo segundo es necesario para poder ejecutar herramientas de administración como *PhpLDAPadmin.*

Si bien es posible configurar *chillispot* para que se conecte a otro servidor *WEB* externo, e instalar también las herramientas de administración en otro servidor, el criterio que se ha seguido es el de mantener el núcleo de herramientas y servicios necesario para la operación del modelo de infraestructura, contenido dentro del mismo modelo.

De esta forma se minimiza la cantidad de equipos que interactúan y se evita que la información sensible viaje a través de varios equipos por la red. Como consecuencia de ello, disminuyen los puntos de falla, es más

difícil que la información sea interceptada a través de un *sniffer* de red, y se logra una pequeña ganancia en rendimiento.

5. SERVIDOR PPPOE

Para implementar el método de acceso *PPPoE*, es necesario instalar un servidor *PPP*, el servidor *PPPoE* y configurar ambos para que la autenticación se efectúe contra el servidor *RADIUS.*

Para autenticar a los usuarios de este método de acceso usando el protocolo *RADIUS*, es necesario utilizar un *plugin* denominado *radius.so.* Para poder aplicar las políticas de uso, como se detalla en la sección Herramientas para Aplicación de Políticas, es necesario conocer los valores de los atributos *RADIUS* que envía el servidor al aceptar el acceso remoto del cliente. Para ello se necesita utilizar otro *plugin* denominado *radattr.so.* Dicho *plugin* genera un archivo */var/run/radattr.pppx*, donde *pppx*, hace referencia al nombre de la *interfaz ppp* donde está conectado el cliente (ej ppp0, ppp1, pppx). En el archivo se especifican pares de atributo-valor, extraídos del mensaje de aceptación de acceso que envía el servidor *RADIUS.*

Ambos *plugins* son provistos por la aplicación *pppd*, y permiten que el servidor *PPPoE* efectúe la autenticación de usuarios a través de *PAP, CHAP, MSCHAP y MSCHAPv2*, contra un servidor *RADIUS* usando la aplicación *radiusclient1.* Los detalles de configuración de esta aplicación pueden observarse en la sección Servidor RADIUS.

El método de autenticación de usuarios elegido es *MSCHAPv2*, debido a que posee el nivel de seguridad más alto dentro de los métodos de autenticación soportados por el *plugin radius.so.*

La utilización de *MSCHAPv2*, requiere que en el *directorio LDAP* se almacene el *hash MD4* de la clave del usuario, para esto se utiliza el *atributo LDAP sambaNTpassword,* vinculado al atributo RADIUS *NT-Password.*

Uno de los problemas que se presenta al combinar *PPPoE* con otro de los métodos de acceso, es que el servidor *PPPoE* utiliza un *pool* de direcciones IP propio para la asignación dinámica de direcciones. Para evitar la duplicación de direcciones IP, se ha dividido el rango dinámico de direcciones a entregar en dos partes, una parte administrada por *PPPoE* y otra parte administrada por los otros métodos de acceso. Si el usuario posee el atributo *RADIUS Framed-IP-Address*, se entrega al cliente el valor de dicho atributo como dirección IP, independientemente de que pertenezca al *pool de direcciones* dinámicas o no.

El servidor *PPPoE* se encarga de enviar al servidor *RADIUS* los datos necesarios para el registro de utilización o *accounting.*

Para la implementación del servidor *PPP* se utilizó *ppp-2.4.4rel8*, y para la implementación del servidor *PPPoE* se usó el *rp-pppoe-3.8* de *Roaring Penguin*[1]. En los *kernels* de Linux posteriores al 2.4.0, *rp-pppoe* soporta modo kernel, pero también soporta correr en espacio de usuario. En general hay que compilar la aplicación especialmente para poder ejecutarla en espacio de *kernel,* como se observa en el Apéndice D.

1 Roaring Penguin PPPoE: http://www.roaringpenguin.com/penguin/openSourceProducts/rpPppoe

La aplicación *pppoe-server* usualmente se instancia en la línea de comandos, si bien se recomienda generar un *script* con ese fin. Se debe ejecutar una instancia[2] el servidor *PPPoE* con los siguientes parámetros:

```
# /usr/sbin/pppoe-server -k -s -T 60 -I eth1 -N 509 -C
PPPoE-UM-01 -R 192.168.180.1 -L 192.168.176.1 -S UM-wireless
```

Texto IV.24.*: Ejecución de una instamcia del servidor PPPoE.*

Los parámetros resaltados en el cuadro de Texto IV.24 se explican en la Tabla IV.9.

Nombre	Significado
-k	Esta opción especifica que el servidor debe usar *modo kernel*, por defecto se utiliza el espacio de usuario. Requiere que el kernel se haya compilado con soporte para *PPPoE.*
-N número	Especifica el número máximo de sesiones concurrentes, por defecto es de 64 sesiones. Se debe tener en cuenta el direccionamiento IP que se desea utilizar.
-R dirección IP	Esta opción configura la primera dirección dinámica que asigna el servidor a un cliente remoto. El servidor automáticamente registra las direcciones entregadas y envía una dirección válida al cliente. Si no se especifica, se usa por defecto la dirección 10.67.15.1. Este valor debe relacionarse con el parámetro -N para obtener la red IP a la que el servidor le entregará direcciones dinámicas.
-L dirección IP	Esta opción configura la dirección *IP* local, si no se define, por defecto es 10.0.0.1.

Tabla IV.9.: Parámetros relevantes para iniciar servidor PPPoE.

Al instanciar el servidor desde la línea de comandos, el proceso se bifurca[2] y se ejecuta como *demonio*.

2 Es posible ejecutar múltiples instancias del servidor PPPoE con configuraciones diferentes.

2 Traducción del inglés de *to fork*. Este verbo se utiliza para indicar cuando un proceso crea

Cuando un cliente inicia sesión, el servidor inicia una instancia de *pppd* que se encarga de la comunicación dentro de la *sesión PPPoE.*

La configuración de *pppd* se encuentra en el archivo */etc/ppp/options* y se puede observar en el cuadro de Texto IV.25.

```
ms-dns 192.168.0.100
ms-dns 192.168.0.254
sync
auth
lock
ipcp-accept-local
ipcp-accept-remote
local
noreplacedefaultroute
plugin radius.so
plugin radattr.so
refuse-pap
refuse-chap
refuse-mschap
require-mschap-v2
```

Texto IV.25.*: Configuración de pppd, archivo /etc/ppp/options.*

El servidor *PPPoE* utiliza un archivo de configuración propio. El mismo se ubica en */etc/ppp/pppoe-server-options.* En el archivo es necesario fijar los parámetros de *MTU*[1] y *MRU*[2], debido al *overhead* de *PPPoE* que no permite aprovechar los 1500 bytes de carga máxima que proporciona un marco *ethernet.* Se puede observar el archivo de configuración en el cuadro de Texto IV.26.

una copia de si mismo, denominada *hijo* del proceso original, que se denomina *padre.*

1 *MTU, Maximum Transmit Unit.* Unidad máxima de transmisión en bytes.

2 *MRU, Maximum Receive Unit.* Unidad máxima de recepción en bytes.

```
# PPP options for PPPoE Server
mtu 1460
mru 1460
```

Texto IV.26.*: Configuración de pppoe, archivo /etc/ppp/pppoe-server-options.*

El significado de las opciones resaltadas en el cuadro de Texto IV.25 es desarrollado en la Tabla IV.10.

Opciones	Significado
ms-dns	Esta directiva permite fijar las direcciones IP de los servidores *DNS* que envía el servidor *PPP* al cliente a través del protocolo *IPCP.*
plugin	Esta directiva indica los *plugins* que se deben utilizar. En particular se deben usar *radius.so y radattr.so.*
refuse-pap	Esta directiva indica que el servidor debe rechazar cualquier intento de autenticación a través del método de autenticación *PAP*.
refuse-chap	Esta directiva indica que el servidor debe rechazar cualquier intento de autenticación a través del método de autenticación *MSCHAP*.
refuse-mschap	Esta directiva indica que el servidor debe rechazar cualquier intento de autenticación a través del método de autenticación *MSCHAP*.
require-mschap-v2	Esta directiva indica que el servidor debe requerir a cualquier cliente que se conecta el uso del método de autenticación *MSCHAPv2*.

Tabla IV.10.: Directivas de configuración relevantes de pppd. (/etc/ppp/options)

Para utilizar este método los clientes deberán crear una conexión *PPPoE* en el sistema operativo fijando parámetros de autenticación idénticos a los desarrollados en el *servidor PPPoE.*

6. SERVIDOR DE BASE DE DATOS

La instalación de un motor de bases de datos en el *Servidor de* Autenticación, obedece a la necesidad de almacenar los datos de registro o *accounting* del servidor *RADIUS.*

La elección *de MySQL* se sustenta en que:

- Puede usarse bajo licencia GPL si se utiliza para desarrollar bajo la misma licencia.

- Es compatible con los módulos del modelo de infraestructura que necesitan de un motor de bases de datos.

Además existen aplicaciones de administración *web* como *phpmyadmin,* que simplifican la administración del motor.

Al igual que el servidor *WEB,* si bien es posible configurar un motor de bases de datos externo al servidor de autenticación, se ha utilizado el criterio mantener el núcleo de herramientas y servicios necesario para la operación del modelo de infraestructura, contenido dentro del mismo modelo.

De esta manera se minimiza la cantidad de equipos que interactúan y se evita que la información sensible viaje a través de varios equipos por la red. Esto genera menor cantidad de puntos de falla, mayor dificultad para interceptar la información y una pequeña mejora de rendimiento.

7. HERRAMIENTAS PARA APLICACIÓN DE POLÍTICAS

Este módulo es uno de los pilares del modelo de infraestructura que se propone. El objetivo es aplicar políticas de uso de la red inalámbrica a través de la aplicación de *perfiles de conectividad* a distintos grupos de usuarios.

Se pueden definir tantos *perfiles de conectividad* como se requiera para implementar las políticas de uso.

Para aplicar estas herramientas es necesario definir los diferentes grupos de usuarios, definir las políticas de uso que se desea implementar y modelar los *perfiles de conectividad* que las reflejan y se deben aplicar a los grupos de usuarios.

Estos *perfiles de conectividad* se ubican en la ou=*profiles* del directorio *LDAP.*

El *uid* de cada objeto que define un *perfil de conectividad* tiene la forma xx-*Nombre*, donde *xx* es un número que puede repetirse e indica el orden en que se procesan las reglas de conectividad.

Los objetos que conforman cada *perfil de conectividad* deben pertenecer a dos clases: *radiusObjectProfile y radiusprofile.* Ambas clases son provistas por *radius.schema*.

Puede observarse la definición de un perfil de ejemplo en el cuadro de Texto IV.27.

```
dn: uid=00-admins,ou=profiles,dc=um,dc=edu
objectClass: radiusObjectProfile
objectClass: radiusprofile
objectClass: top
radiusFramedMTU: 1460
radiusFramedProtocol: PPP
radiusServiceType: Login
radiusFilterId: 50
uid: 00-admins
cn: 00-admins
description: -N marca-admins -t mangle
description: -A marca-admins -t mangle -j MARK --set-mark 50
description: -A PREROUTING -t mangle -s 192.168.176.0/26 -m mark
--mark 0 -j marca-admins
description: -N filtra-admins
description: -A FORWARD -m mark --mark 50 -j filtra-admins
description: -A filtra-admins -j ACCEPT
```

Texto IV.27.*: Definición de un perfil de conectividad en el directorio LDAP.*

Los usuarios se vinculan a un *perfil de conectividad* a través del atributo *LDAP radiusProfileDn*. El valor de este atributo debe ser el *DN* del objeto de la clase *radiusObjectProfile* que contiene la definición del *perfil de conectividad.*

Los *perfiles de conectividad* se definen en el *Directorio LDAP* y se componen de dos tipos de directivas:

- *Atributos RADIUS:* Estos atributos cumplen dos roles, definir el "comportamiento" de una conexión e identificar el *perfil de conectividad*. Definen el "comportamiento" a través de parámetros

generales que, por ejemplo, permiten fijar tiempos máximos de conexión en una sesión o tiempos límite de inactividad. Los atributos también identifican el *perfil de* conectividad a través del valor del atributo *Filter-Id.* Éste es utilizado por *las reglas de conectividad* para "*marcar"* e identificar el tráfico de los usuarios que pertenecen a un perfil determinado. Los valores de estos atributos no siempre son interpretados de la misma manera por los servidores que componen los *métodos de acceso.* Por esta razón, las políticas de uso deben contemplar a los *métodos de acceso* como parte integral de las mismas.

- *Reglas de Conectividad:* Las reglas de conectividad permiten modelar los permisos que posee cada grupo de usuarios. Por ejemplo, si pueden acceder o no a un servicio que brinda un *host* particular de la red interna, los servicios de Internet a los que pueden acceder, etc. Para formular dichas reglas se optó por utilizar un *meta*-lenguaje que coincide con la sintaxis del comando *iptables* de *GNU Linux*. Como consecuencia, el *meta-lenguaje* permite utilizar la riqueza semántica que posee dicho comando para definir situaciones de capa de red. Además no requiere de capacitación adicional para los administradores de red que están familiarizados con el uso de esta herramienta. Para almacenar las *reglas de conectividad* se realiza una sobrecarga del atributo *description* que brinda el objeto *radiusObjectprofile (OID=2.5.4.13).* Este atributo *LDAP* soporta múltiples valores con una longitud máxima de 1024 caracteres. Esto permite almacenar una *regla* por cada *valor* de dicho atributo, facilitando de esta forma la lectura del conjunto de reglas.

La implementación del conjunto de *reglas de conectividad* depende del direccionamiento IP particular de cada implementación, ya que las reglas se aplican en la capa de red.

A partir del direccionamiento de red, que se planteó con fines didácticos en la sección Direccionamiento General de Red, se desarrollan algunos ejemplos de subdivisión del direccionamiento IP para los perfiles de conectividad.

Perfil de Conectividad	Red IP	Método de Asignación IP Permitido para el Perfil
00-administrador	*192.168.176.0/26*	*Sólo por usuario*
10-profesor	*192.168.176.0/23 o 192.168.180.0/22*	*Por usuario o dinámicamente*
20-alumno	*192.168.178.0/23 o 192.168.180.0/22*	*Por usuario o dnámicamente*
99-invitado	*192.168.180.0/22*	*Sólo dinámicamente*
00-dirs-dinámicas	*192.168.180.0/22*	*Perfil administrativo*

Tabla IV.11.: Ejemplo de asignación de direcciones IP a perfiles de conectividad.

Como se observa en la Tabla IV.11 , donde se han definido diferentes redes IP para utilizar en cada perfil de conectividad. Se ha definido además, un perfil denominado *00-dirs-dinámicas* con fines administrativos que se usará para aplicar “marcas” al tráfico de las direcciones asignadas de forma dinámica.

La aplicación de los *perfiles de conectividad* se ejecuta en dos partes: la primera parte corresponde a las *reglas de conectividad* y la segunda parte corresponde a los *atributos RADIUS*.

El Servidor de Autenticación también cumple el rol de *gateway* de la red inalámbrica. En efecto, actúa como *router / firewall* y aplica las *reglas de conectividad* al tráfico de red que lo atraviesa. Estas *reglas* se aplican a través del *entono de filtrado de paquetes*[1] que posee el *núcleo* de *Linux.* Para modificar el conjunto de reglas se usa el comando *iptables.*

El proceso que aplica las *reglas de conectividad* al tráfico de red tiene dos fases diferentes. La primera fase "marca" cada paquete, de acuerdo al *perfil de conectividad* que tienen asignado el usuario que origina dicho tráfico. Esta tarea se ejecuta aún antes de que el *gateway* tome decisiones sobre la suerte (si se reenvía o no, ruta que se asigna, etc) del paquete. Para decidir que "marca" le corresponde a cada paquete, se evalúa la dirección de origen del mismo. A fin de simplificar este proceso se vinculan ciertas redes IP a perfiles de conectividad, para asignar las mismas por usuario, como se puede observar en la Tabla IV.11. El *atributo Filter-Id* se usa como "marca" de tráfico, y de esta manera si un paquete pertenece a una red específica, se marca con el valor de dicho atributo.

Las direcciones IP asignadas de forma dinámica no se conocen a priori, e inclusive el servidor *PPPoE* y el servidor de *Portal Cautivo* administran *pools* de direcciones dinámicas diferentes.

Por esta razón, existe el perfil de administrativo *00-dirs-dinámicas*. Este perfil de conectividad, crea una *subcadena* específica dentro de la *cadena PREROUTING* en la *tabla MANGLE.* En esta *subcadena,* los servidores que administran los *pools* de direcciones dinámicas e insertan las reglas que relacionan una dirección IP origen con una "marca" de

1 http://www.netfilter.org

tráfico específica. Este comportamiento se logra a través de *scripts* que se instancian al iniciar una conexión y al concluir la misma.

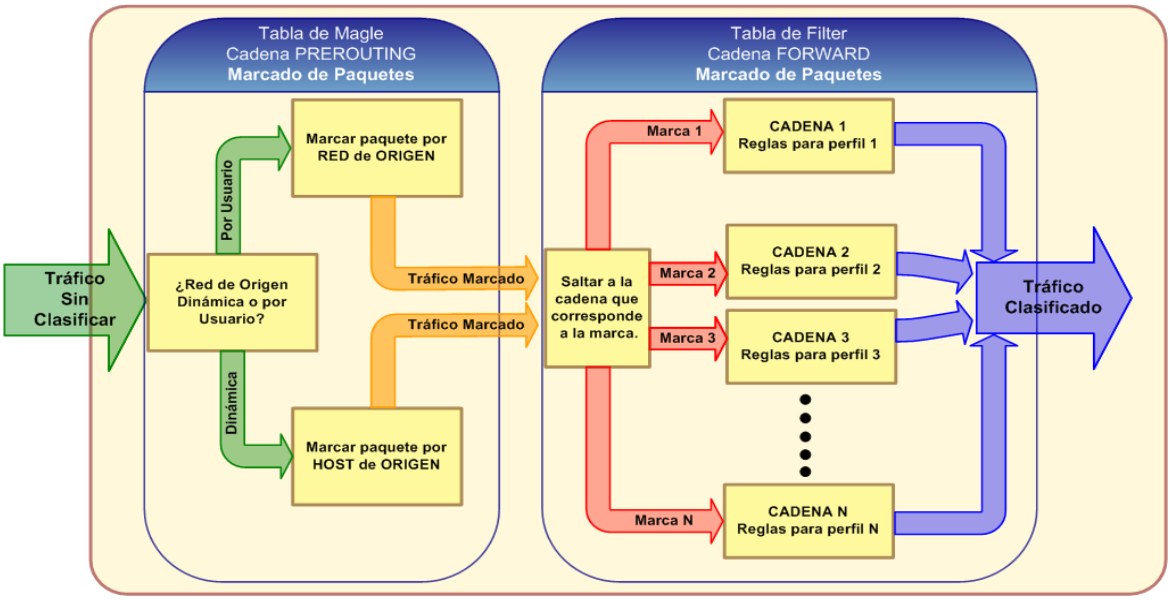

Ilustración IV.5.: Proceso de aplicación de Reglas de Conectividad.

Si bien este comportamiento se podría extender a todas las direcciones IP, incluidas las que se asignan por usuario, se implementa sólo para las direcciones IP dinámicas por razones de rendimiento. De esta manera se evita el crecimiento desmedido de la *tabla MANGLE* y las operaciones de búsqueda en la misma.

La segunda fase "aplica los permisos", es decir filtra el tráfico, de acuerdo a la marca que posee cada paquete. Estos procesos se pueden observar en la Ilustración IV.5. Cada conjunto de permisos que contiene el perfil de conectividad se vuelcan en una *subcadena* diferente dentro de la *cadena FORWARD* de la *tabla FILTER.* El proceso de *filtrado de paquetes* no se realiza en función de las direcciones IP de origen, sino de acuerdo a la "marca" que tiene el paquete, y de esta forma se deriva a la

cadena correspondiente. El orden en que se procesa el conjunto de reglas es importante y determina el comportamiento de cada perfil. Es por esto que los nombres de los perfiles contienen un número que permite determinar el orden de procesamiento de las reglas.

Para aplicar las *reglas de conectividad* almacenadas en los atributos *LDAP* de cada *objeto perfil de conectividad,* se desarrolló un *script* que lee el directorio LDAP y crea los conjuntos de reglas de marcado y filtrado de paquetes. Este *script* se denomina *wifi-policy* y se le pueden enviar los siguientes parámetros:

- *Start:* este parámetro indica a *wifi-policy* que debe leer las reglas del directorio LDAP y crear las cadenas correspondientes para "marcar" y "filtrar" el tráfico. Verifica si fue instanciado previamente y crea en */var/run/wifi* dos archivos. El primero, denominado *wifi.args,* cuya función reflejar el estado de *start* y que contiene una copia de las reglas creadas y orden de creación de las mismas. El segundo contiene el listado de reglas en orden inverso, y se llama *wifi-rev.args.*

- *Stop:* Este parámetro indica a *wifi-policy* que debe borrar las reglas. Éstas deben borrarse en orden inverso, para lo que se procesa el archivo */var/run/wifi-rev.args.* Luego se borran los archivos que reflejan el estado.

- *Status:* Este parámetro usa los archivos de estado para informar si las reglas se han cargado o no. En caso de que las reglas se hayan cargado, las muestra.

- Restart: Este parámetro permite actualizar las reglas configuradas en memoria cuando se han realizado modificaciones en el *directorio LDAP.*

El *script wifi-policy* se encuentra en */etc/init.d/* y se instancia desde la línea de comandos. Para que el mismo se ejecute al momento de inicio, en *Debian*, se debe utilizar el comando *update-rc. Wifi-policy* se puede observar en el Apéndice E.

Los *Atributos RADIUS* que componen un perfil de conectividad se envían, junto con los atributos del usuario, cuando el servidor *RADIUS* envía un mensaje *Access Accept.*

El conjunto de atributos es recibido e interpretado por el *NAS* en función de sus características. Es por eso que no todos los servidores interpretan el conjunto de atributos extendidos definidos por los diccionarios adicionales de los fabricantes.

La siguiente tabla Tabla IV.12, lista un conjunto tentativo de atributos que pueden usarse para modelar el servicio que recibe un perfil determinado. En dicha tabla se muestra también la compatibilidad con los servidores utilizados.

Atributo RADIUS	Servidor PPPoE	Servidor Chillispot	Significado
Session-Timeout	*X*	*X*	El valor en segundos de este parámetro determina el límite de tiempo máximo que puede alcanzar una sesión antes de que el *NAS* desconecte la misma.
Idle-Timeout	*X*	*X*	El valor en segundos de este parámetro determina el límite de tiempo máximo de inactividad que puede alcanzar una sesión antes de que el *NAS* desconecte la misma.

Atributo RADIUS	Servidor PPPoE	Servidor Chillispot	Significado
Simultaneous-Use	*X*	*X*	El valor numérico de este parámetro determina el número de sesiones simultáneas que se permite para una cuenta de usuario.
WISPr-Bandwidth-Max-Up		*X*	El valor en *bits por segundo* de este parámetro es el límite máximo de ancho de banda para *upload.*
WISPr-Bandwidth-Max-Down		*X*	El valor en *bits por segundo* de este parámetro es el límite máximo de ancho de banda para *download.*
WISPr-Session-Terminate-Time		*X*	El valor de este parámetro en formato ISO 8601, determina el momento en que se desconectará a un usuario. (ej: 2007-12-20T19:00:00+00:00)
Chillispot-Max-Input-Octects		*X*	El valor en *bytes* de este parámetro determina que una vez alcanzado ese límite de transmisión se desconecta al usuario.
Chillispot-Max-Output-Octects		*X*	El valor en *bytes* de este parámetro determina que una vez alcanzado ese límite de recepción se desconecta al usuario.
Chillispot-Max-Total-Octects		*X*	El valor en *bytes* de este parámetro determina que una vez alcanzado ese límite, suma de transmisión y de recepción, se desconecta al usuario.

Tabla IV.12.: Atributos RADIUS sugeridos para modelar perfiles de conectividad.

8. HERRAMIENTAS DE ADMINISTRACIÓN

Para realizar la administración básica del modelo se usan aplicaciones web desarrolladas en lenguaje *PHP* y de código abierto. Las dos aplicaciones que se usan son *phpLDAPadmin*[1] *y dialup Admin.*

1 http://phpldapadmin.sourceforge.net/

PhpLDAPadmin es un navegador *LDAP* con interfaz web y permite realizar tareas de administración sobre un directorio *LDAP.* Una de las razones por las que se elige esta herramienta es la capacidad de utilizar *templates* o *moldes* definidos por el usuario. Esto permite definir *moldes* que contienen los atributos necesarios para llevar a cabo una tarea administrativa. De esta forma, se simplifica la utilización y se evitan errores de carga. Por este motivo, se desarrollaron tres *moldes*, uno para crear los objetos que contienen los *perfiles de conectividad,* otro para crear los objetos que definen a los usuarios, y uno adicional que permite definir objetos que contienen usuarios de la red inalámbrica que a su vez son usuarios de un servidor de correo electrónico.

Ilustración IV.6.: Molde para crear perfiles de conectividad en phpLDAPadmin.

En caso necesitarlo, los administradores pueden generar *moldes* propios que se adapten a las necesidades particulares.

Se puede observar en las Ilustraciones IV.6 y IV.7 las capturas de pantalla de dos de los moldes creados. En el *molde* para la creación de usuarios se puede observar un control *combo* desplegado que permite elegir el *perfil de conectividad* deseado dentro de los existentes en el directorio *LDAP.*

Ilustración IV.7.: Molde para crear usuarios en phpLDAPadmin.

La herramienta *dialup admin* se distribuye con la aplicación *freeradius* y está diseñada para administrar servidores *RADIUS.* Principalmente se

puede usar para obtener reportes sobre el *accounting* o registro de uso y para administrar a los usuarios que se encuentran conectados.

Las principales limitaciones se encuentran en el hecho de que está diseñada para trabajar con servidores *PPP* principalmente y que no diferencia los atributos obtenidos desde el perfil de conectividad al que pertenece el usuario de los atributos propios de dicho usuario, lo que puede resultar en un error, al momento de modificar los atributos del usuario.

Sin embargo, se complementa con *phpLDAPadmin* si se utiliza sólo para administrar los registros de uso o *accounting* y obtener información de los usuarios conectados.

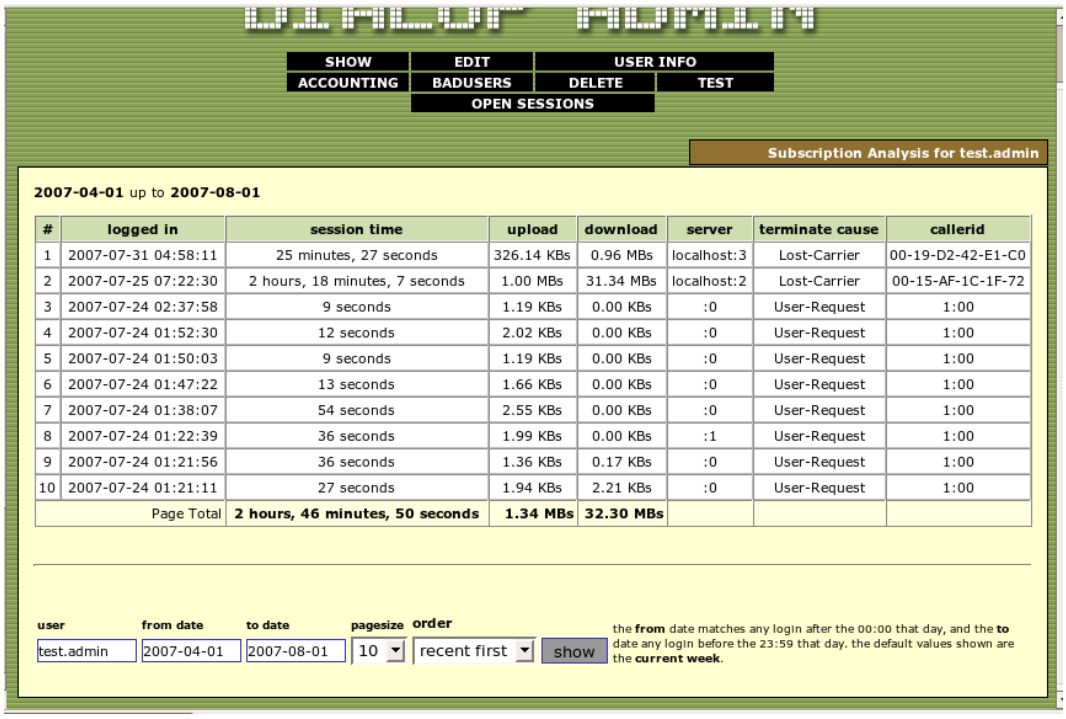

Ilustración IV.8.: Ejemplo de accounting de un usuario con dialup admin.

En la Ilustración IV.8 se puede observar el *accounting* de un usuario en un período de tiempo determinado a través de *dialup admin.* La diferencia en los datos entre los primeros registros y los subsiguientes, obedece a que dicho usuario se ha conectado a través de *métodos de acceso* diferentes.

V. RESULTADOS

Se implementó un prototipo del modelo de infraestructura con el objeto de verificar el comportamiento funcional y el rendimiento comparativo del mismo.

Para la implementación del *Servidor de Autenticación* se utilizaron herramientas de virtualización[1], a fin de minimizar los costos de implementación del prototipo. Para realizar las pruebas se usaron diferentes *AP,* incluyendo un *AP* con sistema operativo Linux. Los modelos de los *AP* usados son:

- Edimax EW-7206APg-Firmware V6.1.
- Linksys WRTSL54GS-Firmware Linux Xwrt.
- AP Linux *MultiBSS* - Interfaz inalámbrica Trendnet TEW-443PI.

Todos los *AP* utilizados en el prototipo cumplen con la norma *IEEE802.11bg*.

Una vez que se verificó el desempeño funcional del prototipo, se procede con las pruebas de rendimiento. Las pruebas de rendimiento se ejecutaron de la siguiente forma:

- Se montó un servidor *web* en la red interna con un archivo de imagen *ISO* de 81 MB.

1 En este caso particular se utilizó *VMWare Server*. http;//www.vmware.com

- Los clientes de la red inalámbrica usaron sistema operativo Linux Ubuntu 7.04, MS-Windows XP, MS-Windows Vista Ultimate, PalmOS 5.04. Para realizar la descarga del archivo *imagen.iso*, en todas las plataformas, excepto PalmOS, se usó la aplicación *wget.*

- Los valores que se utilizan en la comparación son valores promedio de, al menos, tres descargas.

- Al virtualizar el *servidor de Autenticación*, la velocidad máxima de las interfaces de red virtuales queda limitada a 10 Mbits, por lo que las pruebas se realizaron en función de este valor.

- Se tomaron valores de referencia usando las redes cableadas, tanto para la red interna, como para el *Sistema de Distribución.*

Método	Prueba	AP	Backbone [Mbps]	Norma Inalámbrica	Tasa de transferencia [Kb/s]	Tiempo [s]	% de Transferencia de Referencia 1	% de Transferencia de Referencia 2
Pruebas para 802.11b – Tamaño de descarga 85.299.200 bytes								
Referencia 1	Red Interna Cableada Referencia	---	10	---	1.030,00	79	100,00%	133,02%
Referencia 2	Desde Sistema de Distribución	---	10	---	774,29	107,25	75,17%	100,00%
Chillispot	Modelo-AP-Chillispot	AP1	10	802.11b	457,11	188	44,38%	59,04%
Chillispot	Modelo-APLINUX-Chillispot	APLINUX	10	802.11b	538,42	155	52,27%	69,54%
WPA	Modelo-AP-WPA	AP1	10	802.11b	412,71	205,33	40,07%	53,30%
WPA	Modelo-APLINUX-WPA	APLINUX	10	802.11b	540,49	154,2	52,48%	69,80%
PPPoE	Modelo-AP-PPPoE-MTU1460	AP2	10	802.11b	430,96	193	41,84%	55,66%
PPPoE	Modelo-APLINUX-PPPoE-MTU1460	APLINUX	10	802.11b	492,91	168,25	47,85%	63,66%
Pruebas para 802.11g – Tamaño de descarga 85.299.200 bytes								
Chillispot	Modelo-AP-Chillispot	AP1	10	802.11g	655,82	127,67	63,67%	84,70%
Chillispot	Modelo-APLINUX-Chillispot	APLINUX	10	802.11g	677,70	123	65,80%	87,53%
WPA	Modelo-AP-WPA	AP1	10	802.11g	608,63	137,25	59,09%	78,60%
WPA	Modelo-APLINUX-WPA	APLINUX	10	802.11g	699,47	119,5	67,91%	90,34%
PPPoE	Modelo-AP-PPPoE-MTU1460	AP2	10	802.11g	590,59	141,33	57,34%	76,27%
PPPoE	Modelo-APLINUX-PPPoE-MTU1460	APLINUX	10	802.11g	623,42	136,17	60,53%	80,51%

REFERENCIAS:		
	APLINUX	Trendnet TEW443PI
	AP1	Edimax EW-7206APg
	AP2	Linksys WRTSL54GS

Ilustración V.1.: Síntesis de Resultados de Rendimiento.

En la Ilustración V.1 se puede observar una tabla que ilustra la síntesis de los resultados obtenidos. Dicha tabla está dividida en tres partes, los valores de referencia obtenidos, los valores obtenidos para las pruebas realizadas con la norma *IEEE802.11b* y con la norma *IEEE802.11g.* Las referencias a *WPA* en este contexto, se refieren a la implementación del modelo propuesto, por lo que deben interpretarse como *WPA-Chillispot.*

Se observa en los valores de referencia, tomados a través de una conexión *10Base-T-Full Duplex*, que la infraestructura propuesta penaliza la velocidad obtenida en 55 Kb/s .

Se observa también, que utilizando un acceso inalámbrico, la menor velocidad obtenida es el 53,30% de la velocidad de referencia 2, velocidad máxima del *Sistema de Distribución*.

Esto se debe principalmente a que el medio inalámbrico es intrínsecamente un medio compartido o *half-duplex.*

El aumento de la velocidad que se aprecia en las pruebas realizadas con la norma *802.11g,* con respecto a las pruebas realizadas con la norma *802.11b,* se deben a la incidencia de la velocidad interacción del segmento que utiliza el medio compartido inalámbrico. Si bien la velocidad de este medio se aumenta un 490%, se puede observar que la mejora de velocidad en promedio es del 21,15% (ver Tabla V.1).

Método de Acceso	802.11b	802.11g	Diferencia
Chillispot - AP	59,04%	84,70%	**25,66%**
Chillispot - APLINUX	69,54 %	87,53%	**17,99%**
WPA – AP	53,30%	78,60%	**25,3%**
WPA – APLINUX	69,80%	90,34%	**20,54%**
PPPoE – AP	55,66%	76,23%	**20,57%**

Método de Acceso	802.11b	802.11g	Diferencia
PPPoE – APLINUX	63,66%	80,51%	**16,85%**
Promedio	**61,83%**	**82,99%**	**21,15%**

Tabla V.1.: Porcentajes comparativos de velocidad entre normas inalámbricas con respecto a la velocidad máxima del Sistema de Distribución.

Se aprecia también, en la Tabla V.2, una diferencia de velocidad entre las pruebas realizadas con un *AP convencional* frente a un *AP implementado con sistema operativo GNU/LINUX.* Si bien esto se debe a la potencia del hardware de una computadora frente al hardware de un AP, muestra la eficiencia del sistema operativo respecto a las operaciones en red, ya que la ganancia de velocidad es en promedio el 8,97%. Las principales desventajas de un APLINUX frente a un *AP convencional,* son mayor costo del hardware y mayor costo de administración.

Método de Acceso	AP Convencional	AP Linux	Diferencia
Chillispot – 802.11b	59,04%	69,54%	**10,50%**
Chillispot – 802.11g	84,70%	87,53%	**2,83%**
WPA – 802.11b	53,30%	69,80%	**16,50%**
WPA – 802.11g	78,60%	90,34%	**11,74%**
PPPoE – 802.11b	55,66%	63,66%	**8,00%**
PPPoE – 802.11g	76,27%	80,51%	**4,24%**
Promedio	**67,93%**	**76,90%**	**8,97%**

Tabla V.2.: Porcentajes comparativos de velocidad entre AP con respecto a la velocidad máxima del Sistema de Distribución.

Al comparar los *métodos de acceso* se observa la mayor velocidad del método que utiliza a *Chillispot,* seguido por el método *WPA*, frente al método *PPPoE*.

802.11b			
	Chillispot	**WPA**	**PPPoE**
Promedio entre AP	64,29%	61,55%	59,66%
802.11g			
Promedio entre AP	86,12%	84,47%	78,39%

Tabla V.3.: Porcentajes comparativos de velocidad entre métodos de acceso con respecto a la velocidad máxima del Sistema de Distribución.

Los resultados expuestos en la Tabla V.3, muestran que el método *PPPoE* es el más lento de los tres propuestos, esto principalmente se debe a la limitación de *MTU* que surge al encapsular un protocolo de capa de enlace dentro de otro protocolo de la misma capa.

VI. CONCLUSIONES

Utilizando herramientas de software libre fue posible construir un prototipo del modelo de infraestructura propuesto. Dicho prototipo, brinda a los administradores de red, un conjunto de herramientas que permiten representar, aplicar y controlar diferentes **políticas de uso** de la red inalámbrica. A su vez ofrece a los usuarios múltiples formas de conectarse, a través las mismas credenciales.

El diseño del modelo de infraestructura ha mostrado su flexibilidad para poder implementarse también en ámbitos privados o públicos como casos particulares y los resultados obtenidos en las mediciones demuestran un balance entre el rendimiento y la implementación de políticas de seguridad, probando la factibilidad de uso en los escenarios típicos de una red inalámbrica.

El uso de protocolos libres permite que la infraestructura que se utiliza para la implementación, pueda ser usada simultáneamente por otros servicios que requieren de autenticación, disminuyendo el costo total de propiedad de la infraestructura de red al no duplicar servicios de autenticación.

VII. FUTURAS DIRECCIONES

- Desarrollar un meta-lenguaje más completo y específico para los perfiles de conectividad y modificar las herramientas para aplicación de políticas para implementar el uso de *policy routing*.

- Modificar el servidor *pppd* para que ejecute el proceso de autenticación con protocolo *EAP* utilizando un servidor *RADIUS.*

- Modificar el servidor *Chillispot* para usar *802.11i* de forma nativa.

- Ampliar el soporte de las herramientas de administración.

VIII. APÉNDICES E ÍNDICES

1. APÉNDICE A: COMPILACIÓN DE FREERADIUS PARA DEBIAN

Para compilar del paquete *freeradius-1.1.3-3* para la distribución Debian Sarge 3.1, es necesario contar con el entorno de compilación y descargar el archivo fuente con los parches para Debian.

```
# apt-get build-dep freeradius
# apt-get install fakeroot libssl-dev libpq-dev
# apt-get source freeradius
```

Texto VIII.1.*: Apéndice de compilación de freeradius.*

De esta forma se instalan las dependencias necesarias para poder compilar el paquete *freeradius* para Debian, y se descargan los archivos fuente de la aplicación. Los mismos deberán ubicarse en el siguiente directorio */usr/local/src/deb-src/freeradius/.*

Una vez posicionados en el directorio mencionado, se edita el archivo *debian/rules* y se comentan las líneas 26 y 27:

```
#buildssl=--without-rlm_eap_peap --without-rlm_eap_tls –without-rlm_eap_ttls --without-rlm_otp --without-rlm_sql_postgresql --without-snmp
#modulelist=krb5 ldap sql_mysql sql_iodbc
```

Texto VIII.3.*: Apéndice de compilación de freeradius.*

Se debe eliminar la marca de comentario en líneas 28 y 29:

```
buildssl=--with-rlm_sql_postgresql_lib_dir=`pg_config --libdir` --with-rlm_sql_postgresql_include_dir=`pg_config --includedir`
modulelist=krb5 ldap sql_mysql sql_iodbc sql_postgresql
```

Texto VIII.2.*: Apéndice de compilación de freeradius.*

En los parámetros con que llama al archivo *configure* a partir la línea 59, se agregan las siguientes opciones :

```
./configure \
    $(confflags) \
    --prefix=/usr \
    --exec-prefix=/usr \
    --mandir=$(mandir) \
    --sysconfdir=/etc \
    --libdir=$(libdir) \
    --datadir=/usr/share \
    --localstatedir=/var \
    --with-raddbdir=$(raddbdir) \
    --with-logdir=/var/log/$(package) \
   --enable-ltdl-install=no --enable-strict-dependencies \
   --with-large-files --with-udpfromto --with-edir \
   --enable-developer \
   --config-cache \
   ${buildssl} \
   --with-system-libtool \
   --with-openssl-libraries=/usr/lib \
   --with-openssl-includes=/usr/include/openssl
```

Texto VIII.4.*: Apéndice de compilación de freeradius.*

Luego se edita el archivo *debian/control* y en la sección *Build-Depends:* se agrega *libssl-dev* y *libpg-dev* y se borra en la *sección Build-Conflicts: on libssl-dev.*

```
Build-Depends: debhelper (>= 5), libltdl3-dev, libpam0g-dev,
libmysqlclient15-dev | libmysqlclient-dev, libgdbm-dev,
libldap2-dev, libsasl2-dev, libiodbc2-dev, libkrb5-dev, snmp,
autotools-dev, dpatch (>= 2), libperl-dev, libtool, dpkg-dev (>=
1.13.19), libssl-dev, libpq-dev
```

Texto VIII.5.*: Apéndice de compilación de freeradius.*

Una vez que se han completado estas tareas se procede a la creación del paquete Debian:

```
# dpkg-buildpackage -r fakeroot -uc -us
........
#ls -F -1
freeradius-1.1.3/
freeradius_1.1.3-3.diff.gz
freeradius_1.1.3-3.dsc
freeradius_1.1.3-3_i386.changes
freeradius_1.1.3-3_i386.deb
freeradius_1.1.3.orig.tar.gz
freeradius-dialupadmin_1.1.3-3_all.deb
freeradius-iodbc_1.1.3-3_i386.deb
freeradius-krb5_1.1.3-3_i386.deb
freeradius-ldap_1.1.3-3_i386.deb
freeradius-mysql_1.1.3-3_i386.deb
freeradius-postgresql_1.1.3-3_i386.deb
```

Texto VIII.6*.: Apéndice de compilación de freeradius.*

Una vez creados, se puede realizar la instalación de dichos paquetes y de los módulos que se usan con el comando *dpkg -i nombre_paquete.*

Los paquetes creados contienen los módulos *rlm_eap_XXX* necesarios para usar los protocolos *EAP_TLS, EAP_TTLS y EAP_PEAP*.

Compilación de freeradius 1.1.6-4 con Debian Etch 4.0

Se incluye esta actualización para compilar *freeradius_1.1.6-4* en *Debian Etch 4.0*. Actualmente esta es la última versión de *freeradius*, y para compilarla al estilo Debian con soporte para EAP-TLS, EAP-TTLS y EAP-PEAP, se deben seguir los pasos que se detallan a continuación.

```
#cd /usr/local/src/

#wget
http://http.us.debian.org/debian/pool/main/f/freeradius/freerad
ius_1.1.6.orig.tar.gz

#wget
http://http.us.debian.org/debian/pool/main/f/freeradius/freerad
ius_1.1.6-4.dsc

#wget
http://http.us.debian.org/debian/pool/main/f/freeradius/freerad
ius_1.1.6-4.diff.gz

dpkg-source -x freeradius_1.1.6-4.dsc

apt-get build-dep freeradius

apt-get install gcc bizp2 fakeroot libssl-dev
```

Texto VIII.7.*: Compilación freeradius-1.1.6 en Debian 4.0*

Los pasos anteriores habrán desempaquetado los fuentes, aplicado los parches específicos de Debian y descargado las dependencias necesarias para compilar esta aplicación.

Se edita el archivo *freeradius-1.1.6/debian/rules y* alrededor de la línea 51, donde se listan los parámetros para ejecutar el archivo *configure*, se deberán borrar las siguientes líneas :

```
./configure \
      $(confflags)\
      ....#Borrar las siguientes:
      --without-rlm_eap_tls \
      --without-rlm_eap_ttls \
      --without-rlm_eap_peap \
      --without-openssl \
      ......
```

Texto VIII.8.*: Compilación freeradius-1.1.6 en Debian 4.0*

Y en el mismo lugar para reemplazar las líneas borradas se deberán colocar las opciones para OpenSSL, como figura a continuación:

```
./configure \
  $(confflags) \
  --prefix=/usr \
  --exec-prefix=/usr \
  --mandir=$(mandir) \
  --sysconfdir=/etc \
  --libdir=$(libdir) \
  --datadir=/usr/share \
  --localstatedir=/var \
  --with-raddbdir=$(raddbdir) \
  --with-logdir=/var/log/$(package) \
  --enable-ltdl-install=no --enable-strict-dependencies \
  --with-large-files --with-udpfromto --with-edir \
  --enable-developer \
  --config-cache \
  --with-system-libtool \
  --with-rlm_otp \
 --with-rlm_sql_postgresql_lib_dir=`pg_config --libdir` \
 --with-rlm_sql_postgresql_include_dir=`pg_config --includedir`
\
     #Agregar las siguientes opciones:
   --with-openssl \
   --with-openssl-libraries=/usr/lib \
   --with-openssl-includes=/usr/include/openssl
```

Texto VIII.9.*: Compilación freeradius-1.1.6 en Debian 4.0*

Las opciones que se han agregado configuran las bibliotecas de *OpenSSL* necesarias para *TLS.* Las opciones que se borran son las que explícitamente en *Debian* desactivan en la compilación las opciones que requieren enlazar las bibliotecas de *OpenSSL.*

Luego de editar estas opciones, se deben comentar algunas líneas alrededor de la línea 187, que verifican si se ha realizado la compilación enlazando las bibliotecas de OpenSSL, y si es así, lo informan y abortan el proceso.

```
.....
dh_shlibdeps -a
#for pkg in $(shell grep ^Package debian/control | awk '{print $$2}') ; do \
#  if dh_shlibdeps -p $$pkg -- -O | grep -q libssl; then \
#    echo "$$pkg links to openssl" ;\
#    exit 1 ;\
#  fi ;\
#done
dh_installdeb -a
....
```

Texto VIII.10.*: Compilación freeradius-1.1.6 en Debian 4.0*

Una vez salvado el archivo */debian/rules*, se debe editar el archivo *debian/control* y en la sección *Buil-Depends,* se debe *agregar, libssl-dev:*

```
Source: freeradius
Build-Depends: debhelper (>= 5), libltdl3-dev, libpam0g-dev, libmysqlclient15-dev | libmysqlclient-dev, libgdbm-dev, libldap2-dev, libsasl2-dev, libiodbc2-dev, libkrb5-dev, snmp, autotools-dev, dpatch (>= 2), libperl-dev, libtool, dpkg-dev (>= 1.13.19), libpq-dev, libsnmp-dev, libssl-dev
```

Texto VIII.11.*: Apéndice de compilación de freeradius.*

Realizados estos cambios, se procede a la creación del paquete como en la versión *freeradius-1.1.3.*

2. APÉNDICE B: CREACIÓN DE CA Y CERTIFICADOS PARA EAP-TLS

Para poder utilizar *EAP-TLS,* a fin de implementar *EAP-TTLS o EAP-PEAP,* es necesario crear una *Autoridad de Certificación (CA)* y los certificados para el servidor, adicionalmente para utilizar *EAP-TLS* solamente se deben crear certificados para todos los clientes. Además la creación de certificados para el servidor será necesaria para implementar un servidor web con soporte *SSL.*

Es necesario que esté instalada en el sistema la aplicación *OpenSSL* que brinda las herramientas para realizar estas tareas.

Se empieza realizando las siguientes tareas:

```
#cd /etc/ssl/
#cp /usr/share/doc/freeradius/examples/xpextensions .
```

Texto VIII.12.*: Apéndice de OpenSSL*

Este archivo contiene las definiciones necesarias para poder crear lso certificados en formato *PKCS12* necesario para MS Windows XP y MS Windows 2003.

Luego hay que editar el archivo */etc/ssl/openssl.cnf* que contiene los parámetros de configuración por defecto para *OpenSSL.* Es conveniente modificar el directorio por defecto donde se almacenan los certificados y los datos por defecto necesarios para la creación de los mismos. Alrededor de la línea 34, se encuentra una referencia al directorio *./demoCA* que se reemplaza por *./masterCA,* y alrededor de la línea 123 se encuentra la sección con los datos por defecto para los certificados,

donde en las opciones por defecto se deben ingresar los datos de la organización.

```
........
##################################################################
[ CA_default ]
dir   = ./masterCA  # Where everything is kept
certs = $dir/certs  # Where the issued certs are kept
.......
[ req_distinguished_name ]
countryName         = Country Name (2 letter code)
countryName_default = AR
countryName_min     = 2
countryName_max     = 2
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Mendoza
localityName        = Locality Name (eg, city)
0.organizationName  = Organization Name (eg, company)
0.organizationName_default = Universidad de Mendoza

# we can do this but it is not needed normally :-)
#1.organizationName = Second Organization Name (eg, company)
..........
```

Texto VIII.13.*: Apéndice de OpenSSL*

Una vez salvado este archivo, se edita */usr/lib/ssl/misc/CA.sh* para, alrededor de la línea 42, reemplazar *./demoCA* por *./masterCA*, a fin de establecer el directorio donde el *script* crea los archivos de claves y certificados.

```
....
CATOP=./masterCA
....
```

Texto VIII.14.*: Apéndice de OpenSSL*

Una vez realizadas estas tareas de configuración, se procede a crear la *CA* y a crear la solicitud *de firma del certificado (CSR)* para el servidor:

```
/etc/ssl#/usr/lib/ssl/misc/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:
.............

/etc/ssl# openssl req -new -nodes -keyout s erver_key.pem -out
serverreq.pem
```

Texto VIII.15.*: Apéndice de OpenSSL*

Y se procede a firmar y emitir el certificado para el servidor con el siguiente comando:

```
/etc/ssl#openssl ca -out server_cert.pem -days 1825 -extensions
xpserver_ext -extfile xpextensions -infiles ./serverreq.pem
```

Texto VIII.16.*: Apéndice de OpenSSL*

En este punto se ha creado el certificado *server_cert.pem*, que se copia al servidor *RADIUS*.

Si bien en este documento sólo se tratarán los requerimientos de configuración de *EAP-TLS* mínimos, para *usar EAP-PEAP y EAP-TTLS.* Si requiere el uso de certificados para los clientes*,* se pueden generar los certificados para los clientes y exportarlos a formato *PKCS12* a través del siguiente *script:*

```
#!/bin/bash
cd /etc/ssl
OPENSSL=/usr/bin/openssl
DAYS=365
# Solicitud de firma de certificado
$OPENSSL req -new -nodes -keyout $1_key.pem -out $1req.pem
# Firma y emisión del certificado
$OPENSSL ca -out $1_cert.pem -days $DAYS -extensions xpserver_ext -extfile xpextensions -infiles ./$1req.pem
# Exportación a formato PKCS12
$OPENSSL pkcs12 -nodes -export -in $1_cert.pem -inkey $1_key.pem -out $1_cert.p12 -clcerts
```

Texto VIII.17.*: Apéndice de OpenSSL*

La configuración en las computadoras cliente dependerá del sistema operativo de las mismas, pero en todas se debe importar el certificado de la *CA, cacert.pem,* y la clave del cliente *client_key.pem*.

Para completar las necesidades de configuración del servidor *RADIUS*, se crean dos archivos *adicionales, dh y random:*

```
/etc/ssl# openssl gendh >> dh
/etc/ssl# dd if=/dev/urandom of=random count=2
```

Texto VIII.18.*: Apéndice de OpenSSL*

Como resultado de estas tareas se obtienen los archivos necesarios para configurar el servidor *RADIUS* con soporte *EAP-TTLS y EAP-PEAP.*

3. APÉNDICE C: COMPILACIÓN DE CHILLISPOT

La distribución estable para *Debian* de *Chillispot* es la versión 1.0.0, existe una versión de desarrollo, *chillispot-1.1.0,* que agrega las siguientes opciones de configuración:

- *radiusnasip:* esta opción permite especificar el valor para el atributo *NAS-IP-Address*.

- *radiuscalled:* esta opción permite especificar el valor del atributo *Called-Station-ID*, colocando un nombre en vez de la dirección *MAC* de la interfaz inalámbrica.

- *confusername y confpassword:* Estas opciones permiten que a intervalos regulares se chequee el servidor *RADIUS*, y si en los atributos devueltos por el servidor existen los valores para *uamallowed, macallowed o interval*, reemplazan los valores configurados en el archivo de configuración *de chillispot.*

- *conup y condown:* Estas opciones permiten especificar dos scripts que se ejecutarán cuando un cliente se autentique o se desconecte. A los *scripts* especificados se le envían como parámetros los valores de varios atributos *RADIUS.*

Como se necesita utilizar la última de las opciones nombradas, se realiza la compilación al modo *Debian* de la versión 1.1.0.

Durante el proceso de prueba en la implementación del prototipo, se encontró que el *demonio chilli,* al usar el método de autenticación *WPA*,

causa una falla de segmento. Este error no se pudo solucionar modificando la configuración.

Como conclusión, a fin de poder utilizar *chillispot* con *WPA* y con las directivas de configuración *conup/condown* de la versión 1.1.0, hay que compilar una versión intermedia.

Se requiere descargar la siguiente versión *chillispot-20060214.tar.gz.*

```
cd /usr/local/src/
wget http://www.chillispot.org/cvs/chillispot_20060214.tar.gz
tar -xvzf chillispot_20060214.tar.gz
cd chillispot
```

Texto VIII.19.*: Apéndice de compilación de chillispot.*

Se debe editar el archivo *debian/rules* alrededor de la línea 65, ya que existe un error de sintaxis, dice *nstall* en vez de *install.*

```
install: build
        dh_testdir
        dh_testroot
        dh_clean -k
        dh_installdirs


            # Add here commands to install the package into debian/chillispot.
        $(MAKE) install prefix=$(CURDIR)/debian/chillispot/usr
# Debajo de esta línea agregar la letra i(en azul) a nstall.
             install -m 644 -o root -g root doc/chilli.conf $(CURDIR)/debian/chillispot/etc/


# Build architecture-independent files here
```

Texto VIII.20.*: Apéndice de compilación de chillispot.*

Para realizar la compilación del paquete y el empaquetado en formato *debian* se ejecuta el siguiente comando:

```
dpkg-buildpackage -rfakeroot -us -uc
....
cd ..
dpkg -i chillispot_1.1.0_i386.deb
```

Texto VIII.21.*: Apéndice de compilación de chillispot.*

Como resultado se obtiene el archivo instalable en debian *chillispot_1.1.0_i386.deb.* Esta versión de *chillispot* permite usar el método de autenticación *WPA* y las directivas de configuración *conup/condown*, y no causa la falla de segmento de la versión 1.1.0. En caso de que se necesite compilar dicha versión, que se puede descargar desde *http://www.chillispot.org/download/chillispot-1.1.0.tar.gz,* el procedimiento a seguir es el mismo.

Una vez que se ha instalado *chillispot* se deberá verificar la existencia de un dispositivo tun/tap, que utiliza chillispot.

```
# ls -l /dev/net/
total 0
crw-rw-rw- 1 root root 10, 200 2006-09-14 22:35 tun
```

Texto VIII.22.*: Apéndice de compilación de chillispot.*

Si el mismo no existiese, se deberá crear de la siguiente forma:

```
# mkdir /dev/net/
# mknod /dev/net/tun c 10 200
```

Texto VIII.23.*: Apéndice de compilación de chillispot.*

4. APÉNDICE D: COMPILACIÓN DE RP-PPPOE EN MODO KERNEL

Para poder compilar el servidor *PPPoE* con soporte para la opción *-k,* que permite operar en espacio de *kernel*, se requiere que el *kernel* de Linux se haya compilado con soporte para PPP*oE.* La mayoría de los *kernel* nuevos (>2.6.8) están compilados con esta opción. Los paquetes *Debian* estándar no están compilados con esta opción, por lo que sólo se pueden ejecutar en espacio de usuario.

Es necesario descargar las cabeceras de *pppd* y los fuentes de *rp-pppoe:*

```
# apt-get install ppp-dev
# cd /usr/local/src
# apt-get source pppoe
# cd rp-pppoe-3.8/
```

Texto VIII.24.*: Apéndice de compilación de rp-pppoe modo kernel.*

En este punto se han descargado los fuentes de *rp-pppoe-3.8* y los parches necesarios para compilarlo estilo *Debian.* Es necesario modificar el archivo *debian/rules* para indicar la opción para modo *kernel.* Alrededor de la línea 10 se deberá agregar la siguiente opción*:*

```
build-stamp:
        dh_testdir
#       test -f configure || (aclocal; autoconf)
         test -f src/Makefile || (cd src && PPPD=/usr/sbin/pppd
./configure -enable-plugin=/usr/include/pppd)
        $(MAKE) -C src
```

Texto VIII.25.*: Apéndice de compilación de rp-pppoe modo kernel.*

El texto resaltado indica la ubicación de las cabeceras *pppd* descargadas anteriormente.

Una vez que el archivo se ha salvado, se realiza la compilación del mismo y la instalación:

```
# dpkg-buildpackage -rfakeroot -uc -us
.....
# cd ..
# dpkg -i pppoe_3.8-1.1_i386.deb
```

Texto VIII.26.*: Apéndice de compilación de rp-pppoe modo kernel.*

Es necesario un paso adicional para habilitar el plugin *rp-pppoe.so* de *pppd.*

```
# cd /etc/ppp/
# mkdir plugins
# cp /usr/lib/pppd/2.4.4/rp-pppoe.so /etc/ppp/plugins/
```

Texto VIII.27.*: Apéndice de compilación de rp-pppoe modo kernel.*

Como resultado de este procedimiento la aplicación *pppoe-server* soporta la opción *-k,* para trabajar en espacio de *kernel.*

5. APÉNDICE E: *SCRIPTS* PARA APLICAR REGLAS DE CONECTIVIDAD.

A continuación se detallan los *scripts* que se utilizan para aplicar las *reglas* de cada *perfil de conectividad.*

El *script wifi-policy* se observa a continuación y su función es buscar en *el directorio LDAP* las *reglas de conectividad* asociadas a cada perfil y configurar las tablas de *netfilter* a través del comando *iptables.*

```
#!/bin/bash
# wifi-policy script
#
# description: Start/stop/status
#              of profile rules of Wifi-Policy
#
#    Author: Pablo F. Gomez - pablo.gomez@um.edu.ar
#
#  Version 1.3 - 01-05-07 - Changes in start function,
#  in generation of wifi-rev.args
#-------------------------------------------------------
#Version 1.2 - 30-04-07 - Some changes in population
# of $LISTA, because ldapsearch cuts lines 80 character
# or more
#-------------------------------------------------------
# Version 1.1 - 30-04-07 - added ldapsearch parameter -S
#-------------------------------------------------------
#
export PATH=/sbin:/usr/sbin:/bin:/usr/bin
IPTABLES=/sbin/iptables
CONFDIR=/var/run/wifi
TEMP=""
REV=""
SEP=$'\x0A'$'\x20'

[ `id -u` -eq 0 ] || SUDO=sudo

do_start() {
     #  Search in LDAP directory for firewall
     #  rules of profiles

     #  Busco las entradas en ldap que contienen
     #  las reglas de cada profile
```

```
    LISTATEMP=`ldapsearch -x -LLL -S cn -b
ou=Profiles,dc=um,dc=edu description cn`

    LISTA=`echo "${LISTATEMP//$SEP/}" | awk -F :
'/^description:/ { print $2 } '`

    if [ -z ${#LISTA} ]
       then echo "Can't contact LDAP Server or Profiles
rules are empty"
       return 1
    fi

  #   Change Bash Internal Field Separator for newline,
  #   and store the original value in $OLD_IFS
  #   Cambio el Internal Field Separator de Bash,
  #   guardando en $OLD_IFS el valor original

  OLD_IFS=$IFS
  IFS=$'\x0A'

   # wifi.args store the firewall rules for log purpose
   # wifi-rev.args store inverse arguments for stop
   # function use
   #
   #
   # wifi.args guarda los argumentos para propositos de log
   # wifi-rev.args guarda los argumentos para revertir
   # en el stop

   if [ ! -e ${CONFDIR} ]
           then mkdir ${CONFDIR}
   fi

   if [ ! -e ${CONFDIR}/wifi.args ]
        then touch ${CONFDIR}/wifi.args
   else echo "Must stop before to start, try restart"
        return 1
   fi

   if [ ! -e ${CONFDIR}/wifi-rev.args ]
        then touch ${CONFDIR}/wifi-rev.args
   else echo "Must stop before to start, try restart"
        return 1
   fi

   echo "Starting Profiles rules....."

   for f in `printf "$LISTA"`; do
      echo "$f" >> ${CONFDIR}/wifi.args
      IFS=$OLD_IFS
```

```
      $SUDO $IPTABLES $f
      IFS=$'\x0A'
   done

tac ${CONFDIR}/wifi.args | sed -e 's/-N/-X/' -e 's/-A/-D/' >
${CONFDIR}/wifi-rev.args

   IFS=$OLD_IFS
   return 0
}
#-------------------------------------------------------------
-
do_stop() {

   if [ ! -e ${CONFDIR}/wifi-rev.args ]
      then echo "Must start before to stop"
           return 1
   fi
   echo "Stopping Profiles Rules..."

   # Change Bash Internal Field Separator for newline,
   # and store the original value in $OLD_IFS
   # Cambio el Internal Field Separator de Bash, guardando
   # en $OLD_IFS el valor original

   OLD_IFS=$IFS
   IFS=$'\x0A'
   LISTA=`cat ${CONFDIR}/wifi-rev.args`

   for f in `printf "$LISTA"`; do
      IFS=$OLD_IFS
      $SUDO $IPTABLES $f
      IFS=$'\x0A'
   done

   rm ${CONFDIR}/wifi.args
   rm ${CONFDIR}/wifi-rev.args
   IFS=$OLD_IFS

   return 0
}
#-------------------------------------------------------------
-

do_status() {
   if [ ! -e ${CONFDIR}/wifi.args ]
      then echo "Profile Rules Stopped"
           return 0
   else echo "Profiles Rules Started....."
        echo "Rules are: "
```

```
        cat ${CONFDIR}/wifi.args
   fi
   return 0
}

### Main
------------------------------------------------------------
-
case "$1" in

        stop)
                do_stop
                ;;
        start)
                do_start
                ;;
        restart)
                do_stop
                do_start
                ;;
        status)
                do_status
                ;;
        *)
                echo "Usage: /etc/init.d/wifi {start|stop|
restart|status}"
                ;;
esac
```

Los servidores que administran los *pools* de direcciones IP dinámicas, cada vez que un cliente se conecta, ejecutan un script que inserta una regla que vincula la direción IP entregada con el perfil de conectividad de dicho usuario. De igual forma, cuando el cliente se desconecta se borra la regla correspondiente, a través del mismo mecanismo.

Estas reglas se insertan en la subcadena definida por el perfil *00-dirs-dinámicas.* Dicha *subcadena* está en la cadena *PREROUTING* de la tabla *MANGLE.* Las reglas “marcan” los paquetes, de acuerdo al *perfil de conectividad* al que pertenece un usuario dado, pero no “filtran” el tráfico. Las reglas que se encargan de “filtrar” el tráfico se encuentran en la

definición de cada *perfil de conectividad* y se insertan con el *script wifi-policy.*

Los *scripts* que se observan a continuación insertan las reglas correspondientes a las direcciones IP dinámicas entregadas por el *Portal Cautivo*. Estos se ejecutan al iniciar y finalizar una conexión y se denominan *chilli.conup.sh* y *chilli.condown.sh* respectivamente. Ambos se ubican en el directorio */etc/chilli/.*

Script chilli.conup.sh:

```
#!/bin/bash
############################################################
#                    chilli.conup.sh
#
# Script para configurar el parámetro CONUP de Chillispot.
# Especificamente marca cada paquete con el valor
# del atributo Radius FILTER_ID que determina el
# perfil del usuario
# En sólo para los DHCP leases que Chillispot asigna
############################################################
#
# En este caso: Network: 192.168.1.176.0/21
# Static Network:  192.168.176.0/22
# Dynamic Network: 192.168.180.0/22
#  Chillispot Dynamic Network: 192.168.182.0/23
############################################################
# Version 1.1  28-04-2007
############################################################
IPTABLES="/sbin/iptables"


case "$FRAMED_IP_ADDRESS" in
    192.168.182*) $IPTABLES -A marca-dinamicas -t mangle -s
$FRAMED_IP_ADDRESS/32 -j MARK --set-mark $FILTER_ID
;;
    192.168.183*) $IPTABLES -A marca-dinamicas -t mangle -s
$FRAMED_IP_ADDRESS/32 -j MARK --set-mark $FILTER_ID
;;
esac
```

Script chilli.condown.sh:

```
#!/bin/bash
############################################################
#                      chilli.condown.sh
########
# Script para configurar el parámetro CONDOWN
# Especificamente BORRA la regla que marca cada paquete con
# el atributo Radius FILTER_ID que determina el perfil del
# usuario
# En sólo para los DHCP leases que Chillispot asigna
############################################################
#
# En este caso: Network: 192.168.1.176.0/21
# Static Network:  192.168.176.0/22
# Dynamic Network: 192.168.180.0/22
#    Chillispot Dynamic Network: 192.168.182.0/23
############################################################
#
# Version V.1.1 28-04-2007
############################################################
#
IPTABLES="/sbin/iptables"

case "$FRAMED_IP_ADDRESS" in
     192.168.182*) $IPTABLES -D marca-dinamicas -t mangle -s
$FRAMED_IP_ADDRESS/32 -j MARK --set-mark $FILTER_ID
      ;;
     192.168.183*) $IPTABLES -D marca-dinamicas -t mangle -s
$FRAMED_IP_ADDRESS/32 -j MARK --set-mark $FILTER_ID
      ;;
esac
```

Los *scripts* que se observan a continuación insertan las reglas correspondientes a las direcciones IP dinámicas entregadas por el *Servidor PPPoE*. Los mismos se ejecutan al iniciar y finalizar una conexión y se denominan *pppoe-conup* y *pppoe-condown* respectivamente.

Script /etc/ppp/ip-up.d/pppoe-conup:

```
#!/bin/bash
############################################################
#Script para configurar Politicas de Acceso en PPPoE,
# Especificamente marca cada paquete con el atributo
# Radius FILTER_ID que determina el perfil del usuario
# En sólo para los DHCP leases que PPPoE asigna
############################################################
```

```
# En este caso: Network: 192.168.1.176.0/21
# Static Network:  192.168.176.0/22
# Dynamic Network: 192.168.180.0/22
#      PPPoE Dynamic Network: 192.168.180.0/23
#      Chillispot Dynamic Network: 192.168.182.0/23
# Requiere de radattr.so plugin
############################################################
# Version 1.1  25-07-2007
############################################################
#
IPTABLES="/sbin/iptables"

FILTER_ID=`cat /var/run/radattr.$PPP_IFACE | awk '/^Filter-
Id/ { print $2} '`

case "$PPP_REMOTE" in
    192.168.180*) $IPTABLES -A marca-dinamicas -t mangle -s
$PPP_REMOTE/32 -j MARK --set-mark $FILTER_ID ;;
    192.168.181*) $IPTABLES -A marca-dinamicas -t mangle -s
$PPP_REMOTE/32 -j MARK --set-mark $FILTER_ID ;;
esac
```

Script /etc/ppp/ip-up.d/pppoe-condown:

```
#!/bin/bash
############################################################
# Script para configurar Politicas de Acceso en PPPoE,
# Especificamente marca cada paquete con el atributo
# Radius FILTER_ID
# que determina el perfil del usuario
# En sólo para los DHCP leases que PPPoE asigna
############################################################
# En este caso: Network: 192.168.1.176.0/21
# Static Network:  192.168.176.0/22
# Dynamic Network: 192.168.180.0/22
#      PPPoE Dynamic Network: 192.168.180.0/23
#      Chillispot Dynamic Network: 192.168.182.0/23
# Requiere de radattr.so plugin
############################################################
# Version 1.1  25-07-2007
############################################################
IPTABLES="/sbin/iptables"

FILTER_ID=`cat /var/run/radattr.$PPP_IFACE | awk '/^Filter-
Id/ { print $2} '`

case "$PPP_REMOTE" in
    192.168.180*) $IPTABLES -D marca-dinamicas -t mangle -s
$PPP_REMOTE/32 -j MARK --set-mark $FILTER_ID ;;
    192.168.181*) $IPTABLES -D marca-dinamicas -t mangle -s
```

```
$PPP_REMOTE/32 -j MARK --set-mark $FILTER_ID ;;
esac
```

6. APÉNDICE F: PRUEBAS DE PPPOE CON CIFRADO MPPE

Si bien es deseable utilizar cifrado de datos en el método de accesos *PPPoE*, se encuentran dos problemas relacionados a la hora de implementar el cifrado.

Las pruebas de cifrado con *MPPE (Microsoft Point-To-Point Encryption Protocol)*[MPPE] se llevaron a cabo usando en el prototipo del *servidor de autenticación* el demonio *pppd-2.4.4* y clientes con sistema operativo MS-Windows XPSP2, *MS-Windows 2000 SP4 y MS-Vista Ultimate.*

En los clientes se desactivo la compresión del tráfico, ya que la misma utiliza el algoritmo *MPPC (Microsoft Point-to-Point Compression Protocol*) [MPPC], el cual se encuentra protegido con una licencia que impide la implementación en software libre.

Se observo que en la negociación previa al establecimiento de la conexión *PPP,* en efecto, no se utiliza *MPPC* y se acuerda el uso *de MPPE* entre el servidor y los clientes.

La conexión se establece correctamente, pero cuando el servidor recibe un paquete, responde con un mensaje *Unsupported protocol 0xnnn received* y envía al cliente un mensaje *LCP Protocol Reject.* El número de protocolo varía de forma aleatoria como se ve en el cuadro de Texto VIII.28.

Se realizó una prueba recompilando *el kernel del servidor de autenticación* aplicando un parche que implementa el *algoritmo MPPC.* Dicho parche sólo puede aplicarse hasta la versión de kernel 2.6.13 y puede encontrarse en *http://mppe-mppc.alphacron.de/* .

En las pruebas realizadas con kernel 2.6.13 y el parche *MPPC, f*ue posible usar *cifrado MPPE y compresión MPPC* satisfactoriamente.

El comportamiento observado en las primeras pruebas (sin usar el parche *MPPC)* se podría atribuir a un desfasaje de las cabeceras del protocolo.

Se decide no usar el parche *MPPC* debido a que impide cumplir con el objetivo propuesto de usar software libre y protocolos abiertos en la implementación del modelo de infraestructura. Además, no permite actualizar el kernel de *GNU/Linux* mas allá de la versión 2.6.13.

```
........
rcvd [CCP ConfNak id=0x1 <mppe -H -M -S +L -D -C>]
sent [CCP ConfReq id=0x2 <mppe -H -M -S +L -D -C>]
rcvd [CCP ConfReq id=0x7 <mppe -H -M -S +L -D -C>]
sent [CCP ConfAck id=0x7 <mppe -H -M -S +L -D -C>]
rcvd [CCP ConfAck id=0x2 <mppe -H -M -S +L -D -C>]
MPPE 40-bit stateful compression enabled
........
rcvd [proto=0x49] 31 80 d6 bf 69 32 85 15 1d 55 21 40 04 d7 e7
fb b7 63 ec 39 87 39 9d 5e 31 71 b2 9a 38 e9 7e e1 ...
Unsupported protocol 'Serial Data Transport Protocol (PPP-
SDTP)' (0x49) received
sent [LCP ProtRej id=0x2 00 49 31 80 d6 bf 69 32 85 15 1d 55 21
40 04 d7 e7 fb b7 63 ec 39 87 39 9d 5e 31 71 b2 9a 38 e9 ...]
rcvd [proto=0x6a35] 3b 60 93 46 a1 f8 6a 6e cc be 6c 49 56 d3
c6 8e 1c 33 f3 a2 ae 0a 9a 93 4f 38 cb 00 a6 b2 ee 3c ...
Unsupported protocol 0x6a35 received
sent [LCP ProtRej id=0x3 6a 35 3b 60 93 46 a1 f8 6a 6e cc be 6c
49 56 d3 c6 8e 1c 33 f3 a2 ae 0a 9a 93 4f 38 cb 00 a6 b2 ...]
..........
```

Texto VIII.28.*: Fragmento del log de conexión PPPoE.*

2. APÉNDICE G: BIBLIOGRAFÍA

GAST00: Matthew Gast; 802.11 Wireless Networks The Definitive Guide, 2005, ISBN: 0-596-10052-3 .
IEEE80211: IEEE Computer Society; Wireless LAN MAC and PHY specifications, Junio 2001, www.ieee.org .
IEEE802: IEEE Computer Society; IEEE Standard for Local and Metropolitan Area Networks: Overview and Arch., Marzo 2002, www.ieee.org .
ISOOSI: ISO; IT-Open Systems Interconnection - Basic Reference Model ISO/IEC 7498-1, Junio 1996, www.iso.org .
IEEE8023: IEEE Computer Society; Carrier sense multiple access with collision detection access method., Diciembre-2005, www.ieee.org .
IEEE8021X: IEEE Computer Society; IEEE Standards 802.1X Port based Network Access Control, Junio 2001, .
CHAN00: Praphul Chandra; Bulletproof Wireless Security GSM, UMTS, 802.11 and Ad Hoc Security, 2005, ISBN:0-7506-7746-5 .
EAP: Aboba, B., Blunk, L., Vollbrechtand, J., Carlson, J., y H. Levkowetz; "Extensible Authentication Protocol (EAP)" - RFC 3748, Junio 2004, www.ietf.org .
PPP: W. Simpson; "The Point-to-Point Protocol (PPP)" - RFC1661, Julio 1994, .
EAP-TLS: B. Aboba, D. Simon; "PPP EAP TLS Authentication Protocol" - RFC 2716, Octubre 1999, www.ietf.org .
EAP-TTLS: Paul Funk, Simon Blake-Wilson; "EAP Tunneled TLS Authentication Protocol (EAP-TTLS)" Internet-Draft, Julio 2004, www.ietf.org .
PEAP: V.Kamath, A. Palekar, M. Wodrich; "Microsoft's PEAP version 0 (Implementation in Win XPSP1)"- Internet Draft, Octubre 2002, www.ietf.org .
PPPoE: L. Mamakos, K. Lidl, J. Evans, d. Carrel, D. Simone, R. Wheeler; "A Method for Transmitting PPP Over Ethernet (PPPoE)" - RFC2516, Febrero 1999, www.ietf.org .
LDAP1: J. Sermersheim; "Lightweight Directory Access Protocol (LDAP): The Protocol" - RFC4511, Junio 2006, www.ietf.org .
LDAP2: K. Zeilenga; "LDAP : Directory Information Models"-RFC 4512, Junio 2006, www.ietf.org .
RADIUS: C. Rigney, S. Willens, A. Rubens, W.Simpson; "Remote Authentication Dial In User Service (RADIUS)" - RFC2865, Junio 2000, www.ietf.org .
RADIUSACC: C.Rigney; "RADIUS Accounting" - RFC2866, Junio 2000,

www.ietf.org .
DRAFT-GPATERNO: Giuseppe Paterno; "Using PPP-over-Ethernet (PPPoE) in Wireless LANs", Noviembre 2003, www.ietf.org .
MPPE: G. Pall, G. Zorn.; "Microsoft Point-To-Point Encryption (MPPE) Protocol" - RFC 3078, Marzo 2001, www.ietf.org .
MPPC: G. Pall; "Microsoft Point-To-Point Compression (MPPC) Protocol" - RFC 2118, Marzo 1997, www.ietf.org .

1. APÉNDICE H: ÍNDICE DE ILUSTRACIONES

2. APÉNDICE I: ÍNDICE DE TABLAS