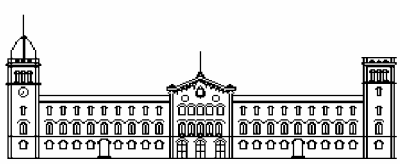


**Treball fi de carrera**

**ENGINYERIA TÈCNICA EN INFORMÀTICA DE SISTEMES**

**Facultat de Matemàtiques**
**Universitat de Barcelona**

---

# IMPLEMENTACIÓ AMB CONTROL DE SEGURETAT D'UNA XARXA WIRELESS USANT SOFTWARE LLIURE

---

**Iván Sánchez Valencia**

Director: Jaume Timoneda Salat
Realitzat a: Departament de Matemàtica Aplicada i Anàlisi. UB

Barcelona, 9 de juliol de 2004

Jaume Timoneda Salat
Titular d'Escola Universitària de Matemàtica Aplicada
Departament de Matemàtica Aplicada i Anàlisi
de la Universitat de Barcelona

Autoritzo:

Al dipòsit del Treball fi de carrera realitzat sota la meva direcció per Iván Sánchez Valencia.

Firma.

Barcelona, 9 de juliol 2004

# RESUMEN

El propósito de este proyecto es implementar controles de seguridad en redes inalámbricas utilizando software libre. La idea es utilizar un PC con una tarjeta inalámbrica para hacer de punto de acceso, utilizando como base el sistema operativo Linux. Antes de proponer soluciones seguras y fiables (a día de hoy), se analizará la evolución de los mecanismos de seguridad que se venían utilizando hasta ahora y se indicarán cuales son sus debilidades y las soluciones que ofrecen los nuevos protocolos ante estas debilidades.

Además, en las primeras soluciones las claves de cifrado son las mismas para todos los clientes y no hay forma de identificarlos ni tampoco denegar acceso a un único cliente. En las soluciones finales se ofrece la posibilidad de autenticar a cada uno de los clientes por separado y utilizar claves de cifrado diferentes para cada uno.

# INDICE

# INTRODUCCIÓN

El objetivo de este proyecto es analizar los controles de seguridad más comunes para el acceso a una red wireless que existen. Algunos de estos métodos son todavía utilizados a pesar de considerarse inseguros, otros apenas se conocen y otros se están utilizando o empezando a utilizar en entornos empresariales, los cuales a priori, requieren un mayor control de seguridad.

A parte de este análisis, también se tratará de explicar los pasos a seguir para montar y configurar un punto de acceso sobre Linux que implemente los controles de seguridad que se consideran más seguros hoy en día.

Antes de empezar, se hará una breve introducción a las tecnologías wireless, así como a sus estándares.

Hasta hace relativamente poco para conectarnos a Internet, necesitábamos pertenecer a una red. Estas redes eran físicas, se tenía que estar siempre buscando una toma de red y un cable para conectar nuestro ordenador con esta red. La necesidad de estar permanentemente en Internet y sobre todo la movilidad, están haciendo que las redes inalámbricas vayan apareciendo poco a poco en empresas, casas, aeropuertos, hoteles… Estas redes permiten que una persona pueda moverse, si necesidad de una conexión física para estar conectada.

Las redes inalámbricas se están introduciendo en el mercado de consumo gracias a unos precios muy asequibles y a un conjunto de entusiastas que han visto las enormes posibilidades de esta tecnología. Las aplicaciones de las redes inalámbricas son infinitas, y van a crear una nueva forma de usar la información.

Lo primero que tenemos que hacer antes que nada es situarnos dentro del mundo inalámbrico. Para ello vamos a hacer una primera clasificación que nos centre ante las diferentes variantes que podemos encontrarnos.

# 1. Redes inalámbricas personales y de consumo

## 1.1. Redes inalámbricas personales

Dentro del ámbito de estas redes podemos integrar a dos principales actores:

a- En primer lugar, y ya conocidas por bastantes usuarios, están las redes que se usan actualmente para el intercambio de información mediante **infrarrojos**. Estas redes son muy limitadas dado su corto alcance, necesidad de "visión sin obstáculos" entre los dispositivos que se comunican y su baja velocidad (hasta 115 kbps). Se encuentran principalmente en ordenadores portátiles, PDAs (Agendas electrónicas personales), teléfonos móviles y algunas impresoras.

b- En segundo lugar **Bluetooth**, estándar de comunicación entre pequeños dispositivos de uso personal, como pueden ser los PDAs, teléfonos móviles de nueva generación y algún que otro ordenador portátil. Su principal desventaja es que su puesta en marcha se ha ido retrasando desde hace años y la aparición del mismo ha ido plagada de diferencias e incompatibilidades entre los dispositivos de comunicación de los distintos fabricantes que ha imposibilitado su rápida adopción. Opera dentro de la banda de los 2.4 Ghz. Para más información sobre el mismo se puede consultar http://www.bluetooth.com.

## 1.2. Redes inalámbricas de consumo

Distinguiremos dos tipos diferentes:

a- **Redes CDMA** (estándar de telefonía móvil estadounidense) y **GSM** (estándar de telefonía móvil europeo y asiático). Son los estándares que usa la telefonía móvil empleados alrededor de todo el mundo en sus diferentes variantes. Vea http://www.gsmworld.com.

b- **802.16** son redes que pretenden complementar a las anteriores estableciendo redes inalámbricas metropolitanas (MAN) en la banda de entre los 2 y los 11 Ghz.

# 2. Redes inalámbricas 802.11

Las redes inalámbricas básicamente se diferencian de las redes conocidas hasta ahora por el enfoque que toman de los niveles más bajos de la pila OSI, el nivel físico y el nivel de enlace, los cuales se definen por el estándar 802.11 del IEEE.

Como suele pasar siempre que un estándar aparece y los grandes fabricantes se interesan por él, aparecen diferentes aproximaciones al mismo lo que genera una incipiente confusión.

Nos encontramos ante tres principales variantes:

a- **802.11a:** fue la primera versión de las redes inalámbricas y llega a alcanzar velocidades de hasta 54 Mbps dentro de los estándares del IEEE y hasta 72 y 108 Mbps con tecnologías de desdoblamiento de la velocidad ofrecidas por diferentes fabricantes, pero que no están (a día de hoy) estandarizadas por el IEEE. Esta variante opera dentro del rango de los 5 Ghz. Inicialmente se soportan hasta 64 usuarios por Punto de Acceso.
Sus principales ventajas son su velocidad, la base instalada de dispositivos de este tipo, la gratuidad de la frecuencia que usa y la ausencia de interferencias en la misma.

Sus principales desventajas son su incompatibilidad con los estándares 802.11b y g, la no incorporación de QoS, la no disponibilidad de esta frecuencia en Europa dado que esta frecuencia está reservada a la HyperLAN2 (Ver http://www.hiperlan2.com) y la parcial disponibilidad de la misma en Japón.

El hecho de no estar disponible en Europa prácticamente la descarta de nuestras posibilidades de elección para instalaciones en este continente.

b- **802.11b:** es la segunda versión de las redes inalámbricas. Alcanza una velocidad de 11 Mbps estandarizada por el IEEE y una velocidad de 22 Mbps por el desdoblamiento de la velocidad que ofrecen algunos fabricantes pero sin la estandarización (a día de hoy) del IEEE. Opera dentro de la frecuencia de los 2.4 Ghz. Inicialmente se soportan hasta 32 usuarios por Punto de Acceso.

Adolece de varios de los inconvenientes que tiene el 802.11a como son la falta de QoS, además de otros problemas como la masificación de la frecuencia en la que transmite y recibe, pues en los 2.4 Ghz funcionan teléfonos inalámbricos, teclados y ratones inalámbricos, hornos microondas, dispositivos Bluetooth..., lo cual puede provocar interferencias.

En el lado positivo está su rápida adopción por parte de una gran comunidad de usuarios debido principalmente a unos muy bajos precios de sus dispositivos, la gratuidad de la banda que utiliza así como de su disponibilidad alrededor de todo el mundo. Está estandarizado por el IEEE.

c- **802.11g:** Es la tercera versión de las redes inalámbricas, y se basa en la compatibilidad con los dispositivos 802.11b y en el ofrecer unas velocidades de hasta 54 Mbps. Funciona dentro de la frecuencia de 2.4 Ghz y dispone de las mismas ventajas e inconvenientes que el 802.11b con la salvedad de que su velocidad es mayor.

Una vez nos centramos en el tipo de red que queremos, en nuestro caso las redes inalámbricas 802.11, es conveniente continuar con una división entre la topología y el modo de funcionamiento de los dispositivos wireless.

Existen dos topologías básicas:

- **Topología Ad-Hoc**. Cada dispositivo se puede comunicar con todos los demás. Cada nodo forma parte de una red **Peer to Peer** o de igual a igual, para lo cual sólo vamos a necesitar el disponer de un SSID igual para todos los nodos y no sobrepasar un número razonable de dispositivos que hagan bajar el rendimiento. A mayor dispersión geográfica de los nodos más dispositivos pueden formar parte de la red, aunque algunos no lleguen a verse entre si y por lo tanto tampoco se puedan comunicar entre ellos. En esta topología no tiene sentido hablar de Puntos de Acceso y los Terminales de Red se deberán poner en modo adhoc para trabajar con esta topología.

- **Topología Infraestructura**, en el cual existe un nodo central al que se le llama Punto de Acceso (AP) y que sirve de enlace para todos los demás clientes (Terminales de Red). Este nodo sirve para encaminar las tramas hacia cualquier otro tipo red. Para poder establecerse la comunicación, todos los nodos deben estar dentro de la zona de cobertura del AP, dos clientes no necesitan tenerse al alcance entre ellos, ya que siempre se comunicaran mediante el AP.

Todos los dispositivos, independientemente de que sean Terminales de Red o Puntos de Acceso tienen dos modos de funcionamiento en el caso de una Topología Infraestructura:

- **Modo Managed**, es el modo en el que el Terminal de red se conecta al AP para que éste último le sirva de "concentrador". El Terminal de Red sólo se comunica con el AP.

- **Modo Master**. Este modo es el modo en el que trabaja el Punto de Acceso, pero en el que también pueden entrar los Terminales de Red si se dispone del chipset y firmware apropiado o de un ordenador que sea capaz de realizar la funcionalidad requerida.

Estos modos de funcionamiento nos sugieren que básicamente los dispositivos WiFi son todos iguales, siendo los que funcionan como Puntos de Acceso realmente Terminales de Red a los que se les ha añadido cierta funcionalidad extra vía firmware o vía software. Para realizar este papel se pueden emplear máquinas antiguas sin disco duro y bajo una distribución especial de Linux llamada LINUXAP – OPENAP o bien configurar manualmente un ordenador con Linux e instalarle y configurar todo lo necesario para que desarrolle la función que le queramos dar.

Esta afirmación se ve confirmada al descubrir que muchos Puntos de Acceso en realidad lo que tienen en su interior es una placa de circuitos integrados con un firmware añadido a un adaptador PCMCIA en el cual se le coloca una tarjeta PCMCIA idéntica a las que funcionan como Terminales de Red.

Adicionalmente, algunos Terminales de Red son capaces de funcionar en un Modo Monitor, mediante el cual recibirán todas las tramas 802.11, incluidas las administrativas.

Otro punto, en el cual centraremos gran parte de nuestra atención en este proyecto es el tema de la seguridad, que es uno de los puntos más importantes cuando se habla de redes inalámbricas. Desde el nacimiento de éstas, se ha intentado disponer de protocolos que garanticen las comunicaciones. Por ello es conveniente seguir puntual y escrupulosamente una serie de pasos que permitan disponer del grado máximo de seguridad que se sea capaz de obtener combinando todas las herramientas que se tengan al alcance.

Se debe tener en cuenta que cuando se trabaja con una red cableada convencional se dispone de un extra de seguridad, pues para conectarse a la misma normalmente hay que acceder al cable por el que circula la información o a los dispositivos físicos de comunicación de la misma. Pero en el caso de una red wireless no sucede lo mismo, de hecho se va a estar "desperdigando" la información hacia los cuatro vientos con todo lo que conlleva.

## 2.1. Asociación y autenticación a un AP

Para que un Terminal de Red sea capaz de comunicarse con otros Terminales wireless, primero debe existir una asociación con un AP. La asociación de un Terminal de Red con un AP sería el equivalente a enchufar un cable de red ethernet en una toma de red. Si no hay asociación, no hay conexión física.

Mientras que a las estaciones terminales se les permite asociarse de forma dinámica con otros APs, en cualquier momento una estación terminal solamente puede estar asociada con un AP.

La asociación es un proceso de tres pasos:

1. Desautenticado y desasociado
2. Autenticado y desasociado
3. Autenticado y asociado

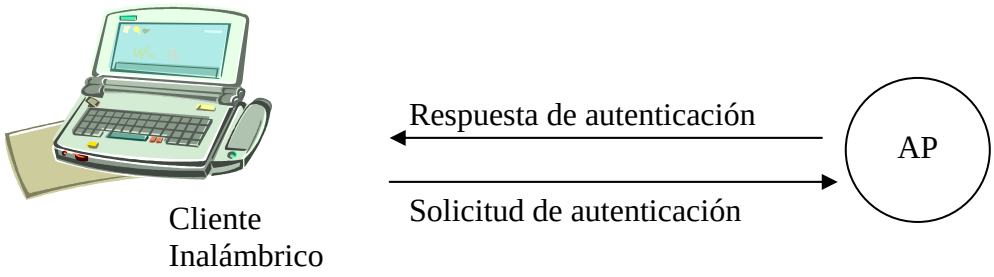
Por lo tanto, para que haya asociación, es requisito indispensable que haya autenticación.

A los mensajes pasados durante estos pasos se les llama tramas de administración (*management frames*). La parte importante en la que se debe hacer énfasis en este proceso es que la asociación no ocurrirá hasta que la autenticación se lleve a cabo.

## 2.2. Modos de autenticación en Infraestructura

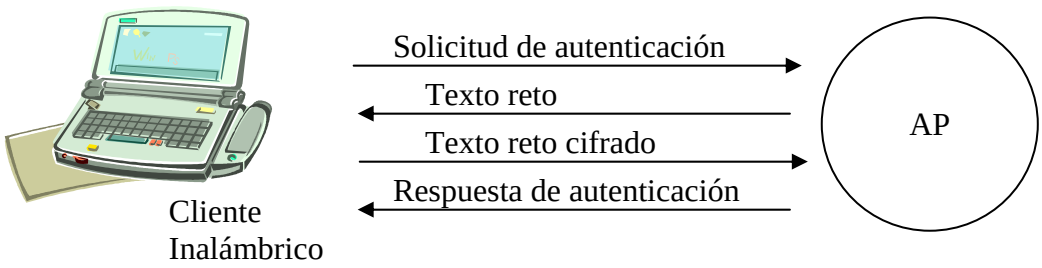
Para que exista asociación, ha tenido que haber autenticación previa. Existen dos modos diferentes de autenticación, OSA y SKA:

1. OSA es una forma muy básica de autenticación. El AP autenticará cualquier Terminal de Red que solicite autorización para asociarse a la red, lo que significa que cualquier Terminal de Red estará autorizado y se podrá asociar con el AP.




2. SKA. La autenticación de llave compartida está basada en el hecho de que ambas estaciones tomando parte en el proceso de autenticación tienen la misma llave "compartida". Se asume que esta llave ha sido trasmitida a ambas estaciones a través de un canal seguro que no sea la propia red wireless. En implementaciones típicas, esto se podría configurar manualmente en la estación cliente y en el AP.

   La primera y la cuarta trama de autenticación de llave compartida son similares a aquella encontrada en sistemas de autenticación abierta (OSA). La diferencia está en la existencia de una segunda y tercera tramas inexistentes en OSA. El Terminal de Red recibe un paquete de texto que es un reto creado usando un generador de números pseudo aleatorios desde el AP, lo cifra usando la llave compartida, y luego lo manda de regreso al AP. Si después de descifrarlo, el texto de reto es igual, entonces la autenticación en *un* sentido es satisfactoria. Para obtener la autenticación mutua, el proceso se repite en la dirección opuesta.




La mayor parte de los ataques contra WEP están basados en capturar la forma cifrada de un texto conocido, con lo que este modo de autenticación es una mala elección ya que da a los atacantes la información necesaria para romper el cifrado WEP y es por lo que la llave de autenticación compartida nunca es recomendada.

La utilización de OSA no implica el no poder utilizar WEP, de hecho es mejor utilizar la autenticación abierta (OSA) y cifrado WEP, que permitirá la asociación y autenticación sin la llave WEP correcta a los clientes pero siempre manteniendo la seguridad de la red (dentro de las limitaciones de WEP), ya que las estaciones que no conozcan la clave WEP válida no serán capaces de enviar o recibir información de forma correcta.

# ESTÁNDARES WIFI – 802.11

- **802.11a** Estándar de comunicación en la banda de los 5 Ghz. (Explicado en la introducción). Velocidades de hasta 54 Mbps.

- **802.11b** Estándar de comunicación en la banda de los 2.4 Ghz. (Explicado en la introducción). Velocidades de hasta 11 Mbps

- **802.11c** Estándar que define las características que necesitan los Puntos de Acceso para actuar como puentes (bridges). Ya está aprobado y se implementa en algunos productos.

- **802.11d** Estándar que permite el uso de la comunicación mediante el protocolo 802.11 en países que tienen restricciones sobre el uso de las frecuencias que éste es capaz de utilizar. De esta forma se puede usar en cualquier parte del mundo.

- **802.11e** Estándar sobre la introducción del QoS en la comunicación entre Puntos de Acceso y Terminales de Red. Actúa como árbitro de la comunicación. Esto permitirá el envío de vídeo y de voz sobre IP.

- **802.11f** Estándar que define una práctica recomendada de uso sobre el intercambio de información entre el Punto de Acceso y el Terminal de Red en el momento del registro a la red y la información que intercambian los Puntos de Acceso para permitir la interportabilidad. La adopción de esta práctica permitirá el Roamming entre diferentes redes.

- **802.11g** Estándar que permite la comunicación en la banda de los 2.4 Ghz. (Explicado en la introducción). Velocidades de hasta 54 Mbps.

- **802.11h** Estándar que sobrepasa al 802.11a al permitir la asignación dinámica de canales para permitir la coexistencia de éste con el HyperLAN. Además define el TPC (*Transmit Power Control*) según el cual la potencia de transmisión se adecua a la distancia a la que se encuentra el destinatario de la comunicación.

- **802.11i** Estándar que define el cifrado y la autenticación para complementar completar y mejorar el protocolo WEP. Es un estándar que mejorará la seguridad de las comunicaciones mediante el uso del protocolo TKIP (*Temporal Key Integrity Protocol*).

- **802.11j** Estándar que permitirá la armonización entre el IEEE, el ETSI HyperLAN2, ARIB e HISWANa.

- **802.11m** Estándar propuesto para el mantenimiento de las redes inalámbricas.

# PREPARANDO EL PUNTO DE ACCESO

El objetivo principal del proyecto consiste en ofrecer una solución viable a la implementación de una red wireless con control de seguridad lo más robusta posible y basada en software libre.

Se partirá de la base de montar un punto de acceso mediante un viejo PC con una tarjeta wireless 802.11b con chipset Prism2/2.5/3 de Intersil (teóricamente también se podría realizar con tarjetas 802.11g con chipset PrismGT/Duette/Indigo, pero no se dispone de dichas tarjetas para probarlas), pero con unas modificaciones que darán soporte al PC para comportarse en modo Master, es decir, como un punto de acceso.

Pasamos a definir los elementos importantes que compondrán el sistema base de nuestro punto de acceso, a nivel de kernel y librerías. Se utilizará el sistema operativo GNU/Linux, con las librerías LibSafe y con un kernel con soporte hostap, freeswan y grsecurity que nos ofrecerá una funcionalidad y seguridad extra:

- **Kernel:** Es el núcleo del sistema operativo, el corazón del sistema y se debe configurar acorde con las necesidades que se tienen.

- **LibSafe:** Se trata de una librería que se carga siempre antes que ninguna otra y comprueba las llamadas de funciones como `strcpy`, `memcpy`, etc. previniendo al sistema ante posibles buffer overflows, bloqueando dicha llamada y enviando una alerta en caso de detectar un intento de overflow.

- **GrSecurity:** aporta seguridad local y remota. Previene al sistema de los típicos ataques (buffer overflows, intento de ejecución de código en pila, etc.) así como impone nuevas restricciones al sistema (limitaciones a la hora de acceder al sistema de ficheros `/proc`, no permite montar unidades en un entorno chroot, no permite escrituras en `/dev/kmem`, etc.). Más información en http://www.grsecurity.org.

- **Hostap-driver:** con estos parches y una tarjeta apropiada, daremos la posibilidad a nuestro sistema de que se comporte como un punto de acceso.

- **Freeswan:** damos soporte al kernel para establecer VPNs (redes privadas virtuales) mediante el protocolo IPSec, aporta seguridad en la comunicación a nivel de red (IP).

Además de la preparación del software a instalar, también será necesario disponer de un firmware versión 1.7.0 o posterior en nuestras tarjetas wireless, que como ya hemos dicho anteriormente, deben tener el chipset Prism2/2.5/3 de Intersil.

Partiendo de dicha base, y haciendo uso de diferentes utilidades y programas, se propondrán diversas posibles soluciones para una implementación de una red wireless con control de seguridad.

# 1. Kernel

En este apartado se tratará de explicar y orientar en la configuración y compilación de un nuevo kernel para el AP. Se indicarán los pasos a seguir y las opciones especiales que se requieran seleccionar en la configuración del sistema.

Lo primero que se debe hacer es descargarse las últimas fuentes de la version 2.4 de su página oficial, http://www.kernel.org, a fecha actual la última version disponible es la 2.4.26.

Una vez descargadas las fuentes, se tendrá un fichero llamado `linux-2.4.26.tar.bz2`, se descomprime estando en el directorio `/usr/src` con el comando `tar jxvf` y se entra en el directorio `linux-2.4.26` que se habrá creado dentro de `/usr/src`.

Una vez se tiene el código fuente listo y estando en el directorio `/usr/src/linux`, el cual será un enlace simbólico al directorio que contenga el código fuente, se deberá ejecutar el comando `make menuconfig` para obtener un menú en el que podremos navegar a través de las diferentes opciones de configuración que nos ofrece el núcleo.

Una vez estamos en el menú, se seleccionarán las opciones necesarias para nuestra máquina (cada PC tiene una configuración diferente, y se deberá conocer el hardware de la máquina para seleccionar y dar soporte a las opciones requeridas) y nos aseguraremos de tener seleccionadas también las siguientes opciones:

En `Loadable module support --->` seleccionar las tres opciones que aparecen dentro, con esto daremos soporte para la carga de módulos al kernel.

En `General setup --->` seleccionaremos las siguientes opciones y así se tendrá soporte para las tarjetas PCMCIA y para el interfaz sysctl:

```
[*] Sysctl support

PCMCIA/CardBus support  --->
      <*> PCMCIA/CardBus support
      [*]   CardBus support
```

Se seleccionarán las opciones de red que se indicarán a continuación en el apartado `Networking options ---->` para obtener el soporte necesario para los servicios deseados (firewall, dhcp,…):

```
      <*> Packet socket
      [*] Network packet filtering (replaces ipchains)
      [*] Socket Filtering
      <*> Unix domain sockets
      [*] TCP/IP Networking
      [*]  IP: multicasting
      [*]  IP: advanced router
      [*]   IP: policy routing
      [*]    IP: use netfilter MARK value as routing key
      [*]    IP: fast network address translation
      [*]   IP: use TOS value as routing key
      IP: Netfilter Configuration  --->
         <*> Connection tracking (required for masq/NAT)
         <*>   FTP protocol support
         <*> IP tables support
         <*>   MAC address match support
         <*>   netfilter MARK match support
         <*>   Multiple port match support
         <*>   TOS match support
         <*>   AH/ESP match support
         <*>   stealth match support
         <*>   Helper match support
         <*>   Connection state match support
         <*>   Connection tracking match support
         <*>   Packet filtering
         <*>     REJECT target support
         <*>   Full NAT
         <*>     MASQUERADE target support
         <*>     REDIRECT target support
         <*>   Packet mangling
         <*>     TOS target support
         <*>     MARK target support
         <*>   LOG target support
```

Se puede obtener más información sobre lo que hace cada una de las opciones seleccionadas en la opción `< Help >` del menú.

A continuación se entra en el menú `Network device support ---->` y se selecciona la opción `[*] Network device support`. Se seleccionará únicamente la tarjeta ethernet que se posea en el AP, pero no la tarjeta de red inalámbrica, ya que el driver que se utilizará será uno externo al código del kernel. Más adelante se verá como configurarlos e instalarlos.

Una vez se han seleccionado todas las opciones, se sale del menú de configuración y se ejecutan los siguientes comandos:

- **`make dep:`** prepara las dependencias necesarias para la compilación.

- **`make clean:`** borra los ficheros innecesarios que puedan existir.

- **`make bzImage:`** compila el kernel Linux con la configuración deseada.

- **`make modules:`** compila los módulos previamente seleccionados.

- **`make modules_install:`** instala los módulos compilados anteriormente

Únicamente queda instalar el kernel y reiniciar el sistema para arrancar con él.

## 2. LibSafe

LibSafe es una librería que se carga en el sistema e intercepta todas las llamadas a funciones de librerías que se sabe que pueden tener fallos de seguridad y que son conocidas como vulnerables.

Internamente, implementa la misma funcionalidad de dichas funciones interceptadas con la salvedad que comprobará una posible sobreescritura incontrolada de datos, evitando así la sobreescritura de la dirección de retorno.

La instalación de dicha librería es muy sencilla, tan solo es necesario instalar el paquete de libsafe en caso de estar disponible para nuestra distribución Linux, o bien obtener el código fuente, compilarlo y copiar la librería en `/lib/`.

Hay dos ficheros de configuración y otro que se debe modificar para utilizar la librería:

- **`/etc/libsafe.exclude`:** Se configuran los programas a los que no se quiera interceptar las llamadas a funciones, se conocen ciertos programas que dejan de funcionar si no se incluyen en esta lista, como por ejemplo `/sbin/hwclock`.

- **`/etc/libsafe.notify`:** Se configura una dirección de correo electrónico a la que se enviara un mensaje en caso de detectar un intento de overflow indicando el usuario, ejecutable, fecha y hora del intento.

- **`/etc/ld.so.preload`:** Se debe incluir la ruta a la librería, para que la cargue el sistema siempre que tenga que ejecutar algún binario y así pueda realizar su cometido.

Las ventajas que tiene esta solución frente a otras disponibles actualmente, es que no es necesario modificar ninguna parte del sistema, tan solo compilar, instalar la librería e incluirla en `ld.so.preload` para que el sistema la cargue siempre antes que ninguna otra.

# 3. GrSecurity

Se trata de unos parches de seguridad para el kernel disponibles en http://www.grsecurity.org.

GrSecurity es una mejora innovadora a la seguridad, utiliza un modelo multicapa separando la detección, prevención y contención. Tiene licencia GPL y ofrece entre muchas otras características:

- Un sistema inteligente y robusto del control de acceso (RBAC) que puede generar políticas con menos privilegio para todo su sistema sin configurarlo. Para según que acciones no es suficiente ser súperusuario, sino que se podría requerir una nueva contraseña.

- Mejora en la seguridad de los entornos chroot incluyendo nuevas restricciones (prohibir montar dispositivos, realizar una llamada a `chroot` estando ya dentro de uno,…).

- Auditoria extensa, permite registrar una serie de eventos considerados potencialmente peligrosos, como el montar o desmontar un dispositivo.

- Prevención de todo tipo de *exploits* relacionados con los fallos de seguridad que tengan que ver con los espacios de direcciones.

- Aleatoriedad adicional en la pila TCP/IP.

- Restricción para que un usuario solo pueda ver sus procesos.

- Cada alarma o intervención de seguridad contiene la dirección IP de la persona que causó la alarma.

- Actuación sobre el sistema de ficheros `/proc`, restringiendo parte de la información a los usuarios normales de sistema.

Se puede obtener una lista completa de las posibilidades de los parches así como documentación extensa de su funcionamiento en su página web.

La instalación de dichos parches tan solo requiere disponer del código de los parches del kernel deseado y aplicar los parches con el comando `patch`. Posteriormente, tan solo se deberá reconfigurar el kernel para seleccionar las nuevas opciones deseadas (`make menuconfig`) y posteriormente compilar el kernel (`make dep clean bzImage modules modules_install`), instalarlo y reiniciar la máquina.

Opcionalmente, y si se ha habilitado soporte para `sysctl`, se pueden activar y desactivar opciones de la seguridad de GrSec una vez esta el sistema en marcha, tan solo será necesario cambiar los valores deseados en los ficheros que se encuentran en `/proc/sys/kernel/grsecurity/`, como por ejemplo registrar en los logs del sistema un cambio de la fecha del sistema:

```
echo 1 > /proc/sys/kernel/grsecurity/timechange_logging
```

## 4. Hostap-driver

Se trata de un driver para Linux desarrollado para las tarjetas wireless basadas en el chipset Prism2/2.5/3 de Intersil.

Este driver soporta el llamado Modo Master (llamado Modo *HostAP* por los creadores del driver), el cual permite comportarse a una estación de trabajo como si fuera un punto de acceso.

En la página principal de dichos drivers se pueden encontrar utilidades extra que se verán más adelante.

Este driver aporta una gran flexibilidad a la hora de configurar un punto de acceso, ya que no se trata de un hardware propietario y con una funcionalidad cerrada y especifica la cual depende de su fabricante, sino que es totalmente configurable y actualizable por nosotros mismos, además de disponer del código fuente de dicho desarrollo. Para ampliar la funcionalidad de nuestro punto de acceso, tan solo hay que instalar los servicios que se deseen, al fin y al cabo es un PC con un sistema operativo libre y de código abierto.

Actualmente, está disponible la versión estable 0.1.3 de dicho driver pero que no se utilizará, ya que no aporta toda la funcionalidad deseada. En lugar de dicha versión, se utilizará la versión de desarrollo 0.2.3, que es la última versión existente en el momento de escribir este documento.

La principal aportación de la versión de desarrollo frente a la versión estable es la inclusión del soporte para WPA, tanto para el *authenticator* (punto de acceso) como para el *supplicant* (cliente wireless). Para poder utilizar WPA, a parte del driver hostap, también necesitaremos disponer como mínimo de la versión 1.7.0 del STA (station firmware) de la tarjeta. Además, también será necesario un demonio para realizar las tareas de mantenimiento de claves, `wpa_supplicant` en caso del cliente y `hostapd` en el caso del servidor. Más adelante se verá como conseguir las versiones de firmware necesarias, como actualizar las tarjetas y los posibles usos y configuraciones de los clientes (`wpa_supplicant`) y AP (`hostapd`).

La instalación del driver se puede realizar tanto en módulos como compilando el kernel con las fuentes parcheadas. Como siempre, en el caso de no tener soporte para módulos en el sistema, se gana en seguridad pero se pierde en modularidad. Tratándose de un driver en fase de desarrollo y debido a su constante evolución y actualización, se ha optado por la solución modular, ya que de lo contrario se debería recompilar todo el kernel y reiniciar el sistema cada vez que se necesitara actualizar la versión del driver.

La primera versión que se probó fue la 0.2.1, que tenía un fallo a la hora de hacer el re-keying de las claves de grupo utilizadas (las que se usan para enviar datos a todos los clientes de la red, broadcast), tema que se solucionó en la versión del servidor CVS y, por tanto, también en la siguiente versión.

Para la configuración de los parámetros internos de la tarjeta wireless, se utilizaran los comandos del paquete wireless-tools, disponible en cualquier distribución Linux, así como también los que nos ofrece el paquete hostap-utils, disponible en la misma página que hostap-driver.

Algunos de estos comandos son:

- **`iwpriv`:** Forma parte de las llamadas wireless-tools y es utilizado para definir el modo de funcionamiento de la tarjeta, por ejemplo si se quiere que trabaje en modo ad-hoc, managed, master o monitor. También se puede indicar el canal en el cual se desea que trabaje, del 1 al 13, siendo esta la lista de frecuencias disponibles:

    - 01 : 2.412 GHz
    - 02 : 2.417 GHz
    - 03 : 2.422 GHz
    - 04 : 2.427 GHz
    - 05 : 2.432 GHz
    - 06 : 2.437 GHz
    - 07 : 2.442 GHz
    - 08 : 2.447 GHz
    - 09 : 2.452 GHz
    - 10 : 2.457 GHz
    - 11 : 2.462 GHz
    - 12 : 2.467 GHz
    - 13 : 2.472 GHz

  También se puede configurar la política para la ACL de MACs:

```
iwpriv wlan0 maccmd <val>
        0: open policy for ACL (default)
        1: allow policy for ACL
        2: deny policy for ACL
        3: flush MAC access control list
        4: kick all authenticated stations
```

Así como configurar los enlaces WDS mediante:

- `iwpriv wlan0 wds_add <mac addr>`
- `iwpriv wlan0 wds_del <mac addr>`

Para obtener una lista detallada de todos los comandos disponibles, consultar el fichero *README* dentro del paquete hostap-driver.

- **`prism2_param`:** Es equivalente a `iwpriv wlan0 prism2_param <param> <val>`, con la diferencia que puede aportar algún parámetro nuevo a la configuración de la tarjeta si no disponemos de las últimas versiones de las wireless-tools.

- **`prism2_srec`:** Comando utilizado para actualizar el firmware de la tarjeta wireless. Para que `prism2_srec` pueda hacer su trabajo, se necesita tener soporte para actualización en el driver (hostap-driver).

# 5. FreeSWAN

Se trata de un paquete que implementa los protocolos IPSec en Linux. Con este paquete, podremos establecer VPNs (redes privadas virtuales).

El paquete contiene una parte de usuario y otra parte de kernel. La parte de usuario es un demonio, que se encargará de establecer los túneles preconfigurados por el administrador, y la parte de kernel, que se puede ejecutar o bien dentro del mismo kernel o bien como un módulo, será la encargada de formar a más bajo nivel los paquetes TCP/IP y encapsularlos dentro del túnel preconfigurado.

En algunos casos, si no se puede implementar ninguna de las soluciones seguras propuestas en este documento, ya sea por no disponer de sistemas Linux, por no disponer de los drivers adecuados para los clientes, por insuficiencia de recursos,... con lo que si no se hiciera algo al respecto, cabría la posibilidad de que alguien con conocimientos suficientes llegase a infiltrarse dentro de la red inalámbrica. Se puede recurrir a soluciones de más alto nivel, como es el de establecer redes privadas virtuales entre cada cliente y el punto de acceso.

Esta solución es una solución alternativa ante la posible imposibilidad de establecer un método físico seguro por el que comunicarse entre el punto de acceso y los clientes.

FreeSWAN implementa VPNs mediante el protocolo IPSec. Una de las principales ventajas con las que cuenta este protocolo es que esta soportado e implementado en muchos de los sistemas operativos (o se puede dar soporte) y dispositivos de electrónica de red, como puede ser el caso de los routers CISCO. Esto, que a priori puede parecer una tontería, es una gran ventaja que nos ofrece la posibilidad de establecer comunicaciones seguras con todos nuestros clientes, teniendo una garantía de comunicación hasta el destino final, ya sea de la red inalámbrica o de la cableada.

Esto es así ya que tenemos la posibilidad de establecer tantos túneles como queramos, con los dispositivos que queramos que no tienen porque estar en nuestra misma red wireless, ni tan siquiera en la red cableada.

Para la instalación del paquete FreeSWAN, primero deberemos descargarnos el código fuente de su página web y tras haberlo descomprimido y desempaquetado, tan solo deberemos ejecutar el comando `make menugo`, asegurándonos antes de tener en `/usr/src/linux` las fuentes del kernel que actualmente utilizamos ya compilado. Dicho comando, nos parcheará el código fuente del kernel para posteriormente ofrecernos la posibilidad de escoger como queremos compilar la parte de FreeSWAN, bien dentro del propio kernel o bien en forma de módulos. Acto seguido, se compilará todo el kernel y módulos de nuevo, así como los demonios necesarios para el funcionamiento de FreeSWAN. Finalmente solo nos quedará cargar el módulo en memoria y arrancar el demonio, o instalar el nuevo kernel y reiniciar la máquina en caso no modular.

Para todos los parámetros de configuración y las claves públicas y privadas de la máquina en cuestión, necesitaremos editar los ficheros `/etc/ipsec.conf` y `/etc/ipsec.secrets`, como se indicará más adelante.

# CONFIGURACIÓN BÁSICA DEL AP

Se empezará definiendo la base para cualquiera de las soluciones propuestas, que es la instalación y configuración básica de la máquina que hará de punto de acceso.

Una breve lista de los servicios que deberá tener dicha máquina es la siguiente:

- **SSH:** Para la administración remota del punto de acceso.

- **DHCP:** Se encargará de la asignación de los parámetros de red a los clientes.

- **DNS:** Servicio encargado de la resolución DNS y creación de una nueva zona privada wireless. Se puede usar el demonio Bind9

- **FIREWALL:** Configuración de las reglas de filtrado de paquetes TCP/IP a través del comando `iptables`.

Señalar que no es obligado tener todos estos servicios, pero si altamente recomendable. Tampoco es necesario configurar todos estos servicios en la misma máquina AP.

El servicio que si que debe poseer la máquina AP (punto de acceso) es **FIREWALL**, y son recomendables tener en el AP también **SSH** y **DHCP**.

El hecho de no configurar **SSH**, supondría tener que desplazarse a la propia consola del AP en caso de necesidad de tareas de mantenimiento o cualquier cambio en la configuración.

El servicio **DHCP** evitará el tener que configurar los parámetros de la red cliente a cliente. Y en caso de tener algún cambio en la configuración, se deberá ir cliente a cliente de nuevo. Con **DHCP** se automatiza la tarea.

El servicio **DNS** (implementado con Bind 9) es recomendable y se puede instalar tanto en la misma máquina AP como en otra, siempre y cuando los clientes de la red wireless tengan acceso al servidor de nombres. Este servidor DNS, debería ser al menos la autoridad de la zona privada correspondiente al segmento wireless.

Finalmente también se indicarán los comandos a ejecutar con tal de poner en marcha la funcionalidad de punto de acceso en el sistema.

# 1. SSH

El servicio de SSH (Secure SHell) consiste en una solución fácilmente implementable y altamente segura para conectar remotamente con una máquina *NIX. Es similar a telnet pero con la salvedad de que los datos circulan cifrados de origen a fin, con el correspondiente beneficio en cuanto a la seguridad y confidencialidad de datos obtenida.

La instalación de dicho servicio no requiere de especial atención, tan solo es necesario instalar el correspondiente paquete de la distribución que se esté utilizando. En cuanto a la configuración, cambiaremos algunos parámetros que proporcionaran seguridad extra. Para ello, debemos editar el fichero de configuración, normalmente `/etc/ssh/sshd_config`, y cambiaremos los siguientes parámetros:

```
Protocol 2
```

Tan solo permitimos utilizar el protocolo SSH-2, no dejando opción a la utilización de SSH-1 que está demostrado que no es tan seguro.

```
PermitRootLogin no
```

El usuario `root` es un usuario común para todos los sistemas *NIX a no ser que se modifique dicho nombre de usuario en el sistema, lo cual representa ya el conocimiento de un usuario válido para un posible atacante. A parte de esto, no es aconsejable permitir entrar directamente como root, es aconsejable entrar como un usuario normal y después hacer uso del comando `su` para obtener privilegios de `root`.

```
PasswordAuthentication no
```

Configuraremos SSH para que no permita la validación mediante contraseña, dejando habilitada únicamente la opción de validarse mediante clave publica-privada, de esta forma, a parte de la necesidad de conocer un usuario válido y su correspondiente contraseña, será necesario también el disponer del fichero *llave*, el cual hará la vez de clave privada. Para que esto funcione, deberemos haber introducido previamente en el directorio `~/.ssh/authorized_keys` la clave pública correspondiente a nuestra clave privada, la cual emplearemos posteriormente para validarnos.

# 2. DHCP

El servicio Dynamic Host Configuration Protocol (DHCP) se utiliza para que una vez un cliente se haya unido físicamente a la red wireless se le asignen los parámetros de red necesarios para que pueda interconectarse posteriormente con el resto de clientes e internet.

Dichos parámetros, entre otros, son los siguientes:

- **Dirección IP:** Dirección que se asigna al cliente que esta solicitándolo, normalmente se reparte una dirección que este libre en ese momento dentro de un rango preconfigurado en el servidor, aunque también se puede hacer asignación estática con determinados clientes (o con todos si se desea), haciendo corresponder siempre una misma dirección de red con la misma MAC.

- **Mascara de red:** La máscara de red indica el número de bits de la dirección de red que pertenecen al *NetID* y cuales pertenecen al *HostID*.

- **Dirección de broadcast:** Dirección de broadcast correspondiente a la red a la que se pertenece. La dirección de broadcast es la misma que la de red pero con todos los bits del *HostID* puestos a 1.

- **Nombre del dominio:** Nombre del dominio al que pertenece la red el cliente.

- **Gateways por defecto:** Dirección de red de la máquina la cual nos hará de gateway, permitiéndonos saltar a otras redes y/o internet.

- **Servidores DNS:** Direcciones de red de los servidores de nombres de dominio, dichas máquinas serán las que nos permitirán resolver nombres a direcciones IP

- **Tiempo de caducidad:** El tiempo configurado aquí, será el que indicará la duración de los parámetros asignados. Una vez agotado este tiempo, se deberá volver a pedir una dirección y resto de parámetros.

La instalación de este servicio no tiene mayor complejidad que la de instalar el paquete correspondiente al servidor de DHCP de nuestra distribución. Posteriormente, se deberá editar el fichero de configuración, en el cual se indicarán los parámetros definidos en la lista anterior y los valores que deben tomar.

Normalmente el fichero de configuración se encuentra en `/etc/dhcpd.conf`.

Un ejemplo de fichero de configuración:

```
option domain-name "wireless.inca.ub.es";
option domain-name-servers ap.wireless.inca.ub.es;

option subnet-mask 255.255.255.0;
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.100 192.168.1.200;
  option routers 192.168.1.254;
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.168.1.255;
  default-lease-time 600;
  max-lease-time 7200;
}

host fantasia {
  hardware ethernet 08:00:07:26:c0:a5;
  fixed-address 192.168.1.10;
}
```

Este fichero configura los siguientes parámetros:

- **Red:** 192.168.1.0/24
- **Nombre de dominio:** wireless.inca.ub.es
- **Servidores DNS:** ap.wireless.inca.ub.es
- **Gateway:** 192.168.1.254
- **Tiempo de caducidad de petición:** 10 minutos

Además asignará siempre la misma dirección IP (192.168.1.10) al PC con dirección MAC 08:00:07:26:c0:a5 y al resto una entre 192.168.1.100 y 192.168.1.200.

El AP podría ofrecer servicio DHCP (entre otros) también a la red externa (ethernet), pero por motivos de seguridad, no se hará. El AP solo ofrecerá servicios a la red wireless, siendo un servicio totalmente independiente de la red ethernet. No se quiere que ningún problema del AP con la red wireless pueda afectar a la externa. Por lo que el servicio DHCP solo tiene sentido que este en marcha cuando se está actuando como punto de acceso, con lo que si se detuviera el servicio de red inalámbrica, se debería detener conjuntamente este servicio también.

Finalmente, el servicio se debe iniciar solo para que se ponga a la escucha de las peticiones del interfaz inalámbrico y no en el resto de interfaces que pudiera tener el punto de acceso, para ello, se debe iniciar el servicio con el siguiente comando:

```
/usr/sbin/dhcpd -q wlan0
```

# 3. DNS

El servicio *Domain Name System* (DNS) se utiliza para que una vez un cliente se ha conectado a la red wireless y tenga asignados una dirección y demás parámetros mediante el servicio DHCP, pueda resolver nombres de red a direcciones IP.

Dicho servicio tan solo se ofrecerá a la red wireless (por los mismos motivos expuestos en la sección DHCP anterior), y no a la red cableada a la que también pertenezca el punto de acceso. A parte de resolver los nombres externos como puede ser www.google.com, también se encargará de gestionar una zona interna y privada para que la red wireless tenga su propio dominio. En este caso, el dominio a gestionar es wireless.inca.ub.es, y la zona inversa es la red privada de clase C 192.168.1.0/24.

El modo de funcionamiento es muy sencillo: todas las peticiones que hagan referencia al dominio interno o a la resolución inversa de nuestra red privada se resolverán directamente, y todas las que no sea capaz de resolver, bien sea un nombre fuera de su zona bien porque no la tiene en caché, serán reenviadas a servidores exteriores.

Siendo Bind 9 el servidor de DNS utilizado y habiendo tenido graves problemas de seguridad remotos en el pasado, es importante tomar medidas extra de seguridad, como por ejemplo cambiar la cadena que devolverá ante una petición del tipo:

```
$ dig @servidor chaos txt version.bind
...
;; ANSWER SECTION:
version.bind.           0       CH      TXT     "What are you doing?"
...
```

que normalmente devolvería

```
version.bind.           0       CH      TXT     "9.2.4rc2"
```

y no nos interesa ofrecer información acerca de las versiones que estamos utilizando, ni tan solo que demonio de DNS es el que esta ofreciendo el servicio.

Otra medida importante a tomar es la ejecución del servicio dentro de un entorno chroot, ya que ante un posible fallo remoto, el atacante estaría más limitado o cuanto menos lo tendría más difícil para salir de la jaula. No debemos olvidar tampoco la seguridad extra que nos aportaba en estos entornos chroot los parches GrSecurity. Para ejecutar en un entorno chroot el demonio y como un usuario `bind` y no `root`, deberemos ejecutar como `root` el siguiente comando:

```
# /usr/sbin/named -u bind -t /usr/local/named
```

A continuación se muestra el árbol que se tiene que crear para que `named` (el demonio que en este caso ofrecerá el servicio DNS) pueda trabajar en el entorno chroot:

```
/usr/local/named
|-- dev
|   |-- null
|   |-- random
|   `-- zero
|-- etc
|   |-- bind -> .
|   |-- db.0
|   |-- db.127
|   |-- db.192.168.1
|   |-- db.255
|   |-- db.empty
|   |-- db.local
|   |-- db.root
|   |-- db.wireless
|   |-- localtime
|   |-- named.conf
|   |-- named.conf.local
|   |-- named.conf.options
|   |-- resolv.conf
|   |-- rndc.key
|   `-- zones.rfc1918
|-- logs
|   `-- named.log
`-- var
    `-- run
        `-- named.pid
```

El contenido del directorio `/etc` depende de la configuración que se tenga, así como también la ubicación de los directorios `/logs` y `/var` y lo que ellos contienen. Lo que si que tiene que estar exactamente igual es el directorio `/dev`.

Unos ficheros de configuración podrían ser los que siguen:

- **`named.conf`**

```
include "/etc/bind/named.conf.options";

zone "." {
        type hint;
        file "/etc/bind/db.root";
};

zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};
```

```
zone "wireless.inca.ub.es" {
        type master;
        file "/etc/bind/db.wireless";
};

zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.192.168.1";
};
```

- **named.conf.options**

```
options {

        forwarders {
                172.17.1.1;
                172.17.1.2;
        };

        auth-nxdomain no;    # conform to RFC1035

        directory "/";
        pid-file "/var/run/named.pid";
        statistics-file "/logs/named.stats";
        listen-on port 53 { 192.168.1.254; 127.0.0.1; };
        query-source address 172.17.1.254 port 53;
        version "What are you doing?";

};

logging {

        channel channel_log
        {
                file "/logs/named.log";
                severity debug;
        };

        category default { channel_log; };
};
```

- **db.wireless**

```
$TTL    604800
@       IN      SOA     wireless.inca.ub.es. root.wireless.inca.ub.es.
(
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      wireless.inca.ub.es.
@       IN      A       192.168.1.254
ap      IN      A       192.168.1.254
pc001   IN      A       192.168.1.1
pc002   IN      A       192.168.1.2
pc003   IN      A       192.168.1.3
...
```

- **db.192.168.1**

```
$TTL    604800
@       IN      SOA     wireless.inca.ub.es. root.wireless.inca.ub.es.
(
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      wireless.inca.ub.es.
254     IN      PTR     ap.wireless.inca.ub.es.
;
1       IN      PTR     pc001.wireless.inca.ub.es.
2       IN      PTR     pc002.wireless.inca.ub.es.
3       IN      PTR     pc003.wireless.inca.ub.es.
...
```

Estos son los ficheros de configuración básicos que han sufrido algún cambio partiendo de la instalación por defecto del paquete de Bind9. Estos dos últimos definen la zona interna wireless y la resolución inversa de IP para la red privada wireless.

# 4. FIREWALL

El punto de acceso que se está configurando, ofrece varios servicios y tiene que permitir a los clientes autorizados comunicarse con el exterior y no únicamente con la red wireless. Para ello, y para filtrar conexiones no deseadas o inválidas, se deberán configurar unas reglas básicas en el firewall del sistema.

A continuación una lista de servicios que ofrecerá el punto de acceso:

- **DNS:** Servicio que se debe ofrecer SOLO a los clientes de la red wireless.

- **SSH:** Servicio al que únicamente debe ser capaz de acceder el/los administradores de sistemas desde una o varias IPs de administración.

- **DHCP:** Únicamente se deberá atender a las peticiones que lleguen por el interfaz wireless, ya que es la red que gestiona el punto de acceso.

La política de seguridad será denegar todo excepto lo explícitamente permitido, que serán aquellos servicios que ofrecerán así como las posibles conexiones salientes que se necesiten establecer.

Las reglas de permisividad en cuanto al acceso al exterior dependerán de la solución de seguridad que se adopte. En caso de tener un control de seguridad a nivel físico-enlace, se permitirá la salida a todas las máquinas que formen parte de la red, ya que ya habrán pasado por la validación y son usuarios válidos. En el caso de que el control de seguridad adoptado sea a nivel de red (IP), no se permitirá el acceso al exterior a través de los interfaces de red normales (`wlan0`), sino que tan solo se permitirá la

comunicación hacia el exterior a aquellas máquinas que hayan establecido una VPN con el punto de acceso, con lo que ya existirá un control de acceso previo para todo el tráfico que circule por el interfaz virtual `ipsec0`, que será el que hace referencia a las conexiones de las VPN.

La política por defecto de denegar todo requiere de los siguientes comandos:

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

A continuación se deberá dar acceso a cada uno de los servicios. En el caso de servicios que utilicen el protocolo TCP, tan solo se debe especificar la permisividad de la entrada de datos, ya que para la salida se utilizará una regla genérica que permitirá los paquetes que sean de salida y tengan un estado de RELATED o ESTABLISHED, esto es gracias a que la implementación del firewall de Linux es *stateful*, lo que quiere decir que es capaz de reconocer el estado de un paquete y saber si es de una conexión nueva, establecida, si se está intentando establecer en ese momento,… y en consecuencia, filtrar en función de su estado.

En cambio, para las conexiones UDP, se deberá indicar explícitamente los paquetes que pueden entrar y los que pueden salir de la máquina.
Para permitir la conexión al servicio SSH desde la dirección IP de administración 10.0.0.1 (la cual es una IP ajena a nuestras dos redes de área local) se ejecutará la siguiente instrucción (presuponemos que la IP externa del AP es 172.17.1.254):

```
iptables -A INPUT –s 10.0.0.1 -d 172.17.1.254 -p tcp -m tcp \
--dport 22 -j ACCEPT
```

Para cada dirección IP que se quiera dar acceso a SSH, se deberá introducir una regla como esta pero cambiando la dirección 10.0.0.1 por la deseada.

Para cada servicio que se quiera permitir el acceso, se tiene que ejecutar una instrucción similar a la de SSH, obviamente, se deberá cambiar la información del puerto al que se permite conectar por el correspondiente al servicio a dar acceso y las direcciones permitidas (también se pueden introducir direcciones de red o dar acceso a todas las direcciones).

Para que el tráfico TCP relacionado con alguna conexión pueda entrar, se requiere de la siguiente línea:

```
iptables -A INPUT -d 172.17.1.254 -i eth0 -m state \
--state RELATED,ESTABLISHED -j ACCEPT

iptables -A INPUT -d 192.168.1.254 -i eth0 -m state \
--state RELATED,ESTABLISHED -j ACCEPT
```

Para los servicios UDP, como por ejemplo DHCP, se deberán ejecutar líneas como las que siguen:

```
iptables -A INPUT -p udp -m udp --dport 67 -j ACCEPT
```

```
iptables -A OUTPUT -s 192.168.1.254 -p udp -m udp --sport 67 -j ACCEPT
```

Por último tan solo queda indicar como se deberán gestionar las conexiones que deban ser reenviadas de la red interna (wireless) hacia la externa (internet). Se deberán enmascarar las conexiones relacionadas con los clientes wireless cambiando la dirección origen de los paquetes por la dirección pública del AP, además se deberá "recordar" a que cliente corresponde cada conexión, ya que cuando el tráfico venga de vuelta, se debe ser capaz de volver a cambiar la dirección destino por la del cliente correspondiente y reenviar dicho tráfico a su destino.

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j SNAT \
--to-source 172.17.1.254
```

Con esta instrucción estamos configurando al sistema para que haga la conversión de direcciones, concretamente estamos diciendo que introduzca en la cadena `POSTROUTING` (deberá hacer lo que se le indique DESPUÉS de haber enrutado el tráfico) de la tabla `(-t) nat` (*Network address translation*) la orden de que lo que venga con origen 192.168.1.x (dirección de red perteneciente a la red wireless) y tenga como salida el interfaz `eth0` (es el correspondiente al interfaz externo) que le cambie la dirección origen por 172.17.1.254 (la dirección externa del AP).

A parte de esta conversión de direcciones, también se debe permitir el reenvío (*forward*) de los paquetes que vayan de la red interna a la externa y permitir las respuestas a dicho tráfico saliente. No se permitirá iniciar conexiones desde el exterior hacia la red wireless.

```
iptables -A FORWARD -s 192.168.1.0/24 -i wlan0 -o eth0 -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Se puede encontrar un ejemplo de lo que podría ser el script de configuración del firewall en la sección apéndices.

# 5. Punto de Acceso Básico

En este apartado se verá como configurar el sistema para que se comporte como un punto de acceso básico, sin ningún tipo de cifrado ni control de usuarios, únicamente poner en funcionamiento el modo Master.

El comando para cambiar el modo de funcionamiento de la tarjeta es:

```
iwconfig wlan0 mode master
```

Con esto se pone la tarjeta en modo Master, que es el modo en que trabajan los puntos de acceso. A continuación, se deberá poner un identificador de red (ESSID):

```
iwconfig wlan0 essid inca
```

En este ejemplo, se ha puesto el identificador de red `inca` a la red wireless. Ahora cualquier cliente wireless compatible con el estándar 802.11b (recordemos que el 802.11g es compatible con 802.11b) será capaz de enlazarse a la red wireless si configura el essid `inca`.

Únicamente queda configurar los parámetros de la tarjeta de red wireless:

```
ifconfig wlan0 192.168.1.254 netmask 255.255.255.0 \
broadcast 192.168.1.255
```

Esta solución tiene problemas de seguridad; mejor dicho, carece de seguridad y no se ha configurado ningún método de cifrado ni validación para unirse a la red, cosa que no es nada aconsejable: cualquier persona con un dispositivo compatible con nuestra red será capaz de conectarse a ella y utilizar sus recursos. Y no solo eso, sino que también existe la posibilidad de que escuche todas las comunicaciones que circulen por ella e incluso modificar los datos que se transmiten. Esto es posible debido a que el AP transmite periódicamente su SSID, los clientes conocen la existencia de la red wireless y su SSID, y pueden unirse a la red.

En otro modo de funcionamiento, se puede configurar el AP para que no envie las tramas administrativas que contienen el SSID, por tanto, la estación inalámbrica que desee asociarse con el AP debe tener configurado el mismo SSID que el AP, en caso contrario no habrá asociación del cliente con el AP.

En contra de lo que pudiera parecer y debido a que las tramas de administración en las WLAN's 802.11 son siempre enviadas de forma abierta, este modo de operación no provee ninguna seguridad. Un atacante puede interceptar las tramas administrativas que envíen los clientes al AP y así descubrir el SSID.

Por ello, y para hacer una primera introducción a la seguridad wireless, se introducirá un elemento nuevo: cifrado WEP.

# 6. Punto de Acceso WEP

El estándar 802.11 para comunicaciones LAN inalámbricas introdujo el protocolo de privacidad equivalente al cableado (WEP) para evitar la intercepción y enmascaramiento de los datos: estos circulan por el aire y es algo trivial de realizar para cualquiera en disposición de un equipo de radio.

El objetivo principal de WEP es proteger la confidencialidad de los datos de los usuarios de escuchas indeseadas. WEP es parte de un estándar internacional; ha sido integrado por los fabricantes en sus dispositivos 802.11 y hoy día su uso es común.

EL protocolo WEP se utiliza para proteger los datos a nivel de enlace de las transmisiones inalámbricas. Es descrito con detalle en el estándar 802.11.

## 6.1. Finalidad

El protocolo WEP fue diseñado para imponer tres metas de seguridad principales:

- **Confidencialidad:** El objetivo fundamental de WEP es evitar escuchas fortuitas.

- **Control de acceso:** Un segundo objetivo del protocolo es proteger el acceso a la infraestructura de red inalámbrica. El estándar 802.11 incluye una característica opcional para desechar aquellos paquetes que no estén apropiadamente cifrados usando WEP, a la vez que los fabricantes promocionan la habilidad de WEP para proveer control de acceso.

- **Integridad de datos:** Un objetivo relacionado es prevenir la manipulación de los mensajes transmitidos; el campo *checksum* se incluye con este propósito.

En los tres casos, la afirmación de la seguridad del protocolo "reside en la dificultad de obtener la clave pública por medio de ataques por fuerza bruta".

## 6.2. Funcionamiento

WEP confía en una clave secreta `k` compartida entre los partícipes del grupo de comunicaciones para proteger el cuerpo del mensaje de una trama de datos. El cifrado de una trama se efectúa según sigue:

**Suma de comprobación (*checksum*):** Primero, se calcula una suma de integridad `c(M)` sobre el mensaje *M*. Concatenamos ambos para obtener un texto sin cifrar `P=(M, c(M))`, el cual será empleado como entrada para la segunda etapa. Nótese que `c(M)`, y por tanto `P`, no depende de la clave `k`.

**Cifrado:** En la segunda etapa se cifra el texto plano *P* con el algoritmo RC4. Elegimos `v` un vector de inicialización (IV). El algoritmo RC4 genera un *keystream* –una secuencia larga de bytes pseudo aleatorios- como una función de `v` y la clave `k`. Este *keystream* viene referido como `RC4(v, k)`. Posteriormente se realiza un or-exclusivo (XOR, denotado por `^`) sobre el texto plano con el *keystream* obtenido para conseguir el texto cifrado:

```
C = P ^ RC4(v, k)
```

**Transmisión:** Finalmente, se transmite `v` y el texto cifrado a través del enlace de radio.

Simbólicamente, este proceso puede ser representado como:

`A -> B: v,(P ^ RC4(v, k));` donde `P = (M, c(M))`

A partir de ahora se usará el término *mensaje* (simbólicamente, `M`) para hacer referencia a la trama inicial de datos que se desea procesar, el término *texto plano* (`P`) para hacer referencia a la concatenación de mensaje y *checksum* como es enviado al algoritmo de

cifrado RC4, y el término *texto cifrado* (C) para referirse al texto cifrado tal y como se transmite por el enlace de radio.

Para descifrar una trama protegida por WEP, el receptor invierte el proceso de cifrado. Primero, genera el *keystream* RC4(v, k) y efectúa un or-exclusivo contra el texto cifrado para recuperar el texto plano original:

```
P' = C ^ RC4(v, k)
= (P ^ RC4(v, k)) ^ RC4(v, k)
= P
```

Después, el receptor verificará el *checksum* del texto descifrado P' separándolo en la forma (M', c'), recalculando el *checksum* c(M'), y comprobando que coincide con el *checksum* recibido c'. Esto asegura que sólo las tramas con un *checksum* válido son aceptadas por el receptor.

## 6.3. Configuración

Después de esta descripción sobre el funcionamiento del protocolo WEP, veamos como configurarlo.

Partiendo siempre de la configuración anterior, en la que ya habíamos configurado el modo de funcionamiento (master), el SSID (inca) y la dirección del AP (con ifconfig), bastará con indicar la clave WEP que se quiere utilizar, tanto por la parte del cliente como por la parte del AP. Se hace con el siguiente comando:

```
# Clave en hexadecimal
iwconfig wlan0 key 4b289f84eadb348
# Clave en ASCII
iwconfig wlan0 key s:trekker
```

Si se introduce una clave en ASCII se reduce considerablemente el número de posibles claves WEP, por ese motivo, lo ideal es poner el valor de una clave aleatoria en hexadecimal, sin ningún sentido y lo más larga posible.

## 6.4. Vulnerabilidad

Hoy día hay dos clases de implementación WEP: la clásica, tal cual viene estipulada en el estándar, y una versión extendida desarrollada por algunos fabricantes para proveer claves más largas.

El estándar WEP especifica el uso de claves de 40 bits. Esta longitud de clave es lo suficientemente corta como para realizar ataques prácticos de fuerza bruta a individuos y organizaciones, con recursos de cálculo bastante modestos. No obstante, es trivial extender el protocolo para usar claves más largas, y de hecho algunos fabricantes ofrecen en sus productos versiones de 128 bits (que realmente emplean 104 bits, a pesar de su engañoso nombre). Esta extensión convierte en imposibles los ataques por fuerza bruta incluso para el adversario con los mayores recursos, dada la tecnología actual. A

pesar de ello, se verá que hay atajos que permiten evitar el uso de ataques por fuerza bruta a la clave para descubrir la misma, lo que convierte en inseguras incluso a las implementaciones WEP de 128.

Se puede encontrar un completo análisis sobre el modo de funcionamiento y el análisis de sus fallos en un apéndice, aquí simplemente se comentarán datos prácticos acerca de lo necesario para vulnerar la seguridad de una clave WEP.

Requisitos para conseguir romper la seguridad WEP:

- Ordenador con tarjeta de red wireless capaz de funcionar en modo *Monitor*
- Sniffer de red (Kismet, AirSnort,…)
- Crackeador WEP (AirSnort, WEPCrack,…)

Como se observa, son pocos y asequibles los requisitos necesarios para ponerse manos a la obra. Por poner un ejemplo, se citará un texto de la página principal de AirSnort:

“AirSnort requiere aproximadamente la captura de 5-10 millones de paquetes cifrados. Una vez obtenidos, AirSnort puede descifrar la contraseña del cifrado en segundos.”

A continuación una pregunta que se encuentra en la FAQ del paquete AirSnort:

“¿Sobre cuanto tiempo se tardaría en conseguir la contraseña con AirSnort?”

Y la respuesta a la pregunta:

“Para encontrar una contraseña WEP, AirSnort necesita cierto número de paquetes con claves débiles. Utilizando cifrado de 128 bits se pueden generar cerca de dieciséis millones de claves WEP, aproximadamente nueve mil son débiles. Los paquetes interesantes son los de clave débil.

La mayoría de las contraseñas WEP se pueden sacar con cerca de dos mil paquetes interesantes. Algunas con tan solo 1200-1500 paquetes y otras con 3500-4000.

Para hacerse una idea, pongamos que un negocio (no demasiado grande) tenga cuatro empleados utilizando la misma contraseña. Estos empleados consultan páginas web continuamente durante todo el día (no son empleados muy buenos), con lo que generarán alrededor de un millón de paquetes al día. Del millón de paquetes, aproximadamente ciento veinte paquetes serán interesantes, así que después de unos dieciséis días la red estará prácticamente crackeada.

En cualquier caso, la red expuesta no tiene excesivo tráfico, con lo que las redes que tengan tráfico abundante durante todo el día pueden ser crackeadas en un solo día.”

A parte de esta importante vulnerabilidad, WEP tiene otro problema, y es que independientemente de que se pueda o no crackear la contraseña puesta, cualquier usuario válido en la red (es decir, que disponga de la clave WEP), podrá escuchar, interceptar y modificar todo el tráfico que circule por la red wireless ya que siempre se utiliza la misma clave para todo y para todos.

Como podemos observar con estos casos prácticos, no podemos confiar en la seguridad que nos ofrece WEP.

# 7. Punto de Acceso WPA

WPA es un borrador del protocolo 802.11i (concretamente la versión 3.0 del borrador), que se empezó a diseñar a principios del 2003 y se prevé que este finalizado durante el 2004, se prevé que en septiembre del 2004 se empiecen a certificar productos con este estándar. WPA ha sido diseñado para ser compatible con 802.11i cuando sea estándar, evitando así posibles problemas de compatibilidad que pudieran surgir.

El 802.11i será el próximo protocolo estándar de seguridad para redes Wi-Fi 802.11.

Wi-Fi Protected Access (WPA) de la alianza Wi-Fi y la IEEE ha sido diseñado para sustituir el anterior protocolo de seguridad WEP de forma temporal y hasta que salga a la luz el nuevo 802.11i, ya que WEP, como se acaba de ver es muy vulnerable.

No se requiere un cambio de hardware para utilizar WPA, con una actualización de firmware es suficiente.

Una vez 802.11i este finalizado, tendremos disponible el nuevo WPA2, que cumplirá dicho estándar.

## 7.1. Funcionamiento general

WPA utiliza RC4 para el cifrado, como WEP, pero con una clave diferente para cada paquete (TKIP). Además, implementa un mecanismo de protección para la autenticación por paquete, evitando así la interceptación, alteración y reenvío de los paquetes que circulan por la red (MIC).

### Privacidad WPA: TKIP

El problema con la privacidad de WEP es debido al envío por el aire del vector de inicialización (IV), sin cifrado alguno. Cuando se tiene una red wireless muy ocupada, el envío del IV se repite de vez en cuando cada ciertas horas. Al capturar varios paquetes que contienen el mismo IV, los intrusos pueden averiguar la clave WEP al repetir operaciones XOR y lograr acceso ilegal a la red.

Para el estándar 802.11, el cifrado de privacidad equivalente con cable (WEP) es opcional. Para WPA, se requiere el cifrado con el protocolo TKIP. El protocolo TKIP sustituye a WEP con un algoritmo de cifrado nuevo más seguro que, sin embargo, utiliza las utilidades de cálculo de los dispositivos inalámbricos existentes para realizar las operaciones de cifrado.

TKIP, *Temporal Key Integrity Protocol*, es una modificación de WEP, del que se duplica la longitud del vector de inicialización (de 24 a 48 bits) para evitar la repetición de un mismo valor, y un método de renovación automática de la clave de encriptación entre los dispositivos wireless.

#### Integridad del Mensaje WPA: MIC

*Message Integrity Check* o comprobación de integridad del mensaje ha sido diseñado para prevenir que intrusos capturen paquetes, los alteren y los reenvíen. La función MIC, la cual se conoce como "Michael", es un *hash* criptográfico de un solo sentido, que reemplaza el checksum CRC-32 utilizado en WEP.

#### Conclusión

WPA soluciona todos los problemas de WEP conocidos en autenticación, privacidad de datos y problemas de integridad. Es un mecanismo de seguridad WLAN razonable, útil para poner una red inalámbrica segura en producción.

## 7.2. Modos de funcionamiento

WPA, tiene dos modos de funcionamiento: WPA-PSK y WPA-EAP también denominados *WPA-Personal* y *WPA-Enterprise*. En WPA-PSK se preconfigura una clave “maestra” la cual dará acceso a la red, mientras que en WPA-EAP se configuran usuarios y una vez el usuario se ha validado, el punto de acceso y el cliente se intercambian la clave de cifrado inicial.

En la siguiente subsección nos centraremos en como configurar WPA-PSK. El modo WPA-EAP se verá más adelante en configuraciones más avanzadas.

## 7.3. Configuración WPA-PSK

El funcionamiento de WPA-PSK es parecido a WEP, únicamente se necesita configurar una clave que será compartida por todos los usuarios de la red wireless y servirá de semilla para crear una clave en constante rotación, de forma que cada paquete de información lleva una clave completamente diferente a los anteriores.

Para utilizar WPA, se deberá disponer del paquete hostapd en el AP y tener actualizada la tarjeta con la versión de firmware 1.7.0 o superior.

#### Actualización de firmware

Como se acaba de ver, para utilizar WPA con tarjetas 802.11b es necesaria una actualización del firmware. Como este proyecto se basa en tarjetas con chipset Intersil prism2/2.5/3, en este apartado se indicará como se puede conseguir el firmware necesario para una tarjeta y como actualizarla.

Hay dos tipos de actualizaciones, una que se realiza en la memoria flash de la tarjeta y otra en la RAM. La primera es permanente y tiene el inconveniente que si se actualiza con una versión incorrecta se puede dejar la tarjeta inutilizable. La segunda en RAM, tiene el problema que cada vez que se inicializa la tarjeta se debe actualizar primero el firmware, ya que la memoria RAM es volátil y se pierden las actualizaciones. Aquí se explicará como actualizar en la memoria flash, ya que después de haberse realizado muchas pruebas, no ha sido posible conseguir una actualización en RAM que fuera funcional, todas han dado problemas.

Antes de actualizar, se deberá realizar en el driver hostap una pequeña modificación. Por defecto viene sin la posibilidad de actualizar las tarjetas en flash, por razones de seguridad. Se debe descomentar la línea número 62 del fichero `driver/modules/hostap_config.h` (puede variar algo según la versión)

```
/* #define PRISM2_NON_VOLATILE_DOWNLOAD */
```

que debe quedar

```
#define PRISM2_NON_VOLATILE_DOWNLOAD
```

Una vez hecho esto, se recompila el driver y se sustituye por el que actualmente esté cargado.

A continuación se debe extraer la información de nuestra tarjeta para saber qué firmware será el apropiado:

```
# hostap_diag wlan0
Host AP driver diagnostics information for 'wlan0'

NICID: id=0x8002 v1.0.0 (HWB3163-01,02,03,04 Rev A)
PRIID: id=0x0015 v0.3.0
STAID: id=0x001f v1.4.9 (station firmware)
```

La información importante es la *(station firmware)*. Como se observa, la tarjeta del ejemplo tiene una versión 1.4.9 que no sirve para utilizar WPA.

El nombre de los ficheros de firmware es de la forma:

<Tipo><Plataforma><Versión Mayor / Menor><Variante_Versión>.HEX

**Tipo:** puede ser I (Inicial), P (Primario), S (Secundario), or T (Terciario). Además, puede ser también R (Primario para RAM) o A (Secundario para RAM). Utilizar únicamente ficheros que empiecen por P o S (o R y A si es RAM). De lo contrario se pueden echar a perder las tarjetas.

**Plataforma:** es un carácter identificador de plataforma que hace referencia al NICID de la tarjeta. Véase la tabla de identificadores expuesta a continuación.

**Versión Mayor / Menor:** son dos caracteres de versión mayor y dos de version menor.

**Variante_Versión:** son dos caracteres de la variante de la versión.

| Component ID | Interface | MAC | NVRAM | RAM Config | Release Code ID | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Initial | Primary | Secondary | Tertiary |
| 8000 EVB/PI | PCMCIA | 3841 | Atmel | x8 | 0 (G) | 0 (G) | 0 (G) | 0 (G) |
| 8003 | PCMCIA | 3841 | AMD | x8 | 1 (H) | 1 (H) | 1 (H) | 1 (H) |
| 8009 EVB/PII | PCMCIA | 3841 | Atmel | x8 | 1 (G) | J | J | J |
| 8008 | PCMCIA | 3841 | | x8 | 4 | 4 | 1 (H) | 1 (H) |
| 800A EVB3842 (see note 1) | PCMCIA/USB | 3842 | | x8 | A | F, U, X, Y | F, U, X, Y | F, U, X, Y |
| 800B, 8012, 8016, 801A | PCMCIA & PCI | 3842 | AMD | x8 | A | F | F | F |
| | | | | x16 | B | | | |
| 800C, 8013, 8017, 801B | PCMCIA & PCI | 3842 | SST | x8 | C | K | | |
| | | | | x16 | D | | | |
| 800D, 8014, 8018, 801C | PCMCIA & PCI | 3842 | Large Serial | x8 | E | L | | |
| | | | | x16 | | | | |
| 800E, 8015, 8019, 801D | PCMCIA | 3842 | Large Serial | x8 | N/A | M | M | M |
| 800F, 801E | USB | 3842 | AMD | x8 | A | U | U | N/A |
| | | | | x16 | B | | | |
| 8010, 801F | USB | 3842 | SST | x8 | C | O | | |
| | | | | x16 | D | | | |
| 8011, 8020 | USB | 3842 | Large Serial | x8 | E | P | | |
| | | | | x16 | F | | | |

Por ejemplo:

- PK010004.HEX
  Firmware primario v1.0.4 para NICID 800C, 8013, 8017, 801B.

- SF010409.HEX
  Station firmware v1.4.9 for NICID 800B, 800C, 800D, 8012, 8013, 8014, 8016, 8017, 8018, 801A, 801B, 801C.

- S1010409.HEX
  Station firmware v1.4.9 for NICID 8003, 8008.

Se dan a continuación unas direcciones donde se pueden encontrar firmwares para las tarjetas con chipset Prism2/2.5/3 de Intersil:

- http://linux.junsun.net/intersil-prism/firmware
- http://www.red-bean.com/~proski/firmware
- http://www.netgate.com/support/prism_firmware
- http://www.sydneywireless.com/hostap/
- http://www.demarctech.com/products/reliawave-rwz/reliawave-rwz-200mw-prism2-5-pcmcia-card.html

Una vez se ha identificado el firmware necesario para nuestra tarjeta y se ha descargado, se procederá con la actualización de la tarjeta. En el caso del ejemplo que se está siguiendo, se tiene un NICID 8002, que es equivalente al 8003, consultando la tabla, se ha encontrado un fichero S1010701.HEX.

Antes de intentar actualizar el firmware, comprobaremos la salida del comando `prism2_srec`, esta disponible en el paquete hostap-utils.

```
# prism2_srec wlan0 S1010701.HEX
srec summary for S1010701.HEX
Component: 0x001f 1.7.1 (station firmware)


Verifying update compatibility and combining data:
OK.
```

Si obtenemos una respuesta positiva, en un principio no se tendrá ningún problema para actualizar la tarjeta, con lo que se procede con la actualización en la memoria flash (se añade el modificador –f a la ejecución del comando):

```
# prism2_srec -f wlan0 S1010409.HEX
srec summary for S1010701.HEX
Component: 0x001f 1.7.1 (station firmware)


Verifying update compatibility and combining data:
OK.

Downloading to non-volatile memory (flash).
Note! This can take about 30 seconds. Do _not_ remove card during
download.
OK.
Components after download:
  NICID: 0x8002 v1.0.0
  PRIID: 0x0015 v0.3.0
  STAID: 0x001f v1.7.1
```

En estos momentos ya se tiene en la tarjeta el firmware necesario.

Hostapd

Hostapd es un paquete opcional para el driver hostap. Añade más características a la gestión básica de IEEE 802.11 que incluye el driver hostap. Puede utilizar un servidor RADIUS externo para la validación, dar o denegar acceso basándose en la dirección MAC del cliente, actúa como un autenticador IEEE 802.1X, permite gestión dinámica de claves WEP, WPA y WPA2 (IEEE 802.11i/RSN).

En este caso nos será necesario para configurar WPA. Descargaremos el paquete de hostapd de su página (http://hostap.epitest.fi), lo descomprimiremos y compilaremos.

```
# wget http://hostap.epitest.fi/releases/hostapd-0.2.3.tar.gz
--15:08:30--  http://hostap.epitest.fi/releases/hostapd-0.2.3.tar.gz
           => `hostapd-0.2.3.tar.gz'
Resolving hostap.epitest.fi... 194.100.116.233
Connecting to hostap.epitest.fi[194.100.116.233]:80... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 138,900 [application/x-gzip]

100%[======================================>] 138,900      100.99K/s

15:08:32 (100.86 KB/s) - `hostapd-0.2.3.tar.gz' saved [138900/138900]

# tar zxvf hostapd-0.2.3.tar.gz
hostapd-0.2.3/
hostapd-0.2.3/.cvsignore
hostapd-0.2.3/ChangeLog
...
hostapd-0.2.3/hostap_common.h
hostapd-0.2.3/wireless_copy.h
# cd hostapd-0.2.3
# make
gcc -MMD -O2 -Wall -g -DHOSTAPD_DUMP_STATE -I../driver/modules -
I../utils   -c -o hostapd.o hostapd.c
...
gcc -o hostapd hostapd.o eloop.o ieee802_1x.o eapol_sm.o radius.o
md5.o rc4.o common.o iapp.o ieee802_11.o config.o ieee802_11_auth.o
accounting.o sta_info.o driver.o receive.o radius_client.o sha1.o
wpa.o aes_wrap.o
# ls -l hostapd
-rwxr-xr-x    1 root     root       571929 2004-06-26 15:10 hostapd
```

Creamos los ficheros de configuración para hostapd:

```
ap:/etc# ls -l hostapd.*
-rw-r--r-- 1 root    root          231 2004-05-26 08:10 hostapd.accept
-rw-r--r-- 1 root    root         8104 2004-06-15 21:17 hostapd.conf
-rw-r--r-- 1 root    root          147 2004-05-24 13:12 hostapd.deny
```

El fichero `hostapd.accept` contendrá la lista de direcciones MAC a las que se permite que se autentiquen al AP, en `hostapd.deny` de las que no se permite autenticarse. En `hostapd.conf` la configuración del AP.

Para utilizar WPA-PSK es necesario tener las siguientes opciones:

```
# Interfaz wireless a gestionar
interface=wlan0
# Se comportara como un demonio poniéndose en background
daemonize=1
# SSID de la red wireless
ssid=inca
# Se permite acceso a todas las MAC
macaddr_acl=0
# ACL MACs permitidas en caso de macaddr_acl=1
accept_mac_file=/etc/hostapd.accept
# ACL MACs denegadas en caso de macaddr_acl=0
deny_mac_file=/etc/hostapd.deny
# Configuramos el AP en modo OSA, necesario para WPA
auth_algs=1
# Activamos WPA
wpa=1
# Modo de WPA, en este caso WPA-PSK
wpa_key_mgmt=WPA-PSK
# Contraseña compartida para el acceso a la red
```

```
wpa_passphrase=InCaInCa
# Algoritmos de cifrado permitidos, CCMP se utilizara en WPA2
wpa_pairwise=CCMP TKIP
```

Una vez configurado todo, tan solo queda ejecutar el demonio hostapd:

```
hostapd -B /etc/hostapd.conf
```

Como de costumbre, a parte de configurar el modo AP, deberemos configurar la dirección IP del AP perteneciente a la red inalámbrica:

```
ifconfig wlan0 192.168.1.254 netmask 255.255.255.0 broadcast 192.168.1.255
```

Con la tarjeta wireless introducida y estos dos comandos, ya se dispone de un AP con WPA-PSK.

## 7.4. Problemas WPA-PSK

### Genéricos

WPA-PSK no podrá funcionar en modo Ad-Hoc. Más que un problema concreto de WPA-PSK es un problema genérico referente a WPA.

No proporciona control de acceso basado en usuarios, sino en dispositivos. Es decir, serán los dispositivos los que estén configurados para acceder a la red, después, cualquier persona que utilice ese dispositivo tendrá acceso a la red wireless (por dispositivo hacemos referencia al PC o cualquier otro terminal de red).

WPA-PSK utiliza siempre una misma clave para dar acceso a los dispositivos (a no ser que se cambie periódicamente a mano y se disponga de un método seguro para redistribuir la nueva clave), con lo que si queremos denegar el acceso a la red a uno de los usuarios no nos será posible, la única forma sería cambiando la clave a mano y volviéndola a distribuir entre los usuarios legítimos.

La clave de acceso es un secreto compartido entre todos los usuarios legítimos de la red, con lo que cuanto mayor sea el número de usuarios, mayor será la dificultad de mantener dicha clave en secreto.

### Criptográficos

Un estudio realizado por Robert Moskowitz, director de ICSA Labs, indica que el sistema utilizado por WPA para el intercambio de la información necesaria para la generación de las claves de cifrado es muy débil.

Según este estudio, WPA en determinadas circunstancias es incluso más inseguro que WEP. Cuando las claves preestablecidas utilizadas en WPA utilizan palabras presentes en el diccionario y la longitud es inferior a los 20 caracteres, el atacante sólo necesitará interceptar el tráfico inicial de intercambio de claves. Sobre este tráfico, realizando un

ataque de diccionario, el atacante puede obtener la clave preestablecida, que es la información necesaria para obtener acceso a la red.

No es un problema nuevo, pues fue apuntado durante la verificación inicial del protocolo. Es solo una muestra de que una implementación inadecuada puede afectar negativamente cualquier sistema de cifrado.

El problema solo es explotable bajo una serie de circunstancias muy concretas. Este problema puntual no es, en absoluto, una indicación de la debilidad de WPA. Únicamente es un recordatorio de la necesidad de utilizar claves convenientemente largas y que incluyan caracteres especiales.

### Conclusión

WPA-PSK no es apropiado cuando el objetivo es tener un control de acceso por usuarios, en cambio es ideal si el ámbito de la red es pequeño, como un hogar o pequeña empresa. Aún y así, es conveniente utilizar claves lo suficientemente largas y sin sentido (que no se puedan encontrar en un diccionario) para garantizar dicha seguridad.

Teniendo en cuenta estos puntos, WPA-PSK se puede considerar seguro.

# CONFIGURACIÓN AVANZADA DEL AP

En esta sección se intentará arreglar los problemas que se tenían con las soluciones anteriores. Como se ha visto hasta este punto, la solución más segura sería una configuración WPA-PSK, siempre y cuando se configuraran claves lo suficientemente seguras. Pero uno de los problemas de esa configuración es que toda la seguridad se basa en una única clave para todos los usuarios, cosa que no conviene en un entorno empresarial.

Para solventar dicho inconveniente, se pasará a configuraciones más avanzadas, para las que se requerirá un elemento extra de configuración, un servidor RADIUS.

Los elementos que entran en juego en esta configuración son los mismos que los definidos en la configuración básica añadiendo los siguientes dos servicios:

- **HOSTAPD:** Demonio que se encargará de gestionar toda la funcionalidad y configuración del punto de acceso. Introducido previamente en el apartado WPA-PSK, ahora se verá el resto de sus posibilidades de dicho demonio.

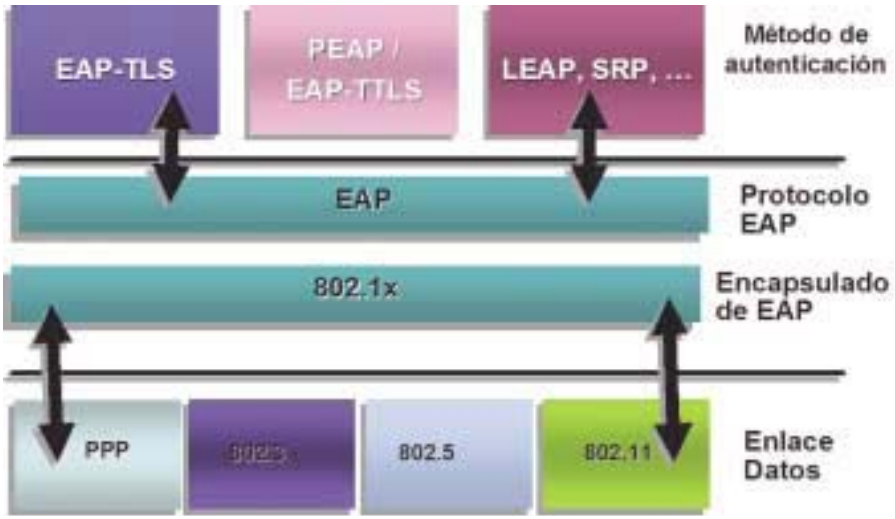
- **RADIUS:** Servicio encargado de la validación de usuarios.

El servicio de hostapd es necesario que se encuentre en la misma maquina, ya que es el que proporciona la funcionalidad de punto de acceso y tiene que trabajar conjuntamente con el driver. En cambio, el servicio RADIUS puede estar tanto en la misma como en cualquier otra máquina, siempre y cuando este accesible para el AP.

## 1. Hostapd

Como ya se ha dicho en el apartado WPA-PSK, hostapd proporciona herramientas adicionales para mejorar la seguridad de una red wireless.

En este caso se hará uso del estándar 802.1x, que define un protocolo para encapsular protocolos de autenticación sobre protocolos de la capa de enlace de datos. IEEE 802.1x permite utilizar diversos métodos para autenticar al usuario a través del protocolo de autenticación extensible (EAP). IEEE 802.1x es una ampliación de la capa de enlace de datos:




De esta forma se podrá autenticar a los usuarios en lugar de a los dispositivos.

IEEE 802.1x define tres entidades:

- **Supplicant:** es el solicitante, es el componente de la estación de red que realiza la autenticación con el servidor de autenticación.

- **Authenticator:** es el autenticador, el AP realiza dicha tarea y se encarga de pasar los paquetes entre el *supplicant* y el *Authentication server*. Además tiene un PAE con funcionalidad de autenticador para controlar la autorización virtual del puerto. De esta forma sabrá si aceptar o no paquetes de una estación determinada.

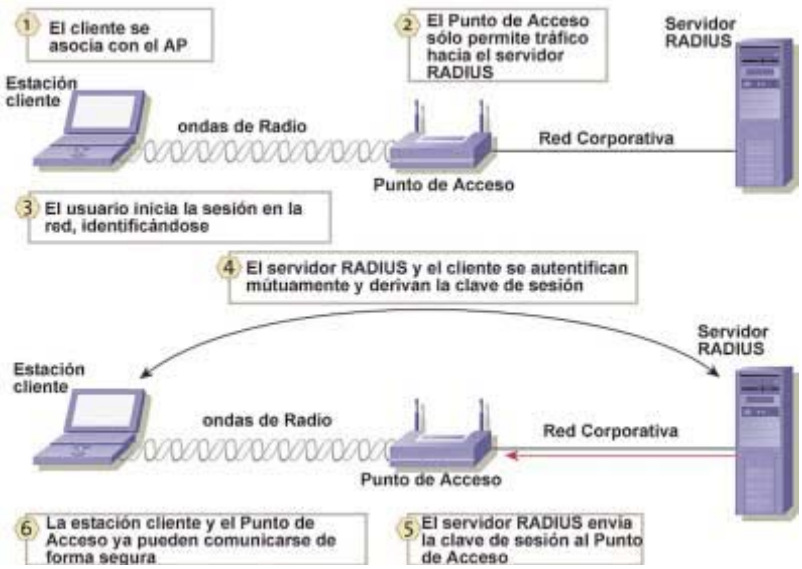
- **Authentication Server:** es el servidor de validación, reside en un servidor AAA, lo más común es utilizar un servidor RADIUS para esta tarea, es el encargado de decidir si la estación esta autorizada o no para enviar y recibir tráfico.

802.1x utiliza un método de control de acceso basado en el concepto de puerto (PAE, Port Acess Entity). El autenticador crea un puerto lógico por cliente, existiendo dos caminos uno autorizado y otro no. Mientras el cliente no se ha autenticado con éxito únicamente se permite tráfico 802.1x/EAP hacia el servidor de autenticación.

El solicitante cuando pasa a estar activo en el medio, selecciona y se asocia a un AP. El autenticador (situado en el AP) detecta la asociación del cliente y habilita un puerto para ese solicitante, permitiendo únicamente el tráfico 802.1x, el resto de tráfico se bloquea. El cliente envía un mensaje «EAP Start». El autenticador responde con un mensaje «EAP Request Identity» para obtener la identidad del cliente, la respuesta del solicitante «EAP Response» contiene su identificador y es retransmitido por el autenticador hacia el servidor de autenticación. A partir de ese momento el solicitante y el servidor de autenticación se comunicarán directamente, utilizando un cierto algoritmo de autenticación que pueden negociar. Si el servidor de autenticación acepta la autenticación, el autenticador pasa el puerto del cliente a un estado autorizado y el tráfico será permitido. Los métodos de autenticación definidos en WPA son: EAP-TLS, EAP-TTLS y PEAP. Estos métodos se basan en la infraestructura de clave pública (PKI) para autenticar al usuario y al servidor de autenticación, utilizando certificados digitales.

La premisa es la existencia de una Autoridad de Certificación (CA) de confianza para la corporación, que emita certificados para los usuarios y el servidor de autenticación. La CA puede ser privada (empresarial) o pública (basada en CAs de Internet como VeriSign...).

A continuación un esquema de lo que sería la autenticación:




Hostapd contiene la funcionalidad de autenticador y tiene que ser utilizado con el driver hostap que implementa la funcionalidad PAE: mecanismo simple mediante el que se deniegan las tramas normales que procedan de un puerto no autorizado y permite el paso de las tramas EAPOL de todos los puertos, incluyendo los no autorizados, de esta forma se permite la autenticación y en caso de ser valida se pasara de un puerto desautorizado a otro autorizado.

Resumiendo, el driver hostap se encargará de la funcionalidad PAE, y hostapd de la funcionalidad de autenticador.

Para la configuración de hostapd, se utilizará el fichero `/etc/hostapd.conf`, donde se especificarán:

- **OSA:** El punto de acceso tiene que estar en modo OSA para que los clientes puedan asociarse con el AP y enviar las tramas EAPOL de autenticación.

- **IEEE 802.1x:** Será necesaria la validación mediante este protocolo, también se tendrá que indicar la configuración referente al servidor de validación.

- **WPA:** Se utilizará este protocolo a nivel físico/enlace para cifrar y mantener la integridad de los datos.

- **WPA-EAP:** Así tendremos un control sobre usuarios en lugar de sobre dispositivos.

A continuación los parámetros mínimos a definir en el fichero `/etc/hostapd.conf` a modo de ejemplo:

```
# Interfaz wireless que se pondrá en modo Master (AP)
interface=wlan0
# Se ejecutará hostapd como demonio, en segundo plano
daemonize=1
# Identificador de la red wireless
ssid=inca
# Política a seguir con las direcciones MAC
macaddr_acl=0
# Ficheros con ACLs de MACs
accept_mac_file=/etc/hostapd.accept
deny_mac_file=/etc/hostapd.deny
# Configuramos el AP en modo OSA
auth_algs=1
# Se requiere validación mediante protocolo IEEE 802.1x
ieee8021x=1
# Configuración necesaria para comunicarse con el servidor RADIUS
# La propia dirección del AP
own_ip_addr=192.168.1.254
# El nombre de la maquina AP con su dominio incluido
nas_identifier=ap.wireless.inca.ub.es
# Direccion IP del servidor RADIUS que se utilizara para la validación
# En este caso es la misma que la del AP, pero no tiene porque serlo
auth_server_addr=192.168.1.254
# Puerto donde escucha peticiones el servidor RADIUS
auth_server_port=1812
# Contraseña mediante la que se autorizará al AP para ser
# autenticador, esta configurada en el servidor RADIUS
auth_server_shared_secret=InCa.
# Aquí exactamente lo mismo que las tres anteriores lineas
acct_server_addr=192.168.1.254
acct_server_port=1813
acct_server_shared_secret=InCa.
# Configuramos el uso de WPA
wpa=1
# WPA-EAP
wpa_key_mgmt=WPA-EAP
# Algoritmos de cifrado aceptados
wpa_pairwise=CCMP TKIP
```

Una vez configurado todo, tan solo queda lanzar el demonio hostapd:

```
hostapd -B /etc/hostapd.conf
```

Como de costumbre, a parte de configurar el modo AP, deberemos configurar la dirección IP del AP perteneciente a la red inalámbrica:

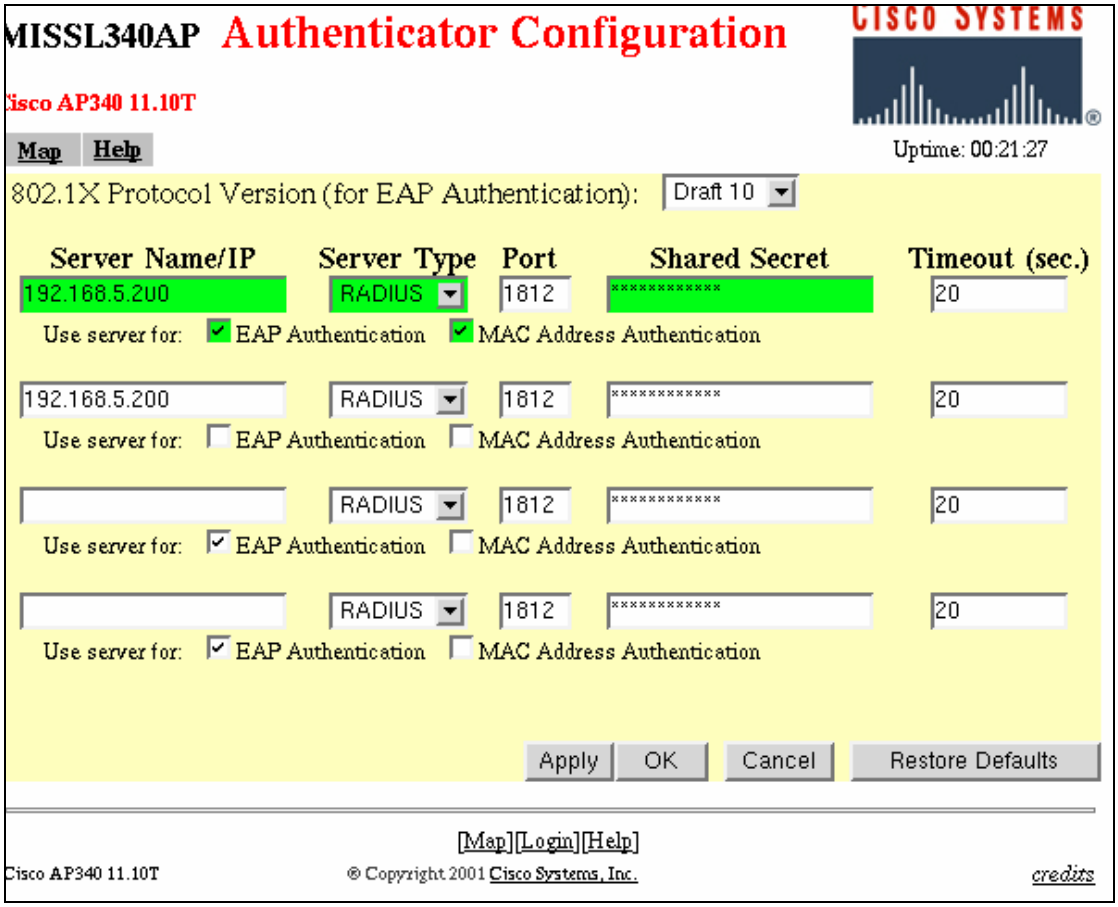
```
ifconfig wlan0 192.168.1.254 netmask 255.255.255.0 broadcast 192.168.1.255
```

Con la tarjeta wireless introducida y estos dos comandos, ya se dispone de un AP con WPA-EAP, quedará tener correctamente configurado el servidor RADIUS, tal como se verá en la siguiente sección.

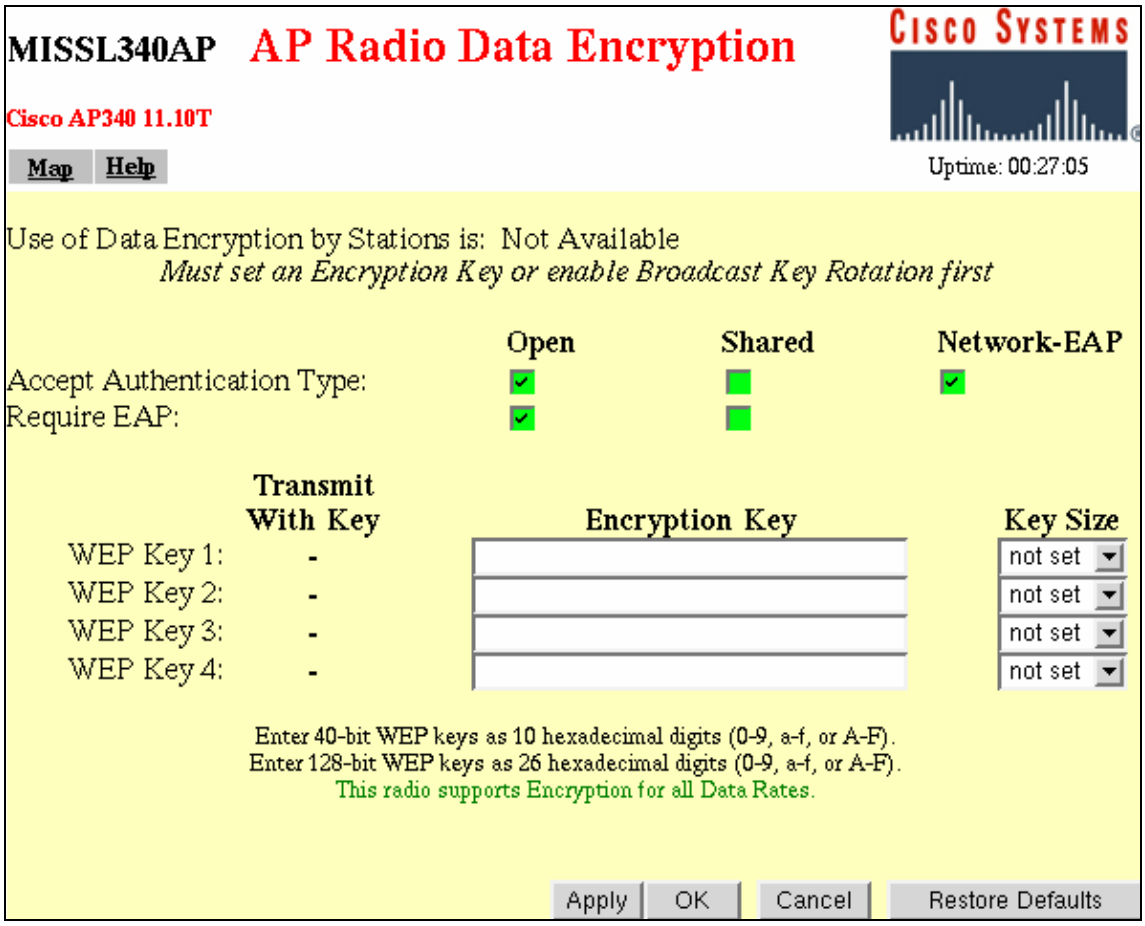
Una vez realizados todos estos pasos, dispondremos las mismas ventajas que cualquier punto de acceso comercial pero con el añadido de que en nuestro caso se trata de un PC, totalmente modular, actualizable y flexible para hacer con el lo que se desee: Por ejemplo control de ancho de banda, proxy transparente, bloquear cierto trafico, etc.

A continuación se muestra lo que sería la configuración de EAP en un punto de acceso comercial.

El correspondiente a la configuración del servidor de validación RADIUS:



El equivalente a la configuración del tipo de validación y modo de funcionamiento del AP (OSA):




La solución propuesta es totalmente compatible con los estándares que se implementan.

## 2. RADIUS

RADIUS (*Remote Authentication Dial In User Service*) es un ejemplo de un servicio AAA (Autenticación, Autorización y Accounting), en este caso, se utilizará la implementación FreeRADIUS, que es totalmente libre.

Este apartado se centrará en la instalación del servicio así como en varias utilidades que ofrece el paquete FreeRADIUS. Más adelante se dedicará un capítulo entero a los diferentes métodos de validación interesantes para redes wireless, y también la forma de configurarlos y ponerlos en funcionamiento.

Remarcar que para la funcionalidad necesaria, la versión de FreeRADIUS mínima es la 1.0.0-pre1. En el momento de la creación de este documento, la última versión disponible es la 1.0.0-pre3.

Así pues, descargamos el código de dicha versión, lo desempaquetamos y lo compilamos sin opciones especiales:

```
#wget ftp://ftp.freeradius.org/pub/radius/freeradius-1.0.0-pre3.tar.gz
--21:31:29--      ftp://ftp.freeradius.org/pub/radius/freeradius-1.0.0-
pre3.tar.gz
           => `freeradius-1.0.0-pre3.tar.gz'
Resolving ftp.freeradius.org... 64.24.0.50
Connecting to ftp.freeradius.org[64.24.0.50]:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD /pub/radius ... done.
==> PASV ... done.    ==> RETR freeradius-1.0.0-pre3.tar.gz ... done.
Length: 2,206,391 (unauthoritative)

100%[============================>] 2,206,391    221.02K/s    ETA 00:00

21:31:42   (220.95   KB/s)   -   `freeradius-1.0.0-pre3.tar.gz'   saved
[2206391]

# tar zxvf freeradius-1.0.0-pre3.tar.gz
freeradius-1.0.0-pre3/
freeradius-1.0.0-pre3/doc/
...
freeradius-1.0.0-pre3/acconfig.h
freeradius-1.0.0-pre3/CREDITS
freeradius-1.0.0-pre3/COPYRIGHT
# cd freeradius-1.0.0-pre3
# ./configure
creating cache ./config.cache
checking for gcc... gcc
...
creating ./config.status
creating Makefile
creating config.h
# make
make[1]:  Entering  directory  `/usr/local/freeradius/freeradius-1.0.0-
pre3'
Making all in libltdl...
make[2]:  Entering  directory  `/usr/local/freeradius/freeradius-1.0.0-
pre3/libltdl'
/bin/sh ./libtool --mode=compile gcc -DHAVE_CONFIG_H -I. -I. -I.     -
g  -O2  -D_REENTRANT  -D_POSIX_PTHREAD_SEMANTICS  -DOPENSSL_NO_KRB5    -
Wall -D_GNU_SOURCE -DNDEBUG -c ltdl.c
...
make[2]:  Leaving  directory  `/usr/local/freeradius/freeradius-1.0.0-
pre3/doc'
make[1]:  Leaving  directory  `/usr/local/freeradius/freeradius-1.0.0-
pre3'
# make install
/usr/local/freeradius/freeradius-1.0.0-pre3/install-sh  -c  -d  -m  755
/usr/local/sbin
/usr/local/freeradius/freeradius-1.0.0-pre3/install-sh  -c  -d  -m  755
/usr/local/bin
...
See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
```

Una vez ejecutados todos estos pasos, ya tendremos en el directorio `/usr/local/etc/raddb` los ficheros de configuración por defecto. Antes de poner el servicio en marcha, será necesario configurar los parámetros según nuestras necesidades.

# CONFIGURACIÓN RADIUS

En esta sección se parte de la base que ya se tiene el servidor FreeRADIUS instalado y con los ficheros de configuración por defecto en `/usr/local/etc/raddb`, con lo que se empezará introduciendo los diferentes protocolos de validación que son útiles para redes wireless y después se explicará cada uno un poco más en detalle.

Los protocolos son los siguientes:

- **EAP-TLS:** Validación mediante certificados.

- **EAP-PEAP:** Requiere certificado sólo del servidor. Establece un túnel TLS a través del cual se realizará la validación, a partir de ahí, se intercambian la información necesaria para la validación. Dentro del túnel únicamente se admite EAP.

- **EAP-TTLS:** Similar a PEAP, pero dentro del túnel admite cualquier protocolo de autenticación.

Los tres protocolos necesitan un certificado para el servidor que haya sido expedido por una CA de confianza, y además, EAP-TLS, también necesitará certificados para los clientes expedidos por la misma CA.

Se pueden configurar todos los métodos que se desee, será el cliente quien decida el método a utilizar para la autenticación (siempre y cuando esté configurado dicho método).

EAP-PEAP y EAP-TTLS utilizarán el certificado del servidor para iniciar una comunicación segura y establecer un túnel cifrado mediante el que negociarán la autenticación y las claves a utilizar.

Se modificarán tres ficheros de configuración, `eap.conf`, `client.conf` y `users`. En versiones anteriores, el fichero de configuración `eap.conf` no existía y los parámetros referentes a EAP se encontraban en dentro del fichero `radiusd.conf`:

- **`eap.conf`**: Configuración referente al método de autenticación a utilizar, específico para cada protocolo a utilizar.

- **`client.conf`**: Configuración necesaria para permitir al AP actuar de autenticador. Es común para todos los métodos EAP a utilizar.

- **`users`**: Aquí se darán de alta los usuarios de la red wireless. Común a todos los métodos a utilizar.

## eap.conf

Un fichero con EAP-TLS, EAP-PEAP/MSCHAPv2 y EAP-TTLS/MD5 configurado es el siguiente (seguidamente se comentan las opciones interesantes):

```
eap {

   default_eap_type = tls

   md5 {
   }

   tls {
      private_key_password = secret
      private_key_file = ${raddbdir}/certs/ap.pem
      certificate_file = ${raddbdir}/certs/ap.pem
      CA_file = ${raddbdir}/certs/root.pem
      dh_file = ${raddbdir}/certs/dh
      random_file = ${raddbdir}/certs/random
      check_cert_cn = %{User-Name}
   }

   ttls {
      default_eap_type = md5
   }

   peap {
      default_eap_type = mschapv2
   }

   mschapv2 {
   }
}
```

- **default_eap_type:** Indica el método EAP a utilizar por defecto, no es el único ni se está obligado a utilizarlo. Si el cliente intenta autenticarse mediante otro método aceptado por el servidor, no habrá problemas. Pero si no indica ninguno, se intentará con el especificado aquí.

- **md5**
- **mschapv2**

  } Métodos que utilizan usuario y contraseña para la validación.

- **ttls**
- **peap**

  } `default_eap_type:` idem que el general.

- **tls**

  - **private_key_password:** contraseña de la clave privada del servidor.
  - **private_key_file:** fichero que contiene la clave privada del servidor.
  - **certificate_file:** fichero que contiene el certificado del servidor.

- **`CA_file:`** certificado de la CA que ha emitido todos los certificados (clientes y servidor).

- **`dh_file`**
- **`random_file`**

  } Ficheros de contenido aleatorio

- **`check_cert_cn:`** Indica si se comprobará el nombre de usuario introducido con el campo CN del certificado del cliente

Para generar el fichero `dh_file` se ejecutará:

```
# openssl dhparam -check -text -5 512 -out dh
Generating DH parameters, 512 bit long safe prime, generator 5
This is going to take a long time
.....................................................+............+.........
..+.........+.....................................................
.....................................................................+.....
.....................+...+.....................................
.....................................................................
.........................++*++*++*++*++*++*++*
DH parameters appear to be ok.
```

Por ejemplo, para generar el `random_file` se puede ejecutar el siguiente comando:

```
# dd if=/dev/urandom of=random count=1024 bs=1
```

Y ya se tienen los ficheros `dh` y `random` que hacen falta.

Para la creación de los certificados puede consultarse el apéndice C, en él también pueden encontrarse scripts para la generación automática.

### `client.conf`

Se tiene que añadir en el fichero lo siguiente:

```
client 192.168.1.254 {
        secret          = InCa.
        shortname       = ap.wireless.inca.ub.es
}
```

Donde 192.168.1.254 es la dirección del AP, `secret` es la contraseña necesaria para que FreeRADIUS le reconozca como autenticador y `shortname` es el nombre por el que conocemos a dicho autenticador, normalmente su nombre DNS.

Se pueden añadir tantas entradas como se desee, normalmente una para cada punto de acceso de la red.

`users`

En este fichero se introducirán los usuarios de la red y sus contraseñas con el valor en claro, por lo que el fichero solo debe ser de lectura para los administradores.

Ejemplo:

```
user@inca            Auth-Type := EAP, User-Password == "mi_passwd"
```

Se configura un usuario llamado `user@inca` con contraseña `mi_passwd`. Nótese que la contraseña será necesaria para los métodos de validación que lo requieran, por ejemplo EAP-TTLS/MD5 entre otros. Si no es necesaria la contraseña, se ignorará, como en EAP-TLS.

# 1. EAP-TLS

Los usuarios y el servidor de autenticación deben tener un certificado digital. El solicitante, tras la asociación y la creación del puerto de acceso, envía su identificación (nombre de usuario) hacia el autenticador y éste lo reenvía hacia servidor de autenticación. Este último envía su certificado al cliente, el cliente lo valida y responde con su certificado. El servidor de autenticación comprueba si el certificado es válido (si ha sido emitido por una CA de confianza) y si el CN del certificado corresponde con el nombre de usuario antes enviado. Si es así, autentica al cliente; en caso que algún paso falle, deniega el acceso. Una vez validado el cliente, cliente y servidor establecen la clave de cifrado para esa sesión, y el servidor de autenticación la envía al punto de acceso, de forma que ya puede comunicarse con el cliente de forma segura utilizando dicha clave.

Como inconveniente, aunque es bastante relativo, encontramos que el nombre del usuario se envía en claro, con lo que cualquiera puede tener acceso a él. Decir que sin el certificado correspondiente a ese usuario, el nombre de usuario no sirve para nada.

Otro inconveniente en según que entornos puede ser la tediosa tarea de tener que generar un certificado para cada cliente y distribuirlos de forma segura, que en una red mediana/grande, puede llevar mucho tiempo y dolores de cabeza.

# 2. EAP-PEAP

Este protocolo fue diseñado por Microsoft, Cisco y RSA Security.

Consiste en crear un túnel TLS utilizando el certificado del servidor y posteriormente se negocia el protocolo de autenticación. Ofrece autenticación mutua, generación de claves y protección del nombre de usuario, ya que la validación se realiza a través del túnel TLS creado.

Tiene dos fases:

- Autenticación por parte del servidor hacia el cliente utilizando el certificado del servidor y opcionalmente autenticación del cliente hacia el servidor también con certificado.

- Si no ha habido autenticación del cliente hacia el servidor, se negociará la autenticación mediante EAP en el interior del túnel TLS creado.

# 3. EAP-TTLS

Diseñado por Funk Software. Es muy parecido a EAP-PEAP. También se crea un túnel TLS mediante el que se realizará la autenticación. Ofrece autenticación mutua, generación de claves, negociación del conjunto de cifrado de datos y protección del nombre de usuario, ya que se envía a través del túnel TLS.

Como en EAP-PEAP, tenemos dos fases:

- Establecer el túnel TLS y autenticación por parte del servidor hacia el cliente (opcionalmente autenticar al cliente también).

- Si el cliente no se ha autenticado, se usa el túnel TLS para autenticación del cliente utilizando un protocolo de autenticación (no tiene que ser EAP como sucedía en EAP-PEAP).

La negociación del conjunto de cifrado de datos consiste en que el cliente y el AP envían al servidor RADIUS los tipos de cifrado de datos que soportan, el servidor RADIUS escogerá un algoritmo soportado por ambos y enviará su elección al cliente y al AP. En caso de que cliente o AP no envíen sus preferencias, la negociación se hará en la capa de enlace entre cliente y AP, como sucede con el resto de protocolos (EAP-TLS, EAP-PEAP,...).

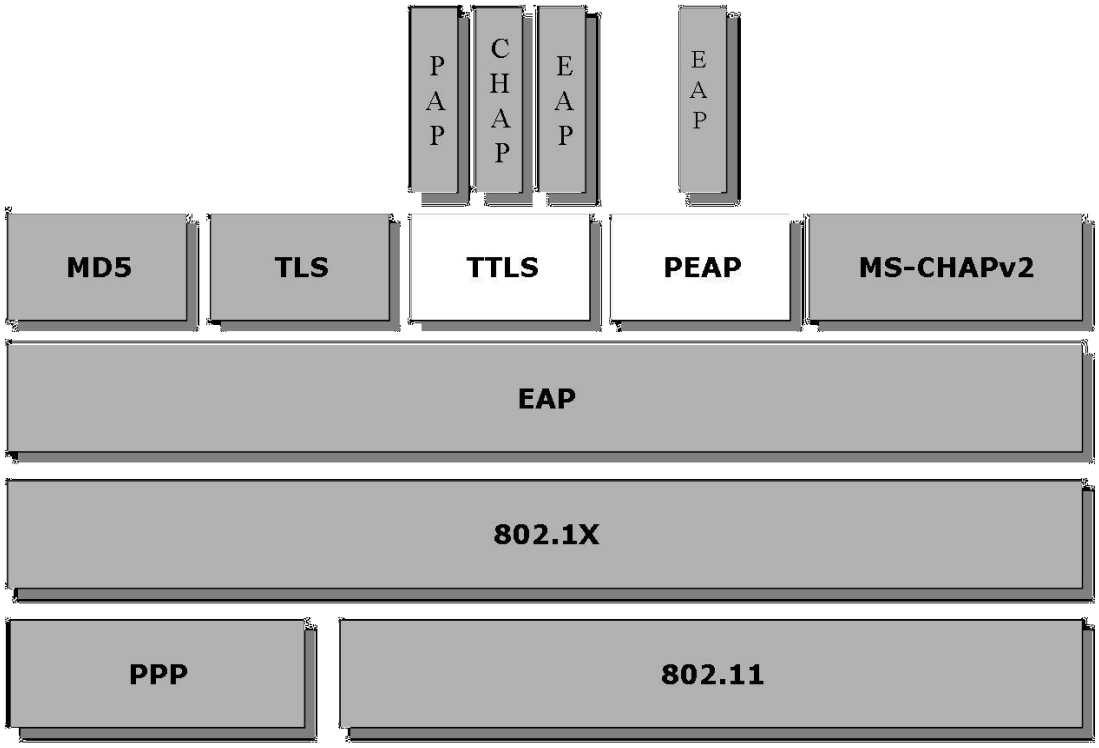
En la generación de claves, el servidor RADIUS y el cliente negocian una clave y parámetros necesarios para el cifrado entre AP y cliente (igual que sucede con los otros protocolos, TLS, PEAP,...). Posteriormente el servidor TTLS enviará dicha información al AP para que se pueda comunicar correctamente con el cliente.

# 4. Comparativa de los protocolos

Tabla comparativa de lo que ofrece cada uno de los métodos:

| | **EAP-PEAP** | **EAP-TTLS** | **EAP-TLS** |
|---|---|---|---|
| **Validación del servidor** | Certificado | Certificado | Certificado |
| **Validación del cliente** | Cualquier método EAP | Cualquier método de autenticación | Certificado |
| **Protección del nombre de usuario** | Si, TLS | Si, TLS | No |
| **Negociación del algoritmo de cifrado** | No | Si | No |
| **Ataques EAP: Session hijacking (secuestro de sesión), Man-in the middle, ataques de diccionario** | Protegido (TLS) | Protegido (TLS) | Protegido (TLS) |

Arquitectura EAP:



# CONFIGURACIÓN DEL CLIENTE

En esta sección se explicará como configurar un cliente para cada una de las soluciones propuestas:

- **Cliente básico:** Únicamente se necesitará un driver para nuestra tarjeta wireless y el paquete `wireless-tools`.

- **Cliente WEP:** Se necesitará lo mismo que en el caso anterior para configurar esta opción.

- **Cliente WPA:** Se necesitarán los paquetes `wpa_supplicant` y `wireless-tools` además de un driver que sea compatible con `wpa_supplicant`. Si la tarjeta no es una 802.11g, también será necesario disponer de una (802.11b) con chipset Prism2/2.5/3 de Intersil con un firmware 1.7.0 o superior.

La única novedad a instalar en el cliente respecto a lo explicado en la sección del AP es el `wpa_supplicant`, que únicamente es necesario si se opta por una solución WPA.

Para instalar `wpa_supplicant`, deberemos descargar el código de la última versión disponible de su página principal, desempaquetarlo, descomprimirlo, escoger lo que se quiere compilar del paquete y compilarlo. Las opciones a escoger son las siguientes:

```
CONFIG_DRIVER_HOSTAP
CONFIG_DRIVER_PRISM54
CONFIG_DRIVER_HERMES
CONFIG_DRIVER_MADWIFI
CONFIG_DRIVER_ATMEL
CONFIG_WIRELESS_EXTENSION
CONFIG_IEEE8021X_EAPOL
CONFIG_EAP_MD5
CONFIG_MSCHAPV2
CONFIG_EAP_TLS
CONFIG_EAP_PEAP
CONFIG_EAP_TTLS
CONFIG_EAP_GTC
CONFIG_EAP_SIM
CONFIG_PCSC
```

Y se deben copiar al fichero `.config` dentro del directorio del código.

En nuestro caso el fichero `.config` contendrá lo siguiente:

```
CONFIG_DRIVER_HOSTAP=y
CONFIG_WIRELESS_EXTENSION=y
CONFIG_IEEE8021X_EAPOL=y
CONFIG_EAP_MD5=y
CONFIG_MSCHAPV2=y
CONFIG_EAP_TLS=y
CONFIG_EAP_PEAP=y
CONFIG_EAP_TTLS=y
```

Ya que únicamente escogemos el driver que utilizaremos y los métodos de autenticación explicados en la configuración del AP.

Después solo quedará ejecutar `make` para que se compile lo necesario, que serán 3 ficheros, `wpa_supplicant`, `wpa_cli` y `wpa_passphrase`.

Los drivers soportados por `wpa_supplicant` son:

- **Hostap driver** (versión de desarrollo): El explicado en este proyecto.

- **Linuxant DriverLoader**: Es capaz de hacer funcionar los drivers NDIS de Windows en Linux.
  http://www.linuxant.com/driverloader/

- **Agere Systems Inc**.: http://www.agere.com/support/drivers/

- **Madwifi**: para tarjetas basadas en el chipset Atheros (ar521x).
  http://sourceforge.net/projects/madwifi/

- **ATMEL AT76C5XXx**: para tarjetas USB y PCMCIA.
  http://atmelwlandriver.sourceforge.net/

## 1. Cliente básico

Un cliente básico tan solo es capaz de asociarse a un AP con el SSID configurado siempre y cuando éste no tenga establecido ningún tipo de control de acceso.

Se deberá conocer de antemano es el SSID de la red a la que se quiere acceder. Los pasos a seguir son:

- **SSID:** Configurar el SSID de la red a la que accederse.

- **Modo Managed:** Poner la tarjeta en modo managed, para que se intente asociar a un AP.

- **Parámetros de Red:** Se deberá ejecutar un cliente DHCP para que se nos asigne una dirección y demás parámetros de red o bien establecerlos manualmente (pero con riesgo de colisionar con otros clientes).

Se configura el SSID a `inca` con

```
iwconfig wlan0 essid inca
```

Para poner la tarjeta de red en modo managed ejecutamos

```
iwconfig wlan0 mode managed
```

Y finalmente se ejecuta el cliente de DHCP para el interfaz wireless, en nuestro caso `wlan0`:

```
dhclient –e –pf /var/run/dhclient.wlan0.pid \
–lf /var/run/dhclient.wlan0.leases wlan0
```

## 2. Cliente WEP

Un cliente WEP necesita, además de lo especificado en la configuración anterior, una clave con la que cifrar los datos a enviar (protocolo WEP).

Para configurar dicha clave se ejecutará

```
iwconfig wlan0 key <clave en hexadecimal>
```

o

```
iwconfig wlan0 key s:<clave en ASCII>
```

justo antes de arrancar el cliente DHCP.

## 3. Cliente WPA

Aquí dividiremos en dos grupos: WPA-PSK y WPA-EAP.

- **WPA-PSK:** La configuración sería muy parecida al cliente WEP, ya que únicamente se debe especificar la clave WPA-PSK a utilizar para el cifrado. La diferencia sería que no es suficiente con el driver y la tarjeta para realizar todo el trabajo de cifrado y descifrado, sino que es necesaria la utilización del programa `wpa_supplicant`. Por tanto, la configuración para este modo de funcionamiento se hará toda en el fichero de configuración `/etc/wpa_supplicant.conf`.

- **WPA-EAP:** Toda la configuración para este modo se hará también en `/etc/wpa_supplicant.conf`. Esta vez, dicho fichero tendrá más parámetros para la configuración, que dependerán del método EAP a utilizar.

Parámetros generales que se deben incluir en el fichero de configuración, independientemente de si se utiliza WPA-PSK o WPA-EAP:

- **SSID:** Indica el SSID de la red a la que nos queremos asociar

- **Protocolo:** Normalmente será WPA, pero también se podría utilizar RSN (*Robust Security Network*), que hace referencia a lo que será el próximo estándar para redes Wi-Fi: 802.11i. Por el momento RSN se basa en las especificaciones descritas en los borradores del protocolo.

- **Gestión de claves:** `PSK` o `EAP`, dependiendo del modo que configuremos (aunque también admite `IEEE8021X` utilizando claves WEP dinámicas y `NONE` para no utilizar cifrado o una clave WEP estática).

- **Algoritmo de cifrado:** `CCMP` o `TKIP`. CCMP utiliza AES y formará parte del nuevo estándar 802.11i.

- **Algoritmo de cifrado para grupo:** `CCMP`, `TKIP`, `WEP104` o `WEP40`. Hace referencia al algoritmo que se utilizará para el cifrado del tráfico multicast o broadcast.

## 3.1. WPA-PSK

Para este modo, se deben especificar en el fichero de configuración los siguientes parámetros:

```
network={
        # WPA, configuracion general
        ssid="inca"
        proto=WPA
        key_mgmt=WPA-PSK
        pairwise=CCMP TKIP
        group=CCMP TKIP WEP104 WEP40

        # WPA-PSK
        psk=fbea84528c5edc1f86aa18a389976a0e2c4a7e67f4056e4e42b36c5...
}
```

Con esto tendremos configurado el cliente para que se una a la red con SSID `inca` utilizando el protocolo WPA y WPA-PSK como gestor de claves. Los algoritmos de cifrado a utilizar son CCMP y TKIP, en este orden, si puede utilizar CCMP lo utilizará, y sino hará uso de TKIP. Para el cifrado de grupo soportará todos los algoritmos.

Finalmente, la clave se genera con el comando `wpa_passphrase` pasándole como parámetro el SSID y una cadena (`InCaInCa` en el ejemplo):

```
# wpa_passphrase inca InCaInCa
network={
        ssid="inca"
        #psk="InCaInCa"
        psk=fbea84528c5edc1f86aa18a389976a0e2c4a7e67f4056e4e42b36c5...
}
```

Una vez configurado será necesario arrancar el demonio `wpa_supplicant` y el cliente de DHCP. `wpa_supplicant` se ejecuta de la siguiente forma:

```
wpa_supplicant -B -c/etc/wpa_supplicant.conf -iwlan0
```

donde el parámetro `–B` indica que se ejecutará en segundo plano, `-c` el fichero de configuración (`/etc/wpa_supplicant.conf`) y `–i` el interfaz wireless (`wlan0`). El cliente DHCP se arrancará como en los apartados anteriores.

## 3.2. WPA-EAP

Este modo tiene varias opciones de configuración, veamos las soportadas: TLS, PEAP y TTLS.

Para utilizar TLS se requieren certificados expedidos por una misma CA (Autoridad Certificadora) tanto para el servidor como para cada uno de los clientes. En cambio, para PEAP y TTLS solo es necesario certificado para el servidor, y el certificado CA para el cliente, de esa forma validará la autenticidad del servidor e iniciará una conexión segura mediante la que se negociará el protocolo a utilizar para la autenticación (algunos de ellos: MD5, TLS, SIM, MSCHAPv2, GTC,…).

Dependiendo del protocolo escogido para usar en el túnel (TTLS o PEAP), se requerirá una u otra información adicional. Por ejemplo, si se escoge TLS, se necesitarán certificados y si se escoge MD5 hará falta un par usuario/contraseña.

A continuación se verán algunos ejemplos de configuraciónes comunes (EAP-TLS, EAP-PEAP/MSCHAPv2 y EAP-TTLS/MD5).

### EAP-TLS

Para configurar EAP-TLS necesitaremos nuestra clave privada, nuestro certificado y el certificado de la CA, que se utilizará para validar el certificado del servidor. Dichos certificados nos los proporcionará el administrador de la red, así como el nombre de usuario que se deberá utilizar. El nombre de usuario debe coincidir con el campo CN de nuestro certificado, si no coincide, el servidor denegará el acceso.
Crearemos un directorio (`/etc/cert`) en el que copiaremos el certificado CA (`root.ca`) y nuestra clave privada y certificado (`usuario@inca.pem`). En este caso, nuestra clave privada esta protegida por contraseña (`InCa.`).

Supongamos que el certificado tiene valor CN a "`usuario@inca`", la configuración sería la siguiente:

```
network={
        ssid="inca"
        proto=WPA
        key_mgmt=WPA-EAP
        pairwise=CCMP TKIP
        group=CCMP TKIP
        eap=TLS
        identity="usuario@inca"
        ca_cert="/etc/cert/root.ca"
        client_cert="/etc/cert/usuario@inca.pem"
        private_key="/etc/cert/usuario@inca.pem"
        private_key_passwd="InCa."
}
```

### EAP-PEAP/MSCHAPv2

Para configurar EAP-PEAP/MSCHAPv2 se necesitará un nombre de usuario, una contraseña y, como siempre, el certificado de la CA que emitió el certificado y clave del servidor.

Nótese que en este caso se creará un túnel TLS a través del cual se realizará la validación con el protocolo MSCHAPv2, pero que podría utilizarse con otros protocolos como MD5, TLS,…

Así pues, el fichero de configuración `/etc/wpa_supplicant.conf` quedaría de la siguiente forma:

```
network={
        ssid="inca"
        key_mgmt=WPA-EAP
        eap=PEAP
        anonymous_identity="anonimo@inca"
        phase2="auth=MSCHAPV2"
        identity="usuario@inca"
        password="mi_passwd"
        ca_cert="/etc/cert/root.ca"
}
```

Observamos nuevos parámetros de configuración:

- **`anonymous_identity`:** es el usuario que se envía en la primera conexión para establecer el túnel TLS. Se puede poner cualquier cosa siendo aconsejable que sea diferente del usuario real.

- **`phase2`:** se especifican parámetros de configuración para la segunda fase de la autenticación, la que circula dentro del túnel TLS creado.

- **`identity`:** en este caso, es el usuario que circula por el interior del túnel.

- **`password`:** Contraseña que se utilizará para realizar la autenticación

`identity` y `password` han de coincidir con una entrada del fichero `users` de RADIUS.

EAP-TTLS/MD5

Este método es muy parecido al anterior (EAP-PEAP/MSCHAPv2), también se genera un túnel TLS utilizando el certificado del servidor para después negociar el protocolo a utilizar dentro del túnel para realizar la autenticación.

Fichero de configuración:

```
network={
        ssid="inca"
        key_mgmt=WPA-EAP
        eap=TTLS
        anonymous_identity="anonimo@inca"
        phase2="auth=MD5"
        identity="usuario@inca"
        password="secret"
        ca_cert="/etc/cert/root.ca"
}
```

# IPSEC

Recordemos que lo que se pretende en un primer momento es implementar un control de seguridad en redes wireless. A pesar de ello, esta solución no es específica para redes wireless, se puede implementar en cualquier red, ya sea cableada o sin cables.

Lo deseable sería obtener un método fiable en el nivel físico, sin necesidad de recurrir a niveles más altos de la pila TCP/IP. Esto no siempre es posible, ya que nos podemos encontrar con limitaciones impuestas por las tarjetas que utilicemos o bien por el sistema operativo empleado. Teniendo en mente esto, debemos pensar en una solución alternativa que siempre sea posible implementar.

Este es el caso del protocolo IPSec. IPSec es un protocolo pensado para ofrecer una seguridad a nivel de red y es opcional para IPv4 y obligatorio para IPv6.

Ofrece tanto cifrado como autenticación a nivel de red, lo cual nos será realmente útil para nuestro propósito.

Existen varias posibles configuraciones, las principales son:

- Establecimiento de un túnel estático

- Road Warrior

- *Opportunistic encryption*

La solución aconsejable sería *Opportunistic encryption* ya que es la que más se adapta a nuestras necesidades.

Para un correcto funcionamiento del túnel estático así como de OE (*Opportunistic Encryption*), se configurará el servidor DHCP para que asigne siempre la misma dirección a cada cliente, en cada una de las secciones se verá el porque.

Road Warrior en cambio está pensado para direcciones dinámicas, con lo que se explicará el funcionamiento y se pondrá un ejemplo de cómo configurarlo pero no se le dará mayor importancia.

Las explicaciones y configuraciones se verán de forma genérica, ya que no es específico a una red wireless. Así pues, no se hablará ni de puntos de acceso ni de clientes, sino de dos máquinas (o extremos) que serán las que establezcan el túnel seguro.

Destacar que FreeSWAN es un proyecto que no está en desarrollo actualmente. Por otra parte, existen otros proyectos que surgieron a partir de FreeSWAN y que si que se continúan desarrollando y ampliando funcionalidades (OpenSWAN y StrongSWAN).

# 1. Túnel estático

Consiste en generar las claves y configurar los dos extremos de la VPN para todos y cada uno de los túneles que se quieran crear. Para dicha configuración es necesario conocer las IPs de los extremos (y sus respectivos gateways) y el intercambio de claves públicas entre ambos por un método seguro.

El establecimiento del túnel se puede realizar automáticamente cuando se lanza el servicio (que puede ser al iniciar la máquina o cuando se necesite), o manualmente cuando se desee establecer la VPN. Cada VPN tiene un nombre único (parámetro `conn`) que la identifica, y será con el que haremos referencia a ella cuando la queramos iniciar o detener.

`/etc/ipsec.conf` es el fichero de configuración principal, en él se definen todos los parámetros generales así como los túneles a crear.

`/etc/ipsec.secrets` es el fichero que contiene la clave pública y privada de la máquina en cuestión. Es un fichero que tiene que estar protegido y mantenido en secreto, ya que toda la seguridad está basada en dicho fichero.

Supongamos que se quiere definir un túnel entre una máquina con dirección 172.17.1.1 y otra con dirección 10.1.1.2. Lo primero que se debe hacer es generar las claves pública y privada, si no se tienen todavía, para cada una de las máquinas.

Para ello, ejecutaremos:

```
ipsec newhostkey --output - --bits 4096 > /etc/ipsec.secrets
```

Y se generará una clave de 4096 bits. En caso de querer una longitud de clave diferente, se cambia el “4096” por la longitud deseada, siempre múltiplo de 16.

Una vez hecho esto, ya se tendrán en cada máquina sus claves necesarias para el túnel.

Cuando se define un túnel, se hace referencia a los extremos con `left` y `right`, normalmente se utiliza el Left para Local y el Right para Remoto, y aunque no es necesario, se tendrá que ser coherente con los extremos (por ejemplo: si a `left` le asignamos la dirección local, hay que asegurarse que `leftrsasigkey` sea la clave pública de dicha máquina):

```
conn VPNtunel
        left=10.1.1.2
        leftnexthop=10.1.1.254
leftrsasigkey=0sAQOC2SIarKvA4cks7QjzL3gKw/n6zqP9IzODAv/cs1Ck...
        right=172.17.1.1
        rightnexthop=172.17.1.254
rightrsasigkey=0sAQOBpROoYr0BBG3UaNVJFcD2euVQsB0r38PcHUDP...
        keyingtries=%forever
        compress=yes
        auto=start
```

Se pasan a comentar dichas opciones de configuración:

- **`left:`** IP de la máquina local.

- **`leftnexthop:`** IP del gateway local.

- **`leftrsasigkey:`** Clave pública local. A continuación se verá como saber la clave de una máquina.

- **`right:`** IP del otro extremo del túnel.

- **`rightnexthop:`** Gateway del host remoto.

- **`rightrsasigkey:`** Clave pública remota. A continuación se verá como extraer la clave de una máquina.

- **`keyingtries:`** Número máximo de intentos para negociar la conexión.

- **`compress:`** Indica si se comprimen los datos a enviar.

- **`auto:`** Indica el tipo de arranque de la conexión a realizar una vez arrancado el servicio IPSec, los tipos más importantes son:

  - **`add:`** Quedará configurada la conexión pero no intentará establecer el túnel al inicio del servicio, sino que se tendrá que hacer manualmente cuando se quiera.

  - **`route:`** Añade una entrada en la tabla de *routing*, y se generará un túnel cuando sea necesario y sea posible (por ejemplo en configuraciones *Opportunistic Encryption*).

  - **`start:`** Configura e intenta establecer el túnel al inicio del servicio IPSec.

  - **`ignore:`** Ignorará la conexión, como si no hubiera nada definido.

Para obtener las claves públicas de ambas máquinas se deberá ejecutar en la máquina que se quiere saber su clave pública:

```
ipsec showhostkey –-right
        # RSA 4096 bits   servidor   Wed Apr  9 11:15:41 2003
rightrsasigkey=0sAQOC2SIarKvA4cks7QjzL3gKw/n6zqP9IzODA...
```

En caso de necesitar la clave pública para el extremo `left`, se ejecutará lo mismo cambiando `right` por `left`, o también se puede cambiar a mano el principio de la línea de `rightrsasigkey` a `leftrsasigkey`, que es la única diferencia en la salida de ambos comandos.

Una vez definidas tantas conexiones como túneles se quieran establecer, se iniciará el servicio IPSec, en caso de no estar arrancado, con `/etc/init.d/ipsec start`.

Si ya estuviera en marcha, se deberán ejecutar los siguientes comandos:

```
ipsec auto --rereadgroups
ipsec auto --rereadsecrets
```

si se ha cambiado alguna clave privada o política de grupo. Y para añadir la nueva conexión si se ha creado o bien modificado algún valor:

```
ipsec auto --add VPNtunel
```

Si lo que se quiere es iniciar un túnel se tendrá que ejecutar:

```
ipsec auto --up VPNtunel
```

para detenelo:

```
ipsec auto --down VPNtunel
```

Para saber si el túnel se ha establecido con éxito, se debe buscar en los logs:

```
104 "VPNtunel" #223: STATE_MAIN_I1: initiate
106 "VPNtunel" #223: STATE_MAIN_I2: sent MI2, expecting MR2
108 "VPNtunel" #223: STATE_MAIN_I3: sent MI3, expecting MR3
004 "VPNtunel" #223: STATE_MAIN_I4: ISAKMP SA established
112 "VPNtunel" #224: STATE_QUICK_I1: initiate
004 "VPNtunel" #224: STATE_QUICK_I2: sent QI2, IPsec SA established
```

Lo verdaderamente importante es `IPsec SA established`, que nos indicará que el túnel se ha establecido.

Nótese que se puede tener el servicio IPSec en marcha sin necesidad de tener ningún túnel funcionando o ni tan si quiera definido.

Si un túnel se intenta poner en marcha y el otro extremo no esta disponible, se intentará establecer el túnel un número de veces definido (parámetro `keyingtries`)

Para una red wireless, en la que cada cliente establece un túnel con el AP se debería definir una conexión en cada cliente con el AP y tantas conexiones como clientes se tengan en la red para el AP. Esto hace que sea una solución incómoda porque la administración del AP y de su fichero `ipsec.conf` se hace interminable.

Con esta configuración los clientes tampoco establecerán ninguna VPN entre ellos, y en caso de querer tener seguridad entre clientes también, se deberían añadir tantas conexiones como clientes en los ficheros de configuración de cada cliente, lo que es inviable para una red mediana o grande.

Una solución a semejante lío podría ser jugar con el enrutamiento, obligando a los clientes a enviar todo su tráfico a un *gateway* por defecto, que seria el AP. De esta forma se llegaría a nivel 3 de la pila TCP/IP, con lo que el AP desencapsularía los datos de un túnel para meterlos en otro túnel (origen y destino). Así se ahorra tener que configurar

todas las conexiones entre clientes, ya que todo el tráfico se envía a través del mismo túnel, el establecido con el AP.

Para que esto funcione, se necesitaría la siguiente tabla de enrutamiento (suponiendo la red 192.168.1.0/24)

```
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.254   0.0.0.0         255.255.255.255 UH    0      0        0 wlan0
0.0.0.0         192.168.1.254   0.0.0.0         UG    0      0        0 wlan0
```

## 2. Road Warrior

Se trata de un túnel estático, solo que las direcciones de los extremos y sus respectivos gateways, no tienen porqué ser conocidas de antemano. Esta configuración soluciona el problema de clientes con direcciones dinámicas.

Esta solución permitiría la asignación dinámica de direcciones a los clientes, ya que los túneles se podrían establecer igualmente.

La configuración solo tiene unas pequeñas diferencias con la de la sección anterior, y son las siguientes:

- **Identificador de conexión**: Se añaden dos parámetros nuevos de conexión, `leftid` y `rightid`, que definen identificadores mediante los que se sabrá a que conexión hace referencia. Recordemos que en la configuración anterior se sabía porque las direcciones eran estáticas, ahora las direcciones son dinámicas, con lo que se identificarán mediante este identificador. Se puede utilizar un identificador equivalente a un nombre de usuario, que identifique de forma única a cada cliente.

- **`right`** (o `left`, hace referencia a la dirección remota, desconocida de antemano por ser dinámica): ya no especificará una dirección concreta sino `%any`, que es una variable especial para este tipo de configuración.

- **`rightnexthop`** (o `leftnexthop`, hace referencia a la dirección del gateway remoto, desconocida de antemano por ser dinámica): ya no especificará una dirección concreta sino `%defaultroute`, que es una variable especial para este tipo de configuración.

Esta configuración tiene los mismos problemas que la del apartado anterior: ficheros de configuración demasiado largos y configuración tediosa. La única diferencia con la configuración anterior es que se podrían asignar direcciones dinámicas sin que afecte al establecimiento de los túneles.

Una posible variante de esta configuración sería tener una única clave pública y privada para todos los clientes. De esa forma, el fichero de configuración del AP sería muy manejable y corto, y para los clientes se tendrían los mismos ficheros de configuración.

Para poder hacer esto, sería necesario añadir a la configuración de `ipsec.conf` el parámetro `uniqueids=no`, lo que permitiría establecer más de un túnel para cada definición de conexión. En caso contrario, en el momento que un cliente se conecte, se desconectaría automáticamente el que estuviera conectado en ese momento, ya que sólo puede existir una única conexión con los mismos identificadores (parámetro `conn`).

La solución de una única clave pública y privada para todos los clientes, aunque sencilla y rápida, no es aconsejable, ya que se está basando toda la seguridad en tan solo 2 ficheros de texto que contienen las claves, en caso de robo se comprometería toda la red y se deberían generar nuevas claves así como redistribuirlas a todos los clientes. Además, no se podría denegar el acceso a un usuario concreto, ya que se estaría bloqueando a todos los usuarios. En caso de querer bloquear la entrada a un usuario, se deberían generar nuevas claves y redistribuirlas a todos los que continuaran siendo usuarios legítimos. Nótese que la distribución de los ficheros referentes a los clientes se tiene que hacer de forma segura, ya que contienen la clave secreta.

## 3. Opportunistic encryption

Quizá esta sea la opción más interesante, ya que ahorra gran trabajo en entornos en los que hay muchas máquinas y se quiere que todas sean capaces de establecer un túnel, en caso que sea necesario, con cualquiera de las restantes máquinas.

La solución se basa en introducir una entrada del tipo TXT en el registro DNS correspondiente por cliente. Esta entrada contendrá la clave pública del cliente, con lo que cada vez que una máquina se quiera conectar con otra, simplemente se consultarán los servidores DNS y establecerá un túnel para la comunicación si están disponibles las claves de ambos (aunque para una máquina sólo es necesario conocer la clave pública de su “corresponsal”, también se da la situación simétrica y por tanto las claves públicas de las dos tienen que estar accesibles para establecer el túnel). Tiene la ventaja de poderse utilizar también con IPs dinámicas, siempre y cuando las entradas DNS sean correctas (se puede utilizar algún servicio DNS de tipo dinámico).

En un primer momento, las conexiones pueden realizarse de forma más lenta de lo normal, ya que cuando se intenta conectar con una máquina, primero se consulta al servidor DNS para saber si tiene o no clave pública, con lo que como mínimo se debe esperar un *timeout* para dicha petición. En caso de no disponer de clave pública, se establecería comunicación normal, sin cifrado, y en caso de disponer de la clave, se establecería el túnel.

Esta solución depende totalmente del correcto funcionamiento del servicio de DNS, de forma que si este servicio falla o resulta comprometido, dejará de ser válida y segura.

Esta configuración tiene además la ventaja de la sencillez en configuración de tanto los clientes como el AP, ya que solo se tiene que especificar que se quiere utilizar OE (*Opportunistic Encryption*) y añadir una entrada por cliente en el servidor DNS conteniendo la clave pública.

Con esta solución, tenemos la ventaja que si se quiere eliminar a un usuario en concreto, bastará con eliminar su clave pública del DNS.

Para que esta configuración sea totalmente segura, tan solo tiene que permitirse tráfico IPSec, denegando todo aquel que no vaya cifrado. De esta forma será necesaria la intervención de un administrador de red para dar acceso a los usuarios añadiendo la clave pública en el DNS.

Se pueden obtener las líneas a introducir en el servidor DNS automáticamente con el comando:

```
ipsec mailkey --me admin@wireless.inca.ub.es \
--forward pc001.wireless.inca.ub.es
```

que creará el fichero `OE_mail_pc001.wireless.inca.ub.es` conteniendo la entrada a añadir en el servidor DNS.

Opcionalmente se puede también añadir al DNS la entrada para la resolución inversa:

```
ipsec mailkey --me admin@wireless.inca.ub.es --reverse 192.168.1.1
```

y en el fichero `OE_mail_192.168.1.1` se tendrá, como en el caso anterior, la línea a introducir en el fichero correspondiente del servidor DNS.

Se puede comprobar si se han introducido correctamente las claves en el servidor DNS ejecutando:

```
ipsec verify --host ap.wireless.inca.ub.es
```

y la salida debe ser similar a:

```
Checking ap.wireless.inca.ub.es for Opportunistic Encryption:
Looking for TXT in forward map: ap.wireless.inca.ub.es          [OK]
Looking for TXT in reverse map: 254.1.168.192.in-addr.arpa.     [OK]
```

Una vez colocadas todas las claves públicas necesarias en el servidor DNS, deberá añadirse una nueva entrada en `/etc/ipsec.conf` para configurar IPSec con OE:

```
conn sg-to-anyone
        left=%defaultroute
        right=%opportunistic
        authby=rsasig
        keyingtries=2
        auto=route
```

También deberemos configurar IPSec para que solo acepte conexiones cifradas (a través de túneles), ya que se está basando toda la seguridad y autenticación de la red en ello. Para conseguirlo, se deberá definir la política `private`, con una entrada especial en el fichero `/etc/ipsec.conf`:

```
conn private
        type=tunnel
        right=%defaultroute
        left=%opportunisticgroup
        failureshunt=drop
        auto=route
```

Así, se intentará la creación de un túnel cifrado con cualquier máquina que quiera establecer una conexión. Si el túnel no se puede crear (por no existir su clave pública en el servidor DNS) se denegará la conexión.

Para finalizar se tendrá que indicar que queremos utilizar la política `private` para cualquier destino, con lo que añadiremos la línea `0.0.0.0/0` en el fichero `/etc/ipsec.d/policies/private`. Se reinicia el servicio y ya se tendrá configurado el AP para utilizar OE.

Únicamente hay que tener en cuenta que para cada cliente nuevo se deberá introducir su correspondiente clave pública en el servidor DNS y asegurarse que el servidor DHCP siempre le asigna la misma dirección IP y que además corresponda con la configurada en el servidor DNS. Por supuesto, habrá que configurar los ficheros de IPSec en la estación cliente como se acaba de indicar.

# CONCLUSIONES

Inicialmente, se quería montar un AP en una red cableada existente (direccionamiento público) para dar acceso a la red inalámbrica (direccionamiento privado). Su cometido sería controlar todos los accesos de la red inalámbrica hacia el exterior así como mantener un control de acceso a la propia red inalámbrica. Además se deseaba poder garantizar integridad y confidencialidad en las comunicaciones.

El AP actúa de cortafuegos y servidor de autenticación (RADIUS) al mismo tiempo, aunque éste último se podría configurar en una máquina distinta. Como cortafuegos, tan solo se permitirán conexiones de la red inalámbrica hacia el exterior de los clientes validados, y en ningún caso al revés.

Finalmente se han podido llevar a cabo todos los objetivos, se han dado a conocer todas las soluciones que se venían utilizando hasta hace poco así como las que se están imponiendo actualmente en los entornos empresariales, que requieren una mayor seguridad ante posibles agresiones, robo, pérdida de datos,…

Todas las pruebas realizadas se han hecho con Linux, tanto parte de cliente como de servidor, que era el objetivo del proyecto. De todas formas, el sistema se basa en los estándares definidos por las organizaciones internacionales (IEEE, IETF), con lo que cualquier sistema que cumpla los estándares será compatible con las soluciones propuestas. Concretamente se ha buscado información sobre la compatibilidad con sistemas MS Windows, y se ha llegado a la conclusión que, aunque puedan surgir problemas, actualmente se puede llegar a configurar los clientes para utilizar las soluciones WEP, WPA-PSK y WPA-EAP. En lo referente a protocolos EAP, está probada la compatibilidad con EAP-TLS y con EAP-PEAP/MSCHAPv2.

En cuanto a la solución alternativa de utilización de VPNs (IPSec) para securizar la red y autenticar a los usuarios, decir que también es compatible siempre y cuando IPSec sobre Linux soporte certificados X.509 (la versión OpenSWAN ya lo tiene), que son los utilizados en Windows.

# APÉNDICES

## A. Configuración firewall, script de ejemplo

```
#!/bin/bash
#

echo "Configurando firewall:"


###############################################################################
#
# Configuracion de la red de area local.
#
# Rango de la red local y localhost IP. /24 significa que solo se
# utilizaran los primeros 16 bits de la direccion IP, lo mismo que una
# mascara de red 255.255.255.0
#

echo -n "  · Declarando variables de entorno:"

echo -n " WIRELESS"
export WIRELESS_IP="192.168.1.254"
export WIRELESS_IP_RANGE="192.168.1.0/24"
export WIRELESS_BCAST_ADRESS="192.168.1.255"
export WIRELESS_IFACE="wlan0"

echo -n " IPSEC"

### ATENCION!!! Cambiar ipsec0 por ipsec1 para probarlo con wifi ###

export IPSEC_IFACE="ipsec0"

###############################################################################
#
# Localhost Configuration.
#

echo -n " LO"
export LO_IFACE="lo"
export LO_IP="127.0.0.1"


###############################################################################
#
# Internet Configuration.
#

echo -n " INET"
export ROUTER_IP="172.17.1.1"
export INET_IP="172.17.1.254"
export INET_IP_RANGE="172.17.1.0/24"
export INET_BCAST_ADRESS="172.17.1.255"
export INET_IFACE="eth0"


###############################################################################
#
# Internet IP's.
#

echo -n " IPs"
export HORA="172.17.1.2"
export APT="172.17.1.2"
```

```
export INET="0.0.0.0/0"
################################################################################
#
# IPTables Configuration.
#

echo -n " IPTABLES"
export IPTABLES=/sbin/iptables

echo


################################################################################
#
# Borramos TODO lo configurado en el firewall.
#
# Quitamos todas las reglas configuradas
# Ponemos todas las politicas a ACCEPT
# Borramos las chains configuradas
#

#
# Reseteamos y aceptamos todo en la tabla mangle
#

echo "  · Aceptamos TODO lo entrante y saliente"

${IPTABLES} -t mangle -P PREROUTING ACCEPT
${IPTABLES} -t mangle -F PREROUTING
${IPTABLES} -t mangle -P INPUT ACCEPT
${IPTABLES} -t mangle -F INPUT
${IPTABLES} -t mangle -P FORWARD ACCEPT
${IPTABLES} -t mangle -F FORWARD
${IPTABLES} -t mangle -P OUTPUT ACCEPT
${IPTABLES} -t mangle -F OUTPUT
${IPTABLES} -t mangle -P POSTROUTING ACCEPT
${IPTABLES} -t mangle -F POSTROUTING

#
# Reseteamos y aceptamos todo en la tabla nat
#

${IPTABLES} -t nat -P PREROUTING ACCEPT
${IPTABLES} -t nat -F PREROUTING
${IPTABLES} -t nat -P POSTROUTING ACCEPT
${IPTABLES} -t nat -F POSTROUTING
${IPTABLES} -t nat -P OUTPUT ACCEPT
${IPTABLES} -t nat -F OUTPUT

#
# Reseteamos y aceptamos todo en la tabla filter
#

${IPTABLES} -t filter -P INPUT ACCEPT
${IPTABLES} -t filter -F INPUT
${IPTABLES} -t filter -P FORWARD ACCEPT
${IPTABLES} -t filter -F FORWARD
${IPTABLES} -t filter -P OUTPUT ACCEPT
${IPTABLES} -t filter -F OUTPUT

#
# Eliminamos las chains creadas
#

${IPTABLES} -F bad_tcp_packets &> /dev/null
${IPTABLES} -X bad_tcp_packets &> /dev/null
${IPTABLES} -F tcpincoming_packets &> /dev/null
${IPTABLES} -X tcpincoming_packets &> /dev/null
${IPTABLES} -F tcpoutgoing_packets &> /dev/null
${IPTABLES} -X tcpoutgoing_packets &> /dev/null
${IPTABLES} -F udpincoming_packets &> /dev/null
${IPTABLES} -X udpincoming_packets &> /dev/null
${IPTABLES} -F udpoutgoing_packets &> /dev/null
${IPTABLES} -X udpoutgoing_packets &> /dev/null
${IPTABLES} -F allowed &> /dev/null
${IPTABLES} -X allowed &> /dev/null
```

```
#############################################################################
#
# Si STOP, entonces salimos y no configuramos ninguna regla.
#
# En caso de parar el firewall, salimos del script y dejamos todas las
# politicas en ACCEPT y sin ninguna regla ni chain de filtrado
#

if [ "$1" == "stop" ] ; then
  echo "  · Done"
  exit
fi


#############################################################################
#
# Configuracion de las politicas de seguridad.
#
# Por defecto denegamos todo
# Creamos las chains que necesitaremos para nuestra configuracion
# Y configuramos las chains
#

#
# Politicas por defecto DROP
#

echo "  · Configurando politicas a DROP"

${IPTABLES} -P INPUT DROP
${IPTABLES} -P FORWARD DROP
${IPTABLES} -P OUTPUT DROP

#
# Creamos las chains que utilizaremos
#

echo "  · Creando y configurando chains necesarias"

${IPTABLES} -N allowed
${IPTABLES} -N bad_tcp_packets
${IPTABLES} -N tcpincoming_packets
${IPTABLES} -N tcpoutgoing_packets
${IPTABLES} -N udpincoming_packets
${IPTABLES} -N udpoutgoing_packets

#
# Configuramos los paquetes TCP "ilegales"
#

${IPTABLES} -A bad_tcp_packets -p tcp -m tcp ! --tcp-flags SYN,RST,ACK SYN -m state --
state NEW -j DROP

#
# Aquí se denegarian los paquetes que entraran por el interfaz de internet y
# tuvieran como origen una direccion IP privada, ya que seria imposible,
# pero en nuestro ejemplo, nuestra red de internet es una privada, asi que
# comentamos dichas reglas
#
#${IPTABLES} -A bad_tcp_packets -s 192.168.45.0 -i ${INET_IFACE} -j DROP
#${IPTABLES} -A bad_tcp_packets -s 10.0.0.0/255.0.0.0 -i ${INET_IFACE} -j DROP
#${IPTABLES} -A bad_tcp_packets -s 172.16.0.0/255.255.0.0 -i ${INET_IFACE} -j DROP

#
# Configuramos los paquetes permitidos
#

${IPTABLES} -A allowed -p tcp -m tcp --tcp-flags SYN,RST,ACK SYN -j ACCEPT
${IPTABLES} -A allowed -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
${IPTABLES} -A allowed -p tcp -j DROP
```

```
#############################################################################
#
# Configuracion de MASQUERADING.
#
# Configuramos la regla necesaria para poder tener conectividad desde
# la intranet hacia internet de forma transparente
#

echo "  · Setting up MASQUERADING"

#${IPTABLES} -t nat -A POSTROUTING -s ${WIRELESS_IP_RANGE} \
#-o ${INET_IFACE} -j SNAT --to-source ${INET_IP}
${IPTABLES} -t nat -A POSTROUTING -s ${WIRELESS_IP_RANGE} \
-o ${INET_IFACE} -j SNAT --to-source ${INET_IP}


#############################################################################
#
# Configuracion de las reglas de filtrado.
#
# Configuracion de las reglas necesarias para poder ofrecer los servicios
# Configuracion para denegar cualquier paquete "ilegal"
#

#
# Configuramos el routing necesario para MASQUERADING
#

echo "  · Setting up FORWARDING"

${IPTABLES} -A FORWARD -p tcp -j bad_tcp_packets
${IPTABLES} -A FORWARD -s ${WIRELESS_IP_RANGE} -i ${WIRELESS_IFACE} -o ${INET_IFACE} -
j ACCEPT
${IPTABLES} -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

#
# Denegamos los paquetes TCP "ilegales"
#

echo "  · Setting up INPUT"

${IPTABLES} -A INPUT -p tcp -j bad_tcp_packets

#
# Permitimos cualquier cosa que entre por lo
#

${IPTABLES} -A INPUT -s ${LO_IP} -i ${LO_IFACE} -j ACCEPT
${IPTABLES} -A INPUT -s ${WIRELESS_IP} -i ${LO_IFACE} -j ACCEPT
${IPTABLES} -A INPUT -s ${INET_IP} -i ${LO_IFACE} -j ACCEPT

#
# Permitimos lo que entre desde el propio servidor
#

${IPTABLES} -A INPUT -s ${LO_IP} -i ${WIRELESS_IFACE} -j ACCEPT
${IPTABLES} -A INPUT -s ${WIRELESS_IP} -i ${WIRELESS_IFACE} -j ACCEPT
${IPTABLES} -A INPUT -s ${INET_IP} -i ${WIRELESS_IFACE} -j ACCEPT
${IPTABLES} -A INPUT -s ${LO_IP} -i ${INET_IFACE} -j ACCEPT
${IPTABLES} -A INPUT -s ${WIRELESS_IP} -i ${INET_IFACE} -j ACCEPT
${IPTABLES} -A INPUT -s ${INET_IP} -i ${INET_IFACE} -j ACCEPT

#
# Permitimos la entrada de conexiones establecidas
#

${IPTABLES} -A INPUT -d ${INET_IP} -i ${INET_IFACE} \
-m state --state RELATED,ESTABLISHED -j ACCEPT

#
# Configuramos los paquetes de salida de redes locales
# y de la propia maquina
#

echo "  · Setting up OUTPUT"
```

```
#
# Permitimos lo que salga hacia nuestra propia maquina
#

${IPTABLES} -A OUTPUT -p tcp -j bad_tcp_packets
${IPTABLES} -A OUTPUT -s ${LO_IP} -d ${LO_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -s ${LO_IP} -d ${WIRELESS_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -s ${LO_IP} -d ${INET_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -s ${WIRELESS_IP} -d ${LO_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -s ${WIRELESS_IP} -d ${WIRELESS_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -s ${WIRELESS_IP} -d ${INET_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -s ${INET_IP} -d ${LO_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -s ${INET_IP} -d ${WIRELESS_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -s ${INET_IP} -d ${INET_IP} -j ACCEPT

#
# Permitimos la salida de conexiones establecidas
#

${IPTABLES} -A OUTPUT -s ${INET_IP} -o ${INET_IFACE} -m state \
--state RELATED,ESTABLISHED -j ACCEPT


###############################################################################
#
# Reglas para permitir todo lo configurado.
#
# Se permiten las conexiones TCP configuradas
# Se permite el trafico UDP configurado
#

#
# Permitimos las conexiones configuradas
#

#${IPTABLES} -A INPUT -i ${INET_IFACE} -p tcp -j tcpincoming_packets
#${IPTABLES} -A OUTPUT -o ${INET_IFACE} -p tcp -j tcpoutgoing_packets
${IPTABLES} -A INPUT -p tcp -j tcpincoming_packets
${IPTABLES} -A OUTPUT -p tcp -j tcpoutgoing_packets

#
# Permitimos los paquetes UDP configurados
#

#${IPTABLES} -A INPUT -i ${INET_IFACE} -p udp -j udpincoming_packets
#${IPTABLES} -A OUTPUT -o ${INET_IFACE} -p udp -j udpoutgoing_packets
${IPTABLES} -A INPUT -p udp -j udpincoming_packets
${IPTABLES} -A OUTPUT -p udp -j udpoutgoing_packets


###############################################################################
#
# Reglas para permitir todos los servicios ofrecidos.
#
# Se permiten las conexiones TCP a servicios
# Se permite el trafico UDP a servicios
#

#
# Configuramos los servicios de la maquina
#

echo -n "  · Setting up services:"


echo -n " SSH"
# SSH
${IPTABLES} -A tcpincoming_packets -d ${INET_IP} -p tcp -m tcp \
--dport 22 -j allowed
${IPTABLES} -A tcpincoming_packets -d ${WIRELESS_IP} -p tcp -m tcp \
--dport 22 -j allowed

echo -n " APT-GET"
# APT-GET
${IPTABLES} -A tcpincoming_packets -s ${APT} -d ${INET_IP} -p tcp -m tcp \
--sport 80 --dport 1024:65535 -j allowed
```

```
${IPTABLES} -A tcpoutgoing_packets -s ${INET_IP} -d ${APT} -p tcp -m tcp \
--dport 80 --sport 1024:65535 -j allowed

echo -n " DNS"
# DNSin
${IPTABLES} -A udpincoming_packets -d ${INET_IP} -p udp -m udp --dport 53 \
-j ACCEPT
${IPTABLES} -A udpincoming_packets -d ${WIRELESS_IP} -p udp -m udp \
--dport 53 -j ACCEPT
# DNSout
${IPTABLES} -A udpoutgoing_packets -s ${INET_IP} -p udp -m udp --sport 53 \
-j ACCEPT
${IPTABLES} -A udpoutgoing_packets -s ${WIRELESS_IP} -p udp -m udp \
--sport 53 -j ACCEPT

#${IPTABLES} -A udpoutgoing_packets -s ${INET_IP} -p udp -m udp --dport 53 \
#-j ACCEPT

echo -n " NTP"
# NTPin
${IPTABLES} -A udpincoming_packets -s ${HORA} -d ${INET_IP} -p udp -m udp \
--sport 123 -j ACCEPT
# NTPout
${IPTABLES} -A udpoutgoing_packets -s ${INET_IP} -d ${HORA} -p udp -m udp \
--dport 123 -j ACCEPT

echo -n " DHCP"
# DHCPin
${IPTABLES} -A udpincoming_packets -p udp -m udp --dport 67 -j ACCEPT
#DHCPout
${IPTABLES} -A udpoutgoing_packets -s ${INET_IP} -p udp -m udp --sport 67 \
-j ACCEPT

echo -n " Conex_salientes"
# Conexiones salientes
${IPTABLES} -A tcpoutgoing_packets -p TCP -s ${INET_IP} -d ${INET} \
--sport 1024:65535 -j allowed
${IPTABLES} -A tcpoutgoing_packets -p TCP -s ${WIRELESS_IP} -d ${INET} \
--sport 1024:65535 -j allowed

echo


##############################################################################
#
# Configuracion de los modulos adicionales.
#
# Todos los siguientes modulos se encuentran en el directorio ${SCRIPTS}
# Para ejecutarlos o no, comentar o descomentar la linea pertinente
#

#
# Configuracion para las VPN
#

echo "  · Setting up VPN"

### ATENCION!!! Cambiar INTE_IP por WIRELESS_IP para probarlo con wifi!!! ###

# IPSEC ESP protocol
${IPTABLES} -A INPUT -p 50 -d ${WIRELESS_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -p 50 -s ${WIRELESS_IP} -j ACCEPT
# IPSEC AH protocol
${IPTABLES} -A INPUT -p 51 -d ${WIRELESS_IP} -j ACCEPT
${IPTABLES} -A OUTPUT -p 51 -s ${WIRELESS_IP} -j ACCEPT

# UDP
${IPTABLES} -A udpincoming_packets -p UDP -d ${WIRELESS_IP} --dport 500 \
-j ACCEPT
${IPTABLES} -A udpoutgoing_packets -p UDP -s ${WIRELESS_IP} --sport 500 \
-j ACCEPT


echo "  · Done"
```

# B. Análisis protocolo WEP

Todo el contenido de éste apéndice ha sido extraído de la página http://www.redlibre.net/wiki/moin.cgi/Vulnerabilidad_20WEP

## 1. Perspectiva de ataques

Antes de describir los ataques debemos discutir la viabilidad de llevarlos a la práctica. Una barrera común a los ataques en subsistemas de comunicación es el acceso a los datos transmitidos. A pesar de ser transmitido a través de ondas de radio abiertas, el tráfico 802.11 requiere una infraestructura significante para ser interceptado. Un agresor necesita equipo capaz de monitorizar frecuencias de 2.4GHz y entender la capa física del protocolo 802.11; para ataques activos, además es necesario estar en disposición de transmitir en las mismas frecuencias. Una inversión considerable que realizan los fabricantes en investigación y desarrollo va a parar en esta clase de dispositivos y herramientas.

Por este motivo podría existir la tentación de desestimar los ataques que requieran acceso a la capa física como impracticables; por ejemplo, esta actitud se estableció en las comunicaciones celulares hace tiempo. Sin embargo, tal posición es peligrosa. En primera instancia no nos protege contra agresores con alto nivel de recursos tecnológicos que tengan la capacidad de invertir tiempo y dinero en conseguir acceso a los datos. Esto resulta especialmente peligroso cuando aseguramos una red inalámbrica interna de una compañía, ya que el espionaje industrial es un negocio muy rentable.

El equipo necesario para monitorizar e inyectar tráfico 802.11 está disponible para los consumidores en forma de interfases inalámbricos Ethernet. Lo único necesario es alterarlos para monitorizar y transmitir tráfico cifrado. Es posible realizar ataques pasivos usando equipos de serie modificando la configuración de los controladores. Los ataques activos son más complejos, pero no por ello quedan fuera de nuestro alcance. Las tarjetas PCMCIA Orinoco fabricadas por Lucent permiten actualizar su firmware; un concentrado esfuerzo de ingeniería inversa podría permitir una versión modificada que permitiera la inserción de tráfico arbitrario. La inversión de tiempo requerida es importante; pero hay que tener en cuenta que es un esfuerzo de una única vez –ya concluido, el firmware alterado puede ser distribuido a través de un servidor web o por medio de círculos clandestinos-. Por este motivo consideramos que es prudente asumir que agresores motivados podrían disponer de acceso total a la capa física para ataques tanto pasivos como activos. Otros argumentos que justifican nuestra postura son los propios documentos WEP. En ellos se dice que “la escucha furtiva es un problema familiar para los usuarios de otros tipos de tecnología inalámbrica”. No profundizaremos más en las dificultades asociadas a los ataques dirigidos a la capa física del acceso. A partir de ahora centraremos nuestra atención en los ataques con propiedades criptográficas.

## 2. El riesgo de reutilización del "keystream"

WEP proporciona confidencialidad de datos por medio de un algoritmo de cifrado de flujo llamado RC4. Los cifrados de flujo funcionan expandiendo una clave secreta (o, como en el caso de WEP, una pública IV y una secreta) en una clave arbitrariamente larga de bits pseudo aleatorios (el *keystream*). El cifrado se lleva a efecto aplicando or-exclusivos al *texto plano*. Para descifrar se tiene que generar un *keystream* idéntico basado en IV y la clave secreta, para después aplicar de nuevo la función XOR sobre el texto cifrado.

Una debilidad bien conocida de los algoritmos de cifrado de flujo es que cifrando dos mensajes con la misma clave y vector `v` se puede revelar información sobre ambos mensajes:

Si `C1=P1^RC4(v,k)`
y `C2=P2^RC4(v,k)`

Entonces

```
C1^C2=(P1^RC4(v,k))^(P2^RC4(v,k))
     =P1^P2
```

En otras palabras, aplicando XOR a los dos textos cifrados (`C1` y `C2`) provocamos la cancelación del *keystream*, y el resultado que obtenemos es el or-exclusivo de ambos textos planos (`P1^P2`).

Por lo tanto, la reutilización del *keystream* puede llevar a un número de ataques: como caso especial, si el texto plano de uno de los mensajes es conocido, el texto plano del otro está disponible inmediatamente. Más generalmente, los textos planos habituales que se manejan a menudo tienen una redundancia suficiente como para permitir recuperar `P1` y `P2` dado sólo `P1^P2`; existen técnicas conocidas como, por ejemplo, resolver este problema buscando dos textos en inglés sobre los que, aplicados un XOR, resulten en el valor dado `P1^P2`. Es más, si tenemos *n* textos cifrados en los que se reutilice un mismo *keystream* tendremos lo que comúnmente se denomina un problema de profundidad *n*. Descifrar el tráfico en profundidad se facilita en tanto en cuando *n* aumente, ya que el resultado del XOR de cada par de textos planos puede ser calculado, y se conocen varias técnicas clásicas para resolver esta clase de problemas (por ejemplo el análisis de frecuencias, y demás).

Hágase notar que se requieren dos condiciones para que esta clase de ataque tenga éxito:

- La disponibilidad de textos cifrados en los que alguna porción del *keystream* sea utilizada más de una vez, y

- Un conocimiento parcial de parte del texto plano.

Para prevenir estos ataques, WEP utiliza un IV diferente por cada paquete para variar el proceso de generación del *keystream* para cada trama de datos transmitida. WEP genera el *keystream* `RC4(v,k)` como una función de la clave secreta `k` (que es la misma para todos los paquetes) y un vector de inicialización público `v` (que cambia para cada paquete); de este modo, cada paquete recibe un *keystream* diferente. El vector IV se incluye en la parte no cifrada de la transmisión, de modo que el receptor pueda saber qué IV utilizar para obtener el *keystream* necesario para la decodificación. IV está por tanto disponible también para los agresores, pero la clave secreta sigue siendo desconocida y mantiene la seguridad del *keystream*.

El uso de un IV diferente por cada paquete tiene la intención de prevenir ataques derivados de la reutilización reiterada de un mismo *keystream*. A pesar de todo, WEP no consigue alcanzar esta meta. Describiremos a continuación varios casos prácticos de ataques al WEP basados en la reutilización del *keystream*. Antes de nada, se comentará cómo localizar casos de reutilización de *keystream*; y se mostrará cómo aprovechar estos casos sacando ventaja de información parcial que se espera que contengan los textos planos más comunes.

**Localizar casos de reutilización del *keystream***. Una debilidad potencial de la reiteración del *keystream* puede producirse por una gestión inadecuada de IV. Puesto que por lo general la clave compartida `k` no cambia, la reutilización de IV's casi siempre provoca la reutilización de claves *keystream*. Ya que los IV's son públicos, el duplicado de IV's puede ser fácilmente detectado por los posible agresores. Por lo tanto, cualquier reciclado de valores de IV expone el sistema a ataques por reutilización de un mismo *keystream*. Nos referiremos a tales reiteraciones de valores de IV como *colisión*.

El estándar WEP recomienda (pero no requiere) que IV cambie en cada paquete. Sin embargo, no dice nada acerca de los mecanismos aconsejables para seleccionar IV's y, por esta razón, algunas implementaciones del sistema lo hacen precariamente. En concreto, las tarjetas PCMCIA examinadas reestablecen IV a 0 cada vez que se reinician, e incrementan IV en uno en cada paquete posterior. Estas tarjetas se reinician automáticamente cada vez que se introducen en un portátil, algo que se espera pase a menudo. En consecuencia, los *keystream* correspondientes a IV's de valor bajo son susceptibles de ser reutilizados muchas veces durante el tiempo de vida de la clave privada.

Peor aún, el estándar WEP tiene defectos de arquitectura que exponen a todas las implementaciones WEP –sin importar lo cuidadoso de su implementación- a riesgos serios de reutilización del *keystream*. El campo IV utilizado en WEP tiene una longitud de tan sólo 24 bits, prácticamente garantizando que se usará un mismo IV en múltiples mensajes. Un cálculo rápido muestra que un punto de acceso ocupado que transmita paquetes de 1500 bytes a una media de 5Mbps de ancho de banda (la velocidad máxima correspondería a 11Mbs) agotará todos los valores posibles de IV en menos de doce horas. Incluso en instalaciones con menor ocupación de canal, un agresor paciente puede encontrar duplicados fácilmente. Dado que la longitud de IV está predefinida en 24 bits, sin dependencia de otros parámetros, esta vulnerabilidad es inevitable.

Detalles en implementaciones concretas pueden provocar reutilizaciones del *keystream* más frecuentemente. Una implementación que utilizase un IV aleatorio para cada paquete produciría una *colisión* cada 5000 paquetes aproximadamente, que se resumen en tan sólo varios minutos de transmisión. Lo peor de todo es que el estándar 802.11 no exige que IV cambie en cada paquete, lo que podría permitir el uso de un IV idéntico en todos los paquetes sin que ello suponga una disconformidad con la norma estándar.

**Explotando la reutilización del *keystream* para leer tráfico cifrado.** Una vez localizados dos paquetes con el mismo IV se pueden aplicar varios métodos para recuperar el texto plano. Si el texto plano de uno de los mensajes es conocido resulta sencillo derivar directamente los contenidos del otro.

Hay muchas formas de obtener candidatos plausibles de texto plano. Muchos campos del tráfico IP son predecibles, ya que los protocolos utilizados usan estructuras de mensaje perfectamente conocidas con contenidos predecibles. Por ejemplo, las secuencias de entrada a sistemas son bastante uniformes para la mayor parte de los usuarios, y también lo son los contenidos –por ejemplo, la palabra `Password:` como mensaje de bienvenida-, que pueden ser utilizados para ataques a la clave. Otro ejemplo podría consistir en la posibilidad de reconocer por análisis de tramas de tráfico y longitud una librería compartida que estuviese siendo transferida en un sistema de red. Esto suministraría una gran cantidad de texto plano conocido que permitiría su utilización para realizar un ataque al *keystream* por reutilización.

Hay métodos más elaborados para obtener el texto plano. Por ejemplo, es posible provocar la transmisión de textos planos conocidos enviando tráfico directamente al terminal móvil desde un ordenador conectado a internet en manos del agresor. El agresor también puede enviar correo electrónico a usuarios y esperar que lo descarguen por medio del enlace inalámbrico. Enviar correo no solicitado (*spam*, en argot) puede ser un buen método para hacer esto sin levantar sospechas.

A veces obtener texto plano conocido puede ser incluso más sencillo. Un punto de acceso que probamos emitía paquetes broadcast de modo cifrado y no cifrado cuando la opción de controlar el acceso a la red estaba desactivada. En este caso, un agresor con una tarjeta 802.11 puede transmitir broadcasts al punto de acceso (que serán aceptados, porque el control de acceso está desactivado) y observar su forma cifrada durante la retransmisión. Es inevitable que esto suceda en una subred que contenga una mezcla de clientes WEP con otros sin soporte para cifrar, ya que los paquetes broadcast deben llegar a todos y cada uno de los clientes; no hay forma de evitar esta técnica para recoger texto plano conocido.

Como conclusión a este apartado recordamos al lector que, como se indicó con anterioridad, incluso sin conocer ningún texto plano todavía es posible analizar, por medio de suposiciones, posibles textos planos susceptibles de ser transmitidos que puedan desembocar en la obtención del la clave privada.

## 2.1. Diccionarios para descifrar

Una vez que se obtiene el texto plano de un mensaje interceptado el agresor puede aislar el valor del *keystream* utilizado para cifrar el mensaje, ya sea por análisis de IV's o por otros métodos. Es posible usar este *keystream* para descifrar cualquier otra trama que utilice un mismo IV. Según transcurre el tiempo, el agresor puede construir una tabla de *keystreams* que correspondan a distintas IV. La tabla completa requerirá poco espacio –unos 1500 bytes por cada una de las 224 IV posibles, o más o menos unos 24Gb- así que es concebible que un agresor dedicado pueda, después de un poco de esfuerzo, acumular datos suficientes como para construir todo un diccionario de decodificación, especialmente cuando consideramos que las claves sólo son cambiadas de forma ocasional. La ventaja para el agresor radica en que una vez que tiene la tabla disponible es posible descifrar inmediatamente cada texto cifrado con muy poco esfuerzo.

Desde luego, la cantidad de trabajo necesario para construir semejante diccionario restringe este ataque sólo a lo individuos más persistentes, que deseen emplear tiempo y dinero en vencer la seguridad WEP. Podría decirse que WEP no está diseñado para defenderse de tales ataques, ya que las claves de 40 bits se pueden descubrir fácilmente por medio de la fuerza bruta en una cantidad de tiempo relativamente corta con recursos moderados. A pesar de todo, los fabricantes ya han comenzado a extender WEP con soporte para claves más largas, aunque esto no implique cambio alguno en el tamaño efectivo del diccionario del agresor (el tamaño del diccionario no depende de la longitud de la clave, sino del tamaño de IV, que está prefijado en 24 bits).

Es más, el diccionario del agresor puede hacerse más práctico aprovechando el comportamiento de las tarjetas PCMCIA que reinician el vector IV a 0 cada vez que son reiniciadas. Puesto que en los casos más comunes las tarjetas son iniciadas al menos una vez al día, el agresor puede limitarse a construir un diccionario centrado sólo en los primeros miles de IV's, lo que le permitirá descifrar la mayoría de los paquetes que circulen a través del punto de acceso. En una red con numerosos clientes 802.11 las colisiones en los primeros miles de IV's serán abundantes.

## 2.2. Gestión de claves

El estándar 802.11 no especifica cómo llevar a cabo la distribución de claves. Depende de un mecanismo externo para poblar la matriz de cuatro claves compartida globalmente. Cada mensaje contiene un campo identificador de clave especificando el índice de la matriz que se utiliza para el cifrado. El estándar también permite asociar una clave específica de la matriz para cada estación móvil; sin embargo, está práctica no es habitual. La mayoría de las instalaciones usan una única clave para la toda red.

Esto perjudica severamente la seguridad del sistema, puesto que un secreto compartido por muchos no es fácil de ocultar. Algunos administradores de red intentan aliviar este problema no revelando la clave secreta a los usuarios, configurando ellos mismos cada terminal. Sin embargo esta forma de actuar sólo conduce a una mejora nimia, ya que las contraseñas siguen permaneciendo almacenadas en los terminales clientes. Como evidencia anecdótica, baste mencionar que conocemos a un grupo de estudiantes universitarios que obtuvieron la clave privada de la red simplemente para poder usar sistemas operativos no soportados por los administradores de red.

La reutilización de una clave única por muchos usuarios ayuda también a convertir los ataques en algo más práctico, porque aumenta la posibilidad de colisión de IV's. La posibilidad de una colisión casual aumenta proporcionalmente con número de usuarios, y si además tenemos en cuenta que las tarjetas PCMCIA establecen a 0 el vector IV cada vez que son reiniciadas todos los usuarios reutilizarán *keystreams* correspondientes a un pequeño rango de IV's. El hecho de que muchos usuarios compartan las mismas claves también significa que es difícil sustituir esta información, porque resulta comprometido ponerla en boca de todos. Además esto no será habitual puesto que cambiar una clave requiere que todos y cada uno de los usuarios reconfiguren su adaptador inalámbrico. En la práctica estimamos que puedan pasar meses, o incluso más tiempo, antes de que se cambien las claves privadas, lo que permite al potencial agresor disponer de una generosa cantidad de tiempo para buscar instancias de reutilización de *keystreams*.

## 2.3. Resumen

Los ataques descritos en esta sección demuestran que el uso de cifrados de flujo es peligroso porque la reutilización de *keystreams* puede tener consecuencias devastadoras. Cualquier protocolo que utilice un cifrado de flujo debe prestar especial cuidado en asegurar que el *keystream* nunca sea reutilizado.

Bajo estas premisas, un diseñador de protocolos debe prestar especial atención a las complicaciones que supone añadir cifrado de flujo en cualquier algoritmo de cifrado.

# 3. Autenticación de mensajes

El protocolo WEP utiliza el campo de *checksum* para verificar la integridad de los paquetes y que estos no sean modificados durante su tránsito. El *checksum* viene implementado como un CRC-32, el cual es parte de la carga cifrada del paquete.

A continuación argumentaremos que una suma CRC es insuficiente como para asegurar que un agresor no puede manipular un mensaje: No es un código de autenticación criptográficamente seguro. Los códigos CRC fueron diseñados para detectar errores aleatorios en mensajes; no son resistentes contra ataques maliciosos. Como demostraremos, esta vulnerabilidad del CRC esta exagerada todavía más por el hecho de que los datos del mensaje están cifrados usando un cifrado de flujo.

## 3.1. Modificación del mensaje.

En primera instancia, mostraremos que los mensajes pueden ser modificados en tránsito sin que el cambio sea detectado, violando las metas de seguridad.

Usaremos la siguiente propiedad del los *checksum* WEP:

**Propiedad 1** *El checksum WEP es una función lineal del mensaje.*

Con esto queremos decir que la suma de comprobación se distribuye sobre la operación XOR, por ejemplo, `c(x^y)=c(x)^c(y)` para todas las elecciones posibles de `x` e `y`. Esta propiedad es genérica para todas las sumas CRC.

Una consecuencia de la propiedad anterior es que se pueden hacer modificaciones controladas en un texto cifrado sin corromper el CRC. Fijemos nuestra atención en un hipotético texto cifrado `C` que hemos interceptado antes de que pudiera llegar a si destino:

```
A->(B):(v,C)
```

Asumimos que `C` corresponde a un mensaje desconocido `M`, así que:

```
C=RC4(v,k)^(M,c(M))
```

Mantenemos que es posible encontrar un nuevo texto cifrado `C'` que descifra a `M'`, donde:

```
M'=M^d
```

Y `d` puede ser elegida arbitrariamente por el agresor. Entonces, estaremos en disposición de sustituir la transmisión original con nuestro nuevo texto cifrado engañando al origen,

```
(A)->B:(v,C')
```

y en el descifrado, el receptor `B` obtendrá el mensaje `M'` con CRC correcto.

Sólo resta describir cómo obtener `C'` desde `C` de forma que `C'` descifre a `M'` en lugar de `M`. La observación clave es darse cuenta de que los cifrados de flujo, como el RC4, también son lineales, por lo que podemos reordenar los términos.

Sugerimos el truco siguiente: Haga un XOR de `(d,c(d))` en ambos lados de la primera ecuación que le mostramos arriba para obtener un nuevo texto cifrado `C'`:

```
C'=C^(d,c(d))
   =RC4(v,k)^(M,c(M))^(d,c(d))
   =RC4(v,k)^(M^d,c(M)^c(d))
   =RC4(v,k)^(M',c(M^d))
   =RC4(v,k)^(M',c(M'))
```

En esta derivación, usamos el hecho de que el *checksum* WEP es lineal, o sea, que `c(M)^c(d)=c(M^d)`. Como resultado, hemos mostrado como modificar `C` para obtener un nuevo texto cifrado `C'` que descifrará a `P^d`.

Esto implica que se pueden realizar cambios arbitrarios a un mensaje cifrado sin temor a ser detectado. Por este motivo el *checksum* WEP falla en la provisión de integridad en los datos, una de sus tres metas principales.

Nótese que este ataque puede aplicarse sin un conocimiento total de `M`: el agresor sólo necesita conocer el texto cifrado original `C` y la diferencia en texto plano deseada `d` para poder calcular `C'=C^(d,c(d))`. Por ejemplo, para invertir el primer bit de un mensaje, el agresor establecerá `d=1000…0`. Esto permitirá al agresor modificar un paquete solamente con un conocimiento parcial de su contenido.

## 3.2. Inyección de mensajes

Ahora demostraremos que WEP no proporciona un control de acceso seguro. Utilizaremos para ello la siguiente propiedad del *checksum* WEP:

**Propiedad 2** *El checksum WEP es una función sin cifrar del mensaje.*

Como consecuencia, el campo *checksum* también puede ser calculado por el adversario que conoce el mensaje.

Esta propiedad de la suma de integridad de WEP permite la evasión de las medidas de control de acceso. Si un atacante puede conseguir un texto plano correspondiente a alguna trama transmitida, estará en disposición de inyectar tráfico arbitrario en la red. Como vimos anteriormente, el conocimiento del texto plano y el texto cifrado de una misma trama revela el *keystream*. Este *keystream* puede ser reutilizado para crear nuevos paquetes, usando un mismo IV. Esto significa que si el agresor alguna vez consigue obtener el texto plano íntegro `P` de cualquier paquete cifrado `C`, puede resolver el *keystream* empleado para cifrar el paquete:

```
P^C = P^(P^RC4(v,k))=RC4(v,k)
```

Podría ahora construir la forma cifrada para un mensaje `M'`:

`(A)->B:(v,C')`

donde

`C'=(M',c(M'))^RC4(v,k)`

Nótese que el mensaje generado utiliza el mismo valor de IV que el mensaje original. Por tanto, el ataque funciona sólo porque:

**Propiedad 3** *Es posible reutilizar viejos valores de IV sin disparar ninguna alarma en el receptor.*

Cuando conocemos un IV junto con su correspondiente secuencia *keystream* `RC4(v,k)`, esta propiedad permite reutilizar el *keystream* conocido y burlar el mecanismo de control de acceso WEP.

Una defensa natural contra este ataque podría ser no permitir la reutilización de IV's en múltiples paquetes, requiriendo que todos los receptores se atengan a esta prohibición. Sin embargo, aunque el estándar 802.11 recomiende encarecidamente no utilizar un mismo IV, no obliga a ello. Por este motivo cada receptor debe aceptar IV's repetidos o arriesgarse a no ser compatible con dispositivos semejantes. Consideramos esto como un fallo del estándar 802.11.

En redes, uno oye a menudo la máxima "sea prudente con lo que envía y liberal con lo que acepte". Sin embargo, cuando la seguridad es la meta, esta guía puede ser muy peligrosa: ser liberal con el material que aceptamos significa que cada opción de baja seguridad ofrecida por el estándar ha de ser soportada por todos, y por extensión también disponible para el atacante. Esta situación es análoga a los ataques descubiertos para el cifrado SSL, los cuales también emplean el uso de las debilidades de un sistema con un sistema que incluía opciones de alta y baja seguridad. En consecuencia, sugerimos que el estándar 802.11 debería ser más específico respecto a la prohibición del reempleo de IV's y otros comportamientos potencialmente dañinos.

Este ataque no depende de la *Propiedad 1* del *checksum* WEP (linealidad). De hecho, sustituir cualquier función sin cifrar en lugar del CRC no tendrá efectos sobre la viabilidad del ataque. Sólo un mensaje con código cifrado de autenticación (MAC) tal como SHA1-HMAC proveerá suficiente fuerza como para prevenir este ataque.

## 3.3. Descifrado de los mensajes

Lo que puede resultar sorprendente es que la habilidad para modificar paquetes en el aire sin posibilidad de ser detectados nos lleva a descifrar paquetes en tránsito. Considere WEP desde el punto de vista del adversario. Ya que WEP utiliza un cifrado de flujo supuestamente seguro (RC4) atacar a la criptografía directamente carece de sentido. Pero aunque no podamos descifrar el tráfico por

nosotros mismos, todavía hay alguien que puede: el punto de acceso. En cualquier protocolo criptográfico el descifrador legítimo debe en todos los casos disponer la clave secreta para descifrar por definición. La idea es engañar al punto de acceso para que descifre algún texto cifrado por nosotros. Como se desprende de lo dicho anteriormente, la posibilidad de modificar paquetes transmitidos pone a nuestra disposición dos formas sencillas que explotar al punto de acceso en este sentido.

### Redirección IP

El primero recibe el nombre de Ataque de "redirección IP", y puede ser empleado cuando el punto de acceso WEP opera como un enrutador IP con conexión a Internet; algo habitual, porque WEP se emplea normalmente para suministrar acceso a usuarios móviles y otros.

En este caso, la idea es capturar un paquete cifrado del aire, y emplear la técnica de *modificación del mensaje* de modo que tenga una nueva dirección de destino: una que controle el agresor. El punto de acceso decodificará el mensaje y lo reenviará a su nuevo destino, donde el agresor podrá leer el paquete, ahora sin cifrar. Hágase notar que nuestro paquete modificado viajará desde la red inalámbrica hacia Internet, y por ese motivo la mayoría de los firewalls permitirán su tránsito sin ninguna traba.

El método más sencillo para modificar la IP de destino consiste en imaginar la dirección de destino original y entonces aplicar la técnica *modificación del mensaje* para cambiarla por otra de nuestro antojo. Hacerse una idea de la dirección de destino original suele ser sencillo; por ejemplo, todo el tráfico entrante tendrá como destino una IP correspondiente a la subred inalámbrica, la cual debería ser sencilla de determinar. Una vez que el tráfico entrante es descifrado, la dirección IP de los otros extremos se revela, y en tráfico saliente puede ser descifrado del mismo modo.

Para que este ataque funcione no sólo se debe modificar la dirección de destino, sino que además hay que asegurarse de que el *checksum* IP del paquete modificado sea correcto –de otra forma, el paquete sería rechazado por el punto de acceso–. Puesto que el paquete alterado difiere del original sólo en la dirección IP de destino, y ya que ambos valores de IP –original y modificado- se conocen, se puede calcular el *checksum* para después aplicarlo al paquete. Supongamos que las palabras de 16 bits más y menos significantes de la dirección IP de destino fueran $H$ y $L$, y que quisiéramos cambiarlas a $H'$ y $L'$. Si el antiguo *checksum* fuera $X$ (que no tiene por qué ser necesariamente conocido, ya que está cifrado), el nuevo sería

$$X' = X + H' + L' - H - L$$

Donde las sumas y restas corresponden a operaciones de complemento a uno.
El reto reside en que sólo sabemos cómo modificar un paquete aplicando un XOR, y que no estamos obligados a saber qué valor necesitamos aplicar a $X$ para

obtener X′, incluso a pesar de que sabemos lo que necesitamos sumar (o sea, H′+H′L−H−L).

Ahora se discutirán tres mecanismos para intentar corregir el *checksum* IP del paquete modificado:

**Conocemos el *checksum* IP original:** Si esto sucediera, entonces simplemente calcularíamos X′ según la fórmula anterior, y modificaremos el paquete aplicando un XOR en X^X′, lo que ajustaría el *checksum* IP al valor correcto para X′.

**Desconocemos el *checksum* IP original:** Si desconocemos X la labor se complica. Dado E=X′−X, necesitamos calcular d=X′^X.

De hecho, no hay información suficiente para calcular d dado sólo E. Por ejemplo, si E = 0xCAFE, podría ser que:

- X′=0xCAFE, X=0x0000, así que d=0xCAFE
- X′=0xD00D, X=0x050F, así que d=0xD502
- X′=0x1EE7, X=0x53E8, así que d=0x4D0F
- ...

Sin embargo, no todos los $2^{16}$ valores posibles son válidos para d, y algunos son mucho más probables que otros. En el ejemplo anterior tenemos cuatro valores para d (0x3501, 0x4B01, 0x4D01, 0x5501) que se dan menos del 3% de las veces. Es más, somos libres de hacer todos los intentos que queramos –cualquier asunción incorrecta será silenciosamente ignorada por el punto de acceso–. Dependiendo del valor de E, se puede hacer un pequeño número de intentos con alto porcentaje de éxito. Finalmente, el descifrado con éxito de un paquete puede ser empleado para fomentar el descifrado de otros; por ejemplo, el único campo que varía en la cabecera IP entre dos máquinas que mantenga un flujo de comunicación es el identificador de cabecera. Por esto, el conocimiento de la cabecera IP completa de un paquete puede ser utilizado para predecir la cabecera de paquetes circundantes, o para estrechar la búsqueda a un pequeño abanico de posibilidades.

**Conseguir que X = X′:** Otra posibilidad es compensar el cambio aplicado en el campo de destino por un cambio en otro campo, tal que el *checksum* para el paquete permanezca intacto. Cualquier campo de cabecera conocido que no afecte al envío del paquete es susceptible de ser útil, como por ejemplo el campo de IP de origen. Asumiendo que la dirección IP de origen de un paquete que queramos descifrar sea conocida (podemos obtenerla, por ejemplo, efectuando alguno de los ataques descritos anteriormente), basta con que sustraigamos E de los 16 bits menos significativos de la IP de origen, y tendremos un paquete con el mismo *checksum* que el original. Sin embargo, es posible que al modificar la dirección de origen de esta forma el paquete sea rechazado, basándose en reglas

de filtrado; pueden emplearse otros campos del encabezado para ajustar el *checksum*.

Agresores con recursos avanzados monitorizando una red de clase B completa podrían incluso realizar los ajustes necesarios sólo en el campo de destino, eligiendo `L'=H+L-H'`. Por ejemplo, si el destino original para un paquete es 10.20.30.40 y el agresor controla la subred 192.168.0.0/16 seleccionando la dirección 192.168.103.147 obtenemos un *checksum* idéntico para ambos paquetes, pero el paquete será enviado a una dirección de su control.

## Ataques de Reacción

Hay otro método para manipular un punto de acceso y romper la seguridad WEP que es aplicable siempre y cuando WEP sea empleado para proteger tráfico TCP/IP. Este ataque no requiere conectividad a Internet, así que podría aplicarse incluso cuando los ataques de redirección son imposibles. A pesar de todo, sólo es efectivo contra tráfico TCP; otros protocolos IP no pueden ser descifrados utilizando esta técnica.

En nuestro ataque, monitorizamos la reacción de un receptor de un paquete TCP y utilizamos nuestras observaciones para deducir información sobre el texto plano desconocido. Nuestro ataque descansa en el hecho de que un paquete TCP es aceptado sólo si el *checksum* TCP es correcto, y cuando es aceptado, se envía un paquete de confirmación como respuesta. Hágase notar que los paquetes de confirmación son fácilmente identificables por su tamaño, sin que requieran ser descifrados. Por este motivo, la reacción del receptor revelará cuándo el *checksum* es válido cada vez que un paquete es descifrado.

El ataque se efectúa según lo siguiente. Interceptamos un texto cifrado (`v, C`) con un texto plano desconocido `P`:

```
A->(B):(v,C)
```

Invertimos unos cuantos bits en `C` y ajustamos el CRC cifrado, obteniendo un nuevo texto cifrado `C'` con un *checksum* WEP válido. Construimos y transmitimos `C'` al punto de acceso:

```
(A)->B:(v,C')
```

Por último, observamos si el eventual receptor devuelve un paquete TCP ACK; esto nos descubrirá si el texto modificado pasó las comprobaciones *checksum* TCP y si fue aceptado por el receptor.

Téngase en cuenta que podemos elegir qué bits de `C` invertimos de la forma que queramos, usando técnicas de *modificación de mensaje*. El truco es el siguiente: Con una elección inteligente de los bits que invirtamos, podemos asegurar que el *checksum* TCP permanece inalterado exactamente cuando la condición $P_i \text{^} P_{i+16}=1$ se mantiene en el texto plano -condición de un sólo bit-. Por tanto, la presencia o ausencia de un paquete ACK revelará un bit de información del

texto plano P. Repitiendo el ataque para muchos valores de i, podemos obtener prácticamente todo el texto plano P, deduciendo las pocas variables que resten con técnicas clásicas.
Más adelante explicaremos cómo elegir qué bits alterar. Por ahora, los detalles no son demasiado importantes. Lo principal es que hemos explotado la buena voluntad del receptor para descifrar textos cifrados arbitrarios. La respuesta del receptor a nuestro paquete –ya sea aceptar o ignorar– puede ser vista como un canal secundario, semejante a aquellos explotados en ataques de consumo de potencia y capacidad de respuesta, que nos permite aprender sobre el texto plano. Utilizamos al receptor como oráculo que descifre el texto cifrado por nosotros. Esto se conoce como *ataque de reacción,* ya que funciona analizando las reacciones del receptor ante nuestras falsificaciones.

**Detalles técnicos.** Hasta ahora hemos aplazado la explicación de los detalles técnicos sobre cómo elegir nuevos paquetes C′ que engañen al receptor para ir resolviendo el texto plano P.

Recordemos que el *checksum* TCP es la suma en complemento a uno de las palabras de 16 bits del mensaje M. Es más, la suma en complemento a uno es equivalente más o menos a la suma módulo $2^{16}$–1. De aquí se desprende que, hablando aproximadamente, el *checksum* TCP de un texto plano P es válido sólo cuando P equivale a $0 \bmod 2^{16}-1$.

Definimos C′=C^d, de modo que d especifique qué posición de bit invertir, eligiendo d como sigue: se escoge una i arbitrariamente, se ponen los bits i e i+16 en d a uno, y cero en las demás posiciones. La propiedad clave de la suma módulo de $2^{16}$–1 es que P^d es equivalente a $P \bmod 2^{16}-1$ mientras $P_i \text{^} P_{i+16}=1$. Puesto que asumimos que el *checksum* TCP es válido para el paquete original (P equivale a $0 \bmod 2^{16}-1$), esto significa que el *checksum* TCP será válido para el nuevo paquete (por ejemplo, $P \text{^} d = 0 \bmod 2^{16}-1$) justamente cuando $P_i \text{^} P_{i+16} = 1$. Este mecanismo ofrece un bit de información del texto plano, como antes comentamos.

## 3.4. Resumen

En esta sección, se ha mostrado la importancia de utilizar códigos de autenticación de mensajes seguros, como SHA1-HMAC, para proteger la integridad de las transmisiones. El uso de CRC es inapropiado para este propósito, y de hecho cualquier función sin codificar falla a la hora de defenderse contra los ataques descritos en esta sección. Un MAC seguro es particularmente importante a la hora de diseñar protocolos, ya que la pérdida de integridad en los mensajes en una capa del sistema puede llevar a romper el secreto en el sistema global.

# C. Generación de certificados SSL

## 1. Certificado autorizador

Este certificado será el que firmará el resto (clientes y servidores) para darles validez. Es un certificado que se deberá distribuir a todos los clientes y servidores de una forma segura para que puedan comprobar la validez de los respectivos certificados.

Script de generación

```
#!/bin/sh

# Variables por defecto para sistema Debian
SSL=/usr
export PATH=${SSL}/bin/:/usr/lib/ssl/misc/:${PATH}
export LD_LIBRARY_PATH=${SSL}/lib

rm -rf demoCA
echo "**********************************************************"
echo "Creando clave privada y certificado autofirmado"
echo "Cuando se pregunte, sobrescribir el valor del campo CN"
echo "**********************************************************"
echo
openssl req -new -x509 -keyout newreq.pem -out newreq.pem \
-passin pass:secret -passout pass:secret

echo "**********************************************************"
echo "Creando una nueva jerarquía CA (utilizada después por el”
echo "comando CA) con el certificado y clave privada anterior"
echo "**********************************************************"
echo
echo "newreq.pem" | CA.pl -newca >/dev/null

echo "**********************************************************"
echo "Creando la raiz CA"
echo "**********************************************************"
echo
openssl pkcs12 -export -in demoCA/cacert.pem -inkey newreq.pem \
-out root.p12 -cacerts -passin pass:secret -passout pass:secret
openssl pkcs12 -in root.p12 -out root.pem -passin pass:secret \
-passout pass:secret
openssl x509 -inform PEM -outform DER -in root.pem -out root.der
rm -rf newreq.pem
```

Ahora se dispone del fichero `root.pem`, que contiene la clave privada y el certificado CA. Se guarda bien guardado ya que toda la seguridad estará basada en certificados firmados con este. En caso de robo o pérdida, se debería generar uno nuevo y revocar todos los generados con el antiguo.

Para la creación de los siguientes certificados, deberemos estar situados en el mismo directorio que cuando ejecutamos este script.

El fichero `root.pem` generado será el que se utilizará para certificar las claves de los clientes y servidor. Consta de dos partes: clave privada y certificado.

```
...
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
...
-----BEGIN RSA PRIVATE KEY-----
...
-----END RSA PRIVATE KEY-----
```

A los clientes únicamente se les tiene que proporcionar la parte del certificado, ya que si se pasara también la clave privada serían capaces de generarse ellos mismos certificados válidos. Para ello, tan solo tenemos que borrar todo lo que hay detrás de la línea:

```
-----END CERTIFICATE-----
```

Y guardarlo como un fichero nuevo que será el que suministremos a todos los clientes y servidores.

## 2. Certificado del servidor

Éste será el utilizado por el servidor, y también deberá tenerse mucho cuidado para que nadie desautorizado pueda acceder a él, ya que se perdería toda la seguridad de la red.

Script para la generación del certificado (recibe como parámetro el nombre de salida del certificado), cuando se pida el valor del campo CN, se deberá introducir el nombre DNS del servidor en el que se haga uso de él:

```
#!/bin/sh

# Variables por defecto para sistema Debian
SSL=/usr
export PATH=${SSL}/bin/:/usr/lib/ssl/misc/:${PATH}
export LD_LIBRARY_PATH=${SSL}/lib

echo "******************************************************"
echo "Creando la clave privada y certificado para el servidor"
echo "Cuando se pregunte, introduce el nombre de la maquina"
echo "en el campo Common Name"
echo "******************************************************"
echo
openssl req -new -keyout newreq.pem -out newreq.pem \
-passin pass:secret -passout pass:secret
openssl ca -policy policy_anything -out newcert.pem \
-passin pass:secret -key secret -infiles newreq.pem
openssl pkcs12 -export -in newcert.pem -inkey newreq.pem \
-out $1.p12 -clcerts -passin pass:secret -passout pass:secret
openssl pkcs12 -in $1.p12 -out $1.pem -passin pass:secret \
-passout pass:secret
openssl x509 -inform PEM -outform DER -in $1.pem -out $1.der
```

```
rm -rf newert.pem newreq.pem
```

El fichero con extensión `pem` será el que se tenga que poner en el servidor como clave privada y certificado. Contiene la misma estructura que el root.pem visto anteriormente. En este caso no necesitamos separarlo, ya que utilizaremos el mismo fichero como llave privada y certificado.

## 3. Certificado del cliente

Se tendrá que generar uno para cada cliente que se quiera, y el campo CN del certificado deberá coincidir con el nombre del usuario, en caso contrario se le denegará el acceso.

El script recibe como parámetro el nombre del fichero de salida:

```
#!/bin/sh

# Variables por defecto para sistema Debian
SSL=/usr
export PATH=${SSL}/bin/:/usr/lib/ssl/misc/:${PATH}
export LD_LIBRARY_PATH=${SSL}/lib

echo "******************************************************"
echo "Creando clave privada y certificado para el cliente"
echo "Cuando se pregunte por el Common Name introduce el"
echo "nombre del usuario. Es el mismo usuario que el utilizado"
echo "en el nombre de usuario de FreeRADIUS"
echo "******************************************************"
echo
openssl req -new -keyout newreq.pem -out newreq.pem \
-passin pass:secret -passout pass:secret
openssl ca -policy policy_anything -out newcert.pem \
-passin pass:secret -key secret -infiles newreq.pem
openssl pkcs12 -export -in newcert.pem -inkey newreq.pem \
-out $1.p12 -clcerts -passin pass:secret -passout pass:secret
openssl pkcs12 -in $1.p12 -out $1.pem -passin pass:secret \
-passout pass:secret
openssl x509 -inform PEM -outform DER -in $1.pem -out $1.der
rm -rf newcert newreq.pem
```

El fichero con extensión `pem` será el que se tenga que pasar al cliente como clave privada y certificado. Contiene la misma estructura que el root.pem visto anteriormente. En este caso no necesitamos separarlo, ya que utilizaremos el mismo fichero como llave privada y certificado.

# GLOSARIO

Para poder entender la forma de implementar mejor la seguridad en una red wireless, es necesario comprender primero ciertos elementos:

- **AAA:** Abreviatura de *Authentication*, *Authorization* y *Accounting*, sistema en redes IP para saber a qué recursos informáticos tiene acceso el usuario y registrar la actividad del usuario en la red.

  - **Autenticación** es el proceso de identificación de un individuo, normalmente mediante un nombre de usuario y contraseña. Se basa en la idea de que cada individuo tendrá una información única que le identifique o que le distinga de otros.

  - **Autorización** es el proceso de aceptar o denegar el acceso de un usuario a los recursos de la red una vez éste ha sido autenticado con éxito. La cantidad de datos y servicios a los que el usuario podrá acceder dependen del nivel de autorización que tenga establecido.

  - **Accounting** es el proceso de rastrear la actividad del usuario mientras accede a los recursos de la red, incluso la cantidad de tiempo que permanece conectado, los servicios a los que accede y los datos transferidos durante la sesión. Los datos registrados durante este proceso se utilizan con fines estadísticos, de planificación de capacidad, facturación, auditoria y reparto de costes.

  A menudo los servicios AAA requieren un servidor dedicado. RADIUS es un ejemplo de un servicio AAA.

- **ACL**. *Access Control List*, lista asociada a un objeto que identifica las entidades que pueden acceder al objeto y sus derechos de acceso. Por ejemplo, una lista asociada a un archivo que identifica los usuarios que pueden acceder a él, y sus derechos de acceso. También puede ser una lista la que identifique las entidades que no pueden acceder al objeto.

- **CA:** *Certification Authority* o Entidad emisora de certificados. Organización que emite certificados. La entidad emisora de certificados autentica la identidad del propietario del certificado y el servicio que el propietario está autorizado a utilizar, emite nuevos certificados, renueva certificados existentes y revoca certificados que pertenecen a usuarios que ya no están autorizados a utilizarlos.

- **CNAC**. *Closed Network Access Control*. Impide que los dispositivos que quieran unirse a la red lo hagan si no conocen previamente el SSID de la misma.

- **DS**: *Distributed System* o Sistema Distribuido, y la única diferencia con WDS es que los enlaces entre los puntos de acceso se realizarían mediante una red cableada, evitando así las interferencias que se causan con WDS.

- **IEEE:** *Institute of Electrical and Electronics Engineers*, organismo internacional encargado hoy en día de la promulgación de estándares para redes de comunicaciones.

- **IPSEC:** *Internet Protocol SECurity*, funciones de seguridad implementadas en el nivel de red de la pila de protocolos (autenticación y cifrado). Es opcional para IPv4 y obligatorio para IPv6. Éste es el estándar que FreeS/WAN implementa.

- **OSA:** *Open System Authentication*, cualquier interlocutor es válido para establecer una comunicación con el Punto de Acceso, con lo que si no se requiere validación mediante el protocolo EAP, se permitirá unirse a la red sin ninguna restricción, en caso contrario, se requerirá autenticación previa para poder pertenecer a la red wireless.

- **PKI:** *Public Key Infrastructure*, es un sistema para verificar la autenticidad de cada interlocutor implicado en una transacción de Internet, ofrecer protección frente al fraude o al sabotaje y evitar el rechazo de las transacciones con el fin de ayudar a los consumidores y a los minoristas a protegerse frente a la denegación de las transacciones. Unas organizaciones de terceros de confianza, llamadas autoridades certificadoras, emiten certificados digitales (que son archivos adjuntos a los mensajes electrónicos) que especifican componentes clave de la identidad del usuario. Durante una transacción en Internet, los mensajes cifrados y firmados se dirigen automáticamente a la autoridad de certificación, donde se verifican los certificados para que la transacción pueda continuar. La PKI puede estar incorporada en aplicaciones de software o bien ofrecerse como servicio o como producto. Los líderes de e-business están de acuerdo en que las PKI son cruciales para la seguridad y la integridad de las transacciones, y el sector del software está adoptando estándares abiertos para su uso.

- **QoS:** *Quality of Service*, es la capacidad de una red IP para garantizar servicio ininterrumpido, cuando sea necesario, para aplicaciones que requieren un gran ancho de banda como puedan ser las videoconferencias.

- **SKA:** *Shared Key Authentication*, es el método mediante el cual ambos dispositivos disponen de una misma clave de cifrado, entonces, el dispositivo Terminal de Red pide al Punto de Acceso autenticarse. El Punto de Acceso le envía una trama al Terminal de Red, que si éste a su vez devuelve correctamente cifrada, le permite asociarse con él.

- **SSID**. *Service Set IDentifier*, y es una cadena de 32 caracteres máximo que identifica a cada red inalámbrica. Los Terminales de Red deben conocer el nombre de la red para poder unirse a ella.

- **TKIP:** *Temporal Key Integrity Protocol*. Al contrario que WEP, utiliza claves de sesión dinámicas de 128 bits, para cada usuario, cada sesión y cada paquete. Los usuarios deben acceder a través de un servidor de autenticación, típicamente un RADIUS. Una vez autenticados mutuamente, el servidor genera una clave "master" que transmite de manera segura al cliente y que será utilizada para enviar el resto de claves auxiliares que serán utilizadas durante esa sesión.

- **VPN**: *Virtual Private Network* o Red Privada Virtual, y consiste en formar un "túnel" entre dos maquinas diferentes para obtener una seguridad extra en cuanto al transporte de los datos. Concretamente se gana en confidencialidad de datos, ya que todo el tráfico que circule por dentro del túnel se cifrará automáticamente al "entrar" y se descifrará al "salir" de dicho túnel. De esta forma se puede establecer un túnel entre dos routers de dos redes diferentes y todo el tráfico que circule de una red a otra, será cifrado transparentemente al usuario a nivel 3 (capa de red, IP).

- **WDS**: *Wireless Distributed System*, lo que quiere decir que mediante varios puntos de acceso se forma una única red wireless. Los puntos de acceso establecen entre ellos enlaces inalámbricos para intercambiarse información acerca de qué clientes tiene cada uno conectados a ellos. Gracias a esto, dos clientes conectados a puntos de acceso diferentes se pueden llegar a conectar el uno con el otro de forma transparente, como si lo hicieran con un cliente que estuviera conectado al mismo punto de acceso.

- **WEP**. *Wired Equivalent Privacy*, y fue introducido para intentar asegurar la autenticación, protección de las tramas y confidencialidad en la comunicación entre los dispositivos inalámbricos. Puede ser WEP64 (40 bits reales), WEP128 (104 bits reales) y algunas marcas están introduciendo el WEP256. Es INSEGURO debido a su arquitectura, por lo que el aumentar los tamaños de las claves de cifrado sólo aumenta el tiempo necesario para romperlo.

- **WPA.** *Wi-Fi Protected Area***,** y es el sustituto de WEP debido a las debilidades que presenta este último. Fue aprobado en abril del 2003, desarrollado para mejorar las características de seguridad del estándar WEP y permitir su implementación en productos inalámbricos que actualmente soportan WEP, pero la tecnología incluye dos mejoras con respecto a este último: emplea el protocolo de integridad de claves TKIP y la autenticación de usuarios se realiza mediante el protocolo EAP.

# BIBLIOGRAFIA


**Introducción**

- http://www.mailxmail.com/curso/informatica/wifi [visitado 25-03-2004]

**Conceptos básicos**

- http://www.monografias.com/trabajos14/segur-wlan/segur-wlan.shtml [visitado 25-03-2004]
- Edd Dumbill, Brian Jepson y Roger Weeks, *Linux Unwired*, O'Reilly, 2004

**Servicio DNS**

- Paul Albitz y Cricket Liu, *DNS and BIND 4th Edition*, O'Reilly, 2001

**Servicio SSH**

- Daniel J. Barrett y Richard E. Silverman, *SSH, The Secure Shell: The Definitive Guide*, O'Reilly, 2001

**Servicio firewall**

- Elizabeth D. Zwicky, Simon Cooper y D. Brent Chapman, *Building Internet Firewalls*, O'Reilly, 2000

**Funcionamiento WEP**

- http://www.redlibre.net/wiki/moin.cgi/Vulnerabilidad_20WEP [visitado 12-04-2004]

**Funcionamiento WPA**

- http://wiki.madridwireless.net/WPA [visitado 29-05-2004]
- Francisco García López, Revista *Antena de Telecomunicación*, Págs. 24-27, Diciembre 2003

**Servicio RADIUS**

- Jonathan Hassell, *RADIUS*, O'Reilly, 2002
- *RADIUS for UNIX Administrator's Guide*, Lucent Technologies, 1999
- http://rbirri.9online.fr/howto/Freeradius_+_TTLS.html [visitado 11-06-2004]

**Protocolo 802.11**

- Matthew Gast, *802.11 Wireless Networks, The Definitive Guide*, O'Reilly, 2002

**VPNs**

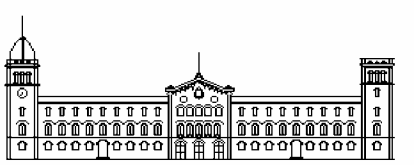
- Charlie Scott, Paul Wolfe y Mike Erwin, *Virtual Private Networks*, O'Reilly, 1999

**Actualizando firmware prism2/2.5/3**

- http://linux.junsun.net/intersil-prism/ [visitado 17-04-2004]

**Glosario**

- http://www.virusprot.com/Glosarioc.html [visitado 1-07-2004]
- http://www.liderazgoymercadeo.com/glos_buscar.asp [visitado 19-05-2004]
- http://www.eveliux.com/articulos/stds.html [visitado 20-04-2004]
- http://publib.boulder.ibm.com/tividd/glossary/from_es/html/TivoliGlossary04.htm [visitado 27-06-2004]



**ENGINYERIA TÈCNICA EN INFORMÀTICA DE SISTEMES**

**UNIVERSITAT DE BARCELONA**

Treball fi de carrera presentat el dia de de 200
a la Facultat de Matemàtiques de la Universitat de Barcelona,
amb el següent tribunal:

Dr. President

Dr. Vocal 1

Dr. Secretari

Amb la qualificació de: