AUDITORIA REDES INALÁMBRICAS

NOMBRE	APELLIDOS	DNI
Carlos	Ortega Marchamalo	03208832X
Pablo	Collado Soto	03227949Z

PUESTO: 4 GRUPO: 3ºD

Cada pareja de alumnos tiene que rellenar este documento y entregarlo al finalizar la sesión de laboratorio al profesor.

1. Indique cuál es el tamaño teórico de dicho diccionario compuesto por todas las posibles claves de hasta 9 caracteres. Tenga en cuenta que el tamaño mínimo de la clave en PSK es de 8 caracteres

Teniendo en cuenta que existen 62 caracteres alfanuméricos (sin contar la ñ) tendremos *62^8 + 62^9 = 1,376e16*. Si nos anticipamos y ejecutamos la orden del siguiente enunciado veremos que se generarán exactamente *13755426651848448* posibles contraseñas.

2. Genere el diccionario usando la herramienta *crunch* mediante la siguiente orden:
   `crunch 8 9 –f /usr/share/crunch/charset.lst mixalpha-numeric`
   Indique el tamaño del diccionario resultado en bytes y número de palabras. Detenga la generación.

De acuerdo con la salida del comando el resultado tendría *137335926412899584 B* de tamaño lo que equivale a *121 PB*, una cantidad ingente de información. Tal y como adelantábamos antes esa cantidad de información la integran *13755426651848448 palabras*.

3. Estime el tiempo medio necesario para determinar la clave teniendo en cuenta el tamaño del diccionario y la velocidad de verificación (que podemos obtener ejecutando *pyrit benchmark*)

La salida de **`pyrit benchmark`** nos muestra que nuestra máquina puede calcular *2727.4 PMKs/s*. Para comprender la medida debemos estudiar el *handshake* en 4 pasos que forma parte del proceso de autenticación de WPA/WPA2. Parte del proceso necesita generar una clave compartida por pares, la **PMK**, que se obtiene aplicando la función PBKDF2 sobre la tupla SSID + PSK. El benchmark por tanto nos dice cómo de rápido podemos generar estas PMKs a partir del SSID que introduciremos en Pyrit y de las contraseñas que vayamos leyendo del diccionario.

Por tanto, si el diccionario tiene *13755426651848448* palabras y las podemos probar a razón de *2727.4 palabras/s* tardaremos *5.04e12 s*, lo que equivale a prácticamente *159926 años* de trabajo ininterrumpido.

4. Genere este nuevo diccionario usando la herramienta ***crunch***. Ten en cuenta que la forma de indicar a ***crunch*** que en una determinada posición hay un número es mediante el carácter ***%***. Escriba la orden utilizada e indique el tamaño del diccionario resultando en bytes y número de palabras.

Para generar el diccionario debemos redireccionar la salida de crunch a la terminal (es decir, a stdout) para que salga a un archivo. Podemos lograrlo con el operador de redirección de la shell indicando el nombre del archivo de salida. Lo que es más, debemos especificar el tamaño máximo y mínimo aún teniendo empleando una plantilla que rellenar. Para definir esta plantilla empleamos la opción -t. Con todo al final llegamos a que para generar el diccionario dict.txt en el directorio de trabajo debemos ejecutar **`crunch 9 9 -t 918%%%%%% > dict.txt`**. Al hacerlo veremos que nuestro diccionario tiene un tamaño de *10000000 B* aportado por *1000000 palabras*. Sabiendo que cada carácter ocupa 1B puede sorprendernos que el tamaño sea de *10000000 B* en vez de *9000000 B* ya que cada número de teléfono tiene 9 dígitos. No debemos olvidarnos del carácter de nueva línea (‘*n*’) que se añade al final de cada entrada del diccionario. Con ello cada línea pesará *10 B* dándonos por tanto un tamaño total de *10000000 B* tal y como observamos.

5. Realice el ataque para obtener la clave. Indique las órdenes que ha usado para preparar pyrit y obtener la clave.

Para auditar la red se nos proporciona una captura de tráfico de varios *handshakes* pero solo debemos analizar el de la red cuyo ESSID es *WLAN_3C5A*. Si ejecutamos **`pyrit -r captura3C5A.cap analyze`** veremos cómo encontramos varios puntos de acceso. Por ello optamos por invocar **`pyrit -r captura3C5A.cap -e WLAN_3C5A -o wlan_3c5a.pcap`** strip lo que generará el archivo *wlan_3c5a.cap* con los mensajes de interés para nosotros. Como solo existe tráfico de 1 red dentro de este nuevo archivo nos basta con ejecutar time **`pyrit -r wlan_3c5a.pcap -i dict.txt attack_passthrough`** para intentar encontrar la contraseña con el diccionario que habíamos generado antes. Nótese que no incluimos la opción *-e* porque no existe otro punto de acceso con lo que conseguimos reducir la extensión del comando en un intento de aumentar su legibilidad. Asimismo lo hemos ejecutado precedido de la orden time para obtener el tiempo de ejecución real que ha sido de *5 minutos y 15,059 segundos*. En el siguiente ejercicio veremos si ésto concuerda con lo que cabría esperar.

6. ¿Cuál es la clave obtenida?.Indique el tiempo (aproximado) que ha tardado en obtener la captura. ¿Se corresponde con su estimación?

Al ejecutar el ataque descrito en la pregunta anterior hemos visto cómo la clave de la red es 918856501. También comentábamos que el programa había tardado *5 minutos y 15,059 segundos* en obtenerlo. Teniendo en cuenta los *2727.4 PMKs/s* que nuestro equipo era capaz de calcular y las 10^6 contraseñas totales el tiempo que habríamos esperado tardar es de *(10^6 / 2) / 2727.4 = 3,06 minutos*. Teniendo en cuenta que *a priori* todas las contraseñas son equiprobables podemos pensar que en media solo deberemos recorrer la mitad del diccionario hasta encontrar la clave, de ahí el factor de 2 por el que dividimos. No obstante nuestra clave se encontraba en la parte final del diccionario pues al estar ordenado y ser los dígitos tras el prefijo conocido 8 y 8 vemos que este tiempo se desvía de la media. Podemos comprobarlo si asumimos que para encontrar la contraseña si ésta se encuentra en el rango 91888XXXX nos llevará un tiempo aproximado de *(10^6 / 10) * 8,8 / 2727.4 = 5,38 minutos*, algo muy próximo al tiempo obtenido.

En definitiva, vemos que en este caso el tiempo es superior al estimado a priori debido a que la contraseña se encontraba hacia la parte final de nuestro diccionario.

7. Para continuar con la herramienta Crunch, genere los siguientes diccionarios e indique las órdenes que usado para cada uno de ellos:
    a. Diccionario de palabras de 9 caracteres con la primera letra en mayúsculas, resto en minúsculas y que el 9º carácter sea un símbolo.
    b. Diccionario que contenga todas las posibles combinaciones de estas tres palabras: Casa, Juan y Dados.
    c. Diccionario de teléfonos móviles (9 números) que empiecen por 60912(patrón 60912NNNN).
    d. Diccionario de palabras de 8 letras, que contengan **admin** en el medio (patrón XXadminX)

I. **`crunch 9 9 -t ,@@@@@@@^`**

II. **`crunch 1 1 -p Casa Juan Dados`** (Se deben especificar dos valores de longitud (1, 1) aunque se ignoren)

III. **`crunch 9 9 -t 60912%%%%`**

IV. **`crunch 8 8 -t @@admin@`** (Al no especificarlo se asumen letras minúsculas teniendo en cuenta que son las más comunes para nombres de usuario. En sistemas UNIX los nombres solo permiten minúsculas por defecto)

8. Si observamos los parámetros de dichas redes, vemos que una emplea CCMP (WPA2) y otra TKIP (WPA). ¿Influye el uso de CCMP o TKIP en la resistencia de la red inalámbrica ante un ataque de diccionario? Justifique su respuesta.

No, pues siempre es posible cambiar la contraseña del Wi-Fi por una más simple la cual sería descifrable a través de un ataque de diccionario, por lo que estos protocolos no suponen una barrera ante estas actuaciones.

Tanto TKIP como CCMP son algoritmos de encriptación que se basan en el algoritmo de cifrado de flujo RC4 y en el de cifrado de bloques AES, respectivamente. Nuestro ataque sin embargo no trata de desencriptar tráfico de ninguna manera. Nosotros queremos obtener la clave de conexión de la red para lo que primero debemos calcular todas las posibles PMKs. Esta PMK sirve como entrada para calcular la PTK. Esta PTK es una entrada a un hash que genera el MIC que se encuentra en la respuesta de vuelta al punto de acceso. Vemos cómo TKIP y CCMP se emplean para calcular este MIC y para cifrar posteriormente todo el tráfico que se curse por el enlace inalámbrico. No obstante, no atacamos este tráfico encriptado. Tan solo debemos saber qué algoritmo se está empleando para poder comprobar si nuestras pruebas son un éxito o no pero esto no afecta para nada a nuestro vector de ataque. Prueba de esto es que se pueden obtener las contraseñas de sendas redes con el mismo mecanismo.

9. A partir de la información que tenemos sobre este caso, genere el diccionario usando ***crunch***. Indique la orden utilizada, y el tamaño del diccionario en palabras y bytes.

Sabiendo que el diccionario se compone de contraseñas que siguen la plantilla "AUT12XXXX" donde X denota un caracter alfabético podemos generar la lista de palabras deseada con **`crunch 9 9 -t AUT12@@@@ > aut_dict.txt`**. Esto nos muestra que el diccionario tendrá *456976 palabras* que suponen un total de *4569760 bytes*. Al ser palabras de 9 caracteres estamos en el mismo caso que el comentado en el ejercicio 4.

10. Estime el tiempo medio necesario para obtener la clave para ambas redes usando ***pyrit***:

Si obviamos que ambas redes tengan el mismo SSID deberíamos probar el diccionario con cada una de las 2. Tal y como discutíamos antes podemos esperar que para 456976 palabras y una tasa de *2727.4 PMKs/s* encontremos la clave tras *(456976 / 2) / 2727.4 = 1.4 minutos*. Ya que deberíamos repetir el proceso para la segunda red tardaríamos un total de *2.8 minutos* en encontrar ambas claves en media.

11. De acuerdo con la estimación anterior, en condiciones normales no podría obtener las claves para las dos redes inalámbricas antes del final de la sesión de laboratorio siguiendo el mismo procedimiento del apartado anterior. ¿Cómo podríamos aprovechar el hecho de que los SSIDs son iguales para ambas redes?

Para poder valorar este hecho de cara a nuestra situación debemos saber qué es lo que estamos calculando en realidad con el ataque. Nosotros queremos obtener la contraseña de la red. Tal y como se observa en la página 166 de las mejoras al estándar 802.11 del IEEE la clave compartida PSK, que es la misma que la PMK (pág 17) se deriva a partir de la clave que empleamos para configurar la red y el SSID, es decir, el nombre de la red. Este SSID es "AUTOMATICA" para ambas redes. Destacamos que la distinción entre ESSID y SSID no se especifica en el propio estándar del IEEE y suele ser idéntica.

Por lo tanto, al tener ambas redes el mismo SSID solo tenemos que calcular las PMKs un vez y las podemos reutilizar para ambos puntos de acceso. Para poder sacar provecho de esta situación utilizaremos la base de datos de pyrit en vez de calcular las PMKs "al vuelo" como veníamos haciendo. Con este método las PMKs que empleemos se quedarán almacenadas y se podrán reutilizar para ataques posteriores. En el siguiente ejercicio detallamos cómo llevar a cabo esta operación.

12. Anote todas las órdenes necesarias para llevar a cabo el ataque:

Comenzamos por importar las contraseñas a la base de datos. Teniendo en cuenta que el diccionario generado es **`aut_dict.txt`** ejecutamos **`pyrit -i aut_dict.txt import_passwords`**. Depués solo debemos ejecutar **`pyrit -r AUT01.cap -e AUTOMATICA attack_batch`**. Esta orden reutilizará todas las PMKs ya calculadas o las obtendrá en caso de que no existan. En este primer caso tendremos que calcular todas las PMKs hasta encontrar la contraseña válida pero éstas ya estarán listas para la siguiente orden.

Ahora con solo ejecutar **`pyrit -r AUT02.cap -e AUTOMATICA attack_batch`** veremos que se obtiene la clave de forma casi instantánea porque resulta que ya se había calculado para el caso anterior pero no había resultado válida.

Con esto mostramos que pyrit puede también trabajar en un modo con persistencia que nos permite reutilizar trabajo ya hecho sin tener que volver a repetirlo de nuevo. Esto resulta muy ventajoso para este caso.

13. Indique el tiempo necesario para llevar a cabo el descubrimiento de la clave para cada una de las redes. ¿Qué tiempo le llevaría auditar otra tercera red con estas mismas características (mismo ESSID)?. Justifique la respuesta.

Tal y como veíamos romper esta tercera red debería ser casi inmediato. Para obtener la clave de la primera red (captura AUT01.cap) tuvimos que calcular el 92,6 % de las PMKs posibles con lo aún en el improbable caso de que la PMK de esta tercera red no pertenece a ese 92,6% probar las PMKs ya calculadas y en su defecto calcular el 7,4% restante no debería suponer un retardo considerable. Esto se generaliza para más redes con lo que siempre que tengan el mismo SSID y su contraseña se ajuste a la plantilla definida anteriormente obtener sus contraseñas sería ya instantáneo.