



Packin' the PMK

Sobre la robustez de la autenticación de WPA/WPA2

Cédric Blancher y Simon Marechal

`cedric.blancher@eads.net`
Computer Security Research Lab
EADS Innovation Works

`simon@banquise.net`
Invitado especial
Organización no revelada ;)

BA-Con - 30 de Septiembre - 1ro de Octubre 2008
<http://ba-con.com.ar/>



Agenda

- 1 Autenticación WPA/WPA2
- 2 Evaluación de WPA-PSK
 - Como funciona ?
 - Costo teórico del ataque
 - Comparación entre implementaciones
 - Evaluación de la solidez de passwords
 - Límites de los ataques prácticos
- 3 Reflexiones sobre WPA-EAP
 - Autenticación EAP
 - Pwneando la Master Key
 - Consideraciones prácticas
- 4 Conclusión



Introducción

Seguridad Wi-Fi...

- WEP está mutilado y roto
- WPA vino a reemplazarlo
- Ahora, tenemos WPA2

Introducción

Seguridad Wi-Fi...

- WEP está mutilado y roto
- WPA vino a reemplazarlo
- Ahora, tenemos WPA2

Preguntas

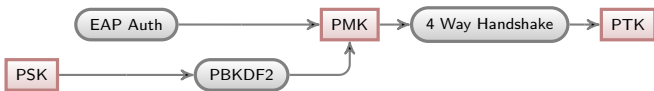
- En que son buenos WPA y WPA2?
- Cuanto tiempo van a durar?

Agenda

- 1 Autenticación WPA/WPA2
- 2 Evaluación de WPA-PSK
 - Como funciona ?
 - Costo teórico del ataque
 - Comparación entre implementaciones
 - Evaluación de la solidez de passwords
 - Límites de los ataques prácticos
- 3 Reflexiones sobre WPA-EAP
 - Autenticación EAP
 - Pwneando la Master Key
 - Consideraciones prácticas
- 4 Conclusión

Modos de autenticación

- Secreto pre-compartido (Preshared secret - PSK)
- EAP



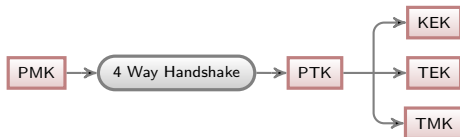
Jerarquía de claves

- Autenticación lleva a una Master Key (MK)
- Pairwise Master Key (PMK) se derivan de la MK

Una clave para gobernarlos a todos...

De la MK vienen todas las claves siguientes

- Pairwise Master Key
- Claves de intercambio de clave
- Claves de encriptación
- Claves de autenticación si se aplica



Obtener la Master Key == Obtener todo el resto

La opción de clave pre-compartida (Preshared Key)

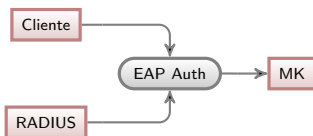
- MK es su PSK
- PMK se deriva de la MK



Obtener la PSK == Obtener la MK

La opción EAP

- Autenticación entre el cliente y RADIUS
- MK se deriva de la autenticación
- MK enviada al AP por RADIUS



Ownear cliente+RADIUS == Obtener MK



Agenda

- 1 Autenticación WPA/WPA2
- 2 Evaluación de WPA-PSK
 - Como funciona ?
 - Costo teórico del ataque
 - Comparación entre implementaciones
 - Evaluación de la solidez de passwords
 - Límites de los ataques prácticos
- 3 Reflexiones sobre WPA-EAP
 - Autenticación EAP
 - Pwneando la Master Key
 - Consideraciones prácticas
- 4 Conclusión



Calculando la PMK

La Master Key (MK)

- es tu clave secreta o contraseña
- 8 a 63 caracteres ASCII imprimible (código entre 32 y 126)

La Pairwise Master Key (PMK)

- se deriva de la Master Key y de datos del AP usando la función PBKDF2
- la función de derivación consume mucho tiempo



El ataque

Recuperando los datos relevantes

- se deben capturar durante el handshake
- es posible forzar el handshake
- solo funciona para un solo SSID



El ataque

Recuperando los datos relevantes

- se deben capturar durante el handshake
- es posible forzar el handshake
- solo funciona para un solo SSID

Probando una Master Key

El ataque

Recuperando los datos relevantes

- se deben capturar durante el handshake
- es posible forzar el handshake
- solo funciona para un solo SSID

Probando una Master Key

- 1 por cada posible MK, calcular el correspondiente PMK

El ataque

Recuperando los datos relevantes

- se deben capturar durante el handshake
- es posible forzar el handshake
- solo funciona para un solo SSID

Probando una Master Key

- 1 por cada posible MK, calcular el correspondiente PMK
- 2 calcular el PTK (cuatro llamados a HMAC-SHA1 usando PMK y nonces)



El ataque

Recuperando los datos relevantes

- se deben capturar durante el handshake
- es posible forzar el handshake
- solo funciona para un solo SSID

Probando una Master Key

- 1 por cada posible MK, calcular el correspondiente PMK
- 2 calcular el PTK (cuatro llamados a HMAC-SHA1 usando PMK y nonces)
- 3 finalmente, obtener el MIC (un llamado a HMAC-SHA1) y compararlo con el handshake capturado



La función PBKDF2

Algoritmo de PBKDF2

```
x1 = HMAC_SHA1(MK, SSID + '\1');  
x2 = HMAC_SHA1(MK, SSID + '\2');  
for(i=1; i<4096; i++) {  
    x1 = HMAC_SHA1(MK, x1);  
    x2 = HMAC_SHA1(MK, x2);  
}  
return x1 + x2;
```

Costo

- 8192 llamados a HMAC-SHA1



La función HMAC-SHA1

Algoritmo para HMAC(secreto, valor)

poner secreto en dos buffers de 64 bytes, Bi y Bo, rellenando con ceros

Bi

```
secret00000000000000000000000000000000  
00000000000000000000000000000000
```

Bo

```
secret00000000000000000000000000000000  
00000000000000000000000000000000
```




La función HMAC-SHA1

Algoritmo para HMAC(secreto, valor)

obtener SHA1(Bo)

330b72d384df41adf440e1d8aeb543ab73eecb8a

Resumen

$$\begin{aligned} HMAC_SHA1(s, v) = \\ SHA1((s \oplus 0x5c) || SHA1((s \oplus 0x36) || v)) \end{aligned}$$

La función SHA1

Descripción

- es una función de hash criptográfica
- funciona sobre bloques de 64 bytes, rellenando la entrada del usuario (padding)
- produce un digest de 20 bytes
- la parte principal de esta función se llama "BODY"
- las otras partes tienen un costo amortizado de cero

El truco HMAC

Recordatorio

- queremos $\text{SHA1}(\text{Bo} \parallel \text{SHA1}(\text{Bi} \parallel \text{value}))$

Qué se va a calcular

- $\text{BODY}(\text{secreto} \wedge 0x5c)$
- $\text{BODY}(\text{valor} + \text{padding})$
- $\Rightarrow \text{hash1}$
- $\text{BODY}(\text{secreto} \wedge 0x36)$
- $\text{BODY}(\text{hash1} + \text{padding})$
- $\Rightarrow \text{resultado}$



El truco HMAC

Recordatorio

- queremos $\text{SHA1}(\text{Bo} \parallel \text{SHA1}(\text{Bi} \parallel \text{value}))$

Qué se va a calcular

- $\text{BODY}(\text{secreto} \wedge 0x5c)$
- $\text{BODY}(\text{valor} + \text{padding})$
- $\Rightarrow \text{hash1}$
- $\text{BODY}(\text{secreto} \wedge 0x36)$
- $\text{BODY}(\text{hash1} + \text{padding})$
- $\Rightarrow \text{resultado}$

- si el secreto es constante ...

El truco HMAC

Recordatorio

- queremos $\text{SHA1}(\text{Bo} \parallel \text{SHA1}(\text{Bi} \parallel \text{value}))$

Qué se va a calcular

- $\text{BODY}(\text{secreto} \wedge 0x5c)$
- $\text{BODY}(\text{valor} + \text{padding})$
- $\Rightarrow \text{hash1}$
- $\text{BODY}(\text{secreto} \wedge 0x36)$
- $\text{BODY}(\text{hash1} + \text{padding})$
- $\Rightarrow \text{resultado}$

- si el secreto es constante ...
- ... dos llamados a BODY se pueden guardar en caché



La función BODY – inicialización

Algoritmo

```
unsigned int K[80];  
memcpy(K, input, 64);  
a = ctx[0]; b = ctx[1]; c = ctx[2];  
d = ctx[3]; e = ctx[4];
```

Cantidad de operaciones

- asignar 32 bits en memoria : 22



La función BODY – expansión de la entrada

Expandir la entrada

```
K[i] = K[i-3] ^ K[i-8] ^  
      K[i-14] ^ K[i-16];  
K[i] = rotate_left(K[i],1);
```

Cantidad de operaciones

- asignar 32 bits en memoria : 1
- operaciones elementales : 4
- se hace 64 veces

La función BODY – rondas

Algoritmo

```
STEP(v, w, x, y, z, m, c):  
    z += F(w, x, y) + c + K[m];  
    z += rotate_left(v, 1);  
    w = rotate_left(w, 30);
```

Cantidad de operaciones

- asignar 32 bits en memoria : 2
- operaciones elementales : 5 + costo de F
- 4 rondas de 20 pasos
- el costo promedio de F es 3.75 operaciones

La función BODY – finalizar

Algoritmo

```
a += ctx [0]; b += ctx [1]; c += ctx [2];  
d += ctx [3]; e += ctx [4];
```

Cantidad de operaciones

- asignar 32 bits en memoria : 5
- operaciones elementales : 5

La función BODY – resumen

Cantidad de operaciones elementales

- inicialización : 0
- expansión de la entrada : 4 veces 64
- rondas : 8.75 veces 80
- finalizar : 5
- total : 961

Comparación con MD5

- función BODY de MD5 : 496
- si se crackea un solo MD5 : 317

Costo de la función PBKDF2

Cantidad de operaciones elementales

- se necesitan 8192 llamados a HMAC-SHA1 usando los mismos secretos
- esto es, $2 + 8192 * 2$ llamados a SHA1
- o sea 15.7M operaciones elementales

Velocidad teórica de la función PBKDF2

Hipótesis : procesadores perfectos

- leer/escribir en memoria es gratis
- no hay penalidades

Velocidades

- implementación perfecta en SSE2 corriendo a 3Ghz en un x86 single core, unos 500 pruebas/s
- implementación perfecta en CELL (PS3, 7 SPU), unos 2,840 pruebas/s
- implementación perfecta en Linux CELL, unos 2,440 pruebas/s

Implementaciones en el mundo real

Aircrack

- 650 pruebas/s en Xeon E5405 (4x2Ghz)
- 650 pruebas/s en Opteron 2216 (4x2.4Ghz)
- "pipe multithreading", falla en AMD

Productos Pico Computing

- en un LX25 FPGA, 430 pruebas/s
- en un FX60 FPGA, 1,000 pruebas/s

Pyrit (Proyecto GPU)

- unos 6,000 pruebas/s en Tesla C870

Otros métodos de cracking

"rainbow tables" WPA-PSK

- en realidad son tablas para buscar PMK
- 1,000,000 passwords para 1000 SSIDs precomputados

Implementación de Jason Crawford CELL

- *"Lockheed rompe redes Wireless encriptadas con WPA usando un cluster de 8 Sony PlayStations"*
- para qué me molesto, ya está roto :/
- performance desconocido

Implementación en la arquitectura CELL

CELL benchmark

- no es un verdadero cracker, solo un benchmark
- en Linux, por lo tanto solo 6 SPUs disponibles
- se llena el pipeline crackeando 16 passwords al mismo tiempo

Implementación en la arquitectura CELL

CELL benchmark

- no es un verdadero cracker, solo un benchmark
- en Linux, por lo tanto solo 6 SPUs disponibles
- se llena el pipeline crackeando 16 passwords al mismo tiempo

Resultados

- 2,300 pruebas/s
- cercano al valor teórico 2,400 pruebas/s
- valor esperado en CELL

Mi implementación

NVidia CUDA cracker

- un cracker (casi) completo, requiere el input de un aircrack-ng modificado
- CUDA es sencillo : de no saber nada a esto en 4 días

Mi implementación

NVidia CUDA cracker

- un cracker (casi) completo, requiere el input de un aircrack-ng modificado
- CUDA es sencillo : de no saber nada a esto en 4 días

Resultados

- 4,400 pruebas/s en un 8800 gts
- 12,000 pruebas/s en un gtx280
- se debería poder mejorar
- más o menos equivalente a Pyrit



El mejor costo beneficio

Comparaciones del costo raw

Tipo	pruebas/s	costo	pruebas/s/\$
LX25	430	385\$	1.1
Q6600	800*	190\$	4.2
Q9550	900*	325\$	2.77
CELL	2300	400\$	5.75
gtx280	12,000	440\$	27.3
gtx260	9200*	300\$	30.6

Pero ...

- velocidades marcadas con * no son benchmarks reales, sino resultados interpolados
- el costo de CELL de 400\$ es por una *PlayStation completa*



Función de evaluación de la solidez de passwords

Una función F da la solidez s de un password p :

$$F(p) = s.$$

Función de evaluación de la solidez de passwords

Una función F da la solidez s de un password p :

$$F(p) = s.$$

Propiedades deseables

- ① computar $F(p)$ efectivamente dado cualquier p
- ② dado un s_{max} , enumerar y generar todas los passwords $\{p_0, p_1, \dots p_n\}$ donde $F(p_i) < s_{max}, 1 \leq i \leq n$
- ③ generar el conjunto $\{p_a, p_{a+1}, \dots p_b\}$ donde $F(p_i) < s_{max}, a \leq i \leq b$ sin generar $\{p_0, \dots p_{a-1}\}$
- ④ evaluar la solidez en una escala detallada



Métodos conocidos

Búsqueda en diccionarios

es débil si está en un diccionario

⇒ limitado a passwords "conocidos"

Métodos conocidos

Búsqueda en diccionarios

es débil si está en un diccionario

⇒ limitado a passwords "conocidos"

Complejidad del conjunto de caracteres

un password sólido contiene letras, números y al menos 3 caracteres especiales

⇒ Aún así se pueden crear passwords débiles

Métodos conocidos

Búsqueda en diccionarios

es débil si está en un diccionario

⇒ limitado a passwords "conocidos"

Complejidad del conjunto de caracteres

un password sólido contiene letras, números y al menos 3 caracteres especiales

⇒ Aún así se pueden crear passwords débiles

Test de cracking

es débil si se puede crackear en menos de 4 horas con john en mi máquina

⇒ requiere recursos computacionales compatibles con el análisis de riesgo

Un mejor método

Cadenas de Markov

- la distribución de probabilidad condicional de las letras L_n en un password es una función de la letra anterior, L_{n-1} , escrito $P(L_n|L_{n-1})$
- por ejemplo, $P(\text{sun}) = P(s).P(u|s).P(n|u)$
- para mantener números amigables,
 $P'(x) = -10.\log(P(x))$
- $P'(\text{sun}) = P'(s) + P'(u|s) + P'(n|u)$



En la práctica

Funciona bien

- tiene las propiedades deseadas
- crackea más efectivamente que *john -inc* (en mis pruebas!)
- existe un patch para *john*

En la práctica

Funciona bien

- tiene las propiedades deseadas
- crackea más efectivamente que *john -inc* (en mis pruebas!)
- existe un patch para *john*

Ejemplos de solidez

- "chall", solidez 100
- "chando33", solidez 200
- "chaneoH0", solidez 300
- "chanlLr%", solidez 400
- "chanereaAiO4", solidez 500
- "%!", solidez 1097

Hipótesis

Fuerza del atacante

Atacante	Tiempo disponible	Recursos (GPUs)
Wardriver	15 minutos	1
Individuo	7 días	2
Organización grande	1 año	1024

Fuerza del defensor

- escenario del peor caso : usuario de mac :)
- password tiene 12 caracteres o menos

Passwords no muy buenos

Fuentes estadísticas

- un foco enfocado en Apple que fue owneado
- los passwords fueron publicados en texto plano en 4chan

Solidez de passwords

- 628,753 passwords
- solidez media : 245
- solidez mediana : 197
- las "bases" más comunes de los passwords :
password, qwerty, apple, letmein

Solidez de passwords crackeables

Ahora

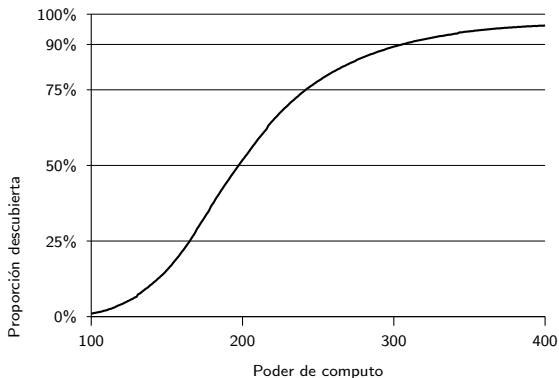
- wardriver : 7.2M, solidez markov de 169
- individuo : 14.5G, solidez markov de 239
- organización grande : 387.8T, solidez markov de 344

En 10 años, 32 veces más rápido (Moore)

- wardriver : 345.6M, solidez markov de 202
- individuo : 464.5G, solidez markov de 273
- organización grande : 12409T, solidez markov de 388

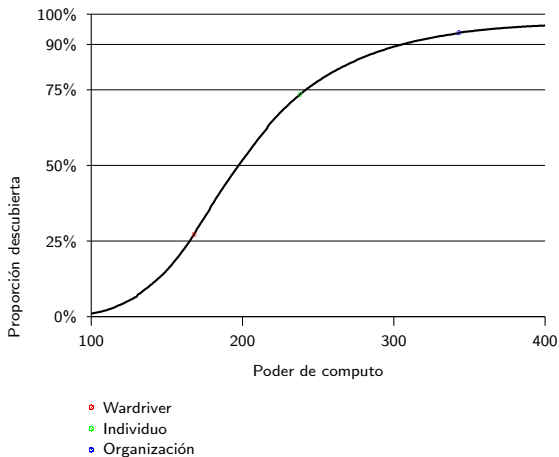


Proporción descubierta vs. poder de computo



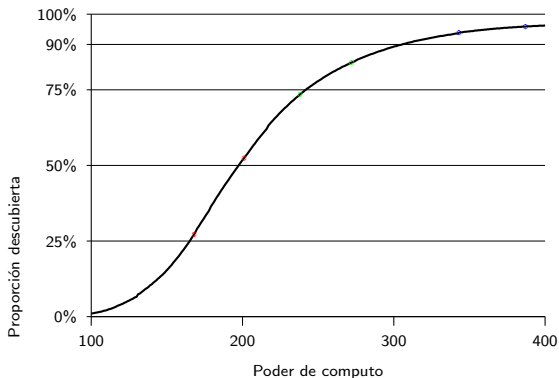


Proporción descubierta vs. poder de computo





Proporción descubierta vs. poder de computo



- ◻ Wardriver
- ◻ Individuo
- ◻ Organización

Extrapolación en 10 años

Agenda

- 1 Autenticación WPA/WPA2
- 2 Evaluación de WPA-PSK
 - Como funciona ?
 - Costo teórico del ataque
 - Comparación entre implementaciones
 - Evaluación de la solidez de passwords
 - Límites de los ataques prácticos
- 3 Reflexiones sobre WPA-EAP
 - Autenticación EAP
 - Pwneando la Master Key
 - Consideraciones prácticas
- 4 Conclusión

Problemas usuales

La solidez de EAP depende directamente de una buena configuración

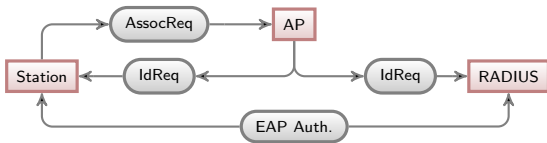
- Buena elección del método EAP
- Autenticación RADIUS apropiada

En particular...

Verificar estrictamente el certificado RADIUS para evitar MiM

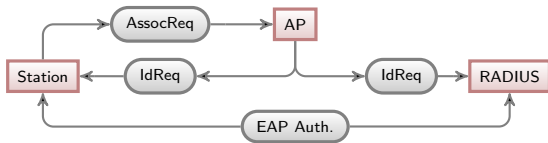
Mirando más cuidadosamente

AP actúa como retransmisor entre el cliente y un servidor RADIUS



Mirando más cuidadosamente

AP actúa como retransmisor entre el cliente y un servidor RADIUS



Comunicación EAP directa entre el cliente y RADIUS

Que pasa si...

Hay una falla explotable dentro de EAP ?

- Capacidad de ejecutar código arbitrario
- Acceso a la base de datos RADIUS
- Acceso al backend
- Etc.

Más importante

Capacidad de generar tráfico RADIUS !

De la transmisión MK

Notificación AP

Cuando se realiza la autenticación, RADIUS notifica AP

- EAP Success (3) o Failure (4)
- se manda MK usando MS-MPPE-Recv-Key (atributo 17)
- mensaje HMAC-MD5 (atributo 80)



Injectando MK arbitrarias

- Ejecutá tu shellcode
- Elaborá un EAP Success
- Poné tu propia MK en MS-MPPE-Recv-Key
- Lográ que se mande al AP

Pequeño problema...

Necesitas computar un mensaje HMAC-MD5



Eludiendo HMAC-MD5

- No conoces el secreto de RADIUS
- Pero owneaste el servidor...

Ideas

- Leer secreto de la configuración / memoria
- Pedirle a RADIUS to elabore el paquete por vos

Métodos que dependen del producto

En la práctica

Algunas cosas hechas o por hacer

- EAP fuzzing (vulnerabilidades)
- EAP fingerprinting (id)
- Exploits

Entonces...

El atacante puede tener su propio MK enviado de vuelta al AP

Todavía no es el final...

Todavía se necesita hacer un 4-Way Handshake

- Hackear un WPA/WPA2 supplicant !
- Módulo específico para wpa_supplicant

Paso a paso

- Responder el pedido EAP del AP
- Empezar el diálogo EAP con RADIUS
- Gatillar la vulnerabilidad
- Mandar exploit
- Agarrar EAP Success del AP

Cuando terminaste...

Al final...

- Rogue client empieza la autenticación
- Explora el servidor RADIUS
- Se autentica con MK arbitraria
- Termina el diálogo WPA/WPA2 con AP

Cuando terminaste...

Al final...

- Rogue client empieza la autenticación
- Explota el servidor RADIUS
- Se autentica con MK arbitraria
- Termina el diálogo WPA/WPA2 con AP

Más importante...

Puede acceder a la red a través de Wi-Fi

Agenda

- 1 Autenticación WPA/WPA2
- 2 Evaluación de WPA-PSK
 - Como funciona ?
 - Costo teórico del ataque
 - Comparación entre implementaciones
 - Evaluación de la solidez de passwords
 - Límites de los ataques prácticos
- 3 Reflexiones sobre WPA-EAP
 - Autenticación EAP
 - Pwneando la Master Key
 - Consideraciones prácticas
- 4 Conclusión

Selección de PSK

Recomendaciones

- si es posible, usar un valor de 64 bytes al azar, o uno de los esquemas de autenticación más seguros
- usar passwords que no sean derivados de una palabra conocida y con una solidez de 400 o más en la escala de Markow debería ser seguro para los próximos años
- simplemente use "chanereaAiO4", es seguro !

Selección de PSK (cont.)

Cuidado

- el cracker puede tener un mejor modelo para sus ataques
- frases "reales" pueden parecer seguras porque son largas, pero lo más probable es que sean débiles
- criptográficas defectos podrían ser descubiertos y explotados

El futuro de PSK

Setup automático de la clave

- muchas soluciones propietarias, y un estándar
- automágicamente configura la red y los settings de seguridad
- elimina el input del usuario, no más claves débiles (ojalá)

Wi-Fi Protected Setup

- estándar de la Wi-Fi Alliance
- autentica el aparato
 - in-band : ingresando un código PIN, apretando un botón
 - out-of-band : conectando una memoria USB, leyendo los RFIDs
- puede ser atacado durante la primer asociación

Consideraciones sobre EAP

Recomendaciones

- Elegir cuidadosamente el método EAP
- Asegurarse que los clientes puedan autenticar RADIUS
- Proteger la maquina RADIUS
- Proxear la autenticación a otro servidor AAA

Consideraciones sobre EAP

Recomendaciones

- Elegir cuidadosamente el método EAP
- Asegurarse que los clientes puedan autenticar RADIUS
- Proteger la maquina RADIUS
- Proxear la autenticación a otro servidor AAA

Cuidado

- Verificar el certificado RADIUS, siempre
- Contra tu propio CA, únicamente



El fin...

Gracias por su atención

Preguntas?