aRC-FL-Cracking 003 (01/08/2003)

Tiempo de cracking

(por Furious Logic [aRC])

Advertencia

Antes de poner en práctica el presente documento y cualquier material asociado al mismo, sea éste naturaleza tangible o intangible usted debe estar totalmente de acuerdo con todos los términos y condiciones siguientes:

Del software

Cualquier software se proporciona tal como está, sin ninguna garantía expresa ni implícita de ningún tipo.

aRC no garantiza ni asume responsabilidad alguna en cuanto a la integridad o exactitud de cualquier información contenida en el software.

Ni los miembros ni los colaboradores ni los invitados aRC se hacen responsables por el uso que se le pueda dar al software.

Al examinar, copiar, ejecutar, instalar o utilizar el software, el lector está aceptando su total conformidad con todos los términos y condiciones enunciados.

Del documento

Al abrir este documento, el lector acepta incondicionalmente su total y exclusiva responsabilidad legal, de acuerdo a las leyes vigentes en su país, por el uso de las técnicas experimentales, educativas y/o de investigación aquí vertidas en materia de programación especializada de computadoras.

En caso de discrepar con alguno de los puntos descritos, deberá eliminar inmediatamente el presente documento y todo material asociado al mismo.

Agradecimientos

A Lotus Corporation (TM) por su excelente producto Word Pro, parte integral de la suite de oficina Lotus SmartSuite Millenium, cuyas características dejan sin aliento al procesadorcillo de textos de Micro$$oft. Gracias a aquellos programadores por su excelente trabajo.

A FontLab 3.00F de FontLab Developers Group por permitirnos asignar permiso completo a las fuentes true type protegidas contra copia y a Acrobat Distiller 5.0 de Adobe Systems por su excelente resultado en la creación del documento electrónico en formato PDF.

A Borland International (¿Inprise?) por sus magníficos compiladores C++ Builder y Delphi. ¡Pero ya dejen de hacer crecer los ejecutables por favor!

A LiuTaoTao, ZhuNanHao por su maravillosa herramienta TRW2000, de tamaño ultra-compacto, que nos ahorra horas de trabajo. Agradeceríamos muy de veras, a quien nos pueda proporcionar la nueva dirección de este programa, debido a que http://www.knlsoft.com no lo es más. Lo último que leímos en su web fue el lanzamiento de una versión muy superior a la 1.23.

Objetivos

Enseñar a utilizar un procedimiento metódico y organizado de investigación, aplicable a toda materia y no exclusivamente al cracking.

Desensamblar un programa utilizando el sentido común, evitando la chapucería y siempre terminando lo iniciado.

Demostrar, una vez más, que crackear no es difícil cuando se tiene una sólida base teórica.

Aclaración

Damos por hecho que el lector ya tendrá las herramientas para su labor. De cualquier forma, las mencionaremos nuevamente en el transcurso de la lección presente.

Al fin, después de tantos días, presentamos a los lectores, una lección que consideramos con más sabor. Algunos visitantes del canal recibieron un adelanto y luego de ir cayendo uno por uno hasta no quedar ninguno, lo que lamentamos, les presentamos a todos la versión ampliada y nada resumida de aquella quasi-clase. Si alguno logra cruzar el umbral, lo felicitamos de antemano. A quienes prefieran quedarse en el camino, les encargamos nuestros saludos para todo el club de los "no-se-puede" al que finalmente se unirán.

Tenga presente el método de aprendizaje enunciado en aRC-FL-Cracking 001 Iniciación:

1. Escuchar. Los canales IRC por ejemplo. ¡No nos referimos a #solteros ni #amigos, sino a #crackers, #cracker, #delphi32, #programming, #assembler, etc.!
2. Leer. Si no puede comprar libros, Internet es la solución o conviértase en un "ratón de biblioteca". También puede buscar en las enormes bibliotecas virtuales que existen. Hace un tiempo, http://www.attrition.org/~modify/texts era nuestra favorita. Supuestamente el FBI la clausuró.
3. Escribir. Sí, es verdad que la información ya está escrita en el libro, pero su mente asimilará mucho más si vuelve a escribir lo asimilado en sus propias palabras. No mande hacer el trabajo en el cybercafé de enfrente. ¡Piense por sí mismo! Y si le duele la cabeza de tanto pensar, mejor dedíquese a algo "menos exigente".
4. Practicar. ¿Qué es lo que se practica? -La teoría- ¿y qué va a practicar si no tiene un sólido fundamento teórico? La práctica solo es posible cuando se domina la teoría.
5. Enseñar. Esta es la mejor parte. No tiene que ser un genio para poder enseñar, solo tiene que centrarse en un tema a la vez, dominarlo bien y escribir al respecto. Las preguntas de sus alumnos desafiarán sus conocimientos y lo instarán a aprender más. No se conforme, rétese a sí mismo y no rete a los demás.

Insistimos con más teoría

Antes de entrar de lleno en el propósito de esta lección, adicionemos algunos conceptos a nuestro conocimiento básico. Es inadmisible un investigador informático o under, término propuesto en la lección inicial, ignorante de su entorno tecnológico. Tomaremos esta sección como un refrescante recreo.

"El recreo se hizo para estudiar" - Anónimo

11-20 Bytes

(1) Lo único necesario para formatear "por casualidad", la pista 0 del disco duro de nuestro laboratorio personal, o mejor aún, del disco duro de la universidad. ¡Aaauuchhh!

ASCII

(1) Siglas de *American Standard Code for Information Interchange*, "código estándar americano para el intercambio de información".
(2) Tabla de 128 caracteres comunes que permite que la información que se utiliza en las computadoras, sea fácilmente intercambiable entre sistemas distintos.
(3) Posteriormente se agregaron 128 caracteres adicionales, haciendo un total de 256, pero éstos últimos si son variables de acuerdo a diversos factores.
(4) Sus competidores son las tablas EBCDIC, UNICODE y ANSI, entre otros. ANSI, solo es una variante de las tablas ASCII que agrega forma a cada caracter ASCII en sus segundos 128 caracteres.

ASCIIZ

(1) No se trata de una variante de las tablas ASCII.
(2) Es un término que se utiliza para indicar una cadena de caracteres formada por caracteres ASCII y que termina o debe terminar con el caracter cero o *zero* (la inicial de esta palabra se utiliza como sufijo para el término).

Bootstrap

(1) Programa grabado en el sector de carga de un disco duro, diskette, cdrom, zip.
(2) Cualquier programa pequeño puede ser grabado allí. ¿Suena interesante no es así?

Buffer

(1) Se traduce como "memoria de almacenamiento intermedio".
(2) Es una reducida cantidad de memoria, sumamente veloz, que almacena los datos que serán transmitidos entre un periférico y el disco duro.
(3) Ejemplos de uso: buffer de teclado, buffer de grabadora de CD, buffer de disco duro, buffer de tarjeta de sonido, etc.

Checksum

(1) Acrónimo para *Summation Check* traducido como "suma de verificación".
(2) Se obtiene sumando el valor binario de cada caracter implicado en el bloque de datos analizado. Esta suma acompaña al bloque de datos en su transmisión. Cuando los datos llegan a su destino, se vuelve a calcular el checksum y se compara con el que se recibió. Ambos deben ser idénticos para asegurar la no existencia de errores.

CRC

(1) Siglas de *Cyclical Redundancy Checking* traducidas como "verificación de redundancia cíclica".
(2) "Los mensajes transmitidos, se dividen en longitudes predeterminadas, que usadas como dividendos, son divididas por un divisor fijo. El resto de la división es agregado al mensaje y enviado con el mismo". El receptor calculará el CRC y lo comparará con el recibido. Su no coincidencia significa error de datos. (Alan Freedman, "*CRC*", *Diccionario de computación*).
(3) Existen muchos otros algoritmos de cálculo del CRC.

CISC

(1) Siglas para *Complex Instruction Set Computer* que se traducen como "conjunto de instrucciones complejas de computadora".
(2) Tipo de microprocesador que posee una gran cantidad de instrucciones internas, lenguaje de máquina, para poder programarlo.
(3) Su procesamiento es de UNA SOLA INSTRUCCIÓN POR VEZ.
(4) Aquí se encuentran todos los procesadores de la serie Intel, AMD, Cyrix. Es decir computadoras personales comunes.

Dumpear

(1) *Spanglish* proveniente del término anglosajón *dump*, que se traduce como "volcar".
(2) Todo programa, al ser ejecutado, se carga en memoria como si fuese un duplicado. Volcar es el procedimiento mediante el cual se graba ese duplicado en un archivo físico.
(3) Se utiliza comúnmente con los programas comprimidos.
(4) Utilice el término volcar que es la palabra correcta.

Empaquetar/Desempaquetar

(1) Procedimiento que consiste en aplicar un algoritmo determinado a uno o más archivos para reducir el espacio de almacenamiento que ocupan.
(2) Los compresores de ejecutables debieran interesarnos. El estudio detallado de las técnicas de compresión, enriquecerá el nivel de nuestro bagaje técnico elevado en el cracking.

Ingeniería Reversa

(1) Proviene de *Reverse Engineering*.
(2) No se aprende en ningún centro de estudios.
(3) Desde las más antiguas épocas, la antigua tradición del underground nos ordenaba no escolarizar su aprendizaje. Hasta este día, no conocemos ninguna Escuela de Ingeniería Reversa y esperamos nunca llegar a conocer una. La razón de ello, resulta evidente: "No existe placer más gratificante, ni recompensa más apreciada, que descubrir los mecanismos ocultos del funcionamiento de un programa, uno mismo". No es igual ver la película "El Señor de los Anillos", que imaginarla mientras se leen los 3 tomos de la obra literaria, más los 2 tomos que conforman su preámbulo y postámbulo respectivamente. El verdadero mérito de la Ingeniería Reversa y de la tecnología underground está es realizar nuestros propios descubrimientos.
(4) Es el procedimiento que consiste en ejecutar un programa e ir retrocediendo en su funcionamiento para descubrir sus algoritmos de programación. Requiere de nuestra total atención, intuición, deducción y el más razonable sentido común.
(5) Ocasionalmente denominada Ingeniería Inversa.

Ingeniería Social

(1) No se aprende en ningún centro de estudios.
(2) Técnica que consiste en obtener información de un individuo, sin que éste se percate de que la está compartiendo.
(3) En su momento, fue muy empleada por los phreakers (especialistas en tecnología de las comunicaciones). El cracking puede valerse de ella si...

Kernel

(1) Término yanquilandés para "núcleo".
(2) Base fundamental sobre la que se apoya todo sistema operativo. Algunos kernel son buenos, otros son malos, otros son como los que trae Micro$$oft Windown en todas sus versiones.
(3) Teóricamente, es uno de los más bajos niveles sobre el cual se puede crackear.
(4) Es poco frecuente trabajar con el cracking a este nivel.

PE

(1) Formato nuevo de archivos ejecutables que reemplazó a los antiguos NE, LE y otros. El conocimiento pleno de su estructura nos mostrará el siguiente escaño que debemos alcanzar en el cracking.
(2) Se refiere al pseudosistema operativo Micro$$ft Window$$.
(3) Proviene de la especificación COFF (*Common Object File Format*, "formato común de archivos objeto") que es común en sistemas UNIX. Como siempre, Micro$$oft robando las ideas de los sistemas operativos genuinos.

RISC

(1) Siglas para *Reduced Instruction Set Computer* que se traduce como "conjunto reducido de instrucciones de computadora".
(2) Tipo de microprocesador que posee una cantidad reducida, comparada con CISC, de instrucciones internas o lenguaje de máquina para su programación.
(3) Su procesamiento es de ±6 INSTRUCCIONES A LA VEZ dependiendo del diseño del procesador.
(4) Aquí tenemos a los procesadores PowerPC, PowerMac de Motorola (Macs), y Alfa. Existen computadoras personales con este tipo de procesadores, aunque su costo es obviamente elevado. La mejor de ellas es la Silicon Graphics Workstation con la que se realizan los renderizados y animaciones para las películas de alto presupuesto.
(5) Por ejemplo, un procesador RISC de 1,000 MHz es incomparablemente más veloz que uno CISC de 1,000 MHz debido a las razones presentadas.

Smoke test

(1) Traducido como "prueba de humo". Es nuestra prueba favorita.

(2) Prueba de equipos nuevos o reparados que consiste en encenderlos. ¡Si no sale humo, entonces funcionan!
(3) ¿Esto es tecnología? ¡Quien lo imaginaría, pero así es!. ¡Investígue este principio!.

Thread

(1) Traducido como "hilo de multitarea".
(2) La técnica que utiliza un sistema operativo para simular el multiprocesamiento y la multitarea es en base a los *threads*. Los *threads* son minúsculos fragmentos de segundo asignados a cada proceso que se ejecute, turnándolos apropiadamente, para que nos den la apariencia de estar ejecutándose todos al mismo tiempo.
(3) La multitarea real solo existe en procesadores RISC.
(4) Cabe destacar, que la multitarea del sistema operativo OS/2 sobre un procesador CISC, es sorprendente a pesar de sus limitaciones de hardware. Los programadores de Micro$$oft deberían visitarlos más a menudo e incluso a diario para ver si logran aprender a programar decentemente.

Versión alfa

(1) Versión inicial de un programa, que está compuesta de código no terminado y lleno de errores (*bugs*) por corregir.
(2) El afán por captar la atención del público está haciendo circular este tipo de software, aún cuando no está listo para su comercialización.
(3) Alfa "α" es la primera letra del alfabeto griego. De allí proviene su utilización en el sentido de "primera".

Versión beta

(1) Versión en fase de prueba de un software determinado, destinada al grupo de programadores evaluadores o *beta testers*.
(2) Se está haciendo muy común la distribución de este tipo de software por razones netamente mercantilistas. Sin embargo, también se distribuye para ser evaluado gratuitamente por los usuarios, que se convierten en los evaluadores que no cobran.
(3) Beta "β" es la segunda letra del alfabeto griego. De allí proviene su utilización en sentido de "segunda".

Versión demo

(1) Apócope anglosajón del término *demonstration* que significa "demostración" o "muestra".
(2) Algunas compañías de software e incluso programadores particulares, diseñan versiones recortadas de sus productos. Es decir, versiones que carecen de algunas funciones. A estas versiones se las denomina demo.
(3) Existen unas pocas versiones autodenominadas demo que realmente son versiones *shareware*, porque en realidad no están recortadas.
(4) Estas versiones NO son buenas candidatas para practicar el cracking.

Versión freeware

(1) El término *freeware* se traduce como "gratuito".
(2) Es un tipo de software que combate el monopolio y el afán lucrativo excesivo impuesto por las grandes corporaciones transnacionales. Presenta como alternativa viable, las magníficas ofertas de la investigación tecnológica de vanguardia que ofrecen los programadores freeware. No estamos en contra de obtener dinero por el arduo trabajo, pero cifras como $3,500.00 por un pedazo de plástico de 12 cm de diámetro es, sin lugar a dudas, un robo descarado.
(3) No se debe confundir "gratuidad" con "mediocridad". El hecho de que muchos programadores alrededor del mundo diseñen software GRATUITO no significa, de ninguna manera, que sus productos sean de mala calidad. Por el contrario, los programas freeware demuestran toda la dedicación, vocación y pasión de un programador por su carrera, suficientes alicientes para inspirarlo a crear sin percibir nada a cambio. La programación freeware es similar a un *hobbie*.
(4) Los detractores de este alturado software lo llaman "socialismo informático".

Versión open source

(1) El término imperialista *open source* significa "código abierto".
(2) No es necesariamente un software gratuito.
(3) Es un tipo de software que se distribuye con su código fuente.
(4) Linux es uno de sus más conocidos ejemplos.

Versión shareware

(1) El término *shareware* puede ser traducido como "para comprar" o "software compartido".
(2) Se trata de un software cuya única distinción entre los demás, es que puede ser utilizado por el usuario durante un período de tiempo de evaluación limitado o sólo para ser ejecutado una cantidad determinada de veces. Transcurrido el período de prueba o agotada la cantidad de ejecuciones permitidas, el software deberá ser comprado o desinstalado. Algunas veces se bloquea y no permite ser ejecutado nuevamente, incluso si lo reinstalamos.
(3) Los programas shareware son distribuidos con la idea de probar las bondades del producto antes de comprarlo.
(4) A diferencia del software demo, las versiones shareware no están recortadas sino que, generalmente, son versiones completas con algunas opciones desactivadas.
(5) Estas versiones SÍ son excelentes candidatos para practicar el cracking.

Y dice así:

Lo primero es organizarnos. Aquella arcaica idea de tener todo desordenado tan sólo nos asemeja a bárbaros empíricos y no a especialistas fundamentalistas en tecnología informática como corresponde. De ninguna manera somos los "Médicos brujos de las computadoras".

Necesitamos disponer de una Ficha de Control preparada para contener lo siguiente:

1. Código o número de ficha. Puede ser secuencial o codificado.
2. Nombre del programa.
3. Versión específica del programa.
4. Plataforma de ejecución. No siempre será Winbugs ¿o sí?.
5. Copyright o derechos de autor.
6. URL del autor.
7. Descripción y breves comentarios acerca del programa.
8. Directorio de ubicación de cada objetivo.
9. Archivo(s) objetivo(s) a modificar.
10. Tamaño(s) en bytes del(de los) archivo(s) objetivo.
11. Observaciones iniciales.
12. Herramientas utilizadas.
13. Bytes a modificar. Si necesita más espacio, utilice el reverso de la hoja para continuar.
14. Fecha de éxito. ¿Cómo olvidar la fecha de una victoria?
15. Hora de éxito. Precisión ante todo.
16. Tiempo que nos tomó el crack manual que no incluya el tiempo que nos tomó la programación.

Ficha N°	:	
Programa	:	
Versión	:	
Plataforma	:	
Copyright	:	
URL	:	
Descripción	:	
Directorio	:	
Objetivo(s) / Tamaño(s)	:	
Observaciones iniciales	:	
Bytes	:	
Fecha	:	
Hora	:	
Duración	:	

Observe nuestra muestra personal. No la copie, sea creativo y mejórela ¿o acaso intenta decirnos que no tiene suficiente capacidad mental para hacerlo? (¿Eso dolió no? ¡Qué bueno! ¡Trabaje!)

Ahora que ya tenemos varias decenas de fotocopias de nuestra ficha original, deberemos aprovisionarnos de muchas hojas de papel que nos servirán para nuestros bosquejos. No sea tacaño al escribir que para eso se inventaron el lápiz y el papel. Plasme sus ideas en un papel, pero organizadamente como estipula el método científico. De paso, a ver si se consigue un libro acerca de Técnicas de Investigación Científica, luego verá que le es muy útil.

Necesitamos las herramientas siguientes:

(1) Un desensamblador. W32Dasm. La última versión es la 8.93.
(2) Un editor hexadecimal como Hackers View 6.82 (Hiew). También podríamos utilizar el Binary View 5.32 mejor conocido como Biew y es gratuito (editor hexadecimal favorito del autor).
(3) Un administrador de archivos. Nada de explorador de windows ni nada parecido. Eso no sirve. Busque Total Commander. La última versión es la 5.50. Si prefiere, puede utilizar WinNc 3000 Professional (nuestro administrador de archivos favorito) u otro programa similar y con las mismas características de Total Commander o mejores. Aceptamos sugerencias.
(4) Un buen cuaderno en donde anotar todo lo que aprenda.
(5) Una buena taza de café negro como vuestra conciencia, para evitar el "agotamiento mental" que pudiera sobrevenirle. Mejor olvide el café y acepte este consejo: Siempre duerma 8 horas completas, no más, no menos. El trabajo mental es el más agotador. Su organismo y el underground le agradecerán el reparador sueño.

Configuración de Total Commander

Configuraremos Total Commander (en adelante TC) para que nos permita desensamblar (W32Dasm) y editar (Hiew) directamente. Sólo lo explicamos en esta lección.

1. En uno de los 2 paneles que muestra TC, seleccionaremos el directorio del programa `Hiew`.
2. Con el botón izquierdo arrastraremos el ejecutable, `hiew32.exe` ó `hiewdemo.exe`, hacia una zona libre de la barra de herramientas para agregar un nuevo acceso a nuestro programa.
3. Click derecho en el nuevo ícono creado, opción `Change` y en el campo `Parameters` escribiremos `%p%n` (en minúsculas). También podríamos cambiar el ícono y el tooltip o mensaje emergente.
4. Enseguida haremos lo propio con el programa `W32Dasm`, cuyo ejecutable debiera ser `w32dsm89.exe`. Repetiremos los pasos 1 al 3 por cada acceso directo adicional que necesitemos.
5. Eligiendo el menú `Configuration`, opción `Button bar` accederá a las opciones de la barra en general. O también, click derecho en una zona vacía de la barra de botones, opción `Change`.
6. La ayuda de TC explica que `%p%n` (en minúsculas) carga el archivo en que se encuentra el cursor (barra de color), con el programa para el que hemos creado el acceso en la barra de botones.
7. Esta es la configuración óptima para las opciones de TC:
   7.1 Menú `Configuration, Options`
      7.1.1 `Layout`: Todas las casillas de verificación activadas.
      7.1.2 `Operation`: Seleccione `"Also select directories"` y también `"Letters – with search dialog"`. Deje el resto como está.
      7.1.3 `Display`: Marque `"Show hidden/system files"` y también `"Show filename in file list as tooltip"`. Deje el resto tal cual.
      7.1.4 `Packer`: Proporcione las rutas que se le solicitan. Que no le falte ningún compresor.
   7.1 En ambos paneles de la pantalla principal, pulse el botón `Ext` para ordenar los listados por extensión de archivo.
   7.2 Menú `Configuration, Save position`
   7.3 Menú `Configuration, Save settings`
7. Dentro del directorio de winbugs encontrará un archivo `wincmd.ini`. Guarde una copia de seguridad del mismo, para no tener que repetir el proceso de configuración en caso de tener que reinstalar todo nuevamente, a menos que su versión de windows sí sea estable (¡¿?!)
8. Dentro del directorio de TC encontrará el archivo `default.bar` que contiene su barra de herramientas personalizada. Guarde una copia también.
9. Pruebe el resto de opciones: `F3, F5, F6, F7, CTRL+↓, ALT+↓, CTRL+D, CTRL+E ...`

Configuración de W32Dasm

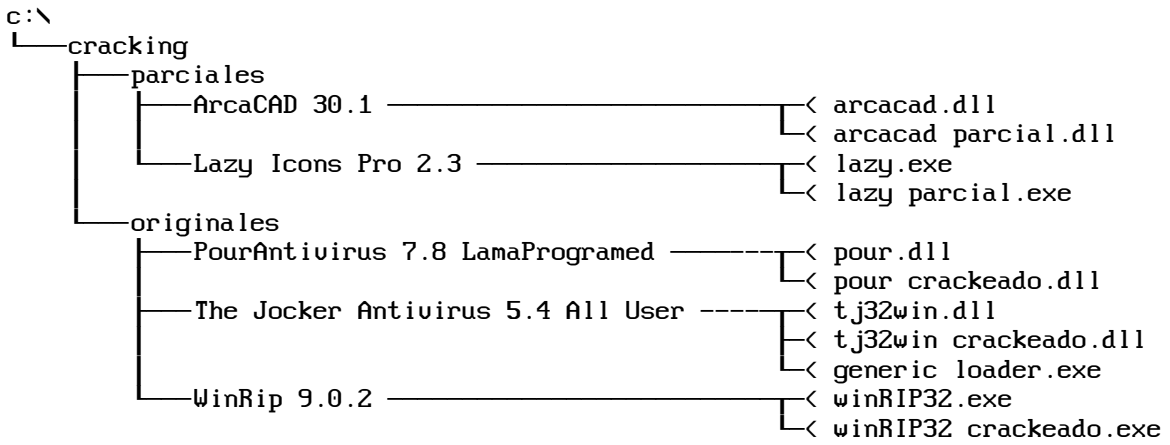
W32Dasm (en adelante W32) tiene muy buenas características y opciones diversas pero es necesario mejorarlo con algunas adiciones:

1. Es imprescindible el patch de `Cold Coder` para W32 porque realiza unas modificaciones muy interesantes, que además son necesarias para que funcione con un acceso directo en TC.
2. Al cargar W32, menú `Disassembler`, en `Disassembler options`, todos los `enable` deben estar activados. No se requiere ninguna otra modificación.

Preparación de directorios

Hemos modificado el método que utiliza **The Keyboard Caper** o **tKC** fundador de **Phrozen Crew** y actual fundador de **Phrozen Hell**, miembro también del famoso grupo de cracking conocido como CIA, en sus tutoriales incluidos en el afamado programa Oscar. Adáptelo a su gusto, pero mantenga la idea que se vierte.

1. Creamos un directorio **Cracking** y dentro de él, **Originales** y **Parciales**.
2. **Originales** contendrá un directorio por cada programa que ya hemos crackeado. Debe ser denominado con el nombre y versión del programa. Por ejemplo **"WinRIP 9.0.2"**.
3. **Parciales** contendrá un directorio por cada programa que aún no terminamos de crackear o que tenemos en la lista de espera. Debe ser denominado con el nombre y versión del programa. Por ejemplo **"WinRIP 9.0.2"**.
4. Cada directorio dentro de **Originales**, contendrá el archivo **"WinRIP32.exe"** y su correspondiente **"WinRIP32 crackeado.exe"**, que es el mismo ejecutable pero con los bytes ya modificados. Ambos archivos nos servirán para comparar y generar **"crack.exe"** por ejemplo.
5. Cada directorio dentro de **Parciales**, contendrá el archivo **"WinRIP32.exe"** y su correspondiente **"WinRIP32 parcial.exe"** que es el ejecutable con algunos bytes editados.
6. Cuando finalicemos un crack, moveremos su directorio de **Parciales** hacia **Originales** y renombraremos su contenido **"WinRIP32 parcial.exe"** por **"WinRIP32 crackeado.exe"**.
7. Es posible que exista más de un objetivo. En tal caso, siempre conservaremos 2 archivos por cada objetivo: 1 original y 1 crackeado (1 parcial en el caso del directorio **Parciales**).
8. Aquí el árbol de directorios que resume los pasos anteriores:



Nota para despistados: Queda sobreentendido que **"WinRIP32.exe"** sólo es un ejemplo. ¡Esta aclaración es una ofensa!

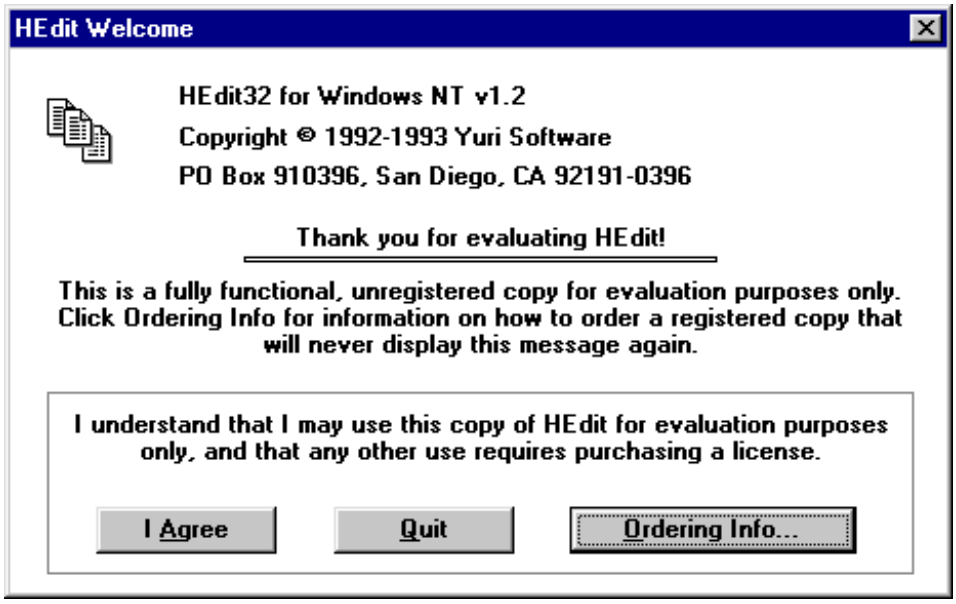
Tiempo de cracking

Ahora ya podemos crackear porque todo lo necesario está disponible. Nuestro objetivo será el primer éxito del autor, sin falsa modestia, un antediluviano editor hexadecimal llamado **Hedit32 1.2**. Ponemos unos VQF de Sandra y Michael Cretu con su misteriosa agrupación Enigma, y estamos listos (Hedit32 1.2 acompaña a este tutorial)

1. Cargamos TC. En el panel izquierdo seleccionamos el directorio de Hedit. En el panel derecho seleccionamos el directorio **Parciales** y creamos el directorio **Hedit 1.2** dentro de él.
2. Observamos detenidamente el contenido del directorio de **Hedit 1.2** y nos encontramos con 3 archivos que nos llaman la atención (siempre **.exe** y **.dll** preferiblemente): **edit.dll**, **hedit.exe** y **superpad.exe**.
   Observación: Algunas compañías de software crean falsos archivos **.dll** pretendiendo engañarlo. ¡No se deje sorprender!
3. Un alto porcentaje de probables objetivos son los archivos ejecutables (**.exe**) que cargan el programa principal. En segundo lugar, se encuentran las librerías de enlace dinámico (**.dll**) cuyo

nombre es similar al del ejecutable principal o también de nombre sugestivo como `rarreg.dll` ("reg" es un probable apócope de *registration*, "registro"). Es importante anotar esto porque se da con mucha frecuencia.

4. De todas formas cargamos `superpad.exe` y al examinar su funcionamiento, descubrimos que solo es un editor mejorado de texto plano. No parece necesitar de nuestros servicios.
5. De los 2 archivos que nos quedan intuimos que el objetivo es `hedit.exe`. Por lo tanto, lo copiamos hacia el panel derecho (`cracking\parciales\Hedit 1.2`).
6. En el mismo directorio donde reside el objetivo, panel izquierdo, copiamos `hedit.exe` como `1.exe`. (En TC movemos el cursor hacia `hedit.exe` y presionamos `SHIFT+F5`, escribimos `1.exe` y aceptamos).
7. En el panel izquierdo cargamos `hedit.exe`, y lo primero que vemos es un bonito nag con todas sus molestias. Anotamos su contenido para utilizarlo en aRC-FL-Cracking 004 ¡Lo logré! (No, aún no)

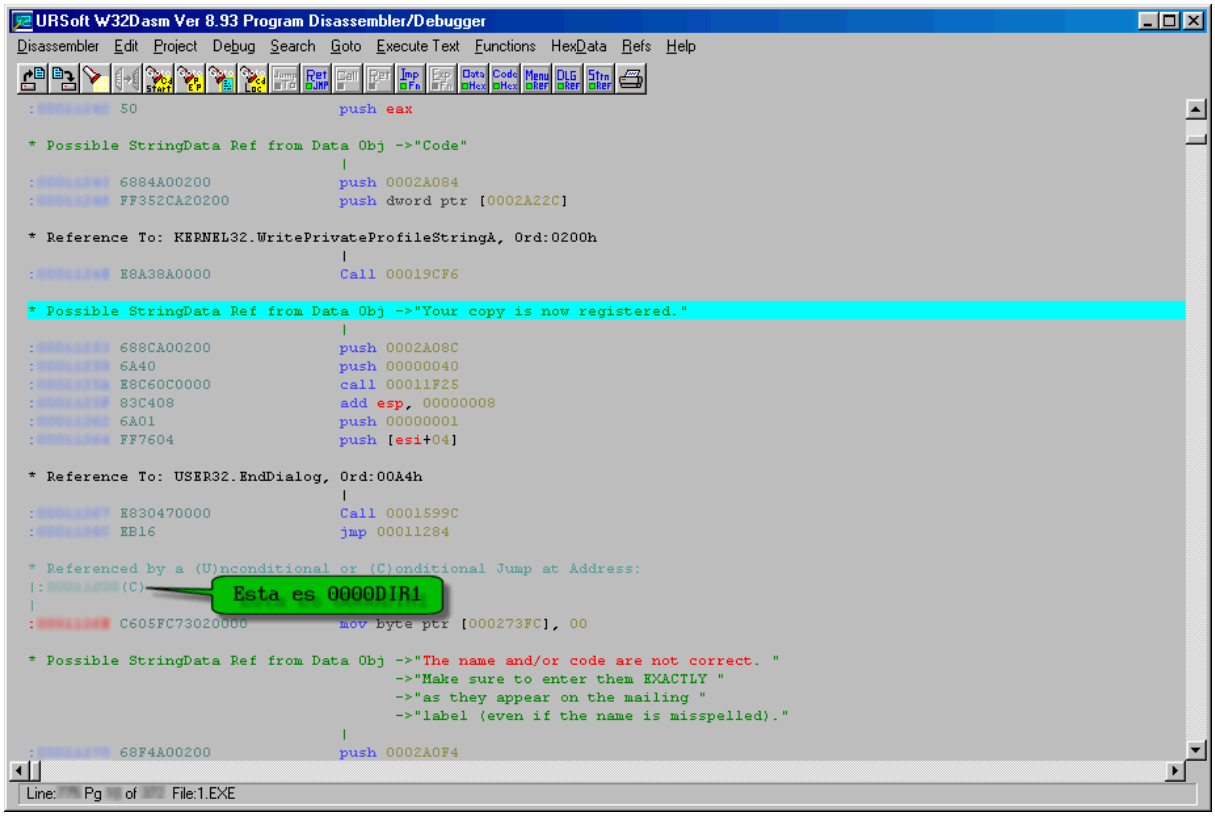


8. En TC, con el cursor situado en `1.exe` presionamos el botón W32, en unos segundos será desensamblado y veremos la pantalla de W32 que podríamos maximizar para una mayor comodidad. (Si no aparece esa pantalla, revise la sección Configuración de W32Dasm enunciada líneas arriba)
9. Volvemos a nuestro nag, presionamos el botón `I Agree` ("Estoy de acuerdo") para poder cargar el programa y proseguir con el trabajo.
10. Busquemos alguna opción para registrarnos, al encontrarla y seleccionarla, escribimos lo que nos solicita, un nombre y un código. Nosotros siempre usamos `Furious Logic` como nombre y `777666777` como código respectivamente. Puede usar el suyo propio, siempre que no use el `123456789` ni el `999999999` porque ambos están muy trillados. Use combinaciones de caracteres fáciles de recordar para usted. Sea creativo es este aspecto.
11. Al aceptar, aparece el típico mensaje de error. Anotemos sus primeras palabras:

    `The name and/or code are not correct...` ("el nombre y/o código no es correcto")

12. Ahora, en W32 seleccione menú `Search`, opción `Find text` y aparecerá un cuadro de diálogo en donde debemos escribir el texto que acabamos de anotar en nuestra hoja borrador. Luego aceptamos para que se de inicio a la búsqueda.
13. Aparecerá la primera coincidencia, entonces pulsamos `Find next` para averiguar cuántas coincidencias adicionales existen. Si no existen más ocurrencias, cerramos el diálogo de búsqueda porque hemos hallado el punto de partida para nuestro trabajo.
14. Las líneas de código son las que empiezan con un número cuyo prefijo es ":" (dos puntos).
15. Con las flechas de cursor del teclado, subiremos lentamente unas pocas líneas y encontramos otro mensaje interesante:

`Your copy is now registered.` ("Ahora su copia está registrada")



16. A partir de ambos mensajes, el de error y el de éxito, podemos suponer que en alguna parte líneas arriba, debe existir una bifurcación controlada por un **Jxxxx** de cualquier tipo (**JNE**, **JE**, **JA**, **JB**, etc.).
17. Volvamos a la ubicación del mensaje de error y leamos el comentario que se encuentra sobre él: (Obviamente este tutorial NO les va a mencionar ninguna dirección, por si algún lamer intenta el camino fácil del plagio y esto no es digno de un under. Toda referencia a direcciones se sustituirá con **0000DIRn** (en donde **n** representa un número).

    ```
    * Referenced by (U)nconditional or (C) condicional Jump at Address
    |:0000DIR1 (C)
    ```

    Lo que nos está diciendo es que en esa dirección, en **0000DIR1**, hay una bifurcación condicional **Jxxxx** que nos trae a este mensaje de error.
18. La pregunta del millón es: ¿Qué pasaría si no ocurre la bifurcación hacia la dirección del mensaje de error, sino que el flujo del programa prosigue su curso normalmente? Averiguémoslo de inmediato.
19. Pulsamos **SHITF+F12** o seleccionamos el menú **Goto**, opción **Goto Code Location**. Escribimos **0000DIR1** (los ceros a la izquierda no son significativos) y aceptamos. Como la dirección está muy cerca, podríamos usar el teclado para ubicarlo algunas líneas arriba. Practique ambos métodos.
20. Ahora que hemos encontrado la dirección del salto, nos situamos en ella. Una barra de color verde claro debe estar marcándola. Allí pulsamos **CTRL+→** para verificar si efectivamente nos conduce al mensaje de error. Si es así, estaremos nuevamente en el famoso mensaje. **CTRL+←** para regresar al origen de la bifurcación.
21. Ahora, en lugar de bifurcar, bajaremos unas líneas de código para investigar qué sucede.
22. Mientras estamos bajando unas líneas, encontramos la referencia a la palabras como **Name**, **%d** y **Code**. Las anotaremos porque suenan interesantes. Podrían ser claves de registro o entradas de un archivo **.ini** tal vez. Eso es sumamente común, recuérdelo para futuros cracks.
23. Si bajamos aún más, volveremos a encontrar el mensaje de felicitación por nuestro registro. ¡Excelente! Ése es el camino a seguir. Es tiempo de "corregir aquel error" con Hiew. Recuerde anotar nuestra valiosa dirección **0000DIR1**.
24. Pero antes, debemos hacer algunas minúsculas anotaciones adicionales. Un corto descanso que aprovecharemos en repasar una pequeña tabla que ya conocen hasta el cansancio. Hemos

agrupado los códigos de significado similar en secciones separadas y también agregamos sus equivalentes hexadecimales y matemáticos cuando corresponde.

Jxxxx	HEXADECIMAL		EXPLICACIÓN	EQUIV.
JE	74xx	0F84yy	*Jump if equal*, bifurcar si es igual (ZF=1).	=
JZ	74xx	0F84yy	*Jump if zero*, bifurcar si ZF=1.	
JNE	75xx	0F85yy	*Jump if not equal*, bifurcar si no es igual (ZF=0).	<>
JNZ	75xx	0F85yy	*Jump if not zero*, bifurcar si ZF=0.	
JB	72xx	0F82yy	*Jump if below*, bifurcar si es inferior (CF=1).	<
JNAE	72xx	0F82yy	*Jump if not above nor equal*, bifurcar si no es superior ni igual (CF=1).	
JA	77xx	0F87yy	*Jump if above*, bifurcar si es superior (CF=0 y ZF=0).	>
JNBE	77xx	0F87yy	*Jump if not below nor equal*, bifurcar si no es inferior ni igual (CF=0 y ZF=0).	
JBE	76xx	0F86yy	*Jump if below or equal*, bifurcar si es inferior o igual (CF=1 ó ZF=1).	<=
JNA	76xx	0F86yy	*Jump if not above*, bifurcar si no es superior (CF=1 ó ZF=1).	
JAE	73xx	0F83yy	*Jump if above or equal*, bifurcar si es superior o igual (CF=0).	>=
JNB	73xx	0F83yy	*Jump if not below*, bifurcar si no es inferior (CF=0).	
JG	7Fxx	0F8Fyy	*Jump if greater*, bifurcar si es mayor (ZF=0 y SF=OF).	>
JNLE	7Fxx	0F8Fyy	*Jump if not less nor equal*, bifurcar si no es menor ni igual (ZF=0 y SF=OF).	
JL	7Cxx	0F8Cyy	*Jump if less*, bifurcar si es menor (SF<>OF).	<
JNGE	7Cxx	0F8Cyy	*Jump if not greater nor equal*, bifurcar si no es mayor ni igual (ZF<>OF).	
JGE	7Dxx	0F8Dyy	*Jump if greater or equal*, bifurcar si es mayor o igual (SF=OF).	>=
JNL	7Dxx	0F8Dyy	*Jump if not less*, bifurcar si no es menor (SF=OF).	
JLE	7Exx	0F8Eyy	*Jump if less or equal*, bifurcar si es menor o igual (ZF=1 ó SF<>OF).	<=
JNG	7Exx	0F8Eyy	*Jump if not greater*, bifurcar si no es mayor (ZF=1 ó SF<>OF).	
JC	72xx	0F82yy	*Jump if carry*, bifurcar si CF=1 (si hay acarreo).	No
JNC	73xx	0F83yy	*Jump if not carry*, bifurcar si CF=0 (si no hay acarreo).	No
JO	70xx	0F80yy	*Jump if overflow*, bifurcar si OF=1 (si hay desbordamiento).	No
JNO	71xx	0F81yy	*Jump if not overflow*, bifurcar si OF=0 (si no hay desbordamiento).	No
JS	78xx	0F88yy	*Jump if sign*, bifurcar si SF=1 (si hay signo).	No
JNS	79xx	0F89yy	*Jump if not sign*, bifurcar si SF=0 (si no hay signo).	No
JP	7Axx	0F8Ayy	*Jump if parity*, bifurcar si PF=1 (si hay paridad).	No
JPE	7Axx	0F8Ayy	*Jump if parity even*, bifurcar si hay paridad par (PF=1).	
JNP	7Bxx	0F8Byy	*Jump if no parity*, bifurcar si PF=0 (si no hay paridad).	No
JPO	7Bxx	0F8Byy	*Jump if parity odd*, bifurcar si hay paridad impar (PF=0).	
JCXZ	E3xx	No	*Jump if CX is zero*, bifurcar si CX=0.	No
JECXZ	E3xx	No	*Jump if ECX is zero*, bifurcar si ECX=0.	No
JMP	EBxx	Varias	*Jump*, bifurcar obligatoriamente.	No

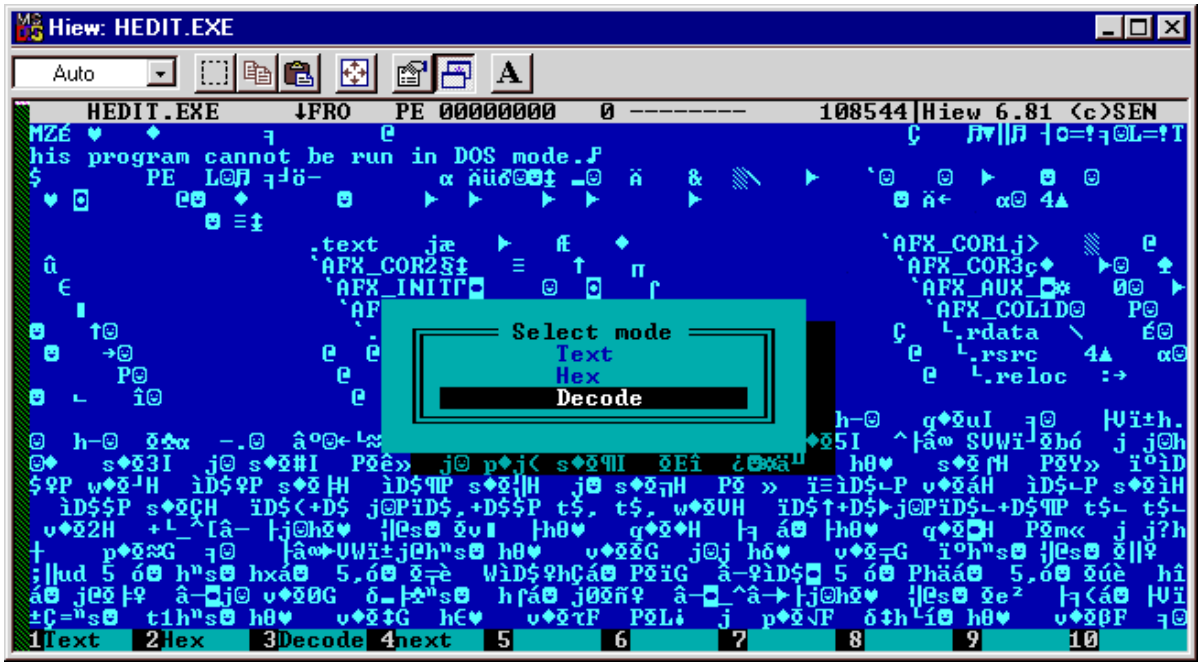
Dónde:

xx representa el desplazamiento relativo en 8 bits (byte)

yy representa el desplazamiento relativo en 16 bits (word) ó 32 bits (double word)

Es imprescindible aprender el uso de esta tabla. Para guiarnos, primero buscaremos el código mnemónico en la primera columna o en su defecto el código hexadecimal en la segunda columna para bifurcaciones tipo **short**, o en la tercera columna para tipo **near** (consulte su enciclopedia favorita de lenguaje assembler sobre tipos de bifurcaciones). Una vez encontrado, leeremos la explicación de la cuarta columna para saber qué hace esta instrucción. Seguido, averiguaremos su correspondiente equivalencia matemática en la quinta columna y buscaremos su instrucción complementaria. Así, si la bifurcación resultase ser equivalente a ">", buscaremos su complementaria, es decir "<=" y para terminar, anotamos su respectivo código hexadecimal.

25. Ya terminado el descanso, retornamos a la clase. Como ya tenemos todo listo, solo nos falta editar. Con el directorio de **Hedit 1.2** en el panel izquierdo, situamos el cursor de TC en **hedit.exe** y pulsamos en el botón que tenemos preparado para Hiew, botón Hiew.
26. De inmediato, aparece la pantalla de Hiew con su vista por defecto en modo texto. Con **F4** cambiamos a modo **decode**. También puede optar por pulsar **ENTER** repetidas veces para alternar entre los diferentes tipos de vista. Practique ambos métodos.



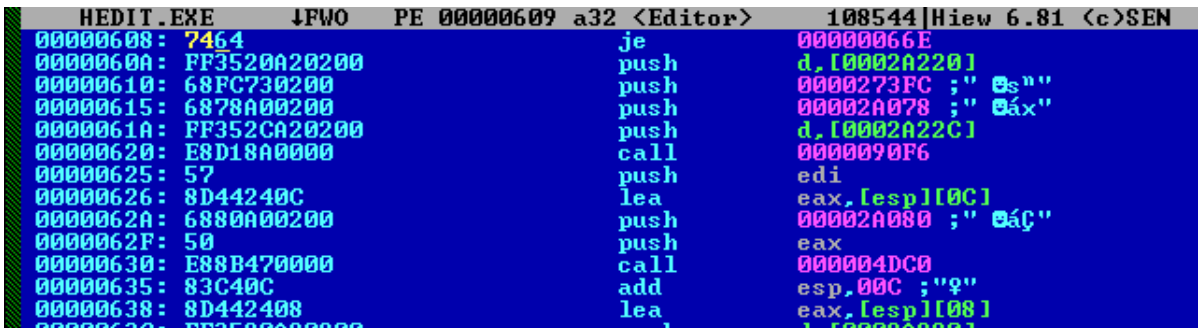
27. Ahora la pantalla nos mostrará un interesante listado desensamblado (Sí, son esos números de colores. ¡Qué bonitos!).
28. ¿Qué dirección vamos a editar? Revisamos nuestras anotaciones y la afortunada es **0000DIR1**. **''Elemental mi estimado cachorro''**. Y luego andan diciendo por allí que el cracking es difícil. Pulsamos **F5** (comando **Goto**, ir hacia) y escribimos **0000DIR2**. ¿Qué? ¿De dónde salió **0000DIR2**? Observación: Todos los manuales que hemos leído hasta el momento, concuerdan en que se debe hacer lo siguiente: (lea todo antes de continuar).

    En W32, marcar con la barra verde claro la instrucción a editar. Luego, en la barra de estado, leer el offset (desplazamiento) absoluto **0000DIR2** correspondiente a la dirección virtual **0000DIR1**. La dirección absoluta **0000DIR2** siempre aparece a continuación de la palabra **@offset** en el pie de la ventana o barra de estado.

    En Hiew, en modo decode, pulsar **F5** y escribir esa dirección absoluta **0000DIR2**.

    Eso es lo que comúnmente se enseña. No obstante, permítame decirle que usted NO necesita la dirección absoluta **0000DIR2**. Solamente necesita la dirección virtual ya anotada y denominada como **0000DIR1**. Simplemente, antes de escribirla ponemos "." (punto) y luego la dirección **0000DIR1**. El punto, le indica a Hiew que la dirección que se está especificando es virtual y no absoluta, como sucedería si no hubiese un punto precediéndola.

    Biew es un editor hexadecimal más amigable en este aspecto, porque nos proporciona una lista de opciones para seleccionar qué clase de dirección estamos escribiendo: **F2**, **Dissasembler**, **F5**, **0000DIR1**, **F2 (Virtual)**, **ENTER**.
29. Ahora que ya comprendimos, escribamos **.0000DIR1** en Hiew (Note el punto precedente) y ya nunca más nos preocuparemos por la dirección absoluta **0000DIR2**. Su función será necesaria solo para nuestro programa que automatice en crack. Presionamos **ENTER**.
30. **F3** para editar, escribimos el código hexadecimal complementario que previamente hemos extraído de la tabla de bifurcaciones arriba descrita.



Grabamos los cambios pulsando **F9**.

31. **ESC** ó **F10** para salir.
32. Antes que nada, copiamos el **hedit.exe** modificado, panel izquierdo de TC, hacia su directorio **Hedit 1.2** que está en parciales, con el nombre de **hedit parcial.exe**, panel derecho.
33. Ejecutamos el programa en cuestión, olvidándonos por ahora del fastidioso nag, buscamos la opción para registrarnos, rellenamos los datos que nos solicitan y listo, estamos registrados. Menú **Help, About** y nuestro nombre aparece allí.

No fue difícil ¿verdad?, excepto por tres insignificantes detalles:

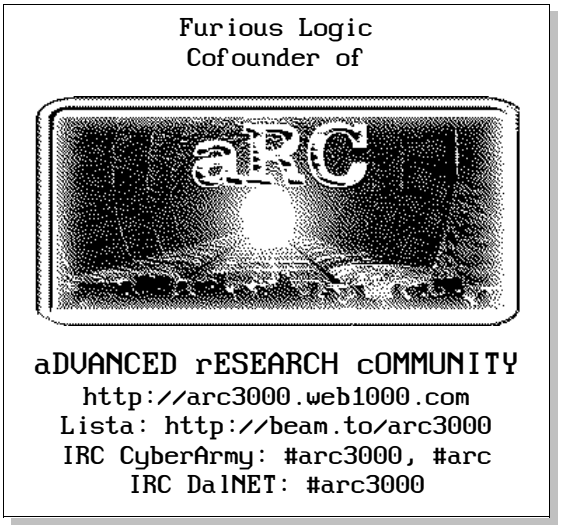
1. El nag sigue fastidiando.
2. Al salir y volver a cargar el programa e ir a **Help, About** dice: **Unregistered copy...** (¿?)
3. Cuando se ingresa un nombre y código que sí son válidos, el programa dice que el código no es correcto.

El crack está bien, pero debe ser mejorado. Esta es su "tarea para la casa". Como está en ciernes, puede utilizar la tabla de bifurcaciones equivalentes proporcionada. La siguiente lección contemplará estos "insignificantes detalles" que le animamos a resolver antes de leerla.

Derechos de autor

El autor puede ser contactado en:

IRC CyberArmy /server -m irc.cyberarmy.com: #arc3000, #arc
IRC DalNet: #arc3000
Email: furiouslogic@eml.cc



"Porque buscamos la libertad que sólo en el conocimiento podemos encontrar"