Politécnico costa atlántica
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

Manual de Técnicas Cracking y Anticracking De Software Para el Área de Desarrollo de la compañía Delaware

Manual of Cracking and Anticracking Software Techniques for the Development Area of the Delaware company

REVISTA GESTIÓN, COMPETITIVIDAD E INNOVACIÓN

Nilsan María Rojas Cubi *

Jhon Jairo Montejo Pérez **

Dougglas Hurtado Carmona ***

* Ingeniera de Sistemas de la Escuela Colombiana de Carreras Industriales, especialista en Gerencia de Proyectos en inteligencia de Negocios graduada en el 2021 de la universidad Politécnico Grancolombiano, certificada en SCRUM Fundamentals, SCRUM Master e ISTQB. Con 10 años de experiencia como consultor funcional, gestión de proyectos, gestión de base de Datos (Oracle, SQL y MySQL), conocimientos en TOGAF. Actualmente me desempeño como Ingeniera QA de Automatización en el Ministerio de Hacienda y Crédito Público. cubir_nm83@hotmail.com

** Ingeniero de Sistemas de la Escuela Colombiana de Carreras Industriales. Con 8 años de experiencia como Ingeniero de Desarrollo, manejo de metodología Scrum y gestión administrativa, gestión de bases de datos, modelado de sistemas, seguridad informática, módulo SAP HCM, análisis de datos (Data Science), programas de ofimática e inglés intermedio. jhonmontejo@hotmail.com

** Doctor en Gestión de la Innovación. Magister en Ingeniería de Sistemas y Computación, Ingeniero de Sistemas. Perito Informático. dougglas@gmail.com

Fecha de recepción: 15 de julio de 2020

Fecha de aceptación: 13 de diciembre de 2020

Citación:

Rojas Cubi, N. M., Montejo Pérez, J. J., y Hurtado Carmona, D. (2020). Manual de Técnicas Cracking y Anticracking De Software Para el Área de Desarrollo de la compañía Delaware. Gestión, Competitividad e innovación(Julio-Diciembre 2020), 20-30.

RESUMEN

Este documento tiene como objetivo principal mostrar las diferentes técnicas de craqueo y anti-craqueo más comúnmente utilizadas. El craqueo establece un método para probar la resistencia de un software frente a ataques y posible distribución ilegal. El cracking se puede describir como el conjunto de técnicas utilizadas para codificar, analizar y estudiar los principios de un programa sin tener el código fuente para el que existen programas capaces de hacerlo. Entre las técnicas más utilizadas tenemos el software de cifrado anticracking y el cifrado, se suele denominar programas de protección: empaquetadores. El uso de estos programas es una gran ventaja ya que se pueden crear licencias, crear programas DEMO o restricciones, crear protección contra depuradores y / o desensambladores, comprimir una aplicación, crear un solo ejecutable sin dependencias y algunas opciones dependiendo del empaquetador que se utilice.

Palabras Claves*: Técnicas, Cracking, Anticracking.*

ABSTRACT

This document is primarily intended to show the different cracking techniques most commonly used and anticracking. The cracking establishing a method for testing the resistance of a software from attacks and possible illegal distribution. The cracking can be described as the group of techniques used for encoding, analyzing and studying the principles of a program without having the source code for which there are programs capable of doing. Among the most commonly used techniques have anticracking encryption software and encrypted, it is usually called protective programs: packers. The use of these programs are a great advantage since it can create licenses, creating programs DEMO or restrictions, create protection against debuggers and / or disassemblers, compress an application, create a single executable without dependencies and a few options depending on the packer being used.

Keywords: Techniques, Cracking, Anticracking.

1. Introducción

Puede afirmarse en general que los crackers suelen ser estudiantes interesados en la tecnología de la información (si bien existen varias excepciones). Una vez que estas personas eligen su profesión en el mismo campo, normalmente también se especializan en algún otro campo similar, como la programación en ensamblador, la criptografía, etc.

Generalmente su conocimiento del cracking parte de su habilidad para programar en ensamblador, algunos de ellos desarrollan ambas aptitudes simultáneamente. Paradójicamente para muchos de los crackers más destacados, el cracking constituye tan solo una distracción, estudian algo completamente distinto (química, economía, etc.).

El cracking se puede describir como el grupo de técnicas empleadas para codificar, analizar y estudiar los principios de un programa sin disponer de su código fuente. Por ejemplo,

cuando un desarrollador crea un programa, comienza escribiendo el código fuente en el lenguaje de programación elegido para compilar y ejecuta. De esta manera nadie podría editar el programa sin disponer del código fuente y realizar de nuevo la compilación, pero esto es falso, y es precisamente en lo que el cracking se basa.

Una de las actividades más practicadas por los crackers será reventar las claves necesarias para obtener privilegios de root. De esta forma obtienen información privilegiada, la posibilidad de auto concederse cuentas con altos privilegios de uso e incluso la posibilidad de utilizar programas que no están habitualmente a disposición de cualquiera.

En un principio los programas solían venir grabados en disquete. Aunque no se crea, esto no lo ponía más fácil, al contrario. Muchos programadores incluían en sus juegos sistemas de protección, y es en el mundo de los juegos donde algunos de los mejores crackers comenzaron su carrera.

Un método de crack primitivo consistía en cambiar el copyright del programa para revenderlo. Esto era bastante fácil incluso hoy en día se realiza constantemente. Basta con usar un editor de archivo y cambiar el copyright escribiendo otro encima. Para evitar esto, un buen programador incluirá en sus programas un código CRC de validación.

Otro sistema más avanzado era el de la inclusión de protecciones comerciales, como GUARD o COPYPROTECT. Para poder reventar estos sistemas se necesitaba saber programación ensamblador o la utilización de un desprotector tipo copion. Hoy en día ya casi no se ven opciones por ahí. Hubo una época en que se veían mucho. Los había de dos tipos. Los que actuaban por software y los que actuaban por velocidad.

Otro tipo de protección era cambiar el número de sectores y de pistas del disquete haciendo que fallaran los programas de copias. Sin embargo, algunos copiones como el COPYWRIT permitían analizar el disco y formatearlo con esos sectores anómalos.

Como vera el arte del crackeo se basaba en el ingenio desde un principio, utilidades especiales y mucho método de prueba error. En la actualidad también se trata de saber programar. Un sistema actual que se utiliza para proteger el uso de programas es la limitación de fecha. Pasada una fecha determinada, el programa no se ejecuta. Este sistema y otros implican conocimientos en lenguaje máquina y por supuesto programas rastreadores.

Actualmente no es necesario que un pirata informático o cracker sea un experto en sistemas, sólo necesita un navegador para Internet y un conocimiento básico de SQL o manejo de scripts. Además, en Internet existen cientos, quizás miles, de sitios que ofrecen programas y scripts de piratería fáciles de descargar y de utilizar.

Si bien es cierto que Internet ha contribuido a una nueva forma de vida y de negocio, también ha impactado en la seguridad de las computadoras, particularmente por la velocidad que significa en la difusión de vulnerabilidades descubiertas y herramientas para su explotación. Muchas de estas vulnerabilidades provienen de fallas en el diseño de software, llámese sistemas operativos, manejadores de bases de datos, paquetería, programas de chateo, para escuchar música, ver videos, sistemas de uso en los teléfonos, equipos inalámbricos y dispositivos de red, entre otros.

Lo que antes se conocía como el perímetro de seguridad, actualmente ha cambiado en forma significativa por el tipo de acceso que se utiliza. En redes internas, el acceso siempre era desde la misma red dentro del edificio.

Hay empresas que no dan la importancia debida al establecimiento de políticas de seguridad. En caso de tenerlas, minimizan su aplicación estricta. Por otro lado, no son cuidadosas de monitorear las vulnerabilidades a las que se encuentran expuestas, poniendo en alto riesgo la información alojada en sus equipos, tales como de tarjetas de crédito, contable y de finanzas, del diseño de sus productos y servicios, de sus clientes y proveedores, etc. La falta de una cultura de seguridad informática en los empleados de la empresa es consecuencia de ello y de una pobre capacitación o entrenamiento en estos temas. Se piensa en términos de "gastos" y de no ser lo suficientemente grande o importante como para que alguien se fije en su empresa para atacarlos.

Prevenir contra los posibles ataques de los piratas informáticos. Realizar con cierta periodicidad análisis de vulnerabilidades de los sistemas que utiliza. Tomar algunas medidas efectivas para prevenirse también es el uso de sistemas personales de firewall, de detección de intrusos, antivirus y anti troyanos. .

2. Técnicas utilizadas en cracking y anti-cracking

Ingeniería inversa

La protección que utiliza Ingeniería Inversa es a partir del programa ya compilado, crear impedimentos al cracking. Esto se hace normalmente con programas protectores denominados: packers, empacadores. La utilización de estos programas son una gran ventaja ya que permiten crear licencias, crear programas DEMO o con restricciones, crear protección contra debuggers y/o desensambladores, comprimir tu aplicación, crear un sólo ejecutable sin tener dependencias y unas cuantas opciones que dependen del packer que estemos utilizando.

La utilización de los packers o programas tiene una gran ventaja: que en la mayoría de ellos se pueden ya crear licencias, versiones demo, protecciones contra desensamblados / debuggers, e incluso la disminución del tamaño de la aplicación considerablemente. Por contra, si quieres crear una aplicación comercial, en la mayoría de ellos deberás comprar el producto, lo que requiere que tu aplicación aumente también de precio.

También se debe tener en cuenta que los packers que hace unos años eran muy complicados, actualmente han sido analizados hasta su último bit y gente Newbie es capaz incluso de enfrentarse a este tipo de protecciones. Pero todo esto lo veremos con más detenimiento después. La técnica de la ingeniería inversa constituye la piedra angular del cracking, esta se basa en la descompilación, o compilación inversa, de un programa a un lenguaje de programación, generalmente el más básico es el ensamblador. Existen descompiladores capaces también de descompilar un programa a un lenguaje de programación de alto nivel.

Puede llegar a anularse una buena parte de los sistemas de protección si practicar ninguna modificación al programa: hallando la contraseña, el número de registro, etc., e incluso simplemente estudiando el código del programa.

La protección desde el mismo código fuente

La protección desde el mismo código fuente la realiza el programador. El programador según su propia experiencia, conocimiento de Ingeniería Inversa, imaginación será el que ponga las dificultades al cracking de su software. Crear licencias, programa DEMO y protección puede ser una labor muy costosa en tiempo y pruebas, pero también existe muchísima información en la red.

3. Herramientas de cracking

El procedimiento básico consiste en las descompilación de un programa en un lenguaje de programación, a ensamblador en la mayoría de las ocasiones.

El cracker podrá estudiar el programa de forma pasiva, desensamblarlo con un desensamblador (descompilador que efectúa una compilación inversa en ensamblador) de esta manera se obtiene su código ensamblador estático, o bien aplicar un programa de depuración, un debugger.

El cracker también puede hacer uso de un depurador. Al constituir uno de los pasos claves el uso frecuente de algún depurador (debugger) para realizar el análisis del programa, resulta pertinente intentar evitar su utilización. Razón que conduce a los infractores a aplicar distintos métodos y programas que enmascaren su depurador. Por ejemplo, un programa empleado para detectar un depurador se denomina «FrogsIce».

Cifrados mecánicos

El punto de inicio del cifrado mecánico este en la invención del telar mecánico por parte del francés Joseph Jacquard, que ya por entones usaba tarjetas perforadas para realizar determinadas operaciones. Las primeras computadoras, se diseñaron para reventar claves militares durante la segunda guerra mundial. Las primeras máquinas de cifrado utilizaban rotores, que eran cilindros mecánicos que giraban sobre un eje. En los rotores se distribuyen las letras del alfabeto. En cada posición de cada letra va un contacto que se comunica con el contacto de otra letra distinta en otro rotor distinto. El cifrado dependerá pues de la posición que tengan los rotores entre sí.

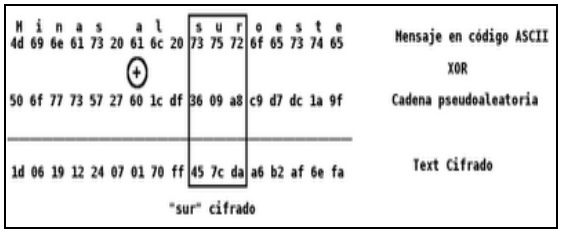
Los rotores no son ni siquiera un sistema moderno, ya que fueron inventados por el presidente norteamericano Thomas Jefferson, y resulta curioso que su invento no fuera adoptado por el ejército de EEUU hasta un siglo después. Mientras tanto, el gran impulsor del cifrado por cilindros fue el francés Etienne Bazeries1, uno de los más grandes criptográficos del Siglo XIX. La complejidad de las máquinas de cifrar como Enigma, radicaba en el número de rotores y las conexiones establecidas entre ellos.

Cifrados modernos

En cambio, los cifrados modernos se dividen en sistemas de clave privada y sistemas de clase pública. En los de clase publica se usa una clave para el cifrado, que es publica y otra el descifrado, que debe permanecer en secreto.

Tal vez el sistema más conocido de clave privada sea el sistema DES (Data Encryption Standard) que fue inventado por técnicos de IBM. Este sistema consta de 256 claves posibles. Hasta el día de hoy no ha sido posible reventarlo y los intentos no han sido

numerosos. Otro aspecto interesante de este sistema de cifrado es que es de tipo comercial, o sea, que, aunque sea muy poderoso usted puede conseguirlo en el mercado.



Etienne Bazeries, fue uno de los más grandes criptográficos del siglo XIX.

El texto cifrado es básicamente el mismo, pero concatenado con el cifrado del código CRC32 que calculamos. Nuestro atacante puede hacer ahora exactamente el mismo procedimiento anterior. Sin embargo, ahora cuando la tropa obtenga el texto falsificado, tendremos un código CRC32 con que validarlo:

```
>>> simple_xor(s+s_code,c).encode('hex')
'1d06191224070170ff457cdaa6b2af6efa6e43e6c4'
>>> cypher_text = simple_xor(s+s_code,c)
>>> cypher_text.encode('hex')
'1d06191224070170ff457cdaa6b2af6efa6e43e6c4'
>>> sur = "\x00" * 9 +"sur" + "\x00" * 9
>>> sur.encode('hex')
'000000000000000000737572000000000000000000'
>>> nor = "\x00" * 9 +"nor" + "\x00" * 9
>>> nor.encode('hex')
'000000000000000000006e6f72000000000000000000'
>>> simple_xor(simple_xor(cypher_text,sur),nor).encode('hex')
'1d06191224070170ff5866daa6b2af6efa6e43e6c4'
>>>
```

y al calcular el CRC32 del mensaje que modificó mi atacante, encontrará:

```
>>> mod_cypher = simple_xor(simple_xor(cypher_text,sur),nor)
>>> mod_cypher.encode('hex')
'1d06191224070170ff5866daa6b2af6efa6e43e6c4'
>>> decrypted = simple_xor(mod_cypher,c)
>>> decrypted.encode('hex')
'4d696e617320616c206e6f726f6573746568be0ef6'
>>> texto = decrypted[:-4]
>>> texto
'Minas al noroeste'
>>> crc_recibido = decrypted[-4:]
>>> crc_recibido.encode('hex')
'68be0ef6'
>>> crc_calculado = struct.pack("<I",CRC(texto,0xFFFFFFFF,0xFFFFFFFF))
>>> crc_calculado.encode('hex')
'12591ed2'
>>> crc_recibido == crc_calculado
False
>>>
```

El cual no es igual al CRC que enviamos en el texto cifrado. La tropa descartará el mensaje y todos felices. Sin embargo, resulta que, por las propiedades del algoritmo de CRC, mi atacante tiene una gran cantidad de posibilidades para atacar este nuevo esquema. El detalle de algunas de estas fórmulas de ataque será el tema de los siguientes artículos de esta serie.

El cifrado DES se basa en sucesivas operaciones matemáticas usando permutaciones y sustituciones de bits en función de la clave. En un principio hubo cierta prevención hacia este sistema por parte del gobierno de EEUU, pero luego se descubrió que con un computador potente y generando y probando un millón de claves por segundo, se tardarían más de 2000 años en reventar la clave original.

4. Métodos de protección y sus puntos débiles

A continuación, se describirán algunos de los métodos de protección de software más conocidos, señalando sus vulnerabilidades teniendo en cuenta que resulta prácticamente imposible crear una protección de software inevitable.

Cifrado

Consiste en emplear datos cifrados para descifrarse posteriormente tecleando un numero de registro (contraseña, etc.) sobre el que no se realiza ninguna comprobación, tampoco se realiza sobre la clave de descifrado (característica muy importante), si se produjera algún error, la clave generada seria errónea.

El único modo de sortear este modo de protección se reduciría a intentar un ataque masivo, lo que, suponiendo que se haya aplicado un buen algoritmo de cifrado, constituye un área casi insuperable. Si el programa emplea un algoritmo tipo RSA y un código de 2084 Bits, quedara protegido con seguridad incluso frente a entidades gubernamentales si bien se ha mejorado mucho el proceso de resolución del problema supone la factorización.

Desafortunadamente esta técnica encierra su propia debilidad. Resuelta valida solo hasta que el cracker consiga la clave. Tan pronto la obtenga de un conocido que se haya registrado o la encuentre en Internet, nada evitara que pueda guardar los datos de forma no cifrada.

Programas incompletos

Otra forma de evitar un programa consiste en retirar aquella parte del código que pudiera ser atacado. Resulta bien simple. Al crear una versión para demostración o de código compartido de un programa, resulta preferible empaquetar estas versiones distinguiéndolas de la versión íntegra del programa (y no al revés, donde la versión de demostración se convierte en la definitiva tras introducir el número de serie.). de este modo, se puede suprimir aquella parte del código de la que se pueda prescindir.

Cualquier programa en demostración o de código compartido (“shareware”) en el que se evite la presencia material del código constituye un programa incompleto. No hay forma de modificar algo que no existe en el programa incompleto. No hay forma de modificar algo que no existe en el programa. El cracker tendría que programa el código el mismo.

El usuario registrado normalmente recibe la versión íntegra del programa. Lo que, no obstante, exige un mecanismo de copia sofisticado para evitar que la versión completa del

programa se difunda ilegalmente. Numerosos programadores han perdido dinero por esta razón.

Tipos de protección disponibles

- Duración limitada.
- Número limitado de ejecuciones
- Otras restricciones numéricas
- Número de registro o contraseña-número de serie
- Archivo clave
- Programas limitados
- Clave hardware (“dongle”)
- Comprobación de la presencia del CDE
- Protección contra la copia del CD
- Protección comercial
- Compresores y codificadores para formato PE
- Programas en Visual Basic

Las técnicas de protección tienden a utilizarse de forma conjunta.

Registro interactivo

Durante los últimos años se viene apreciando una tendencia a realizar de forma interactiva los registros de todo tipo. Esta técnica es aplicable de muchas maneras. Desde el modo más simple de todos, donde el servidor comprueba que la información de registro resulta correcta, al más sofisticado, donde el código del programa nuevo se envía directamente al usuario vía internet.

Estos métodos representan un notable paso hacia delante en la lucha contra la piratería en el software. La parte más vulnerable de la protección, la comparación efectuada entre la información introducida y la referencia, se lleva a cabo en el servidor y no en el ordenador del cliente. A pesar de todo, el programa todavía resulta susceptible a muchas formas de ataque. Aunque dependa de los datos que se le envíen, el cracker puede sortear algunas restricciones. También sigue siendo muy difícil de resolver el problema de las copias de versiones del programa. Una vez que el programa guarda la información sobre su registro su registro ya efectuado, le hace vulnerable.

La mejor forma de protección interactiva consiste en habilitar este tipo de función directamente en Internet en vez del código del programa. Los grandes juegos en red o con servidores constituyen un buen ejemplo. Se exige el código de registro para acceder al servidor y jugar al juego interactivamente. Si el código introducido ya lo ha utilizado otro usuario o resulta incorrecto, es evidente que algún usuario no autorizado está intentado acceder al sistema.

Las protecciones interactivas también sirven para que las empresas controlen los registros realizados y vayan creando una base de datos sobre sus usuarios. No obstante, todo ello puede parecer injusto para quienes no tengan acceso a Internet. Si bien estos usuarios

pueden tener la voluntad de registrarse, se les fuerza a utilizar versiones piratas para evitar los problemas que conlleva el registro interactivo.

Archivo clave

Debido a que su ejecución resulta mucho más difícil, la protección mediante ficheros clave tiende a evitarse y olvidarse. Sin embargo, el invertir varias horas diseñando un buen algoritmo de protección basado en este método compensara el esfuerzo realizado.

Este método alternativo se asemeja en varias maneras a la protección mediante números de registro. Al igual que en este caso, no se trata de comprobar si es o no correcto el archivo clave, sino de utilizar la información contenida en el. La única gran ventaja del archivo clave consiste en que puede albergar una relativa gran cantidad de datos.

Al violar este tipo de protección, el cracker intentara principalmente reconstruir el archivo clave utilizando el código de programa, que a menudo resulta más revelador de su contenido y estructura de lo que sería deseable. Si se asume una buena programación que añada código nuevo al programa basado en esta técnica, volverá imposible la tarea de reconstrucción. O, con mayor exactitud, será posible siempre que el cracker reconstruya el código de programa el mismo. Por tanto, es ese caso el cracker habrá de optar por suprimir las restricciones del programa sin el archivo clave. Podrá suprimir otras protecciones, pero no será capaz de recuperar las funciones del programa.

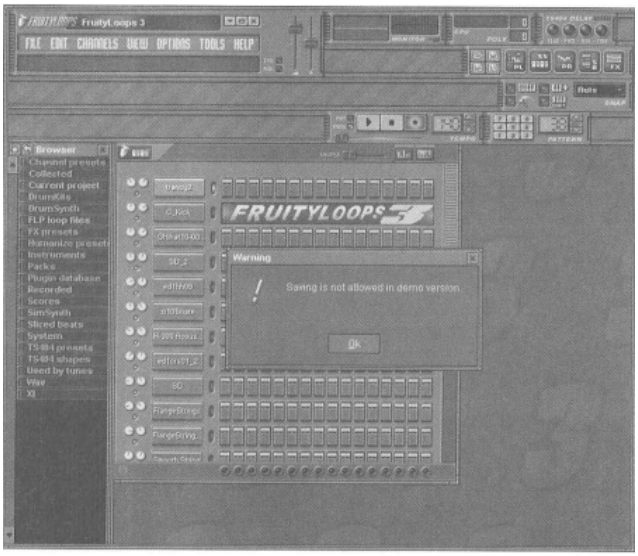
Los desarrolladores normalmente desconocen las posibilidades que brinda este tipo de protección, y así emplean el archivo clave para guardar información sobre el usuario. Sin embargo, esta no resulta una solución nada segura, con esta sencilla estructura, el archivo clave se podrá reconstruir sin grandes problemas. Recuerde: cuanto más complicada la estructura, mejor.

La protección mediante ficheros clave combinados con el uso de otras técnicas constituye, a mi parecer, uno de los métodos de protección más potentes para proteger el software eficazmente. Al igual que sucede con la protección mediante números de serie, debería controlarse la difusión de la información sobre registros ya efectuados, y de nuevo, una alternativa para alcanzar tal finalidad radica en vincular el registro con un ordenador especifico, que para llevarla a cabo, no debe olvidarse que, como sucede con otras protecciones contra el copiado, la necesidad de volver a registrarse al efectuar cambios en el hardware puede desaminar a los posibles usuarios a que compren el software.

Programas limitados

Existen muchas maneras de reducir la funcionalidad de los programas. De hecho, ciertas formas de protección. Pueden considerarse también programas limitados. Sin embargo, aquí me refiero más concretamente a aquellos programas que tienen distintos botones inhabilitados y opciones inaccesibles desde los menús.

El problema principal de este tipo de protecciones no radica en la característica de control en si, sino en el hecho de que dicho mecanismo olvida proteger el propio código del programa. Resulta obvio que de vez en cuando habrá que añadir el programa una o dos de las características deshabilitadas, de manera que el código del programa controlado por el mecanismo de protección nunca debe permanecer en el programa.



Resulta prácticamente imposible guardar nada con esta versión en demo de Fruity Loops.

Si la funcionalidad queda habilitada transcurridos unos segundos, resulta recomendable volver a comprobar que se cumple la condición para su activación o bien aplicar la estrategia recomendada en los tipos de protección anteriormente comentados – cifrar y posteriormente descifrar el código cumplidas las condiciones que permitan su uso, o procesarlos de alguna otra manera, si se dan dichas condiciones -.

No obstante, existen otras formas más sofisticadas de protección aplicando las estrategias ya recomendadas al describir los anteriores tipos de protección. La unidad de hardware puede vincularse directamente al programa, lo que significa que la protección no podrá anularse sin la unidad de hardware. Este parece el mejor modo de proteger el software mediante una unidad de hardware.

Otros tipos de protección más caras demuestran una norma: cuanto más trabaje con acierto el equipo de software en el programa, menos tendrán que preocuparse de proteger su producto ni comparar otro sistema de protección complementario, cuyo manejo desconocen, creyendo que cuanto más caro resulte el precio mejor la protección. Con todo, las empresas gastan una cantidad de dinero en sistemas de protección que no son capaces de explotar completamente.

Compresores y codificadores PE

Codificador y Compresor PE: Herramienta que permite codificar y comprimir (compresor PE) el código ejecutable del programa. La descodificación se realiza transparentemente en el momento de la ejecución y con ello se evita instalar otro programa. Con este método se dificulta enormemente la edición directa del código y otras técnicas de cracking.

Estas dos herramientas representan los métodos de protección frente al cracking más utilizados. Lista de funciones API más utilizadas para este propósito.

Tabla 1. Funciones API, utilizadas en la comprobación de CD

VALOR	SIGNIFICADO
DRIVE_UNKNOWN	No se puede determinar el tipo de unidad.
DRIVE_NO_ROOT_ DIR	El directorio raíz es incorrecto, por ejemplo, cuando no se ha montado ningún volumen en el directorio.
DRIVE_REMOVABLE	El disco se puede retirar de la unidad.
DRIVE_FIXED	El disco no se puede retirar de la unidad.
DRIVE_REMOTE	Disco remoto (red).
DRIVE_CDROM	CD-ROM
DRIVE_RAMDISK	Disco RAM (memoria).

Referencias

McClure, S. Scambray, J. y Kurtz, G. Hackers 3 Secretos y soluciones para la seguridad de redes. Madrid, España: McGraw Hill, Osborne. 2002.

Mitnick, K.D y Simón, W.L. El Arte de la Intrusión. Primera Edición, México D.F: Alfaomega Ra-Ma. Abril de 2007.

Raya, J.S. y Raya C. La seguridad de una red con Netware 5. Edición Original, México D.F: Alfaomega Ra-Ma. 2000.

Rodao, J.de M. Piratas Cibernéticos. Edición, México D.F: Alfaomega Ra-Ma. 2002

Shinder, D.L. Prevención de Delitos Informáticos. Edición, Madrid, España: Ediciones Anaya Multimedia. 2003

McClure, S. Scambray, J. y Kurtz, G. Hackers 3 Secretos y soluciones para la

seguridad de redes. Madrid, España: McGraw Hill, Osborne. 2002.

Zemanek, J. Cracking sin Secretos Ataque y Defensa de Software. Ra-Ma.