Ingeniería Inversa del Software

- Parte II -

La lógica como herramienta de ataque al software

Como ya comentaba en el primero de los artículos que componen esta colección sobre la ingeniería inversa del software (IIS para abreviar, pero no confundir con Internet Information Services), la lógica puede ser nuestra mejor aliada o nuestra más acérrima enemiga según quien la utilice y según sepa emplearla.

Hoy trataré de abrir un camino a la comprensión sobre como emplearía la lógica un cracker inteligente. De esta manera pretendo sembrar una serie de ideas que servirán para prevenirla más eficazmente. Su aspecto más amigable para nosotros, los programadores, no será tratado en profundidad hasta los últimos artículos, pero en todos ellos se irán desgranando ideas al respecto. Seamos pues sagaces, intentando cazarlas al vuelo según aparezcan, para que las captemos como conclusiones, mucho más perdurables en la memoria que las explicaciones sueltas que daré dentro de unos meses.

Antes de comenzar, aclararé que lo que se explicará aquí solo es un complemento a la IIS, sin formar parte de ella. Esta es una forma rápida, limpia e inteligente de atacar al software evitando aplicar las, a veces tediosas, técnicas de la IIS.

Estos otros métodos de ataque basados en la lógica y el sentido común, aunque solo se pueden aplicar de manera selectiva y en determinados casos, permiten tirar abajo las defensas de un programa sin tener que "pelear" con su código ensamblador. Para ello, simplemente se ataca al recurso de la máquina que sirve como base a la protección del software. De esta manera se evita agredir directamente al programa en sí, con todo lo que ello conlleva. A continuación se enumeran algunos de los tipos de software más fácilmente crackeables de esta manera, a la vez que se explica cual es la base de su defensa y cómo se puede llevar a cabo el ataque:

a) Trials de x dias: son aquellos programas que permiten ser usados en modo de prueba durante un periodo limitado de tiempo. La base de la

máquina que atacará un cracker ante este tipo de programa será el calendario del sistema. Basta un lanzador muy sencillito para hacer que un programa de este tipo funcione para siempre. En el siguiente cuadro se muestra y explica el código en VB .NET de este lanzador:

```
Dim hoy As Date 'VARIABLE GLOBAL DONDE ALMACENARÉ LA FECHA ACTUAL

'ESTA FUNCIÓN SE EJECUTA CON EL EVENTO DE INICIAR EL LANZADOR
Private Sub Form_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    hoy = Today() 'ALMACENO LA FECHA ACTUAL
    Today() = DateValue("23/11/2002") 'PONGO LA FECHA DEL DIA DE LA INSTALACION DEL PROGRAMA
    reloj.Interval = 32000 'ESTABLEZCO QUE SE RECUPERE LA FECHA TRAS 32 SEGUNDOS
    reloj.Enabled = True  'ACTIVO EL TIMER Y EMPIEZA A CONTAR EL TIEMPO
    Shell("C:\Archivos de programa\carpeta\programa.exe") 'LANZO EL PROGRAMA
End Sub

'PASADOS 32 SEGUNDOS SE DA EL EVENTO DEL TIMER CADUCADO:
Private Sub reloj_Elapsed(ByVal sender As System.Object, ByVal e As System.Timers.ElapsedEventArgs) Handles reloj.Elapsed
    Today() = hoy 'RECUPERO LA FECHA ACTUAL
    Me.Close() 'CIERRO EL LANZADOR
End Sub
```

b) Trials de x usos: este sistema de protección permite usar el programa a modo de prueba un número máximo de veces. El recurso a atacar es el contador de usos. Éste se halla, habitualmente, en un fichero de la carpeta del programa o en el registro de Windows. Como dato para encontrarlo se sabe el momento exacto en que se ha modificado por ultima vez. ¡¡¡Puede ser imposible de encontrar si el programador se ha molestado en esconderlo!!!

c) Programas con clave aleatoria: son aquellos que proporcionan un libro de claves o algún otro sistema caro de reproducir. Se le pide al usuario que introduzca una determinada palabra o clave diferente cada vez. Como todas las máquinas emplean el reloj como semilla de aleatoriedad, si se

diseña un lanzador similar al de los trials de x días pero alterando también la hora, se conseguirá que siempre pida la misma clave. Hay que tener en cuenta que la clave cambiará si hay otros programas que no sean los del sistema ejecutándose en segundo plano o si se cambia la velocidad de la máquina.

d) Sistemas de protección patentados: son aquellos que han sido desarrollados con ánimo de lucro, para venderlos a desarrolladores de software que quieran proteger sus propiedades sin perder el tiempo en diseñar una defensa propia. Como están patentados, su funcionalidad esta ampliamente descrita en el documento de la patente y es de dominio público, por lo que es fácil saber a que se debe atacar y como diseñar el crack. La pagina oficial de patentes de España se halla en la siguiente dirección: http://www.oepm.es/. Normalmente, cada vez que se crea alguno de estos métodos, en las revistas de informática suelen dar toda la información necesaria para saber en qué se basa el nuevo sistema anticopia y cómo combatirlo.

e) Cualquier otro sistema de protección siempre que la empresa desarrolladora informe sobre cómo lo ha implementado. Éste es, por ejemplo, el caso de Microsoft que cada vez que protege su software "saca pecho" por ello dando todos los detalles al respecto. El ejemplo más reciente ha sido el del Service Pack para Windows XP, del cual han dicho que incluye como seguridad una lista con todas los números de serie que se han distribuido por Internet. En fin, no hace falta ser un hacha para imaginar como saltarse la seguridad...

Como podemos ver, usando la lógica es muy posible saltar la seguridad de determinados programas de forma más sencilla que empleando la IIS ya que, como veremos en futuros artículos, aunque ésta se basa en ideas y conceptos sencillos, su aplicación se hace bastante pesada y engorrosa. Es por ello que los programadores deberemos de tener en mente la importancia de que la base de la seguridad de nuestro software debe ser desconocida para los demás. De no ser así, estaremos abriendo puertas ocultas a los invasores de nuestras creaciones.