Keygen para el KeygenMe#01 de eBuC

Comparación lineal

By deurus
15/11/2013

ÍNDICE

Equipo utilizado:

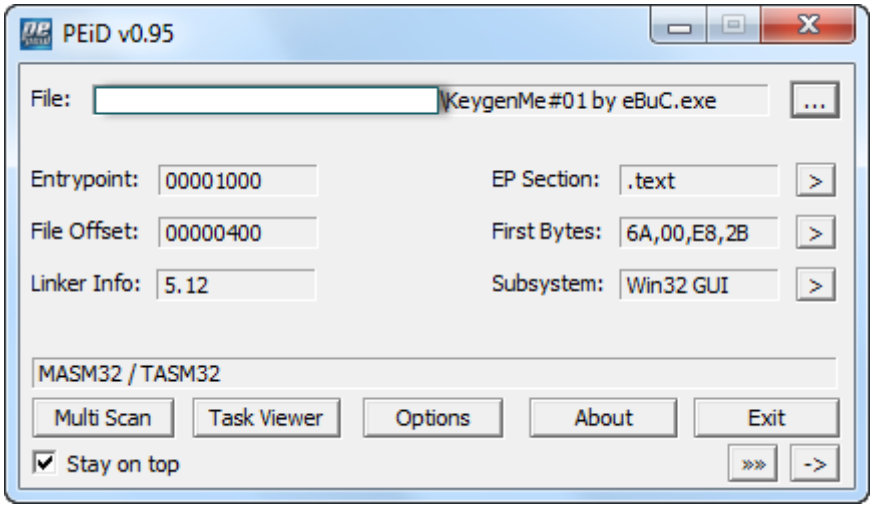
S.O: Windows 7 x32

Depurador: Ollydbg 1.10 (32bits) con plugins

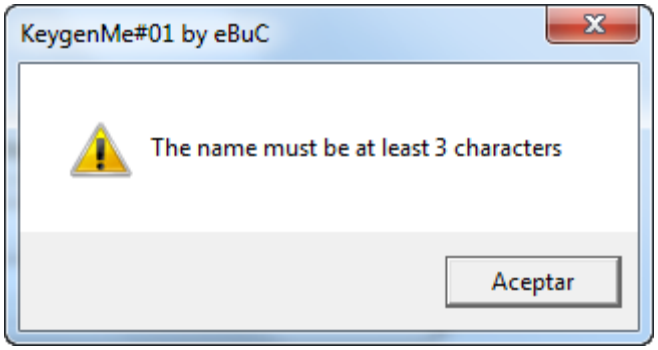
Analizador: PEiD 0.95

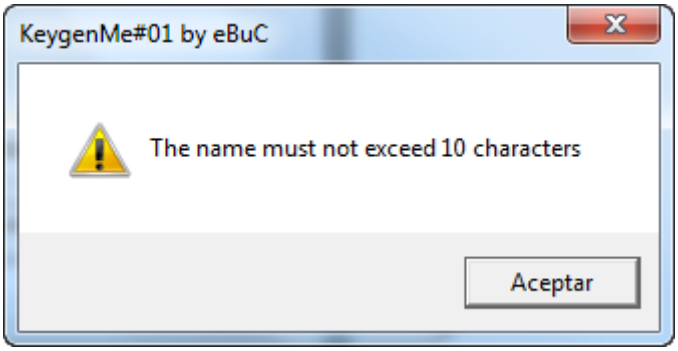
1. Primeras impresiones

Analizamos el programa con **PEiD** y nos muestra que está hecho en ensamblador.



Unas pruebas introduciendo datos nos muestran que el nombre debe tener entre 3 y 10 dígitos.

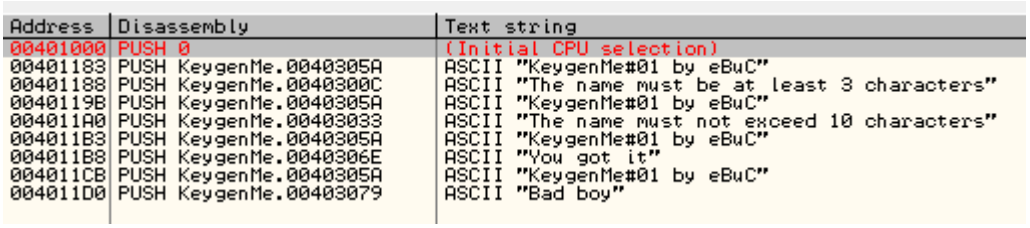




2. Determinando la rutina de creación del serial con Ollydbg

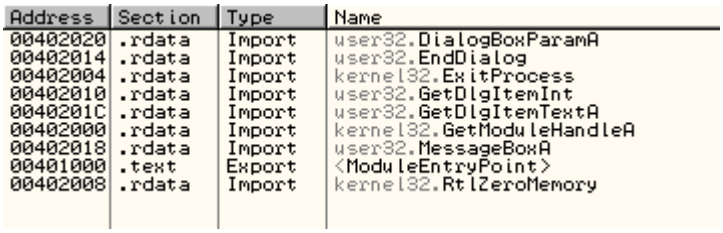
Llegados a este punto tenemos dos opciones que funcionan en el 90% de los casos. La primera es mediante las **referenced strings** o mediante los **names**.

Para el primer caso, con el keygenme cargado en olly, click derecho y **Search** > **All referenced text strings**. Haciendo doble click en "You got it" o en "Bad boy" vamos directamente a la rutina de comprobación del serial o muy cerca de ella en la mayoría de los casos.



Address	Disassembly	Text string
00401000	PUSH 0	(Initial CPU selection)
00401183	PUSH KeygenMe.0040305A	ASCII "KeygenMe#01 by eBuC"
00401188	PUSH KeygenMe.0040300C	ASCII "The name must be at least 3 characters"
0040119B	PUSH KeygenMe.0040305A	ASCII "KeygenMe#01 by eBuC"
004011A0	PUSH KeygenMe.00403033	ASCII "The name must not exceed 10 characters"
004011B3	PUSH KeygenMe.0040305A	ASCII "KeygenMe#01 by eBuC"
004011B8	PUSH KeygenMe.0040306E	ASCII "You got it"
004011CB	PUSH KeygenMe.0040305A	ASCII "KeygenMe#01 by eBuC"
004011D0	PUSH KeygenMe.00403079	ASCII "Bad boy"

Para el segundo caso, haremos click derecho y **Search** > **Name (label) in current módule**, o **Ctrl+N**. Vemos dos llamadas interesantes como son **user32.GetDlgItemInt** y **user32.GetDlgItemTextA**. Lo más seguro es que **user32.GetDlgItemInt** coja del textbox nuestro serial y **user32.GetDlgItemTextA** coja nuestro nombre. Para este caso colocaríamos breakpoints en las dos llamadas.



Address	Section	Type	Name
00402020	.rdata	Import	user32.DialogBoxParamA
00402014	.rdata	Import	user32.EndDialog
00402004	.rdata	Import	kernel32.ExitProcess
00402010	.rdata	Import	user32.GetDlgItemInt
0040201C	.rdata	Import	user32.GetDlgItemTextA
00402000	.rdata	Import	kernel32.GetModuleHandleA
00402018	.rdata	Import	user32.MessageBoxA
00401000	.text	Export	<ModuleEntryPoint>
00402008	.rdata	Import	kernel32.RtlZeroMemory

En mi caso elijo la primera opción. Nada más pulsar en "You got it" nos fijamos un poco más arriba y vemos las funciones donde coge el nombre y el serial y a simple vista se ven las operaciones que hace con ellos.

3. Generando un serial válido

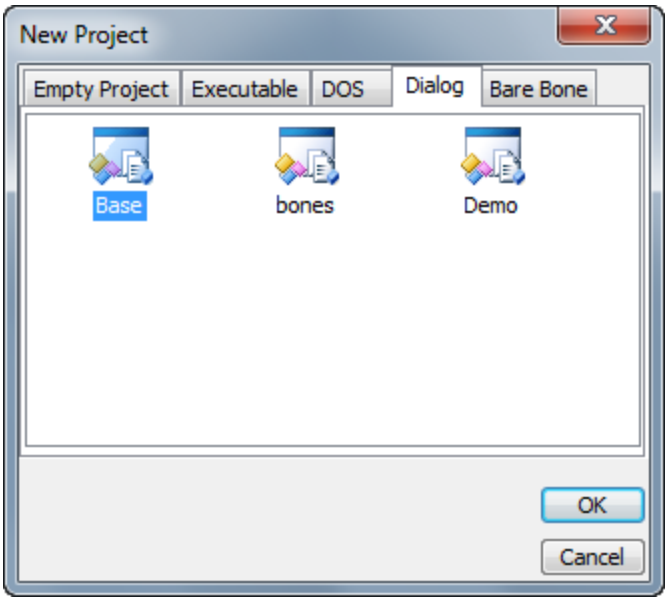
Como se muestra en la imagen siguiente, la creación del serial es muy sencilla y al final la **comparación es lineal** ya que se compara nuestro serial con el serial válido. Veamos el serial válido para el usuario "abc" cuyos dígitos en hexadecimal son 0x61, 0x62 y 0x63.

Letra a	**Letra b**	**Letra c**
Suma + 0x61	Suma + 0x62	Suma + 0x63
Suma * 0x20	Suma * 0x20	Suma * 0x20
Suma xor 0xBEFF	Suma xor 0xBEFF	Suma xor 0xBEFF
Suma / 4	Suma / 4	Suma / 4
Suma = 0x2CB7	**Suma = 0x14777**	**Suma = 0xA116F**
Suma xor 0xBEA4 = **0xAAFCB**		
Serial válido = **700363**		

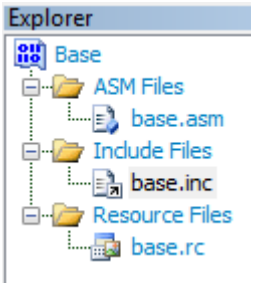
```
Address   Hex dump     Disassembly                          Comment
004010A4  .  FF75 08   PUSH DWORD PTR SS:[EBP+8]            hWnd = 7FFD4000
004010A7  .  E8 74010000 CALL <JMP.&user32.GetDlgItemTextA> GetDlgItemTextA
004010AC  .  83F8 03   CMP EAX,3                            ;>>>Nombre min 3 dÝgitos
004010AF  .v 0F82 CC000000 JB KeygenMe.00401181
004010B5  .  83F8 0A   CMP EAX,0A                           ;>>>Nombre max 10 dÝgitos
004010B8  .v 0F87 DB000000 JA KeygenMe.00401199
004010BE  .  8BF8      MOV EDI,EAX                          kernel32.BaseThreadInitThunk
004010C0  .  33C0      XOR EAX,EAX                          kernel32.BaseThreadInitThunk
004010C2  .  33C9      XOR ECX,ECX
004010C4  .  33D2      XOR EDX,EDX                          KeygenMe.<ModuleEntryPoint>
004010C6  .  C685 00FFFFFF 00 MOV BYTE PTR SS:[EBP-100],0
004010CD  .  C685 00FEFFFF 00 MOV BYTE PTR SS:[EBP-200],0
004010D4  >  0FB68A 94304000 MOVZX ECX,BYTE PTR DS:[EDX+403094] ;>>>Coge la letra del nombre que toque
004010DB  .  018D 00FFFFFF MOV DWORD PTR SS:[EBP-100],ECX   ;>>>Suma + ECX
004010E1  .  3E:8B85 00FFFFFF MOV EAX,DWORD PTR DS:[EBP-100] ;>>>EAX = Suma
004010E8  .  C1C0 05   ROL EAX,5                            ;>>>EAX * 32
004010EB  .  35 FFBE0000 XOR EAX,0BEFF                      ;>>>EAX xor 0xBEEF
004010F0  .  C1E8 02   SHR EAX,2                            ;>>>EAX / 4
004010F3  .  3E:8985 00FFFFFF MOV DWORD PTR DS:[EBP-100],EAX ;>>>Suma = EAX
004010FA  .  42        INC EDX
004010FB  .  3BD7      CMP EDX,EDI
004010FD  .^ 7C D5     JL SHORT KeygenMe.004010D4           ;>>>Salta hasta que recorra todo el nombre
004010FF  .  8B85 00FFFFFF MOV EAX,DWORD PTR SS:[EBP-100]   ;>>>EAX = Suma
00401105  .  35 A4BE0000 XOR EAX,0BEA4                      ;>>>Suma xor 0xBEA4
0040110A  .  8D38      LEA EDI,DWORD PTR DS:[EAX]           ;>>>Copia a EDI el serial valido (Suma)
0040110C  .  33C0      XOR EAX,EAX                          kernel32.BaseThreadInitThunk
0040110E  .  68 00010000 PUSH 100                           lIsSigned = TRUE
00401113  .  8D85 00FEFFFF LEA EAX,DWORD PTR SS:[EBP-200]
00401119  .  50        PUSH EAX                             pSuccess = kernel32.BaseThreadInitThunk
0040111A  .  68 EA030000 PUSH 3EA                           ControlID = 3EA (1002.)
0040111F  .  FF75 08   PUSH DWORD PTR SS:[EBP+8]            hWnd = 7FFD4000
00401122  .  E8 F3000000 CALL <JMP.&user32.GetDlgItemInt>   GetDlgItemInt
00401127  .  8D30      LEA ESI,DWORD PTR DS:[EAX]           ;>>>Copia a ESI el serial introducido
00401129  .  3BF7      CMP ESI,EDI                          ;>>>Comparacion del serial valido con el introducido
0040112B  .v 0F85 98000000 JNZ KeygenMe.004011C9            ;>>>Si no son iguales "Bad boy"
00401131  .v EB 7E     JMP SHORT KeygenMe.004011B1          ;>>>"You got it"
```

4. Generando un keygen con WinASM studio desde cero

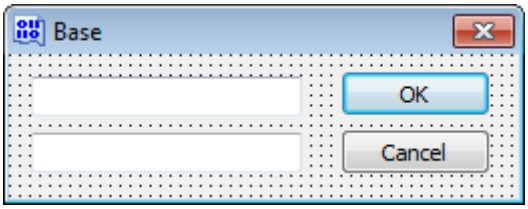
Abrimos WinASM studio y pulsamos en **File > New Project** y en la pestaña **dialog** elegimos **base**.



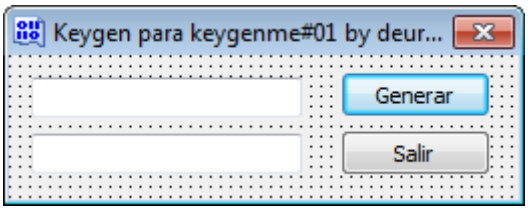
Vemos que se nos generan tres archivos, uno con extensión **asm**, otro con extensión **inc** y otro con extensión **rc**. El archivo **asm** es el que contendrá nuestro código. El archivo **inc** no lo vamos a usar para simplificar las cosas y el archivo **rc** es nuestro formulario al que pondremos a nuestro gusto.



Empecemos con el aspecto del formulario. Por defecto viene como se muestra en la siguiente imagen. Que por cierto, es todo lo que necesitamos para un keygen básico.



Y el aspecto final:



Ahora veamos cómo viene nuestro archivo **asm** inicialmente y que haremos con él. En la siguiente imagen lo indico.

```
.486
.model  flat, stdcall
option  casemap :none   ; case sensitive

include     base.inc

.code
start:
    invoke  GetModuleHandle, NULL
    mov hInstance,  eax
    invoke  DialogBoxParam, hInstance, 101, 0, ADDR DlgProc, 0
    invoke  ExitProcess,  eax
; ------------------------------------------------------------------------
DlgProc proc    hWin    :DWORD,
        uMsg    :DWORD,
        wParam  :DWORD,
        lParam  :DWORD

    .if uMsg == WM_COMMAND
        .if wParam == IDC_OK
; ------------------------------------------------------------------------
;           TODO
; ------------------------------------------------------------------------
        .elseif wParam == IDC_IDCANCEL
            invoke EndDialog,hWin,0
        .endif
    .elseif uMsg == WM_CLOSE
        invoke  EndDialog,hWin,0
    .endif

    xor eax,eax
    ret
DlgProc endp

end start
```

Crearemos un par de secciones con son .data y .data? y declararemos unas variables

Declararemos unas variables locales

Aquí irá nuestro código

Encima de la sección **.code** hemos creado dos secciones como son **.data** y **.data?** y hemos declarado las variables necesarias.

- **szFormat** está declarada en formato integer (%i). Más tarde la utilizaremos junto a la función **wsprintf** para dar formato a un número.
- **szSizeMin**: habla por sí misma.
- **szSizeMax**: habla por sí misma.
- **szCap**: habla por sí misma.
- **szName**: contendrá el nombre introducido.
- **szCode**: contendrá el serial válido.

Nuestro código queda de la siguiente manera:

```
.486
.model  flat, stdcall
option  casemap :none   ; case sensitive

include     base.inc

.data

szFormat    db "%i",0   ;integer
szSizeMin   db "Nombre min 3 digitos",0
szSizeMax   db "Nombre max 10 digitos",0
szCap       db "Keygen para el KeygenMe#01 de eBuC by deurus",0

.data?

szName db 256 dup(?)
szCode db 256 dup(?)

.code
start:
    invoke  GetModuleHandle, NULL
    mov hInstance, eax
    invoke  DialogBoxParam, hInstance, 101, 0, ADDR DlgProc, 0
    invoke  ExitProcess, eax
```

A partir de aquí ya simplemente es escribir el código necesario para generar el serial válido. Una de las ventajas que tiene el ensamblador para hacer keygens sin muchas complicaciones, es que prácticamente es copiar el código que nos muestra Ollydbg. Si os fijáis a continuación, en el botón llamado "IDC_OK" (no le he cambiado el nombre) he puesto todo el código necesario para generar la simple rutina del serial.

Como veis el bucle del nombre es una copia de lo que nos mostró Ollydbg. Una vez que tenemos en EAX nuestro serial válido, mediante la función **wsprintf** guardamos en la variable **szCode** el serial válido con formato integer. Finalmente mediante la función **SetDlgItemText**, mostramos el serial válido en la caja de texto 1002, que es la del serial.

```
; ------------------------------------------------------------------------
DlgProc proc    hWin    :DWORD,
        uMsg    :DWORD,
        wParam  :DWORD,
        lParam  :DWORD
        ;Mi variable local
        LOCAL suma[256]: byte

    .if uMsg == WM_COMMAND
        .if wParam == IDC_OK; Al pulsar el boton Generar se ejecutara esto
; ------------------------------------------------------------------------
        ;Pongo a cero los registro e inicializo las variables
        XOR EBX,EBX
        XOR ECX,ECX
        XOR EDX,EDX
        XOR EDI,EDI
        XOR ESI,ESI
        invoke RtlZeroMemory,addr suma,sizeof suma
        invoke RtlZeroMemory,addr szName,sizeof szName
        invoke RtlZeroMemory,addr szCode,sizeof szCode
        ;Cojo el nombre de la caja de texto 1001
        invoke GetDlgItemText,hWin,1001,addr szName,sizeof szName
        CMP EAX,3h
        JB @MinSize ;Salta si el nombre <3
        CMP EAX,0Ah ;Salta si el nombre >10
        JA @MaxSize
        MOV EDI,EAX
        XOR EAX,EAX
        XOR ECX,ECX
        XOR EDX,EDX
        MOV [suma],0
        ;
@buclenombre:
        MOVZX ECX,BYTE PTR DS:[EDX+szName]
        ADD dword ptr [suma],ECX            ;Suma + hex(letra)
        MOV eax, dword ptr ds:[suma]
        ROL eax,5                           ;Suma * 0x20
        XOR eax, 00000BEFFh                 ;Suma xor 0xBEFF
        SHR eax, 2                          ;Suma / 4
        MOV dword ptr ds:[suma], eax
        INC EDX
        CMP EDX,EDI
        jl @buclenombre
        ;-------------
        MOV eax,dword ptr[suma]
        XOR eax, 00000BEA4h                 ;Suma xor 0xBEA4
        invoke wsprintf, addr szCode, addr szFormat, eax    ;szCode = EAX(Suma) en formato integer
        invoke SetDlgItemText, hWin, 1002, addr szCode      ;Copio szCode al textbox 1002 (el del serial)
        RET

@MinSize:
invoke MessageBox,hWin,addr szSizeMin,addr szCap,MB_ICONEXCLAMATION
RET

@MaxSize:
invoke MessageBox,hWin,addr szSizeMax,addr szCap,MB_ICONEXCLAMATION
RET
; ------------------------------------------------------------------------
```

5. Enlaces

Crackme + Keygen + código fuente en ASM [9KB]

https://deurus.info/archivos/manuales/