

Programando em Assembly no Windows

+ Quando queremos executar uma função, podemos pesquisar pela sua documentação, executá-la em alguma linguagem, estudar seu modo de execução com o debugger e então montá-la no assembly

+ Queremos a função sleep

→ Pesquisamos no google "windows api sleep"

→ No site da docs.microsoft.com vemos o header que devemos passar caso queiramos usá-la em C, e vemos também a DLL que vai ser usada

+ No DEV-C++ criaremos um novo source file

file → new → source file

Lá faremos um código em C com a função sleep

```
#include <synchapi.h>

int main(){
    sleep(4000);
}
```

→ Ao executarmos esse código, vemos que ele demora 4 segundo para rodar

+ O objetivo agora é montá-la em assembly buscando ela diretamente da biblioteca em que está guardada no Kernel

→ Vamos então procurar a localidade (endereço de memória) em que a função está guardada de duas formas

1) Immunity Debugger

The screenshot shows the Immunity Debugger interface. The main window displays assembly code for the 'main' thread in the 'sleep' module. The code is disassembled from hex dump. The instruction at address 0040151D is highlighted: `MOV EAX, DWORD PTR DS:[&KERNEL32.Sleep]`. To the right, the 'Registers (FPU)' window shows the values of various registers. A context menu is open over the assembly window, showing options like 'Backup', 'Copy', 'Binary', 'Assemble', 'Label', 'Comment', 'Add Header', 'Modify Variable', 'Breakpoint', 'Hit trace', 'Run trace', 'New origin here', and 'Go to'.

Address	Hex dump	ASCII
00403000	0A 00 00 00 FF 00 00 00 FF FF FF FF FF FF FF FF
00403010	74 26 40 00 02 00 00 00 FF FF FF FF 00 25 40 00	t&E.0...
00403020	C0 25 40 00 4E E6 40 BB B1 19 BF 44 00 00 00 00	%E.NuE...1D...
00403030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

→ Na linha em que a função Sleep é chamada, vamos atualizar o código para começar



por ela e então apertar o duas vezes para que seja mostrado o endereço de memória (já é mostrado no canto inferior esquerdo em verde)

759811A0	FF25 6C0C9E75	JMP DWORD PTR DS:[&KERNELBASE.Sleep]	KERNELBA.Slee
759811A6	CC	INT3	
759811A7	CC	INT3	
759811A8	CC	INT3	
759811A9	CC	INT3	
759811AA	CC	INT3	
759811AB	CC	INT3	
759811AC	CC	INT3	
759811AD	CC	INT3	
759811AE	CC	INT3	
759811AF	CC	INT3	

→ Endereço 759811A0

(isso na biblioteca Kernel32.dll)

2) Via linha de comando

Usaremos o **arwin**

```
arwin.exe Kernel32.dll Sleep
```

```
C:\Users\catulo\AppData\Local\Temp\b43688c3-6591-4fe4-8bd6-ff7056eae3cd_arwin-master.zip\arwin-master>arwin.exe Kernel32.dll Sleep
arwin - win32 address resolution program - by steve hanna - v.01
Sleep is located at 0x759811a0 in Kernel32.dll
```

→ Mesmo endereço de memória acusado

+ Agora montaremos um código com essa informações

→ No notepad:

-----aguarda-----

global _main

section .text

_main:

```
xor eax, eax
mov eax, 9000
mov ebx, 0x759811a0
call ebx
```

→ Na linha de comando:

```
nasm -f win32 aguarda.txt
golink /console /entry _main aguarda.obj Kernel32.dll
```

→ Assim criamos o aguarda.exe que, quando executado, demora 9s

1) Diferença entre o assembly e o código em C pelo Immunity Debugger

→ Assembly

00401000	\$ 31C0	XOR EAX,EAX	
00401002	. B8 28230000	MOV EAX,2328	
00401007	. BB A0119875	MOV EBX,KERNEL32.Sleep	JMP to KERNE
0040100C	. FFD3	CALL EBX	Sleep
0040100E	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401010	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401012	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401014	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401016	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401018	. 0000	ADD BYTE PTR DS:[EAX],AL	
0040101A	. 0000	ADD BYTE PTR DS:[EAX],AL	
0040101C	. 0000	ADD BYTE PTR DS:[EAX],AL	
0040101E	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401020	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401022	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401024	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401026	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401028	. 0000	ADD BYTE PTR DS:[EAX],AL	
0040102A	. 0000	ADD BYTE PTR DS:[EAX],AL	
0040102C	. 0000	ADD BYTE PTR DS:[EAX],AL	
0040102E	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401030	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401032	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401034	. 0000	ADD BYTE PTR DS:[EAX],AL	
00401036	. 0000	ADD BYTE PTR DS:[EAX],AL	
759811A0=KERNEL32.Sleep (JMP to KERNELBA.Sleep)			

→ C

004014E0	\$ 83EC 0C	SUB ESP,0C	
004014E3	. C705 34504000	MOV DWORD PTR DS:[4050341],0	
004014ED	. E8 BE090000	CALL sleep.00401EB0	
004014F2	. 83C4 0C	ADD ESP,0C	
004014F5	. ^E9 86FCFFFF	JMP sleep.00401180	
004014FA	. 90	NOP	
004014FB	. 90	NOP	
004014FC	. 90	NOP	
004014FD	. 90	NOP	
004014FE	. 90	NOP	
004014FF	. 90	NOP	
00401500	\$ 8D4C24 04	LEA ECX,DWORD PTR SS:[ESP+4]	
00401504	. 83E4 F0	AND ESP,FFFFFFF0	
00401507	. FF71 FC	PUSH DWORD PTR DS:[ECX-4]	
0040150A	. 55	PUSH EBP	
0040150B	. 89E5	MOV EBP,ESP	
0040150D	. 51	PUSH ECX	
0040150E	. 83EC 14	SUB ESP,14	
00401511	. E8 7A090000	CALL sleep.00401E90	
00401516	. C70424 A00F0000	MOV DWORD PTR SS:[ESP],0FA0	
0040151D	. A1 30614000	MOV EAX,DWORD PTR DS:[<&KERNEL32.Sleep>]	Sleep
00401522	. FFD0	CALL EAX	
00401524	. 83EC 04	SUB ESP,4	
00401527	. B8 00000000	MOV EAX,0	
0040152C	. 8B4D FC	MOV ECX,DWORD PTR SS:[EBP-4]	
ESP=0061FF74			

2) Pelos tamanhos



aguarda



sleep

Propriedades de aguarda

Geral Compatibilidade Segurança Detalhes Versões Anteriores



aguarda

Tipo de arquivo: Aplicativo (.exe)

Descrição: aguarda

Local: C:\Users\catulo\Desktop

Tamanho: 1,00 KB (1.024 bytes)

Tamanho em disco: 4,00 KB (4.096 bytes)

Criado em: domingo, 21 de abril de 2024, 01:16:47

Modificado em: domingo, 21 de abril de 2024, 01:16:47

Acessado em: Hoje, 21 de abril de 2024, há 12 minutos

Atributos: ☐ Somente leitura ☐ Oculto Avançados...

OK

Cancelar

Aplicar

Propriedades de sleep

Geral Compatibilidade Segurança Detalhes Versões Anteriores



sleep

Tipo de arquivo: Aplicativo (.exe)

Descrição: sleep

Local: C:\Users\catulo\Desktop

Tamanho: 101 KB (103.833 bytes)

Tamanho em disco: 104 KB (106.496 bytes)

Criado em: sábado, 20 de abril de 2024, 15:28:41

Modificado em: sábado, 20 de abril de 2024, 15:28:41

Acessado em: Hoje, 21 de abril de 2024, há 6 minutos

Atributos: ☐ Somente leitura ☐ Oculto Avançados...

OK

Cancelar

Aplicar