

Executando Comandos SO em Assembly

IMPORTANTE

ascii hex

—

| ABCD 41424344 → D|C|B|A

| DCBA 44434241 → A|B|C|D

PUSH<

| .exe 2f657865 → exe.

|_ exe. 6578652f → .exe

→ o que chega na stack é o contrário do que sai do push

+ Vamos começar estudando em C pra depois executar em assembly em assembly

```
#include <windows.h>
int main() {
    system("cmd.exe /c whoami");
}
```

ou, para executar a calculadora:

```
#include <windows.h>
int main() {
    system("cmd.exe /c calc.exe");
}
```

→ Agora vamos pegar o código gerado e alocar no immunity debugger

The screenshot shows the Immunity Debugger interface. The main window displays assembly code for a program. The code includes standard Windows API calls like `system` and `cmd.exe`. The dump window at the bottom shows the memory dump for the `system` function, including the `ASCII` string `"cmd.exe /c calc.exe"`.

Address	Hex dump	ASCII
00404000	63 6D 64 2E 65 78 65 20 2F 63 20 63 61 6C 63 2E	cmd.exe /c calc.
00404010	65 78 65 00 80 15 40 00 55 6E 6B 6E 6F 77 6E 20	exe. [Unknown]
00404020	65 72 72 6F 72 00 00 00 5F 6D 61 74 68 65 72 72	error..._matherr
00404030	28 29 3A 20 25 73 20 69 6E 20 25 73 28 25 67 2C	(<): %s in %s(<g,
00404040	20 25 67 29 20 20 28 72 65 74 76 61 6C 3D 25 67	%g) (retval=%g
00404050	29 0A 00 00 41 72 67 75 6D 65 6E 74 20 64 6F 6D)...Argument dom
00404060	61 69 6E 20 65 72 72 6F 72 20 28 44 4F 4D 41 49	ain error (DOMAI
00404070	4E 29 00 41 72 67 75 6D 65 6E 74 20 73 69 6E 67	N).Argument sing
00404080	75 6C 61 72 69 74 79 20 28 53 49 47 4E 29 00 00	ularity (SIGN)..
00404090	4F 76 65 72 66 6C 6F 77 20 72 61 6E 67 65 20 65	Overflow range e
004040A0	72 72 6F 72 20 20 4F 56 45 52 46 4C 4E 52 20 00	uan (OVERFLOW)

→ Podemos ver exatamente o que ele está executando e qual dll está chamando

→ o caso de baixo, aplicamos o follow in dump


→ Veja o detalhe do alinhamento de 4 em 4 bytes

→ A dll que está sendo usada é a msvcrt, o que pode ser confirmado ao clicarmos no




E Executable modules					
Base	Size	Entry	Name	File version	Path
00400000	0001D000	004014E0	so		C:\Users\catulo\Desktop\so.exe
760D0000	0023A000	761EF950	KERNELBA	10.0.19041.4165	C:\Windows\System32\KERNELBASE.dll
76C70000	000BF000	76CA5AC0	msvcrt	7.0.19041.3636	C:\Windows\System32\msvcrt.dll
76E30000	000F0000	76E4F8E0	KERNEL32	10.0.19041.4165	C:\Windows\System32\KERNEL32.DLL
77510000	001A4000		ntdll	10.0.19041.4165	C:\Windows\SYSTEM32\ntdll.dll

→ Estabelecemos a linha selecionada como sendo a "new origin here"

→ Executamos o código com o  até que ele mostrasse as dll's usadas e seus respectivos endereços na memória

C CPU - main thread, module so					
004025F0	\$ FF25 B8614000	JMP	DWORD PTR DS:[&msvcrt.system]	msvcrt.system	
004025FE	90	NOP			
004025FF	90	NOP			
00402600	\$ FF25 80614000	JMP	DWORD PTR DS:[&msvcrt._lock]	msvcrt._lock	
00402606	90	NOP			
00402607	90	NOP			
00402608	\$ FF25 50614000	JMP	DWORD PTR DS:[&msvcrt.__dllonexit]	msvcrt.__dllonexit	
0040260E	90	NOP			
0040260F	90	NOP			
00402610	\$ FF25 88614000	JMP	DWORD PTR DS:[&msvcrt._unlock]	msvcrt._unlock	
00402616	90	NOP			
00402617	90	NOP			
00402618	\$ FF25 AC614000	JMP	DWORD PTR DS:[&msvcrt.signal]	msvcrt.signal	
0040261E	90	NOP			
0040261F	90	NOP			
00402620	> FF25 64614000	JMP	DWORD PTR DS:[&msvcrt.__setusermatherr]	msvcrt.__setusermatherr	
00402626	90	NOP			
00402627	90	NOP			
00402628	\$ FF25 90614000	JMP	DWORD PTR DS:[&msvcrt.fprintf]	msvcrt.fprintf	
0040262E	90	NOP			
0040262F	90	NOP			
00402630	\$ FF25 A0614000	JMP	DWORD PTR DS:[&msvcrt.fwrite]	msvcrt.fwrite	
00402636	90	NOP			
00402637	90	NOP			
00402638	\$ FF25 BC614000	JMP	DWORD PTR DS:[&msvcrt.vfprintf]	msvcrt.vfprintf	
0040263E	90	NOP			
0040263F	90	NOP			
00402640	\$ FF25 8C614000	JMP	DWORD PTR DS:[&msvcrt.abort]	msvcrt.abort	
00402646	90	NOP			
00402647	90	NOP			
00402648	\$ FF25 90614000	JMP	DWORD PTR DS:[&msvcrt.calloc]	msvcrt.calloc	
0040264E	90	NOP			
DS:[004061B8]=76CB3D30 (&msvcrt.system)					
Local call from 00401515					

→ Apertamos novamente o  e chegamos no ambiente em que podemos ver o endereçamento de memória da msvcrt.dll:

C CPU - main thread, module msvcrt			
76CB3D30	8BFF	MOV	EDI,EDI
76CB3D32	55	PUSH	EBP
76CB3D33	8BEC	MOV	EBP,ESP
		so.<ModuleEntryPoint>	

→ No caso, o endereço é 76CB3D30

→ Também podemos ver essa informação usando o arwin:

```
C:\Users\catulo\Downloads\arwin-master\arwin-master>arwin.exe msvcrt.dll system
arwin - win32 address resolution program - by steve hanna - v.01
system is located at 0x76cb3d30 in msvcrt.dll
```

→ Agora vamos estruturar o comando que queremos executar

→ Vamos converter o texto do comando para hexadecimal com um site qqr q faz isso

[ASCII to Hex | Text to Hex Code Converter \(rapidtables.com\)](https://www.rapidtables.com/convert/text/ASCII-to-hex.aspx)

→ Queremos passar o calc.exe (para iniciar a calculadora)

→ calc.exe00 = 63 61 6C 63 2E 65 78 65 00 → o 00 no final indica que o comandp terminou
63 61 6c 63 = calc
2e 65 78 65 = .exe

→ Para enviarmos, devemos mandar ao contrário:

00000000
2e 65 78 65 = .exe = 6578652e
63 61 6c 63 = calc = 636c6163

→ Vamos montar um programa que abra meu cmd:

cmd.exe = 63 6D 64 2E 65 78 65

exe0 = 65 78 65 00 = 00657865
cmd. = 63 6d 64 2e = 2e646d63

-----chama.txt-----

extern system ;extern é adicionado pois o golink não reconheceu a system, então anunciamos aqui dentro do código

global _main

section .text

_main:

```
push 0x00657865 ;00exe
push 0x2e646d63 ;.cmd
push esp ;aponta para o topo da stack
pop eax ;retira do topo da stack e manda para eax
push eax
mov ebx, 0x76cb3d30 ;system
call ebx ;chama a system
```

```
nasm -f win32 chama.txt
```

```
golink /console /entry _main chama.obj msvcrt.dll
```

→ Ao executar o chama.exe, ele abrirá um novo terminal

~~~~~  
Ativida do Longato → executar o cmd.exe /c calc.exe  
~~~~~

cmd.exe /c calc.exe = 63 6D 64 2E 65 78 65 20 2F 63 20 63 61 6C 63 2E 65 78 65

cmd. = 63 6d 64 2e
exe = 65 78 65 20
/c c = 2f 63 20 63
alc. = 61 63 6c 2e
exe00 = 65 78 65 00

Invertendo:

00 65 78 65

2e 63 6c 16
63 20 63 2f
20 65 78 65
2e 64 6d 63

-----lon.txt-----

extern system
global _main
section .text

_main:
 push 0x00657865 ;00exe
 push 0x2e636c61 ;.cla
 push 0x6320632f ;c c/
 push 0x20657865 ; exe
 push 0x2e646d63 ;.dmc
 push esp ;aponta para o topo da stack
 pop eax ;retira do topo da stack e manda para eax
 push eax
 mov ebx, 0x76cb3d30 ;system
 call ebx ;chama a system
