

Analisando um Buffer Overflow

+ Vamos entender na prática o esquema da stack montado na aula passada



-----check.c-----

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]){

    char nome[16];

    strcpy(nome, argv[1]);

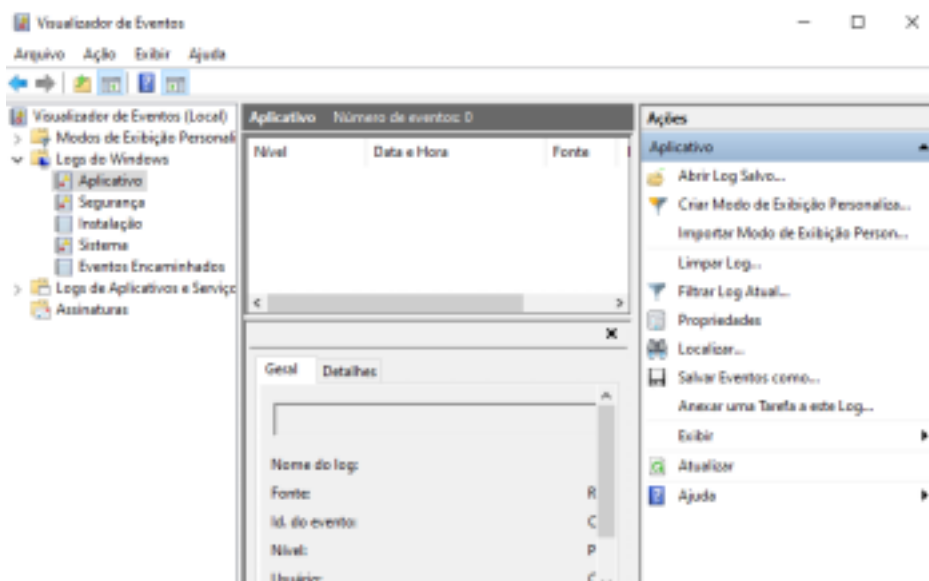
    return 0;

}
```

- esse programa montado em C irá apenas recebe um argumento de no máximo 16 bytes
- A função strcpy pega o segundo parâmetro e associa ao primeiro, que é a variável

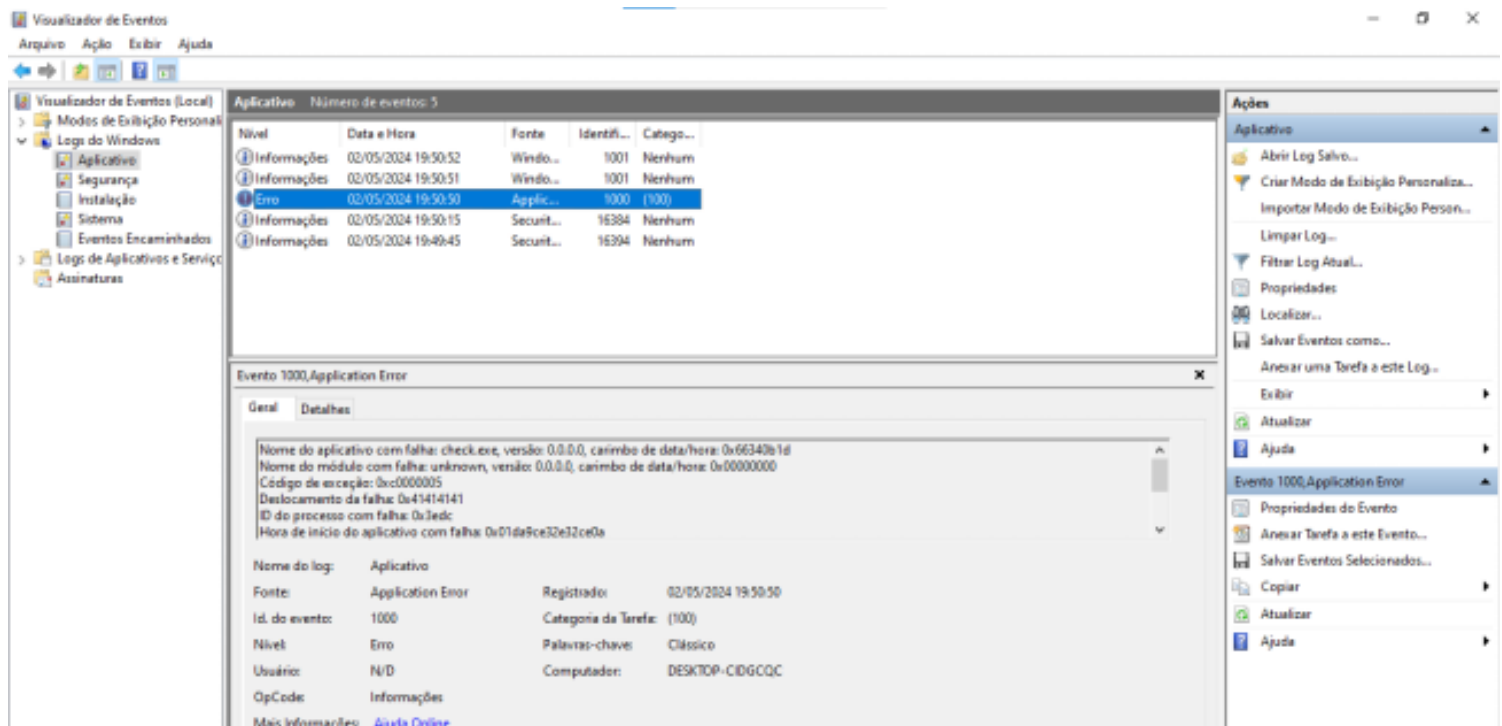
+ Vamos ligar o visor de eventor do windows. Devemos limpá-lo para que possamos ver o erro acontecendo quando passamos um argumento muito grande

check.exe RICARDOLONGATTO



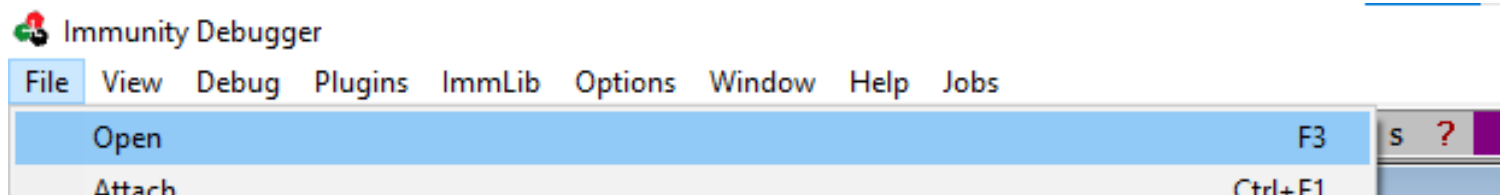
→ Nada apareceu, então foi executado com sucesso

check.exe RICARDOLONGATTOAAAAAAAAAAAAAAAA

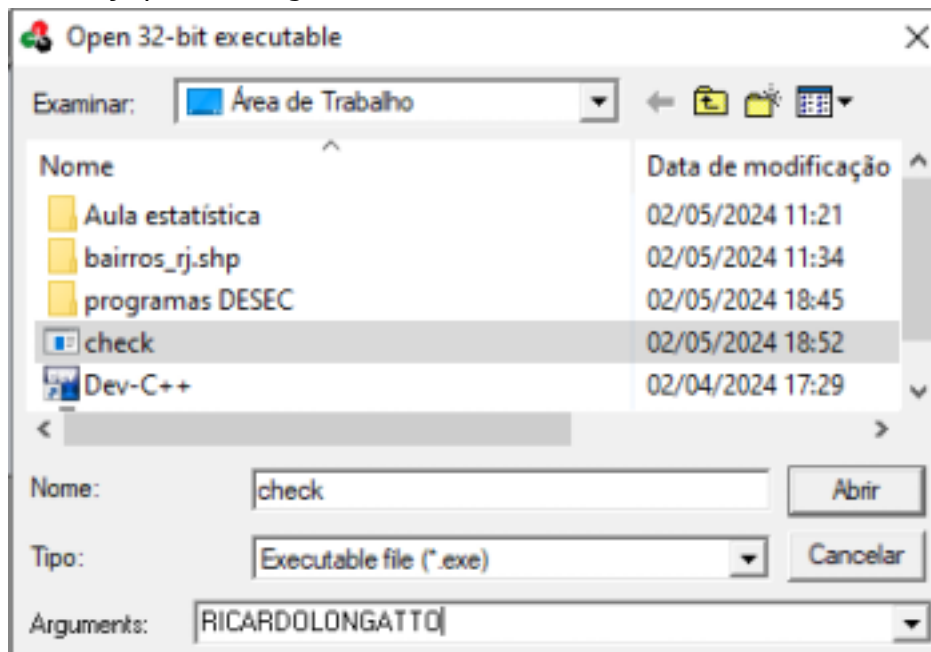


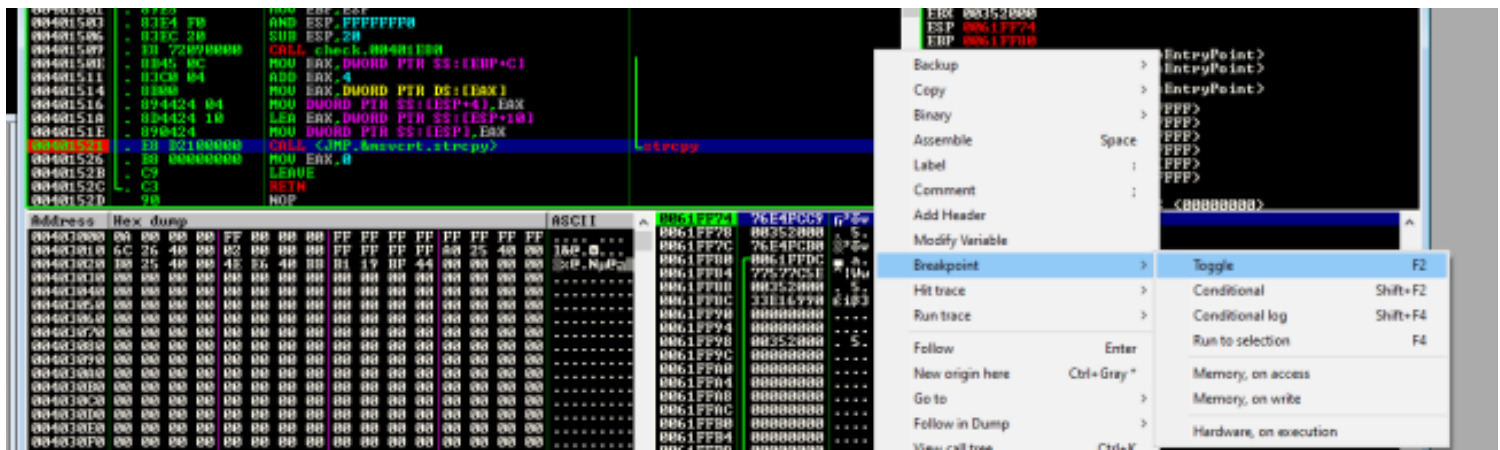
→ Esse deslocamento de falha é o que caracteriza o buffer overflow

+ Agora vamos começar a análise do programa no Immunity Debugger




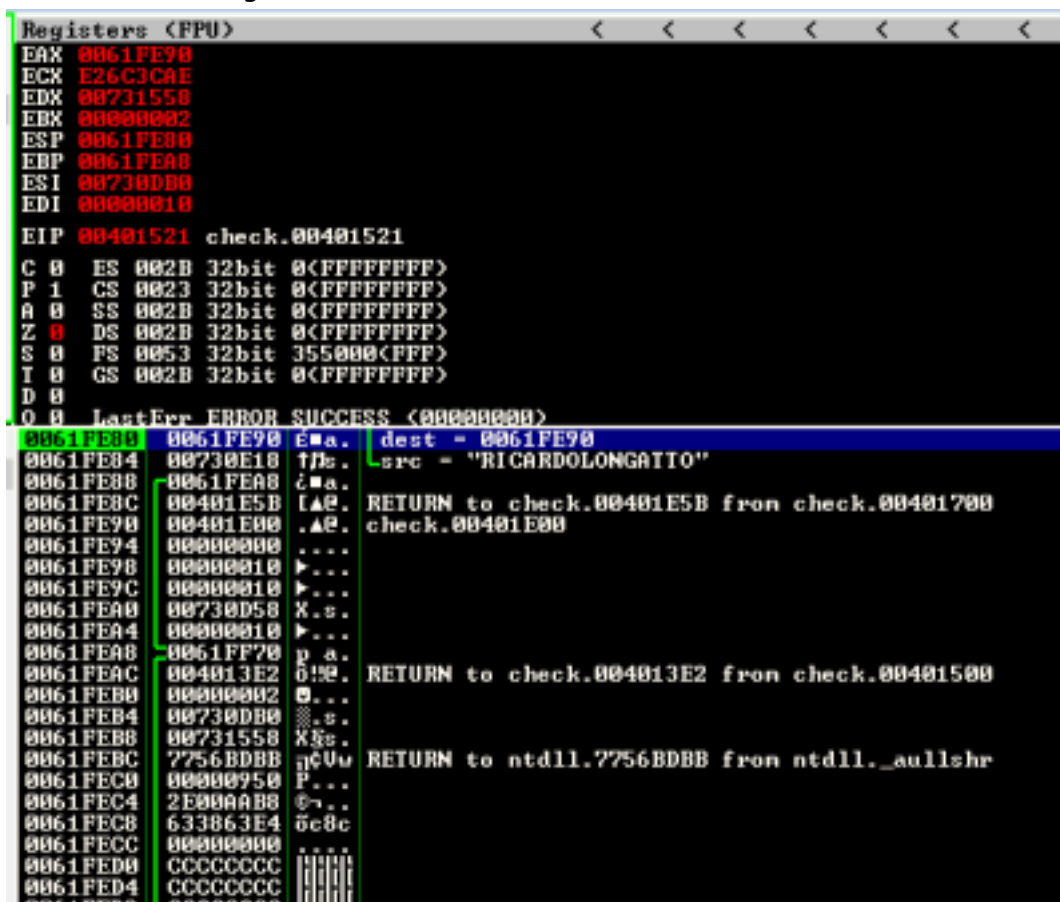
→ Para já passar o argumento:



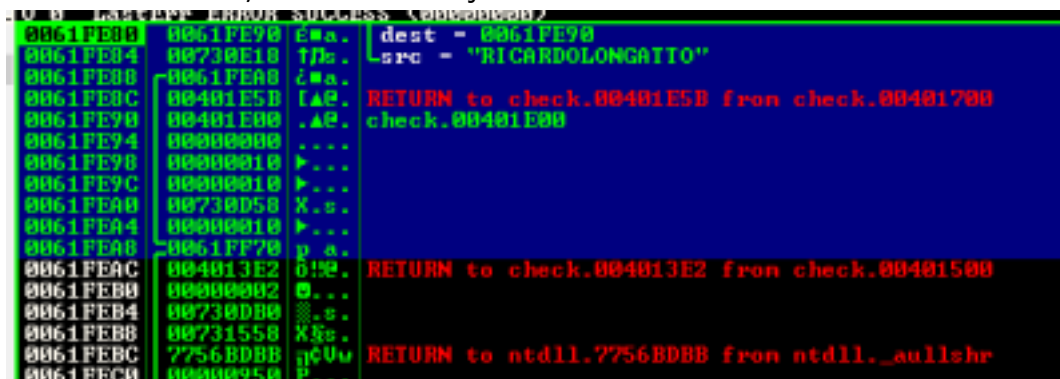


→ Colocamos um breakpoint na função `strcpy` pra que pudéssemos ver a recepção das variáveis setadas

- Ao apertamos o play , o programa para no breakpoint
- Apertamos duas vezes o Fn+F7 para que ele entre na função
- Analisaremos agora a stack:



- Veja que o ESP, que aponta para o topo da stack está registrando o 0061FE80
- A base da stack é o EBP: 0061FEA8
- Abaixo da stack, temos o endereço de retorno: 0040113E2



- Stack está em azul
- Entre o topo e a base teremo o buffer
- Se dermos um **follow in dump** veremos os dados setados

0061FE80	0061FE90	E=a.	dest = 0061FE90
0061FE84	00730E18	↑f.s.	src = "RICARDOLONGATTO"
0061FE88	0061FEA8	↓m.a.	
0061FE8C	00401E5B	[ΔC.	RETURN to check.00401E5B from check.00401700
0061FE90	00401E00	.ΔC.	check.00401E00
0061FE94	00000000	
0061FE98	00000010	↑....	
0061FE9C	00000010	↑....	
0061FEA0	00730D58	X.s.	
0061FEA4	00000010	↑....	
0061FEA8	0061FF70	p.a.	
0061FEAC	004013E2	0!C.	RETURN to check.004013E2 from check.00401500
0061FEB0	00000002	0...	

- Em dest temos o destino onde será alocada a informação: 0061FE90
- Buffer em azul
- Agora a estrutura do começo dessa aula está bem caracterizada
- Para ver a qtd de bytes, basta clicarmos duas vezes onde cada espaço começa
- Após seguirmos com o Fn+F7, chegaremos no ponto de passar todas as variáveis para os espaços reservados no buffer, e ele ficará no ponto de chamar o return

0061FE80	0061FE90	E=a.	ASCII "RICARDOLONGATTO"
0061FE84	00730E18	↑f.s.	ASCII "RICARDOLONGATTO"
0061FE88	0061FEA8	↓m.a.	
0061FE8C	00401E5B	[ΔC.	RETURN to check.00401E5B from check.00401700
0061FE90	41434952	RICA	
0061FE94	4C4F4452	RDOL	
0061FE98	41474E4F	ONGA	
0061FE9C	004F5454	TTO.	
0061FEA0	00730D58	X.s.	

+ Agora passaremos um argumento maior do que o suportado: aaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbb

0061FE78	00000021	?...	
0061FE7C	00401526	&S.C.	RETURN to check
0061FE80	0061FE90	E=a.	
0061FE84	00B90E28	<f.s.	ASCII "aaaaaaaaa
0061FE88	0061FEA8	↓m.a.	
0061FE8C	00401E5B	[ΔC.	RETURN to check
0061FE90	61616161	aaaa	
0061FE94	61616161	aaaa	
0061FE98	61616161	aaaa	
0061FE9C	61616161	aaaa	
0061FEA0	62626262	bbbb	
0061FEA4	62626262	bbbb	
0061FEA8	62626262	bbbb	
0061FEAC	62626262	bbbb	
0061FEB0	00000002	0...	
0061FEB4	00B90DC0	L.J.	

- o endereço de retorno foi também sobrescrito
- Ao executarmos ele, obteremos o erro

CPU - main thread, module check

00401526	B8 00000000	MOV EAX,0
0040152B	C9	LEAVE
0040152C	C3	RETN
0040152D	90	NOP
0040152E	90	NOP
0040152F	90	NOP
00401530	83EC 1C	SUB ESP,1C
00401533	8B4424 24	MOV EAX,DWORD PTR SS:[ESP+24]
00401537	85C0	TEST EAX,EAX
00401539	74 15	JE SHORT check.00401550
0040153B	83F8 03	CMP EAX,3
0040153E	74 10	JE SHORT check.00401550
00401540	B8 01000000	MOV EAX,1
00401545	83C4 1C	ADD ESP,1C
00401548	C2 0C00	RETN 0C
0040154B	90	NOP
0040154C	8D7426 00	LEA ESI,DWORD PTR DS:[ESI]
00401550	8B5424 28	MOV EDX,DWORD PTR SS:[ESP+28]
00401554	894424 04	MOV DWORD PTR SS:[ESP+4],EAX

Registers (FPU)

EAX	00000000
ECX	00B90E4C
EDX	ABABAB00
EBX	00000002
ESP	0061FEAC ASCII "bbbb"
EBP	62626262
ESI	00B90DC0
EDI	00000021
EIP	0040152C check.0040152C
C 0	ES 002B 32bit 0<FFFFFFFF>
P 1	CS 0023 32bit 0<FFFFFFFF>
A 0	SS 002B 32bit 0<FFFFFFFF>
Z 1	DS 002B 32bit 0<FFFFFFFF>
S 0	FS 0053 32bit 2C4000<FFF>
T 0	GS 002B 32bit 0<FFFFFFFF>
D 0	
O 0	LastErr ERROR SUCCESS <00000000>


Address	Hex dump	ASCII
00403000	0A 00 00 00 FF 00 00 00 FF FF FF FF FF FF FF FF
00403010	6C 26 40 00 02 00 00 00 FF FF FF FF A0 25 40 00	l&e.0...
00403020	B0 25 40 00 1F BB 25 C7 E0 44 DA 38 00 00 00 00	%0 %v%0

Address	Hex dump
00403000	0A 00 00 00 FF 00 00 00 FF FF FF FF FF FF FF FF
00403010	6C 26 40 00 02 00 00 00 FF FF FF FF A0 25 40 00
00403020	B0 25 40 00 1F BB 25 C7 E0 44 DA 38 00 00 00 00
00403030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00403040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00403050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00403060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00403070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00403080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00403090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Registers (FPU)

EAX	00000000
ECX	00B90E4C
EDX	ABABAB00
EBX	00000002
ESP	0061FEB0
EBP	62626262
ESI	00B90DC0
EDI	00000021
EIP	62626262
C 0	ES 002B 32bit 0<FFFFFFFF>
P 1	CS 0023 32bit 0<FFFFFFFF>
A 0	SS 002B 32bit 0<FFFFFFFF>
Z 1	DS 002B 32bit 0<FFFFFFFF>
S 0	FS 0053 32bit 2C4000<FFF>
T 0	GS 002B 32bit 0<FFFFFFFF>
D 0	

Error



Don't know how to step because memory at address 62626262 is not readable. Try to change EIP or pass exception to program.

OK

- O programa quebrou pois o return está apontando para o 62626262
- Quanto maior o buffer alocado, maior será a stack
- Uma observação é que se mudarmos o return de 0 para 5, veremos a diferença de mov eax, 0 para mov eax, 5

