

Assembly do IA-32 em ambiente Linux

TPC8 e Guião laboratorial

Alberto José Proença

Objectivo

A lista de exercícios/tarefas propostos no TPC8 / Guião laboratorial reforça a análise laboratorial (e a ferramenta associada, o depurador `gdb`) referente ao conjunto de **instruções e técnicas para suporte à invocação e execução de funções em C**.

Os exercícios para serem resolvidos antes da aula TP estão assinalados com uma caixa cinza.

Buffer overflow

1. O seguinte código C mostra uma implementação (de baixa qualidade) de uma função que lê uma linha da *standard input*, copia a *string* lida para uma novo local de memória, e devolve um apontador para o resultado.

```

1 /* Isto é código de qualidade questionável.
2    Tem como objectivo ilustrar más técnicas de programação. */
3 char *getline()
4 {
5     char buf[8];
6     char *result;
7     gets(buf);
8     result = malloc(strlen(buf));
9     strcpy(result, buf);
10    return(result);
11 }
```

a) ^(A) **Construa** um `main` simples que invoque a função `getline` e confirme que o programa executável “desmontado” (*disassembly*) até à chamada da função `gets` é semelhante a:

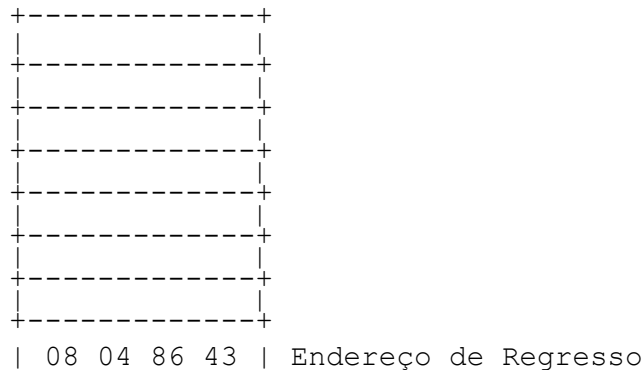
1	08048524	<getline>:		
2	8048524:	55	push	%ebp
3	8048525:	89 e5	mov	%esp,%ebp
4	8048527:	83 ec 10	sub	\$0x10,%esp
5	804852a:	56	push	%esi
6	804852b:	53	push	%ebx
7	804852c:	83 c4 f4	add	\$0xfffffffff4,%esp
8	804852f:	8d 5d f8	lea	0xfffffffff8(%ebp),%ebx
9	8048532:	53	push	%ebx
10	8048533:	e8 74 fe ff ff	call	80483ac <_init+0x50> <i>Invoca gets</i>

b) ^(A) Considere o seguinte cenário: a função `getline` é invocada com o endereço de regresso `0x8048643`, o registo `%ebp` com o valor `0xbffffc94`, o registo `%esi` com `0x1`, e o `%ebx` com `0x2` (confirme estes valores ou outros similares); introduz-se a *string* “012345678901”. Confirme que o programa termina anormalmente.

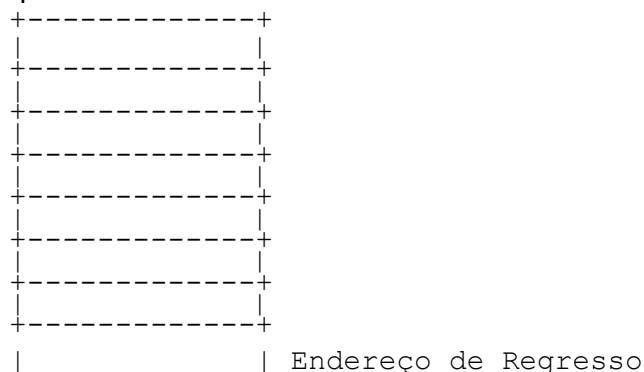
Pretende-se detectar o local onde ocorreu a anomalia na execução do programa, com o auxílio de um depurador.

Nota: deverá chegar à conclusão que tal aconteceu na execução da instrução `ret` da função `getline`.

- c) (A/R) Considerando que a *stack* “cresce para cima”, preencha o diagrama (da *stack frame*) com o máximo de indicações, logo após execução da instrução da linha 6 (no código desmontado). Coloque em cada caixa (que representa 4 *bytes*) o respectivo valor em hexadecimal (se conhecido), e à direita uma etiqueta que ajude a esclarecer o conteúdo da *stack* (por ex., “Endereço de Regresso”).
Confirme agora a *stack frame* que construiu. Indique a posição de `%ebp`.



- d) ^(R) Modifique o diagrama para mostrar os valores expectáveis após a invocação da função `gets` (linha 10), e depois confirme esses valores.



- e) ^(R) Para que endereço acha que o programa está a tentar regressar?

Resp.: _____

Confirme a sua previsão.

- f) ^(R) Que registo(s) ache que foi(oram) corrompido(s) no regresso da função `getline` e como? Confirme a sua previsão.

- g) ^(B) Para além do problema de *buffer overflow*, que duas outras coisas estão erradas no código de `getline`?