Assembly do IA-32 em ambiente Linux

TPC8 e Guião laboratorial

Alberto José Proença & Luís Paulo Santos

Objetivo

A lista de exercícios/tarefas propostos no TPC8 / Guião laboratorial, para execução no servidor, 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**.
Não esquecer que estes trabalhos experimentais deverão ser realizados no servidor Unix de SC, à semelhança dos trabalhos anteriores.

Este guião permite a resolução do trabalho de modo autónomo e a sua **entrega é obrigatória**, na plataforma de *e-learning* da UC, antes das 12h00 de quarta 22-abr-20.

Buffer overflow

1. **(A)** 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 um novo local de memória, e devolve um apontador para o resultado.

```
1 /*  Isto e' codigo de qualidade questionavel.
2     Tem como objetivo ilustrar tecnicas deficientes de
programacao. */
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 }
```

2. **(A)** **Construa** um `main` simples que invoque a função `getline` e compile-o sem qualquer otimização, i.e., com `-O0`; confirme que o programa executável "desmontado" (*disassembled*) da função `getline` até à chamada da função `gets` é semelhante a:

```
1   8048474 <getline+0>:     push  %ebp
2   8048475 <getline+1>:     mov   %esp,%ebp
3   8048477 <getline+3>:     sub   $0x18,%esp
4   804847a <getline+6>:     sub   $0xc,%esp
5   804847d <getline+9>:     lea   -0x8(%ebp),%eax
6   8048480 <getline+12>:    push  %eax
7   8048481 <getline+13>:    call  8048360 <gets@plt>    ;Invoca gets
```

3. **(A)** Execute o programa introduzindo uma *string* suficientemente longa (por exemplo, `1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2`) e confirme que o programa termina anormalmente. **Pretende-se** ao longo deste Guião laboratorial detetar o local onde ocorreu a anomalia na execução do programa, com o auxílio de um depurador.

4. **(A/R)** Analise a execução do código desmontado no exercício **2** (função `getline`) até à linha 5. Ao longo da execução, vários valores e registos que definem a *stack frame* associada a esta função serão alterados. Observando esses valores e registos e seguindo as instruções *assembly* permite-nos deduzir os valores que definem e constituem o quadro desta função (*stack frame*).
Preencha o diagrama da *stack frame* que é gerado até este ponto da execução, com a estimativa desses valores e endereços.

5. **(A/R)** **Confirme** agora a *stack frame* que construiu, colocando um *breakpoint* na linha 5 de `getline` e executando o programa. Indique a posição de `%ebp`. **Confirme** que o endereço de regresso está correto, examinando o código da função `main()`.

6. **(R)** **Preencha** o diagrama relativo à *stack frame* de `getline`, após a execução da função `gets`, usando a *string* de 12 carateres sugerida no exercício **3**.

7. **(R)** **Identifique** as células de memória da *stack frame* que foram alteradas após executar a função `gets`. **Descreva** detalhadamente o impacto destas alterações na restante execução do programa.

8. **(R)** **Identifique** o(s) registo(s) que foi(oram) corrompido(s) no regresso da função `getline` e **mostre** como foram modificados.

9. **(R)** **Identifique e caracterize** os problemas associados a utilização da função `gets`.

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

Nº	Nome:	Turma:

Resolução dos exercícios (deve ser redigido manualmente)

1. Código C de um `main` simples que invoque a função `getline`

Copie para aqui o código C de um `main` simples que colocou no servidor remoto (para invocar a função `getline`).

2. Análise do código desmontado

Compile o código C sem qualquer otimização (com `-O0`) e **copie** para aqui o código executável "desmontado" (*disassembled*) da função `getline` até à chamada da função `gets`, mostrando (com um *print screen* ou foto do monitor) todos os comandos que usou para compilar e ter o código desmontado da função.
Anote cuidadosamente o código desmontado, ignorando as fases de arranque e término da função.

3. Execução do código

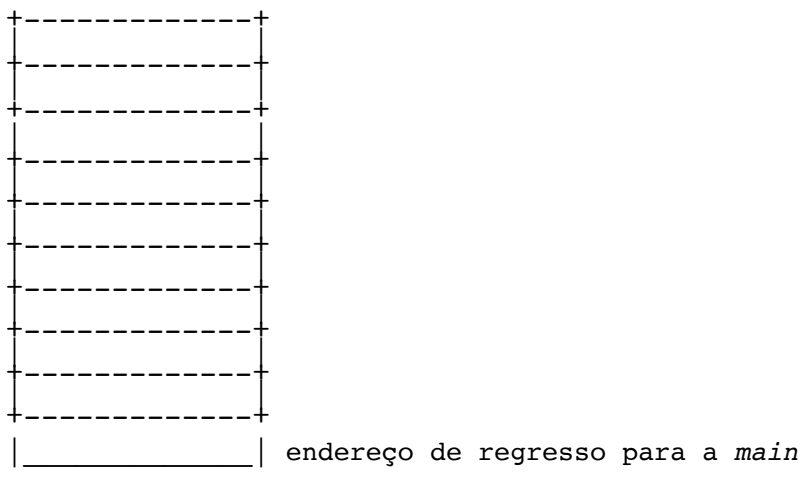
Replique aqui tudo que apareceu no monitor assim que mandou executar o código (incluindo os caracteres que tiver introduzido e o resultado da execução do código).

4. Estimando o quadro da função `getline` na *stack*

A anomalia que constatou poderá ser (i.e., vamos assumir que possa ser) devida a uma utilização menos correta da *stack*, que tenha conduzido a que a execução do código tenha tentado aceder a uma zona de memória que não faz parte da área de memória (ou do *segmento de memória*) que estava alocado a este código. E onde é mais provável que tal possa acontecer é no regresso de uma função, se o valor do endereço de regresso (que está na *stack*) tiver sido indevidamente modificado.
Para verificar se foi isto que aconteceu, temos de analisar o quadro da função `getline`.

Preencha o diagrama do quadro da função `getline` (a sua *stack frame*) que é gerado até este ponto da execução, (até à linha 5) com a estimativa dos seus valores e endereços, procedendo assim:

- **a. coloque** dentro de cada caixa (que representa 4 células de memória) o respetivo valor em hexadecimal (não precisa de pôr `0x`);
- **b. coloque** à esquerda de cada caixa o endereço mais baixo das 4 células lá representadas;
- **c. coloque** à direita de cada caixa uma etiqueta que descreva o que representa a caixa.
  Nota: algumas das caixas (com 4 células de memória) podem conter valores arbitrários.

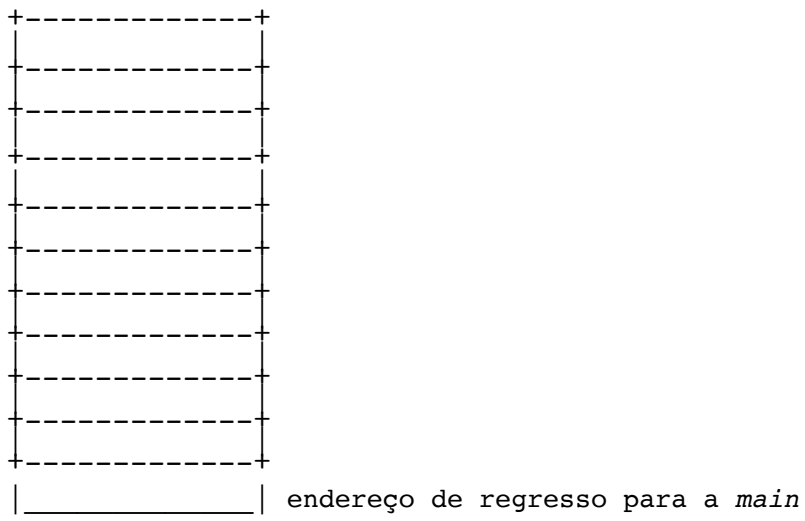


5. Confirmação de valores do quadro da função `getline` na *stack*

Vamos confirmar a *stack frame* que construiu, colocando um *breakpoint* na linha 5 de `getline` e executando o programa.
Quando este parar no *breakpoint*, veja os valores dos registos relevantes para a *stack frame* (`%ebp` e `%esp`). **Complemente/confirme** a sua estimativa de endereço de regresso, examinando o código da função `main()`.
Coloque aqui esses 3 valores dos registos e **mostre** os valores que o *debugger* apresentou no monitor para o conteúdo da *stack frame* (apresente da mesma maneira que apareceu no seu monitor).

6. Nova análise do quadro da função `getline` na *stack*

Preencha o diagrama seguinte relativo à *stack frame* de `getline`, estimando os valores dos conteúdos das caixas, após a execução da função `gets`, usando a *string* de 12 carateres sugerida no exercício **3**.



7. (e 8.) Explicação da alteração do quadro da função `getline` na *stack*

Identifique no diagrama em cima as células de memória da *stack frame* que foram alteradas após executar a função `gets`.
Descreva detalhadamente o impacto destas alterações na restante execução do programa.
Identifique o(s) registo(s) que foi(oram) corrompido(s) no regresso da função `getline` e mostre como foram modificados.
Identifique e caracterize os problemas associados à utilização da função `gets`.