TPC8 e Guião Laboratorial

Dicas para validação da execução do trabalho

Este problema cobre uma gama variada de tópicos: quadros de ativação de funções na *stack* (*stack frames*), representação de *strings*, código ASCII e ordenação de *bytes* em memória. O trabalho usa uma versão compilada sem otimização (para melhor compreender o processo de compilação), onde apenas o registo `%ebp` é salvaguardado antes de se reservar espaço para o vetor `buf`. Os valores de endereços indicados aqui poderão não corresponder aos que foram analisados nas aulas.

Há 2 pistas relevantes que deverão ser consideradas:
(i) a sugestão dada no enunciado para testarem o código com uma *string* longa, e
(ii) a indicação do compilador de que a função `gets` é "perigosa" (por ter "cadastro criminal"...).
Assim, é de desconfiar que a anomalia poderá estar no uso do espaço de memória alocado à *string* (no quadro da função `getline`, na *stack*) e na utilização da função `gets`.

2. Código desmontado e devidamente anotado (sem anotações na fase de arranque):

```
(gdb) disas getline
Dump of assembler code for function getline:
0x08048400 <getline+0>:    push   %ebp
0x08048401 <getline+1>:    mov    %esp,%ebp
0x08048403 <getline+3>:    sub    $0x18,%esp
0x08048406 <getline+6>:    sub    $0xc,%esp
0x08048409 <getline+9>:    lea    0xfffffff8(%ebp),%eax
0x0804840c <getline+12>:   push   %eax
0x0804840d <getline+13>:   call   0x8048304
```

Reparar onde o `gcc` decidiu colocar a *string* local `buf`, em `(%ebp-8)`

```
# reserva espaço na stack p/ 24 bytes
# reserva espaço para mais 12  bytes
# %eax= buf
# coloca na stack argumento p/ gets
# invoca a função gets c/ argum buf
```

3. Análise do que aconteceu quando se mandou executar o código:

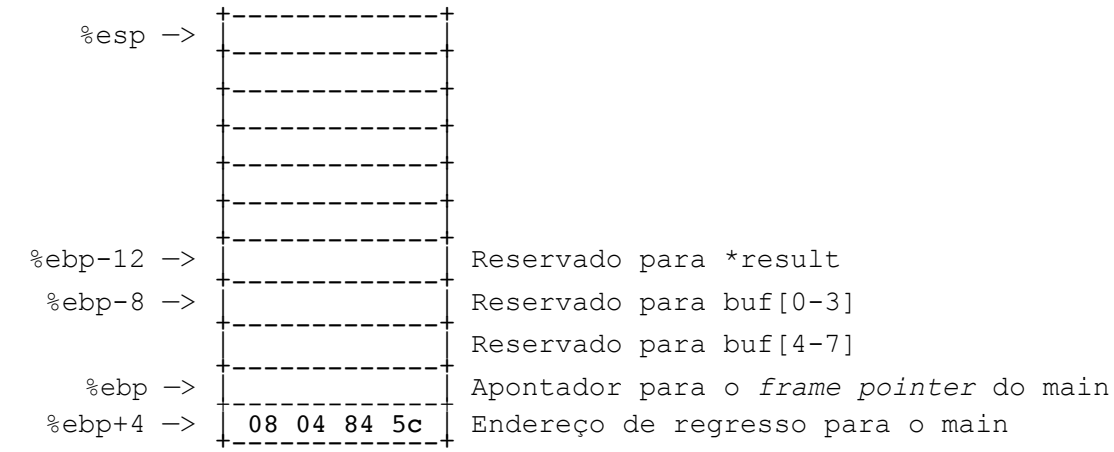
```
[docente@sc TPC8]$ ./main-ensaio
123456789012

Segmentation fault
[docente@sc TPC8]$
```

4. Na construção do quadro de ativação da função `getline` na *stack,* antes de chamar a função `gets` (a *stack* cresce para cima), o endereço de regresso está no código da `main`: é o endereço da instrução a seguir à instrução de `call getline`. Veja em baixo:

```
0x08048457 <main+16>:    call   0x8048400 <getline>
0x0804845c <main+21>:    mov    %eax,0xfffffffc(%ebp)
```

O valor do apontador para o quadro de ativação da função `main` (o *frame pointer* do `main`) obtém-se lendo o valor em `%ebp` após uma paragem logo no início de `getline`.



5. Após paragem no *breakpoint* indicado, obterá os valores do *stack pointer* e do *frame pointer* de `getline` (estes deverão ser valores do género `0xbfffxxxx` para o código desmontado nesta resolução, pois este quadro de ativação da função deverá estar empilhado por cima do quadro do `main`).
Estes valores poderão ser colocados no diagrama anterior, à esquerda das caixas. De notar que a diferença entre os valores de `$esp` e `$ebp` é de `0x24`, ou seja `0x18+0xc`, correspondendo aos valores nas duas instruções `sub` executadas até ao ponto de paragem.

```
Breakpoint 1, 0x08048409 in getline ()
(gdb) info registers
...
esp            0xbfff9054       0xbfff9054
ebp            0xbfff9078       0xbfff9078
```

Para confirmar os valores no quadro de ativação deverá "examinar" em hexadecimal as 2 *words* (32 bits na terminologia do `gdb`) a partir da posição apontada por `$ebp`:

```
(gdb) x/2xw $ebp
0xbfff9078:    0xbfff9088      0x0804845c
```

Apontador para o quadro do `main`

Endereço de regresso para o `main`

6. Vamos analisar o estado do quadro de `getline` após regressar de `gets` e assumindo que foi introduzida a *string* `123456789012`.

A função `gets` limita-se a ler uma linha do *standard input* até encontrar o caráter `newline` ou uma condição de erro, terminando a seguir a escrita da *string* com o caráter `null` (ASCII `0x00`); neste caso, vai ler os 12 carateres da *string* e acrescentar o `null`; mas como apenas tem reservado para a *string* um *array* de 8 elementos, veja o que acontece...

Visualize o conteúdo das células de memória do quadro da função com o comando do `gdb` `x/12xw $esp`. Este comando mostra as 12 “palavras” (**`w`**, *words* – que em IA-32 são 4 *bytes*, cada equivalente a uma “célula” do diagrama da *stack*) em hexadecimal (**`x`**, hex) a partir do endereço de memória contido no registo `%esp` (cujo valor aponta para o topo da *stack*).

```
+-------------+
| 34 33 32 31 | buf[0-3]
+-------------+
| 38 37 36 35 | buf[4-7]
+-------------+
| 32 31 30 39 | Apontador para o quadro do main
+-------------+
| 08 04 84 00 | Endereço de regresso para o main
+-------------+
```

Observe o quadro da função, agora apenas representado na parte mais relevante:

- os carateres `1`, `2`, `3`, `...` estão lá claramente identificados (em ASCII, `31`, `32`, `33`, `...`) bem como o caráter `null` (em ASCII `00`);
- destacam-se (com cor diferente e a *bold*) as 5 células de memória que foram indevidamente alteradas.

7. **(e 8.)** Analise o que aconteceu no quadro da função:

- o *byte* menos significativo do endereço de regresso da função foi indevidamente alterado; este programa está a tentar regressar ao endereço `0x08048400`, uma vez que o *byte* menos significativo **foi modificado** (*overwritten*) pelo caráter terminador (*null*);
- o valor guardado do apontador para o quadro do `main` foi modificado para `0x32313039`, e este valor será o “recuperado” para `%ebp` antes do regresso de `getline`.

9. A função *gets* não limita o número de caracteres que podem ser lidos,

10. Esta função devolve um apontador para uma zona de memória que já foi libertada. O espaço para `buf` está reservado na *stack* e será libertado no final da função. O mais provável será que o conteúdo de `buf` irá aparecer alterado assim que esse espaço for usado por outra função.

Notas finais:

A chamada de `malloc` deveria ter como argumento `strlen(buf)+1`, e deveria também verificar que o valor a devolver é *`non-null`*.
Para evitar este tipo de problema usar `fgets` ou `scanf` em vez de `gets` (que não é suportado desde 2011).

Se tivesse compilado com otimização `-O2`, seria preciso introduzir bastantes mais valores para ter o mesmo efeito (quantos mais, e porquê?).
Adicionalmente, todos os registos salvaguardados pela função também seriam alterados (porquê?).