Desafio 4

Desafio: Parabéns por chegar até aqui, esse desafio está em um nível diferente dos demais, então não se sinta frustrado se não entender de primeira, a flag não é a mensagem de sucesso, mas o argumento para printá-la. Boa sorte.

Resolução do Desafio:

1) Foi fornecido um arquivo para que pudéssemos analisá-lo.

2) Analisando melhor o arquivo, e após abri-lo pode-se descobrir que é um arquivo ELF.
   O ELF é um padrão comum de arquivo para executáveis, código objeto, bibliotecas compartilhadas, e core dumps.

3) Para que esse arquivo fosse transformado em executável foi utilizado o seguinte comando abaixo:
   sudo chmod +x ./willucrackme

4) Para executá-lo foi usado o seguinte comando:
   ./willucrackme

5) A partir disso pode-se entender a parte do desafio que diz: **"a flag não é a mensagem de sucesso, mas o argumento para printá-la"**
   Ao se executar o comando pode se observar que o executável espera que um argumento seja colocado.

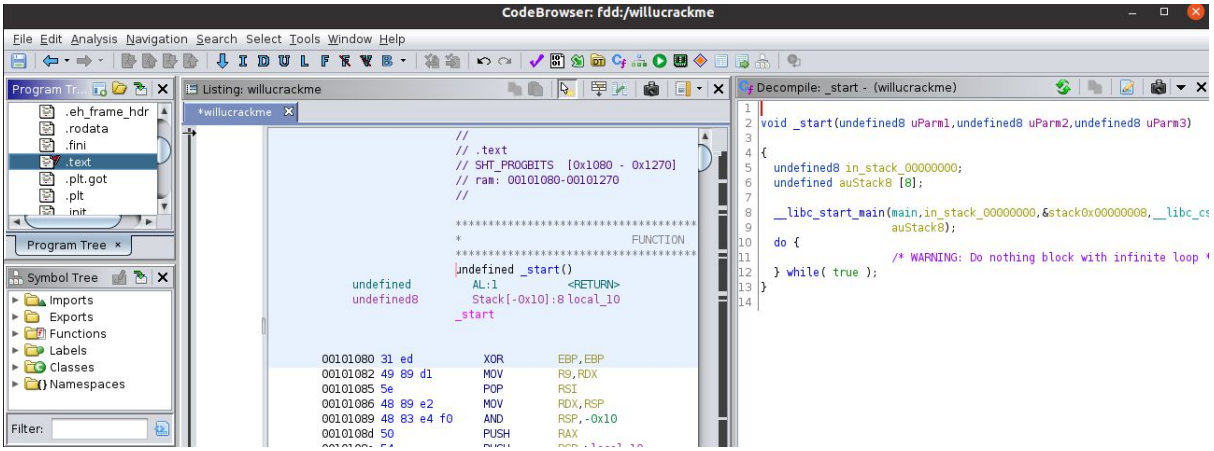
```
bene@bene-Inspiron-5468:~/Desktop$ ./willucrackme
willucrackme: Por favor, argumente
```

6) Ao colocar qualquer argumento aparece a seguinte mensagem, dizendo que o argumento passado está errado. Pode se inferir que ele espera que seja passado mais de um argumento, ou seja argv == 2.

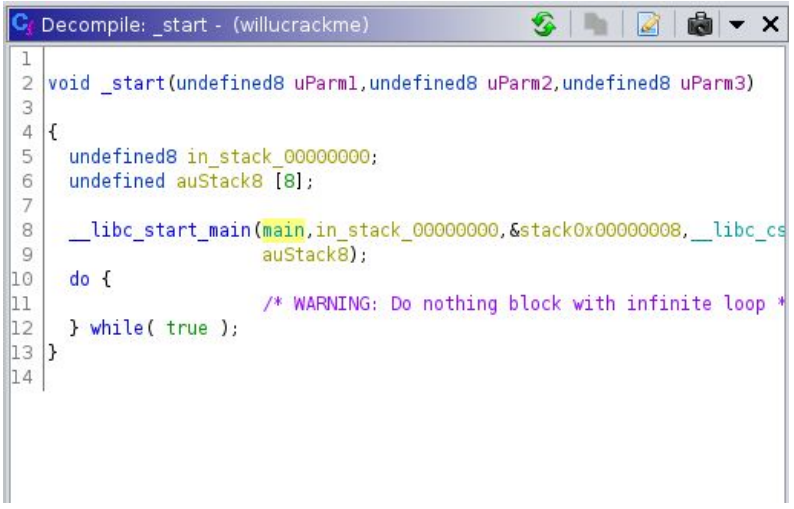
```
bene@bene-Inspiron-5468:~/Desktop$ ./willucrackme dgssff
Errrrrrrrrrrrou, 0x00000000
```

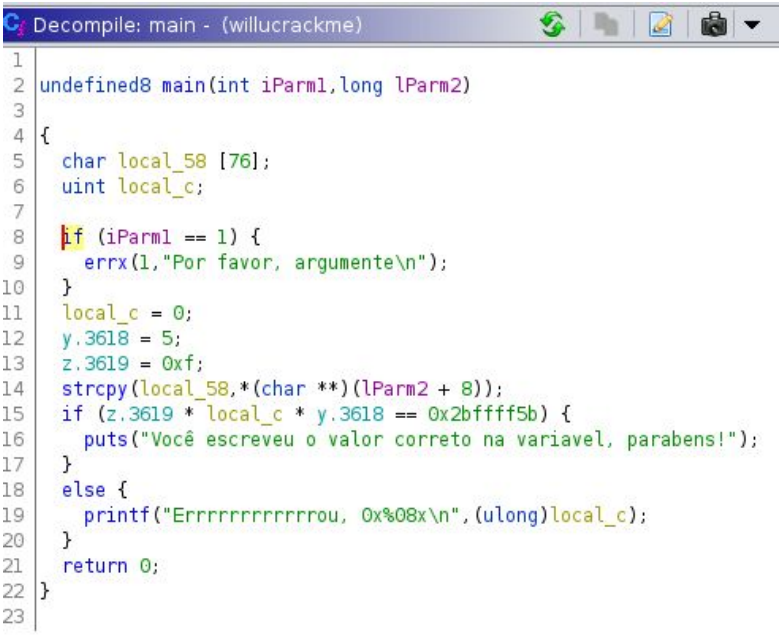
7) Para entender melhor um pouco mais o executável foi utilizado a ferramenta Ghidra, o qual é possível explorar melhor os endereços e o assembly contido no arquivo, o qual facilita a visualização de um pseudo-código gerado por ele de algumas funções contidas no arquivo.

8) As seções do nosso arquivo ELF estão listados do lado esquerdo do programa.
   A seção ***.text*** é a qual iremos analisar pois é nela que estará contido a ***main.***



9) Clicando na parte da main, no código, será mostrado a ***main*** do arquivo.





10) Na **main** pode se observar que para que o executável retorne a mensagem de sucesso seja satisfeita a seguinte equação:

z.3619 * local_c * y.3618 == 02xbffff5b

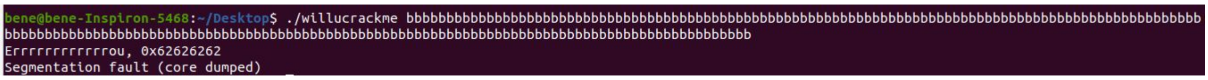
11) Ao se analisar também a mensagem de erro, observa-se que a mesma retorna o ***local_c:***

"Errrrrrrrrrrrou, 0x%08x\n",(ulong)local_c

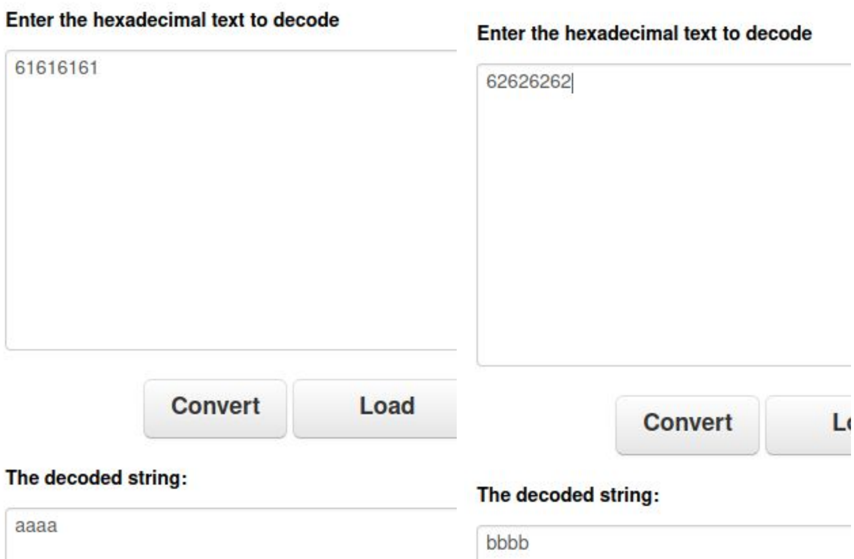
12) Outra informação relevante nesse código é que há uma declaração de vetor chamado local_58 com tamanho 76.

char local_58 [76]

13) Executando o executável no terminal, pode-se perceber que ao colocar um número de string grande o sistema mostra um buffer overflow.



14) Realizando outros testes também observa-se que o valor de **local_c** está ligado com o valor que é colocado como argumento no arquivo.



15) Além disso, como descobrimos que se trata de um buffer overflow e a partir da análise do código concluímos que o valor de **local_c** está depois da posição 76 do vetor ***local_58***, porque antes disso, o valor printado pela mensagem de erro continua sendo 0x0000000.

```
bene@bene-Inspiron-5468:~/Desktop$ ./willucrackme aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Errrrrrrrrrrrou, 0x00000061
```

16) Para que então possamos printar a mensagem de sucesso, precisamos mudar o valor de **local_c** para que após a multiplicação o resultado seja **02xbffff5b.**
Sabendo que os valores das variáveis presentes na multiplicação são esses e sabendo que o resultado é **0x102xbffff5b** (Valor informado pelos próprios membros da RSI).

```
local_c = 0;
y.3618 = 5;
z.3619 = 0xf;
```

Então só precisamos fazer uma conta matemática para acharmos o valor correto de **local_c**, o qual iremos mudar para que o **if** seja verdadeiro e assim imprima a mensagem de sucesso.

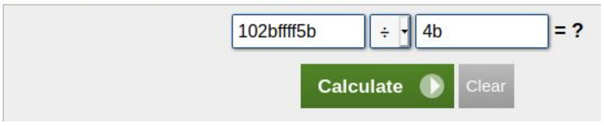
z.3619 * local_c * y.3618 == 02xbffff5b

Fiz os seguintes passos utilizando uma calculadora que multiplica Hexadecimal online:

- 0xf × 5 = 4b

- Hex value:
  102bffff5b ÷ 4b = 37333331

  Decimal value:
  69457674075 ÷ 75 = 926102321



17) Com o intuito de entender um pouco mais sobre o código que foi gerado pelo Ghidra, eu utilizei a técnica chamada como **Análise estática do código**.
Transformei o seguinte código gerado pelo Ghidra em um código em C:

```c
{
  char local_58 [76];


  float local_c = 926102321;
  unsigned char y_3618 = 5;
  unsigned char z_3619 = 0xf;

  int sim =  z_3619 * y_3618 * local_c;

  printf("value of a: %X\n",sim);

  if (z_3619 * local_c * y_3618 == 0x102bffff5b) {
    puts("Você escreveu o valor correto na variavel, parabens!");
  }
  else {
    printf("Errrrrrrrrrrrou, 0x%08x\n",local_c);
  }
  return 0;
}
```

```
main.c:13:3: note: include '<stdio.h>' or provide a declaration of 'p
main.c:16:5: warning: implicit declaration of function 'puts' [-Wimpl
main.c:19:35: warning: format '%x' expects argument of type 'unsigned
value of a: 80000000
Você escreveu o valor correto na variavel, parabens!
```

```c
undefined8 main(int iParm1,long lParm2)

{
  char local_58 [76];
  uint local_c;

  if (iParm1 == 1) {
    errx(1,"Por favor, argumente\n");
  }
  local_c = 0;
  y.3618 = 5;
  z.3619 = 0xf;
  strcpy(local_58,*(char **)(lParm2 + 8));
  if (z.3619 * local_c * y.3618 == 0x2bffff5b) {
    puts("Você escreveu o valor correto na variavel, parabens!");
  }
  else {
    printf("Errrrrrrrrrrrou, 0x%08x\n",(ulong)local_c);
  }
  return 0;
}
```

18) Embora não tenha conseguido transformá-lo fielmente para que ele se comporta-se como o original, consegui entender melhor como o código estava sendo executado.

19) A partir disso, utilizando os valores que conseguimos no tópico **14) ,** e sabendo que o código executável espera uma string e retorna os bytes relacionados a ela em little endian, então foram feitos os seguintes passos:

1- Usou-se o valor em Hexadecimal retornado pela equação:

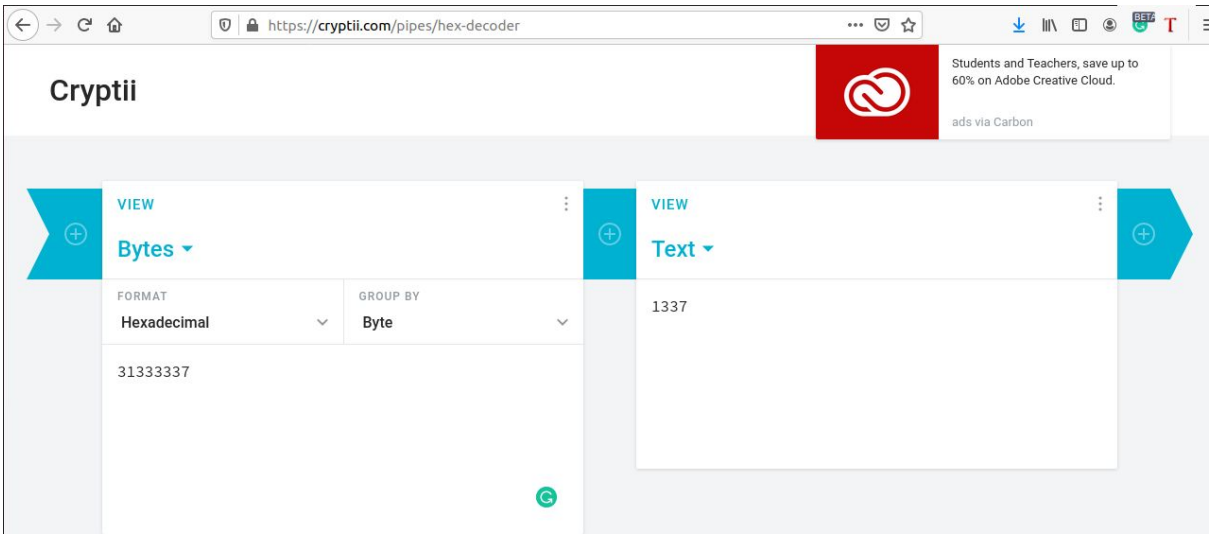
Hex value:

102bffff5b ÷ 4b = 37333331

2- Usou-se o valor em hexadecimal transformando-o em little endian:

31333337

3- Após isso, foi usado um site **'hex-decoder '**que converte o hexadecimal agrupado por bytes para o formato texto:



20) Após esses passos, o valor do texto encontrado foi colocado como parâmetro no terminal e a mensagem de êxito foi informada.

```
bene@bene-Inspiron-5468:~/Desktop$ ./willucrackme ----------------------------------------------------------
--------------1337
Você escreveu o valor correto na variavel, parabens!
```