

**实验报告**



**题目： 缓冲区溢出**

**班 级： 2020211310**

**学 号： 2020211502**

**姓 名： 王小龙**

**学 院： 计算机学院**

**2020年 12 月 7 日**

一、实验目的  
1.理解C语言程序的函数调用机制，栈帧的结构。

2.理解x86-64的栈和参数传递机制

3.初步掌握如何编写更加安全的程序，了解编译器和操作系统提供的防攻击手段。  
3.进一步理解x86-64机器指令及指令编码。

1. 实验环境
2. SecureCRT（10.120.11.12）
3. Linux
4. Objdump命令反汇编
5. GDB调试工具

三、实验内容

登录bupt1服务器，在home目录下可以找到一个targetn.tar文件，解压后得到如下文件：

README.txt；

ctarget；

rtarget；

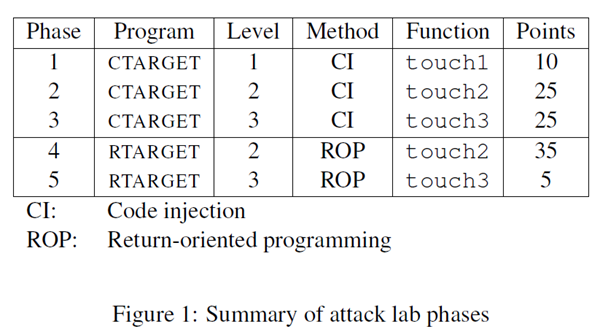
cookie.txt；

farm.c；

hex2raw。

ctarget和rtarget运行时从标准输入读入字符串，这两个程序都存在缓冲区溢出漏洞。通过代码注入的方法实现对ctarget程序的攻击，共有3关，输入一个特定字符串，可成功调用touch1，或touch2，或touch3就通关，并向计分服务器提交得分信息；通过ROP方法实现对rtarget程序的攻击，共有2关，在指定区域找到所需要的小工具，进行拼接完成指定功能，再输入一个特定字符串，实现成功调用touch2或touch3就通关，并向计分服务器提交得分信息；否则失败，但不扣分。因此，本实验需要通过反汇编和逆向工程对ctraget和rtarget执行文件进行分析，找到保存返回地址在堆栈中的位置以及所需要的小工具机器码。实验2的具体内容见实验2说明，尤其需要认真阅读各阶段的Some Advice提示。

本实验包含了5个阶段（或关卡），难度逐级递增。各阶段分数如下所示：



四、实验步骤及实验分析

准备工作：

使用Xshell（10.120.11.12）登录bupt1服务器，使用命令ls查看,使用命令tar -xvf target365.tar解压.

对解压后的文件大致了解：

README.txt:描述目录内容的文件

cookie.txt:存放你攻击用的标识符

rtarget:执行return-oriented-programming攻击的程序

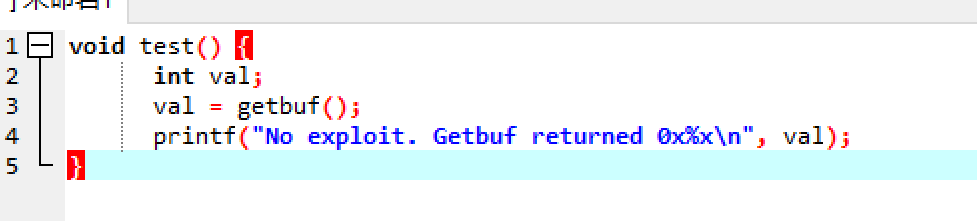
ctarget:执行code-injection攻击的程序

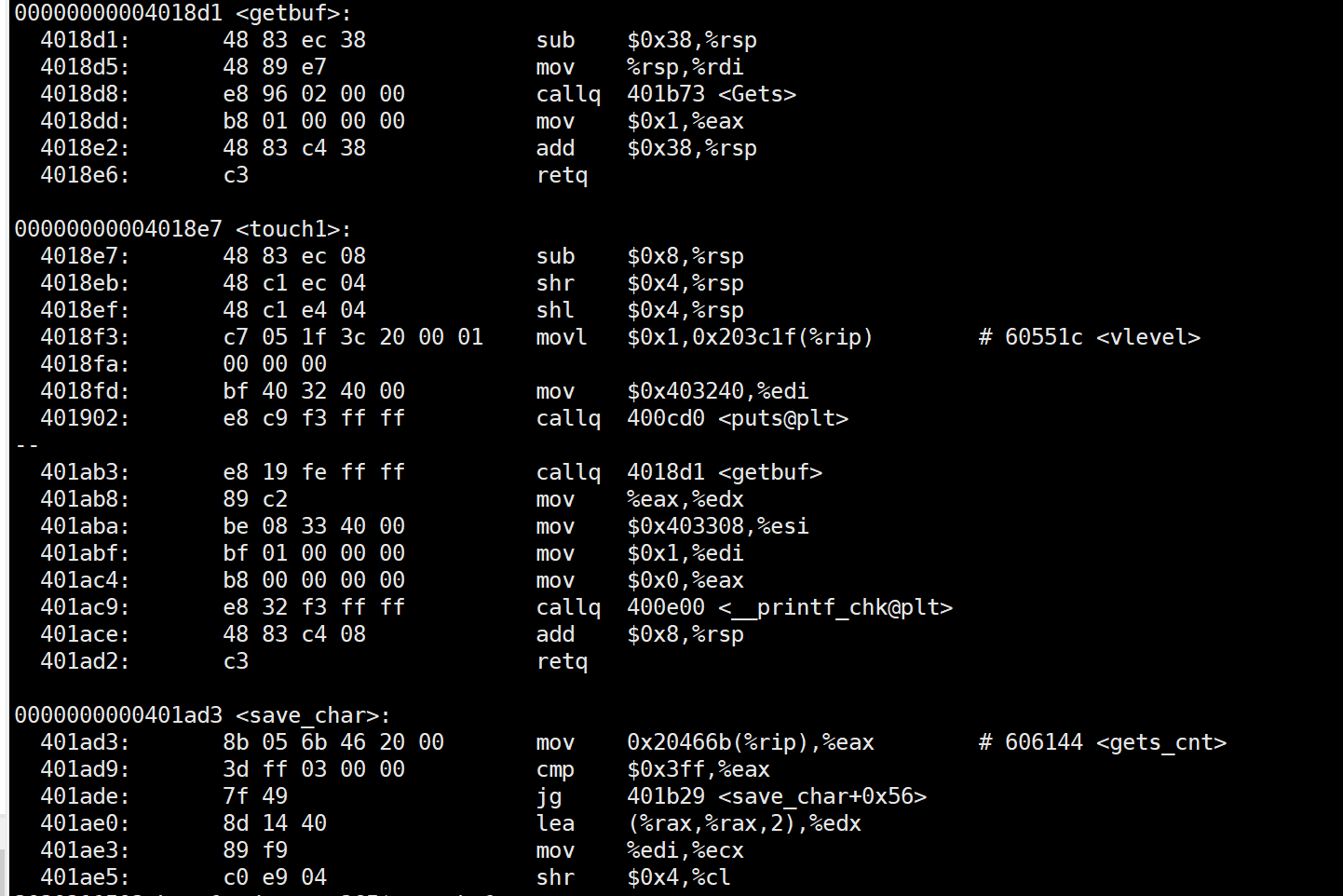
farm.c:“gadget farm产生代码片段用的

hex2raw:生成攻击字符串用的

使用命令objdump -d ctarget > ctarget.txt指令和objdump -d rtarget > rtarget.txt指令生成反汇编代码文件，开始分析。

阶段一：



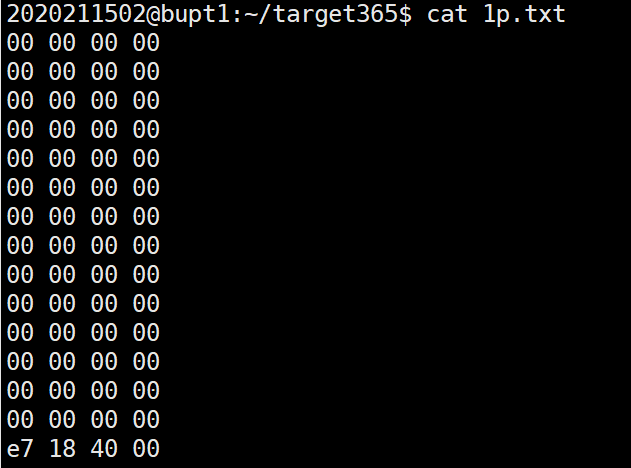


由实验讲义可知，本关要使得getbuf()函数返回的时候，执行touch1()而不是返回test()。

因此我们先确定getbuf创建的缓冲区的大小，查看代码可知，缓冲区大小为0x38（即56）。

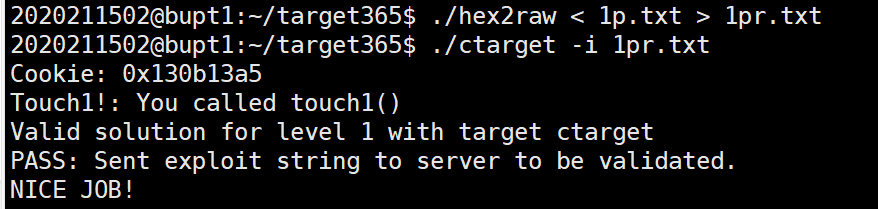
所以我们只需要构造攻击字符串将这56个字节填满，然后再用小端法填入touch1的起始地址，即可完成攻击。

由上图知touch1的地址是0x4018e7,因此攻击序列为：

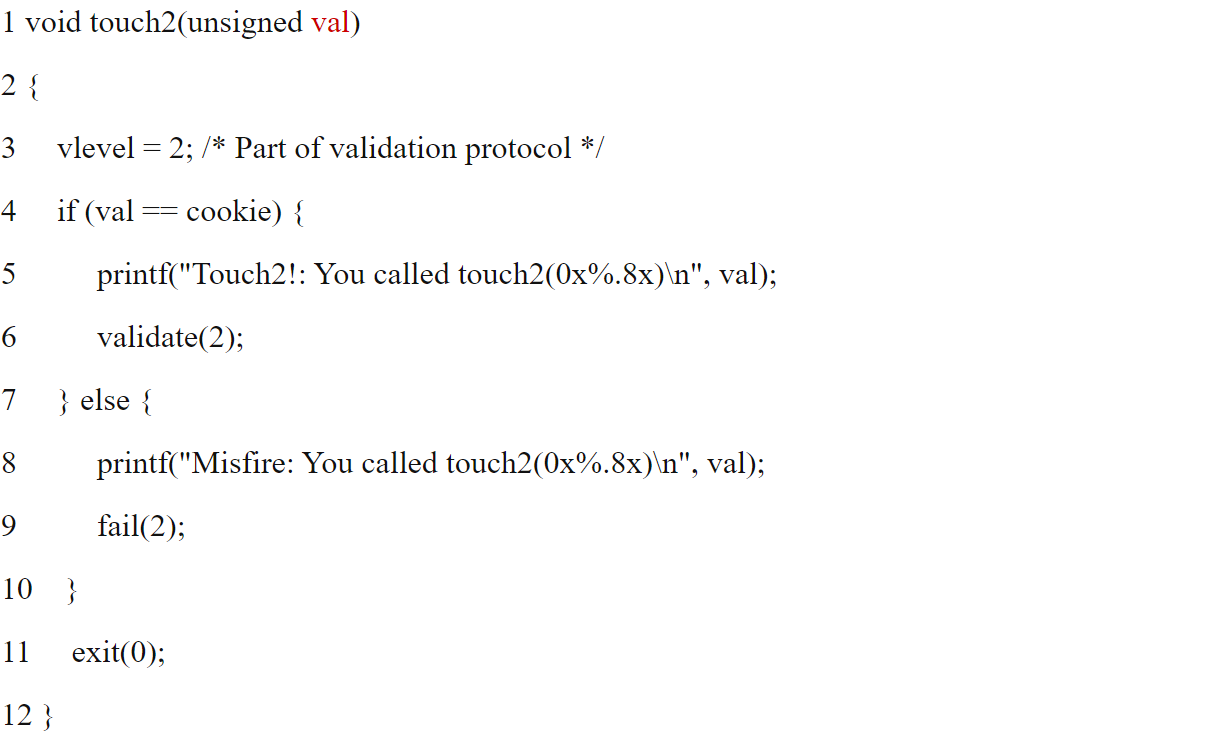


然后使用命令./hex2raw < 1p.txt > 1pr.txt将这个字符文件转换成字节码，

最后使用命令./ctarget -i 1pr.txt,便可通过。



阶段二：

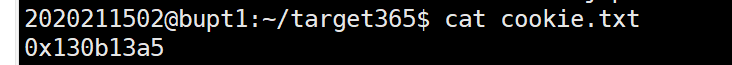


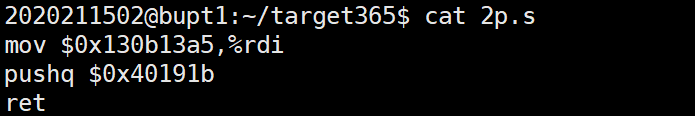
由touch2可知，我们需要把自己的 cookie 作为参数传进去，这里需要把参数放到 %rdi 中，只使用 ret 来进行跳转。

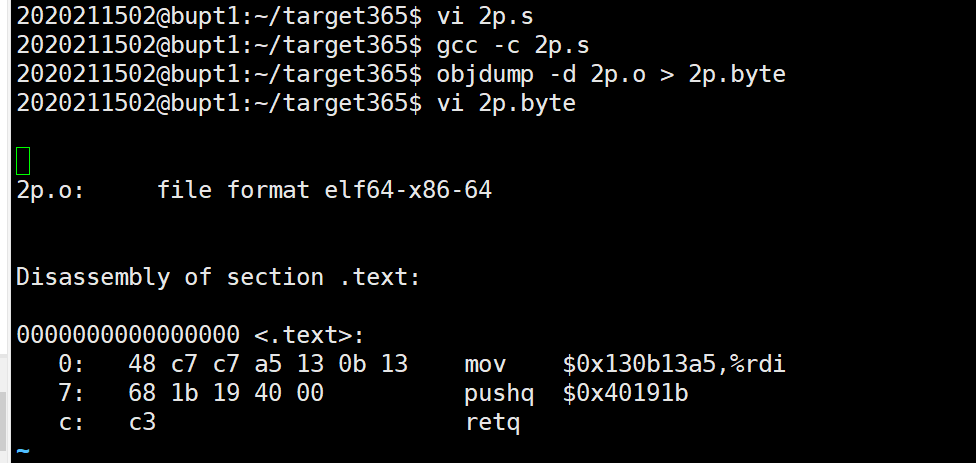
所以需要先来写需要注入的代码(文件2p.s)，

然后使用命令gcc -c 2p.s 编译成二进制文件

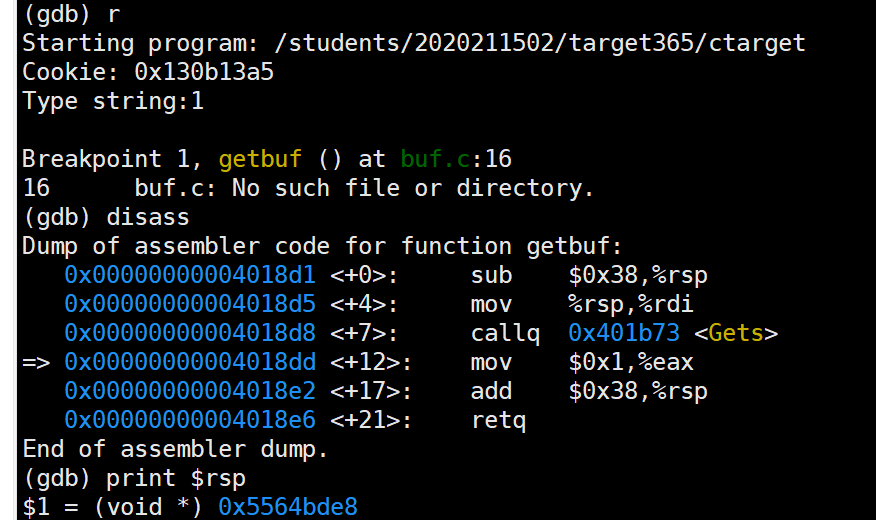
再使用命令objdump -d 2p.o > 2p.byte 反编译观察机器码



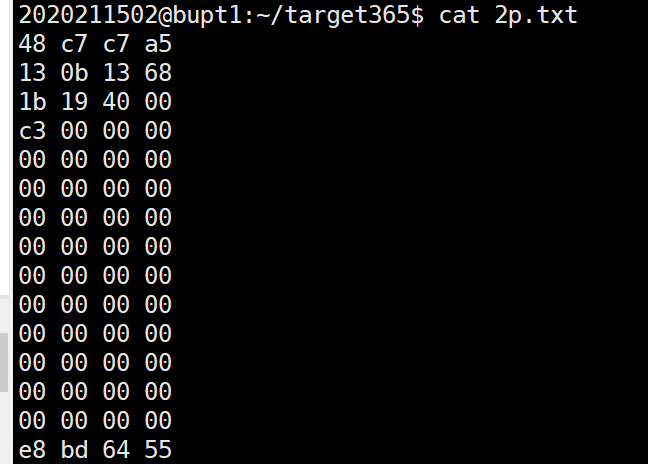




然后需要找出缓冲区的地址，运行gdb，设置断点到getbuf，然后查看%rsp的值，即可得到相应地址.

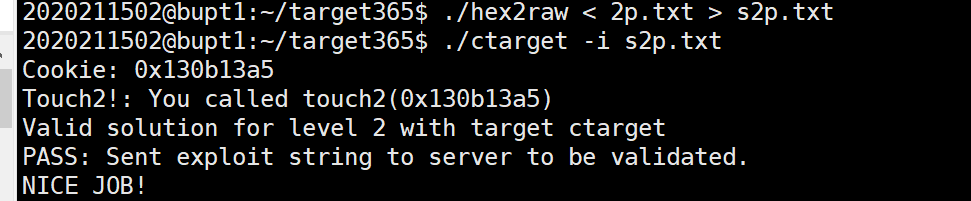


可以看到%rsp指向的位置是 0x5564bde8，这样就可以得到需要输入的字符串了：



然后把字符串转换成字节码：./hex2raw < 2p.txt > s2p.txt

执行命令 ./ctarget -i s2p.txt 就可以看到完成第二阶段的提示了：

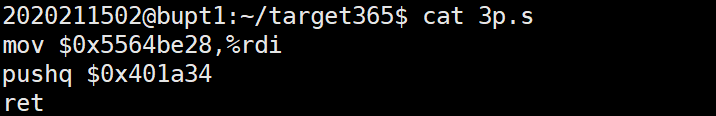


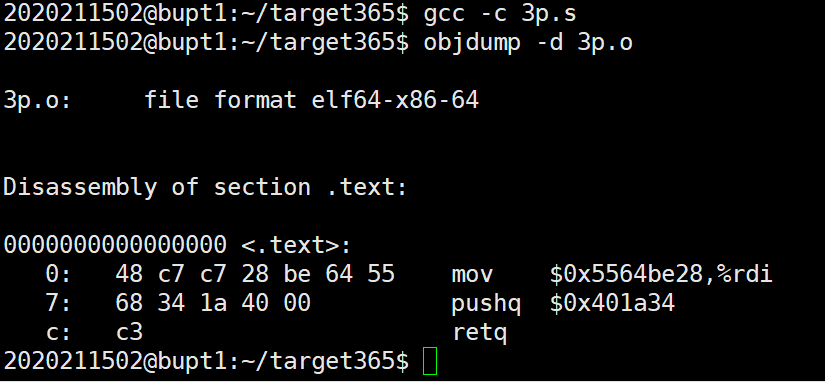
阶段三：

由实验讲义可知，本关需要将cookie转换为字符串传递到touch3,转换0x130b13a5可得31 33 30 62

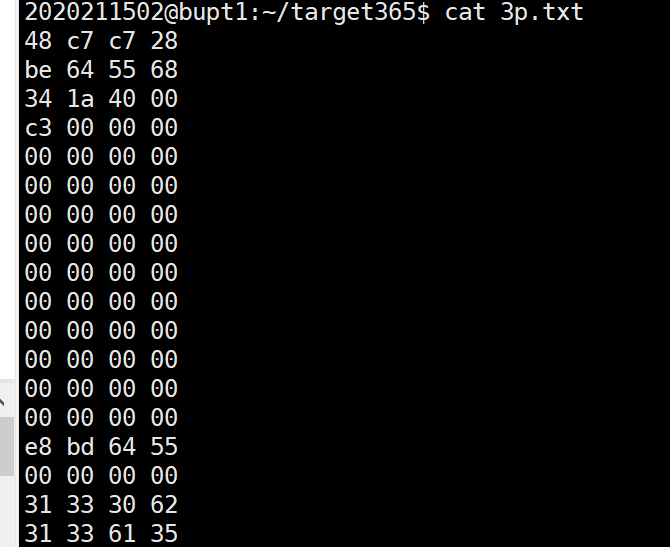
31 33 61 35

因为在touch3内部调用了hexmatch和strncmp函数，所以可能会覆盖一部分缓冲区；但getbuf之上的堆栈是安全的，所以我们可以将cookie存放在getbuf之上的堆栈中，即 %rsp+0x38（缓冲区大小）+0x8（返回地址大小）=0x5564bde8+0x38+0x8=0x5564be28就是cookie应该放置的位置。之后通过和Phase\_2一样的方法得到攻击字节码。输入，通过。

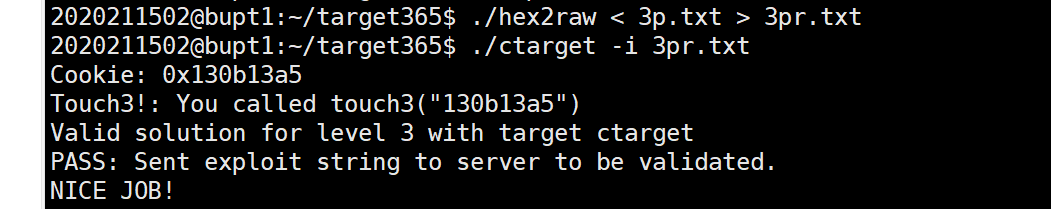




则需要输入的字符串为：



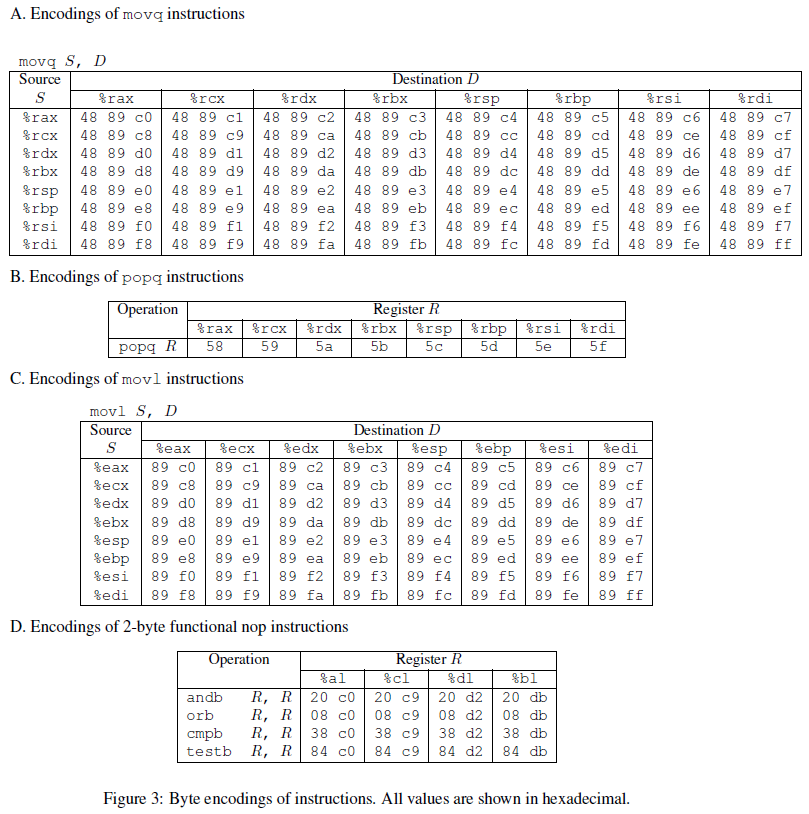
之后输入通过：



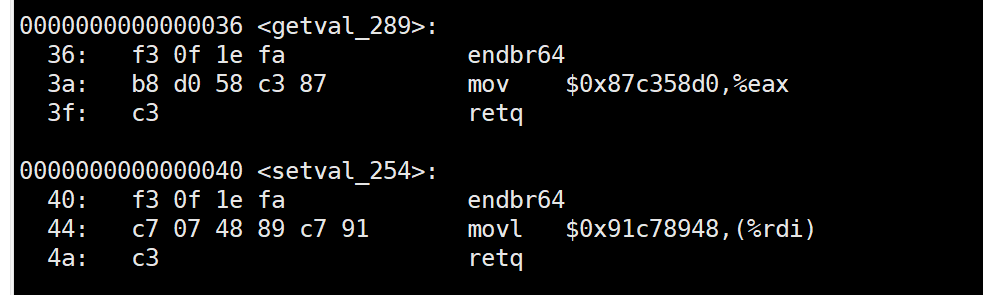
阶段四：

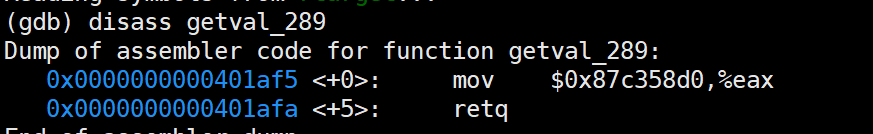
本阶段需要使用ROP攻击的方法完成第二关，所以作为参数，可以把cookie赋给%rdi，再指向touch2;因此思路是用popq %rax和movq %rax,%rdi指令，将cookie作为参数传入。

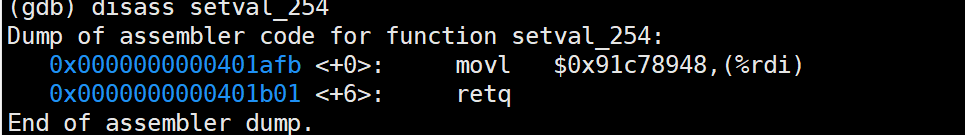
由实验讲义中的编码指令表可知这两个指令分别对应 58 和 48 89 c7 这两个指令；表格如下：



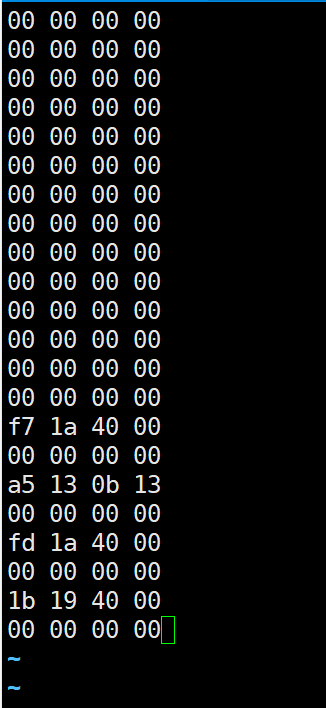
在代码中 start\_farm 到 mid\_farm 之间查找，得到对应的地址分别为 0x401af5+0x2=401af7 和 0x401afb+0x2=401afd；



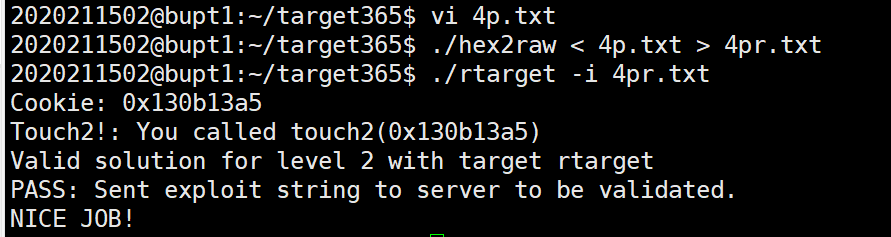




然后构造攻击字符串，具体原理是先用任意字符填充整个缓冲区，再依次传入popq指令地址、cookie地址、movq指令地址、touch2地址,如下图：



最后同上，输入，通过。

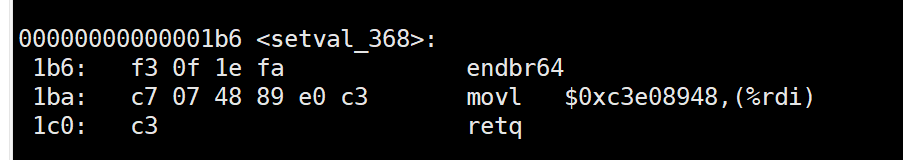


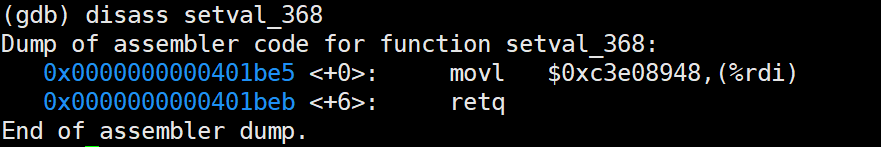
阶段五：

类似于阶段四，需要把cookie 转换成 ascii 码通过缓冲区溢出放到栈的某个位置，然后把指向这个字符串的指针放到 %rdi 中，最后调用 touch3 即可。

需要查找到以下三个gadget：

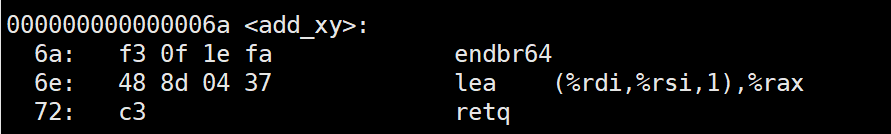
movq %rsp,%rax（48 89 e0）

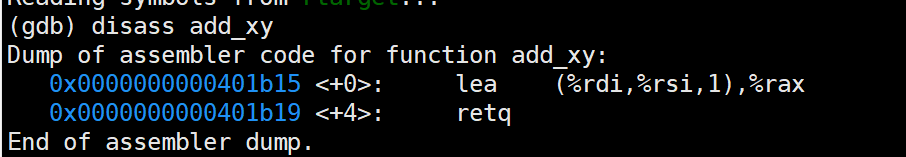




所以第一个地址为0x401be7

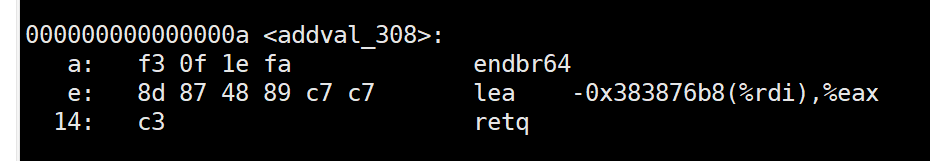
add 0x37,%rax(04 37)

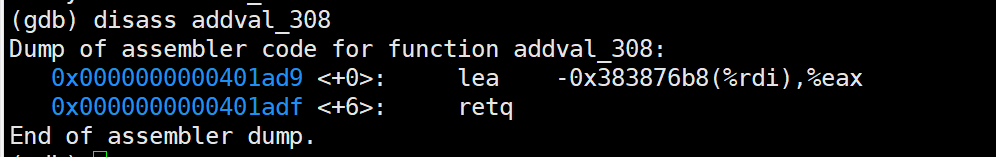




第二个地址为0x401b17

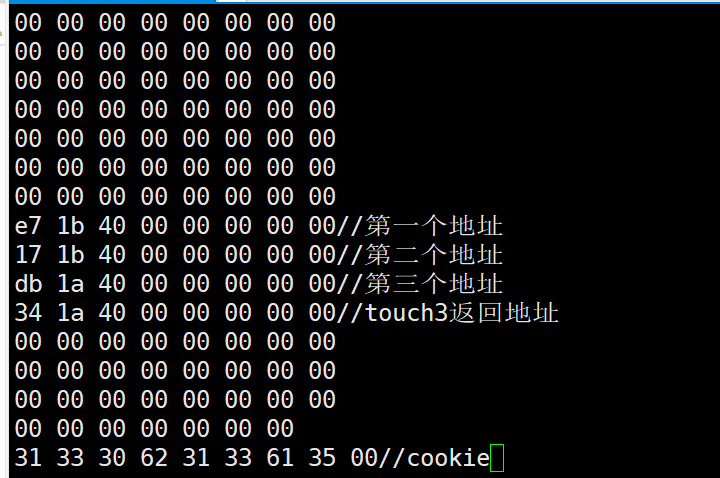
mov %rax,%rdi(48 89 c7)





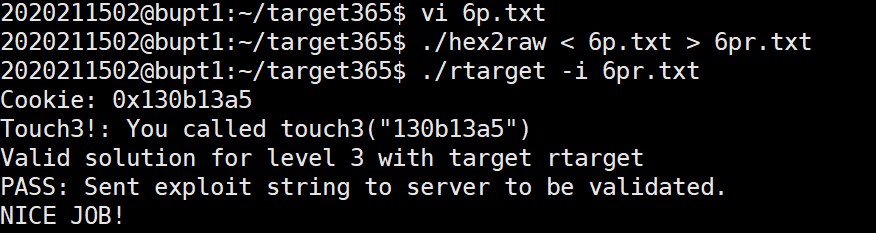
第三个地址为401adb

因此攻击序列为：



同上 ./hex2raw < 6p.txt > 6pr.txt

执行命令 ./rtarget -i 6pr.txt



五、总结体会

总体花费2天，前四个是第一天晚上完成，第五个第一天晚上没成功，到了第二天中午又经过尝试才完成，感觉还是挺难的，找了很多帮助才完成这次实验，但总体还是有很多收获的。