实验报告

一、完成度

五个问题全部解决

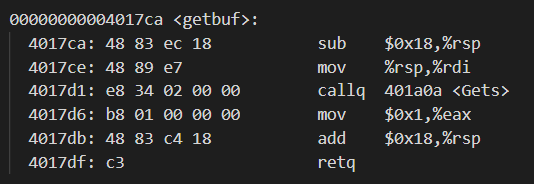
二、准备过程

用objdump命令将ctarget、rtarget都进行反汇编，得到对应的汇编代码

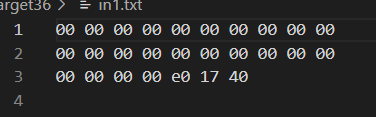
二、详细过程

1.phase1

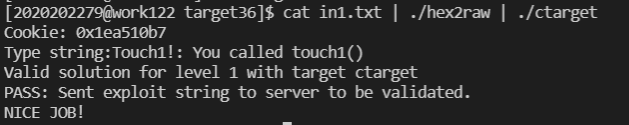
第一题有助教的演示和解说，只要通过调整输入让getbuf函数的return address变为touch1的地址，即可实现对函数touch1的调用。



由getbuf函数的反汇编代码可以看出，在调用gets输入前，%rsp-0x18开辟了24个字节的栈空间，即%rsp+24的位置应该存放的是return address。所以第一个攻击串应该首先任意输入24个字节，然后输入touch1的地址，覆盖return address，即可调用到touch1.



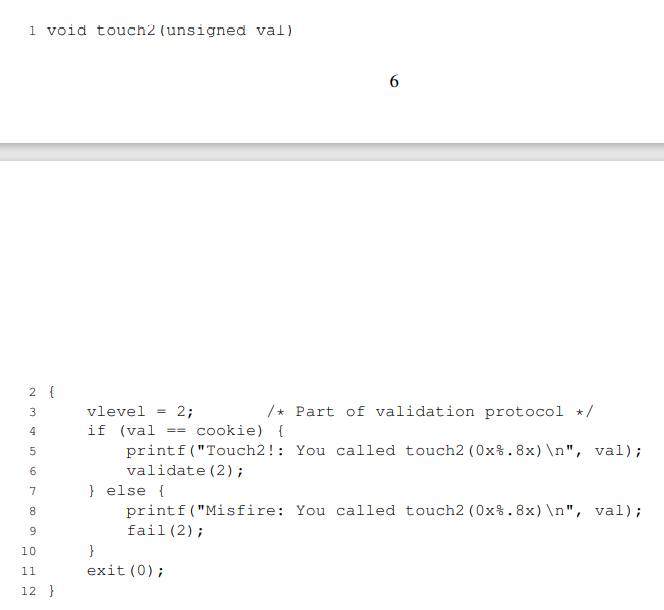
输入touch1的地址时还要注意，要采用小端方式，所以顺序是e0 17 40

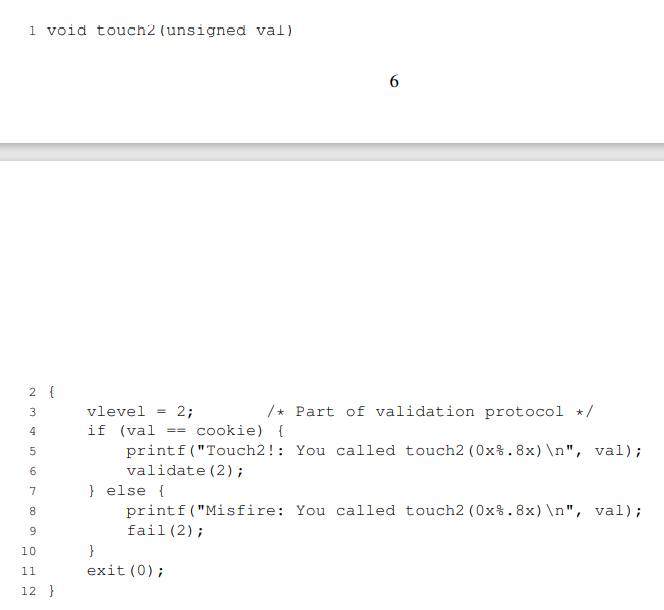


攻击成功！

2.phase2

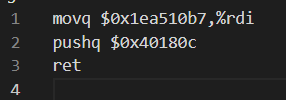
根据说明中touch2函数的c代码，可以看出调用touch2还需要一个传参数，参数的值等于cookie.txt中的值，我的值是0x1ea510b7



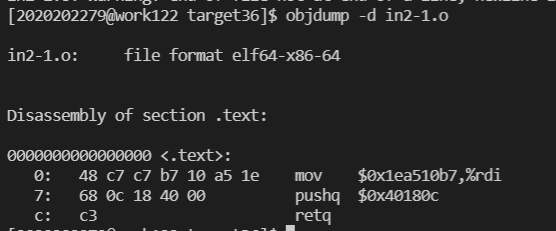


根据所学知识，函数的第一个参数应该放在寄存器%rdi中，所以我要做的应该是：先将正常的return address变成我注入代码的地址（可以直接在此时的栈顶即%rsp注入），然后将cookie的值传给%rdi，再跳转到touch2的地址进行函数调用。但是根据说明，不能使用jmp等指令，所以选择将touch2的地址压栈再ret弹出调用。

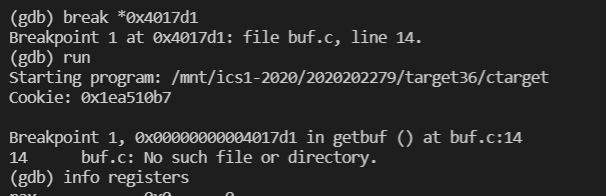
根据以上分析，写出要注入的汇编代码

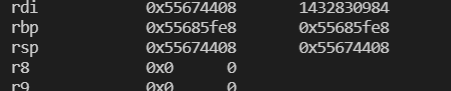


将其编译再反汇编，得到指令代码（16进制数字）

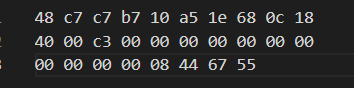


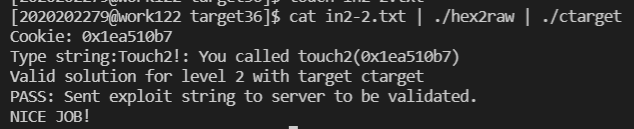
接下来，通过gdb调试，得到输入时栈顶指针%rsp的值





不同于第一个攻击串，这次我要在栈顶注入代码，所以攻击串先输入之前反汇编得到的机器码，然后凑够24字节后，输入寄存器%rsp的值。这样在函数返回的时候，就会转移到我注入的代码处执行，从而实现对touch2的调用。



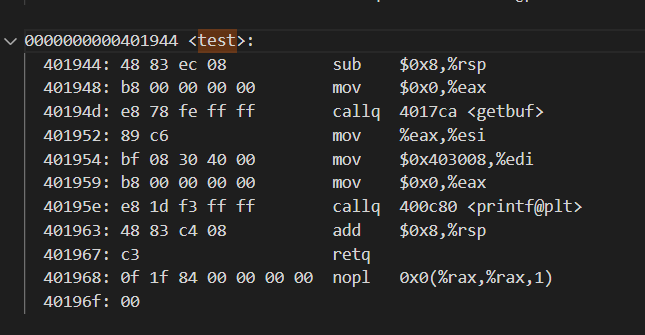


3.phase3

通过说明文档中的touch3函数和hexmatch函数的c语言代码，可以看出这次要传的参数是一个字符串，即cookie中的1ea510b7。根据提示，要注意字符串最后有一个‘\0’，即要多加一个字节的0。用man ascii得到该字符串的十六进制形式：31 65 61 35 31 30 62 37 00。

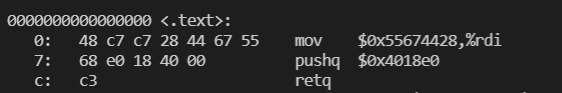
根据提示，如果将字符串数据放在getbuf函数的栈帧中，可能会被hexmatch和strncmp中随即开辟的空间所覆盖，所以应该把数据放在更安全的空间，选择test的栈空间

在test函数调用getbuf处设置断点，查看寄存器%rsp的值，即可得到test的栈空间的地址，为0x55674428

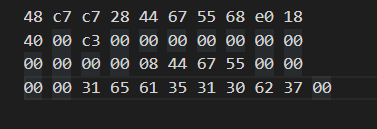


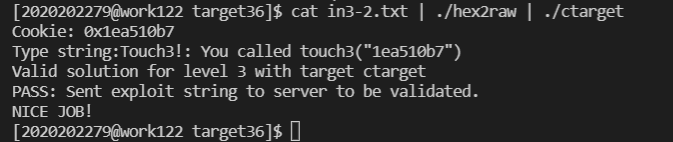


根据以上分析，这次要沿用第二题的思路注入代码，但不同的是参数变成字符串首地址，而且还要在栈空间中存放cookie转化的字符串。对于注入的代码，首先要将字符串首地址（test栈空间的地址）传给%rdi，然后调用touch3函数（地址0x4018e0），将其编译再反汇编得到如下指令代码：



综上所述，本次的攻击串应该首先输入反汇编得到的机器码，然后凑到24字节，再用与攻击串2中相同的寄存器%rsp的值覆盖正常的return address（让函数返回时转移到注入的代码处执行），然后输入cookie转化的16进制字符串即可

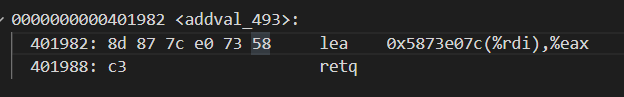


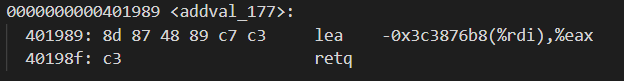


4.phase4

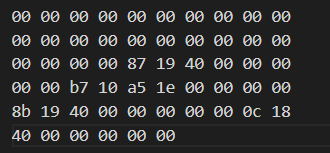
根据说明，这一次攻击不能直接注入代码，而是利用已有代码中的特定的指令序列来实现攻击。本题仍然要调用touch2，根据说明可以知道我应该利用已有代码中的pop和mov指令，将cookie的值传给寄存器%rdi。

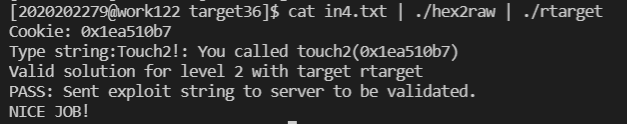
查看start\_farm到mid\_farm中的反汇编得到的机器码，发现合适的pop指令只有pop %rax，所以再寻找movq %rax，%rdi的指令，记录下它们的地址，分别是40 19 87和40 19 8b





所以攻击串应该先凑足24字节，然后用popq %rax指令的地址覆盖return address，再存放cookie的值（0x1ea510b7），再放movq %rax，%rdi指令的地址，再放touch2的地址

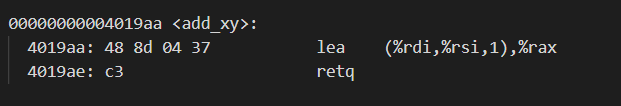




5.phase5

本题的思路与phase4类似，都是要利用已有的代码实现攻击。本题需要调用的也是touch3函数，并且传递字符串作为参数。而且与touch3的分析相似，为了防止hexmatch和strncmp的影响，也要把字符串存放在touch3的更高地址处。因为有栈随机化，所以要为插入的代码找一个基准，即%rsp。然后字符串的地址就是%rsp+bias（偏移量）

因为没有直接的加法指令，所以利用add\_xy中的lea指令实现加法，猜测要把%rsp中的栈指针地址传给%rdi，把偏移量bias的值传给%rsi，然后执行lea指令得到%rax的值即为字符串的首地址，再将该地址传给%rdi后调用touch3即可



通过检查所给代码中可以利用的指令（比较麻烦，不再赘述），可以设计出满足需求的指令：

（从正常的return address开始）

mov %rsp, %rax 40 19 d2

mov %rax, %rdi 40 19 7d

popq %rax 40 19 76

(存放偏移量的值) 48

movl %eax, %ecx 40 1a 69

movl %ecx, %edx 40 1a 7d

movl %edx, %esi 40 19 cb

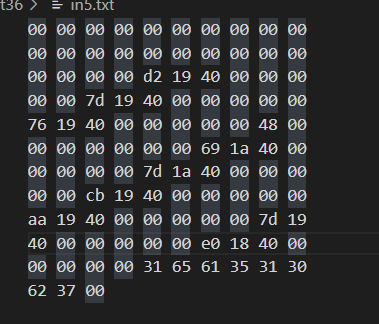
lea (%rdi,%rsi,1), %rax 40 19 aa

mov %rax, %rdi 40 19 7d

(存放touch3的地址) 40 18 e0

（存放cookie转换成的字符串） 31 65 61 35 31 30 62 37 00

由上可以看出，返回地址和字符串首地址之间共有9条指令，所以偏移量是72个字节，即0x48，所以得到攻击串为:



检验，通过！

