AttackLab 实验报告

郭丹琪 2018202067

Ctarget:

Attack1:

需要在 test 函数里在调用完 getbuf 函数返回时跳转到 touch1 函数，即改变调用 getbuf 函数时压入栈的返回地址。

在 getbuf 函数里，先有 sub $0x18,%rsp 语句让%rsp 减去 24，再用 mov %rsp,%rdi 语句让%rdi 存的内容等于%rsp，然后才调用 gets 函数来读取输入的内容。说明在 gets 函数里输入的内容是从调用 gets 前的%rsp 的位置开始存的，而该%rsp 位置距离需要改变的返回地址的位置是刚才的 24 个字节。再根据 gdb 调试得到 touch1 函数的起始地址是 0x555555555ad6。所以可知要先输入 24 个字节的内容，再输入 touch1 函数的起始地址，又因为是小端模式，所以要输入时应为 d6 5a 55 55 55 55 00 00。所以输入 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 d6 5a 55 55 55 55 00 00 即可。

Attack2:

该题需要在调用 touch2 函数前，把%rdi 里存的值设为 cookie 的值，所以需要在输入字符串的时候在里面加一段能将%rdi 里存的值设为 cookie 的值的代码。

找到 cookie 的值为 0x68e2d99a，因为在将%rdi 里存的值设置完后还需调用 touch2 函数，还得通过将 touch2 函数的地址当成函数返回时跳转的地址，所以除了设置%rdi 里存的值，还得加个 ret 语句来使能跳转到 touch2 函数。所以对应的代码和编码为

```
0000000000000000 <.text>:
   0:   bf 9a d9 e2 68          mov    $0x68e2d99a,%edi
   5:   c3                      retq
```

用 gdb 调试得到在输入字符串的时候，存的地址为 0x55678138。又因为 test 函数和 getbuf 函数与 attack1 的相同，即存返回地址的位置也相同。所以输入的字符串首先是上述代码对应的 bf 9a d9 e2 68 c3，然后可以输入任意的 18 个字节的内容。而上述代码在存入的时候，起始地址为 0x55678138，所以这里要再输入 38 81 67 55 00 00 00 00，来让程序能跳转到需要执行的上述代码。而在这段代码执行后返回，需要跳转到 touch2 函数的位置，所以还得再输入 touch2 函数的地址，根据 gdb 调试得到 touch2 函数的起始地址是

0x555555555b04，所以得再输入 04 5b 55 55 55 55 00 00。

所以上面所有的需要综合起来，需要输入 bf 9a d9 e2 68 c3 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 38 81 67 55 00 00 00 00 04 5b 55 55 55 55 00 00，即为插入的代码、凑长度的内容、插入的代码的地址、touch2 函数的地址。

Attack3：

题目要求在调用 touch3 前，将 cookie 的值当成字符串作为 touch3 的参数，即设置%rdi 里存 cookie 字符串的地址。

因为 touch3 里调用的 hexmatch 函数在运行的时候会覆盖之前存输入内容的栈，所以 cookie 的内容不能存在存返回地址前的栈里，而插入的代码因为在 touch3 函数开始前就已执行了，所以可以存在存返回地址前的栈里。

所以考虑先输入要执行的代码，然后到存返回地址的位置存插入的代码的地址，然后是 touch3 函数的地址，然后是 cookie 字符串。输入的内容存的位置的起始地址为 0x55678138，计算后得到要存 cookie 的位置的地址为 0x55678160。所以插入的代码为

```
48 c7 c7 60 81 67 55     mov    $0x55678160,%rdi
c3                       retq
```

然后再用 gdb 调试得到 touch3 函数的地址为 0x0000 5555 5555 5c1b。cookie 为 0x68e2d99a，当成字符串后对应的 16 进制为 36 38 65 32 64 39 39 61 。

所以综合起来要输入 48 c7 c7 60 81 67 55 c3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 38 81 67 55 00 00 00 00 1b 5c 55 55 55 55 00 00 36 38 65 32 64 39 39 61 00，即为插入的代码、凑长度的内容、插入的代码的地址、touch3 函数的地址、cookie 字符串的内容。

Vtarget：

Attack1：

因为栈的地址是随机的，所以需要调用已有的函数来完成。该题是要在调用 touch2 函数前，把%rdi 里存的设为 cookie 的值。因为程序不会执行插入代码，所以 cookie 的值也无法直接赋给%rdi，所以只能将 cookie 的值作为输入的一部分。

当 cookie 的内容是输入的一部分时，cookie 的内容就存在了栈里，所以只能用 pop 操

作来把值存到寄存器里。根据给出的函数和 pop 操作的编码，发现在 addval_447 函数里有 popq %rax 的编码。因为要把值存到%rdi 里，所以再找 movq %rax，%rdi 的编码，发现在 addval_320 里有符合的编码。所以确定要调用这两个函数中的部分内容来把%rdi 里存的内容设为 cookie 的值。

要先调用 addval_447 函数里需要的部分，即在第一个 return address 的位置要存 addval_447 函数里要运行的部分的地址，在调试中得到 addval_447 函数的地址为 0x0000 5555 5555 5cbe，而该函数的对应编码为 8d 87 13 c8 58 90 c3 而需要的编码是从 58 开始的，所以要存入的地址是 0x0000 5555 5555 5cc2。同样得到要存的 addval_320 函数里需要的部分的地址为 0x0000 5555 5555 5cc7。

因为 pop 是在调用 addval_447 函数部分就进行的，所以在输入 addval_320 函数部分的地址前要先输入 cookie 的值。所以综合起来，要输入 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 c2 5c 55 55 55 55 00 00 9a d9 e2 68 00 00 00 00 c7 5c 55 55 55 55 00 00 04 5b 55 55 55 55 00 00，即为凑长度的内容、popq %rax 代码的地址、cookie 的值、movq %rax，%rdi 代码的地址、touch2 函数的地址。

Attack2：

题目要求在调用 touch3 前，将 cookie 的值当成字符串作为 touch3 的参数，即设置%rdi 里存的为 cookie 字符串的地址。

根据给出的代码的编码，要得到栈的地址，只能通过%rsp。在给出的函数里，只能找到 movq %rsp，%rax 的对应编码，所以只能先将%rsp 里存的栈的地址存到%rax 里，再通过 movq %rax，%rdi 来把地址存到%rdi 里。但是因为这两步操作不在同一个函数，中间会有个返回的操作，所以如果直接将 cookie 存在这里%rsp 指向的位置，则前一个函数在返回的时候栈里存的 cookie 就会被当成返回地址，程序就会出错，所以此处不能直接存 cookie。

所以考虑将 cookie 存在离原本%rsp 里的地址有个偏移量的地方。因为 cookie 存的位置相对于存其他内容的位置是人为确定的，即该偏移量在给出的函数里是没有的，所以也只能通过在输入的时候输入该偏移量，然后通过 pop 操作得到该偏移量。

之后需要将该偏移量加到原本存着的地址上，在给出的函数里找到了 add_xy 函数，函数中的 lea (%rdi,%rsi,1),%rax 可以完成对地址的偏移。可以确定在调用该函数时，%rdi 里要存原本%rsp 指向的地址，%rsi 里存偏移量，然后得到的 cookie 的地址就存到了%rax 中。

但是与上题相同，存在栈里的偏移量只能 pop 到%rax 中，所以在 pop 执行完，lea (%rdi,%rsi,1),%rax 执行前，需要将该值存到%rsi 中。但是根据给出的编码发现函数中没有直接让%rax 里的内容存到%rsi 的操作，只能一步一步间接转化。在给出的函数里找到了 movl %eax,%ecx、movl %ecx,%edx、movl %edx,%esi 的编码，所以要在 pop 执行完后，依次执行这几个代码所在部分，让最终%rsi 里存着偏移量。之后就可以执行 lea (%rdi,%rsi,1),%rax，得到 cookie 的地址，再执行 movq %rax, %rdi，然后再调用 touch3 函数即可。因为考虑将 cookie 存在这些操作外，所以经过计算得到偏移量为 0x48。

所以综合起来要输入 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 aa 5d 55 55 55 55 00 00 c7 5c 55 55 55 55 00 00 c2 5c 55 55 55 55 00 00 48 00 00 00 00 00 00 00 c9 5d 55 55 55 55 00 00 af 5d 55 55 55 55 00 00 02 5d 55 55 55 55 00 00 fa 5c 55 55 55 55 00 00 c7 5c 55 55 55 55 00 00 1b 5c 55 55 55 55 00 00 36 38 65 32 64 39 39 61，即为凑长度的内容、movq %rsp，%rax 部分的地址、movq %rax,%rdi 部分的地址、popq %rax 部分的地址、计算出的偏移量 0x48、movl %eax,%ecx 部分的地址、movl %ecx,%edx 部分的地址、movl %edx,%esi 部分的地址、lea (%rdi,%rsi,1),%rax 部分的地址、movq %rax,%rdi 部分的地址、touch3 的地址。