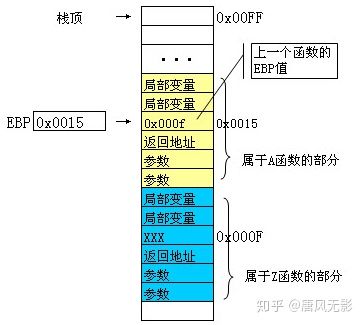
1.修改返回地址，让其指向溢出数据中的一段指令（shellcode）

2.修改返回地址，让其指向内存中已有的某个函数（return2libc）

3.修改返回地址，让其指向内存中已有的一段指令（ROP）

4.修改某个被调用函数的地址，让其指向另一个函数（hijack GOT）



1.因为sh.sendline(payload)是一次把数据送入，并且覆盖数据所以返回地址被覆盖成system函数的地址了。返回地址前面的参数又或者说不是参数是别的东西也没关系都被覆盖了覆盖成为一个EBP的地址和一个参数。当system函数执行时，会向前面调用一个参数，而且system函数在c语言里面的样式是int system(char \*command);只需要command命令的地址就可以了，我们在找到/bin/sh字符串的地址后就可以执行system(/bin/sh)，就可以拿到权限了。这里有一点，caller调用函数和callee被调用的函数在栈上并不是连续在一起的，但是逻辑上是可以这样画出来了。因为call指令把下一个指令的地址压栈后，再jmp到函数内部的。栈上只保留了函数的各种信息，但是具体函数的各种代码不在栈上，而在内存的某一个地方。<.code上>

2. system() 函数地址应该是多少？

要回答这个问题，就要看看程序是如何调用动态链接库中的函数的。当函数被动态链接至程序中，程序在运行时首先确定动态链接库在内存的起始地址，再加上函数在动态库中的相对偏移量，最终得到函数在内存的绝对地址。说到确定动态库的内存地址，就要回顾一下 shellcode 中提到的内存布局随机化（ASLR），这项技术也会将动态库加载的起始地址做随机化处理。所以，如果操作系统打开了 ASLR，程序每次运行时动态库的起始地址都会变化，也就无从确定库内函数的绝对地址。在 ASLR 被关闭的前提下，我们可以通过调试工具在运行程序过程中直接查看 system() 的地址，也可以查看动态库在内存的起始地址，再在动态库内查看函数的相对偏移位置，通过计算得到函数的绝对地址。

最后，“/bin/sh” 的地址在哪里？

可以在动态库里搜索这个字符串，如果存在，就可以按照动态库起始地址＋相对偏移来确定其绝对地址。如果在动态库里找不到，可以将这个字符串加到环境变量里，再通过 getenv() 等函数来确定地址。

可以在动态库里搜索这个字符串，如果存在，就可以按照动态库起始地址＋相对偏移来确定其绝对地址。如果在动态库里找不到，可以将这个字符串加到环境变量里，再通过 getenv() 等函数来确定地址。

