Return-to-libc 攻击与 ROP

S5.1. 用 `"-z noexecstack"` 编译 C 程序后，跳转到栈上代码的缓冲区溢出攻击本应失败，但一些学生仍然攻击成功，且操作没有错误。可能是什么原因？

S5.2. 函数尾声会从保存帧指针的槽位恢复 `ebp`。缓冲区溢出在覆盖返回地址时通常也破坏了该槽位，所以返回后 `ebp` 是攻击者写入的任意值。这会影响跳转到 `system()` 的 return-to-libc 攻击吗？

S5.3. 希望不调用 `system()`，而是通过 `execve()` 执行 `"/bin/sh"`。请说明如何布置攻击栈。输入允许包含零字节，假设复制使用 `memcpy()` 而非 `strcpy()`。

S5.4. `system()` 会调用 `/bin/sh`;若它链接到较新的 `bash`,shell 在发现有效用户 ID 与真实用户 ID 不同时会主动放弃特权。若仍想在 return-to-libc 攻击中使用 `system()`，如何绕过这一保护？输入可包含零字节，假设使用 `memcpy()` 复制。

S5.5. 发起 return-to-libc 攻击时，攻击者不跳到 `system()` 的正常入口，而跳到其函数序言之后的第一条指令。此时应如何构造输入？

S5.6. 地址空间布局随机化能否防御或缓解 return-to-libc 攻击？

S5.7. `Linux` 中的 ASLR 是否对库函数的地址进行随机化，例如 `system()`？

S5.8. ★
假设内存中没有可直接返回到的 `system()`，但已知以下指令序列能启动 shell：

$$A_1, \ldots, A_m; \qquad B_1, \ldots, B_n; \qquad C_1, \ldots, C_t.$$

这些指令并不连续，而是分成位于三个函数末尾的子序列 A、B、C。每段都以 `ret` 结束，地址如下：

```
0xAABB1180:   A_1, ..., A_m, ret
0xAABB2290:   B_1, ..., B_n, ret
0xAABB33A0:   C_1, ..., C_t, ret
```

覆盖返回地址为 `0xAABB1180` 后，易受攻击函数会先执行子序列 A。还应在栈上放置哪些值，才能让 A 末尾的 `ret` 跳到 B，再让 B 末尾的 `ret` 跳到 C？

这种方法称为面向返回编程（ROP），可看作 return-to-libc 的推广。传统 return-to-libc 依赖 `system()` 等完整函数；ROP 则从程序或共享库中挑选已经存在的短

指令序列，即 gadget，并通过栈把它们串联起来。gadget 通常以 `ret` 结束，使执行流能从栈上取出下一段地址。合理组合 gadget 后，即使没有现成的目标函数，也可完成复杂操作。

S5.9. `foo()` 把用户输入复制到其栈帧内的缓冲区，存在溢出漏洞。希望利用它依次调用 `bar()`、`bar()`、`bar()`、`xyz(3, 5)`，最后调用 `exit()`。假设已知各函数地址，请说明在 `foo()` 返回前应如何布置栈，并画出栈图。

```
|                  |
|   provide        |
|   your           |
|   answer         |
|                  |
|                  | <-- ebp of foo()
```

S5.10. `foo()` 存在栈缓冲区溢出，希望最终调用库函数 `xyz(0)`。不能跳过 `xyz()` 的函数序言，输入中也不能出现零字节。另有函数 `setzero(addr)`，可把地址 `addr` 处的四字节内存清零。请构造输入并画出栈图；不要求 `xyz(0)` 返回后程序继续正常运行。

```
|                  |
|   provide        |
|   your           |
|   answer         |
|                  |
|                  | <-- ebp of foo()
```

S5.11. 本题在问题 S5.10. 上增加要求。`foo()` 存在栈缓冲区溢出，希望依次执行以下调用链：

$$\texttt{xyz(0x11111111)} \longrightarrow \texttt{xyz(0)} \longrightarrow \texttt{xyz(0x22222222)}.$$

不能跳过 `xyz()` 的函数序言，输入中也不能出现零字节。可使用 `setzero(addr)` 把指定地址的四字节内存清零。请说明如何构造输入并画出栈图；不要求最后一次 `xyz()` 返回后程序继续正常运行。