CTF PWN入门实验手册

一、 实验目标

- 理解Linux x86-64系统调用，掌握`shellcode`编写方法。
- 了解数组越界和栈溢出漏洞原理。
- 学习GOT劫持和ROP（返回导向编程）漏洞利用技术。
- 熟练使用`pwntools`进行漏洞利用脚本编写。

二、 实验环境与工具

- **工具**: `pwntools`, `pwndbg`（可选，调试用）
- **连接信息**: 本次实验包含4个题目和1个分数平台，请使用`nc`或`pwntools`连接到以下地址和端口：
  - **分数平台 (提交Flag/查分)**: `[服务器IP]:8888`
  - **题目1 (shellcode: open)**: `[服务器IP]:9001`
  - **题目2 (shellcode: openat)**: `[服务器IP]:9002`
  - **题目3 (GOT Hijack)**: `[服务器IP]:9003`
  - **题目4 (ROP)**: `[服务器IP]:9004`

三、 实验题目详解

题目一 & 题目二: Shellcoding

知识点: x86-64汇编、系统调用（Syscall）、Seccomp安全机制。

分析: 程序读取并执行你的shellcode，但`seccomp`限制了可用的系统调用。题目一限制为`open, read, write`，题目二限制为`openat, read, write`。

解法:

1. 使用`pwntools.shellcraft`来生成漏洞利用代码。
2. **题目一**: 构造`open('/flag')` + `read(fd, ...)` + `write(1, ...)`的调用链。
3. **题目二**: 构造`openat(自己查查怎么写)` + `read(fd, ...)` + `write(1, ...)`的调用链。

题目三: 数组下标越界与GOT劫持

知识点: 内存布局（.bss, .got.plt）、数组越界、GOT劫持、**ASLR**、**Libc基址泄露**。

分析: 程序读写一个bss段的全局数组`notes`，且完全没有对数组下标`idx`进行范围检查。由于现代操作系统ASLR（地址空间布局随机化）的存在，我们无法预知`system`函数的准确地址。同时，程序本身没有调用`system`，所以我们不能直接使用`system@plt`。

解法 (两步走战略):

1. **第一步：信息泄露**
   - **目标**: 获取一个已加载的libc函数（如`puts`）在内存中的真实地址。
   - **方法**: `puts`函数被程序调用，其真实地址在首次调用后会被填入`puts@got`。我们可以利用数组越界漏洞，**读取**`puts@got`的内容。

- **计算**: 使用`pwntools`计算偏移。`puts@got`在`elf.got['puts']`，数组基地址在`elf.symbols['notes']`，`offset = (?) // 8`，然后选择 "Read a note" 功能来泄露地址。

2. **第二步：执行代码**
    - **目标**: 计算出`system`函数的真实地址，并劫持GOT表来执行它。
    - **计算Libc基址**: 我们有了`puts`的运行时地址，再结合题目提供的`libc.so.6`文件中`puts`的偏移地址，就可以计算出`libc`被加载到内存的基地址，`puts`的相对偏移为`libc.symbols['puts']`。
    - **计算System地址**: 有了基地址，就能计算`system`的地址，`system`的相对偏移为`libc.symbols['system']`。
    - **利用**: 调用写功能，使用相同的`offset`，将计算出的`system_addr`写入`puts@got`。
    - **触发**: 调用菜单中的`Secret`功能。该功能会执行`puts("sh")`，但由于`puts`的地址已被劫持，实际执行的是`system("sh")`，从而获得shell。

题目四: 简单ROP

知识点: 栈溢出、ROP、x86-64函数调用约定。

分析: 程序存在一个明显的栈溢出。我们的目标是控制程序执行流，去调用`gadget_func`函数，并为其传递参数`'s', 'h', 0`，最终执行`system("sh")`。

解法:

1. **调用约定**: x86-64下，函数的前三个参数通过`rdi`, `rsi`, `rdx`寄存器传递。
2. **寻找Gadgets**: 我们需要能够控制这三个寄存器的gadgets。程序中已提供`pop rdi; ret`, `pop rsi; ret`, `pop rdx; ret`。
3. **构造ROP Chain**: 使用`pwntools`的`ROP`模块可以极大地简化这个过程。
4. **Payload**: 构造`填充数据 + ROP Chain`。填充数据长度为`缓冲区大小(32) + ? = 自己想`。发送payload即可。