操作系统第三次作业

于新雨 计25 2022010841

1

用了 linux 提供的 backtrace 和 backtrace_symbols 的接口

```
❯ ./print
Backtrace (5 frames):
#0: ./print(+0x11fb) [0x55e51e3861fb]
#1: ./print(+0x12d8) [0x55e51e3862d8]
#2: /lib/x86_64-linux-gnu/libc.so.6(+0x29d90) [0x7ff30517cd90]
#3: /lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0x80) [0x7ff30517ce40]
#4: ./print(+0x1105) [0x55e51e386105]
```

代码如下，环境是 linux user mode

```c
#include <stdio.h>
#include <execinfo.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

void print_stack_frame() {
    void *buffer[100];
    char **strings;
    int nptrs;

    // get backtrace addresses
    nptrs = backtrace(buffer, 100);
    printf("Backtrace (%d frames):\n", nptrs);

    // get symbol names for addresses
    strings = backtrace_symbols(buffer, nptrs);
    if (strings == NULL) {
        perror("backtrace_symbols");
        return;
    }

    // print info
    for (int i = 0; i < nptrs; i++) {
        printf("#%d: %s\n", i, strings[i]);
    }

}

int main() {
    print_stack_frame();
    return 0;
}
```

2

选择 KDB 和 KGDB
工作原理：KDB 是 Linux 内核的内置调试器，当触发时直接在本机控制台提供命令行界面进行内核调试。kgdb本质就是在kernel内部建立一个gdb server，通过串口就可以和gdb进行通信来调试了，他们的激活和配置是通过特定的内核启动参数，通过 /proc 或 /sys 文件系统的特殊文件等进行配置

调试用途：KDB 用于在系统发生崩溃或被主动触发时，在本地 console 提供基本的内核调试功能，如查看内存、寄存器和堆栈信息（类似于 Windbg）。KGDB 则主要用于远程内核开发场景，可以实现更复杂的交互式内核调试，包括在 debug 手写驱动时的设置断点、单步执行和实时检查变量等情况。

配置使用方式：
主要见 这里
如设置本机和目标机的串口，关掉 kaslr(其实感觉也不是很必要)，开 gdb 这些

此外，笔者在学习 linux kernel pwn 时有调试 linux 内核的经历（主要是对题目中给的有漏洞的内核驱动文件或者 eBPF 模块进行调试），环境通常是用 qemu 跑那个 bzImage 文件，然后用 -s 选项开了1234 端口，在同个机子上开 gdb，用 `target remote :1234` 命令 attach 上去，然后下断点调试
其中一个启动 qemu 的命令如下

```
qemu-system-x86_64 \
    -kernel bzImage \
    -cpu qemu64,+smep,+smap,+rdrand \
    -m 512M \
    -smp 2 \
    -initrd ./core/rootfs.cpio \
    -append "console=ttyS0 quiet loglevel=3 oops=panic panic_on_warn=1 panic=-1
pti=on page_alloc.shuffle=1 kaslr" \
    -drive file=/flag,if=virtio,format=raw,readonly=on \
    -monitor /dev/null \
    -nographic \
    -no-reboot \
    -s
```