**strace分析系统调用**

计科1201 唐佳娣 1212300112

**1.需分析的源代码如下：**

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

char cmd[256];

void main(){

int ret;

printf("Hello world, this is Linux!");

while(1){

printf(">");

fgets(cmd,256,stdin);

cmd[strlen(cmd)] = 0;

if(fork() == 0){

execlp(cmd,NULL);

perror(cmd);

exit(errno);

} else {

wait(&ret);

printf("child process return %d \n",ret);

}

}

}

**2.使用以上代码创建test.c的源文件，gcc编译得到test可执行文件。使用strace得到的结果如下，并附对其相应的分析：**

strace跟踪系统调用，每一行就对应一个系统调用，格式为：

**系统调用的名称( 参数... ) = 返回值  错误标志和描述**

1.execve("./test", ["./test"], [/\* 59 vars \*/]) = 0

对于命令行下执行的程序，execve(或exec系列调用中的某一个)均为strace输出系统调用中的第一个。strace首先调用fork或clone函数新建一个子进程，然后在子进程中调用exec载入需要执行的程序(这里为./test)

2.brk(0) = 0x175f000

 以0作为参数调用brk，返回值为内存管理的起始地址(若在子进程中调用malloc，则从0x175f000地址开始分配空间)

3.access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

调用access函数检验/etc/ld.so.nohwcap是否存在

4.mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fed6291c000

使用mmap函数进行匿名内存映射，以此来获取8192bytes内存空间，该空间起始地址为0x7fed621c000

5.access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

6.open("/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

调用open函数尝试打开/etc/ld.so.cache文件，返回文件描述符为3

7.fstat(3, {st\_mode=S\_IFREG|0644, st\_size=79789, ...}) = 0

fstat函数获取/etc/ld.so.cache文件信息

8.mmap(NULL, 79789, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fed62908000

调用mmap函数将/etc/ld.so.cache文件映射至内存

9.close(3) = 0

close关闭文件描述符为3指向的/etc/ld.so.cache文件

10.access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

11.open("/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

12.read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\320\37\2\0\0\0\0\0"..., 832) = 832

 调用read，从/lib/x86\_64-linux-gnu/libc.so.6该libc库文件中读取832bytes，即读取[ELF](http://blogimg.chinaunix.net/blog/upfile2/110307091731.pdf)头信息

13.fstat(3, {st\_mode=S\_IFREG|0755, st\_size=1845024, ...}) = 0

14.mmap(NULL, 3953344, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fed62336000

15.mprotect(0x7fed624f1000, 2097152, PROT\_NONE) = 0

使用mprotect函数对0x7fed624f1000起始的2097152bytes空间进行保护(PROT\_NONE表示不能访问，PROT\_READ表示可以读取)

16.mmap(0x7fed626f1000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bb000) = 0x7fed626f1000

17.mmap(0x7fed626f7000, 17088, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fed626f7000

18.close(3) = 0

19.mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fed62907000

20.mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fed62905000

21.arch\_prctl(ARCH\_SET\_FS, 0x7fed62905740) = 0

设置架构的具体进程或线程状态为64位基

22.mprotect(0x7fed626f1000, 16384, PROT\_READ) = 0

23.mprotect(0x600000, 4096, PROT\_READ) = 0

24.mprotect(0x7fed6291e000, 4096, PROT\_READ) = 0

25.munmap(0x7fed62908000, 79789) = 0

调用munmap函数，将/etc/ld.so.cache文件从内存中去映射，与第八行8的mmap对应

26.fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

27.mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fed6291b000

28.fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

29.mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fed6291a000

30.write(1, "Hello world, this is Linux!>", 28) = 28

输出28字节到终端

31.read(0, "./a\n", 1024) = 4

从终端读取输入，此处为：./a（表示运行程序a，而a为创建的一个简单的可执行文件）

32.clone(child\_stack=0, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7fed62905a10) = 4998

clone函数创建一个新的子进程

33.wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 6}], 0, NULL) = 4998

等待子进程结束

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=4998, si\_status=6, si\_utime=0, si\_stime=0} ---

34.write(1, "child process return 1536 \n", 27) = 27

35.write(1, ">", 1) = 1

……从此处开始进入while循环的第二次，之后省略)