结题报告

陈思睿　梁恒宇　吕泓涛　汤力宇

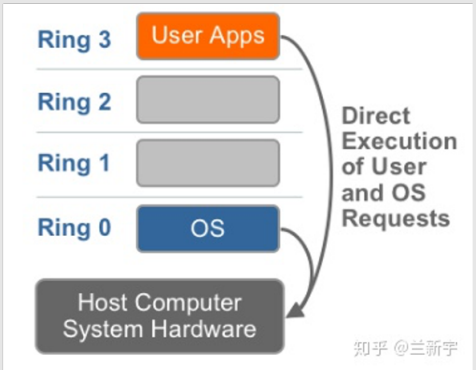
中国科学技术大学

2021 年 7 月 11 日

中国科学技术大学
University of Science and Technology of China

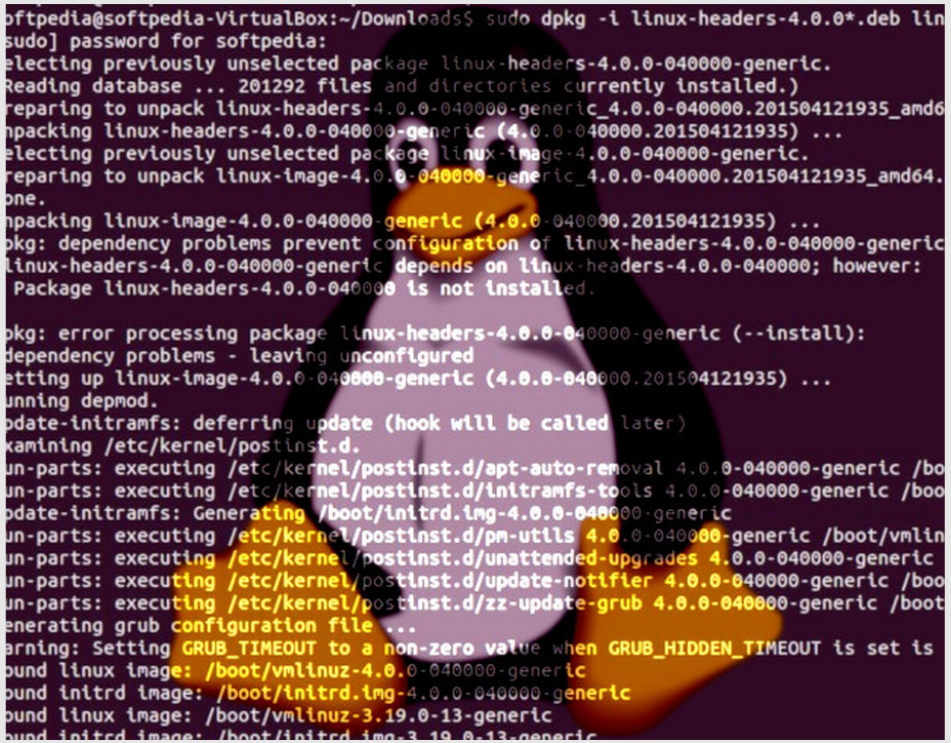
sBPF——介于用户态与内核态之间的轻量级沙盒

- 特权级别——linux 主要分为用户态与内核态
- 用户态进程——内核态进程
- 用户态内存——内核态内存
- **复杂的中断、调度、拷贝机制!!**
  给沙盒的实现带来了困难



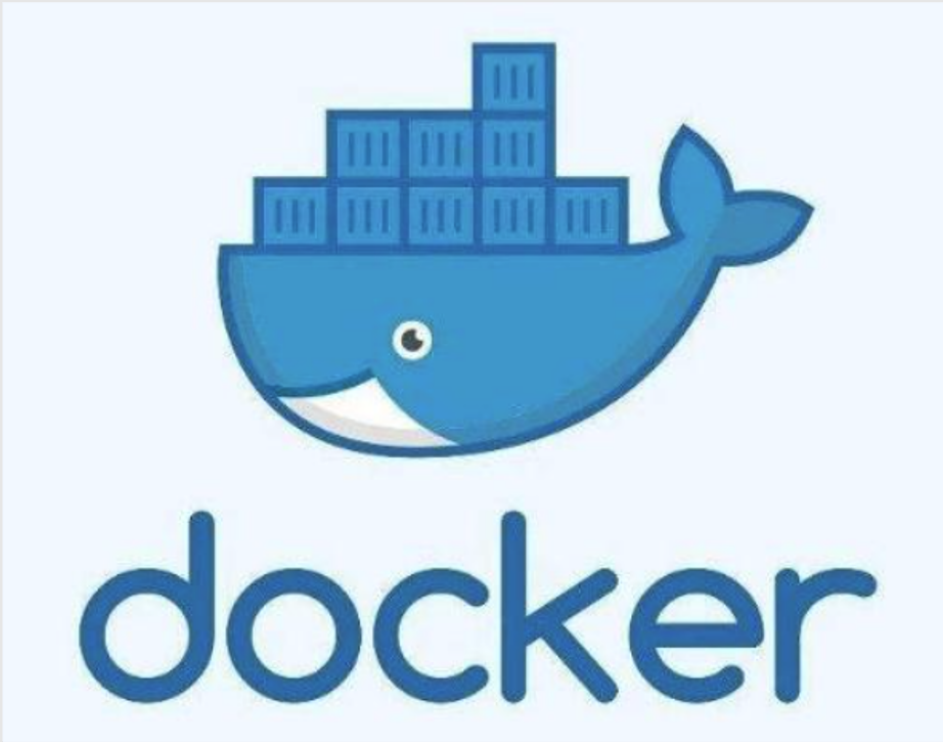
内核态沙盒工具

- Seccomp
- Namespace
- Cgroup
- 灵活性差
- 只能实现最基本的功能
- 大部分无法独立工作，需要用户态程序协作



用户态沙盒

- gVisor
- Docker
- 需要将数据流与控制流在内核态与用户态间多次中转
- 性能损失
- 面临程序被篡改的安全性问题



中国科学技术大学
University of Science and Technology of China

我们的解决方案：半用户态半内核态沙盒

用户：

- 编写沙盒处理程序
- 直接注入到内核中
- 得到一个高度自由的沙盒程序

内核：

- 提供沙盒处理程序挂在点
- 提供沙盒处理程序调用机制
- 在内核中直接完成沙盒执行

中国科学技术大学
University of Science and Technology of China

机制与策略相分离

- 少量内核代码定义写死的沙盒机制大量通用代码提供灵活的沙盒策略，实现不同的行为模式
- 我们做到了：
- 使用最少的内核代码，只起调用功能（20 行内）
- 提供丰富多样的沙盒行为模块代码（3.5 套不同的行为模式）
- 便于测试、升级、实现各种灵活的功能

中国科学技术大学
University of Science and Technology of China

在安全问题上我们的思考

- 数据安全是日常生活中最重要的计算机安全问题
- 企业机密
- 个人隐私
- 重要账密
- 数据绑架

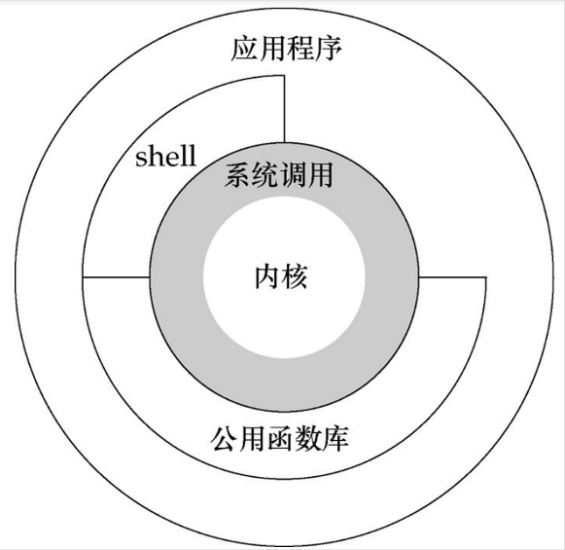
中国科学技术大学
University of Science and Technology of China

我们的选择与妥协

- 重点进行文件保护
- 其余方面通过现有的内核沙盒工具进行防护
- 减轻工作量
- 设计方法论的验证

介入方法

- 介入系统调用
- 在系统调用的进入位置添加静态插桩点，直接获取上下文参数，介入运行流程，交给内核态处理程序运行
- 对比传统沙盒：
- 使用 ptrace 等内核工具
- 动态插桩点对内核版本不稳定
- 用户态处理程序，上下文切换



沙盒处理程序的工作方式

- 直接利用钩子函数实现调用，由于运行在内核态，可以简单直接的使用内核 api 来进行编程
- 对比传统沙盒：
- 复杂的回调机制触发运行
- 依赖系统调用来维持自身运行
- 返回值依赖内核的复杂传回机制传回

中国科学技术大学
University of Science and Technology of China

管理程序工作方式

- 简单的调用被监测的程序
- 简单的诸如沙盒行为代码
- 通过成熟的 api 调用 cgourp 功能实现资源保护
- 整体结构简单
- 对比传统沙盒的工作方式
- 需要管理大量的数据结构和程序段，体量庞大，难以编写和调试

中国科学技术大学
University of Science and Technology of China

从代码角度上来说

- Loader：用户态管理程序
- sBPF_xxx：注入内核的沙盒行为代码
- test_programme：被隔离程序

中国科学技术大学
University of Science and Technology of China

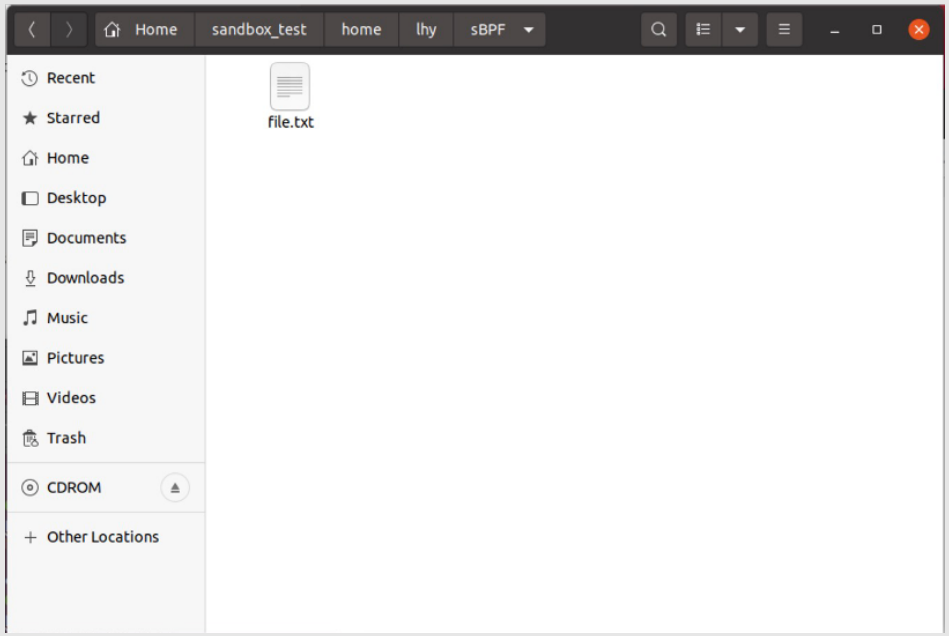
实现的文件保护行为模式

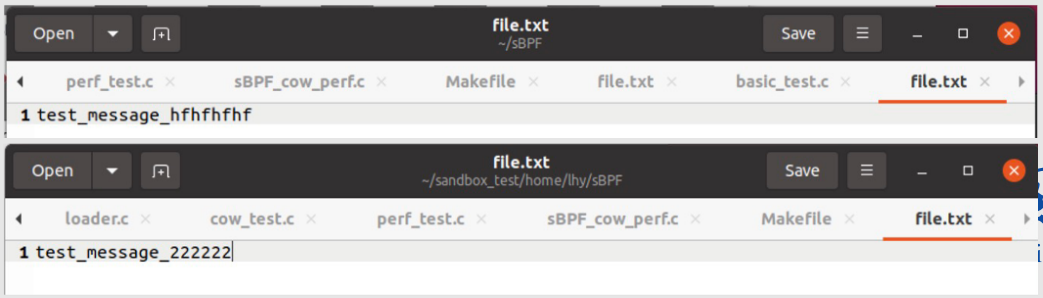
- `sBPF_redir` 简单的目录重映射模式完整的虚拟目录系统映射所有读写操作
- `sBPF_cow` 对写操作进行备份的 copy on write 模式允许读原始文件系统写入文件时，将文件复制到沙盒文件目录对修改过的文件的读写操作都映射到沙盒目录
- `sBPF_permission` 自定义文件访问权限模式

中国科学技术大学
University of Science and Technology of China

行为演示

```
lhy@ubuntu:~/sBPF/test_module$ ./loader sBPF_cow.ko /home/lhy/sandbox_test ./a.o
ut
pid=8883, u_mem=281474427917000, sdir=/home/lhy/sandbox_test
test_message_hfhfhfhf
test_message_222222
```

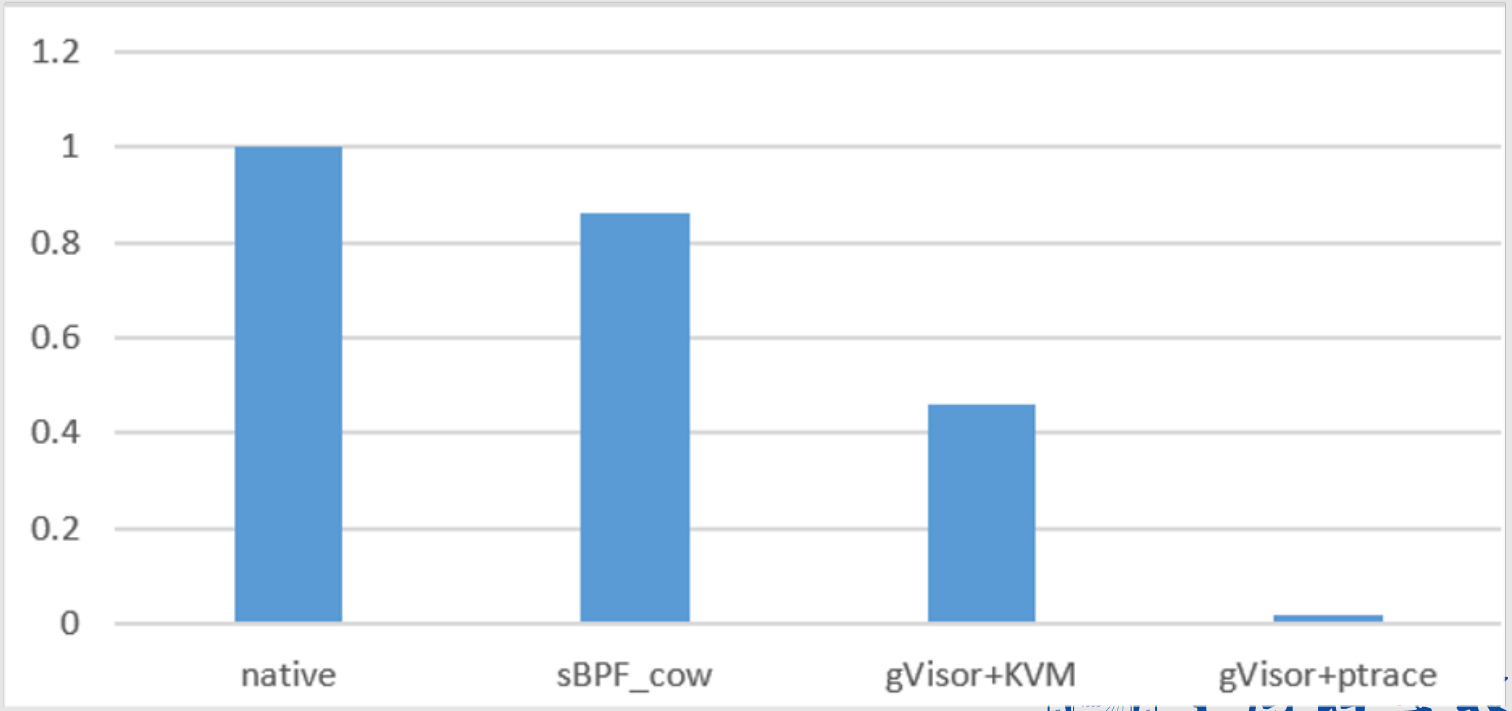




性能测试

```
lhy@ubuntu:~/sBPF/test_module$ ./perf
swap_space 281474114094120
这个程序的PID为: 8556
k
time:7.186629
```

```
lhy@ubuntu:~/sBPF/test_module$ ./perf
swap_space 281474451425432
这个程序的PID为: 8522
k
time:8.274673
```



University of Science and Technology of China

Q&A

中国科学技术大学
University of Science and Technology of China