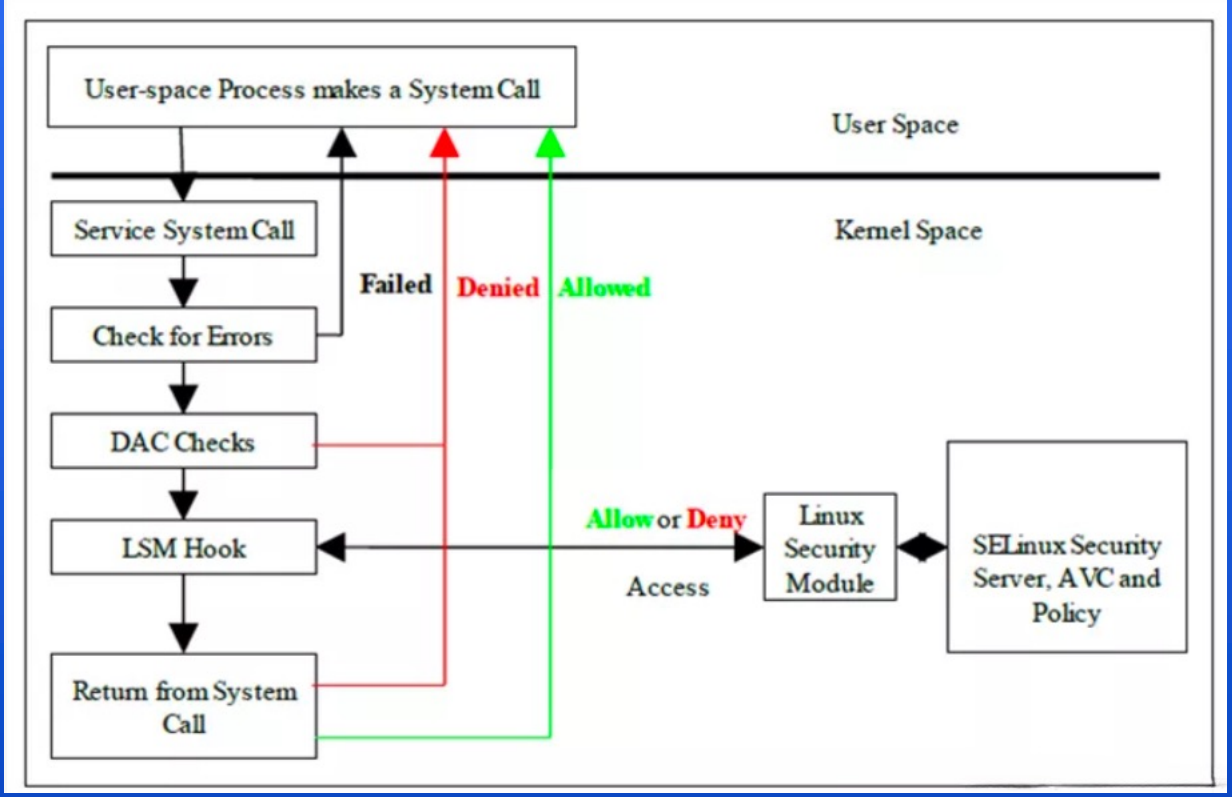
内核安全性探索：围绕eBPF安全审计及safeguard项目

目录

内核LSM框架



- 进程通过系统调用(System Call) 访问某个资源，进入Kernel 后，先会做基本的检测，如果异常则直接返回；
- Linux Kernel DAC 审查，如果异常则直接返回；
- 调用Linux Kernel Modules 的相关hooks，对接到SELinux的hooks，进而进行MAC 验证，如果异常则直接返回；
- 访问真正的系统资源；
- 返回用户态，将结果反馈。

KRSI

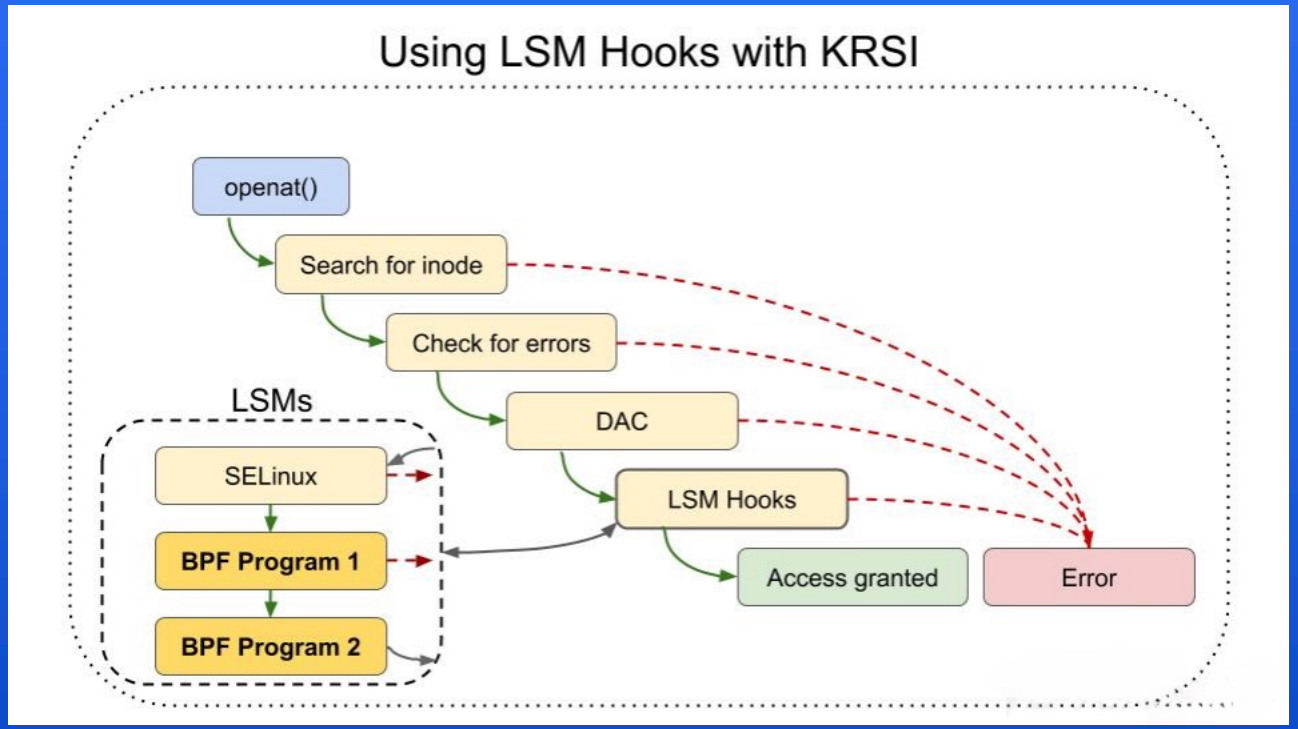
KRSI (Kernel Runtime Security Instrumentation)的原型通过LSM (Linux security module)形式实现，可以将 eBPF program 挂载到 kernel 的 security hook（安全挂钩点）上。内核的安全性主要包括两个方面：Signals 和 Mitigations，这两者密不可分。

- Signals：意味着系统有一些异常活动的迹象、事件
- Mitigations：在检测到异常行为之后所采取的告警或阻断措施

LSM

KRSI 基于 LSM 来实现，KRSI 的工作重心是全面监视系统行为，以便检测攻击。

从这种角度来看，KRSI 可以说是内核审计机制的扩展，使用eBPF 来提供比目前内核审计子系统更高级别的可配置性。



KRSI

1）KRSI 允许适当的特权用户将 BPF 程序挂载到 LSM 子系统提供的数百个钩子中的任何一个上面。

2）为了简化这个步骤，KRSI 在 /sys/kernel/security/bpf 下面导出了一个新的文件系统层次结构——每个钩子对应一个文件。

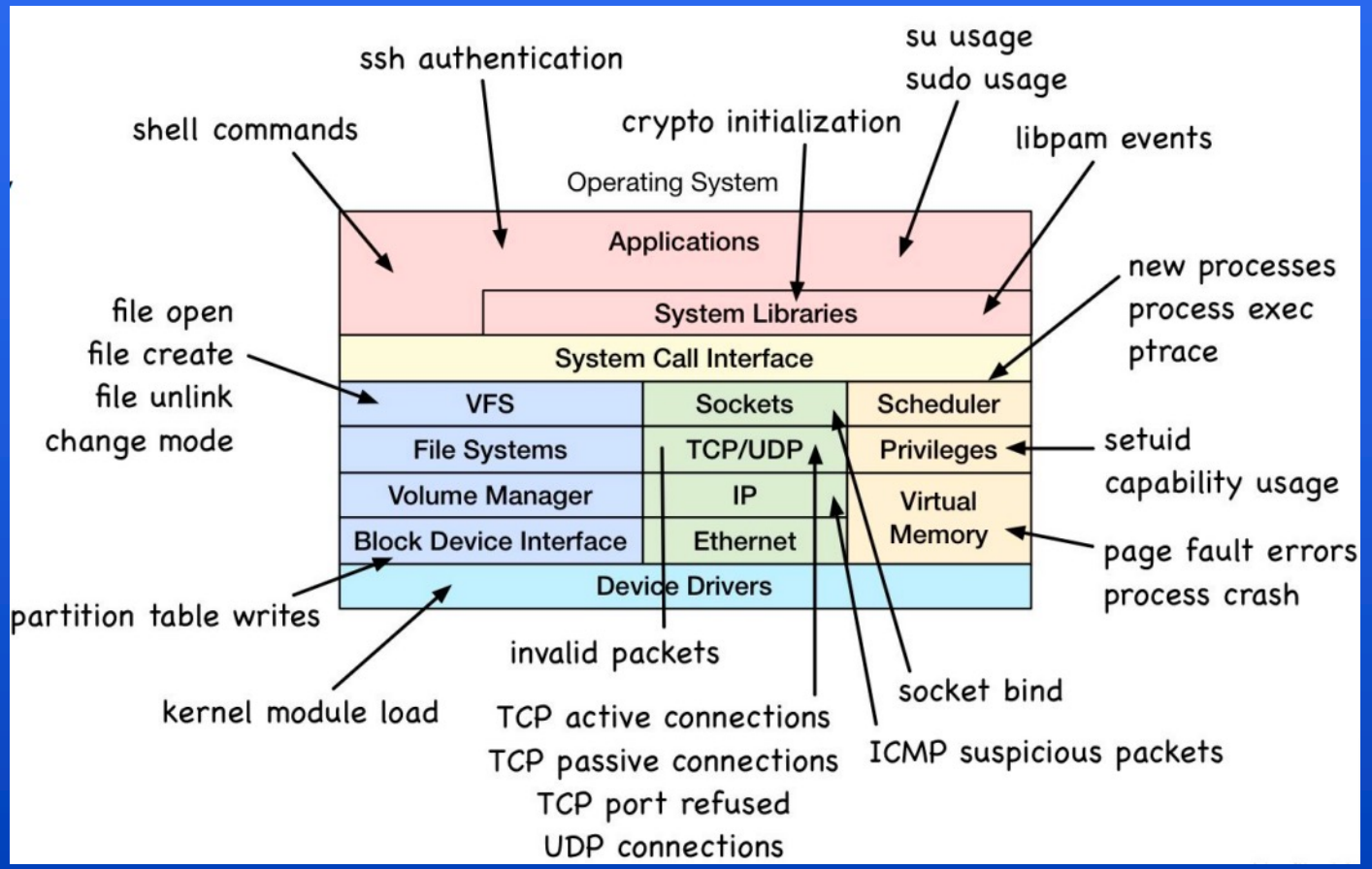
3）可以使用 bpf()系统调用将BPF程序(新的 BPF_PROG_TYPE_LSM 类型)挂载到这些钩子上，并且可以有多个程序挂载到任何给定的钩子。

4）每当触发一个安全钩子时，将依次调用所有挂载的 BPF 程序，只要任一 BPF 程序**返回错误状态**，那么请求的操作将被拒绝。

5）KRSI 能够从函数级别做阻断操作，相比进程具有更细粒度，危险程度也会小得多。

KRSI

eBPF 安全跟踪点



safeguard项目

针对操作系统、内核安全，safeguard是一个基于eBPF的Linux安全防护系统，可以实现安全操作的拦截及审计记录。项目采用libbpfgo库，使用go语言实现顶层控制。目前项目已在openEuler sig-ebpf社区开源，链接：https://gitee.com/openeuler/safeguard。

safeguard项目

审计控制

文件:

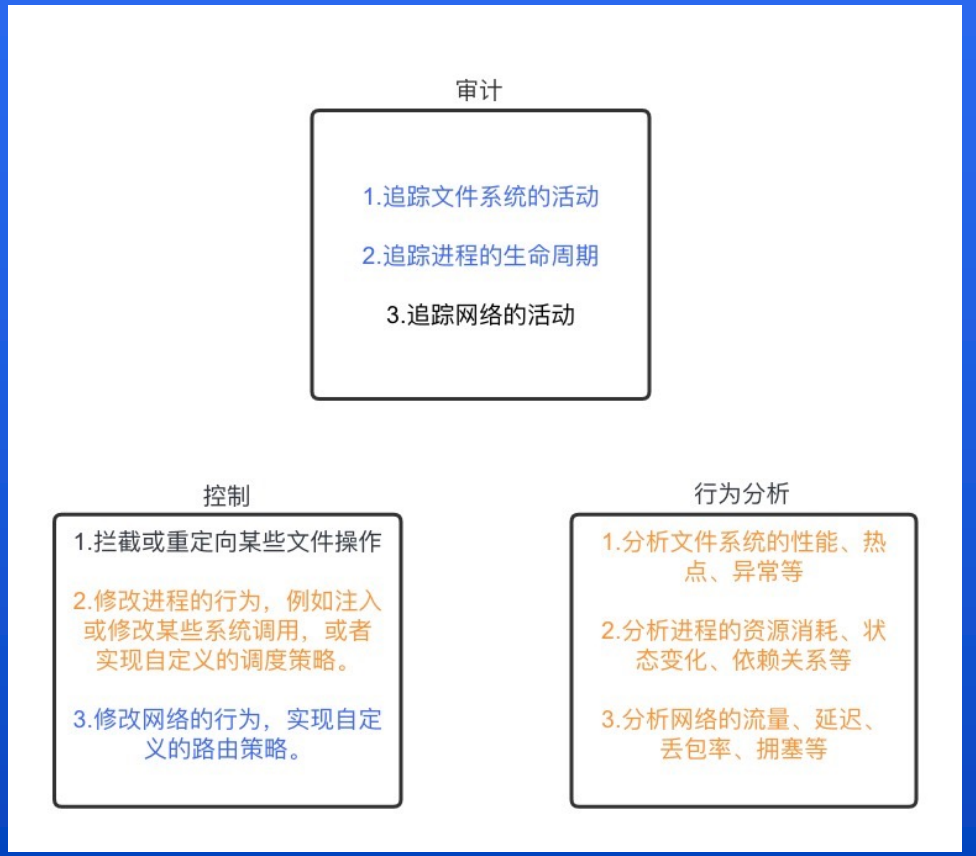
- 追踪文件系统的活动，包括文件的打开、关闭、读写、删除等。
- 修改文件系统的行为，例如拦截某些文件操作，或者实现自定义的安全策略。

进程:

- 追踪进程的生命周期，例如进程的创建、终止、调度、上下文切换等。
- 修改进程的行为，例如注入或修改某些系统调用，或者实现自定义的调度策略。

网络:

- 追踪网络的活动，例如网络包的发送、接收、转发、丢弃等。
- 修改网络的行为，例如过滤或重写某些网络包，或者实现自定义的路由策略。



safeguard项目

特性

- 审计：记录配置文件范围内的行为，并输出日志
- 控制：针对文件，进程，网络的安全访问控制
- 行为分析：收集信息，进行资源，热点，异常等分析
- 主机管理：从安全角度自动化构建细粒度资产信息
- 风险管理：精准发现内部风险，快速定位问题并有效解决安全风险
- 入侵检测：提供多锚点的检测能力，能够实时、准确的感知入侵事件，发现失陷主机，并提供对入侵事件的响应手段。

THANKS

THANKS

THANKS