哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

面向云原生的内核威胁检测系统设计与实现

答辩人：丁浩卓　　指导老师：刘川意

哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

面向云原生的内核威胁检测系统设计与实现

01 背景与意义

02 国内外研究现状

03 主要研究内容

04 研究方案

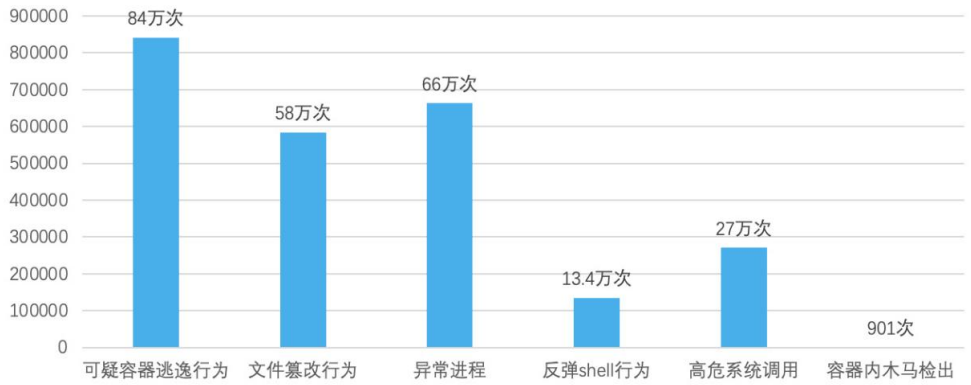
05 目标及进度安排

06 预计困难及方案

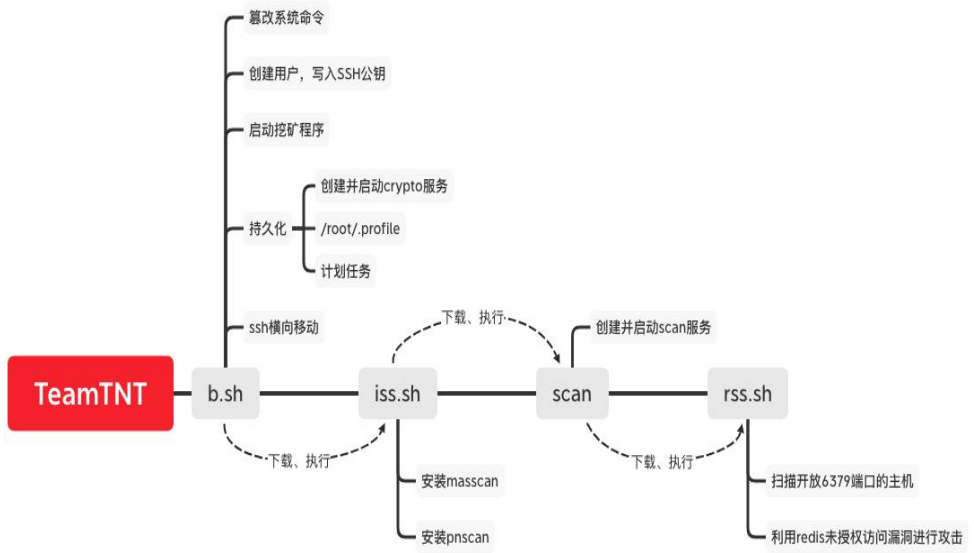
选题背景

云原生虽然具备着更大的灵活性、业务敏捷性和强扩展性，但同时也遭遇了巨大的安全风险。针对云原生的在野攻击数量巨大，容器成为重要的攻击环境。

2021年，腾讯云容器安全服务监测到的可疑容器逃逸行为84万次。



2022年3月初，思科Talos报告了TeamTNT的一个针对网络服务上Docker API攻击活动。



参考来源:
[1] 腾讯云计算（北京）有限公司. 腾讯云容器安全白皮书. 2022.
[2] https://s.tencent.com/research/report/1185.html

选题意义

提升监控容器行为的有效性

监控容器生命周期各阶段行为。在Docker容器启动、运行的各个阶段的对容器行为进行监控，可识别容器异常启动和容器运行时异常。

提升Rootkit病毒检测的有效性

研究Rootkit病毒行为特征。静态源码分析与运行时动态分析相结合。深入源码静态分析Rootkit病毒实现原理，使用bpftrace等跟踪工具在内核函数调用层面对病毒进行运行时动态分析。

提高入侵检测系统的性能

提高容器监控与Rootkit检测效率。以Rootkit检测为例，传统的基于日志系统的入侵检测系统OSSEC检测Rootkit时，需要定时对日志及文件系统进行扫描并进行规则匹配。匹配过程需要占用较多系统资源。基于事件触发的eBPF实现的Rootkit入侵检测更为高效。

国内外研究现状

Rootkit研究

- 2014年，S. A. Musavi等人通过一组特征来量化内核驱动程序中的恶意行为以检测恶意驱动程序。
- 2020年，卢臻针对具有网络连接特征的Rootkit，提出了一种网络流量差异分析法。
- 2020年，Wonjun Lee等人对云原生新环境下的Rootkit攻击场景给出基于机器学习和神经网络模型的检测方案。
- 2022年，文伟平,陈夏润,杨法偿提出从Rootkit启动机制和内存驻留机制进行研究分析提炼恶意代码的检测特征

容器安全

- 2015年，Bui T.基于Docker的内部安全和Docker与Linux内核的安全特性（如SELinux和AppArmor）交互，提出一系列Docker提高安全性的方法。
- 2017年，Jian Z等人针对Docker-escape攻击的方法和特点，提出了一种基于名称空间状态检测的防御方法。
- 2022年，胥柯、张新有、栗晓晗提出CFMAC(containers based on fuzzy mandatory access control)模型对Docker容器进行安全加固

代表性入侵检测技术

- 基于Linux审计框架
- 基于fanotify监听文件系统
- 基于Netlink通信的进程监控
- 周期性检查文件哈希或其他指纹
- 基于Perf与内核挂载点

当前技术存在的局限性

- 缺乏告警的上下文信息
- 告警噪声比大
- 性能影响大

主要研究内容

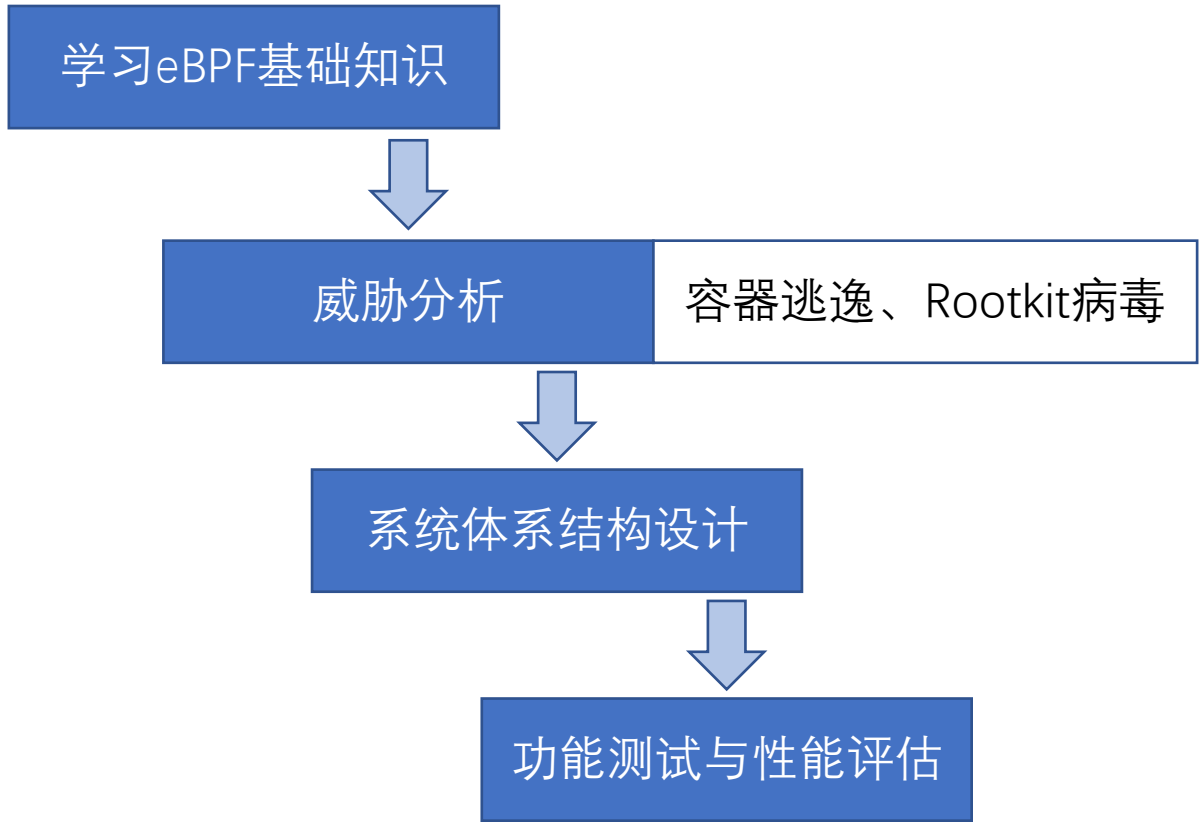
威胁分析

容器逃逸	攻击者在通过入侵攻击获得了容器内某种权限下的命令执行能力前提下，进一步获得通过该容器直接在宿主机上执行命令。
Rootkit攻击	Rootkit是一种特殊的恶意软件，它的功能是在安装目标上隐藏自身及指定的文件、进程和网络链接等信息。

检测技术

容器进程区分	判断进程是否运行在容器中
内核态信息提取	使用eBPF在收集系统运行时信息并将数据共享到其他模块
入侵检测规则匹配	实时处理内核态信息提取模块传入的数据，匹配攻击规则

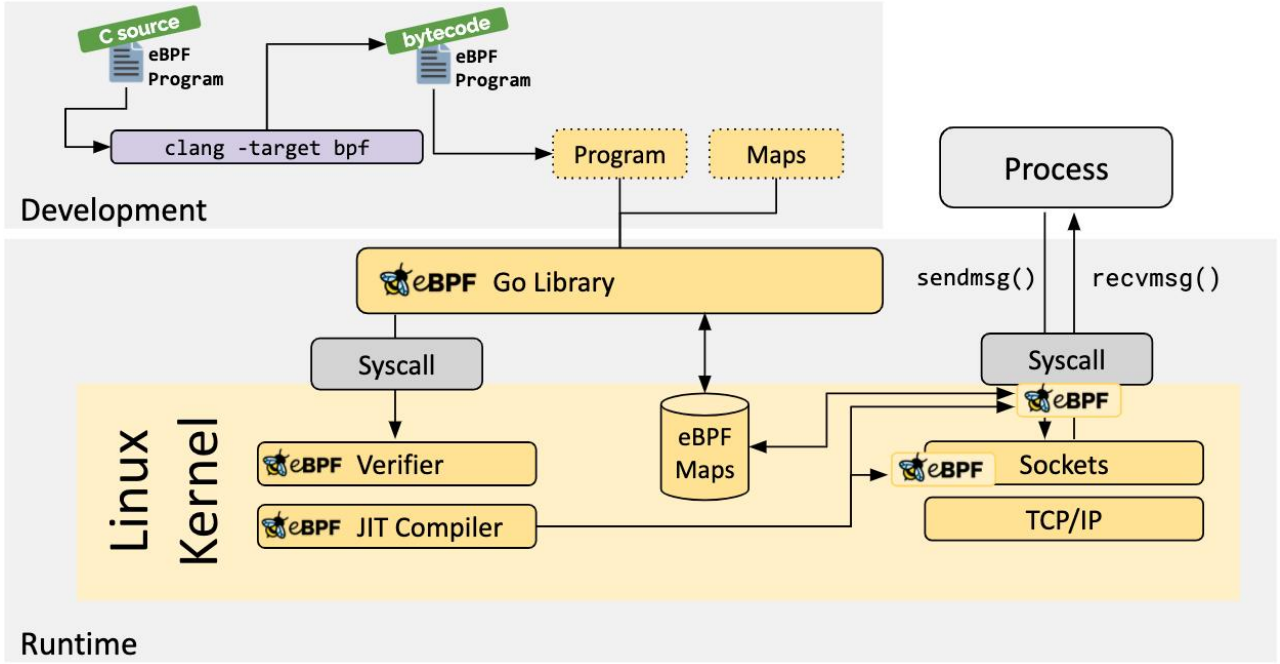
研究方案



研究方案

学习eBPF基础知识

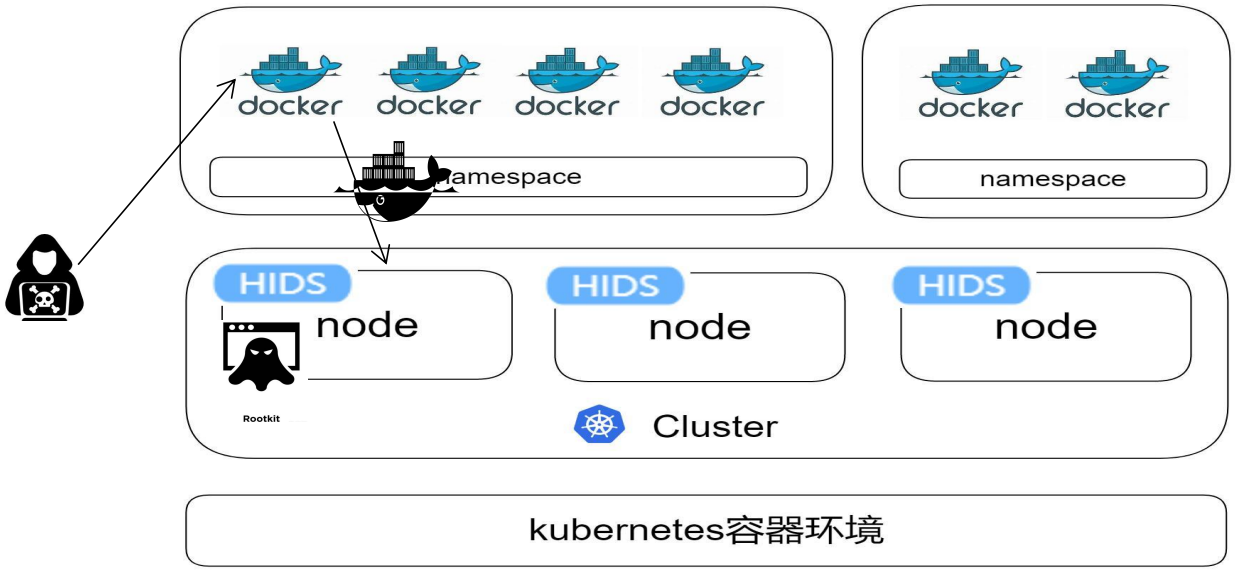
- 学习eBPF原理和基础知识
  - 阅读论文、博客
  - 参考文档学习使用bpftrace、bpftool等工具
- 学习libbpf库的使用
- libbpf编程实践



参考来源：https://ebpf.io/zh-cn/

研究方案

威胁分析



容器逃逸

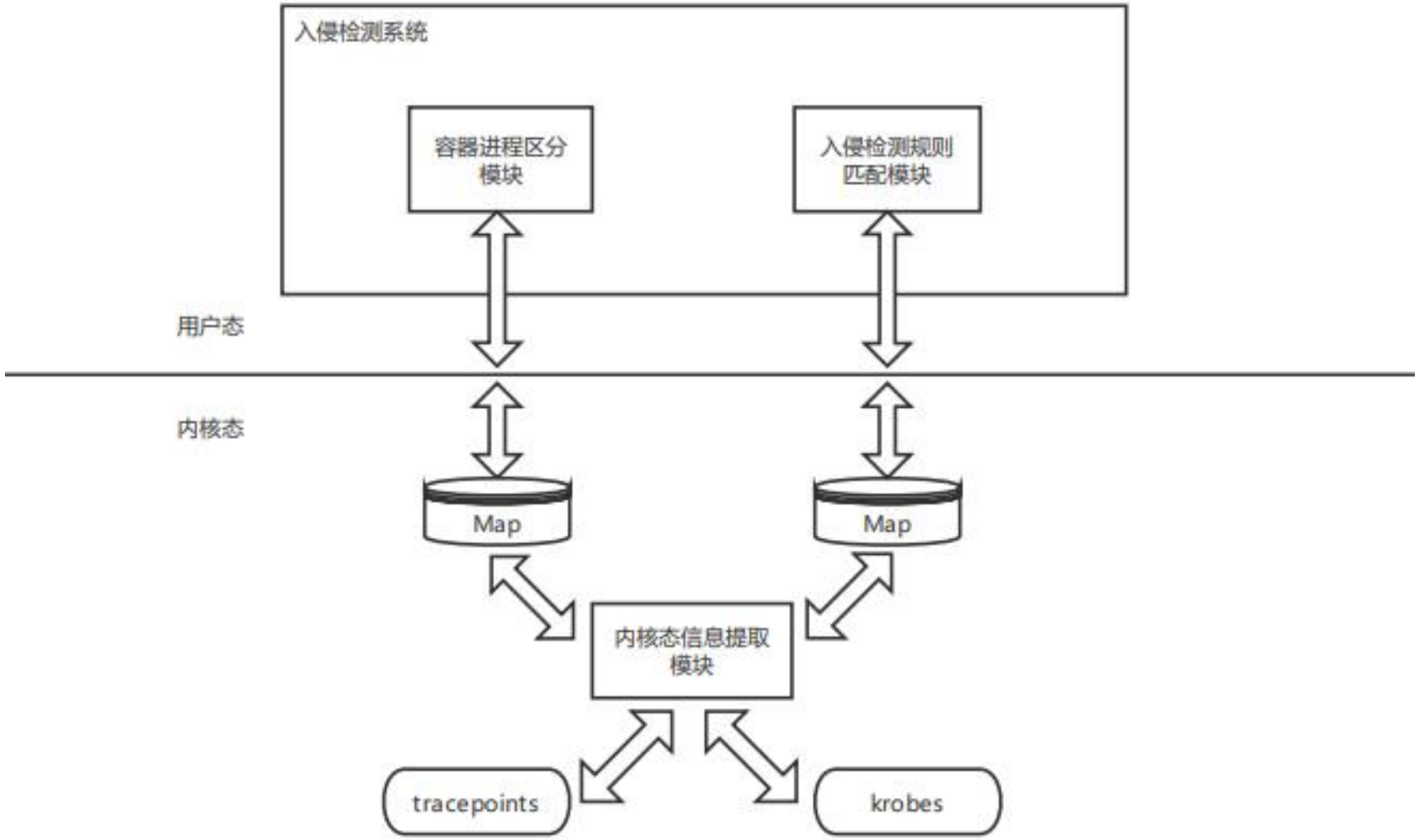
- 阅读Docker官方文档中容器安全相关部分
- Docker容器逃逸复现
- 使用trace工具在内核函数调用层面对容器逃逸行为进行观测分析

Rootkit病毒

- 收集开源的Rootkit病毒样本
- 攻击复现
- 深入源码对Rootkit病毒进行静态分析
- 使用trace工具在内核函数调用层面对Rootkit病毒进行运行时动态分析

研究方案

系统体系结构设计



研究方案

功能测试与性能评估

- **功能测试**
  - 复现3到4种入侵攻击
  - 将系统检测结果和预期结果进行比较，判断系统是否可以完成入侵检测的任务

- **性能评估**，运行常见的性能测试程序，如：Unix Bench等
  - CPU和内存占用率
  - 时延测试：进程上下文切换、文件操作、进程创建、信号处理以及内存访问等常见操作和各类系统调用的时间开销
  - 带宽测试：缓存文件读取、内存拷贝、内存读写、管道通讯、TCP连接等

目标及进度安排

目标

设计**基于eBPF的入侵检测系统**，使用eBPF技术在内核收集运行时的信息，基于特征规则库实现对内核威胁的检测。

- 性能占用率不超过**5%**
- 检测误报率不超过**10%**

进度安排

2022年10月至11月：完成技术背景调研

2022年11月至12月：完成威胁建模分析，提取攻击特征

2023年1月至3月：完成对系统设计实现，实现对内核威胁的检测

2022年4月至5月：完善系统实现，完成性能测试

预计困难及方案

预计困难 1：Rootkit自身极强的隐蔽性

Rootkit是一种特殊的恶意软件，其功能是在安装目标上隐藏自身及指定的文件、进程和网络链接等信息。Rootkit常与木马、后门等其他恶意程序结合使用，具有极强的隐蔽性。

解决方案

- 阅读开源Rootkit源码，深入代码了解实现原理。
- 通过trace技术，在内核函数层面和系统调用层面动态分析Rootkit运行时函数调用特征。

预计困难 2：判断进程是否运行在容器中

若直接监听所有的进程的行为，会在信息共享时产生较大的内存占用，将对系统整体造成较大的性能影响，所以需要针对性地只对运行在容器中的进程进行信息收集。但容器隔离机制复杂，在内核态监控时较难准确判断当前进程是否运行在容器中。

解决方案

- 深入学习容器实现原理与隔离机制
- 了解容器技术的基石
- 学习Linux内核级别环境隔离机制Namespace

请各位老师指正