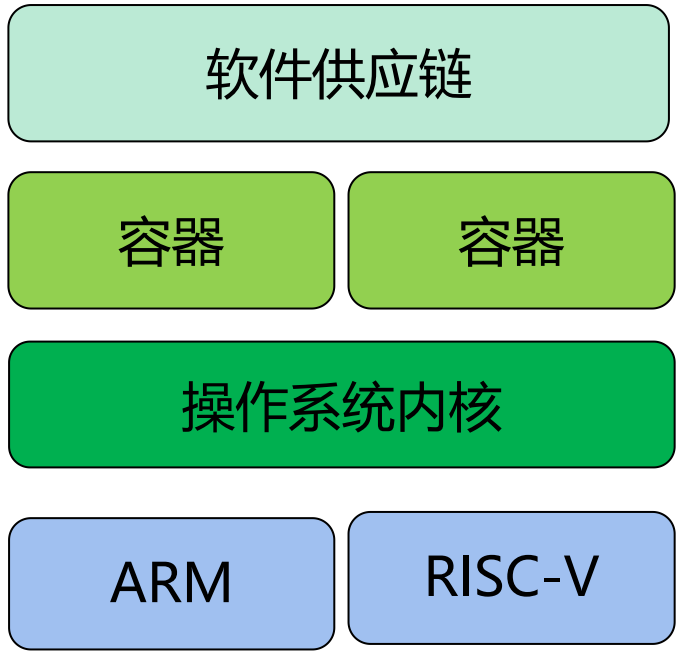
云原生系统新型攻击与防护

汇报人：申文博

个人介绍

- 浙江大学百人计划研究员、博士生导师
- 计算机科学与工程系副主任，移动终端安全-省工程中心副主任
- 研究方向：操作系统安全，容器安全，软件供应链安全
  - 发表论文40余篇，覆盖全部计算机安全四大国际会议
  - 获得3项杰出论文奖(NDSS 16, AsiaCCS 17, ACSAC 22)
  - 主持国家自然科学基金、科技部重点研发课题等多项余项科研项目
- 曾担KNOX内核安全Tech Lead, 三星美国研究院（硅谷）(2015-2019)
  - 负责Linux内核安全Real-time Kernel Protection (RKP)
  - 多项研究成果部署超亿部设备
- 美国北卡州立大学计算机博士，2015
- 哈工大，软件工程本科，2010



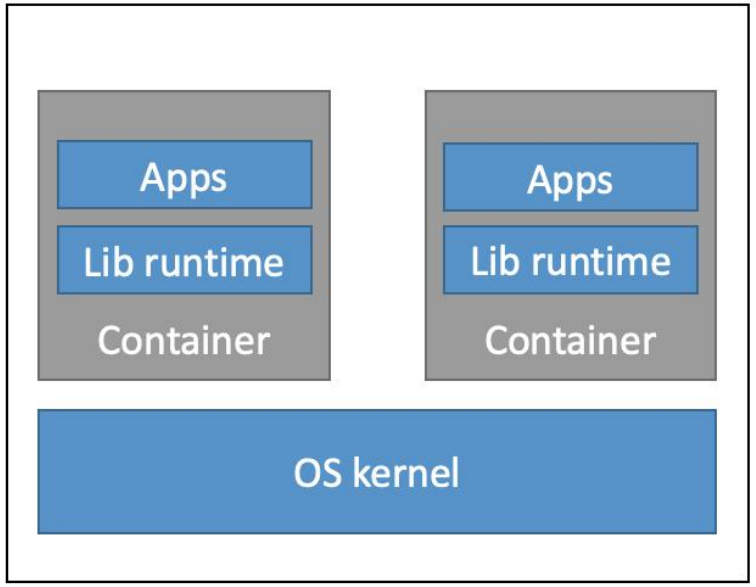
目录

CONTENTS

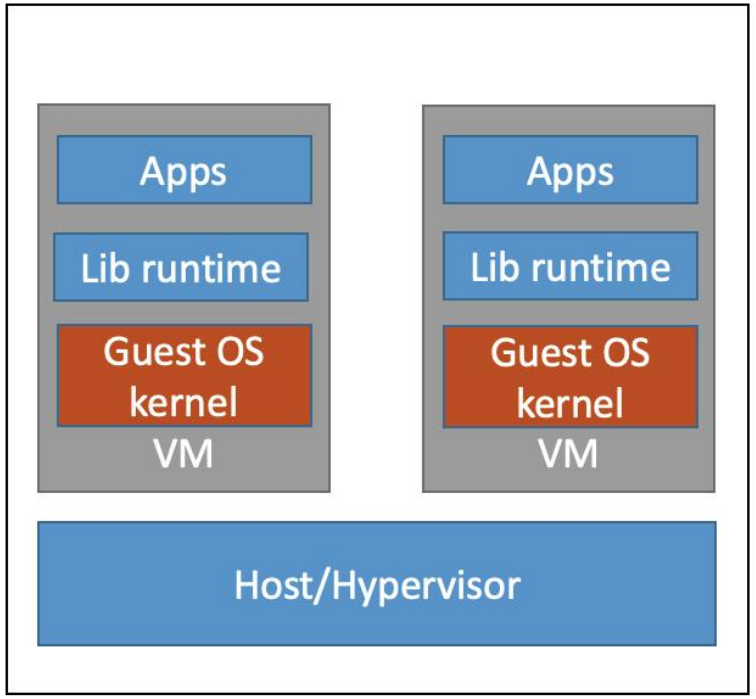
1 云原生架构介绍

容器技术

- 支撑云原生系统的关键技术
  - 与宿主机共享相同的内核，效率高、启动快



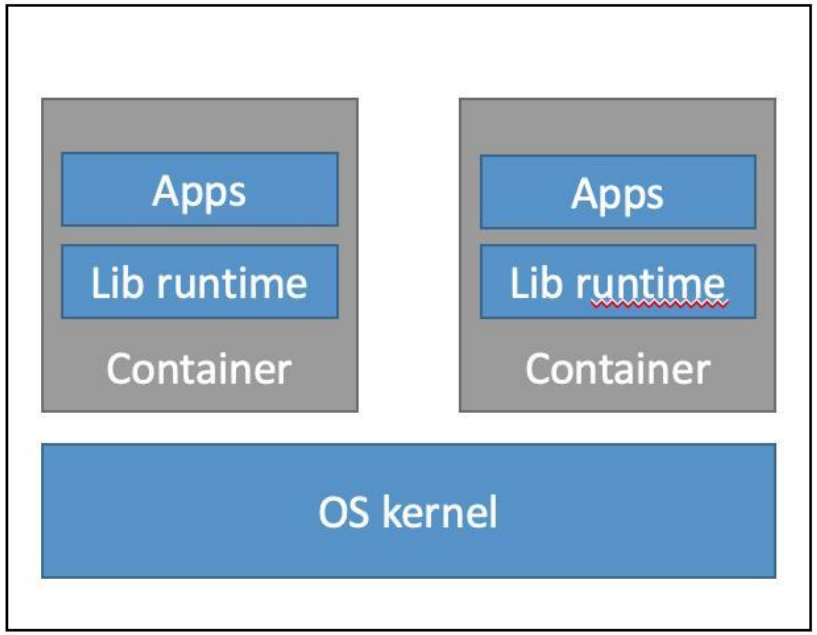
操作系统级虚拟化
（容器）

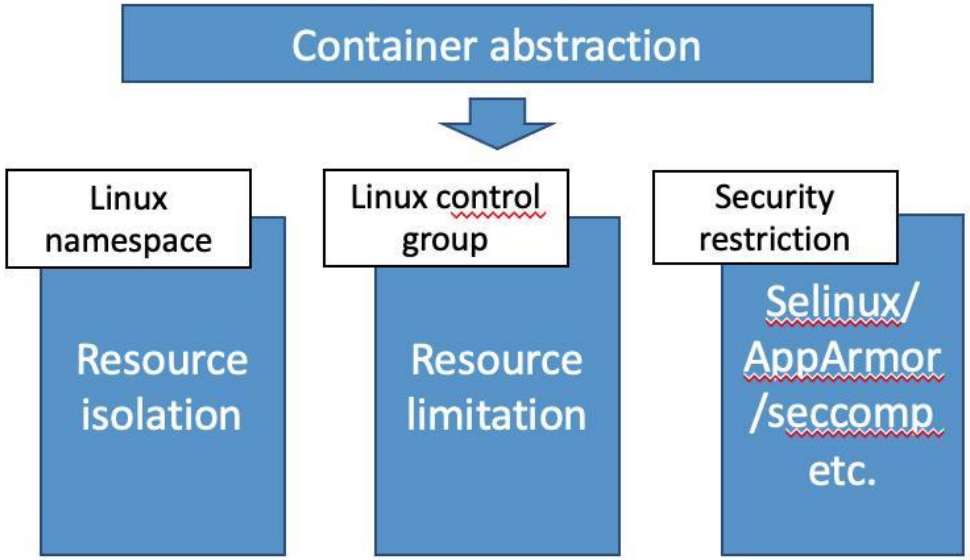


硬件虚拟化
（虚拟机）

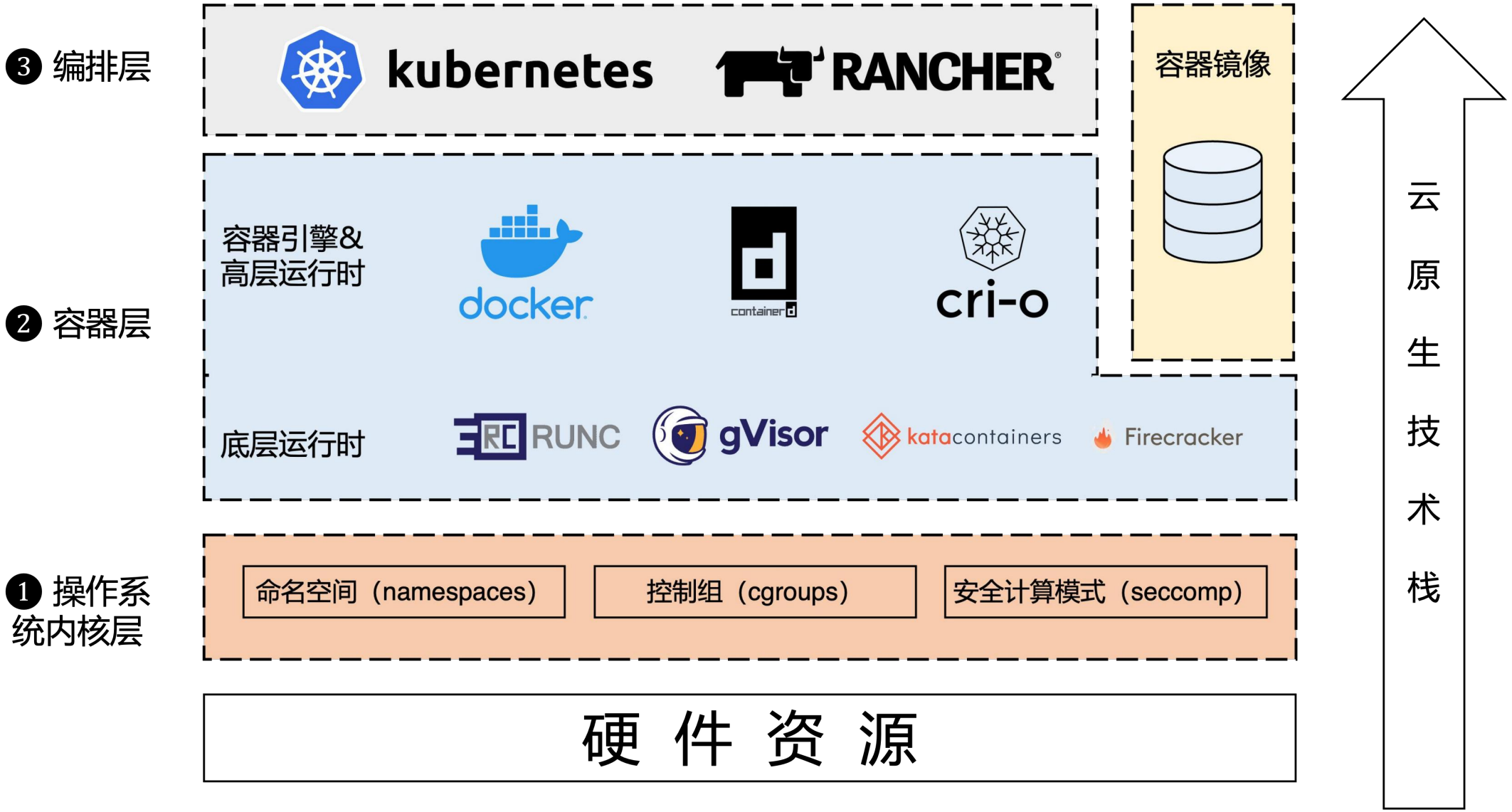
容器技术

- 支撑云原生系统的关键技术
  - 利用宿主机内核提供的机制来进行资源隔离和资源限制

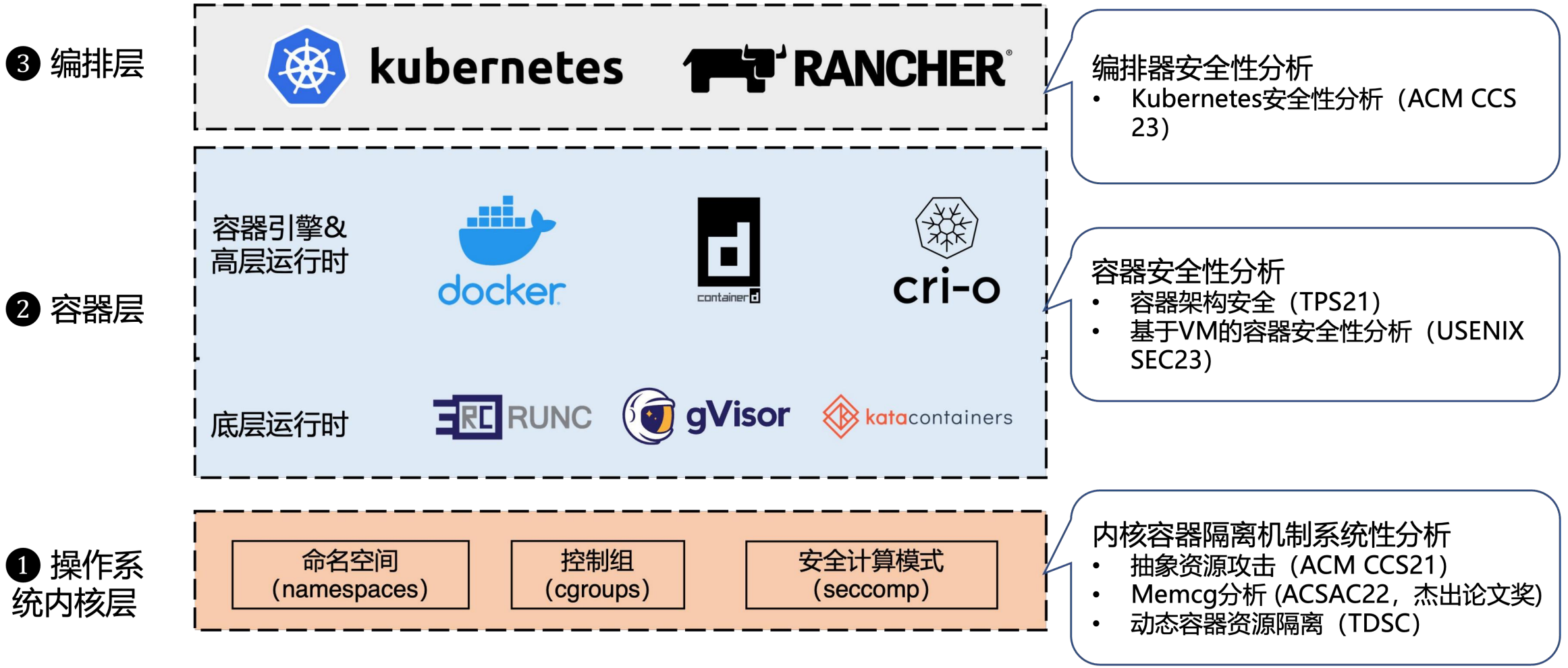




云原生技术栈总览


❸ 编排层
kubernetes
RANCHER
容器镜像
❷ 容器层
容器引擎&
高层运行时
docker
container
cri-o
底层运行时
RUNC
gVisor
katacontainers
Firecracker
❶ 操作系
统内核层
命名空间（namespaces）
控制组（cgroups）
安全计算模式（seccomp）
硬 件 资 源
云 原 生 技 术 栈

云原生研究总览


❸ 编排层
kubernetes
RANCHER
编排器安全性分析
• Kubernetes安全性分析（ACM CCS 23）
❷ 容器层
容器引擎&
高层运行时
docker
container
cri-o
容器安全性分析
• 容器架构安全（TPS21）
• 基于VM的容器安全性分析（USENIX SEC23）
底层运行时
RUNC
gVisor
katacontainers
❶ 操作系统内核层
命名空间
(namespaces)
控制组
(cgroups)
安全计算模式
(seccomp)
内核容器隔离机制系统性分析
• 抽象资源攻击（ACM CCS21）
• Memcg分析 (ACSAC22，杰出论文奖)
• 动态容器资源隔离（TDSC）

云原生研究总览

❸ 编排层

kubernetes RANCHER

编排器安全性分析
- Kubernetes安全性分析（ACM CCS 23）

❷ 容器层

容器引擎&
高层运行时

docker container cri-o

容器安全性分析
- 容器架构安全（TPS21）
- 基于VM的容器安全性分析（USENIX SEC23）

底层运行时

RUNC gVisor katacontainers

❶ 操作系统内核层

命名空间（namespaces） 控制组（cgroups） 安全计算模式（seccomp）

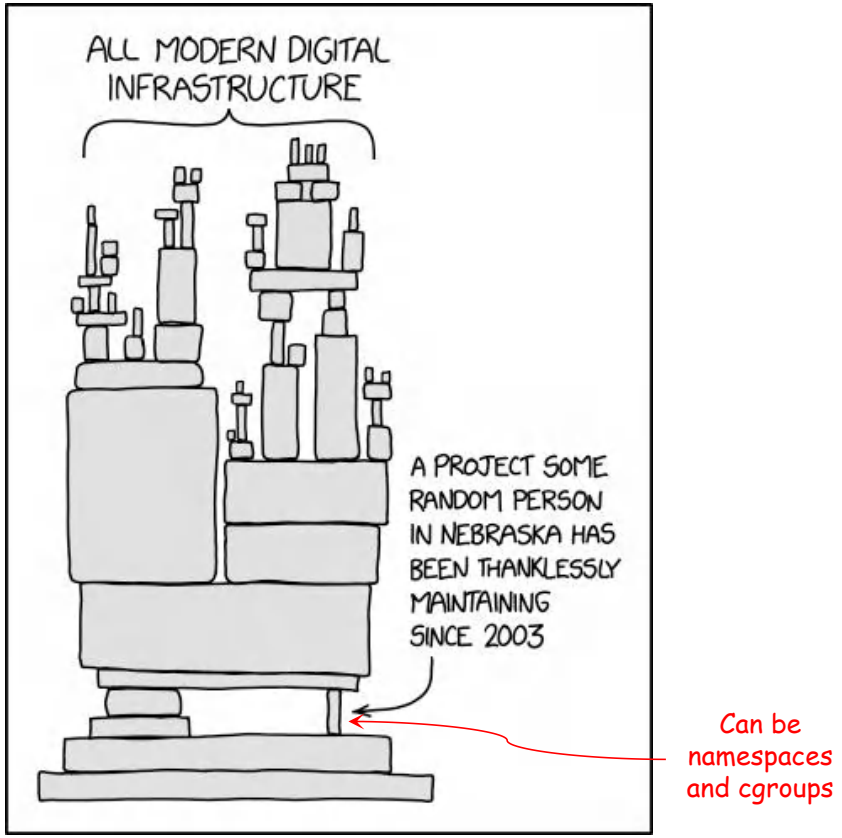
内核容器隔离机制系统性分析
- 抽象资源攻击（ACM CCS21）
- Memcg分析 (ACSAC22，杰出论文奖)
- 动态容器资源隔离（TDSC）

2 新型系统攻防

研究动机

- 云原生社区依赖Linux内核机制构建了大量的容器方案
  - Namespace, control groups，Docker，LXC，runC等
- 关键组件如Kubernetes占有率接近100%
  - Kubernetes has market share of 98.5%
- 这些内核机制/关键组件没有经过系统性的安全分析
- 我们的工作

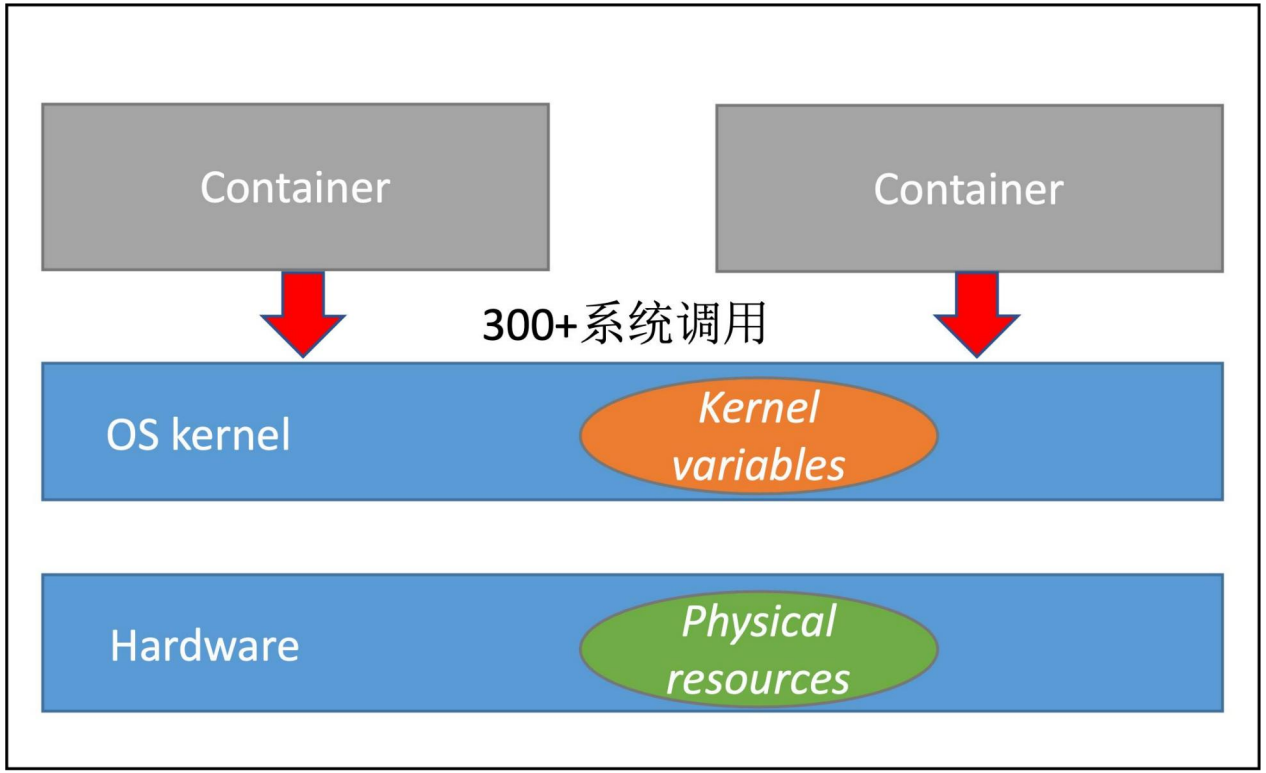
Namepace隔离做的怎么样?	资源隔离分析：抽象资源攻防 (ACM CCS 21)
Cgroups限制做的怎么样?	资源计数分析：Memcg分析（ACSAC 22)
Micro-VM隔离性怎么样?	Micro-VM容器安全分析：操作转发攻防(USENIX SEC 23)
Kubernetes安全性怎么样?	Kubernetes安全分析：过度权限攻防（ACM CCS 23)



容器资源隔离与限制

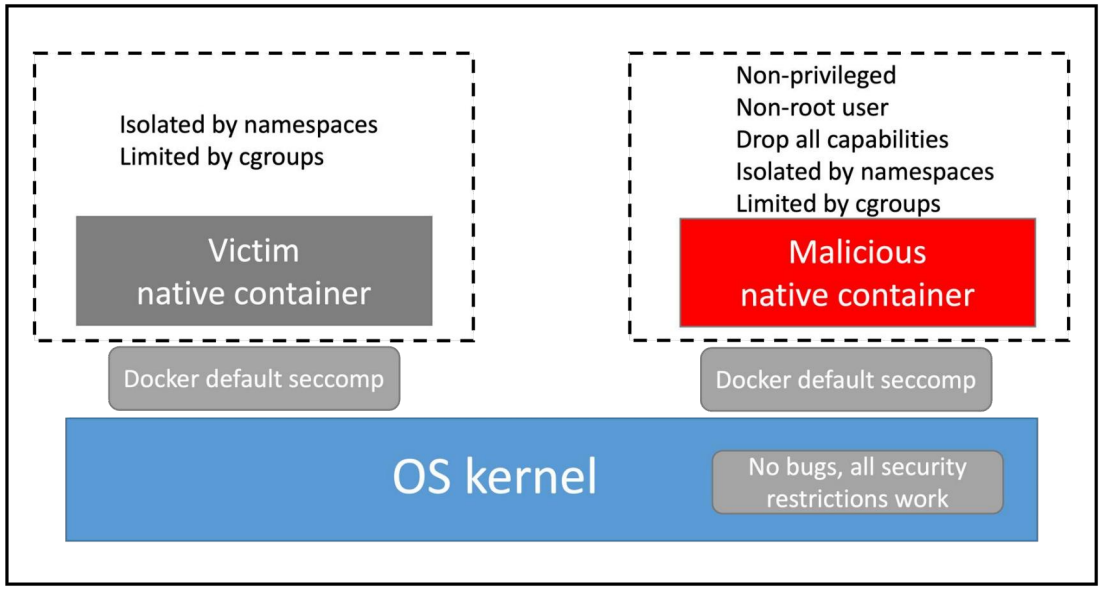
- 由namespaces负责隔离
  - 当前内核支持8种namespaces
  - 包含UTS、IPC、mount、PID、network、user、time、cgroup
- 由control groups进行限制
  - 当前内核支持13种cgroups
  - 主要用于限制CPU、内存和设备资源
- 容器运行是需要共享宿主机物理资源，通过300+系统调用请求内核服务
- 内核使用大量的内核变量来实现这些服务
  - 这些变量包含结构体以及普通变量等

这些内核变量同样在容器间共享，可被恶意容器耗尽，导致DoS攻击

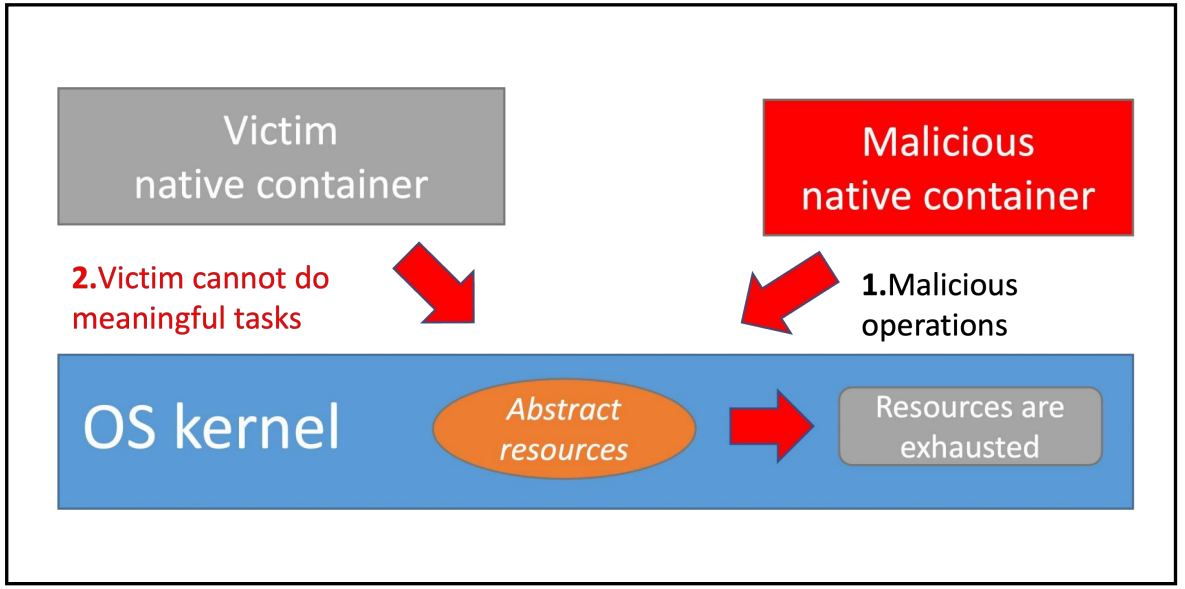


抽象资源攻击

- 抽象资源定义
  - 与CPU，RAM等物理资源相对应
  - 我们把这些内核变量称为抽象资源
- 威胁模型
  - 攻击者无任何权限
  - 系统无任何漏洞



- 攻击的基本思路
  - 内核变量和数据结构（抽象资源）是被所有容器所共享
  - 攻击者可以耗尽抽象资源
  - 对正常容器发起DoS攻击



抽象资源攻击范围

- 利用抽象资源，可攻击操作系统提供的各个主要功能

OS service	Resource name	Attacking results
Process	PID idr	Can not create new process
Memory	dirty ratio	IO performance down for over 90%
Storage	Inode, nr_files	Operations requiring open-new-file fail
IO	pty_count, , netns_ct->count	New SSH connections are rejected
Security and protection	/dev/random	/dev/random reading is suspended

抽象资源攻击范围

- 利用抽象资源，可攻击主流操作系统
  - FreeBSD
    - 通过写脏页消耗dp_dirty_total变量
    - 当dp_dirty_total变量达到上限后，被攻击jail发起的写请求会被延迟
  - Fuchsia
    - 攻击kMaxHandleCount，造成系统crash

Operation system	Abstract resources	Attacking results
FreeBSD	dsl_pool->dp_dirty_total	IO performance down for 46%
	vnode	Reach the maxvnodes limit
Fuchsia	kMaxHandleCount	Whole system crash

抽象资源攻击是在容器技术中特有且通用的攻击!

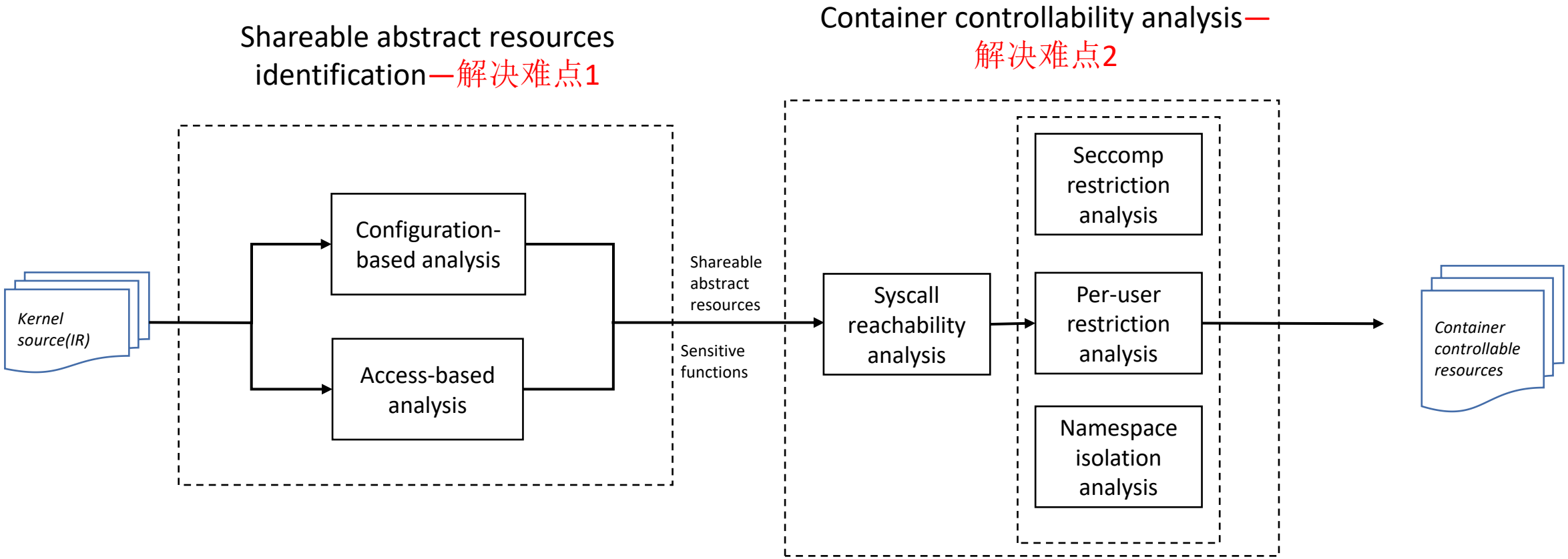
抽象资源攻击范围

- 利用抽象资源，可攻击云平台共享内核容器环境
- 伦理问题考虑
  - 实验时不能影响其他租户，购买单独的物理机
- 我们选取7种抽象资源进行实验，攻击云平台共享内核容器环境
  - 覆盖操作系统所有主要功能

Abstract resources	Local	V1	V2	V3	V4	Attacking results
PID idr	Y	Y	Y	Y	-	Fork fail, victim container is evicted
dirty ratio	Y	Y	Y	-	Y	IO performance down for over 90%
inode	Y	-	-	-	Y	Victim container gets evicted
nr_files	Y	Y	Y	Y	Y	Operations requiring open-new-file fail
pty_count	Y	Y	Y	Y	Y	New SSH connections are rejected
netns_ct->count	Y	Y	Y	Y	Y	Random packets dropping
random entropy	Y	-	Y	-	Y	/dev/random read blocked

抽象资源攻击防御

- 通过静态分析，识别出Linux内核中所有可被攻击的抽象资源
- 难点1：识别出有意义的抽象资源
- 难点2：识别容器是否能耗尽这些抽象资源



静态分析结果

- 资源过滤
  - 筛选掉没有进行量化修改的资源，识别出了1010个抽象资源
- 动态验证：501种资源可被重复触发
  - 700个与驱动无关的资源，其中有389 个资源可被重复触发
  - 310个与驱动相关的资源，我们实验设备的硬件支持其中的218个资源，其中112个资源能被重复触发

Res. Dir.		No.	Repeatedly	True positive
Non-driver		700	389	55.6%
Driver	Have HW	218	112	51.4%
	No HW	92	-	-

Takeaway

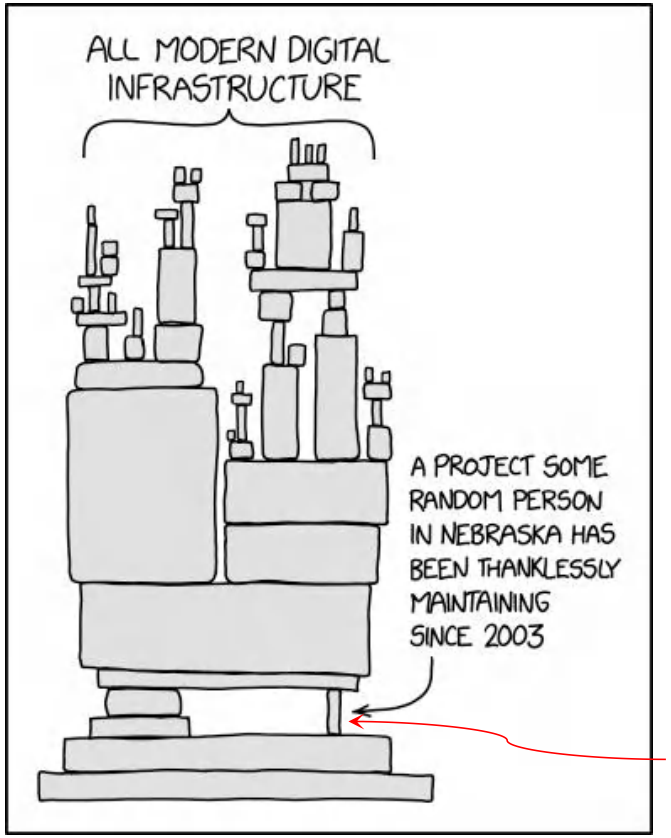
- 新的攻击面
  - 我们揭示了有一个操作系统级虚拟化特有且通用的攻击面
  - 可攻击Linux操作系统各种主要功能
  - 可攻击所有主流操作系统，包含FreeBSD，Fuchsia
  - 可攻击安全容器gVisor
- 攻击实用性评估
  - 在四大云厂商提供的self-deployed native container环境都容易受到抽象资源攻击
- 系统化分析
  - 我们设计实现了一个静态分析工具
  - 我们发现了501个在Linux内核中能被动态且重复触发的抽象资源

Demons in the Shared Kernel: Abstract Resource Attacks Against OS-level Virtualization, ACM CCS'21

研究动机

- 云原生社区依赖Linux内核机制构建了大量的容器方案
  - Namespace, control groups，Docker，LXC，runC等
- 关键组件如Kubernetes占有率接近100%
  - Kubernetes has market share of 98.5%
- 这些内核机制/关键组件没有经过系统性的安全分析
- 我们的工作

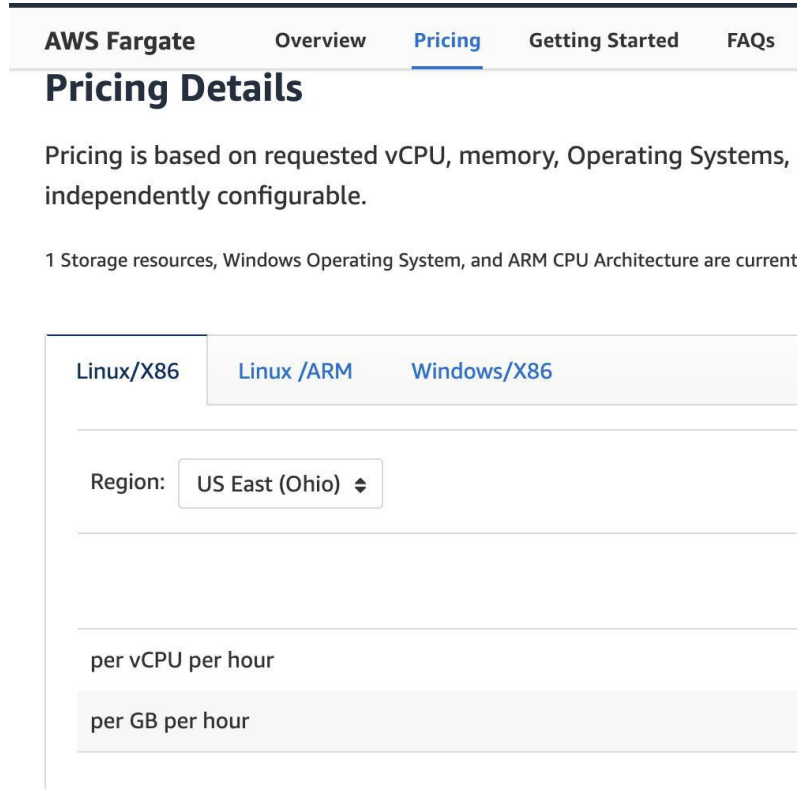
Namepace隔离做的怎么样?	资源隔离分析：抽象资源攻防 (ACM CCS 21)
Cgroups限制做的怎么样?	资源计数分析：Memcg分析（ACSAC 22）
Micro-VM隔离性怎么样?	Micro-VM容器安全分析：操作转发攻防(USENIX SEC 23)
Kubernetes安全性怎么样?	Kubernetes安全分析：过度权限攻防（ACM CCS 23）



研究动机

- 内存是一项最基本的计算资源
  - 对内存使用的计数和限制是容器最核心的需求之一
- Linux提出memory control groups（memcg）
  - 为上层容器实现提供内存计数功能
- Memcg未经安全性分析

- 我们的问题：
  - Memcg内存计数是否存在计数缺失？是否存在安全影响？
    - DoS，Use mem for free
  - 如何在内核代码中识别计数缺失？



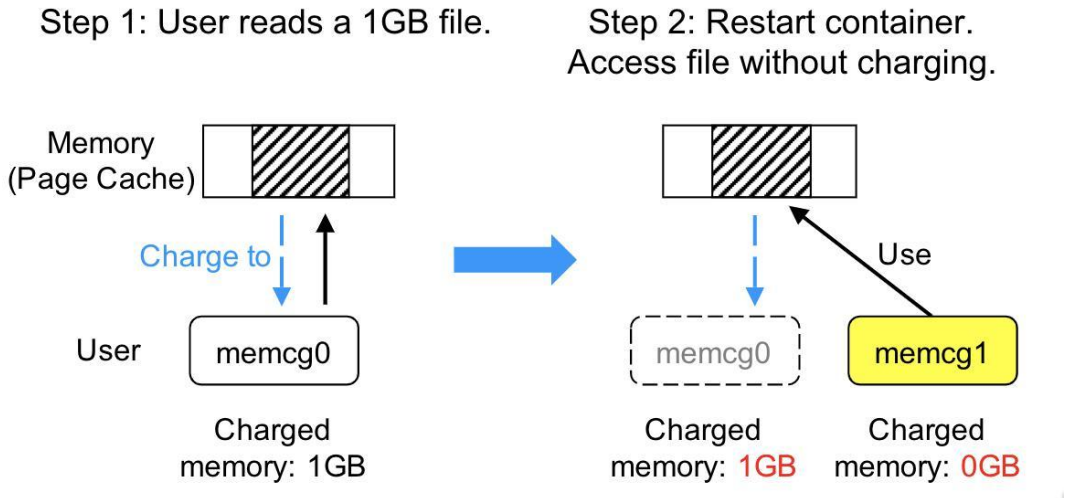
内存计数缺失案例

- 发现了policy design中导致的计数缺失的问题
  - Page cache内存计数仅在分配frame时增加，在free frame时减少
  - First user pays，other users use for free
  - 在first user退出时，计数不会重新分配
  - 导致攻击者可以轻易突破memcg内存限制
- Docker对page cache内存计数描述错误
  - We reported, Docker didn't fix
  - https://docs.docker.com/config/containers/runmetrics/

concepts / Container runtime / Collect runtime metrics

Accounting for memory in the page cache is very complex. If two processes in different control groups both read the same file (ultimately relying on the same blocks on disk), the corresponding memory charge is split between the control groups. It's nice, but it also means that when a cgroup is terminated, it could increase the memory usage of another cgroup, because they are not splitting the cost anymore for those memory pages.

1. 攻击者创建一个容器，加载目标文件，内存被计数
2. 攻击者重启容器，内核会创建一个新的memcg，内存计数为零，达到use memory for free，可轻松突破memcg内存使用限制

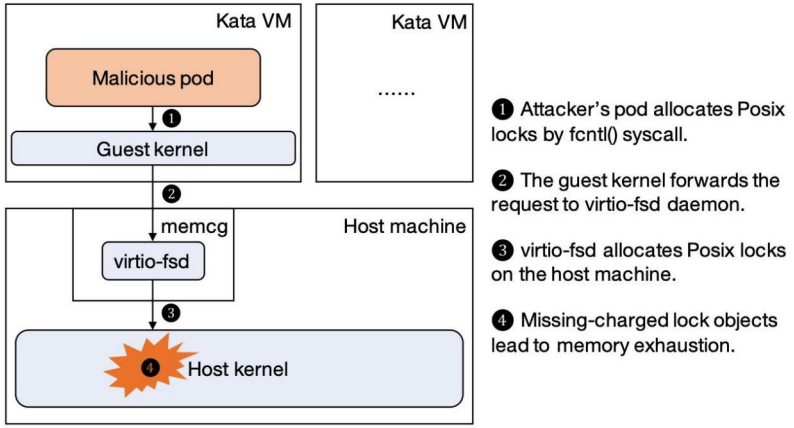


内存计数缺失案例

- 通过内存计数缺失攻击Docker平台
- sem_alloc存在计数缺失
  - 非特权容器用户可进行容器逃逸
  - 轻易打破内存限制，消耗所有系统内存，导致系统OOM
  - 拿到一个新的CVE-2021-3759

- 通过内存计数缺失攻击Kata container
- POSIX lock 存在计数缺失
  - 非特权容器用户可进行虚拟机逃逸
  - 轻易打破内存限制，消耗所有系统内存，导致系统OOM
  - 拿到一个新的CVE-2022-0480

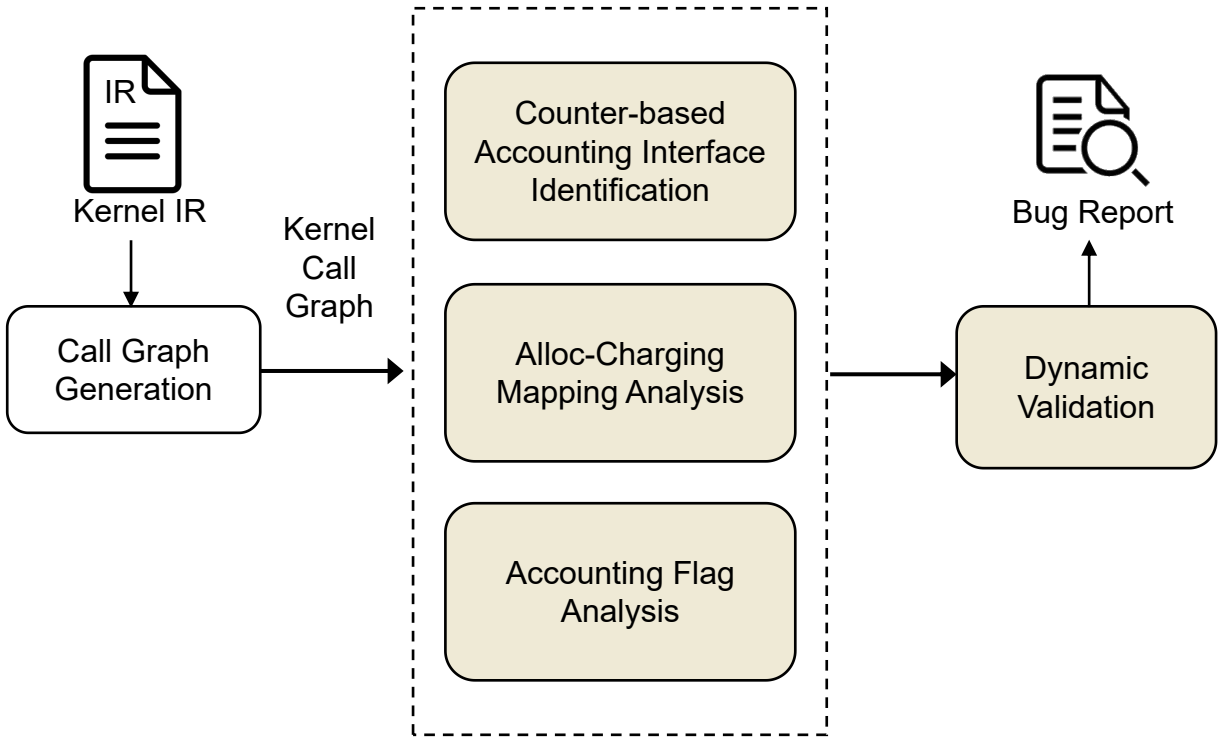
```c
static struct sem_array *sem_alloc(size_t nsems)
{
    struct sem_array *sma;

    if (nsems > (INT_MAX - sizeof(*sma)) / sizeof(sma->sems[0]))
        return NULL;

    sma = kvzalloc(struct_size(sma, sems, nsems), GFP_KERNEL);
    if (unlikely(!sma))
        return NULL;

    return sma;
}
```



内存计数缺失可导致容器逃逸到host，突破内存限制，DoS整个系统

内存计数缺失静态分析

- 技术挑战
  - memg接口缺乏明确定义
  - 内核内存分配路径复杂，难以映射分配-计数
- 提出两项技术
  - Counter-based Accounting Interface Identification
  - Alloc-Charging Mapping Analysis



分析结果验证

- 分析得出242个计数缺失bug
- 进一步验证是否从user space可触发
- 对内核进行instrumentation
- 在用户空间运行test cases
  - LTP test cases
  - Manually developed test cases
- 分析结果
  - 162个可触发，可打破memcg内存限制
  - Report 53个，37个已经修复

Alerts	Triggerable	Exploitable &Reported	Confirmed
242	162	53	37

No.	Source location	Function name	Allocation interface	Triggered by	Confirm Status	Patch Status
1	arch/x86/kernel/ldt.c:157	alloc_ldt_struct	kmalloc	clone	Confirmed	Pending
2	fs/eventfd.c:417	do_eventfd	kmalloc	eventfd	Pending	-
3	fs/eventpoll.c:1014	ep_alloc	kzalloc	epoll_create	Pending	-
4	fs/fcntl.c:899	fasync_alloc	kmem_cache_alloc	fcntl	Confirmed	Pending
5	fs/fs_context.c:234	alloc_fs_context	kzalloc	fsopen	Confirmed	Merged
6	fs/fs_context.c:634	legacy_init_fs_context	kzalloc	fsopen	Confirmed	Merged
7	fs/fsopen.c:100	fscontext_alloc_log	kzalloc	fsopen	Pending	-
8	fs/io-wq.c:1089	io_wq_create	kzalloc	io_uring_setup	Confirmed	Pending
9	fs/io-wq.c:1093	io_wq_create	kzalloc_node	io_uring_setup	Confirmed	Pending
10	fs/io-wq.c:1114	io_wq_create	kzalloc_node	io_uring_setup	Confirmed	Pending
11	fs/io_uring.c:1180	io_ring_ctx_alloc	kzalloc	io_uring_setup	Confirmed	Pending
12	fs/io_uring.c:1184	io_ring_ctx_alloc	kmem_cache_alloc	io_uring_setup	Confirmed	Pending
13	fs/io_uring.c:1197	io_ring_ctx_alloc	kmalloc	io_uring_setup	Confirmed	Pending
14	fs/io_uring.c:7254	__io_sqe_files_scm	kzalloc	io_uring_register	Pending	-
15	fs/io_uring.c:7507	alloc_fixed_file_ref_node	kzalloc	io_uring_register	Pending	-
16	fs/io_uring.c:7546	io_sqe_files_register	kzalloc	io_uring_register	Pending	-
17	fs/io_uring.c:7853	io_uring_alloc_task_context	kmalloc	io_uring_setup	Confirmed	Pending
18	fs/io_uring.c:8044	io_mem_alloc	__get_free_pages	io_uring_setup	Confirmed	Pending
19	fs/io_uring.c:9541	io_register_personality	kmalloc	io_uring_register	Pending	-
20	fs/locks.c:2241	flock	flock_make_lock	flock	Confirmed	Pending
21	fs/locks.c:258	locks_get_lock_context	kmem_cache_alloc	flock	Confirmed	Pending
22	fs/namespace.c:177	alloc_vfsmnt	kmem_cache_zalloc	mount	Confirmed	Pending
23	fs/namespace.c:3267	alloc_mnt_ns	kzalloc	clone	Confirmed	Pending
24	fs/notify/group.c:121	fsnotify_alloc_group	kzalloc	inotify_init1	Confirmed	Merged
25	fs/notify/inotify/inotify_user.c:630	inotify_new_group	kmalloc	inotify_init1	Confirmed	Merged
26	fs/select.c:658	core_sys_select	kvmalloc	select	Confirmed	Pending
27	fs/signalfd.c:278	do_signalfd4	kmalloc	signalfd4	Pending	-
28	fs/timerfd.c:412	timerfd_create	kzalloc	timerfd_create	Pending	-
29	ipc/namespace.c:45	create_ipc_ns	kzalloc	clone	Confirmed	Pending
30	ipc/sem.c:514	sem_alloc	kvzalloc	semget	Confirmed	Pending
31	ipc/sem.c:1853	get_undo_list	kzalloc	semop	Confirmed	Pending
32	ipc/sem.c:1938	find_alloc_undo	kzalloc	semop	Confirmed	Pending
33	kernel/bpf/core.c:117	bpf_prog_alloc	alloc_percpu_gfp	bpf, prctl	Confirmed	Merged
34	kernel/bpf/core.c:85	bpf_prog_alloc_no_stats	__vmalloc	bpf, prctl	Confirmed	Merged
35	kernel/bpf/core.c:89	bpf_prog_alloc_no_stats	kzalloc	bpf, prctl	Confirmed	Merged
36	kernel/events/core.c:11142	perf_event_alloc	kzalloc	perf_event_open	Pending	-
37	kernel/events/core.c:4443	alloc_perf_context	kzalloc	perf	Pending	-
38	kernel/nsproxy.c:56	create_nsproxy	kmem_cache_alloc	setns	Confirmed	Pending
39	kernel/pid_namespace.c:90	create_pid_namespace	kmem_cache_zalloc	clone	Confirmed	Pending
40	kernel/seccomp.c:1481	init_listener	kzalloc	prctl	Pending	-
41	kernel/seccomp.c:565	seccomp_prepare_filter	kzalloc	prctl	Pending	-
42	kernel/signal.c:435	__sigqueue_alloc	kmem_cache_alloc	rt_sigqueueinfo	Confirmed	Pending
43	kernel/time/namespace.c:91	clone_time_ns	kmalloc	clone	Confirmed	Pending
44	kernel/time/namespace.c:97	clone_time_ns	alloc_page	clone	Confirmed	Pending
45	kernel/time/posix-timers.c:458	alloc_posix_timer	kmem_cache_zalloc	timer_create	Confirmed	Pending
46	kernel/user_namespace.c:105	create_user_ns	kmem_cache_zalloc	clone	Confirmed	Pending
47	mm/hugetlb.c:868	resv_map_alloc	kmalloc	memfd_create	Pending	-
48	mm/hugetlb.c:869	resv_map_alloc	kmalloc	memfd_create	Pending	-
49	mm/slab_common.c:245	create_cache	kmem_cache_zalloc	clone	Confirmed	Pending
50	net/core/net_namespace.c:423	net_alloc	kzalloc	clone	Pending	-
51	security/keys/key.c:277	key_alloc	kmem_cache_alloc	add_key	Pending	-
52	security/keys/key.c:282	key_alloc	kmemdup	add_key	Pending	-
53	security/keys/key.c:81	key_user_lookup	kzalloc	add_key	Pending	-

Takeaway

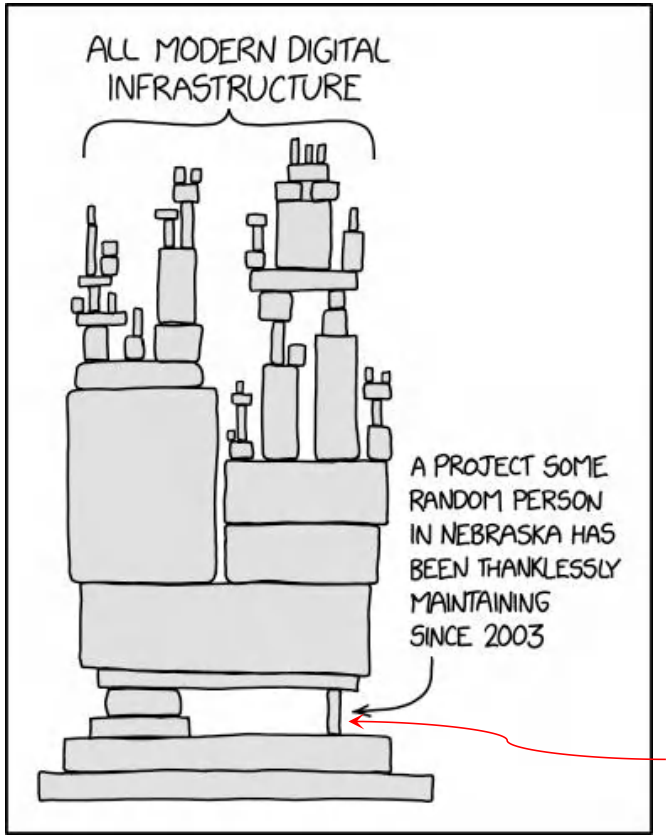
- 内核memcg policy 设计和实现均存在计数缺失
- 识别出计数缺失问题，并进行深入安全影响分析
- 设计、实现自动化工具对memcg进行全面检测
- 发现53个计数bugs，已确认37个并提交patch

Making Memory Account Accountable: Analyzing and Detecting Memory Missing-account bugs for Container Platforms, ACSAC'22, Distinguished Paper Award

研究动机

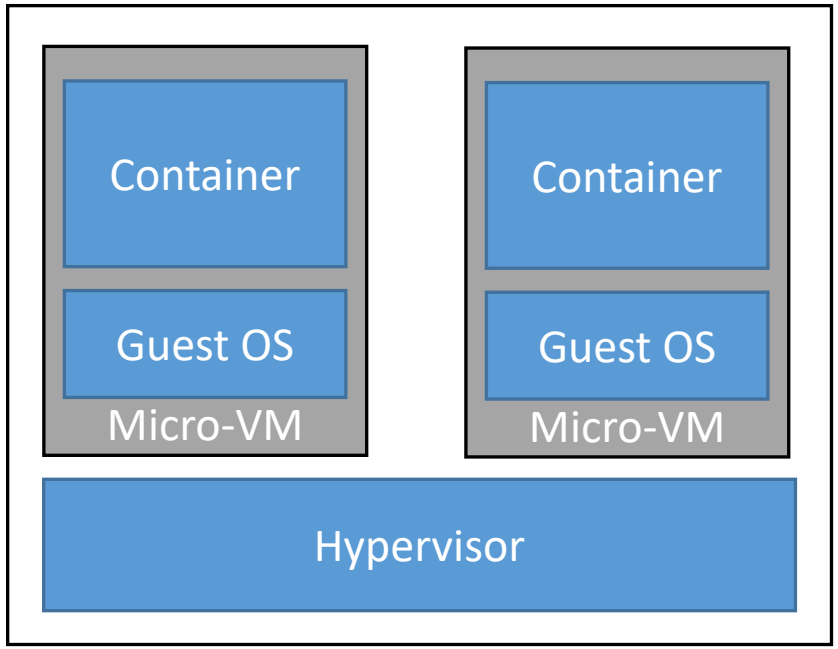
- 云原生社区依赖Linux内核机制构建了大量的容器方案
  - Namespace, control groups，Docker，LXC，runC等
- 关键组件如Kubernetes占有率接近100%
  - Kubernetes has market share of 98.5%
- 这些内核机制/关键组件没有经过系统性的安全分析
- 我们的工作

Namepace隔离做的怎么样?	资源隔离分析：抽象资源攻防 (ACM CCS 21)
Cgroups限制做的怎么样?	资源计数分析：Memcg分析（ACSAC 22)
Micro-VM隔离性怎么样?	Micro-VM容器安全分析：操作转发攻防(USENIX SEC 23)
Kubernetes安全性怎么样?	Kubernetes安全分析：过度权限攻防（ACM CCS 23)



研究动机

- 虚拟机：利用硬件辅助虚拟化技术，每个虚拟机有自己的客户机内核，隔离性好，性能开销大
- 容器：共享宿主机内核，隔离性差，性能好
- Micro-VM based Container
  - 为容器提供专有的客户机内核
  - 虚拟机的隔离性，接近容器的性能
  - 通常用于多组合场景中隔离租户间的容器工作负载
- 已被广泛应用，但安全性缺乏分析



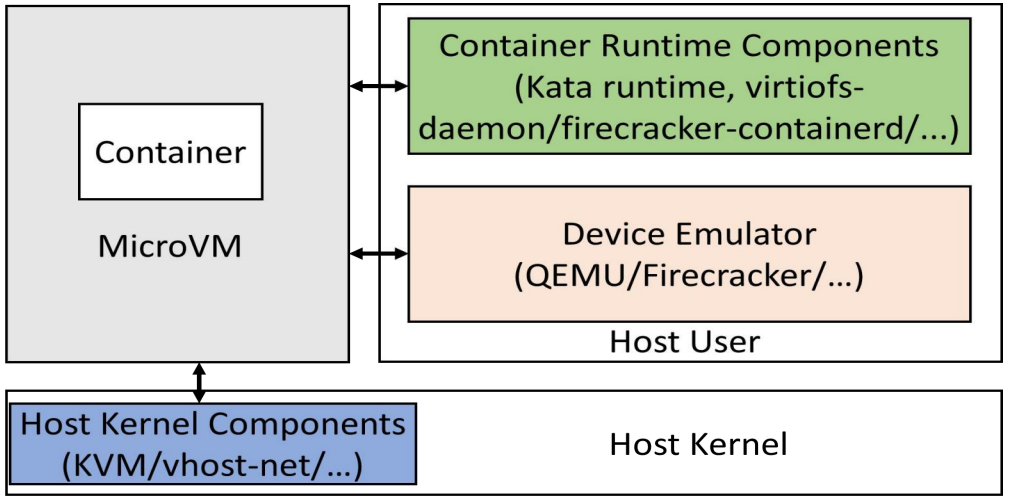
kata containers → **Kata Containers** → **Alibaba Cloud ECI (Elastic Container Instance)**

Firecracker → **Firecracker containerd + Firecracker** → **AWS Fargate**, **AWS Lambda**

MicroVM-based containers架构

- 在一个轻量级VM（microVM）中运行native容器
- 所有的组件可以分为三部分
  - 容器运行时组件
  - 设备模拟器
  - 宿主机内核模块

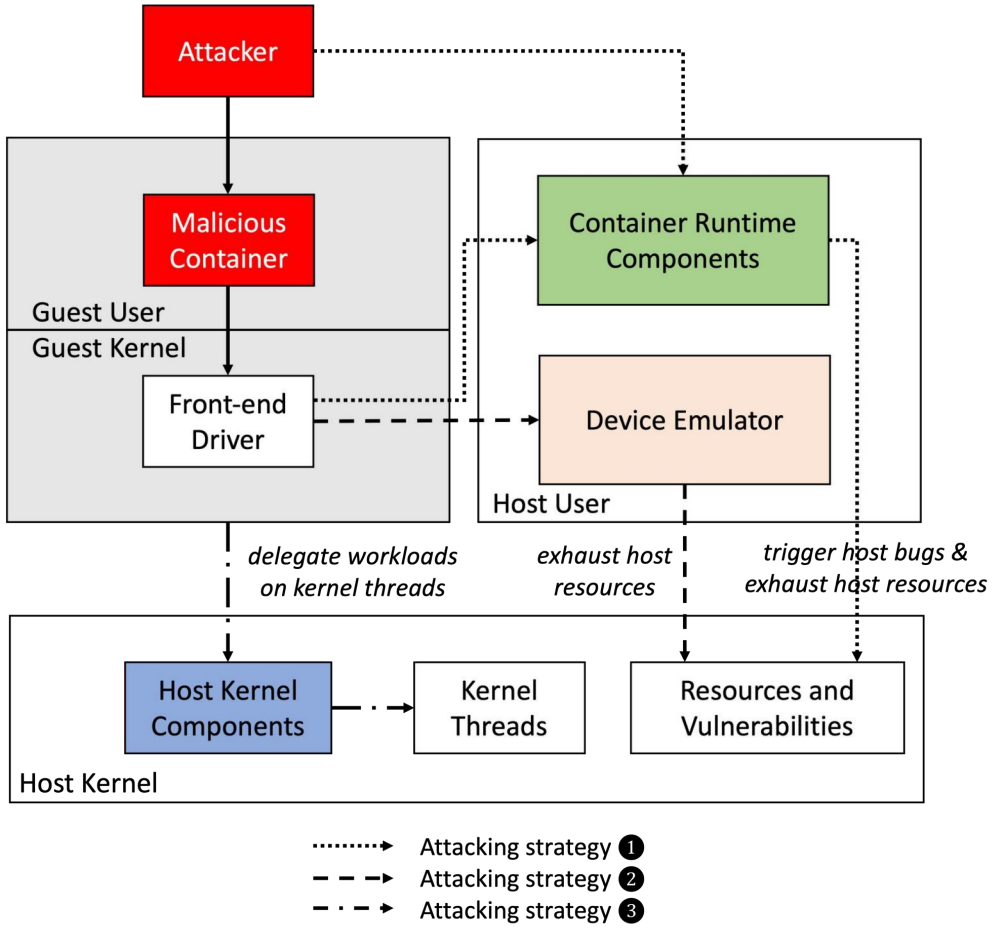
- 基于虚拟机的容器仍然依赖于宿主机内核提供功能支持
- 恶意容器或者容器用户(malicious user)的一些操作会被转发到宿主机内核，由宿主机内核来完成相应的功能
  - Guest syscall -> host syscall
  - Guest syscall -> host kernel function
  - Container management request -> host syscall



这些操作转发（Operation forwarding）安全设计怎么样?
是否可以被利用来打破虚拟机隔离?

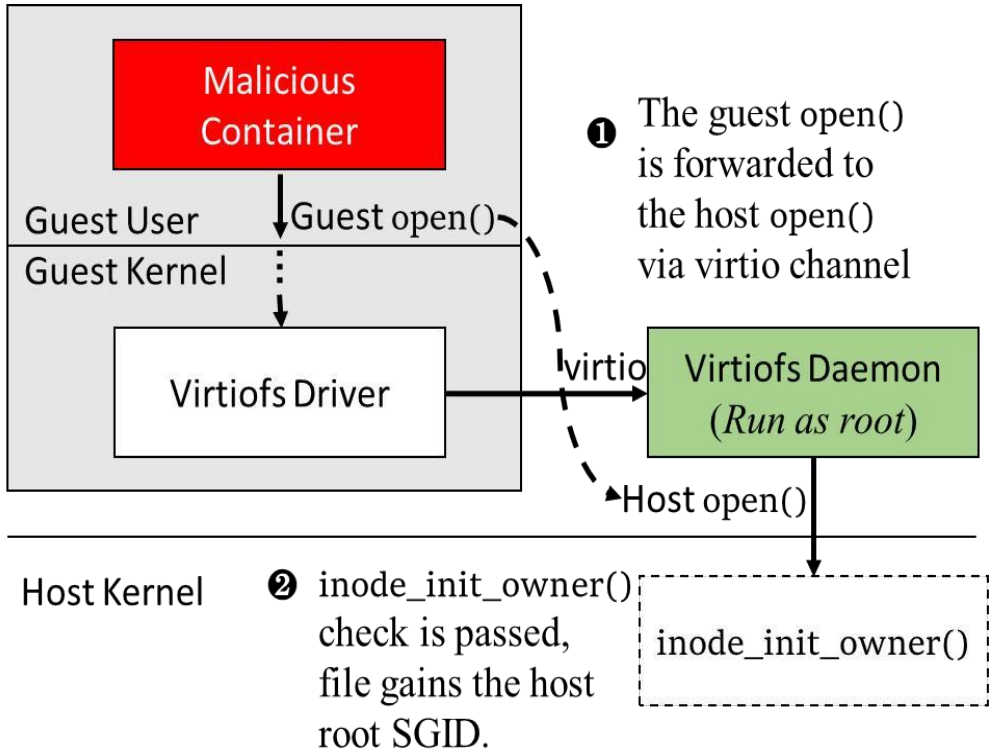
Operation forwarding attacks

- 威胁模型
  - 攻击者控制一个非特权容器
  - 攻击者可以通过客户端发送任意容器管理请求
  - Micro-VM部署常用安全隔离机制
  - 宿主机内核中没有未修复的漏洞+ 硬件虚拟化正常工作
- 基本思路
  - 通过触发operation forwarding路径来发起攻击
    - Guest syscall -> host syscall
    - Guest syscall -> host kernel function
    - Container management request -> host syscall
  - 路径中缺乏正确权限检查：绕过宿主机权限检查，造成提权攻击
  - 路径中缺乏正确资源限制：消耗宿主机的资源，造成DoS攻击
- 攻击策略
  - 通过容器运行时进程，从host user发起攻击
  - 通过设备模拟器，从host user发起攻击
  - 通过Guest driver，从host kernel发起攻击



攻击案例：Virtiofs提权攻击

- Virtiofs介绍
  - 共享文件系统（将宿主机中的目录/文件，共享到虚拟机中）
  - 在Kata Containers中被用于：将容器rootfs，volume从宿主机共享到虚拟机中，然后挂载到虚拟机的runC容器中
  - Virtiofs广泛应用
- 攻击步骤
  - Guest open -> host open
  - Virtiofs daemon进程的supplemental group包含root组
- 攻击影响
  - 允许一个普通用户在创建一个具有SGID位且属于root组的可执行文件
  - 可实现虚拟机逃逸
  - 从Red Hat获得一个CVE-2022-0358
    - Virtiofs daemon drops root group



攻击汇总

Case	Strategy	Trigger Func	Host Func	Impact
Virtiofs daemon escalation	❶	open()	open()	创建具有SGID位的可执行文件；提权
Firecracker-containerd escalation	❶	CreateVM()	chown() & creat()	Chown host目录，清空host文件；提权或DoS
Dirty memory attack(K)	❶	write()	write()	Victim IO性能下降93.4%
Dirty memory attack(F)	❷	write()	write()	Victim IO性能下降86.7%
Nf_conntrack table attack(K)	❸	connect()	tap_sendmsg()	Victim 55.0%丢包率；Nginx服务连接超时
Nf_conntrack table attack(F)	❷	connect()	sendmsg()	Victim 60.0%丢包率；Nginx服务连接超时
Vhost-net attack	❸	sendmsg ()/recvmsg()	handle_tx()/handle_rx	造成1X的带外负载
KVM PIT timer attack	❸	outb()	pit_do_work()	Victim CPU性能下降75%

Takeaway

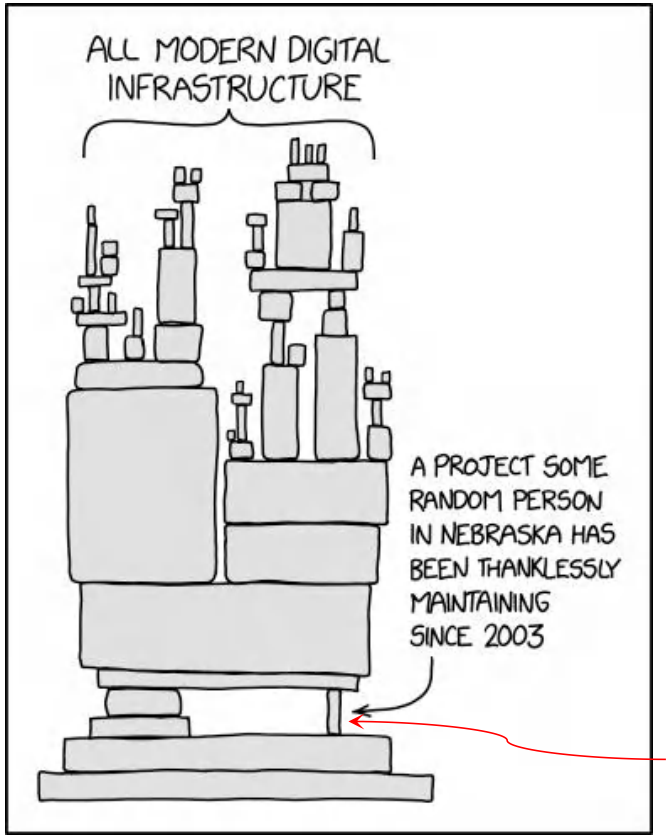
- 提出针对micro-VM container的新型攻击-Operation forwarding attack
- 系统性的分析了攻击发起的方式和造成的安全影响
- 构建了针对Kata container和Firecracker的多种攻击，证明了攻击的practicality
- 给出了相应的缓解措施

Attacks are Forwarded: Breaking the Isolation of MicroVM-based Containers Through Operation Forwarding, USENIX Sec'23

研究动机

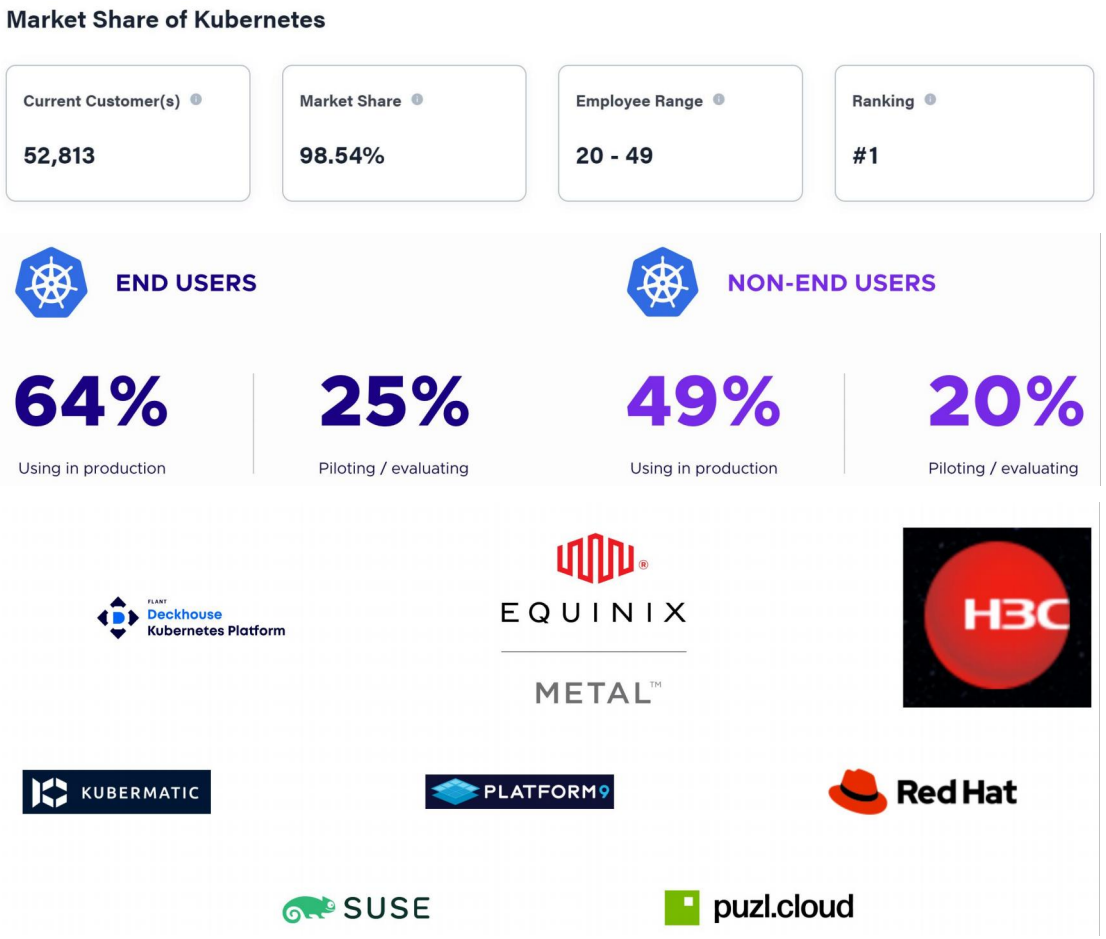
- 云原生社区依赖Linux内核机制构建了大量的容器方案
  - Namespace, control groups，Docker，LXC，runC等
- 关键组件如Kubernetes占有率接近100%
  - Kubernetes has market share of 98.5%
- 这些内核机制/关键组件没有经过系统性的安全分析
- 我们的工作

Namepace隔离做的怎么样?	资源隔离分析：抽象资源攻防 (ACM CCS 21)
Cgroups限制做的怎么样?	资源计数分析：Memcg分析（ACSAC 22)
Micro-VM隔离性怎么样?	Micro-VM容器安全分析：操作转发攻防(USENIX SEC 23)
Kubernetes安全性怎么样?	Kubernetes安全分析：过度权限攻防（ACM CCS 23)

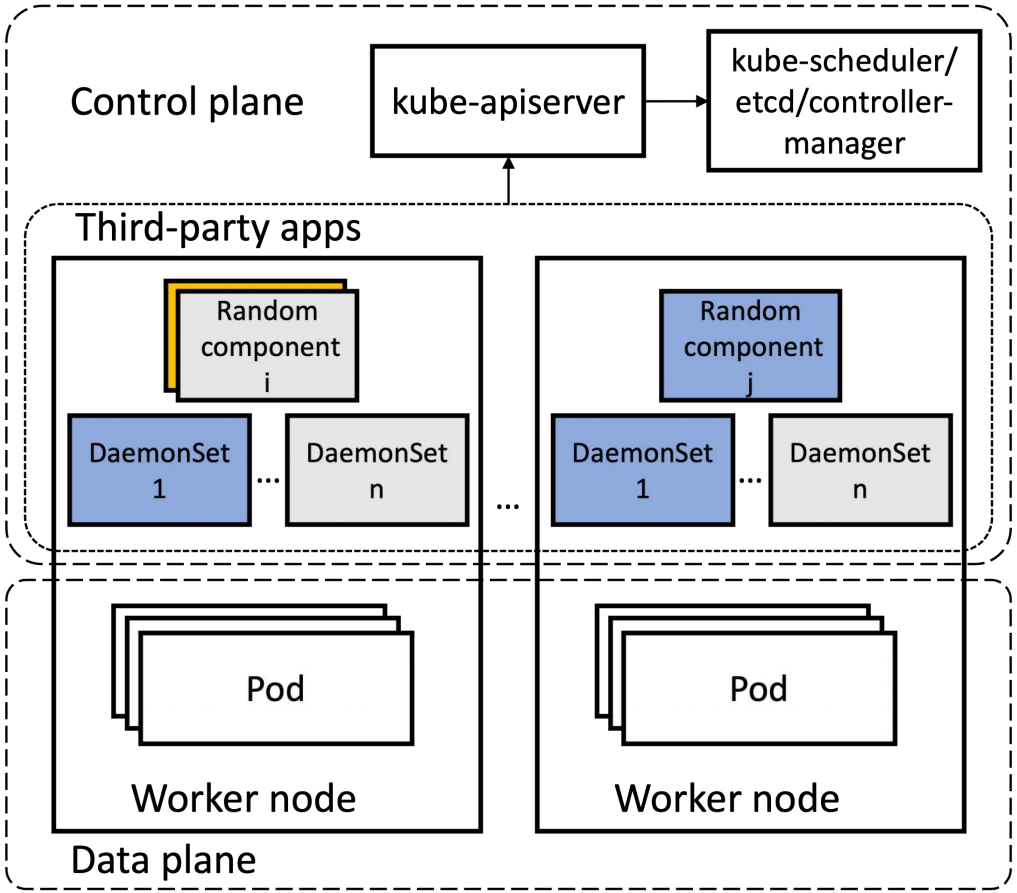


研究动机

Kubernetes占有率接近100%

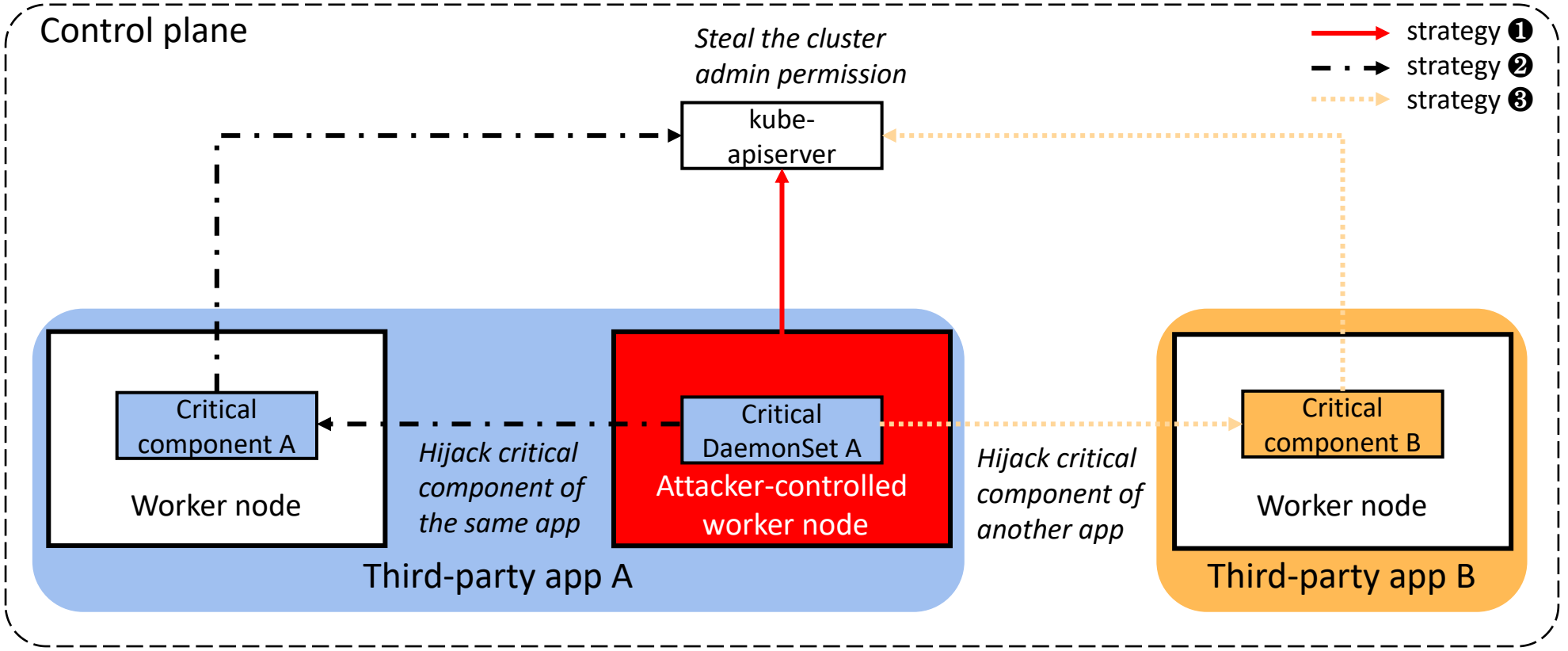


其架构严重依赖api-server，容易出现单点故障



攻击策略

- 三种攻击途径
- 设计了相应的自动化分析工具



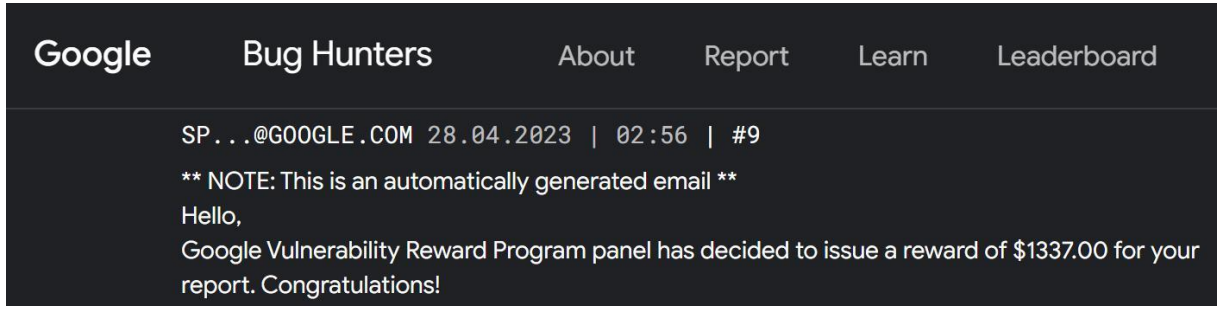
分析结果-CNCF项目

- 分析了所有153个项目，识别出51 问题项目
- 32 项目收到回复
- 获得8个新的CVE

App name	CVE ID	Excessive permissions
CubeFS	2023-30512	get/list secrets
OpenKruise	2023-30617	get/list secrets
Kubevirt	2023-26484	list/patch nodes
		get/list secrets
Fluid	2023-30840	patch nodes
		get/list secrets
Kubewarden	2023-22645	get/list secrets
Open Cluster Management	2023-2250	escalate/bind verbs of clusterroles
		escalate/bind verbs of clusterroles, get/list secrets
Clusternet	2023-30622	* verbs of *.* resources
OpenFeature	2023-29018	list/update verbs of clusterrolebindings

分析结果-四大云厂商

- 四大云厂商均受以上攻击影响
- 拿到Google 安全Bounty



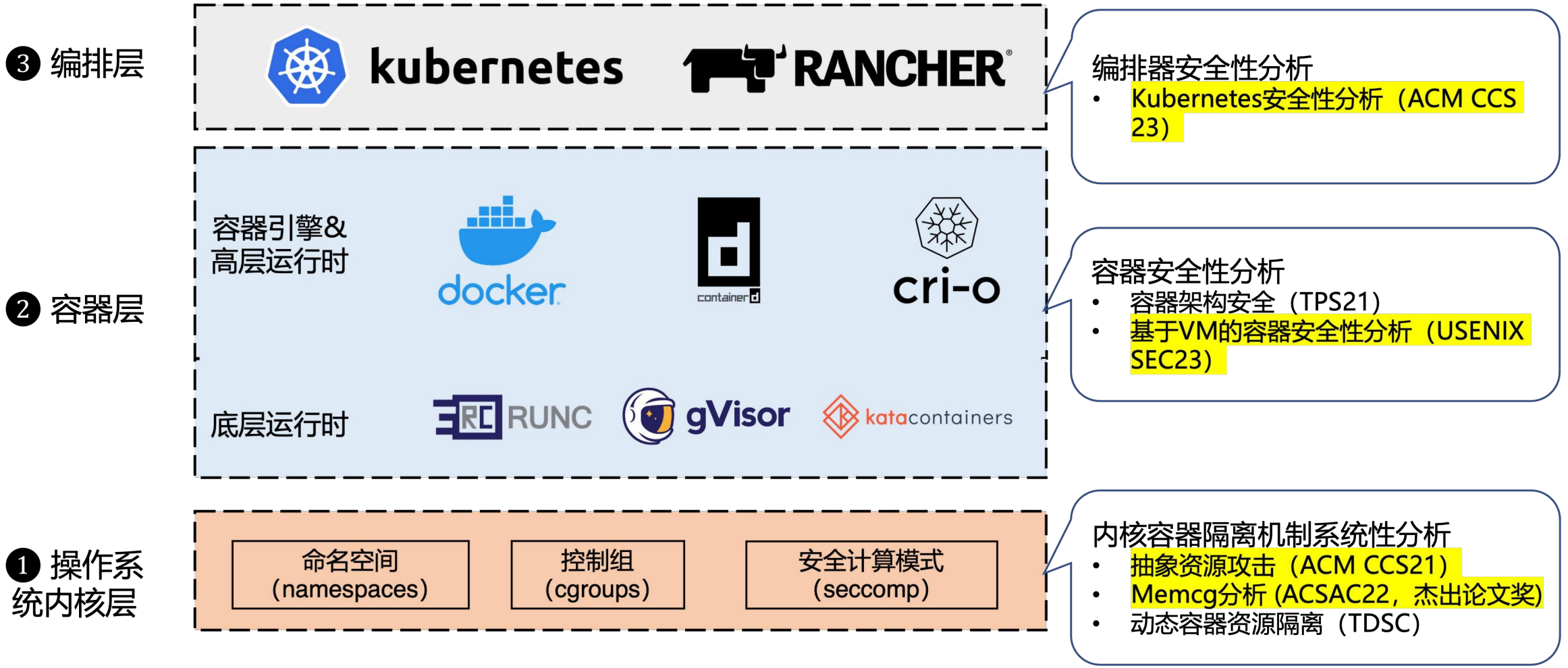
Vendor	App	Excessive permission	Report channel	Report status
Google GKE	Calico Network Policy	patch nodes/status	Google Bug Hunters	Confirmed
	Config Connector	get/list secrets		
	Anthos	delete pods		
		get/list secrets		
Amazon EKS	Amazon VPC CNI	update nodes	AWS Security	Confirmed
	Tetrate Istio Distro	get/list secrets		
	Upbound Universal Crossplane	get/list secrets		
Azure AKS	Secret store CSI driver	get/list secrets	Microsoft Security Response Center	Confirmed
	Azure Policy	get/list *.* resources		
Alibaba Cloud ACK	Prometheus Monitoring	get/list secrets	Alibaba Security Response Center	Confirmed
		get/list secrets		
		get/list secrets		
	CSI Volume Plugin	get/list secrets		
	Node-problem -detector	* verbs of nodes		
	Flannel	patch nodes/status		
	Terway	update/patch nodes		
	Nginx Ingress	list secrets		
	ALB Ingress	get/list secrets		
	MSE Ingress	get/list secrets		
	CloudMonitor Agent	get/list * resources		

Takeaway

- 揭示了过度权限这一新型可攻击面，可用来瘫痪、控制整个K8s集群
- 设计了三种攻击策略，实现了相应的自动化分析攻击
- 发现51/153 CNCF projects存在过度权限问题
- 发现所有四大云厂商存在过度权限问题
- 获得8个CVE以及Google Security Bounty

Take Over the Whole Cluster: Attacking Kubernetes via Excessive Permissions of Third-party Applications, ACM CCS 23

云原生研究总览


❸ 编排层
kubernetes
RANCHER
编排器安全性分析
• Kubernetes安全性分析（ACM CCS 23）
❷ 容器层
容器引擎&
高层运行时
docker
container
cri-o
容器安全性分析
• 容器架构安全（TPS21）
• 基于VM的容器安全性分析（USENIX SEC23）
底层运行时
RUNC
gVisor
katacontainers
❶ 操作系统内核层
命名空间
(namespaces)
控制组
(cgroups)
安全计算模式
(seccomp)
内核容器隔离机制系统性分析
• 抽象资源攻击（ACM CCS21）
• Memcg分析 (ACSAC22，杰出论文奖)
• 动态容器资源隔离（TDSC）

总结

- 容器技术给云计算带来新的机遇，也带来了新的安全挑战
  - 抽象资源攻击
  - 内存计数漏洞
  - 操作转发攻击
  - 冗余权限攻击
- 如何实现高安全性、高性能的云原生系统底座依然是一个开放的问题


CSA GCR cloud security alliance
GREATER CHINA REGION
TCCS
NSFOCUS 绿盟科技
安易科技 ANESEC TECHNOLOGY
THANKS