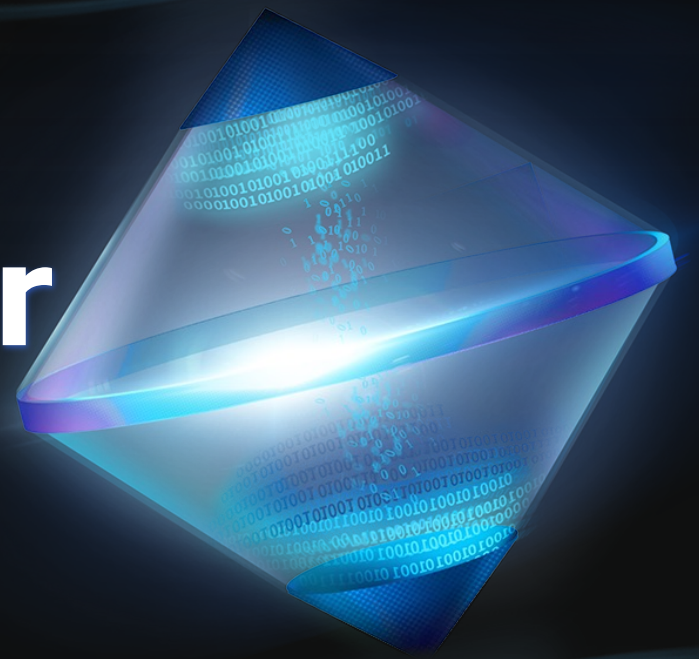
KCon

归源 · 智变

Returning to the source · Intelligent transformation

从 0 到 1 打造云原生容器沙箱——vArmor



演讲人：韦 伟　火山引擎云原生安全负责人

KCon

归源 · 智变

Returning to the source · Intelligent transformation

目录 / CONTENTS

KCon

归源 · 智变

Returning to the source · Intelligent transformation

目录 / CONTENTS

动机与目标

云原生技术趋势

Kubernetes 正在成为云原生操作系统，云原生 Infra 组件（可观测性、安全控制、服务网格、数据库、持续集成...）与容器化微服务的种类和规模都在快速增长

01 大量组织开始使用云原生技术
（44% 的组织在大部分生产环境和业务领域中使用云原生技术）

02 广泛使用云原生技术的组织会更频繁的发布代码
（48% 的组织至少每天发布一次代码）

03 组织采用容器技术的速度超过了云原生技术的成熟度
（30% 的组织在几乎所有开发和部署环节中采用了云原生技术）

04 组织使用云原生技术的最大挑战是缺乏培训和安全性
（安全地使用云原生技术、针对云原生环境的安全防护、合规挑战、易用的安全工具等）

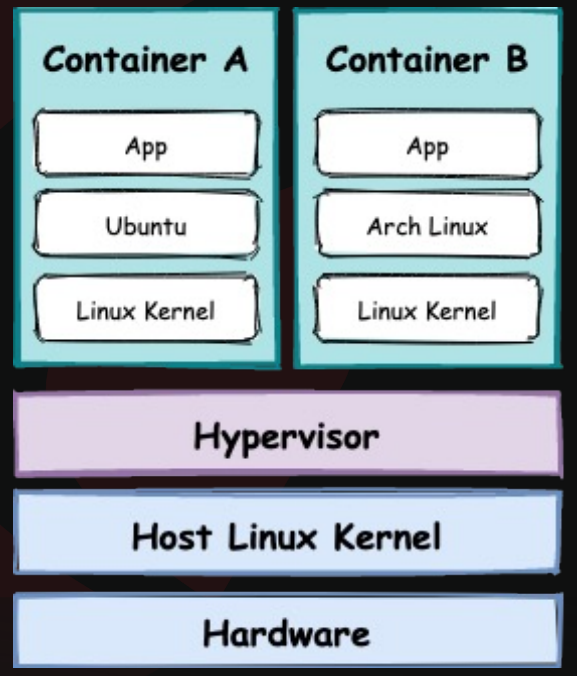
注：数据来自 OcenBase DevCon 2023《云原生技术趋势解读》

动机与目标

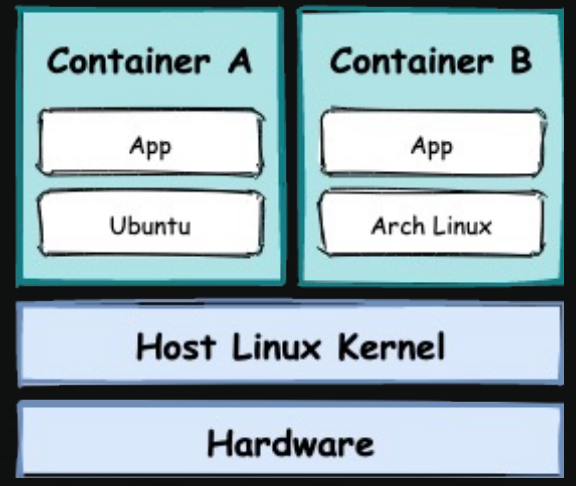
容器加固

增强 Linux 容器隔离性，减少内核攻击面、增加权限提升的难度与成本；

为无法使用硬件虚拟化容器的多租户、sidecar 等场景提供 Plan B；



硬件虚拟化容器
门槛高 & 隔离性强



Linux 容器
开销低 & 隔离性弱

动机与目标

容器加固

增强 Linux 容器隔离性，减少内核攻击面、增加权限提升的难度与成本；
为无法使用硬件虚拟化容器的多租户、sidecar 等场景提供 Plan B；

事中防御

降低不安全配置、0day 漏洞带来的安全风险；
在漏洞被修复前，增加攻击者在容器中进行漏洞利用、横向移动攻击的难度和成本；

Isolation between Linux containers

When designing services with multiple containers, it is crucial to be precise in determining how they should work together—if at all—and what the security implications of such interaction might be. In both ApsaraDB RDS and AnalyticDB our database container shared different Linux namespaces with other operational containers in the K8s pod. Specifically, they shared the PID namespace, which allowed our container to access other processes in the operational containers and the filesystem. This mistake was fatal: it enabled us to perform lateral movement to the operational containers and therefore escape our container!

#BrokenSesame

动机与目标

容器加固

增强 Linux 容器隔离性，减少内核攻击面、增加权限提升的难度与成本；
为无法使用硬件虚拟化容器的多租户、sidecar 等场景提供 Plan B；

事中防御

降低不安全配置、0day 漏洞带来的安全风险；
在漏洞被修复前，增加攻击者在容器中进行漏洞利用、横向移动攻击的难度和成本；

CVE-2021-22555 is a 15 years old heap out-of-bounds write vulnerability in Linux Netfilter that is powerful enough to bypass all modern security mitigations and achieve kernel code execution. It was used to break the kubernetes pod isolation of the kCTF cluster and won 10000$ for charity (where Google will match and double the donation to 20000$).

Laurent Cheylus @lcheylus · 2022年3月17日
Linux Netfilter firewall Vulnerability : exploitable to achieve kernel code execution (via ROP), giving full local privilege escalation or **container** escape - Discovery and Exploitation of **CVE-2022-25636** by @kallsyms nickgregory.me/linux/security...

Towards Cybersecurity @TowardsCybersec · 2022年3月6日
Details have emerged about a now-patched high-severity flaw - **CVE-2022-0492** (CVSS score: 7.0) in the Linux kernel that could potentially be abused to escape a container in order to execute arbitrary commands on the container host.

fattire @fat_tire · 5月15日
PoC code for **CVE-2023-32233** drops tomorrow. Why hasn't @ubuntu_sec updated its kernels to patch this local user->root exploit, which I think might even affect users in **containers**? ubuntu.com/security/CVE-2... openwall.com/lists/oss-secu...

动机与目标

容器加固

增强 Linux 容器隔离性，减少内核攻击面、增加权限提升的难度与成本；

为无法使用硬件虚拟化容器的多租户、sidecar 等场景提供 Plan B；

事中防御

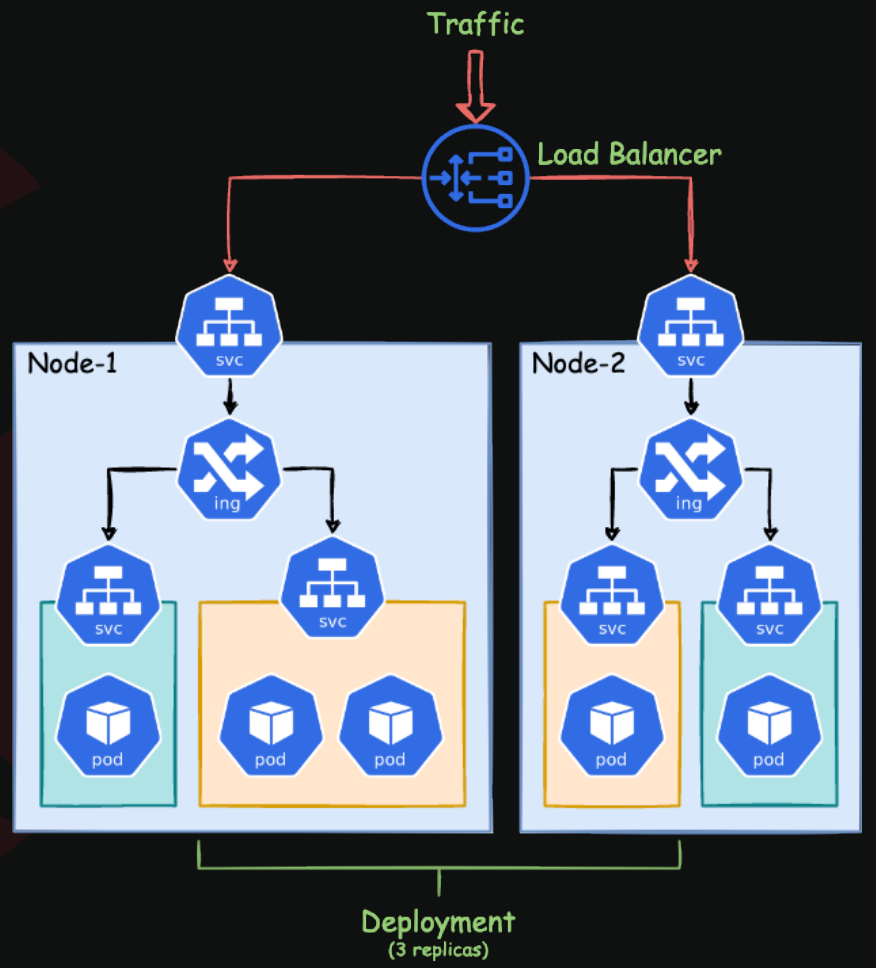
降低不安全配置、0day 漏洞带来的安全风险；

在漏洞被修复前，增加攻击者在容器中进行漏洞利用、横向移动攻击的难度和成本；

云原生化

消除或降低 Kubernetes 安全功能的使用门槛、成本；

通过标准化接口提供一致性体验，打通"**工作负载与加固进程**"、"**业务与安全**"之间的 Gap；



视角差异：业务 vs. 安全

KCon

归源 · 智变

Returning to the source · Intelligent transformation

目录 / CONTENTS

从 0 到 1 打造 vArmor

关键设计决策

vArmor 在设计和实现过程中的核心思想是通过标准化接口为用户提供一致的体验，最大程度消除或缩小云原生场景中"系统安全防护"与"系统研发运维"之间的鸿沟，从而实现一个轻量易用的基于 LSM 的云原生容器沙箱。

Operate as a cloud-native system：遵循 Kubernetes Operator 设计模式，通过声明式 CRD API 抽象和屏蔽底层逻辑，对 Workloads 的多副本容器进行沙箱加固。

Out-of-the-box：针对常见加固需求提供一系列**内置策略，同时**可根据语法自定义策略。**并且支持策略动态调整**，从而实现开箱即用功能。

安全模型

- 拒绝所有未显式声明放行的行为
- 访问控制的粒度细，约束强
- 较难收集应用的所有行为
- 性能损耗相对较大

Deny by Default

我们
的选择

大部分
LSM

Allow by Default

- 仅拒绝显式声明要拦截的行为
- 访问控制的粒度粗，约束弱
- 较容易制定访问控制策略
- 性能损耗相对较小

安全模型

选择 Allow by Default 的原因

01 业务迭代速度快，现有技术手段无法确保收集到应用的所有行为

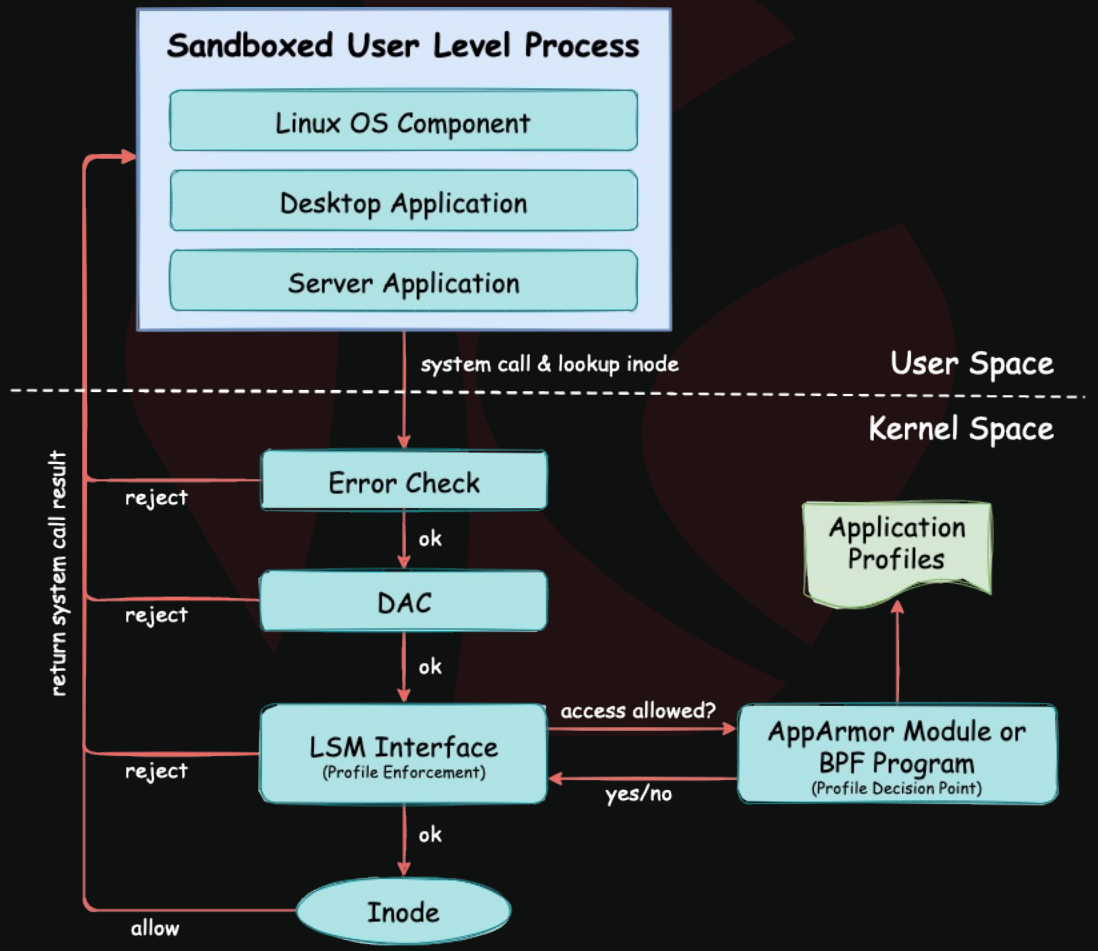
02 相比收集应用的所有合法行为，更容易定义应用的非法、非预期行为

03 仍需考虑沙箱引入的性能成本，在提供沙箱防护时尽量降低性能影响

04 为了同时兼容 AppArmor & BPF LSM

访问控制器

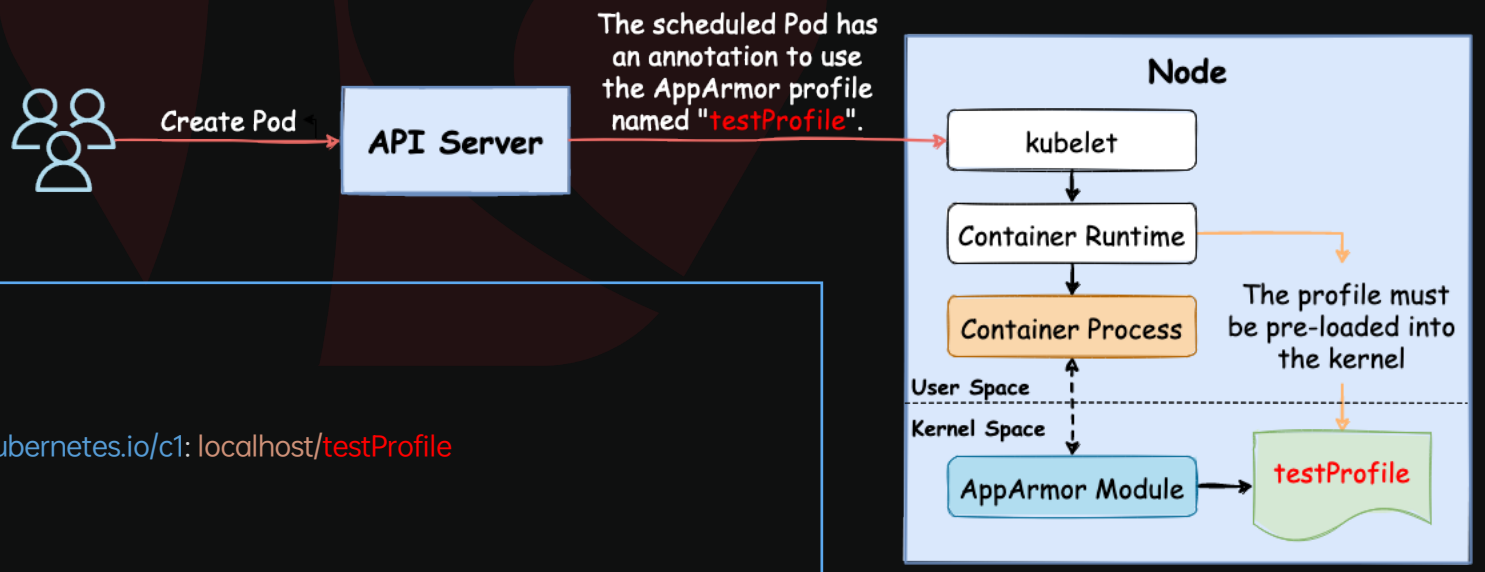
vArmor 利用 Linux Security Modules (LSM) 框架，在 Linux Kernel 中对容器进程的行为进行访问控制，从而实现容器沙箱。



访问控制器

AppArmor LSM

- Kubernetes 自 v1.4 引入对 AppArmor 的支持
- Runtime 为容器设置进程安全上下文 /proc/self/attr/current
- Runtime 包含一个默认粗粒度的 Profile
- 需自行编写并管理自定义 AppArmor Profile，使用门槛较高

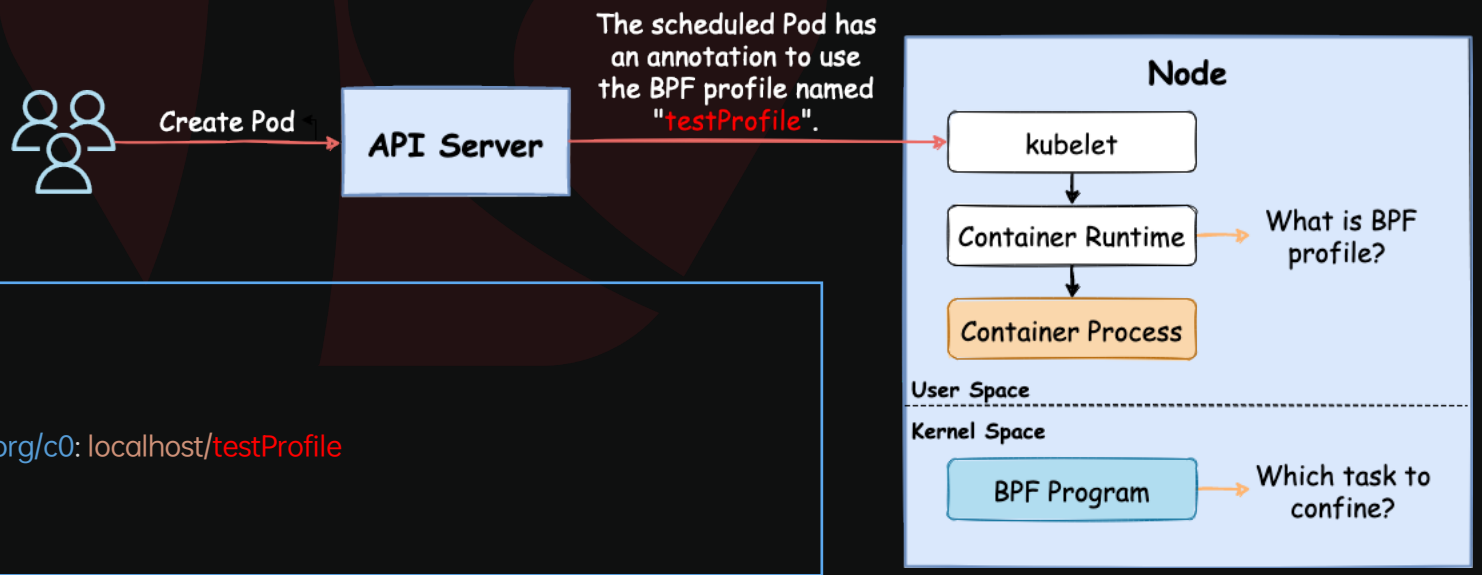



```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    container.apparmor.security.beta.kubernetes.io/c1: localhost/testProfile
  name: test
spec:
  ...
```

访问控制器

BPF LSM

- Linux 自 v5.7 引入对 BPF LSM 的支持
- 提供了一种更加通用且灵活的强制访问控制方法
- 无法使用进程安全上下文 /proc/self/attr/current
- 功能和性能受限



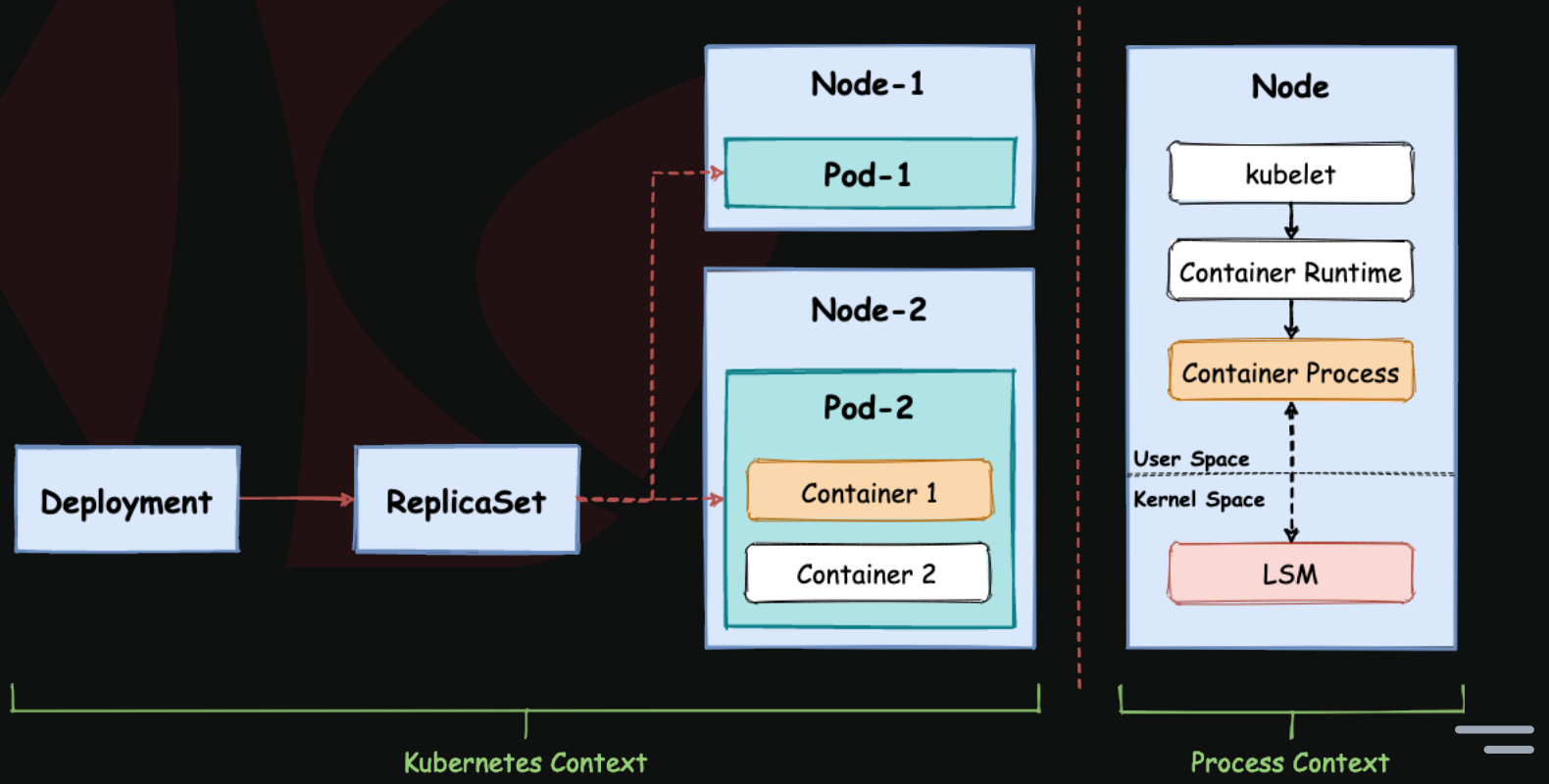

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    container.bpf.security.beta.varmor.org/c0: localhost/testProfile
  name: test
spec:
  ...
```

打通任督二脉 —— K8s 上下文 & 进程上下文

挑战

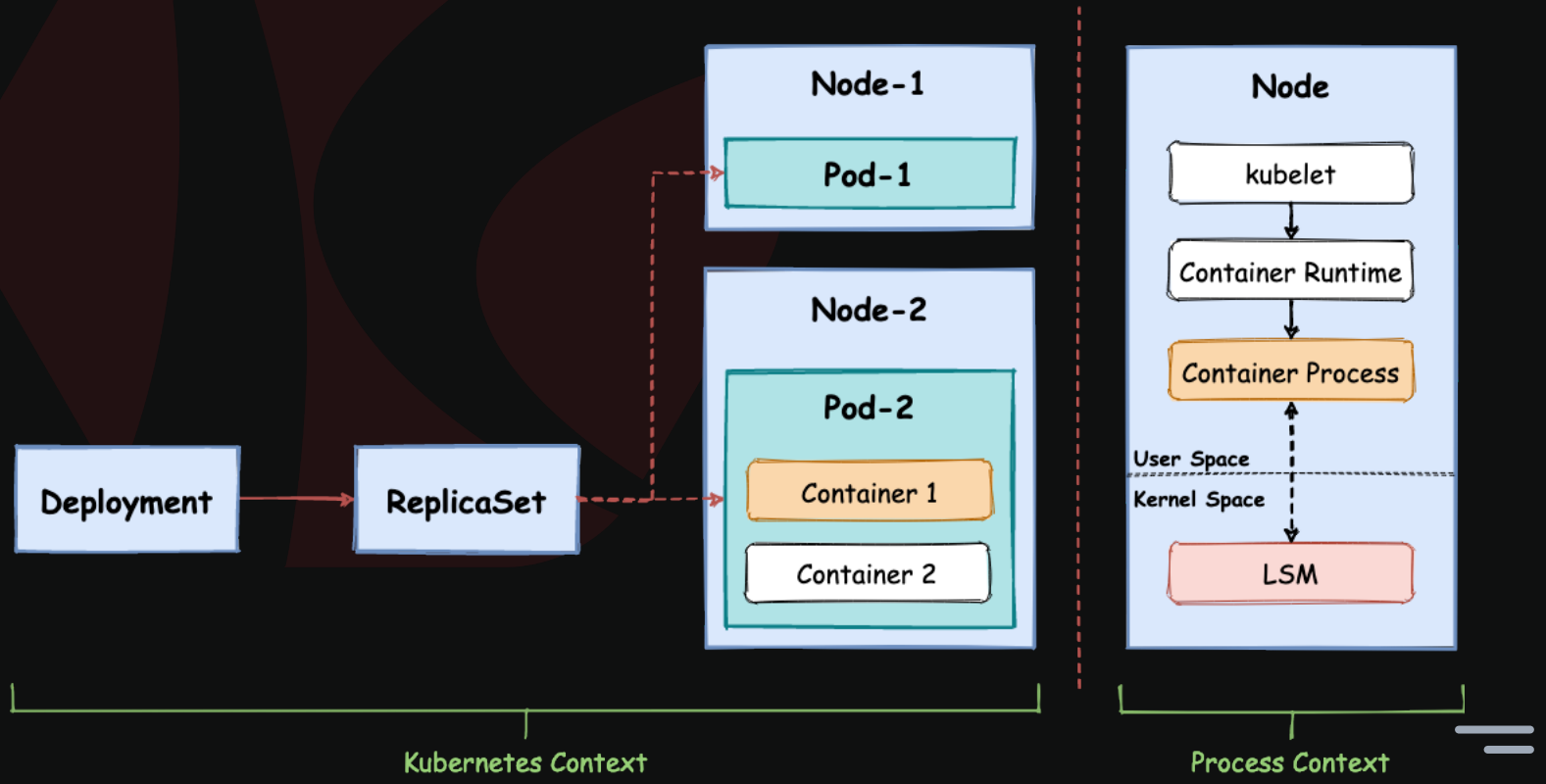
- BPF LSM 无法使用进程安全上下文
- K8s、容器运行时等对 BPF LSM 无额外支持

How can we determine if a process belongs to the target container that we intend to protect for the workload?

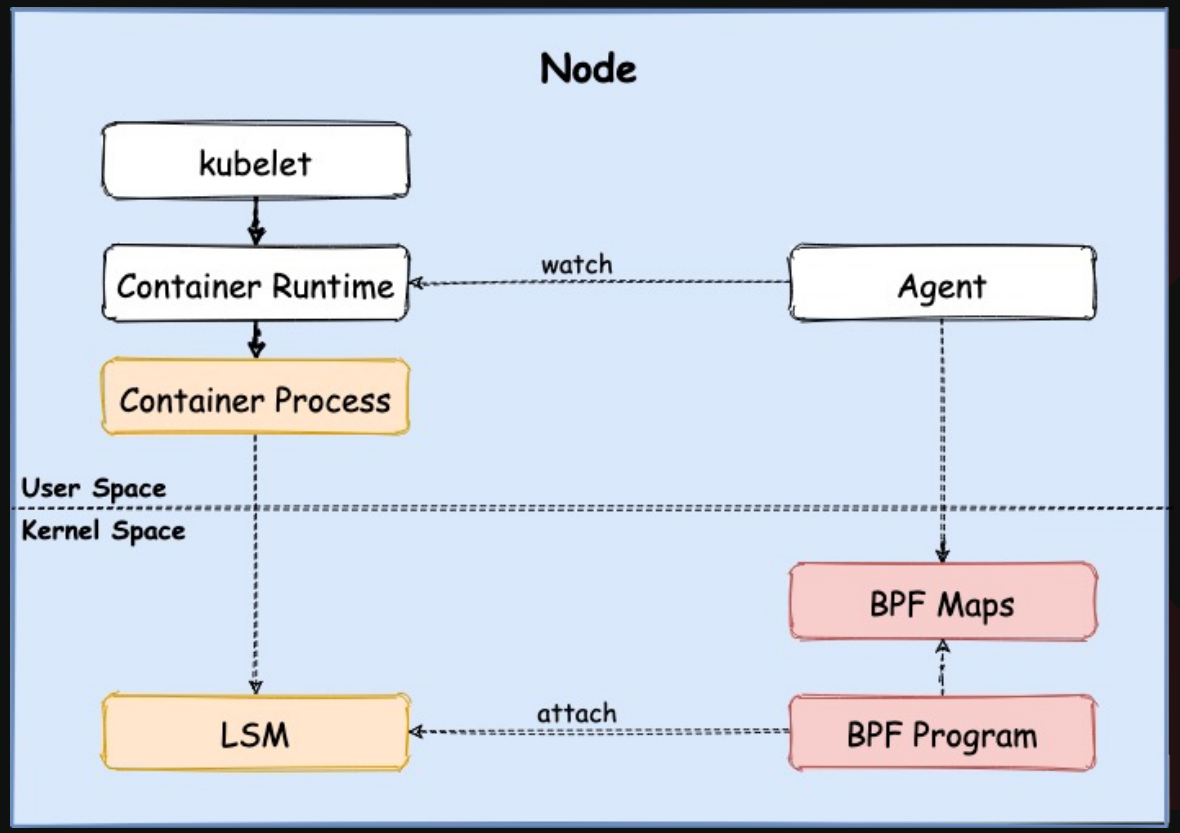


打通任督二脉 —— K8s 上下文 & 进程上下文

- 修改 Container Runtime
- 利用 K8s 的 Informer 机制
- 利用 Runtime 的 Events 机制



打通任督二脉 —— K8s 上下文 & 进程上下文



EVENTS

几乎所有的 Runtime 都支持 Event 机制，可用于获知容器创建、启动、删除等事件，并提供获取必要信息的接口

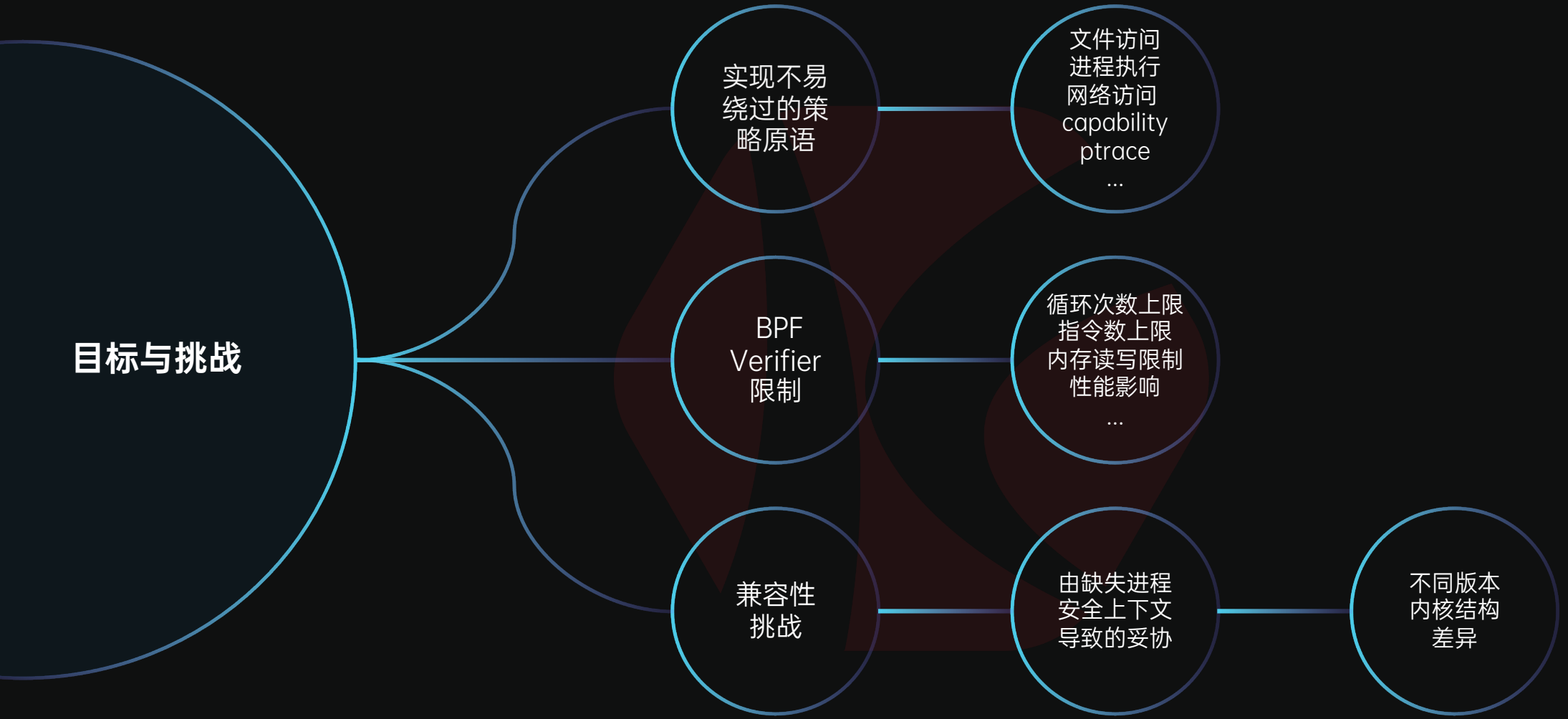
MNT NS ID

BPF Program 可利用容器进程的 mount ns id 来区分当前进程是否为强制访问控制目标

TOCTOU

容忍此处的 TOCTOU 风险，只对 long-running service 进行防护；尽量减少攻击窗口

戴着脚镣跳舞 —— 实现 BPF Enforcer



BPF Verifier 限制

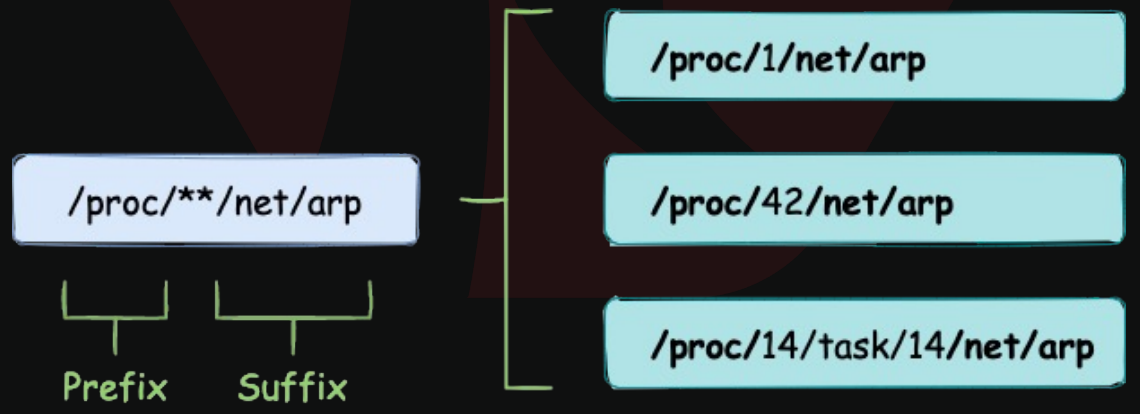
以文件路径匹配为例

- 挑战
  - 策略原语需同时支持全路径匹配和通配符匹配
  - Linux 最长文件路径 4096 Byte，最长文件名 256 Byte，最大目录深度无约束
  - BPF 对堆栈空间、指令数、递归深度等均存在
  - 不能给性能带来显著影响
- 优势
  - BPF LSM 需要 Linux 5.7+，可充分利用 BPF 的新特性：CORE、tail call、helper functions、bounded loop 等
  - Allow By Default 的设计决策

BPF Verifier 限制

以文件路径匹配为例

- 解决方案
  - 控制文件路径提取深度
  - 控制字符串循环次数
  - 控制 Pattern 中的通配符数量（当前仅支持 * 和 **，且不能出现多次）
  - 以 ** 为例，提取出文件路径后，简化为前后缀匹配



BPF Enforcer 兼容性挑战

以 Capability 控制为例

```
$ kubectl exec -n demo demo-2-56cdc47fc-v68f9 -c c0 -it -- /bin/bash
root@demo-2-56cdc47fc-v68f9:/# echo 1 > /tmp/1
bash: /tmp/1: Operation not permitted
```

```
request cap: 0x15
current cap_effective: 0x1fffeffffff
return: 0x0
Returning from:  0xffffffff81094fe0 : ns_capable_common+0x0/0x50 [k
Returning to  :  0xffffffff812fd8c0 : xattr_permission+0x110/0x120
 0xffffffff812feab7 : __vfs_setxattr_locked+0x37/0xf0 [kernel]
 0xffffffff812febd9 : vfs_setxattr+0x69/0x110 [kernel]
 0xffffffffc08d2224
 0xffffffffc08d2224 (inexact)  ovl_check_setxattr
 0xffffffffc08da77f (inexact)  ovl_set_origin
 0xffffffffc08da8a5 (inexact)  ovl_copy_up_inode
 0xffffffffc08dad47 (inexact)  ovl_copy_up_one
 0xffffffffc08db9bb (inexact)  ovl_copy_up_flags
 0xffffffffc08d73f1 (inexact)  ovl_create_or_link
 0xffffffffc08d7b54 (inexact)  ovl_create_object
 0xffffffff812e2954 : path_openat+0xe24/0x1070 [kernel] (inexact)
 0xffffffff812e4783 : do_filp_open+0x93/0x100 [kernel] (inexact)
 0xffffffff812cd568 : do_sys_openat2+0x228/0x2d0 [kernel] (inexact)
 0xffffffff812ceb9b : do_sys_open+0x4b/0x80 [kernel] (inexact)
 0xffffffff8185c65d : do_syscall_64+0x2d/0x70 [kernel] (inexact)
 0xffffffff81a000a9 : entry_SYSCALL_64_after_hwframe+0x61/0xc6 [ker
```

```
static int
xattr_permission(struct inode *inode, const char *name, int mask)
{
 ...

 /*
  * The trusted.* namespace can only be accessed by privileged users.
  */
 if (!strncmp(name, XATTR_TRUSTED_PREFIX, XATTR_TRUSTED_PREFIX_LEN)){
  if (!capable(CAP_SYS_ADMIN))
   return (mask & MAY_WRITE) ? -EPERM : -ENODATA;
  return 0;
 }

 ...

}
```

BPF Enforcer 兼容性挑战

以 Capability 控制为例

- 根因
  - 需要 CAP_SYS_ADMIN 权限才能设置 /tmp 目录 trusted.* 名称前缀的 xattr
  - overlayfs 为此进行了临时提权，此时进程的安全上下文被置空
  - BPF Enforcer 未能感知到临时提权行为，从而导致了非预期阻断

```
static int ovl_copy_up_flags(struct dentry *dentry, int flags)
{
 ...
 old_cred = ovl_override_creds(dentry->d_sb);
 ...
 ovl_copy_up_one() -> ovl_copy_up_inode()
  ovl_set_origin()
   ovl_check_setxattr()
    ovl_do_setxattr()
     vfs_setxattr(dentry, "trusted.overlay.origin", value, size, 0);
 revert_creds(old_cred);
}
```

```
const struct cred *ovl_override_creds(struct super_block *sb)
{
 struct ovl_fs *ofs = sb->s_fs_info;
 // creator_cred is initialized as 0x1fffefffffff in overlayfs module.
 return override_creds(ofs->creator_cred);
}
```

```
$ getfattr -m '' -d var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/9724/fs/tmp
# file: var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/9724/fs/tmp
trusted.overlay.origin=0sAPsdAAHVS+YgD/hGgbtunllG7qwdvRZOACDzrnk=
```

BPF Enforcer 兼容性挑战

以 Capability 控制为例

- 解决方案

  曲线救国——由进程安全上下文缺失导致的妥协

```
struct user_namespace *current_user_ns = get_task_user_ns(current);

if (current_user_ns == init_user_ns && current_effective_mask == 0x1fffeffffff) {
    return ret;
}
```

沙箱策略

- AppArmor Enforcer 策略原语
  - 访问控制功能与内核版本相关（详见 kernel feature matrix for AppArmor）

```
$ ls /sys/kernel/security/apparmor/features
capability  caps  domain  file  mount  namespaces  network_v8  policy  ptrace  query  rlimit  signal
```

  - 根据语法编写策略（详见 syntax of security profiles for AppArmor）

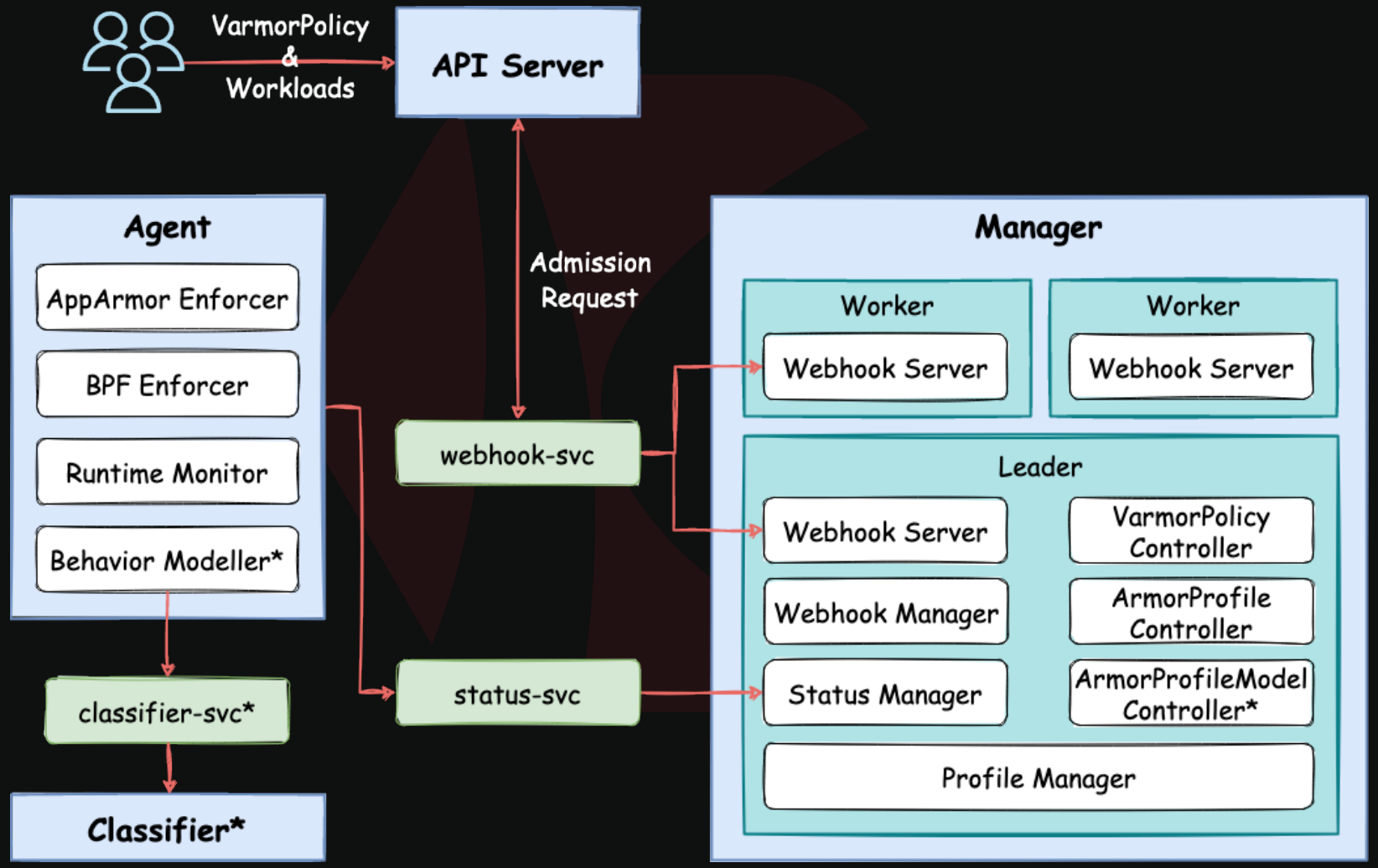
- BPF Enforcer 策略原语
  - 访问控制功能
    - capability
    - 文件访问、程序执行（全路径匹配、有限通配符）
    - 网络外连（IP， CIDR， Port）
    - ...
  - 根据 CRD API 和语法编写策略（详见 The Policy Manual）

沙箱策略

- 开箱即用
  - 内置策略
  - 动态调整（可在无需重启工作负载的前提下，动态调整目标容器的防护策略）

类型	说明
Always Allow	在容器启动时不对其施加任何限制
Runtime Default	使用与容器运行时组件相同的默认策略进行基础防护，防护强度较弱。 （如 containerd 的 cri-containerd.apparmor.d）
Hardening	对容器进行加固，减少攻击面的策略。包括： • 阻断特权容器的常见逃逸向量 • 禁用 capabilities • 阻断部分内核漏洞利用向量
Attack Protection	针对黑客渗透入侵手法进行防护的策略。目的是增加攻击的难度和成本，进行纵深防御。包括： • 容器信息泄露缓解 • 禁止执行敏感行为 • 对特定可执行文件进行沙箱限制（仅限 AppArmor enforcer）
Vulnerability Mitigation	针对 Workloads 不安全配置、0day 漏洞等，提供在漏洞被修复前进行防护的策略，阻断或增加漏洞利用的难度（注：取决于漏洞类型或漏洞利用向量）。

从 0 到 1 打造 vArmor



从 0 到 1 打造 vArmor

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-1
  namespace: demo
  labels:
    sandbox.varmor.org/enable: "true"
    app: demo-1
spec:
  replicas: 6
  selector:
    matchLabels:
      app: demo-1
  template:
    metadata:
      labels:
        app: demo-1
      annotations:
        container.bpf.security.beta.varmor.org/c0: unconfined
    spec:
      containers:
      - name: c0
        image: debian:10
        command: ["/bin/sh", "-c", "sleep infinity"]
      - name: c1
        image: debian:10
        command: ["/bin/sh", "-c", "sleep infinity"]
```

防护对象（示例）

```yaml
apiVersion: crd.varmor.org/v1beta1
kind: VarmorPolicy
metadata:
  name: demo-1-policy
  namespace: demo
spec:
  target:
    kind: Deployment
    selector:
      matchLabels:
        app: demo-1
    containers:
    - c1
  policy:
    enforcer: BPF
    mode: EnhanceProtect
    enhanceProtect:
      hardeningRules:
      - disable-cap-privileged
      - disable-cap-net-raw
      attackProtectionRules:
      - rules:
        - disable-write-etc
        - mitigate-sa-leak
        - disallow-metadata-service
        bpfRawRules:
          files:
          - pattern: "/etc/shadow"
            permissions: [r, w]
          network:
            egresses:
            - ipBlock: 172.16.0.0/11
```

防护策略（示例）

KCon

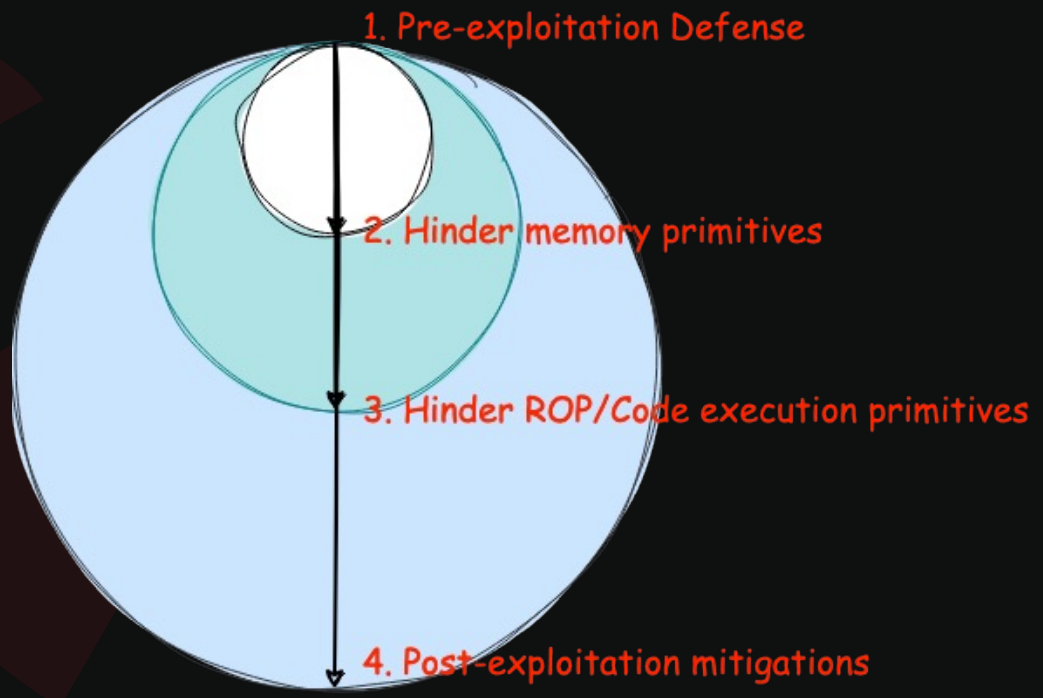
归源 · 智变

Returning to the source · Intelligent transformation

目录 / CONTENTS

缓解内核漏洞利用

- 可以在四个阶段进行防御，越靠后难度与成本越高
- Linux 内核安全现状
  - 主线在不断引入漏洞利用缓解措施
  - 不同发行版会引入各自的安全加固项
  - 第三方加固 patch 进行更深入的防御（KSPP, Grsecurity, VED 等）
- vArmor 当前主要通过减少内核攻击面来防御内核漏洞

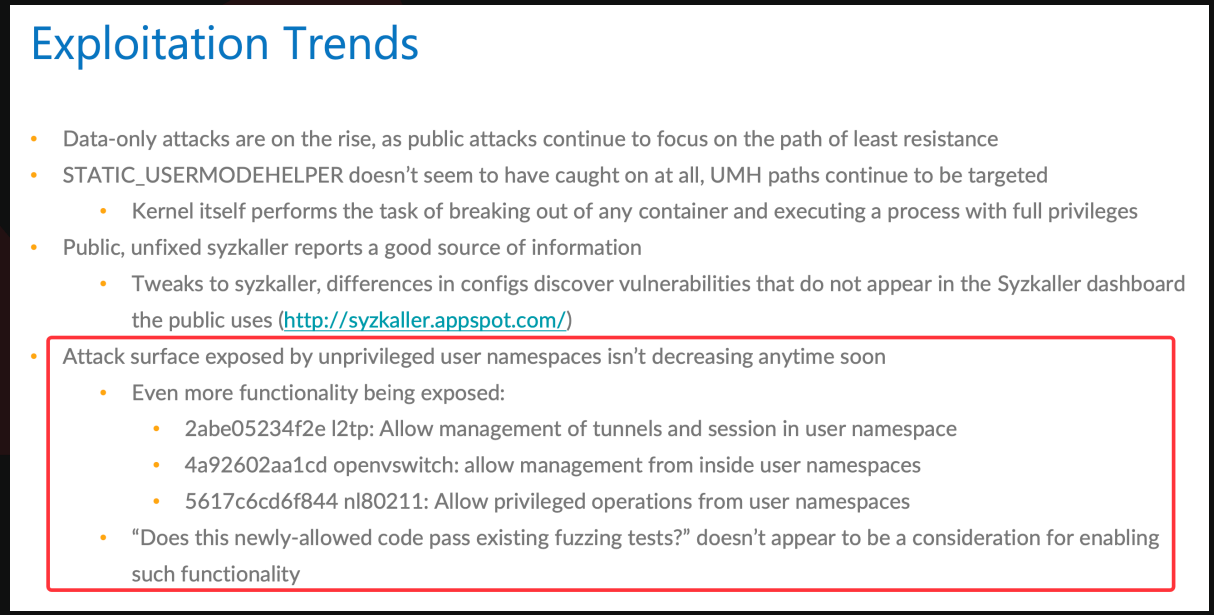


缓解内核漏洞利用

以 User Namespace 引入的内核攻击面为例

- 创建 User Namespace 无需任何特权
- 攻击者获取子 User Namespace 内特权后可触达更多内核代码
- 近年来，利用此攻击面的内核漏洞呈爆发趋势

  CVE-2021-22555, CVE-2022-0185, CVE-2022-25636, CVE-2022-2588, CVE-2023-32233 ...
- 现有防御手段
  - user.max_user_namespaces（可能与业务冲突）
  - kernel.unprivileged_userns_clone（非主线）
  - Kubernetes SeccompDefault（默认关闭）
  - LSM hook: userns_create（Linux v6.1+）



Exploitation Trends

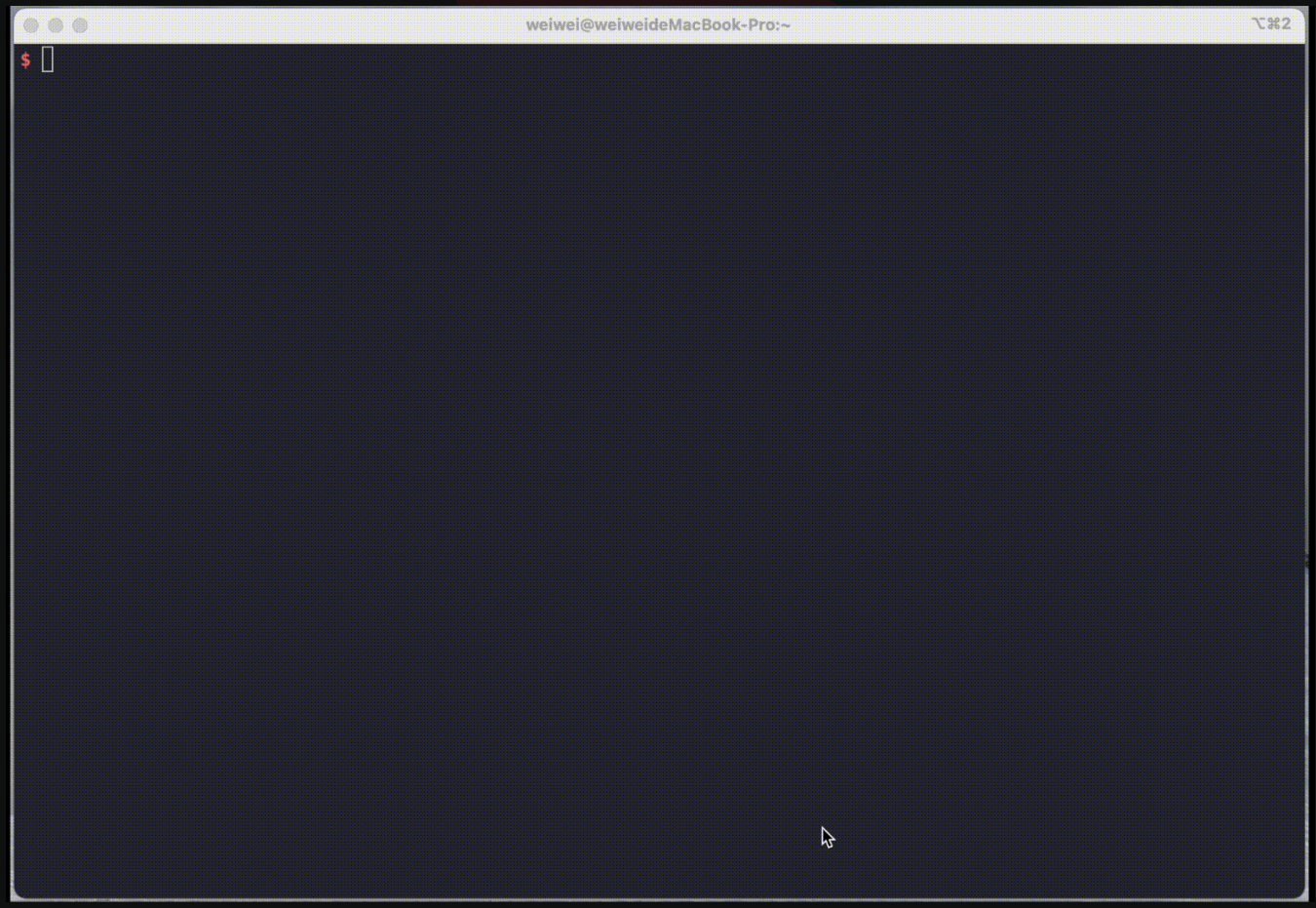
- Data-only attacks are on the rise, as public attacks continue to focus on the path of least resistance
- STATIC_USERMODEHELPER doesn't seem to have caught on at all, UMH paths continue to be targeted
  - Kernel itself performs the task of breaking out of any container and executing a process with full privileges
- Public, unfixed syzkaller reports a good source of information
  - Tweaks to syzkaller, differences in configs discover vulnerabilities that do not appear in the Syzkaller dashboard the public uses (http://syzkaller.appspot.com/)
- Attack surface exposed by unprivileged user namespaces isn't decreasing anytime soon
  - Even more functionality being exposed:
    - 2abe05234f2e l2tp: Allow management of tunnels and session in user namespace
    - 4a92602aa1cd openvswitch: allow management from inside user namespaces
    - 5617c6cd6f844 nl80211: Allow privileged operations from user namespaces
  - "Does this newly-allowed code pass existing fuzzing tests?" doesn't appear to be a consideration for enabling such functionality

<10 Years of Linux Security> by grsecurity

缓解内核漏洞利用

使用 vArmor 缓解 User Namespace 引入的攻击面（以 CVE-2021-22555 为例进行演示）

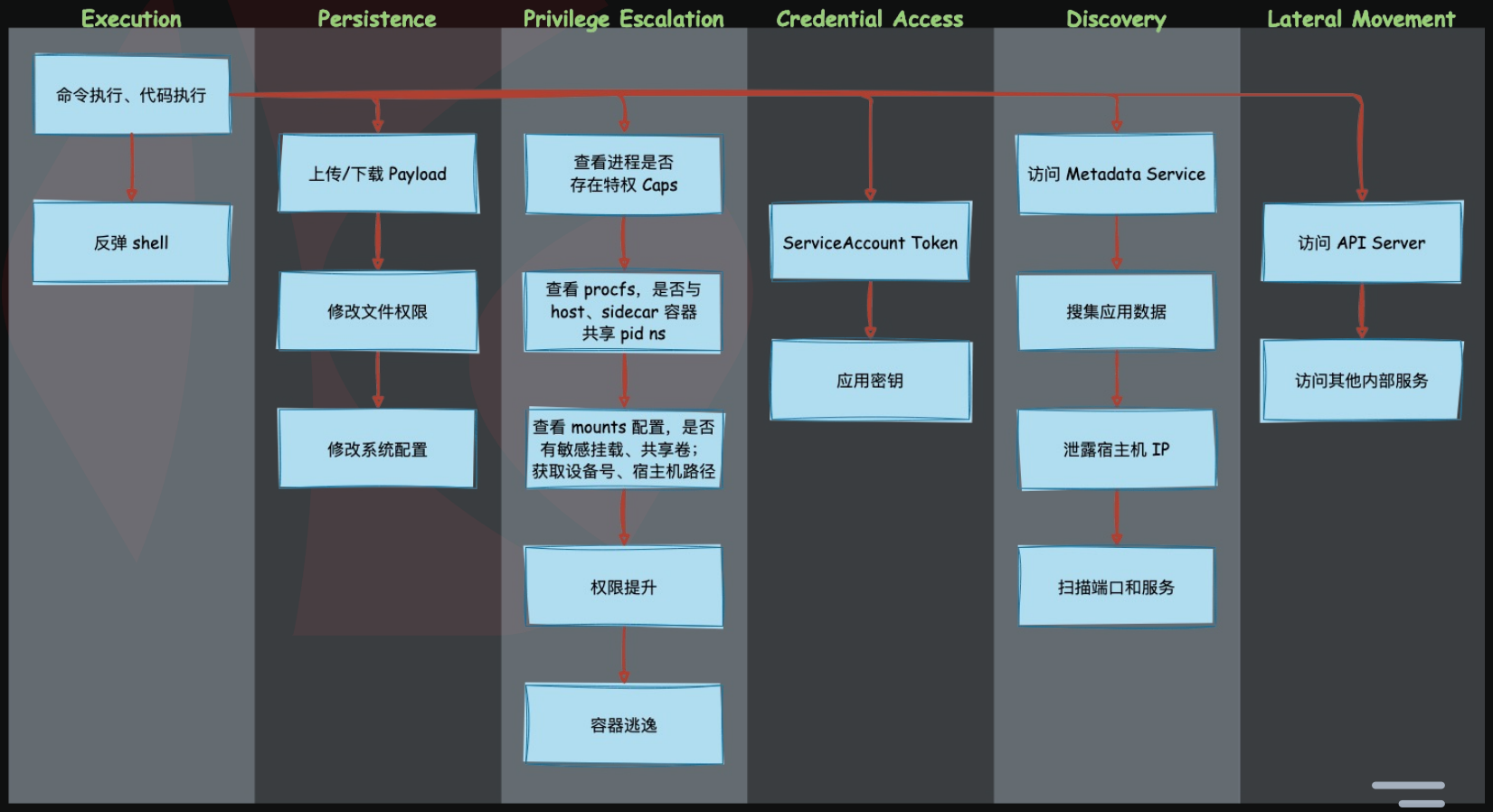
- 缓解方法：允许创建 User Namespace 但限制特权



防御渗透入侵

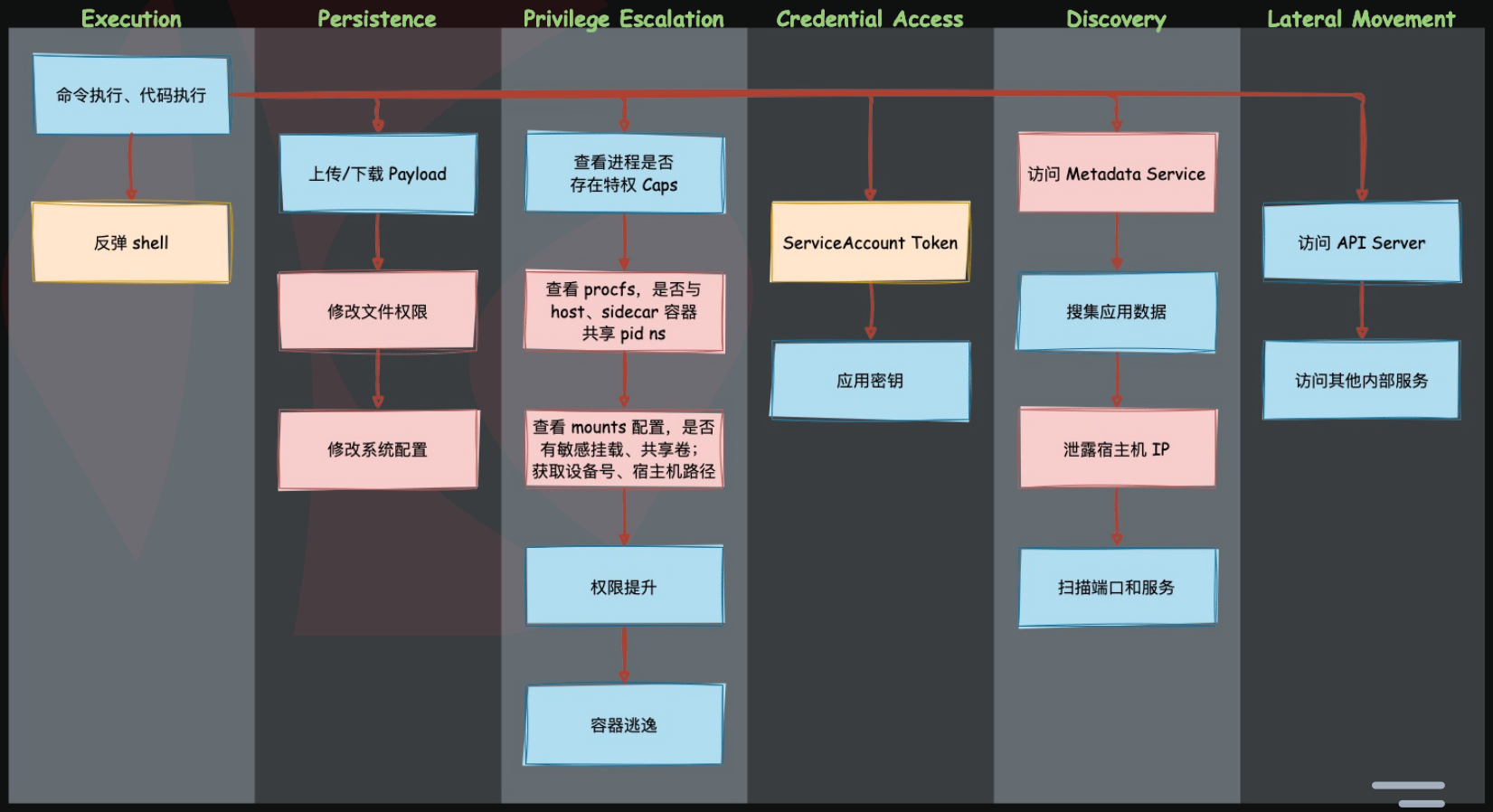
- 容器内应用的正常行为，记作 A
- 攻击者在容器内的渗透行为，记作 B
- 可尝试通过沙箱限制的行为，记作 C

$$C = B - (A \cap B)$$



防御渗透入侵

- vArmor 提供一系列针对渗透行为的内置策略
- 部分策略可能与业务行为冲突
- 部分策略则极少影响业务行为
- 挑战
  - 富容器、线上调试
  - 策略配置存在门槛



防御渗透入侵

以共享 pid ns 的容器场景为例

- 攻击者可通过 /proc/[PID]/root 访问 sidecar 容器的进程文件系统根目录，其权限受以下控制
  - PTRACE_MODE_READ_FSCREDS
  - DAC (Discretionary Access Control)

```
root@share-pid-ns-5ddb696869-xb9bx:/# ls /proc/*/root
ls: cannot access '/proc/1/root': Permission denied
/proc/15/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var

/proc/16/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  sidecar  srv  sys  tmp  usr  var

/proc/23/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  sidecar  srv  sys  tmp  usr  var

/proc/24/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var

/proc/8/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var

/proc/self/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var

/proc/thread-self/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var
root@share-pid-ns-5ddb696869-xb9bx:/#
```

开启防护前

Isolation between Linux containers

When designing services with multiple containers, it is crucial to be precise in determining how they should work together—if at all—and what the security implications of such interaction might be. In both ApsaraDB RDS and AnalyticDB our database container shared different Linux namespaces with other operational containers in the K8s pod. Specifically, they shared the PID namespace, which allowed our container to access other processes in the operational containers and the filesystem. This mistake was fatal: it enabled us to perform lateral movement to the operational containers and therefore escape our container!

#BrokenSesame

防御渗透入侵

以共享 pid ns 的容器场景为例

- 攻击者可通过 /proc/[PID]/root 访问 sidecar 容器的进程文件系统根目录，其权限受以下控制
  - PTRACE_MODE_READ_FSCREDS
  - DAC (Discretionary Access Control)

```
root@share-pid-ns-5ddb696869-xb9bx:/# ls /proc/*/root
ls: cannot access '/proc/1/root': Permission denied
/proc/15/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var

/proc/16/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  sidecar  srv  sys  tmp  usr  var

/proc/23/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  sidecar  srv  sys  tmp  usr  var

/proc/24/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var

/proc/8/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var

/proc/self/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var

/proc/thread-self/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  service  srv  sys  tmp  usr  var
root@share-pid-ns-5ddb696869-xb9bx:/#
```

开启防护前

```
root@share-pid-ns-5ddb696869-xb9bx:/# ls /proc/*/root
ls: cannot access '/proc/1/root': Permission denied
ls: cannot access '/proc/15/root': Permission denied
ls: cannot access '/proc/16/root': Permission denied
ls: cannot access '/proc/23/root': Permission denied
ls: cannot access '/proc/24/root': Permission denied
ls: cannot access '/proc/8/root': Permission denied
/proc/self/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc
 root  run  sbin  service  srv  sys  tmp  usr  var

/proc/thread-self/root:
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc
 root  run  sbin  service  srv  sys  tmp  usr  var
root@share-pid-ns-5ddb696869-xb9bx:/#
```

开启防护后

KCon

归源 · 智变

Returning to the source · Intelligent transformation

目录 / CONTENTS

现状与计划

- 支持的平台

强制访问控制器	必要条件	推荐
AppArmor	• Linux Kernel 4.15 及以上 • 系统启用了 AppArmor LSM	• GKE with Container-Optimized OS • AKS with Ubuntu 22.04 LTS • VKE with veLinux • Debian 10 及以上 • Ubuntu 18.04.0 LTS 及以上 • veLinux
BPF	• Linux Kernel 5.7 及以上 • containerd v1.6.0 及以上 • 系统启用了 BPF LSM	• EKS with Amazon Linux 2 • GKE with Container-Optimized OS • AKS with Ubuntu 22.04 LTS * • ACK with Alibaba Cloud Linux 3 * • OpenSUSE 15.4 及以上 * • Debian 11 及以上 * • Fedora 37 及以上 *(* 需要手动启用 BPF LSM)*

现状与计划

- e2e 测试、可观测性、安全性、兼容性提升
- BPF enforcer
  - 新增策略原语（mount、chmod、ptrace...）
  - seccomp 集成
  - ...
- 内置策略完善与丰富
- Policy Advisor

现状与计划

- vArmor 现已开源，并在持续迭代中，欢迎试用和参与社区建设 https://github.com/bytedance/vArmor
- 感谢 Elkeid Team 的全部现有和历史成员，特别感谢 Ping Shen @shenping-bd

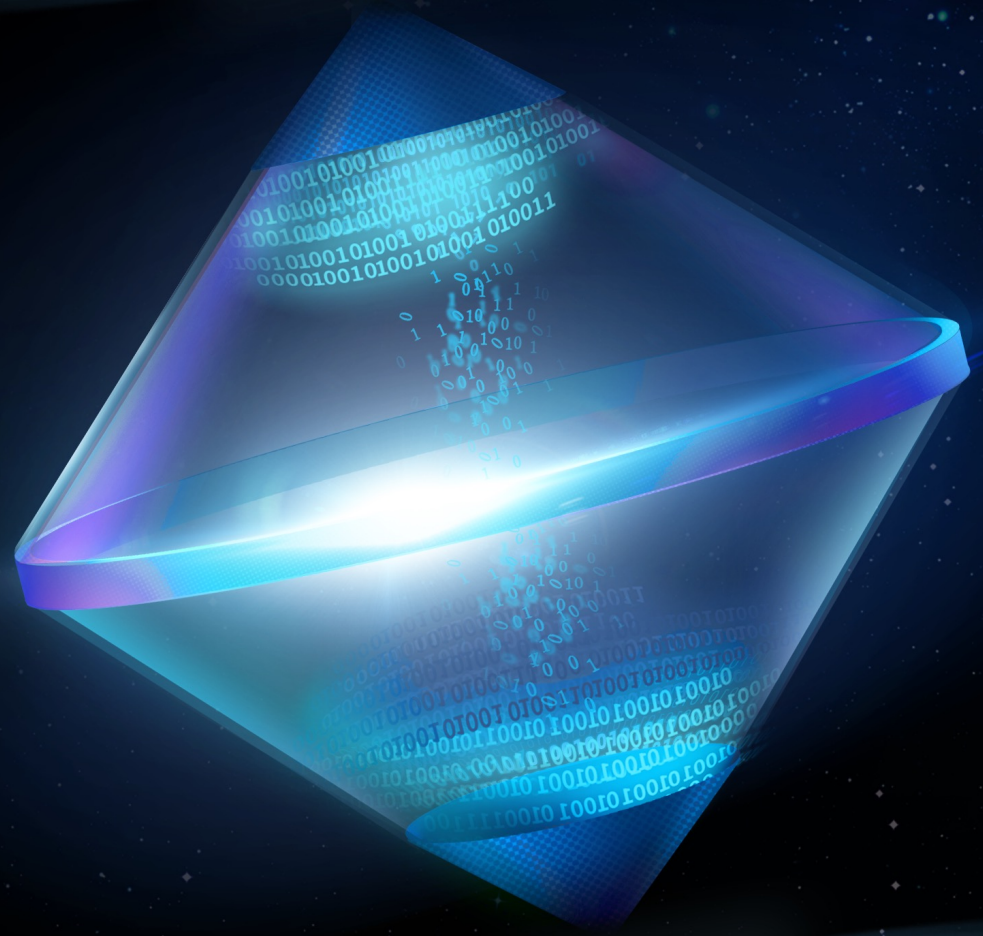
KCon

归源·智变

Returning to the source · Intelligent transformation

感谢您的观看！

THANK YOU FOR YOUR WATCHING



参考链接

- #BrokenSesame: Accidental 'write' permissions to private registry allowed potential RCE to Alibaba Cloud Database Services
- Tetragone: A Lesson in Security Fundamentals
- 10 Years of Linux Security