# 轻量无侵入

## vArmor 赋能云原生容器多场景安全防护

韦伟，李昌昊

字节跳动

2025/11/15

KCD HANGZHOU

CHINA OpenInfra Days

# CONTENT
# 目录



# 01 为什么要做 vArmor

# 云原生环境的容器安全挑战

- runc 容器隔离性不足，相关漏洞频发，但仍被广泛使用
- Kata, gVisor 等安全容器提供了解决方案，但并非普遍适用
- 难以防御攻击者从容器内发起的横向移动攻击

| 技术 | 优势 | 挑战 |
| --- | --- | --- |
| Least Privilege | 容易配置，可有效增加权限提升的难度与成本；<br>禁用特权容器、移除不需要的 capability、以非 root 用户运行、开启 NoNewPrivs、开启只读文件系统... | 一些机制需要应用进行适配；<br>并非适用于所有类型的工作负载（例如系统和基础设施相关）； |
| Rootless Container | 可以缓解很多漏洞；<br>read/write other users' files,<br>modify the kernel, ARP/DNS spoofing... | 出于兼容性考虑，Kubernetes 默认不启用；<br>允许非 root 用户创建用户命名空间会增加内核的攻击面； |
| Seccomp | Kubernetes v1.3 起开始支持，并在 v1.19 GA ；<br>可以过滤不必要的 syscall，阻止部分攻击行为；<br>可以基于 syscall 的特定参数进行拦截，实现细粒度访问控制； | 出于兼容性考虑，Kubernetes 默认不启用，默认策略相对宽泛；<br>一旦生效不可修改； |
| AppArmor | Kubernetes v1.4 起开始支持，并在 v1.30 GA；<br>可以用于实现细粒度的访问控制； | 默认策略比较宽泛；<br>自定义策略需要 AppArmor 专家知识； |
| SELinux | Kubernetes v1.4 起开始支持；<br>可以用于实现细粒度的访问控制； | 在某些宿主目录挂载、持久卷挂载场景下存在兼容性问题；<br>自定义策略需要 SELinux 专家知识； |

| 技术 | 优势 | 挑战 |
| --- | --- | --- |
| Least Privilege | 容易配置，可有效增加权限提升的难度与成本；<br>禁用特权容器、移除不需要的 capability、以非 root 用户运行、开启 NoNewPrivs、开启只读文件系统... | 一些机制需要应用进行适配；<br>并非适用于所有类型的工作负载（例如系统和基础设施相关）； |
| Rootless Container | 可以缓解很多漏洞；<br>read/write other users' files,<br>modify the kernel, ARP/DNS spoofing... | 出于兼容性考虑，Kubernetes 默认不启用；<br>允许非 root 用户创建用户命名空间会增加内核的攻击面； |
| Seccomp | Kubernetes v1.3 起开始支持，并在 v1.19 GA；<br>可以过滤不必要的 syscall，阻止部分攻击行为；<br>可以基于 syscall 的特定参数进行拦截，实现细粒度访问控制； | 出于兼容性考虑，Kubernetes 默认不启用，默认策略相对宽泛；<br>一旦生效不可修改； |
| AppArmor | Kubernetes v1.4 起开始支持，并在 v1.30 GA；<br>可以用于实现细粒度的访问控制； | 默认策略比较宽泛；<br>自定义策略需要 AppArmor 专家知识； |
| SELinux | Kubernetes v1.4 起开始支持；<br>可以用于实现细粒度的访问控制； | 在某些宿主目录挂载、持久卷挂载场景下存在兼容性问题；<br>自定义策略需要 SELinux 专家知识； |

# 容器安全加固现状

- 默认策略过于宽泛，无法防御某些漏洞和横向移动攻击

```
42  const defaultTemplate = `
52    network,
53    capability,
54    file,
55    umount,
56    # Host (privileged) processes may send signals to container processes.
57    signal (receive) peer=unconfined,
58    # runc may send signals to container processes.
59    signal (receive) peer=runc,
60    # crun may send signals to container processes.
61    signal (receive) peer=crun,
62    # Manager may send signals to container processes.
63    signal (receive) peer={{.DaemonProfile}},
64    # Container processes may send signals amongst themselves.
65    signal (send,receive) peer={{.Name}},
66  {{if .RootlessKit}}
67    # https://github.com/containerd/nerdctl/issues/2730
68    signal (receive) peer={{.RootlessKit}},
69  {{end}}
70
71    deny @{PROC}/* w,   # deny write for all files directly in /proc (not in a subdir)
72    # deny write to files not in /proc/<number>/** or /proc/sys/**
73    deny @{PROC}/{[^1-9],[^1-9][^0-9],[^1-9s][^0-9y][^0-9s],[^1-9][^0-9][^0-9][^0-9]*}/** w,
74    deny @{PROC}/sys/[^k]** w,  # deny /proc/sys except /proc/sys/k* (effectively /proc/sys/kernel)
75    deny @{PROC}/sys/kernel/{?,??,[^s][^h][^m]**} w,  # deny everything except shm* in /proc/sys/kernel/
76    deny @{PROC}/sysrq-trigger rwklx,
77    deny @{PROC}/mem rwklx,
78    deny @{PROC}/kmem rwklx,
79    deny @{PROC}/kcore rwklx,
80
81    deny mount,
```

```
56  // DefaultProfile defines the allowed syscalls for the default seccomp profile.
57  func DefaultProfile(sp *specs.Spec) *specs.LinuxSeccomp {
58      nosys := uint(unix.ENOSYS)
59      syscalls := []specs.LinuxSyscall{
60          {
61              Names: []string{
62                  "accept",
63                  "accept4",
64                  "access",
65                  "adjtimex",
66                  "alarm",
67                  "bind",
68                  "brk",
69                  "cachestat", // kernel v6.5, libseccomp v2.5.5
70                  "capget",
71                  "capset",
72                  "chdir",
73                  "chmod",
74                  "chown",
75                  "chown32",
76                  "clock_adjtime",
77                  "clock_adjtime64",
78                  "clock_getres",
79                  "clock_getres_time64",
80                  "clock_gettime",
81                  "clock_gettime64",
```

# 容器安全加固现状

- 默认策略过于宽泛，无法防御某些漏洞和横向移动攻击
- 基础设施相关漏洞修复周期较长
- 管理和应用安全策略过程中存在诸多挑战
- 难以为大规模、快速变化的应用构建“默认拒绝”的安全策略
- 一些环境不支持 AppArmor LSM（例如 EKS、ACS）

# 容器安全加固现状

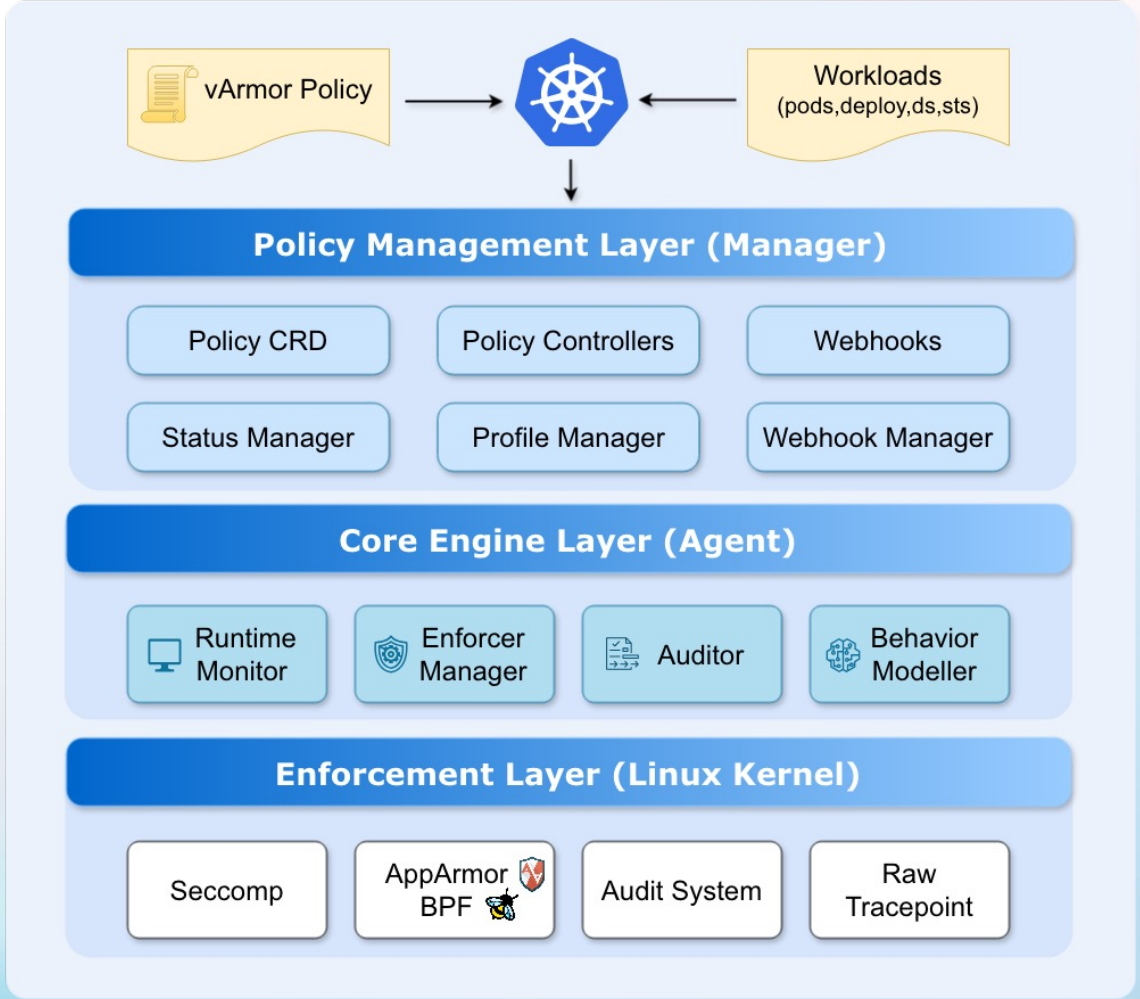
- 默认策略过于宽泛，无法防御某些漏洞和横向移动攻击
- 基础设施相关漏洞修复周期较长
- 管理和应用安全策略过程中存在诸多挑战
- 难以为大规模、快速变化的应用构建“默认拒绝”的安全策略
- 一些环境不支持 AppArmor LSM（例如 EKS、ACS）

vArmor

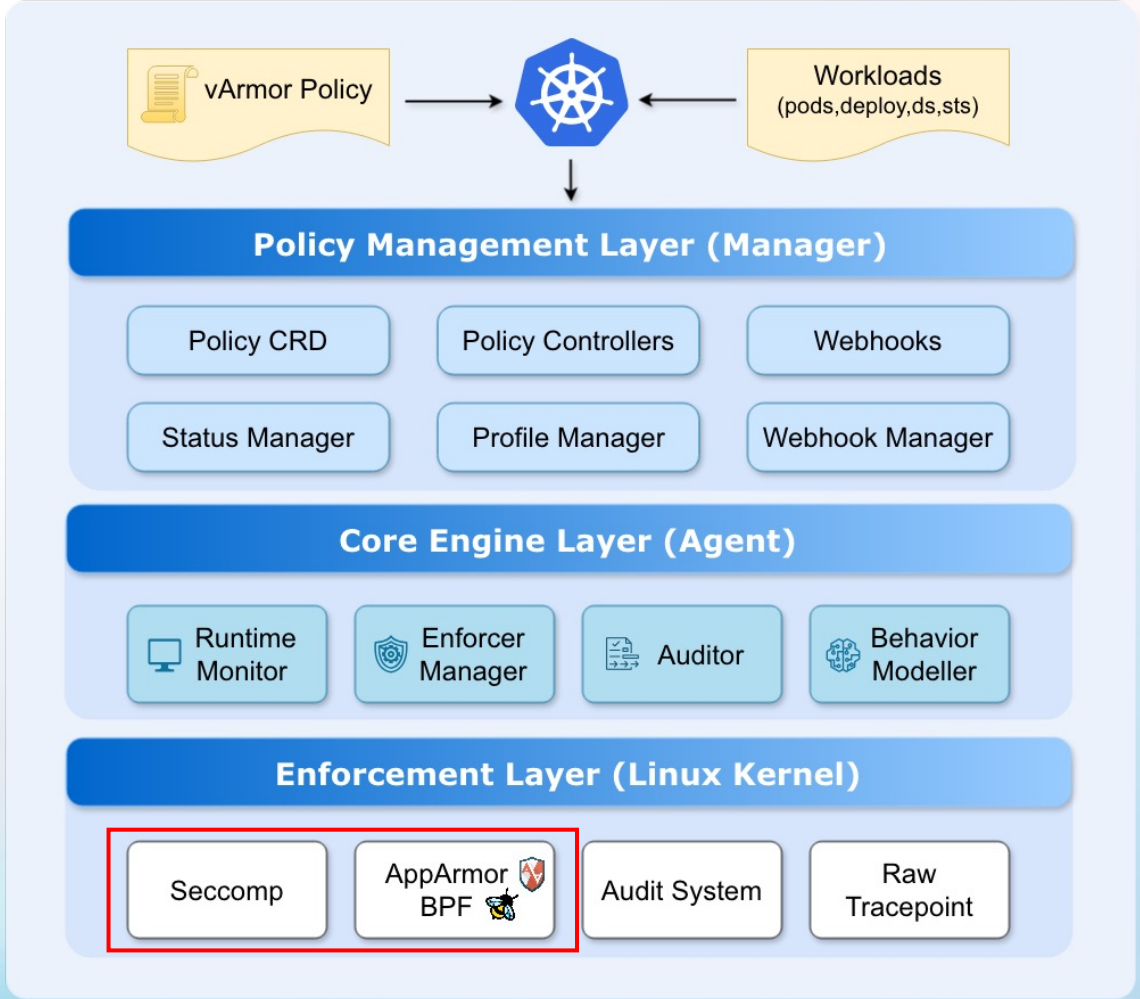
专为解决这些挑战打造

# 02 核心概念与原理

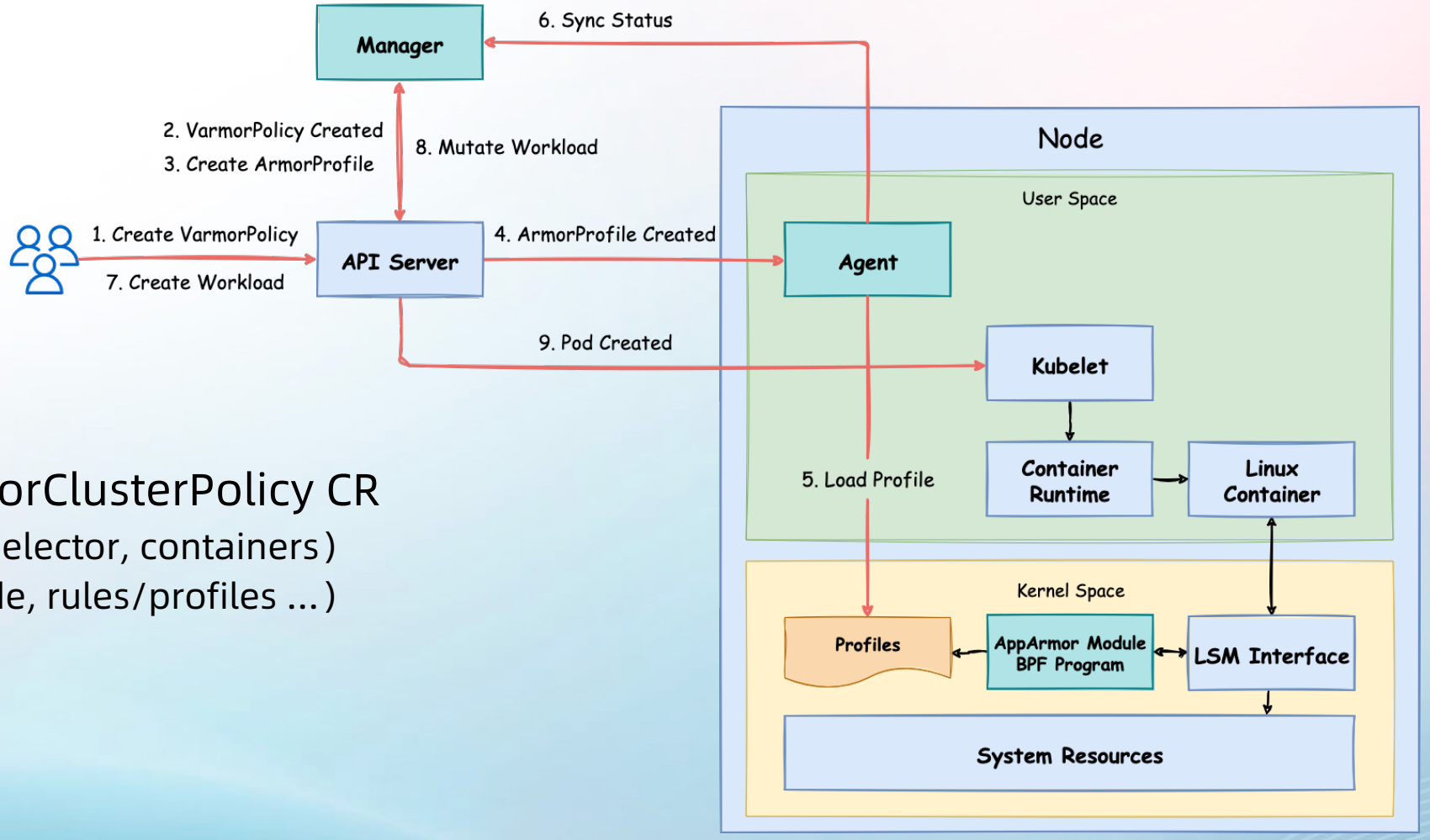
# vArmor 整体架构


vArmor Policy
Workloads
(pods,deploy,ds,sts)
Policy Management Layer (Manager)
Policy CRD
Policy Controllers
Webhooks
Status Manager
Profile Manager
Webhook Manager
Core Engine Layer (Agent)
Runtime Monitor
Enforcer Manager
Auditor
Behavior Modeller
Enforcement Layer (Linux Kernel)
Seccomp
AppArmor BPF
Audit System
Raw Tracepoint

# vArmor 整体架构


vArmor Policy
Workloads
(pods,deploy,ds,sts)
Policy Management Layer (Manager)
Policy CRD
Policy Controllers
Webhooks
Status Manager
Profile Manager
Webhook Manager
Core Engine Layer (Agent)
Runtime Monitor
Enforcer Manager
Auditor
Behavior Modeller
Enforcement Layer (Linux Kernel)
Seccomp
AppArmor BPF
Audit System
Raw Tracepoint

# vArmor 基本原理




- VarmorPolicy & VarmorClusterPolicy CR
  - Target（kind, name, selector, containers）
  - Policy（enforcer, mode, rules/profiles ...）
- ArmorProfile CR
- Policy as Resource

# AppArmor & Seccomp Enforcers

- Manager 为每个 Policy 维护对应的 ArmorProfile 对象
- Agent 响应 ArmorProfile 对象，管理 AppArmor & Seccomp Profiles
- Manager 为 Workloads 配置 Annotations 和 SecurityContext

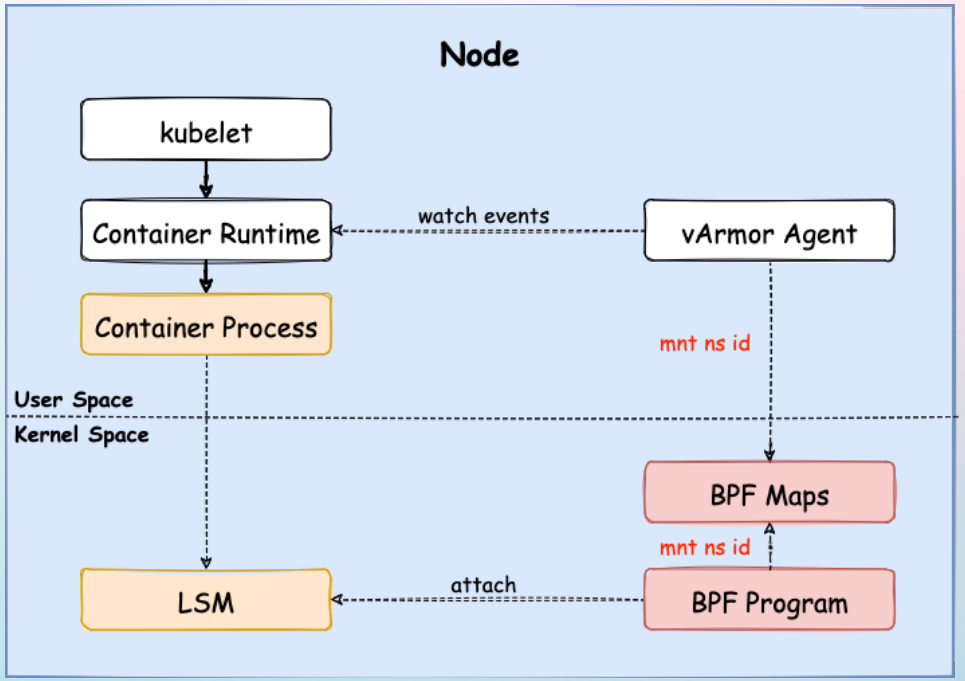
```yaml
apiVersion: crd.varmor.org/v1beta1
kind: VarmorPolicy
metadata:
  name: demo
spec:
  target:
    kind: Pod
    selector:
      matchLabels:
        app: some
    containers:
    - c1
  policy:
    enforcer: AppArmorSeccomp
    mode: EnhanceProtect
    enhanceProtect:
      hardeningRules:
      - disable-cap-privileged
      attackProtectionRules:
      - rules:
        - disable-chmod-x-bit
```

```yaml
apiVersion: crd.varmor.org/v1beta1
kind: ArmorProfile
metadata:
  name: varmor-default-demo
spec:
  target:
    kind: Pod
    selector:
      matchLabels:
        app: some
    containers:
    - c1
  profile:
    enforcer: AppArmorSeccomp
    appArmor: ${PROFILE}
    seccomp: ${PROFILE}
    ...
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test
  lables:
    app: some
  annotations:
    container.apparmor.security.beta.varmor.org/c1: localhost/varmor-default-demo
    container.seccomp.security.beta.varmor.org/c1: localhost/varmor-default-demo
spec:
  containers:
  - securityContext:
      seccompProfile:
        localhostProfile: varmor-default-demo
        type: Localhost
  - securityContext:
      seccompProfile:
        localhostProfile: varmor-default-demo
        type: Localhost
```

# BPF Enforcer

- 在 BPF 程序中实现强制访问控制的策略原语
  (file, execution, network, capability, ptrace, mount ...)
- 借助 BPF maps 管理容器的 BPF Profiles
- 基于事件机制监控容器创建与删除
- 通过 mnt ns id 标识容器进程并实施访问控制

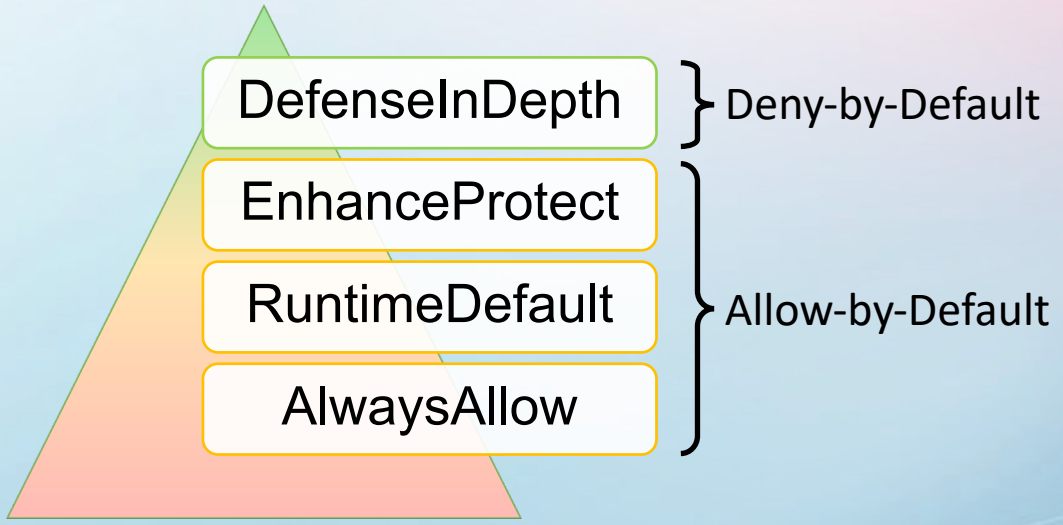


# 03 关键特性与应用场景

# 灵活易用的策略管理

- 支持多种模式按需使用和动态切换
- AppArmor、BPF、Seccomp 可按需混合使用
- 切换模式、更新规则、新增 Enforcer，无需重启工作负载（Seccomp 除外）

| Policy Mode | AppArmor | BPF | Seccomp |
|---|---|---|---|
| DefenseInDepth | ✓ | 🏗 | ✓ |
| EnhanceProtect | ✓ | ✓ | ✓ |
| RuntimeDefault | ✓ | ✓ | ✓ |
| AlwaysAllow | ✓ | ✓ | ✓ |
| BehaviorModeling | ✓ | ✓ | ✓ |



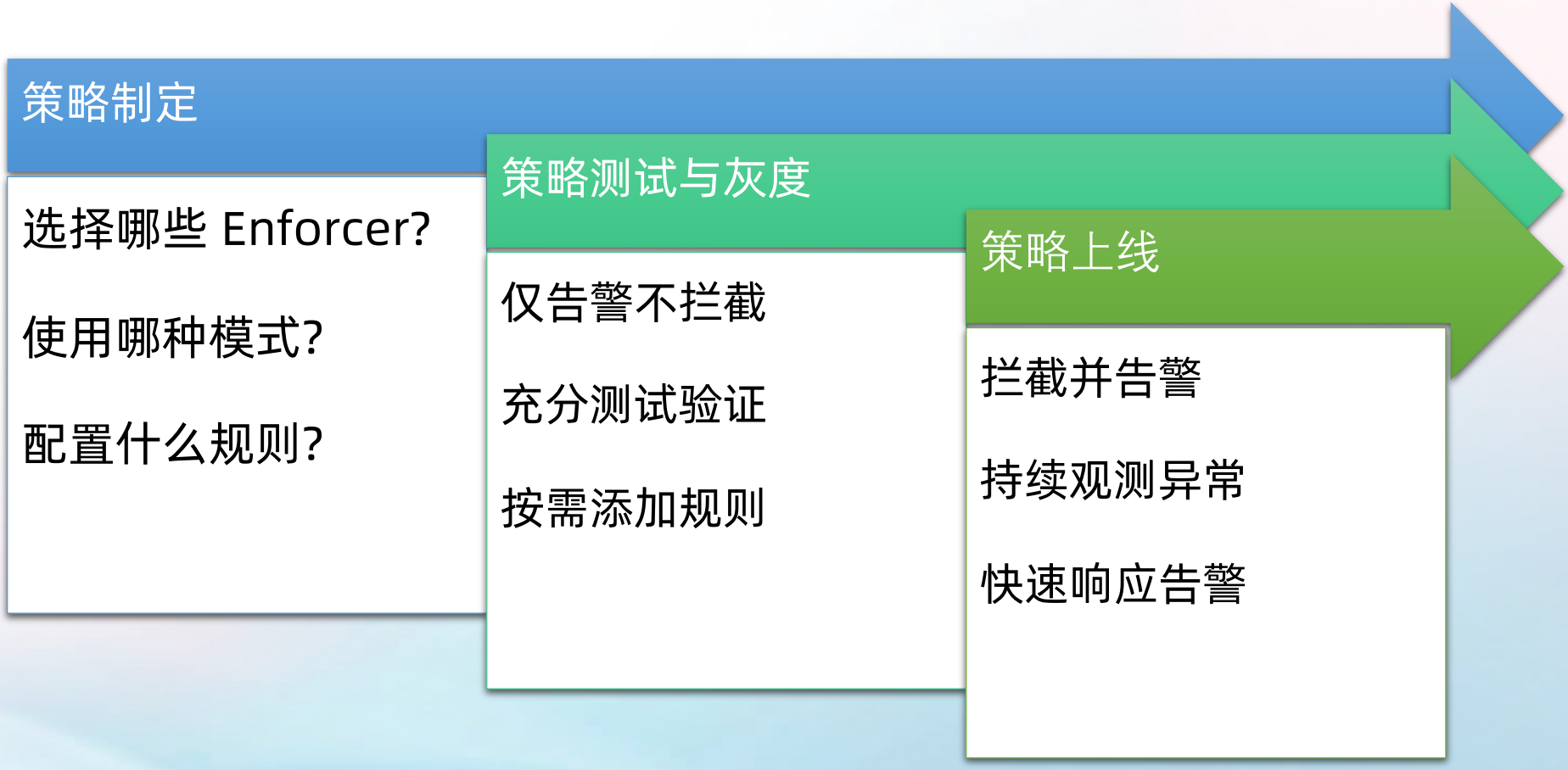
# 开箱即用的内置规则

- vArmor 基于不同 Enforcer 的策略原语实现了大量内置规则
- EnhanceProtect 模式的策略可使用内置规则进行加固，无需专家知识

| Category | Description |
| --- | --- |
| Hardening | Block common escape vectors for privileged containers.<br>Disable capabilities.<br>Block certain kernel vulnerability exploitation vectors. |
| Attack Protection | Mitigate container information leakage.<br>Prohibit execution of sensitive actions. |
| Vulnerability Mitigation | cgroups-lxcfs-escape mitigation, runc override mitigation, dirty-pipe mitigation ... |

```yaml
policy:
  enforcer: AppArmorBPFSeccomp
  mode: EnhanceProtect
  enhanceProtect:
    hardeningRules:
    - disable-cap-privileged
    - disallow-write-core-pattern
    attackProtectionRules:
    - rules:
      - disable-chmod-x-bit
    - rules:
      - mitigate-sa-leak
      targets:
      - /bin/sh
    vulMitigationRules:
    - ingress-nightmare-mitigation
```

# 渐进稳妥的上线流程


策略制定
选择哪些 Enforcer?
使用哪种模式?
配置什么规则?
策略测试与灰度
仅告警不拦截
充分测试验证
按需添加规则
策略上线
拦截并告警
持续观测异常
快速响应告警

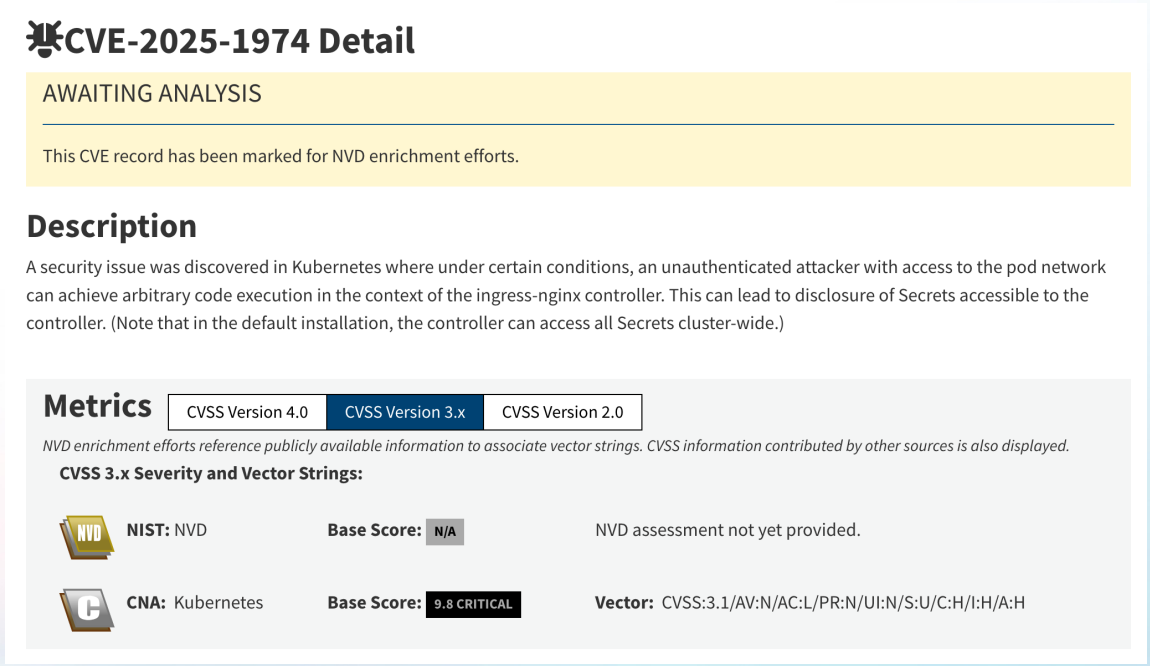
# 按需扩展的自定义接口

- 支持根据语法自定义规则
  - AppArmor
  - Seccomp
  - BPF
- 满足专业用户定制需求

```yaml
enhanceProtect:
  appArmorRawRules:
  - rules: |
      audit deny /etc/shadow rw,
      audit deny /etc/hosts rw,
    targets:
    - /bin/busybox
 syscallRawRules:
  - names:
    - fchmodat
    action: SCMP_ACT_ERRNO
    args:
    - index: 2
      value: 0x40     # S_IXUSR
      valueTwo: 0x40
      op: SCMP_CMP_MASKED_EQ
  bpfRawRules:
    network:
      egress:
        toServices:
        - serviceSelector:
            matchLabels:
              app: nginx
          qualifiers:
          - deny
          - audit
```

# 按需扩展的自定义接口

- 高危漏洞应急响应



**CVE-2025-1974 Detail**

AWAITING ANALYSIS

This CVE record has been marked for NVD enrichment efforts.

**Description**

A security issue was discovered in Kubernetes where under certain conditions, an unauthenticated attacker with access to the pod network can achieve arbitrary code execution in the context of the ingress-nginx controller. This can lead to disclosure of Secrets accessible to the controller. (Note that in the default installation, the controller can access all Secrets cluster-wide.)

**Metrics** CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

*NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.*

**CVSS 3.x Severity and Vector Strings:**

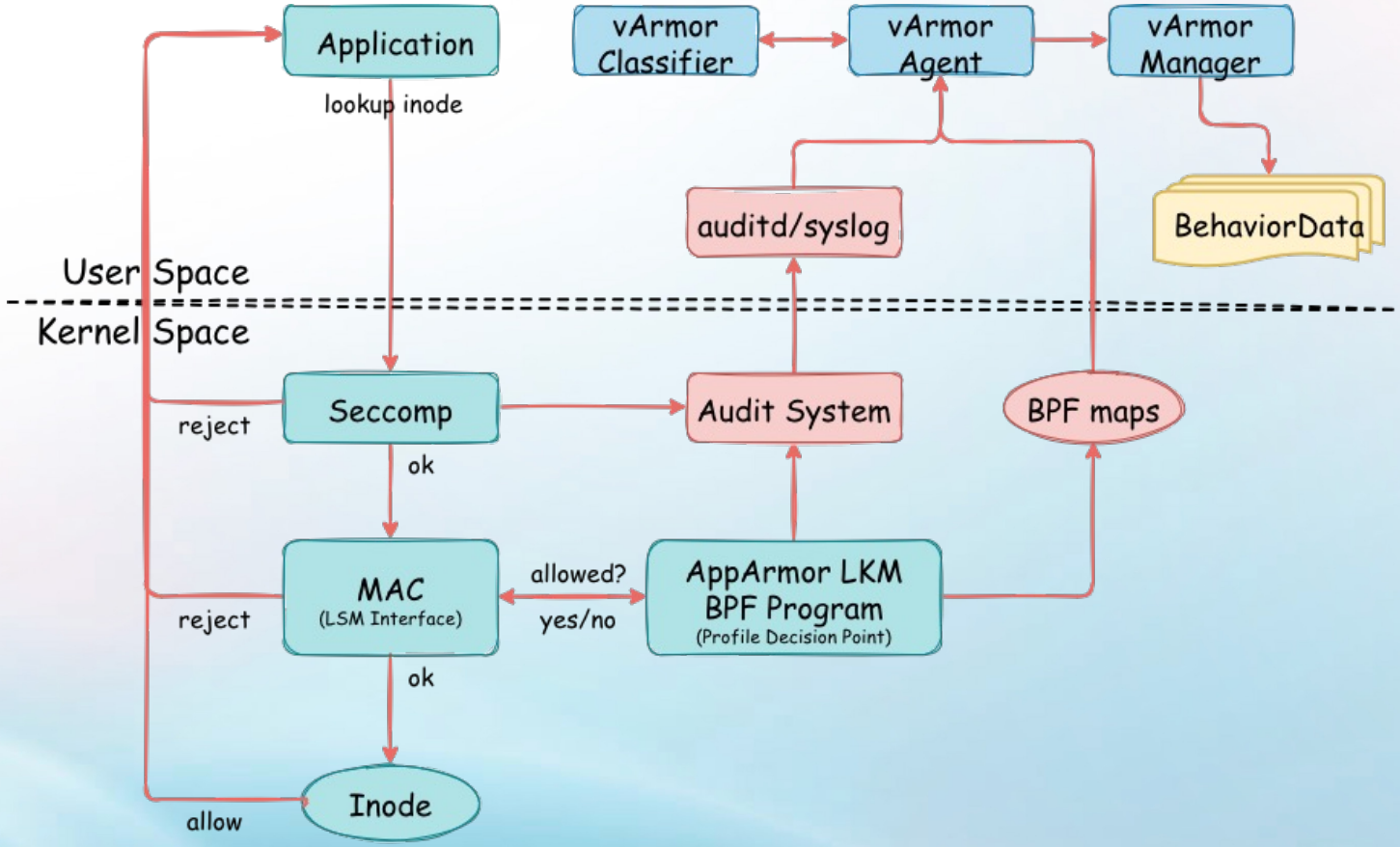
| | | | |
|---|---|---|---|
| NVD | **NIST:** NVD | **Base Score:** N/A | NVD assessment not yet provided. |
| C | **CNA:** Kubernetes | **Base Score:** 9.8 CRITICAL | **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H |

```yaml
enhanceProtect:
  bpfRawRules:
    network:
      egress:
        toServices:
        - namespace: kube-system
          name: ingress-nginx-controller-admission
          qualifiers:
          - deny
          - audit
        - namespace: ingress-nginx
          name: ingress-nginx-controller-admission
          qualifiers:
          - deny
          - audit
```

在修复前临时缓解 IngressNightmare 漏洞

# 多场景赋能的行为建模

- 借助 Audit System & eBPF 技术采集并汇总工作负载的行为数据



# 多场景赋能的行为建模

- 行为数据辅助创建 EnhanceProtect 策略 (筛选内置规则)
- 行为数据辅助构建 Deny-by-Default 策略 (进行深度防御)
- 行为数据辅助特权容器的降权 (推动权限最小化)

相关文档和案例请参见 vArmor 官方文档与博客

# 04 总结与未来展望

# 容器安全加固现状

- ✅ 默认策略过于宽泛，无法防御某些漏洞和横向移动攻击
  —— EnhanceProtect Mode
- ✅ 难以为大规模、快速变化的应用构建“默认拒绝”的安全策略
  —— DefenseInDepth & BehaviorModeling Mode
- ✅ 管理和应用安全策略过程中存在诸多挑战
  —— 以云原生的方式管理安全策略
- ✅ 一些环境不支持 AppArmor LSM（EKS、ACK）
  —— BPF Enforcer
- ✅ 基础设施相关漏洞修复周期较长
  —— 针对漏洞更新自定义规则进行缓解

# 使用 vArmor 进行容器安全加固的现状

## 🤦 新的门槛与挑战

- 组件的安装、使用、运维
- 策略的制定、测试、优化、运维
- 违规事件的运营

## 🎯 新的解决方案–– AI 赋能 & 产品化

- 字节跳动火山引擎云安全中心将上线 vArmor
- AI Agent：用于策略生成、知识问答，以及自动化加固

# 赋能组织轻松落地容器安全

- 🌐 为社区提供更多元的技术选择
- ⚖️ 平衡安全加固与业务发展的需求
- 🚧 降低成熟加固技术的使用门槛
- 💰 将不可控风险转化为可控成本

vArmor

Thanks.

www.varmor.org