22 ОКТЯБРЯ | 10:00-22:00

CONNECT, 5-Й ДОНСКОЙ ПР., 17, МОСКВА

Изолируй с умом: уроки unikernels

Сергей Канибор — Luntry, R&D / Container Security

KUBER CONF от АОТ

whoami

> Настоящая защита – это не запреты, а правильно расставленные ограничения



R&D / Container Security в Luntry

Более 5 лет опыта в ИБ

Специализация - безопасность контейнеров и Kubernetes

Редактор ТГ-канала k8s (in) security

Багхантер

- Участник BI.ZONE Bug Bounty, Yandex Bug Bounty и Standoff
- Автор зарегистрированных CVE и BDU

Спикер

VK Kubernetes
DevOpsConf
OFFZONE
БеКон
PHDays
DevOops

LUNTRY

Agenda

01 Unikernel и изоляция недоверенного кода

02 urunc: архитектура и запуск в Kubernetes

03 Threat model и две critical-уязвимости

LUNTRY

LUNTRY

AI-агенты
и недоверенный код

Эпоха недоверенного кода

LUNTRY

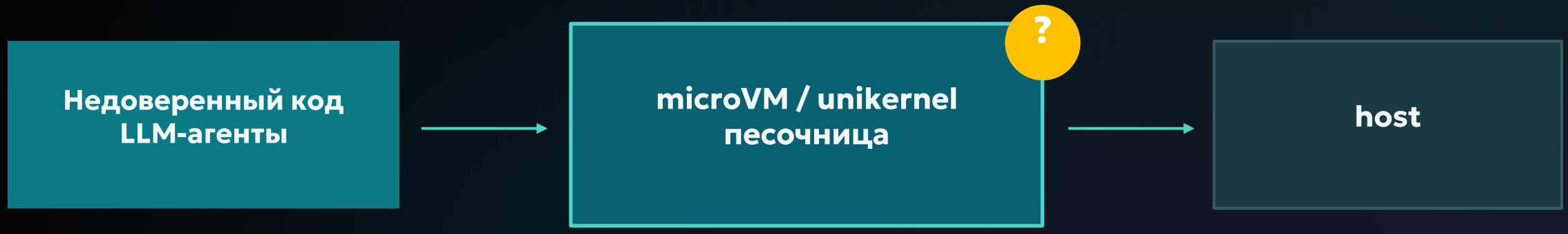
LLM генерируют код, который мы запускаем в проде

AI-агенты выполняют действия автономно

Код и команды приходят извне, часто без ревью

Доверять нельзя — а запускать надо

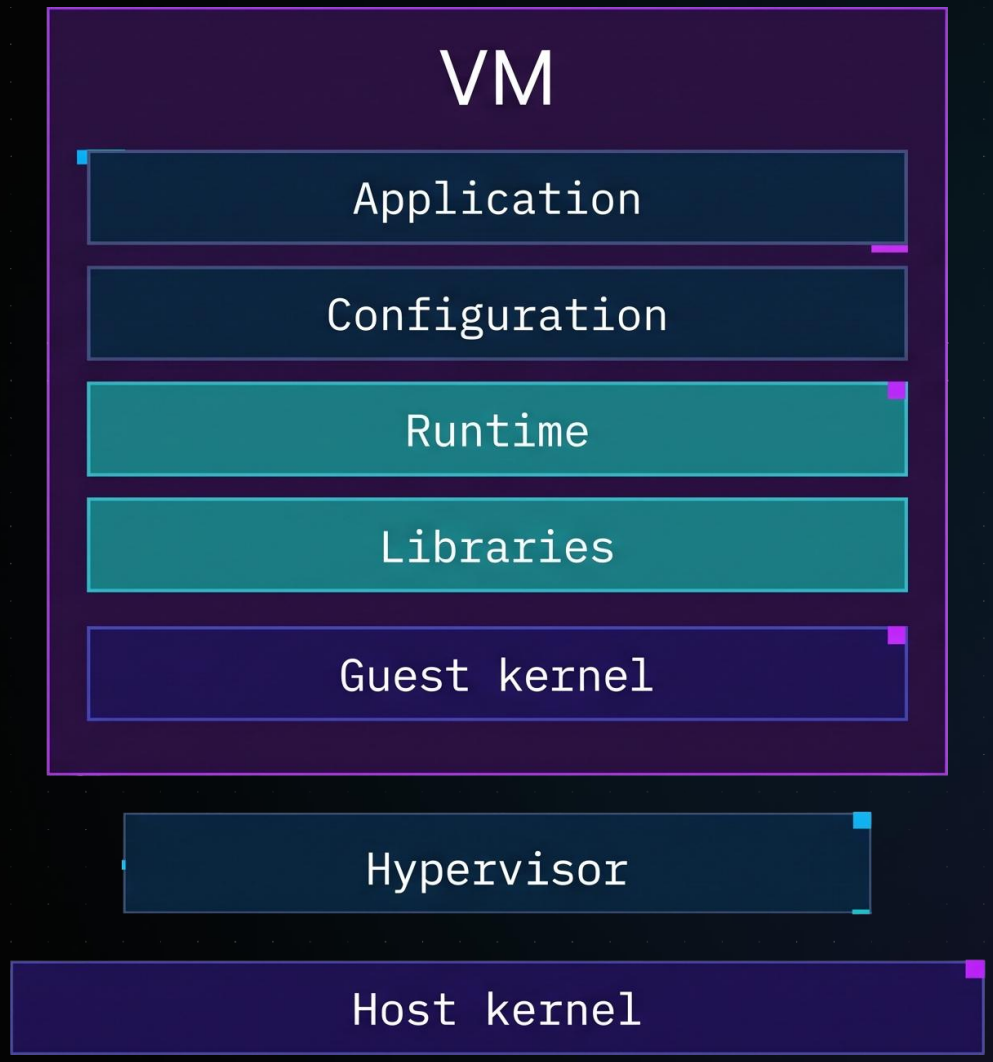
Соблазн: просто всё изолировать



какие гарантии это реально даёт?

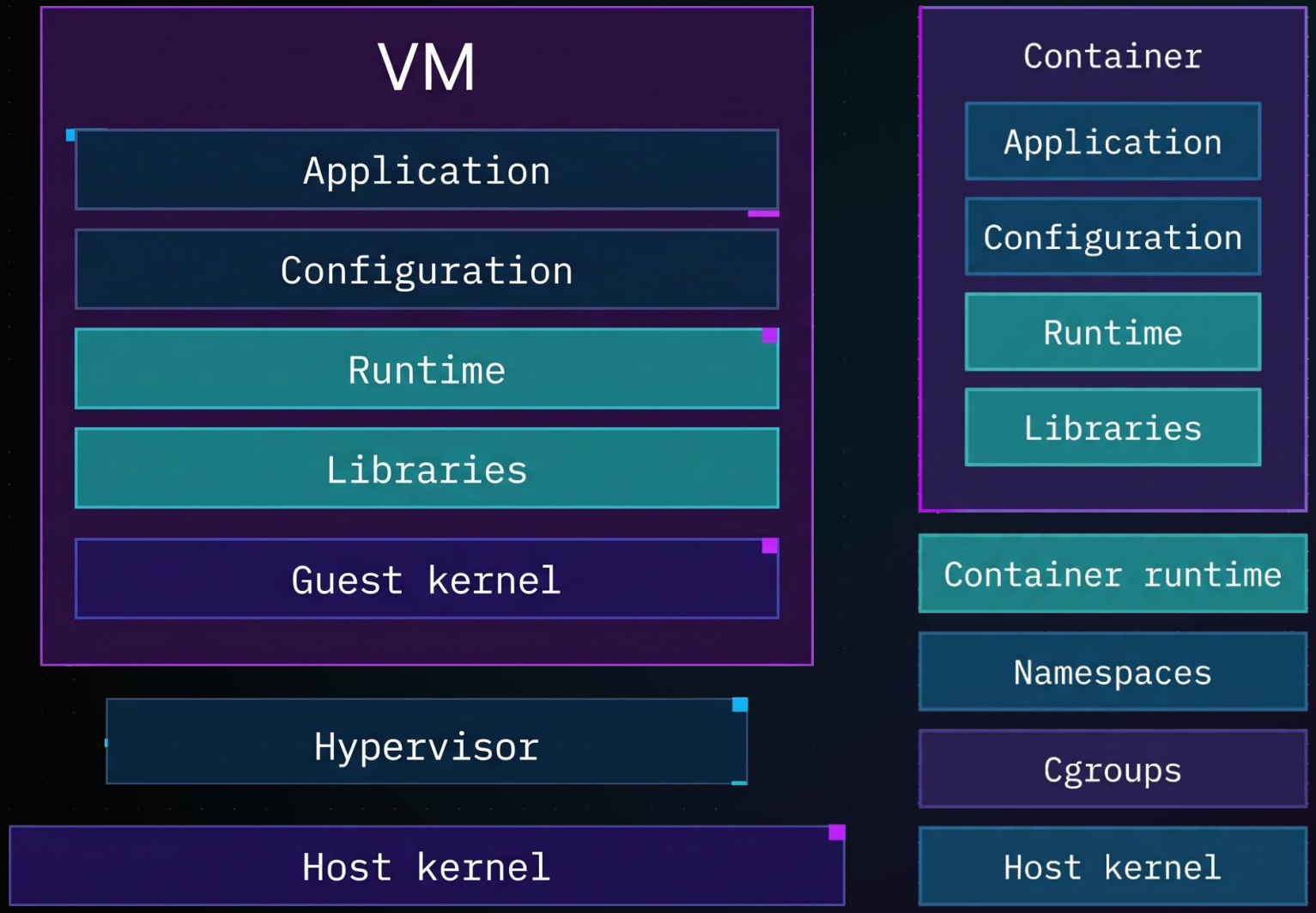
- «Запустим недоверенное в жёсткой песочнице»
- Для большинства microVM / unikernel — чёрный ящик

Парадокс VM и контейнеров



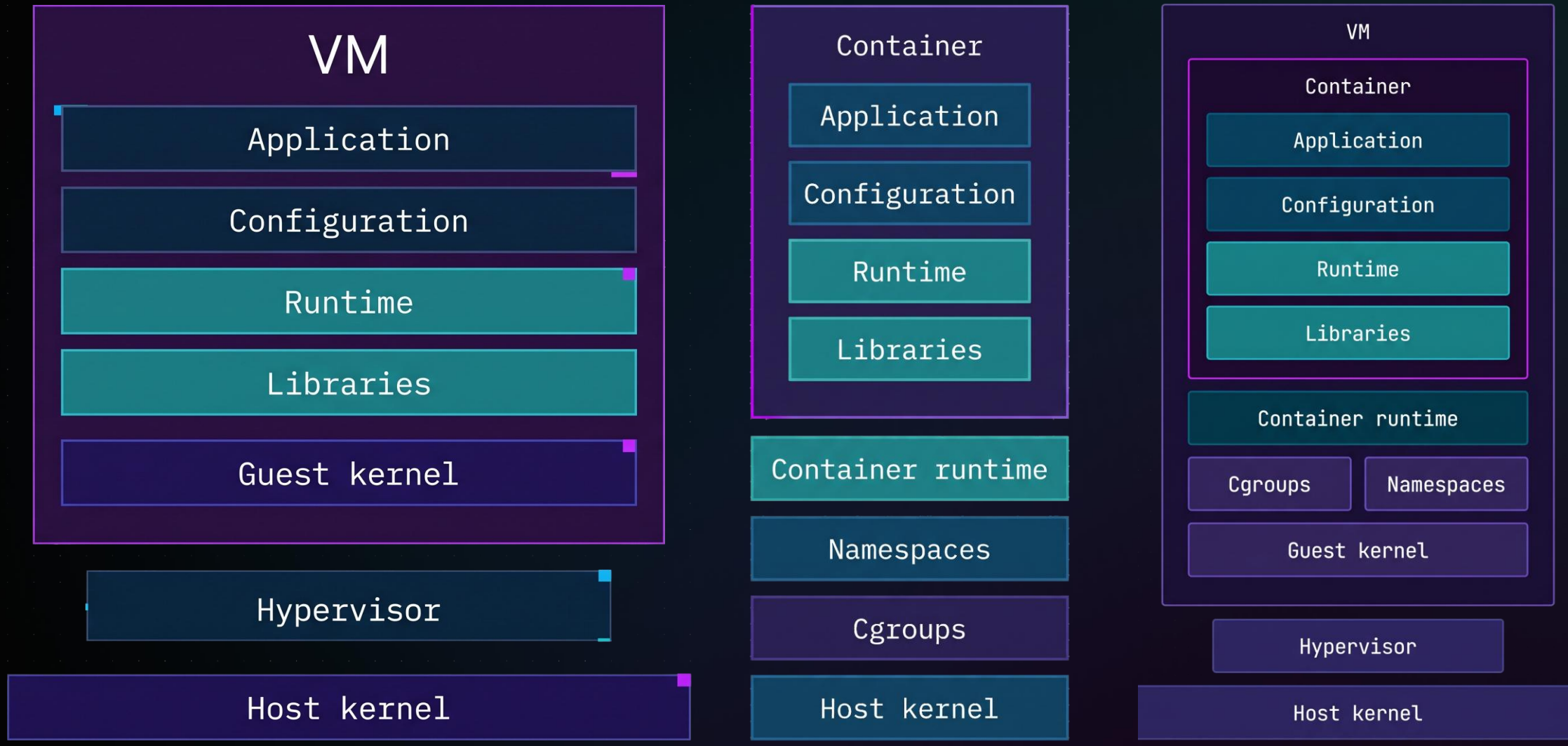
LUNTRY

Парадокс VM и контейнеров



LUNTRY

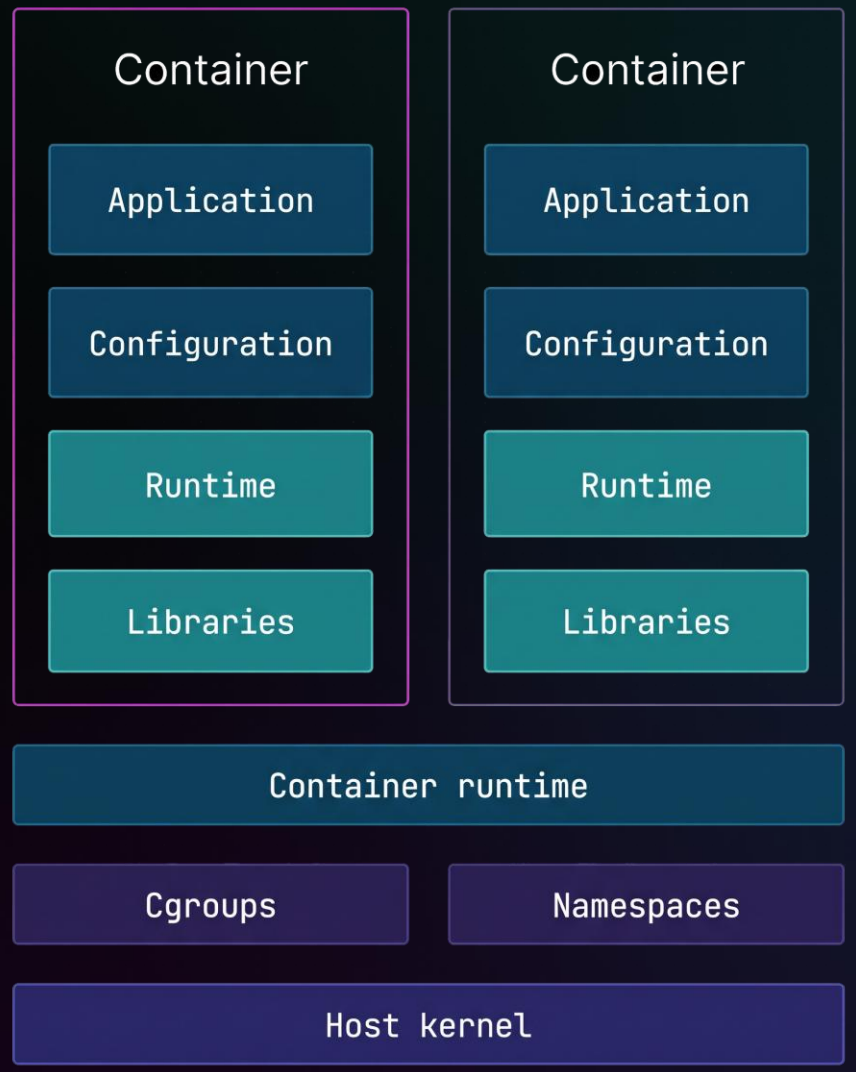
Парадокс VM и контейнеров



LUNTRY

Контейнеры доминируют

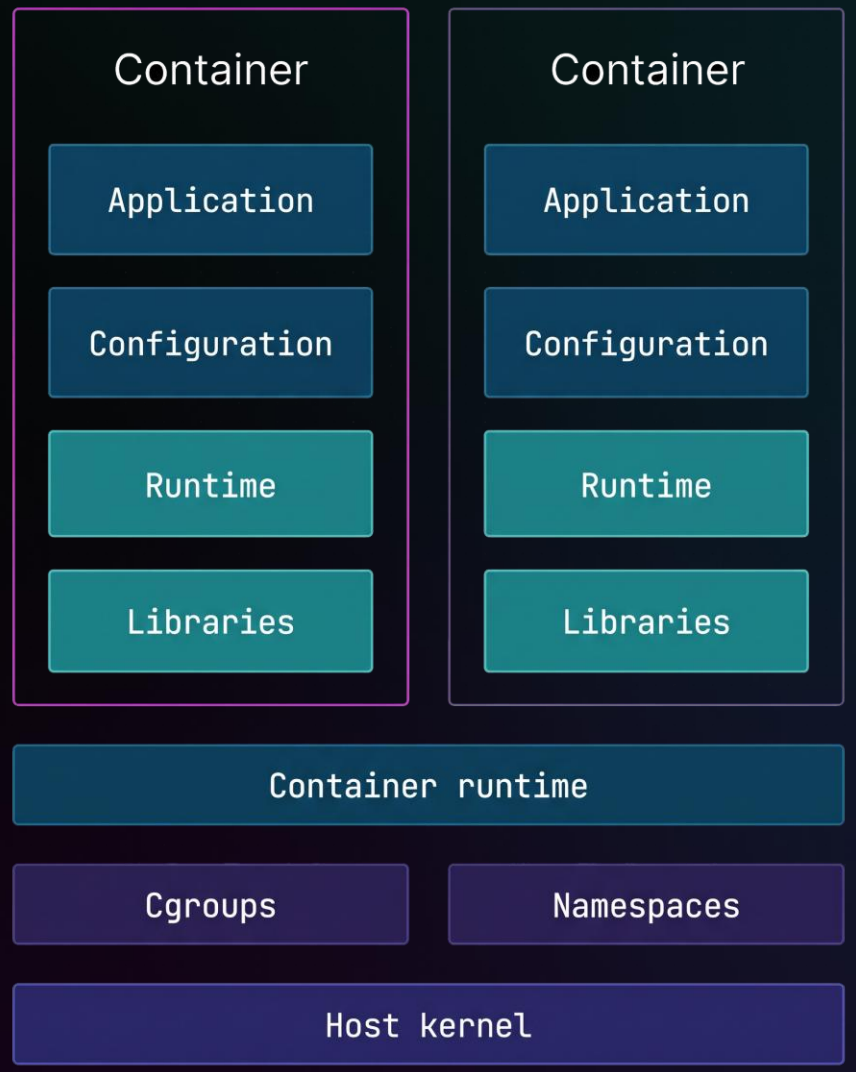
- Де-факто стандарт в разработке и деплое
  - Легковесные
  - Быстрое время спавна
  - Портативные
  - Удобные в использовании
  - Масштабируемые
- Развитая экосистема с множеством инструментов и возможностей
- Но...



LUNTRY

У контейнеров есть существенный недостаток

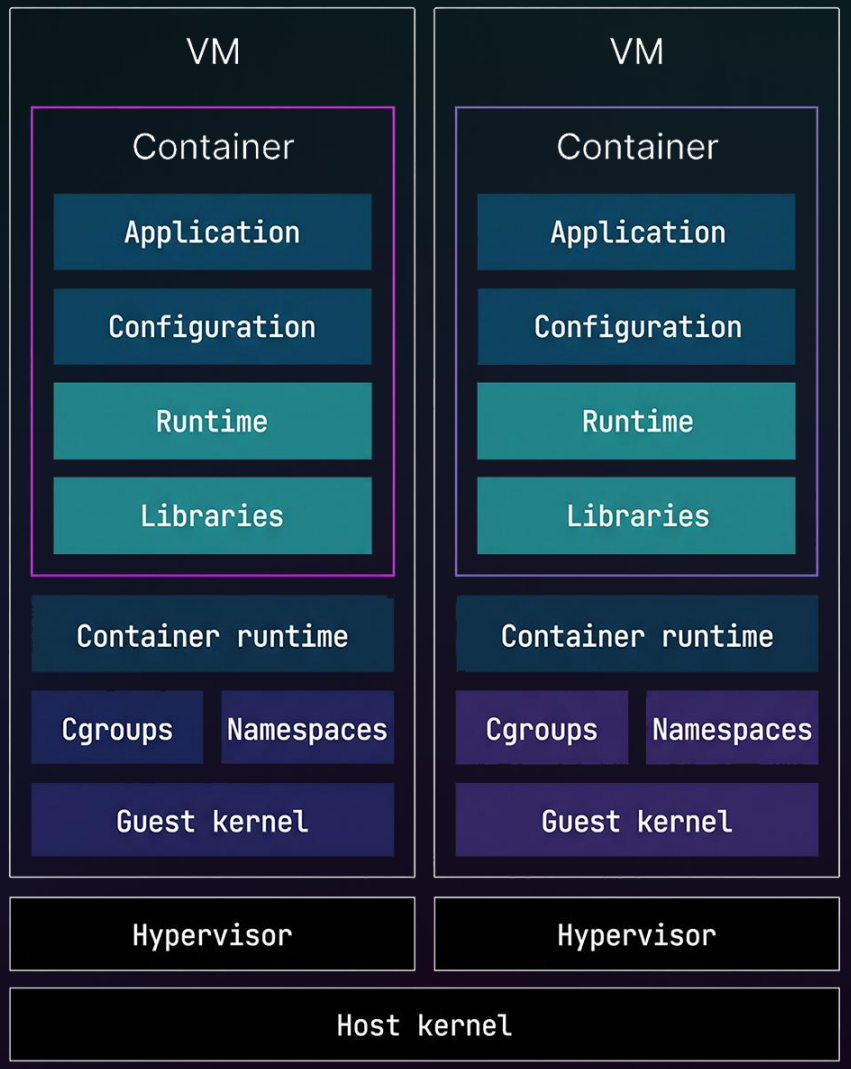
- Контейнеры не изолируют
  - Использование одного и того же ядра
  - Зависят от software компонентов для изоляции
  - Большое количество эксплойтов для различных компонентов



LUNTRY

Назад к (micro)VM

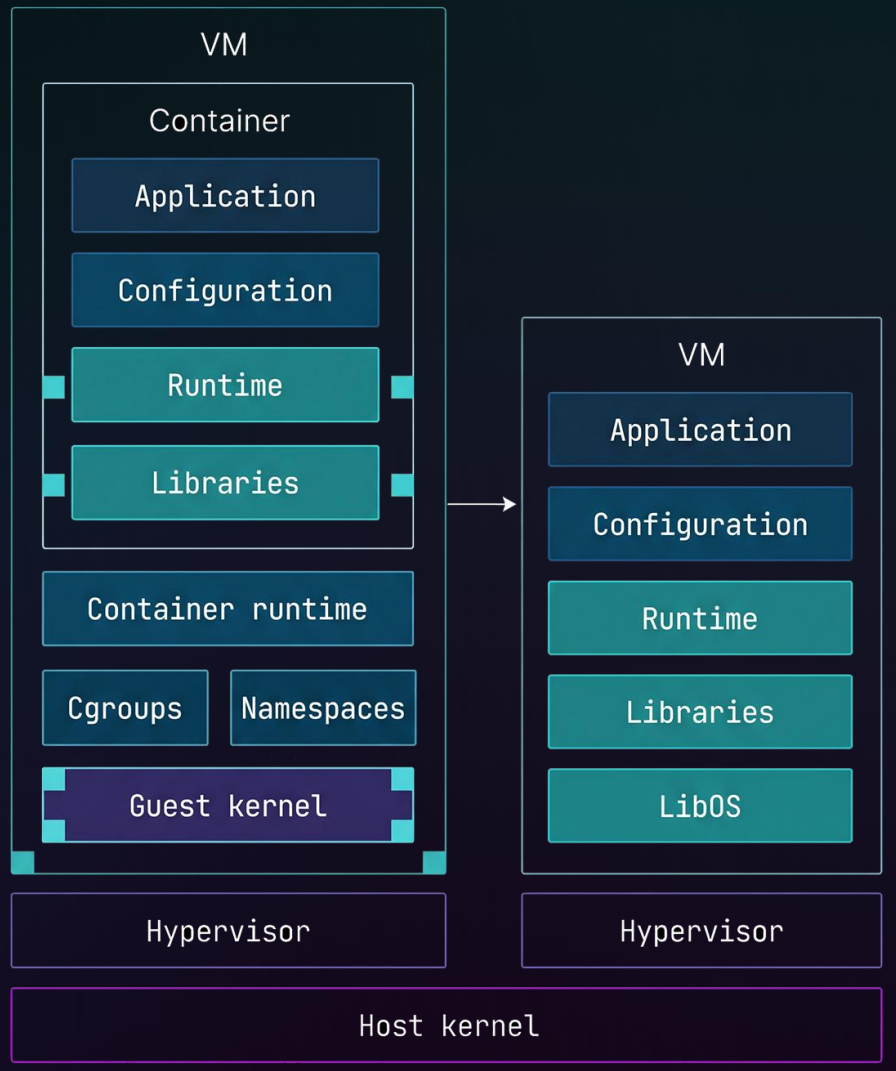
- Сочетание контейнеров и VM
  - Сохранение преимуществ контейнеров
  - Изоляция контейнеров внутри виртуальных машин
- Побочные эффекты
  - Более высокие накладные расходы
  - Усложнение стека



LUNTRY

Назад к (micro)VM

- Сочетание контейнеров и VM
  - Сохранение преимуществ контейнеров
  - Изоляция контейнеров внутри виртуальных машин
- Побочные эффекты
  - Более высокие накладные расходы
  - Усложнение стека

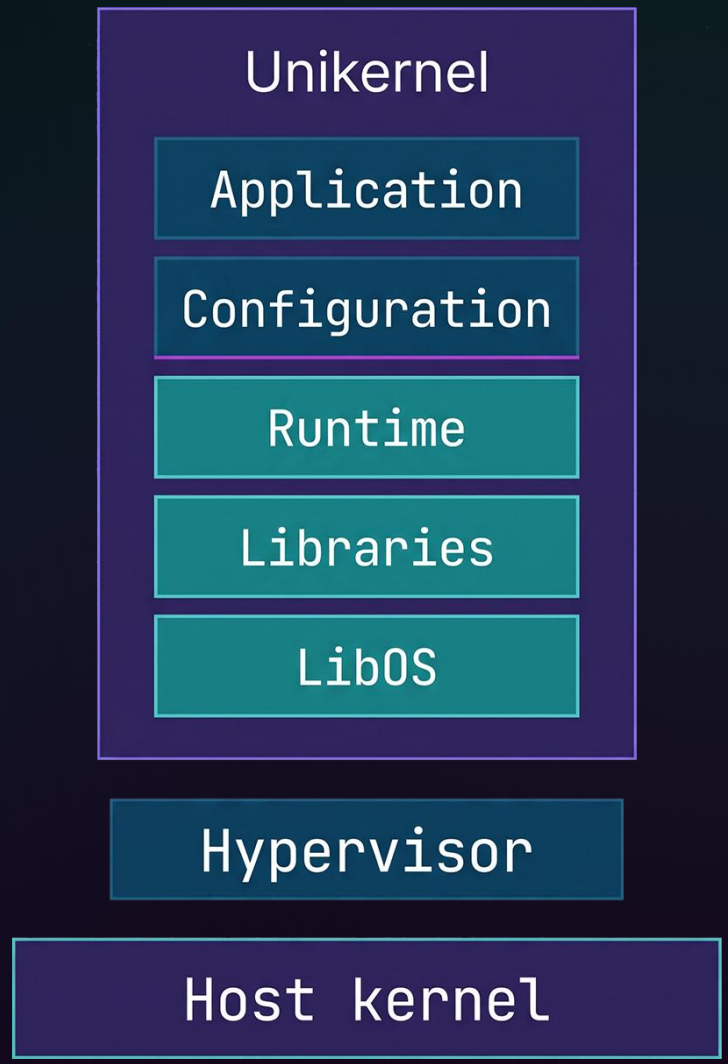


LUNTRY

(Re)Introducing unikernels

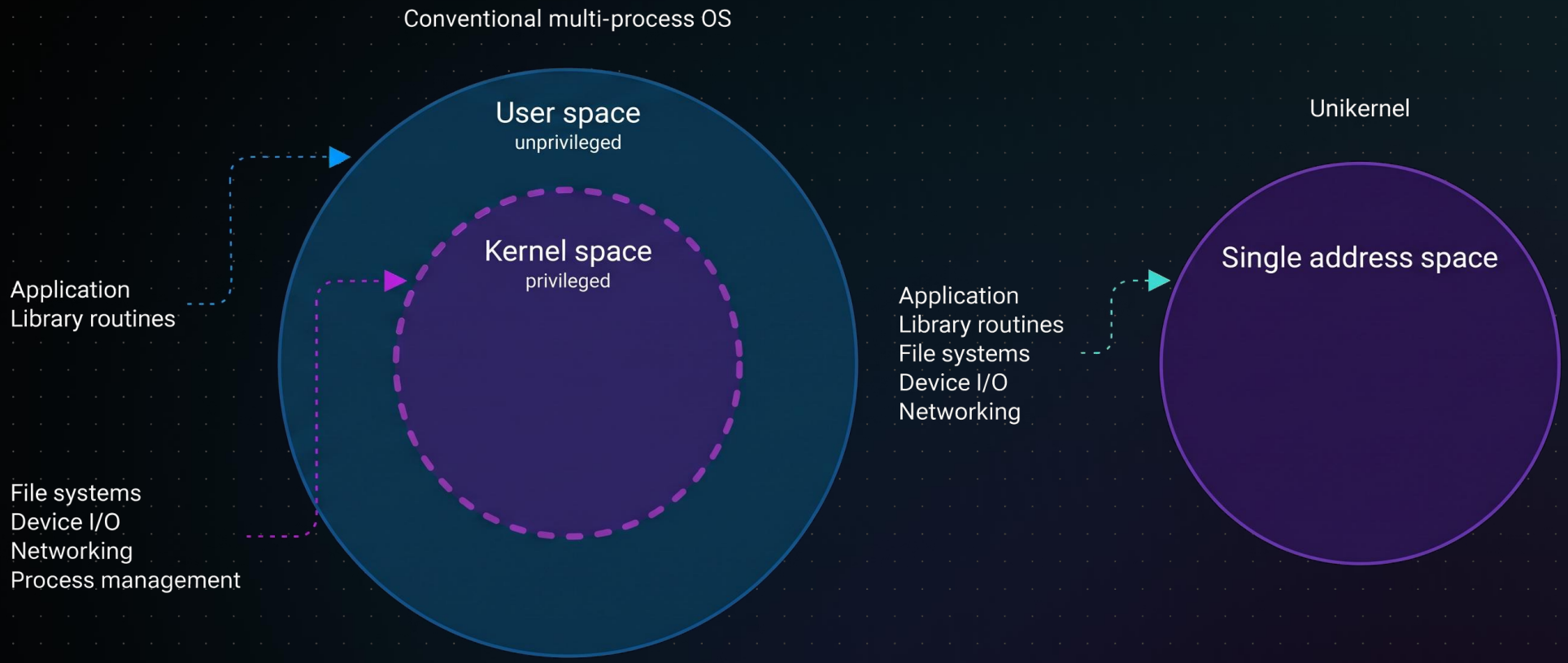
- unikernels
  - Единое адресное пространство
  - Построено с использование LibOS

- Бенефиты
  - Быстрое время загрузки
  - Уменьшение поверхности атаки
  - Настоящая изоляция
  - Меньший объем занимаемой памяти



LUNTRY

Единое адресное пространство

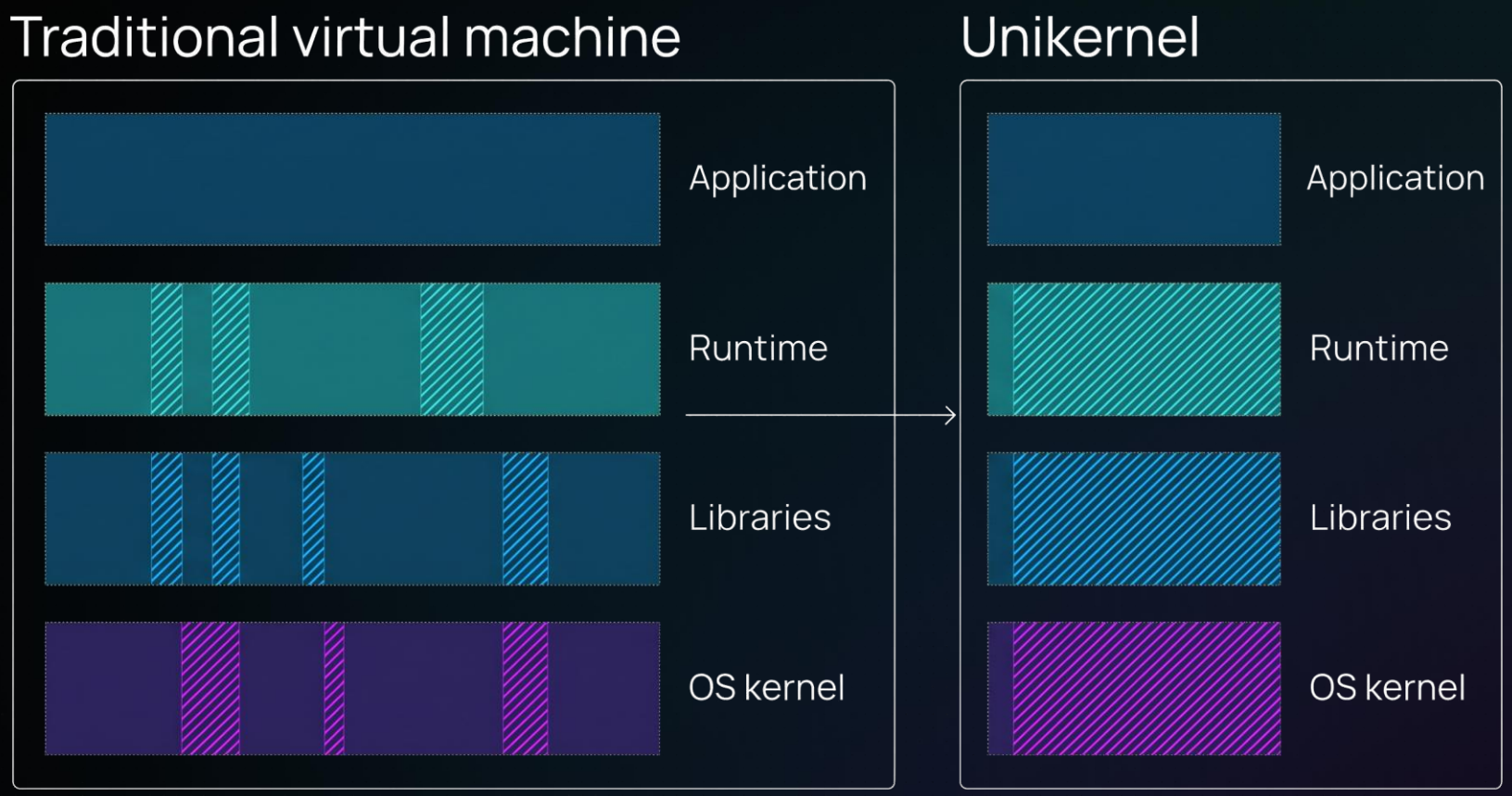


- Приложение и ядро в одном адресном пространстве: syscall — обычный вызов функции
- Внутри гостя нет границы user / kernel: единственная граница — гипервизор

Схема: iximiuz Labs — Hands-On Introduction to Unikernels (Antoine Cotten)

15

Специализация: в образе только нужное

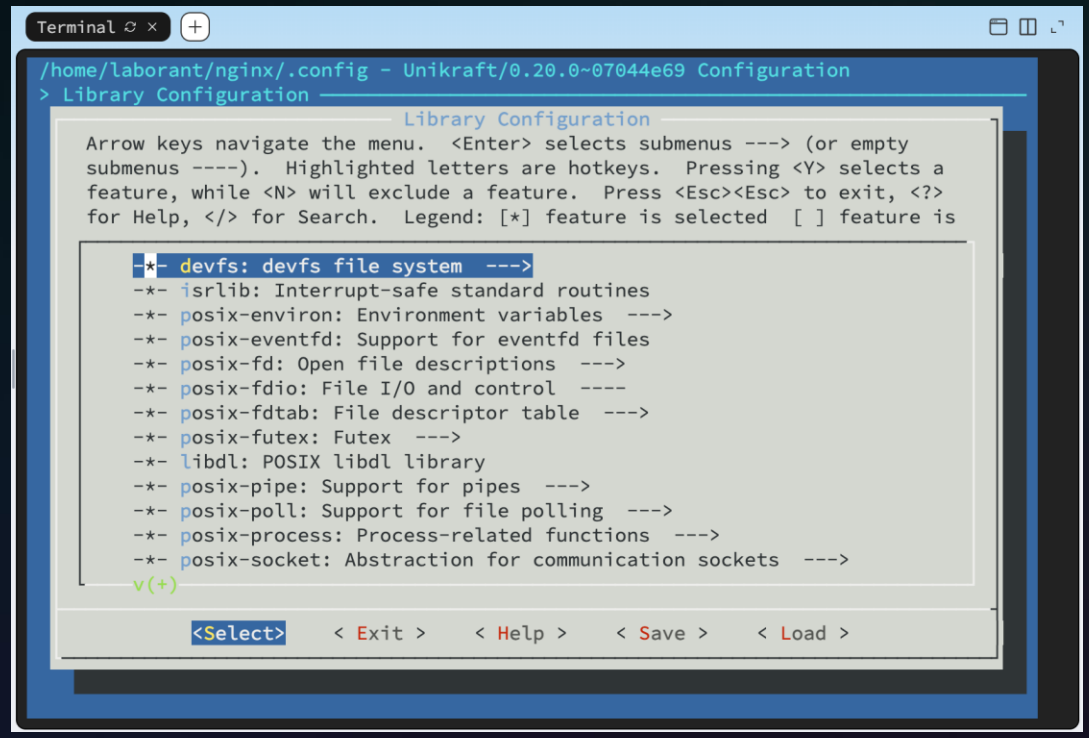


- В образ попадают только те части runtime, библиотек и ядра, которые нужны приложению
- Меньше кода — меньше поверхность атаки, размер образа и время загрузки

Схема: iximiuz Labs — Hands-On Introduction to Unikernels (Antoine Cotten)

Но за изоляцию надо платить

- Нет shell и пользователей — сложный troubleshooting
- Один процесс, нет fork (Postgres, Apache, Redis)
- Нет стандартизации — у каждого свой toolchain
- Нужна глубокая экспертиза для портирования



Unikraft: библиотеки ядра выбираются вручную под приложение

Схема: iximiuz Labs — Hands-On Introduction to Unikernels (Antoine Cotten)

Как запустить в Kubernetes?

- unikernels должны выглядеть как обычный OCI images
- Container runtime не знает как запускать unikernels



Unikraft: библиотеки ядра выбираются вручную под приложение

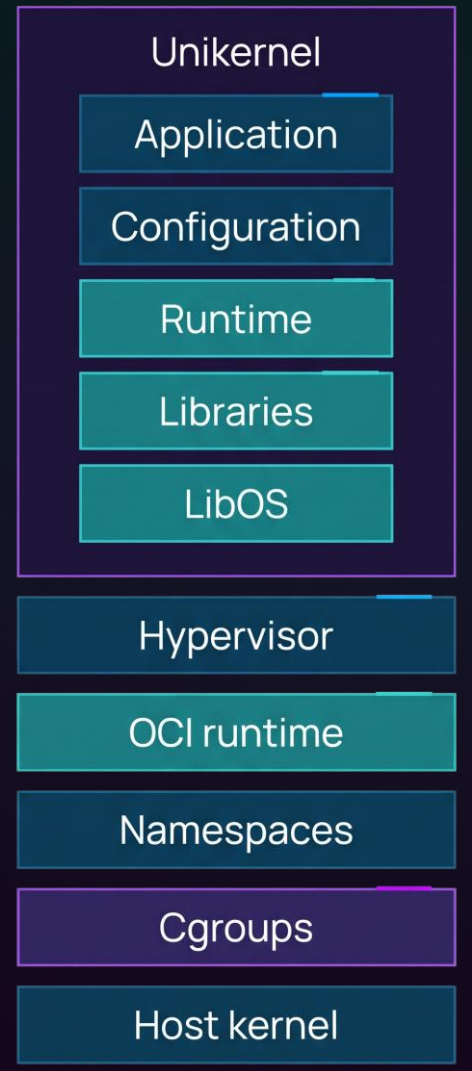


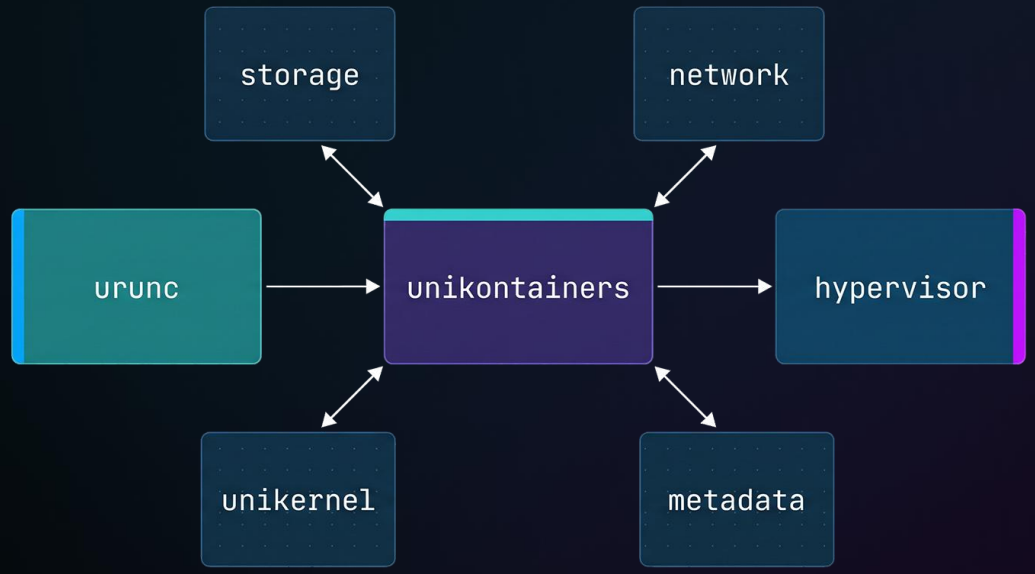
Схема: iximiuz Labs — Hands-On Introduction to Unikernels (Antoine Cotten)

Urunc: unikernels container runtime

LUNTRY

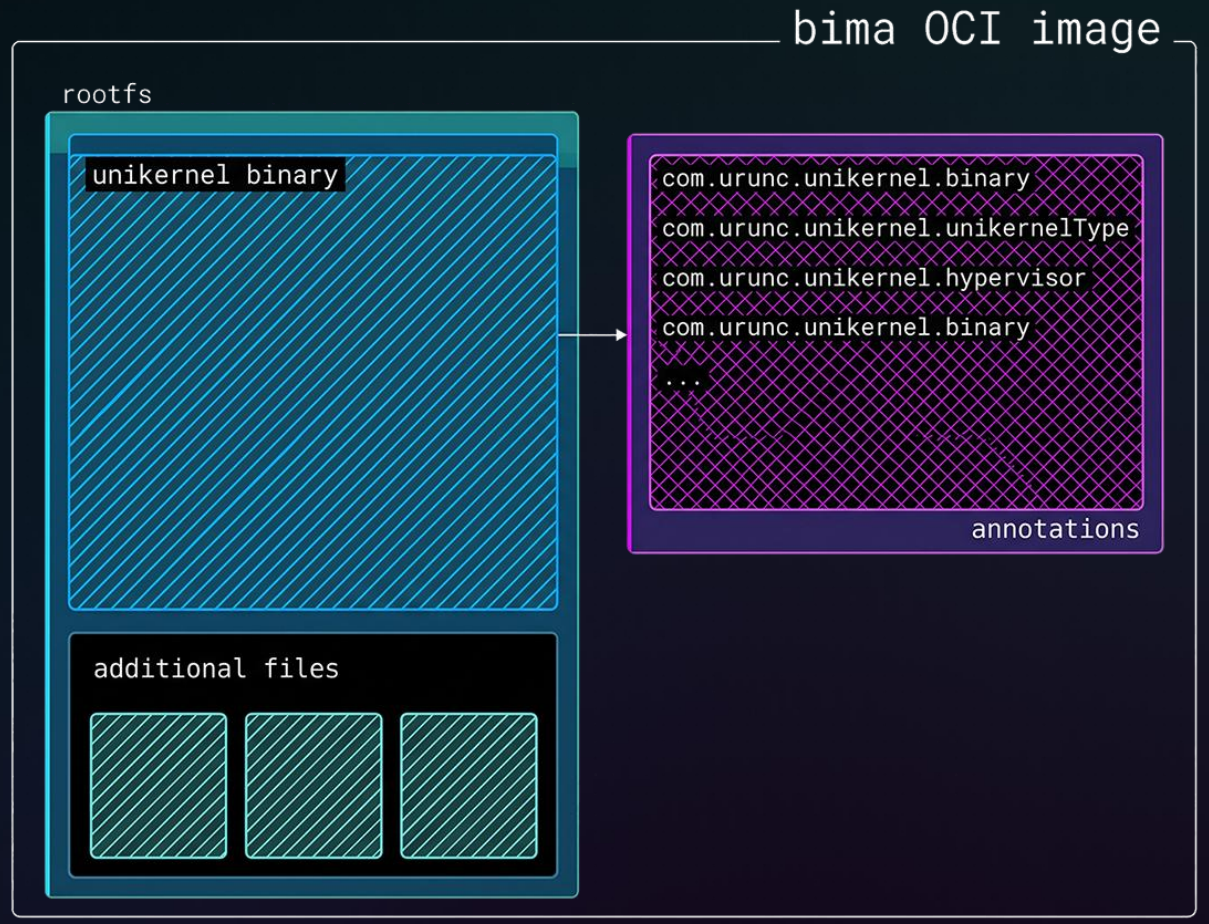
- CRI совеместимый рантайм, написаный на Go
- Рассматривает unikernels как процессы
- Образы unikernels являются артефактами OCI
- Использует гипервизоры для запуска ВМ на основе unikernels

urunc



Urunc: Unikernel OCI images

- Стандартные OCI images
- Может управляться с помощью стандартных инструментов
- Urunc использует специальные аннотации для корректной работы



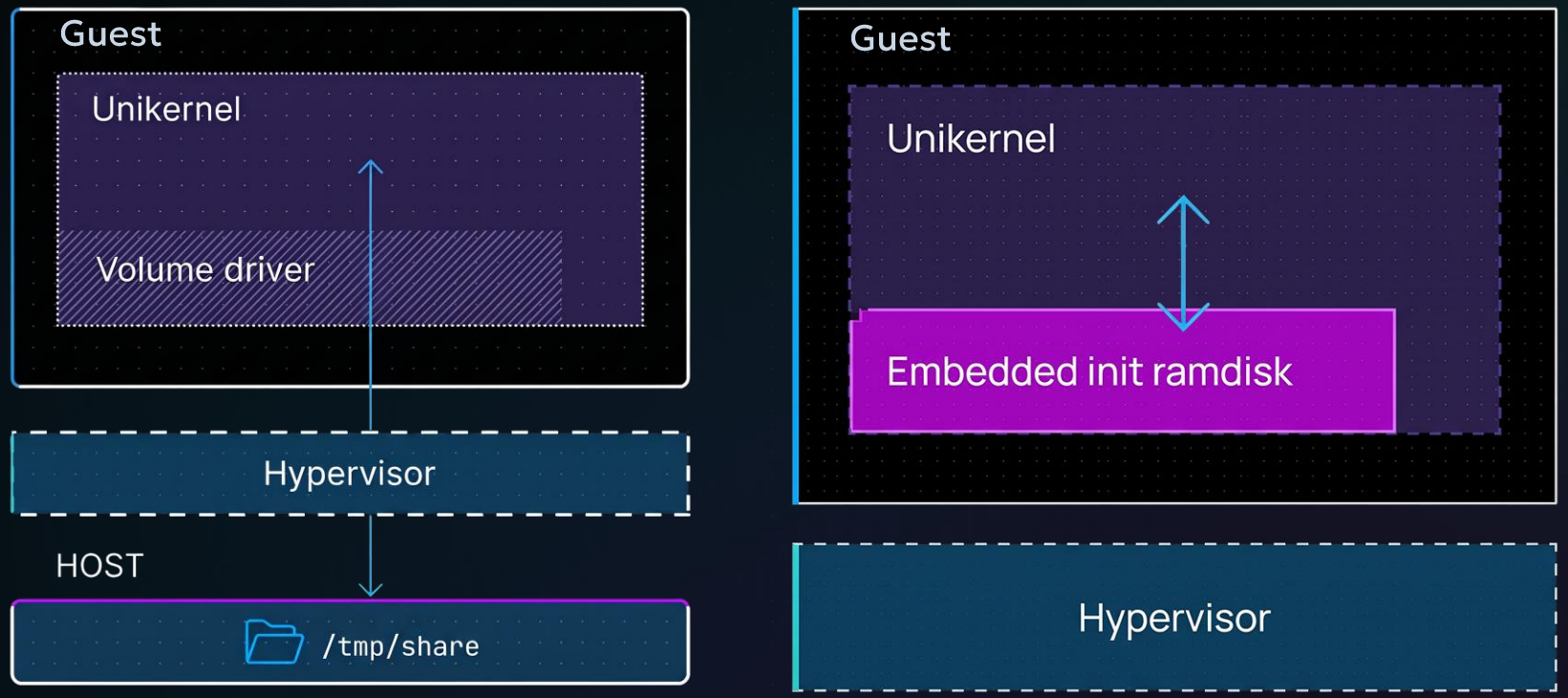
LUNTRY

Аннотации urunc

01	Метаданные хранятся в аннотациях образа и /urunc.json
02	com.urunc.unikernel.binary — путь к unikernel
03	com.urunc.unikernel.unikernelType — тип гостя
04	com.urunc.unikernel.hypervisor — sandbox monitor
05	com.urunc.unikernel.initrd / .block — rootfs и диск
06	Аннотации напрямую управляют запуском песочницы

LUNTRY

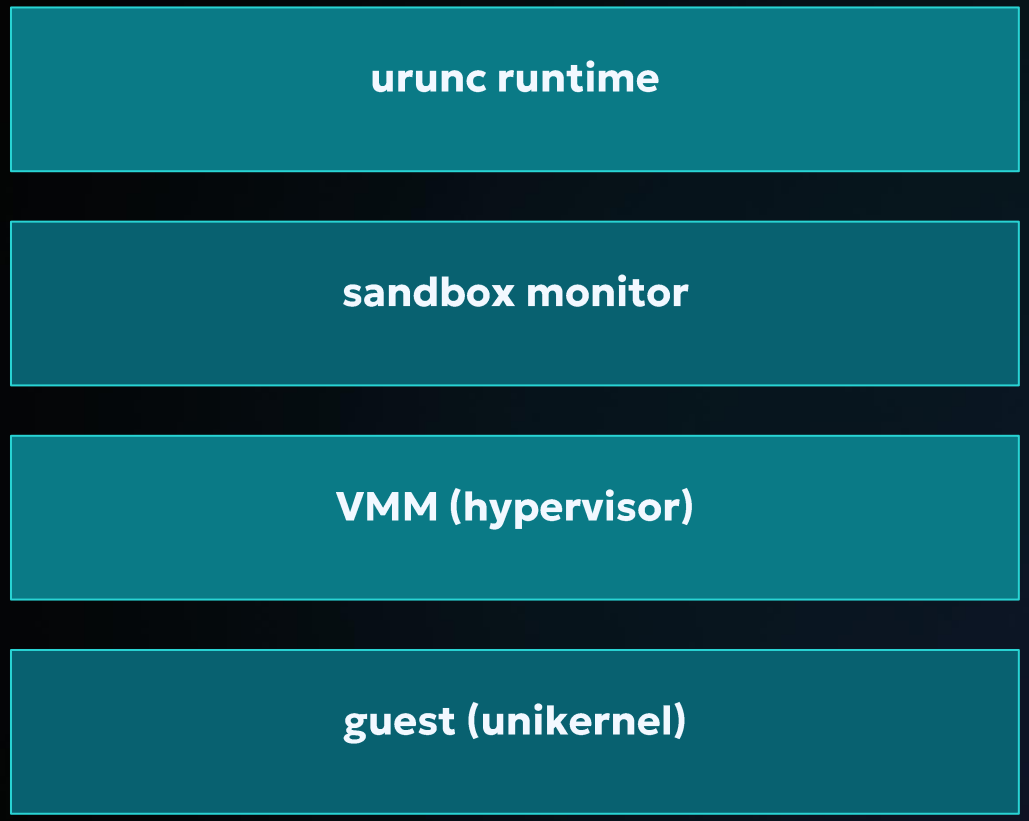
Как гость получает файлы



- Shared FS (9p / VirtIO): гость читает и пишет каталог хоста через гипервизор
- initrd: ramdisk, который монитор загружает в память гостя при старте
- В urunc источник файлов задают аннотации initrd и block — запомним их

Схема: iximiuz Labs — Hands-On Introduction to Unikernels (Antoine Cotten)

Архитектура urunc



- **runtime** — реализует CRI / OCI, разбирает аннотации
- **sandbox monitor** — поднимает изолированное окружение
- **VMM** — гипервизор, запускает виртуальную машину
- **guest** — сам unikernel с приложением
- Каждый слой — своя граница доверия

LUNTRY

Мониторы и гости

LUNTRY

Sandbox monitors

- QEMU
- Solo5-hvt
- Solo5-spt
- Firecracker

Guests

- Unikraft
- Linux
- Rumprun
- MirageOS

Монитор задаётся аннотацией com.urunc.unikernel.hypervisor, гость — unikernelType

LUNTRY

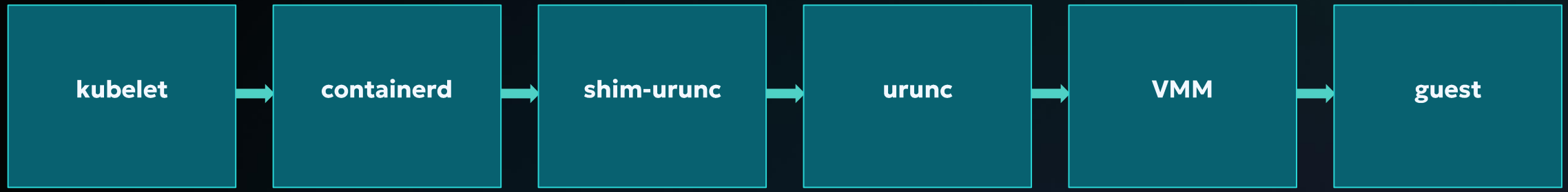
Как это запускается

RuntimeClass и Pod

```
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: urunc
handler: urunc
---
apiVersion: v1
kind: Pod
spec:
  runtimeClassName: urunc
  containers:
  - name: unikernel
    image: registry/app-unikernel:latest
```

LUNTRY

Путь запроса до unikernel'a



- kubelet видит runtimeClassName и идёт в containerd через CRI
- containerd выбирает shim: containerd-shim-urunc-v2
- urunc разбирает аннотации и поднимает sandbox monitor
- monitor запускает VMM, а тот — guest-unikernel

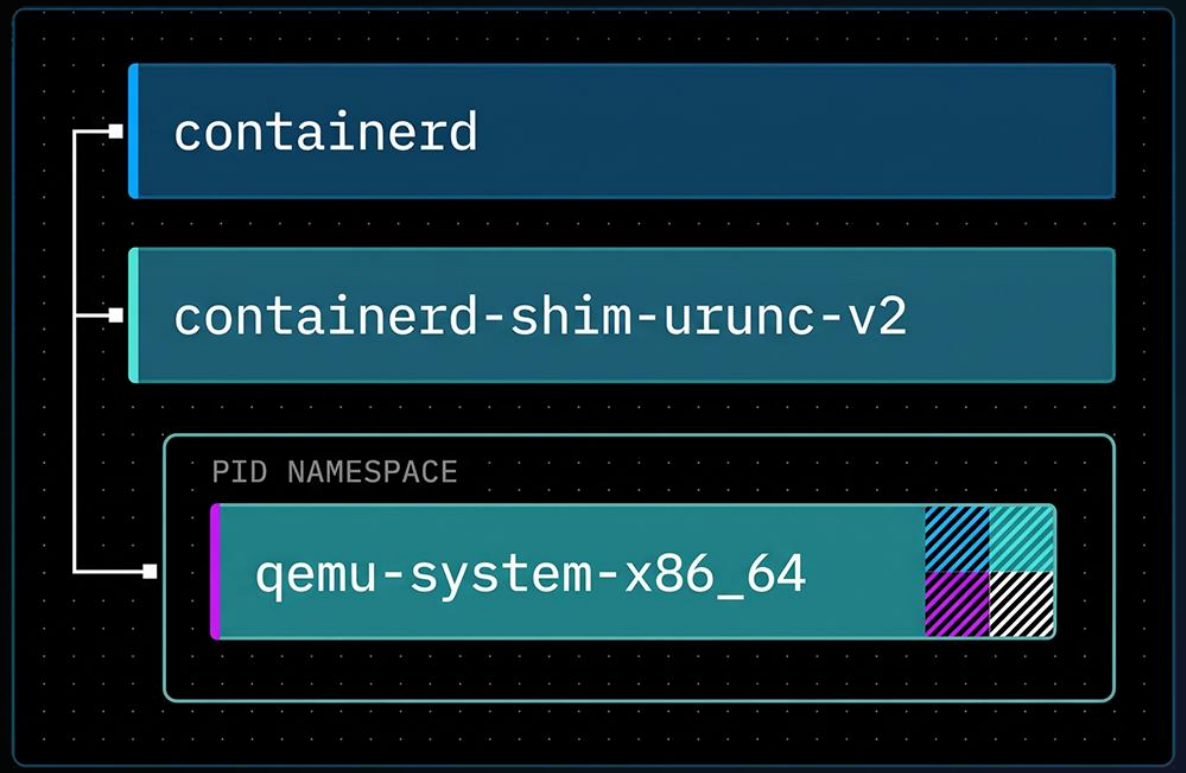
LUNTRY

Жизненный цикл Pod'a

1. kubelet видит runtimeClassName: urunc
2. containerd выбирает shim urunc вместо runc
3. urunc create: разбор аннотаций, подготовка окружения
4. urunc start: sandbox monitor → VMM → guest
5. Для Kubernetes это по-прежнему обычный Pod

LUNTRY

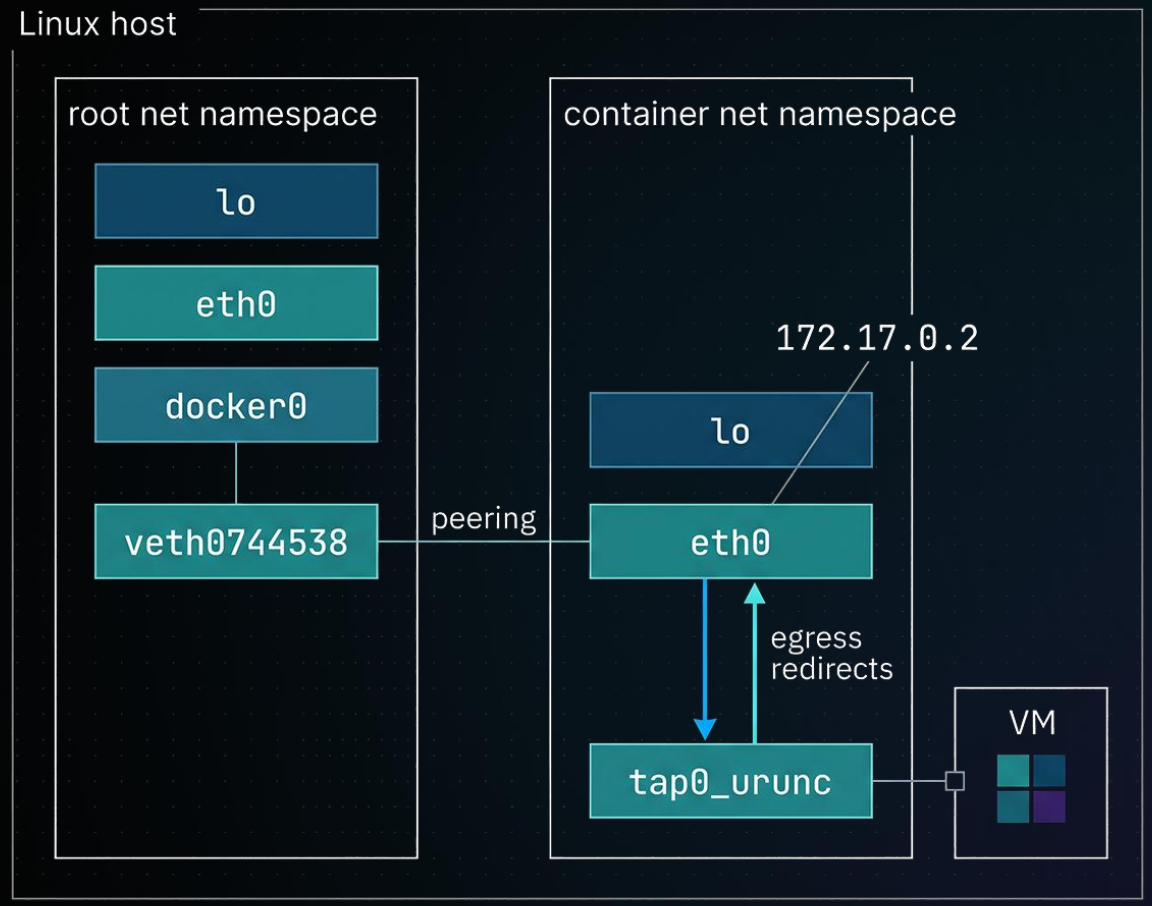
Что видно на хосте: дерево процессов



- VMM — обычный процесс хоста, потомок shim'a runc
- Живёт в namespaces и cgroups контейнера
- Для containerd и kubelet это тот же контейнер: логи, exec, stop
- Командную строку этого процесса собирает runc

Схема: iximiuz Labs — Hands-On Introduction to Unikernels (Antoine Cotten)

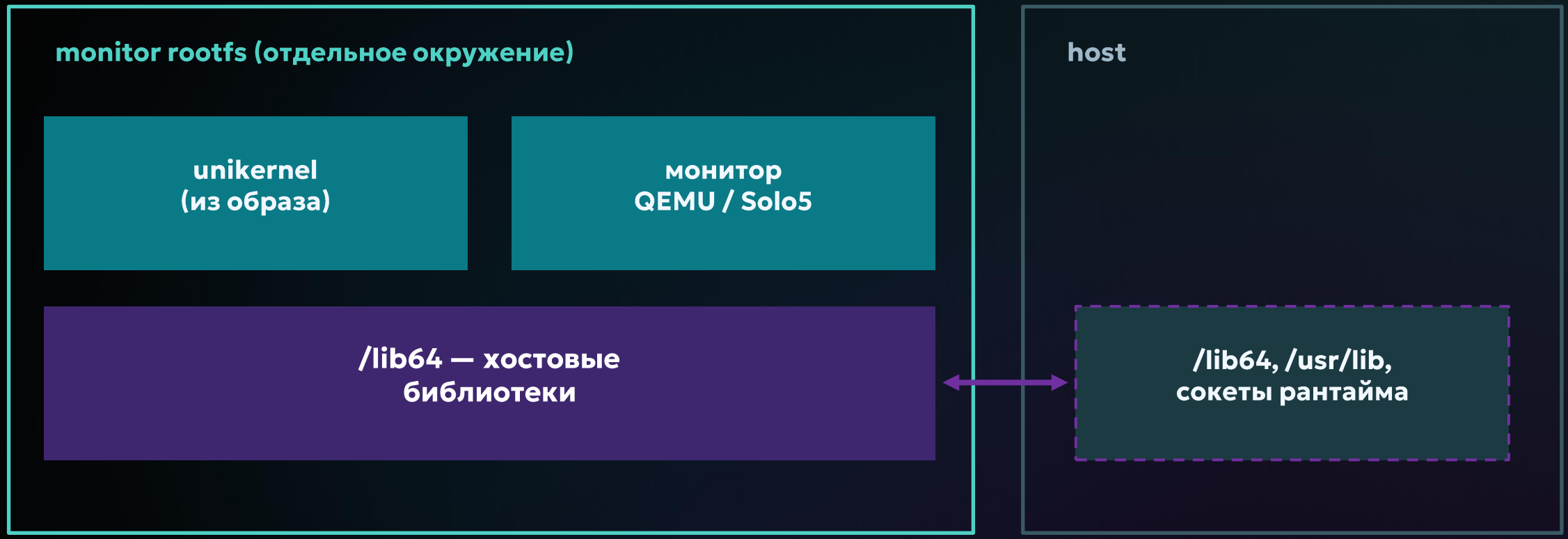
Сеть: TAP и traffic control



- CNI настраивает обычную veth-пару и eth0 в netns Pod'а
- urunc создаёт tap0_urunc и подключает его к VM
- tc redirect перекладывает трафик между eth0 и TAP
- IP Pod'а достаётся гостю — NetworkPolicy и Service работают как обычно

Схема: iximiuz Labs — Hands-On Introduction to Unikernels (Antoine Cotten)

Где выполняется monitor



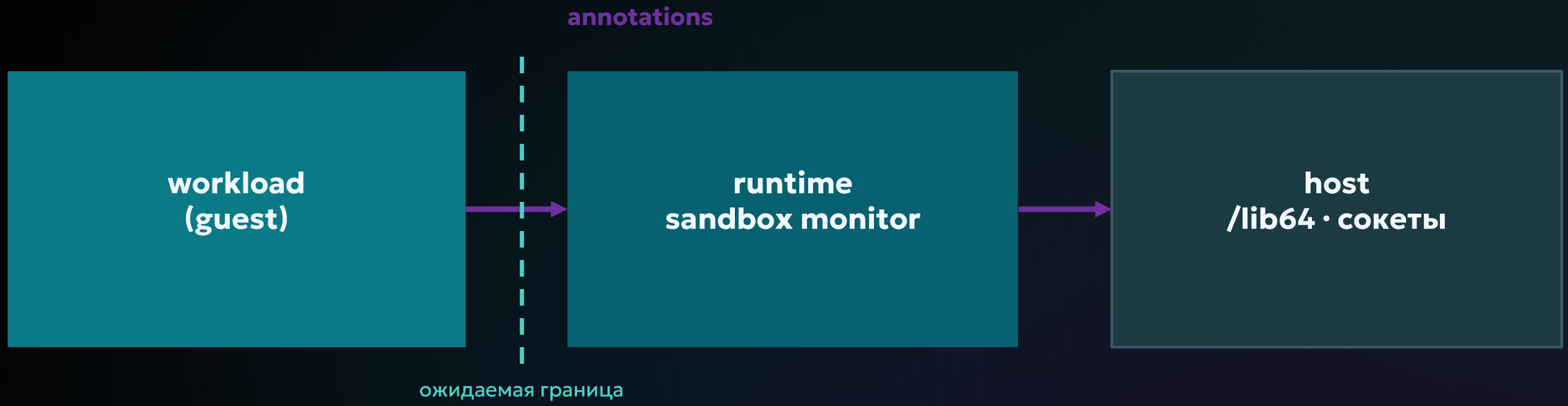
- UID процесса монитора берётся из образа
- Хостовые библиотеки попадают в окружение — здесь реальная граница с хостом

LUNTRY



Threat model

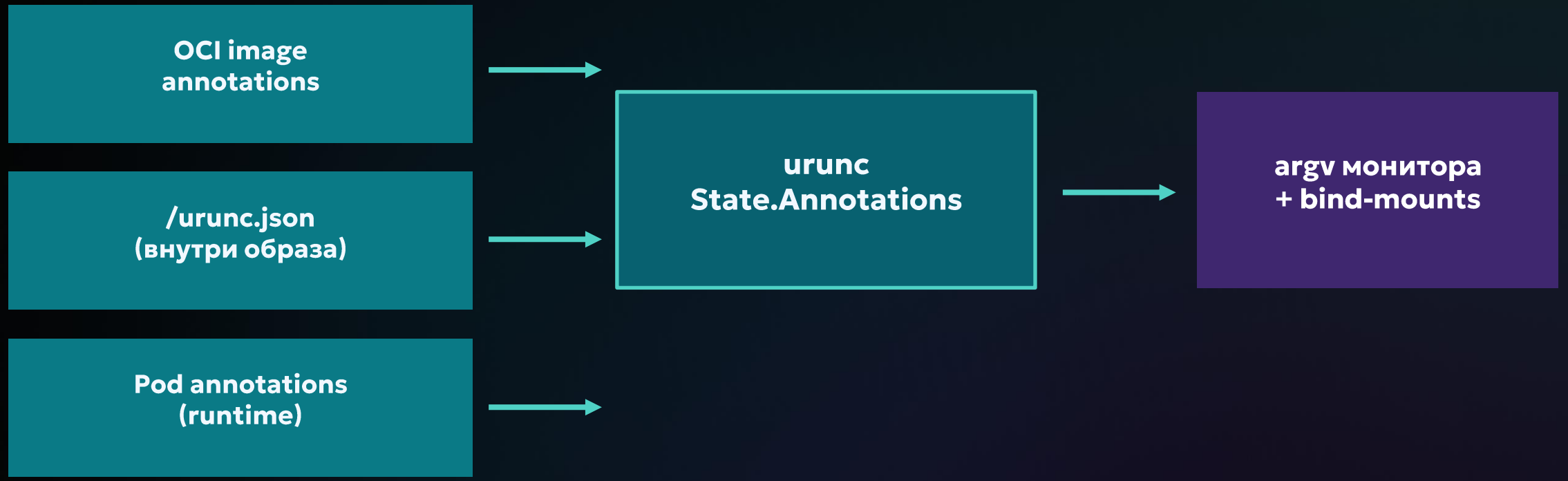
Границы изоляции



- Ожидание: workload заперт внутри виртуальной машины
- Реальность: runtime доверяет аннотациям workload — граница смещается к хосту

LUNTRY

Откуда приходят аннотации



Кто пушит образ или создаёт Pod — тот влияет на запуск песочницы

LUNTRY

Ключевой вопрос

- Что если workload контролирует аннотации?
- Их значения попадают в команду запуска монитора
- Их значения задают, какие файлы монтируются
- Доверенная зона (host) зависит от недоверенного ввода

LUNTRY

Требования безопасности

- Аннотации — недоверенный ввод, нужна валидация
- Параметры нельзя склеивать в строку команды
- Пути от workload нельзя использовать напрямую
- Окружение монитора не должно быть хостовой ФС

LUNTRY

LUNTRY

Уязвимость #1 — Argument injection

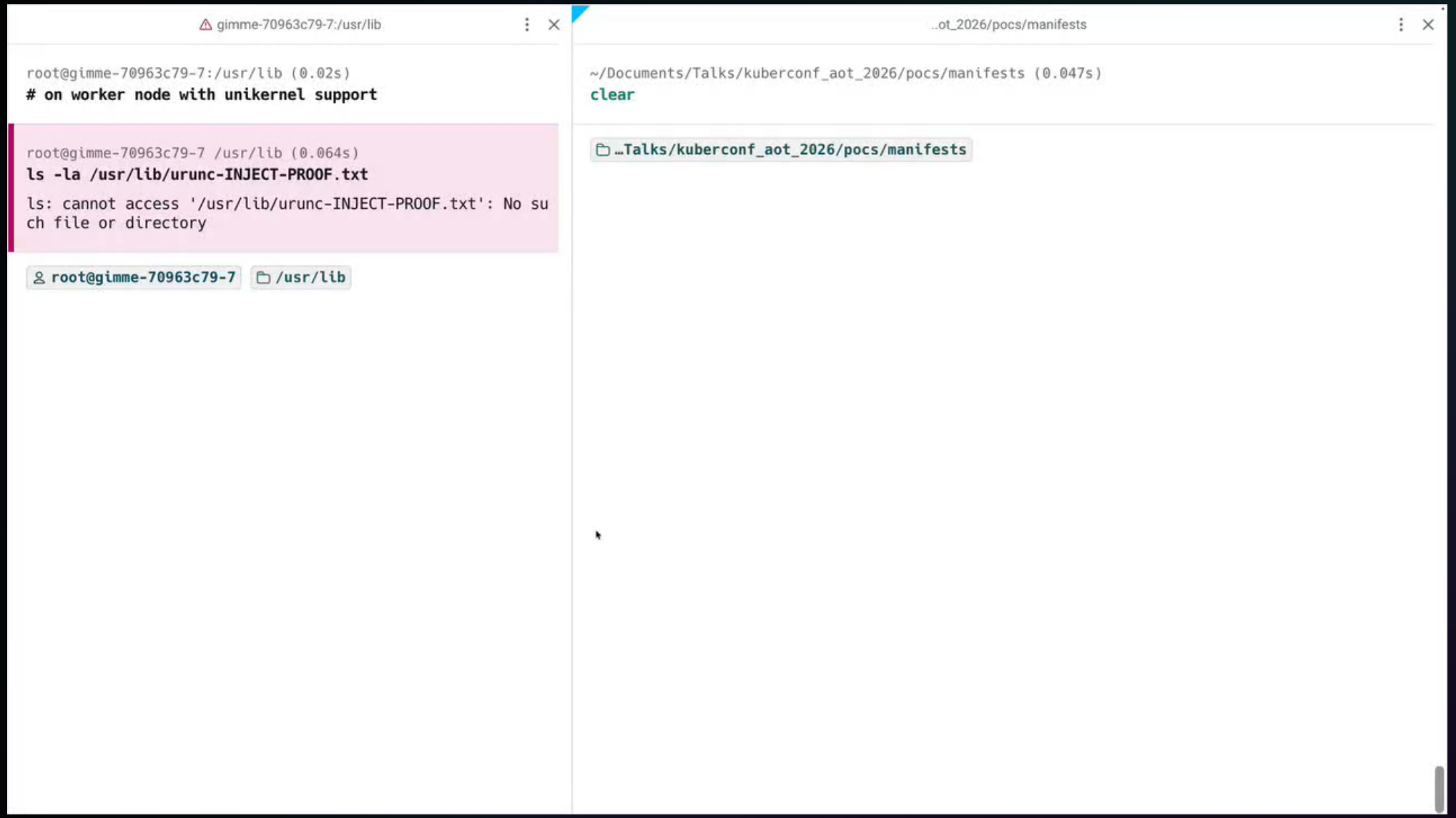
GHSA-crxr-jm9v-349c · CVSS 9.3 Critical

Что сломалось

- Аннотации binary, block, initrd не валидировались
- Из них строкой собиралась команда sandbox monitor
- Затем strings.Split(cmdString, " ") — разбивка по пробелам
- Значение со space → лишние argv-токены монитора
- → инъекция произвольных аргументов QEMU / Solo5

LUNTRY

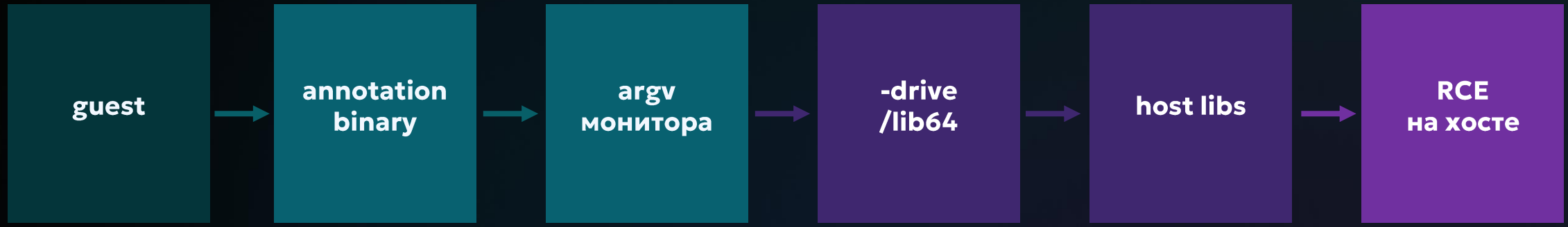
PoC — argument injection (QEMU/Solo5)



LUNTRY

Impact

LUNTRY



- Монитор работает в окружении с хостовыми библиотеками
- Мониторы: QEMU, Solo5-spt, Solo5-hvt; UID процесса берётся из образа

LUNTRY

Уязвимость #2 — Path traversal

GHSA-3385-hmpj-4678 · CVSS 9.3 Critical

Что сломалось

- Аннотации initrd, block, RPCAddress задают монтируемые файлы
- Путь шёл в filepath.Join(MonRootfs, path) без securejoin
- Образ или Pod может указать на файлы хоста через ../
- RPCAddress + vAccel: vsock → bind-mount host-пути (Firecracker)
- → чтение, запись и порча файлов хоста

LUNTRY

initrd и block

- initrd: чтение любого файла из окружения монитора (/lib64...)
- + дозапись bind-mount'ов в произвольные host-файлы
- block: чтение и запись любого файла в окружении
- Гости: Unikraft, Linux, Rumprun, MirageOS

LUNTRY

RPCAddress (Firecracker)

- Задаёт vsock-адрес агента vAccel на хосте
- urunc монтировал директорию этого адреса в окружение
- → в окружение можно принести любую host-директорию
- В связке с initrd/block — чтение/запись host-файлов

LUNTRY

PoC — path traversal

```
# пути из аннотаций шли в filepath.Join(MonRootfs, path), без securejoin
#   -> ../ выводит за пределы rootfs монитора на хостовую ФС

metadata:
  annotations:
    # чтение host-файла внутрь гостя
    com.urunc.unikernel.initrd: "../../../../../lib64/libc.so.6"
    # чтение / запись произвольного host-файла
    com.urunc.unikernel.block: "../../../../../etc/shadow"
    # Firecracker: bind-mount host-сокета в окружение монитора
    com.urunc.unikernel.vAccel: "vsock"
    com.urunc.unikernel.RPCAddress: "/run/containerd/containerd.sock"
```

LUNTRY

Search sessions, agents, files...

gimme-70963c79-7:/tmp

root@gimme-70963c79-7:/tmp (0.018s)

```
# on worker node with unikernels support
```

root@gimme-70963c79-7 /tmp

..ot_2026/pocs/manifests

~/Documents/Talks/kuberconf_aot_2026/pocs/manifests (0.052s)

```
clear
```

…Talks/kuberconf_aot_2026/pocs/manifests

Что починили в runc 0.8.0

- Отказ от склейки строк в CLI монитора
- Валидация значений аннотаций
- securejoin для путей, контролируемых образом
- Разделение rootfs монитора и контейнера
- RO-монтирование неизменяемого; vAccel только по конфигу

LUNTRY

LUNTRY

Делаем выводы

Выводы

- Усиленная изоляция ≠ автоматическая безопасность
- Граница доверия — это обработка ввода от workload
- Аннотации = недоверенный ввод: binary, block, initrd, RPCAddress
- Две critical (CVSS 9.3): argument injection и path traversal
- Исправлено в urunc 0.8.0 — обновляйтесь и проверяйте образы

LUNTRY

Что выбрать для AI-нагрузок

	Граница и цена*	Когда выбирать	AI-атака и что остаётся
urunc	VMM + unikernel мс старт, МБ памяти только портированные образы	однопроцессные inference-сервисы: важны плотность и холодный старт	вредоносный образ / Pod от агента → аннотации → argv и пути монитора → хост (GHSA-crxr, GHSA-3385)
Kata	microVM + гостевой Linux ~0,5–1+ с, 100+ МБ на Pod любой OCI-образ	произвольные недоверенные контейнеры, выделенная GPU, Confidential Containers	kernel exploit после prompt injection остаётся в госте; GPU через VFIO внутри VM
gVisor	user-space ядро Sentry быстрый старт дорогие syscall и I/O	code interpreter и AI-агенты, когда на ноде нет KVM	exploit упирается в Sentry; nvproxy сужает ioctl GPU (ср. CVE-2024-0132, CVE-2025-23266)
Firecracker	KVM microVM ≤125 мс, ≤5 МиБ на VMM не CRI: нужна обвязка	своя платформа песочниц для агентов и FaaS	сильная граница, но GPU нет; риск переезжает в вашу обвязку

Изоляция не спасает control plane: Ray ShadowRay (CVE-2023-48022), открытые Jupyter / MLflow — нужны authn/authz и сегментация

** порядок величин, зависит от VMM, гостя и настроек — мерить на своей нагрузке*

Что проверить у себя

- Обновиться на urunc 0.8.0 и новее
- Валидировать аннотации и /urunc.json образов
- Ограничить, кто создаёт Pod с runtimeClassName: urunc
- Не запускать образы из недоверенных источников

LUNTRY

СПАСИБО ЗА ВНИМАНИЕ!

Сергей Канибор — R&D / Container Security, Luntry

sk@luntry.ru · @r0binak · luntry.ru

KUBER CONF от АОТ