LUNTRY

Предотвращение Runtime угроз в контейнерах и Kubernetes



Дмитрий Евдокимов

Founder&CTO
Luntry

Обо мне

”

Я не верю в то, что систему можно сделать надежной и безопасной, не понимая того, как она устроена.



Основатель и технический директор Luntry

Более 15 лет опыта в ИБ

Специализация – безопасность контейнеров и Kubernetes

Автор ТГ-канала k8s(in)security

Эксперт в сфере безопасности контейнерных сред

- Организатор конференции «БеКон» по БЕзопасности КОНтейнеров
- Бывший редактор рубрик в журнале «ХАКЕР», автор серии статей
- Автор курса «Cloud Native безопасность в Kubernetes»
- Член программного комитета CFP DevOpsConf и HighLoad++

Спикер

VK Kubernetes	Confidence	ZeroNights	БеКон	HITB
DevOpsConf	HackInParis	KuberConf	BlackHat	PHDays
Kazhackstan	HighLoad++	OFFZONE	DevOops	SAS

LUNTRY

О компании Luntry

Luntry – это Комплексная Защита
на всем жизненном цикле
контейнерных приложений
и средств оркестрации на базе Kubernetes

Продукт в реестре Минцифры
https://reestr.digital.gov.ru/reestr/1057835/

Получение серитификата ФСТЭК планируется во третьем квартале 2025 года.

LUNTRY.RU

Функциональность Luntry

Контроль всех Kubernetes-ресурсов

Контроль состояния Kubernetes-кластеров

Контроль соответствия кластера стандартам



Управление уязвимостями образов и best practice

Сетевая безопасность

Анализ прав доступа

Защита Runtime

Функциональность Luntry

Контроль всех Kubernetes-ресурсов

Контроль состояния Kubernetes-кластеров

Контроль соответствия кластера стандартам



Управление уязвимостями образов и best practice

Сетевая безопасность

Анализ прав доступа

Защита Runtime

План вебинара

LUNTRY

01	Чем отличаются детектирование, реагирование и предотвращение
02	Что общего и разного у AppArmor, SeLinux, seccomp
03	Как NetworkPolicy относится к теме предотвращения
04	Что такое Linux Security Module (LSM) и при чем тут eBPF
05	Как Luntry помогает решить задачи, связанные с предотвращением Runtime-угроз
06	Выводы

LUNTRY

Чем отличаются детектирование, реагирование и предотвращение

В предыдущих ~~сериях~~ вебинарах*

01

Runtime Security: на вкус и цвет все фломастеры разные
- Про обнаружение runtime угроз
- Поведенческое обнаружение

02

Ловим злоумышленников и собираем улики в контейнерах Kubernetes
- Про реагирование на runtime угроз
- Сбор артефактов для расследования инцидентов

03

Безопасность контейнеров и Kubernetes для SOC
- Про работу с runtime угрозами в целом
- Поведенческое, сигнатурное, гибридное обнаружение

*- слайды и видео доступно на https://luntry.ru/research

Детектирование, реагирование и предотвращение

Давайте различать

LUNTRY

01

Детектирование

- Происходит только оповещение об обнаружении наблюдаемого/нежелательного события
- На пример: вызов execve() с определёнными аргументами
- Злоумышленник продолжает развивать атаку

02

Реагирование

- На основании некоторой политики принимается решение о той или иной активности в отношении наблюдаемого объекта при обнаружении события
- На пример: заверение процесса, завершение контейнера, снятие дампа ФС/RAM контейнера для расследования
- Злоумышленник имеет временной зазор для развития атаки!

03

Предотвращение

- На основании некоторой политики принимается решение о разрешении или блокировки действия
- Злоумышленник вообще не может выполнить желаемую операцию

Не все prevention, что так называют

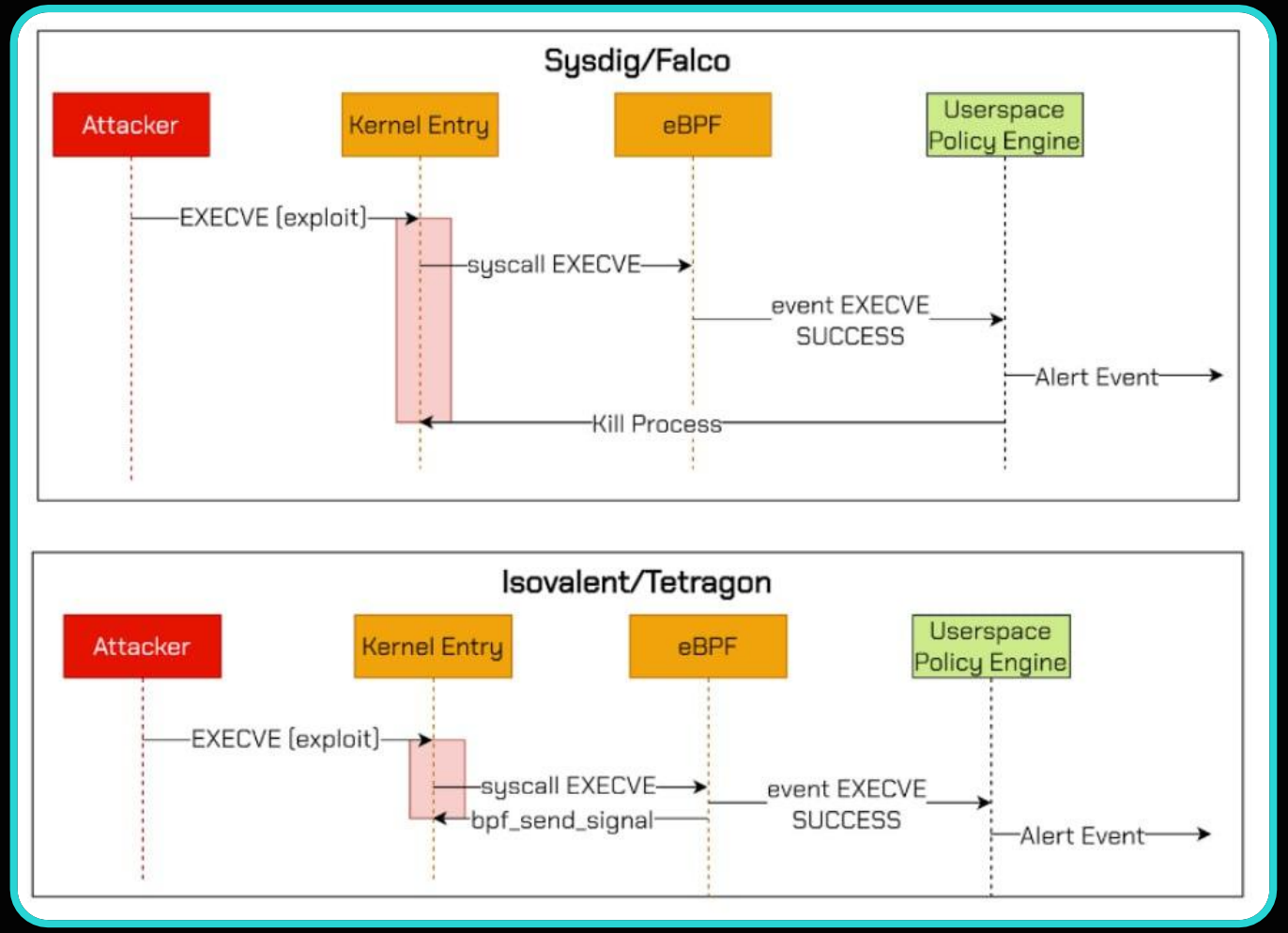
LUNTRY

Реагирование из user space

Отправка сигнала kill

Реагирование из kernel space

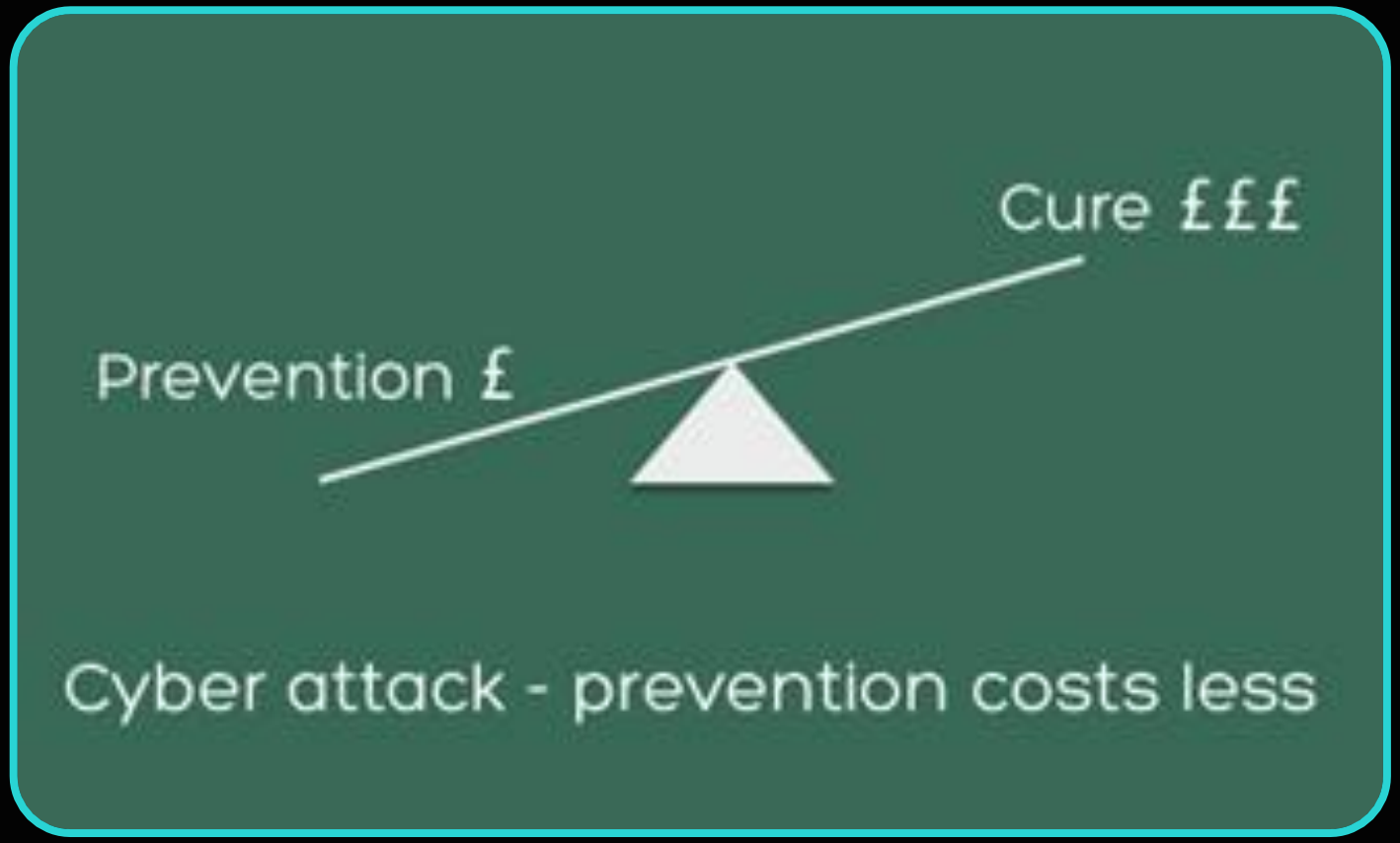
Отправка bpf_send_signal()



Статья “Runtime Security and the Role of eBPF/BPF-LSM”

Предотвращение = экономия

Предотвращение инцидента всегда дешевле расследования инцидента!



LUNTRY



Что общего и разного у AppArmor, SeLinux, seccomp

Seccomp, AppArmor, SeLinux

Технология	seccomp	Linux Security Modules (LSM)	
Реализация	seccomp профиль	AppArmor профиль	SeLinux политика
Краткое описание	Контроль на уровне системных вызовов	Контроль на уровне процессов. Mandatory Access Control (MAC)	Контроль на уровне процессов. Mandatory Access Control (MAC)
Основная цель	Усложнить побег из контейнера на хост через уязвимости ядра	Ограничить возможности контейнера для развития атаки и побега на хост	Усложнить побег из контейнера на хост через уязвимости ядра и некорректные конфигурации
Контроль над	Любым системным вызовом	Доступами к файлам, каталогам, сетевым портам, ресурсам системы, capabilities и действиями процессов	Доступами к файлам, каталогам, сетевым портам, ресурсам системы и действиями процессов
Версия ядра	2.6.12	2.6.36	2.6.0
Версия Kubernetes для GA	1.19	1.30	1.12
Предоставляется	Средствами ОС	Средствами ОС	Средствами ОС
Интеграция с Kubernetes	Раздел SecurityContext.SeccompProfile	Раздел SecurityContext.AppArmorProfile	Раздел SecurityContext.SELinuxOptions
Применение	Файл на Node	Файл на Node	Файл на Node
Режимы работы	Whitelist, Blacklist	Whitelist, Blacklist	Whitelist, Blacklist
Сложность описания и использования	Высокая (Из-за сложности хорошего покрытия)	Средняя	Высокая (Из-за сложности описания)
Значения по умолчанию	Да - RuntimeDefault	Да - RuntimeDefault	Нет
Влияние на производи-тельность	Минимальное	Минимальное	Среднее
Комментарий/ограничения	Тяжело поддерживать для активно развивающихся приложений	Не совместим с SeLinux	Не совместим с AppArmor

LUNTRY



Как NetworkPolicy относится к теме предотвращения

NetworkPolicy

Родной межсетевой экран Kubernetes

1. Контроль сетевого доступа

- ZeroTrust
- Микросегементация

2. Декларативный подход

- Policy-as-Code

3. Нативная скорость работы

- Минимальные накладные расходы (порой даже ускорения)

```yaml
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: app-access-policy
  namespace: default
spec:
  # Применяется к подам с меткой app=backend
  endpointSelector:
    matchLabels:
      app: backend
  # Входящий трафик (ingress)
  ingress:
  - fromEndpoints:
    - matchLabels:
        app: frontend
        namespace: default
    toPorts:
    - ports:
      - port: "8080"
        protocol: TCP
      rules:
        http:
        - method: "GET"
          path: "/api/v1/data"
        - method: "POST"
          path: "/api/v1/submit"
  # Исходящий трафик (egress)
  egress:
  - toEndpoints:
    - matchLabels:
        app: database
        namespace: default
    toPorts:
    - ports:
      - port: "5432"
        protocol: TCP
```

Полезные материалы:

“NetworkPolicy — родной межсетевой экран Kubernetes”, Positive Hack Days 2022

“NetworkPolicy для разработчиков: как, зачем и почему”, Ural Digital Weekend 2023

LUNTRY

iptables VS. eBPF

Технология	iptables	eBPF (tc, xdp)
Реализация	NetworkPolicy политика	NetworkPolicy политика
Краткое описание	Контроль сетевого доступа	Контроль сетевого доступа
Основная цель	Родной межсетевой экран для Pod в кластере	Родной межсетевой экран для Pod в кластере
Контроль над	Сетевыми соединениями (L3/L4/L7)	Сетевыми соединениями (L3/L4/L7)
Версия ядра	2.4	4.8-4.18
Версия Kubernetes для GA	1.7	1.7
Предоставляется	Плагин CNI Например, Calico, Antrea	Плагин CNI Например, Calico, Antrea
Интеграция с Kubernetes	YAML ресурс NetworkPolicy. Возможны и кастомные реализации CNI	YAML ресурс NetworkPolicy. Возможны и кастомные реализации CNI
Применение	YAML в etcd	YAML в etcd
Режимы работы	Как правило Whitelist. В кастомных реализациях возможен Blacklist	Как правило Whitelist. В кастомных реализациях возможен Blacklist
Сложность описания и использования	Средняя	Средняя
Значения по умолчанию	Нет	Нет
Влияние на производи-тельность	Среднее	Минимальное
Комментарий/ограничения	Низкая скорость обработки по сравнению с eBPF, но более простая в отладке	Высокая скорость обработки по сравнению с iptables

LUNTRY



Что такое Linux Security Module (LSM) и при чем тут eBPF

LSM и eBPF

- Linux Security Module (LSM)
- Extended Berkeley Packet Filter (eBPF)
- eBPF LSM - позволяет создавать собственные политики контроля доступа и безопасности без необходимости писать или загружать отдельные модули ядра
- eBPF ставит на LSM специальные хуки
- При срабатывании хука полученные данные можно сверить с установленной политикой и разрешить или запретить выполнение операции

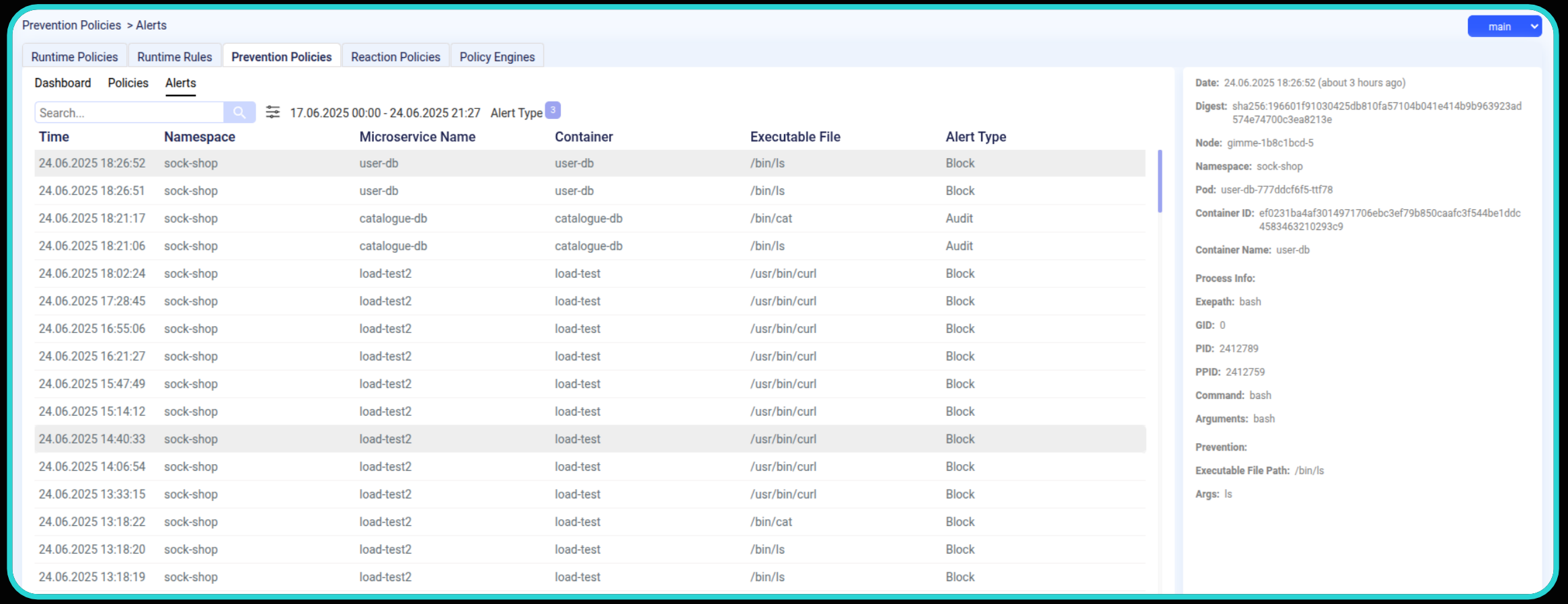
Технология	eBPF_LSM
Реализация	Luntry prevention политика
Краткое описание	Контроль на уровне процессов
Основная цель	Ограничить возможности контейнера для развития атаки и побега на хост
Контроль над	Запуском процессов
Версия ядра	5.7
Версия Kubernetes для GA	Не зависит
Предоставляется	Сторонними средствами защиты. Например, Luntry
Интеграция с Kubernetes	Не зависит
Применение	UI/API
Режимы работы	Whitelist, Blacklist
Сложность описания и использования	Низкая + возможность генерировать AppArmor профиль
Значения по умолчанию	Нет
Влияние на производительность	Минимальное
Комментарий/ограничения	Коммерческое решение

LUNTRY

LUNTRY

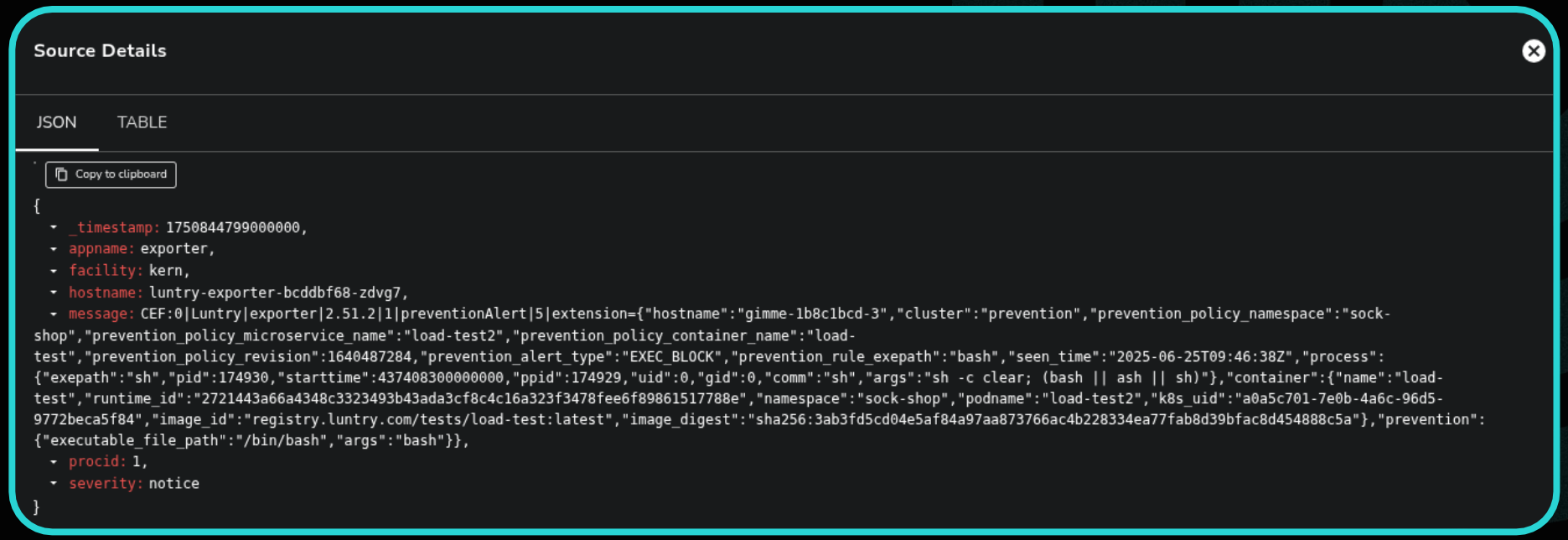
Как Luntry помогает решить задачи, связанные с предотвращением Runtime-угроз

Live Demo


Prevention Policies > Alerts
main
Runtime Policies
Runtime Rules
Prevention Policies
Reaction Policies
Policy Engines
Dashboard
Policies
Alerts
Search...
17.06.2025 00:00 - 24.06.2025 21:27
Alert Type 3
Time Namespace Microservice Name Container Executable File Alert Type
24.06.2025 18:26:52 sock-shop user-db user-db /bin/ls Block
24.06.2025 18:26:51 sock-shop user-db user-db /bin/ls Block
24.06.2025 18:21:17 sock-shop catalogue-db catalogue-db /bin/cat Audit
24.06.2025 18:21:06 sock-shop catalogue-db catalogue-db /bin/ls Audit
24.06.2025 18:02:24 sock-shop load-test2 load-test /usr/bin/curl Block
24.06.2025 17:28:45 sock-shop load-test2 load-test /usr/bin/curl Block
24.06.2025 16:55:06 sock-shop load-test2 load-test /usr/bin/curl Block
24.06.2025 16:21:27 sock-shop load-test2 load-test /usr/bin/curl Block
24.06.2025 15:47:49 sock-shop load-test2 load-test /usr/bin/curl Block
24.06.2025 15:14:12 sock-shop load-test2 load-test /usr/bin/curl Block
24.06.2025 14:40:33 sock-shop load-test2 load-test /usr/bin/curl Block
24.06.2025 14:06:54 sock-shop load-test2 load-test /usr/bin/curl Block
24.06.2025 13:33:15 sock-shop load-test2 load-test /usr/bin/curl Block
24.06.2025 13:18:22 sock-shop load-test2 load-test /bin/cat Block
24.06.2025 13:18:20 sock-shop load-test2 load-test /bin/ls Block
24.06.2025 13:18:19 sock-shop load-test2 load-test /bin/ls Block
Date: 24.06.2025 18:26:52 (about 3 hours ago)
Digest: sha256:196601f91030425db810fa57104b041e414b9b963923ad574e74700c3ea8213e
Node: gimme-1b8c1bcd-5
Namespace: sock-shop
Pod: user-db-777ddcf6f5-ttf78
Container ID: ef0231ba4af3014971706ebc3ef79b850caafc3f544be1ddc4583463210293c9
Container Name: user-db
Process Info:
Exepath: bash
GID: 0
PID: 2412789
PPID: 2412759
Command: bash
Arguments: bash
Prevention:
Executable File Path: /bin/ls
Args: ls

Возможности Luntry prevention политик

- Dashboard для политик
- Ручное и автоматическое генерирование политик
- Детализация: имя процесса, путь с wildcards, путь исполняемого файла, отношение родитель-потомок
- Режимы работы: Allow, Block, Audit (возможны некоторые сочетания)
- На базе eBPF_LSM (требуется ядро с 5.10)
- Полное управление из UI или через API
- Интеграция с SIEM и другими внешними системами





Выводы

Выводы

LUNTRY

01 Использование NetworkPolicy для контроля сети обязательно в Kubernetes

02 Использование AppArmor или решения на базе eBPF_LSM для контроля процессов обязательно

03 Далее уже можно подключать использование кастомного seccomp профиля, дефолтный можно/нужно использовать сразу

04 Luntry позволяет закрыть и работать со всеми аспектами предотвращения runtime угроз из одного интерфейса

LUNTRY

СПАСИБО ЗА ВНИМАНИЕ

ДМИТРИЙ ЕВДОКИМОВ
Founder & CTO Luntry

de@luntry.ru
@Qu3b3c
@k8security

@luntry_official
Подписывайтесь на наш Телеграм