LUNTRY

Безопасность контейнеров и Kubernetes для SOC

Дмитрий Евдокимов
Founder & CTO Luntry

Сергей Канибор
R&D

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++

Докладчик

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

Обо мне

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

R&D / Container Security в Luntry

Более 4 лет опыта в ИБ

Специализация – безопасность контейнеров и Kubernetes

Редактор ТГ-канала k8s (in) security

Деятельность

Багхантер

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

Спикер крупнейших ИТ и ИБ конференций

VK Kubernetes	БеКон
DevOpsConf	DevOops
OFFZONE	PHDays

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/

В процессе получения сертификата ФСТЭК

Теория


50:50
x2
15 3 000 000
14 1 500 000
13 800 000
12 400 000
11 200 000
10 100 000
9 50 000
8 25 000
7 15 000
6 10 000
5 5 000
4 3 000
3 2 000
2 1 000
1 500
Почему вы можете не знать, что злоумышленник проводит вредоносную активность в вашем Kubernetes?
A: Злоумышленник проэксплуатировал неизвестную 0day уязвимость
B: Злоумышленник успел проэксплуатировать не закрытую 1day уязвимость и залил неизвестный вредоносный код
C: Злоумышленник встроил backdoor в библиотеку, что используется в образе
D: Непонятно, что происходит внутри микросервисов

Контейнеры и Kubernetes 101

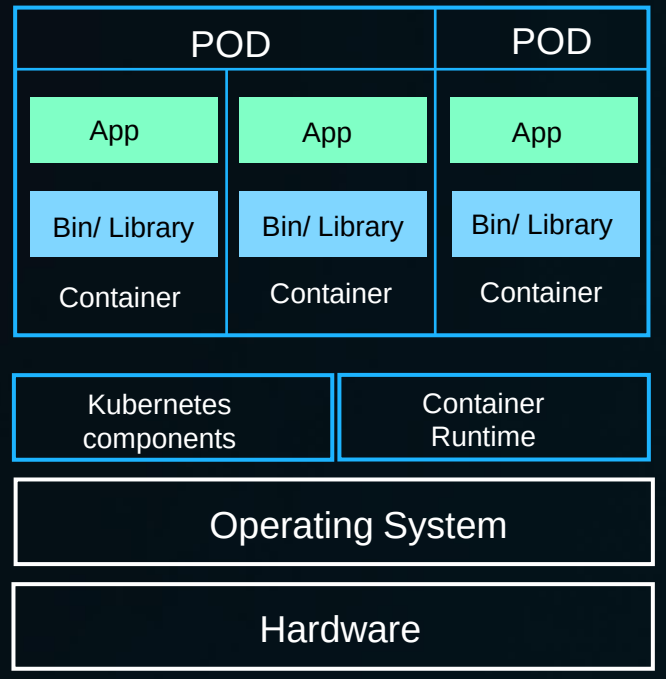
```
root     2966156  0.0  0.0 110128  5932 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2966174  0.0  0.0   1020     4 ?        Ss   Nov19   0:00  |   \_ /pause
root     2966375  0.0  0.0 108720  6356 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
sadm     2966394  0.0  0.0 827512 19728 ?        Ssl  Nov19   0:00  |   \_ node /usr/bin/nodemon /src/index.js
sadm     2966421  0.0  0.0   4460    80 ?        S    Nov19   0:00  |       \_ sh -c node /src/index.js
sadm     2966422  0.0  0.0 967396 16596 ?        Sl   Nov19   0:00  |           \_ node /src/index.js
root     2988902  0.0  0.0 108720  5408 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2988922  0.0  0.0   1020     4 ?        Ss   Nov19   0:00  |   \_ /pause
root     2989066  0.0  0.0 108720  5408 ?        Sl   Nov19   0:26  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2989099  0.0  0.0  31000 23956 ?        Ss   Nov19   0:42  |   \_ /usr/local/bin/python /usr/local/bin/gunicorn -b :8080 --workers 1 --threads 1 --
root     2989116  0.3  0.1 142092 48964 ?        Sl   Nov19  16:50  |       \_ /usr/local/bin/python /usr/local/bin/gunicorn -b :8080 --workers 1 --threads
root     2989333  0.0  0.0 110128  5404 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2989352  0.0  0.0   1020     4 ?        Ss   Nov19   0:00  |   \_ /pause
root      596808  0.0  0.0 110128  6316 ?        Sl   Nov20   0:06  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root      596827  0.0  0.0   1020     4 ?        Ss   Nov20   0:00  |   \_ /pause
root      598309  0.0  0.0 110128  6224 ?        Sl   Nov20   0:07  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root      598334  1.4  5.5 7236340 1832196 pts/0 Ssl+ Nov20  39:39      \_ /docker-java-home/bin/java -Djava.util.logging.config.file=/opt/atlassian/conflue
root      599854  1.0  1.3 7007820 427956 pts/0  Sl+  Nov20  28:11      |   \_ /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -classpath /opt/atlassian/conf
root      701694  0.0  0.0   4288   764 ?        Ss+  Nov20   0:00      \_ /bin/sh
```

Container – это Linux process
с определёнными свойствами/ограничениями

Kubernetes – это платформа (оркестратор) для управления контейнерами

File -> Image

Process -> Container -> Pod -> ReplicaSet -> Deployment



Проблемы SOC и DFIR в Kubernetes

1. Обнаружить инцидент в контейнере
2. Традиционные инструменты SOC не понимают контейнеры
3. Жесткие требования к потреблению ресурсов
4. Динамическое окружение
5. Специфика сетевого взаимодействия в k8s
6. Множество уровней абстракций

IR в Kubernetes

Incident Response

While incident response is primarily a human process, in this section we discuss how Kuberenetes policy management impacts it. Kubernetes incident response should be aligned with established DevSecOps operating principles with an emphasis on recognizing the declarative state, ephemeral nature of workloads, and automated controls.

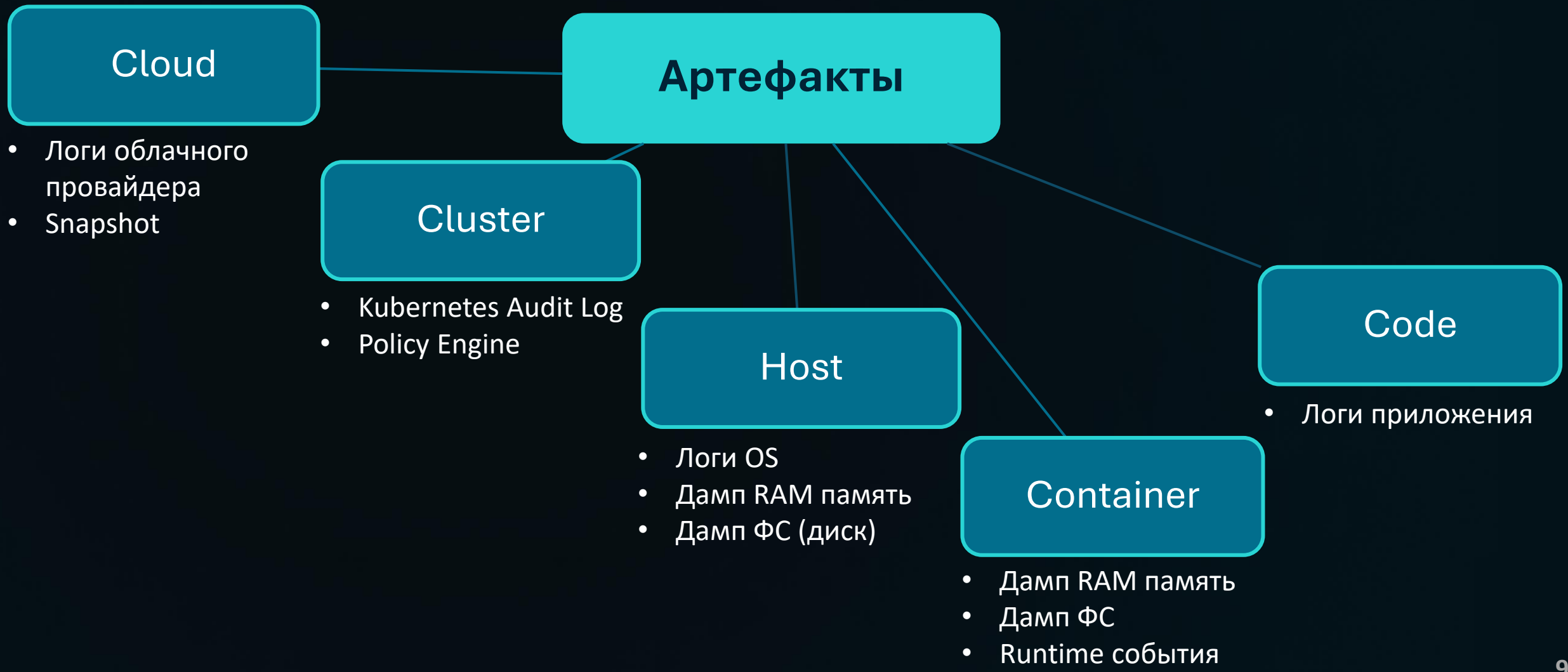
Kubernetes and cloud native technologies introduce new challenges for planning incident response. The volume of telemetry data required to effectively identify and detect attacks is larger due to the short lifespan of containers, and since the persistence of resources is not guaranteed. Telemetry and audit logs need to be ingested and processed automatically instead of manual review and enrichment so that automation workflows can manage and respond to operational status changes in the infrastructure by extracting actionable events out of raw telemetry data. Existing SIEM and SOAR platforms may not be up to the challenge, having focused on manual human operations.

Increasing adoption of Chaos Engineering into Kubernetes incident response planning and simulation helps surface new threats and design better monitors and telemetry ingestion flows. ML-based telemetry analysis can help proactively identify anomaly scenarios and edge cases. It is increasingly important to build automated remediation, using policy-as-code, and to curate and train ML models so that these tools adapt as attackers evolve. Kubernetes policy reports can provide additional data, with long term data collected and stored in the PAP.

"Kubernetes Policy Management"

Источники данных

(следы и артефакты)



- Логи облачного провайдера
- Snapshot

- Kubernetes Audit Log
- Policy Engine

- Логи OS
- Дамп RAM память
- Дамп ФС (диск)

- Дамп RAM память
- Дамп ФС
- Runtime события

- Логи приложения

Модели нарушителя

ВНЕШНИЙ

Попадает внутрь контейнера Pod и оттуда обращается к рядом стоящим сервисам или делает побег из контейнера на Node

ВНУТРЕННИЙ

Злоумышленник с Node или из одной сети с Kubernetes

СКОМПРОМЕТИРОВАННЫЙ РАЗРАБОТЧИК

Контролирует/формирует содержимое YAML ресурсы

Runtime события и матрица угроз

"Threat Matrix for Kubernetes " от Microsoft

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Impact
Using cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access Kubernetes API server	Access cloud resources	Images from a private registry	Data destruction
Compromised image In registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account	Collecting data from pod	Resource hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from proxy server	Application credentials in configuration files	Exposed sensitive interfaces	Application credentials in configuration files		
Exposed sensitive interfaces	SSH server running inside container	Container service account			Access managed identity credentials	Instance Metadata API	Writable hostPath mount		
	Sidecar injection	Static pods			Malicious admission controller		CoreDNS poisoning		
							ARP poisoning and IP spoofing		

- то, что можно обнаружить на основе runtime событий из контейнера

Подходы к обнаружению в Runtime

СИГНАТУРНЫЙ

Предупреждение при совпадении с тем или иным правилом из базы знаний

ПОВЕДЕНЧЕСКИЙ

Фиксирование аномалий при отхождении от модели поведения

ГИБРИДНЫЙ

Сочетание сигнатурного и поведенческого анализа

Live Demo

Домен безопасности Runtime в Luntry

Уникальная черта

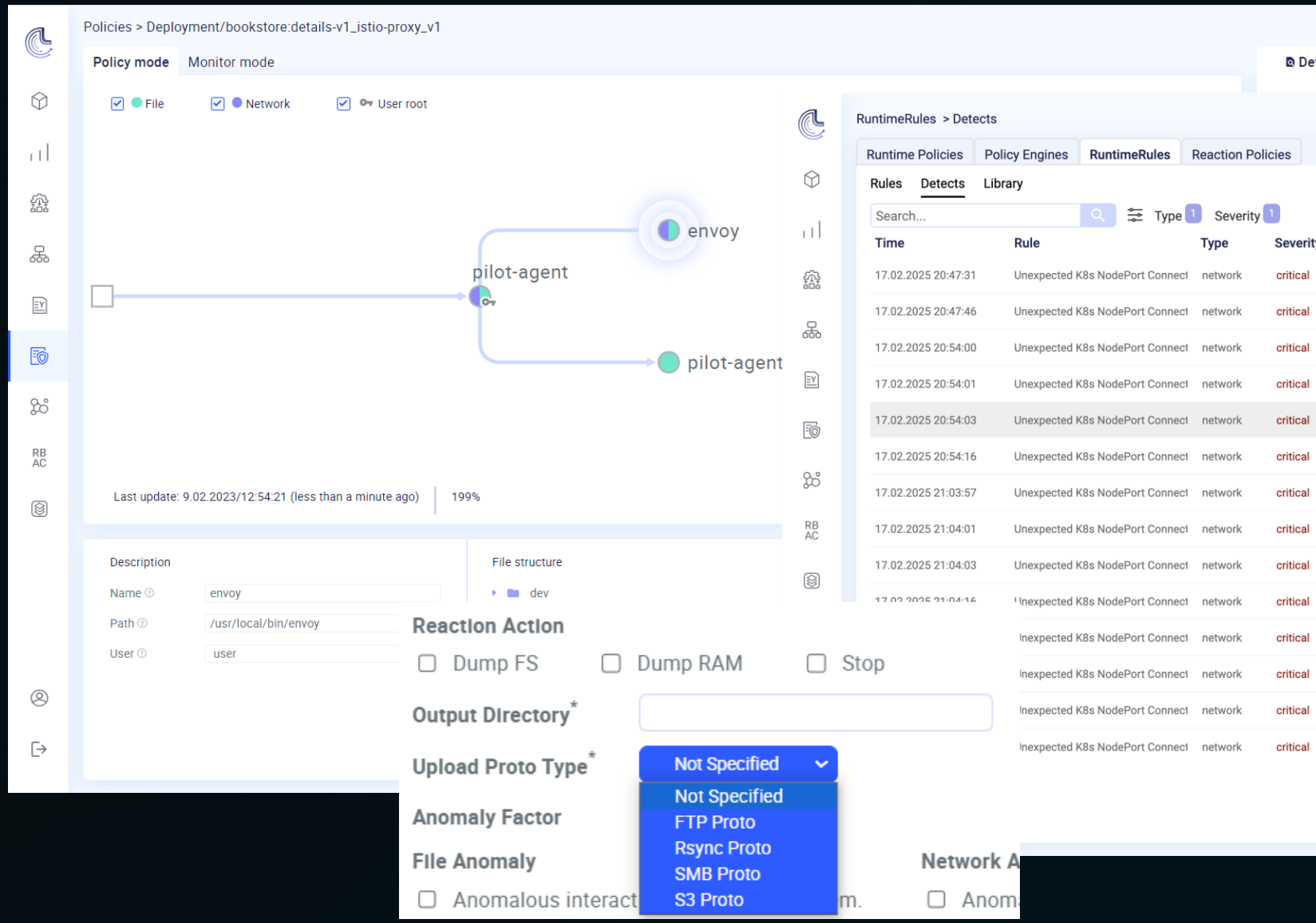
- Гибридный подход (поведение + правила)
- FIM для процессов
- Сбор артефактов для расследования
- Анализ на server-side

Что это дает

- Минимальное количество ложный срабатываний
- Возможность проводить форензику

Что без этого

- Очень сложное внедрение в процессы работы



Полезные материалы

1. Форензика для контейнеров и контейнерных инфраструктур
   CyberCamp 2025
2. Ловим злоумышленников и собираем улики в контейнерах Kubernetes
   Вебинар Luntry 2024
3. Kubernetes Audit Log в арсенале SOC
   SOC Forum 2024
4. Экскурсия по матрицам угроз для контейнеров и Kubernetes
   VK Kubernetes Conf 2023
5. SOC в контейнерах
   SOC Forum 2023
6. EDR vs Containers: актуальные проблемы
   SOC Forum 2023

Спасибо за внимание!

Дмитрий Евдокимов

Founder&CTO Luntry

Email: de@luntry.ru

Twitter: @evdokimovds

@Qu3b3c

Channel: @k8security

Site: www.luntry.ru

luntry.ru

k8security

luntrysolution