2 ИЮНЯ 2026, МОСКВА, ЛОФТ ГОЭЛРО

БЕКОН LUNTRY '26

ЕДИНСТВЕННАЯ КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ
КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

Контейнеры против майнеров или история по криптоджекинг в Kubernetes

Максим Князев | К2 Кибербезопасность

БЕКОН'26

Контейнеры против майнеров

или история про криптоджекинг в Kubernetes



Максим Князев
Старший системный инженер
K2 кибер безопасность

Увлекательная статистика

$723,4 млрд

мировые расходы на публичные облачные сервисы в 2025 году

Gartner

+399%

рост детектов несанкционированного использования вычислительных ресурсов

Symantec / Palo Alto Networks

Информация о спикере

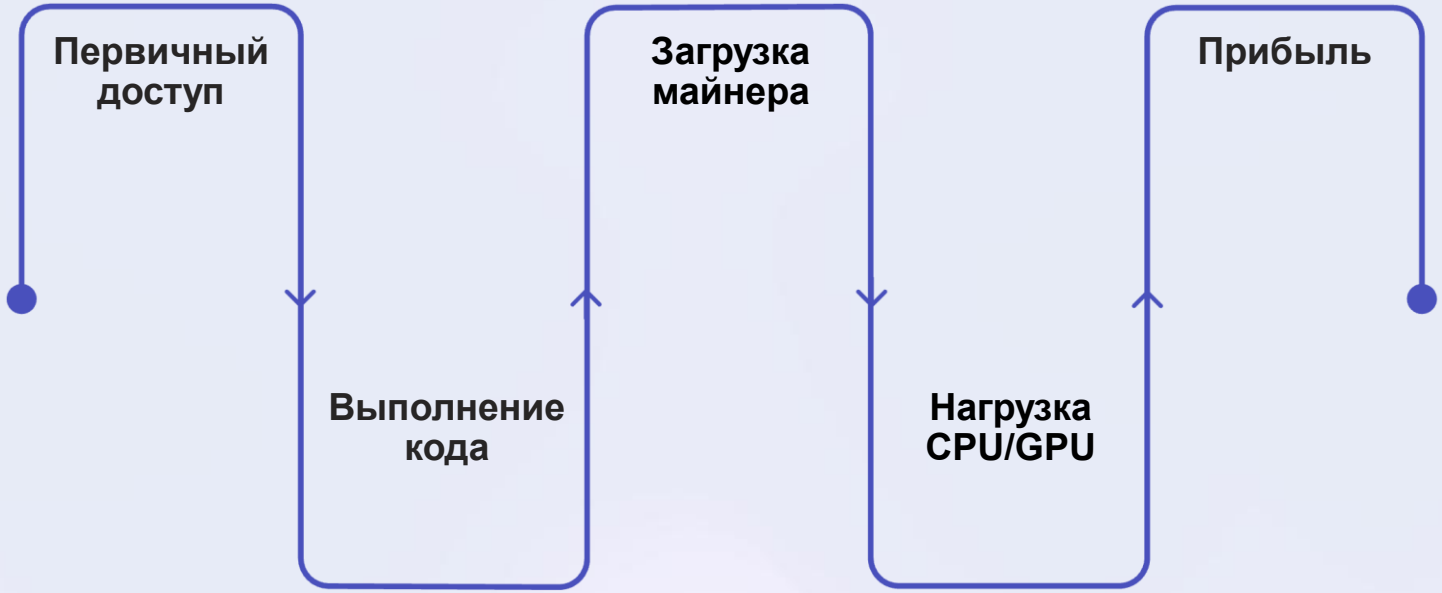
БЕКОН'26

- Старший системный инженер в К2 Кибербезопасность
- Аспирант РТУ МИРЭА
- Специалист в области AppSec, DevSecOps и защиты информации
- Занимаюсь внедрением инструментов SAST, DAST, SCA и контейнерной безопасности
- Исследую безопасность IoT и персонального Интернета вещей
- Автор статей на Хабре и в научных журналах ВАК / Scopus
- Спикер профильных и научных конференций
- Автор Telegram-канала об информационной безопасности и IoT



Что такое криптоджекинг

Криптоджекинг — это монетизация чужой инфраструктуры.
Несанкционированное использование вычислительных ресурсов жертвы для майнинга криптовалюты



Kubernetes API
Точка входа в кластер

Containers / Pods
Носитель вредоноса

ServiceAccount
Привилегии для lateralmovement

Network egress
Связь с mining pool

Cloud resources
Автоскейлинг = ∞ вычислений

Старый сценарий был слишком простым

Классическая цепочка

1. Компрометация сервера
2. Запуск XMRig
3. CPU → 100%
4. kill process

Почему этого больше недостаточно

В такой модели достаточно обычного мониторинга ресурсов и ручной проверки процессов. Он всё ещё встречается, однако на нём нельзя строить современную модель защиты

```
xmrig  1800m  CPU
```

⚠ Злоумышленник в 2026 году так шуметь не будет

Почему в Kubernetes все сложнее

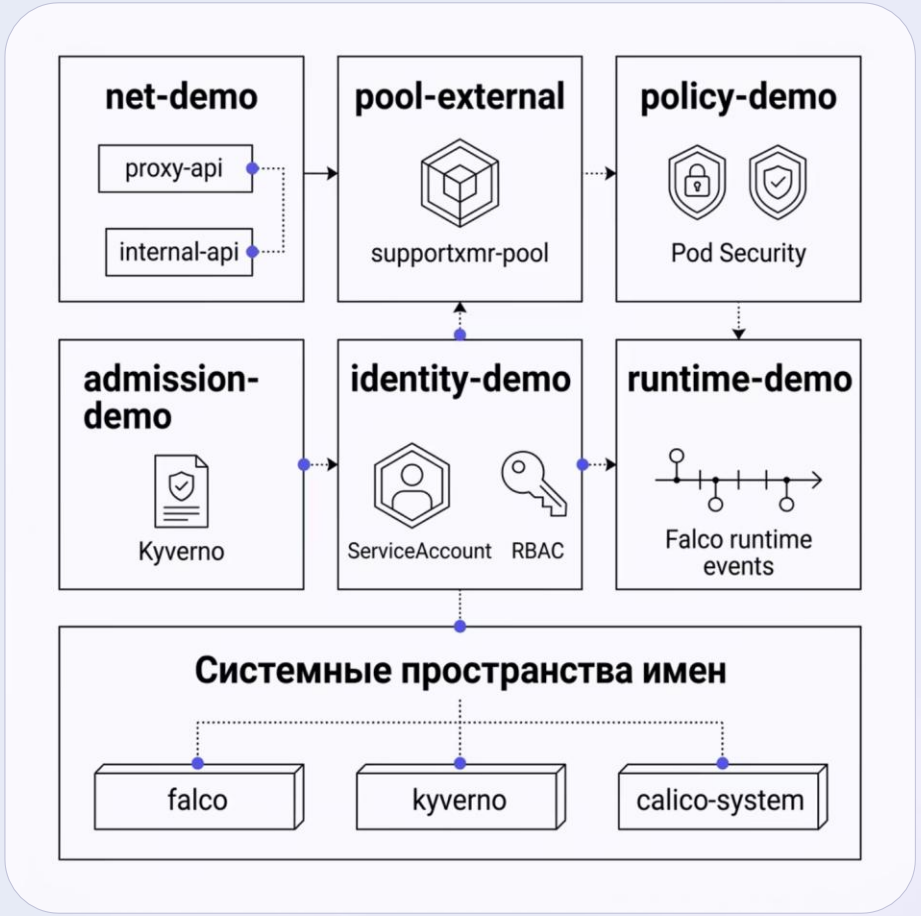


Что скрыто внутри

CPU показывает **симптом**. Контекст объясняет, **почему этот симптом опасен**

- Какой объект создал Pod: Deployment, Job, CronJob или DaemonSet
- Какой ServiceAccount используется и его RBAC-права
- Сработали ли политики admission control
- Какой egress разрешён сетевыми политиками
- Что происходит внутри контейнера во время выполнения

Архитектура стенда



⚠ На стенде могут использоваться и другие инструменты

kind-кластер cryptojacking-net-lab

Каждый компонент демонстрирует отдельный слой защиты

Детектив: собираем улики



Кадр из телефильма «Приключения Шерлока Холмса и доктора Ватсона: Собака Баскервилей», реж. Игорь Масленников, к/с «Ленфильм», Гостелерадио СССР, 1981

Эпизод 1: Когда все очевидно

Метрики

NAME	CPU	MEMORY
monero-worker	1800m	40Mi

Логи майнера

```
miner speed=1380 H/s shares=1/1
new job from pool...
```

Признаки в этом сценарии

- Говорящее имя workload
- Явные логи майнера
- Сетевой endpoint mining pool
- Видимые признаки в CLI

Такой сценарий легко ловится мониторингом ресурсов.
Но злоумышленник не обязан быть таким заботливым

Эпизод 2: Маскировка под системный компонент

Как это выглядит

```
Deployment:   proxy-api
Container:    pause
Image:        platform/runtime-helper:1.0.1
Process:      kworker/u8:2
Namespace:    net-demo
```

```
proxy-api-864b957669-h2h69
  CPU: 979m  MEM: 15Mi
```

⚠ На первый взгляд выглядит как служебный компонент платформы

Реальные кейсы маскировки

Подобные техники задокументированы в разных кампаниях:

Имя процесса kworker/u8:2 имитирует ядро Linux как стандартная техника

CrowdStrike: Dero campaign

DaemonSet с именем proxy-api

Wiz: DERO

Майнер назывался pause, pool зашит в бинарь

Эпизод 3: CPU как симптом

Что видит metrics-server

Workload	CPU
internal-api	0m
proxy-api	979m

Что metrics-server не знает

Мониторинг полезен как **первый слой**,
но сам по себе дает мало информации

> Сигнал есть. **Контекста нет**

Происхождение

Почему этот workload появился в кластере

Идентичность

ServiceAccount и его RBAC-права

Доверие

Является ли образ доверенным

Поведение

Куда подключается и что делает внутри

Эпизод 4: Сетевой слой

БЕКОН'26

Mining pool в стенде

Параметр	Значение
Namespace	pool-external
Service	supportxmr-pool
Port	3333/TCP
Hostname	pool.supportxmr.test

```
nc: connect to pool.supportxmr.test
port 3333 timed out
```

NetworkPolicy как preventive control

До

proxy-api свободно подключается к pool.supportxmr.test:3333

После

default-deny egress блокирует соединение — таймаут

NetworkPolicy не понимает, что это майнер.
Она просто **не даёт workload ходить куда не надо**

Default-deny egress — один из самых эффективных контролей против криптоджекинга в Kubernetes.

Эпизод 5: Разрешен только нужный трафик

В namespace net-demo применены три политики:
default-deny-egress, allow-dns-egress, allow-internal-api.

✅ curl internal-api

`HTTP/1.1 200 OK` — легитимный трафик проходит

🚫 nc pool.supportxmr.test 3333

`timed out` — egress к майнинг-пулу заблокирован

Защита не должна ломать легитимную функцию,
а только ограничивать поведение

Эпизод 6: Runtime-артефакты


```
{
  "pool": "pool.supportxmr.test:3333",
  "algo": "rx/0",
  "agent": "xmrig/6.22.2"
}
```

config.json

pool, algo rx/0, wallet, agent xmrig/6.22.2

kworker/u8:2

Процесс с нормально выглядящим именем

~1000m CPU

Устойчивая нагрузка + логи H/s и shares

Эпизод 7: Pod Security

В namespace policy-demo включён restricted-профиль: enforce, audit и warn.

🚫 Набор блокирующих правил

```
violates PodSecurity
"restricted:latest":
- hostPID=true
- privileged=true
- hostPath=/
- no allowPrivilegeEscalation=false
- no drop: ALL
- no runAsNonRoot=true
- no seccompProfile
```

Что проверяет restricted-профиль

→ **Запрет hostPID и privileged**
Изоляция от хост-процессов

→ **Запрет hostPath монтирования**
Нет доступа к файловой системе узла

→ **Обязательный drop: ALL**
Минимум привилегий по capabilities

Pod Security не ловит майнинг напрямую, но он режет то, что усиливает атаку

Эпизод 8: Policy Engine

БЕКОН'26

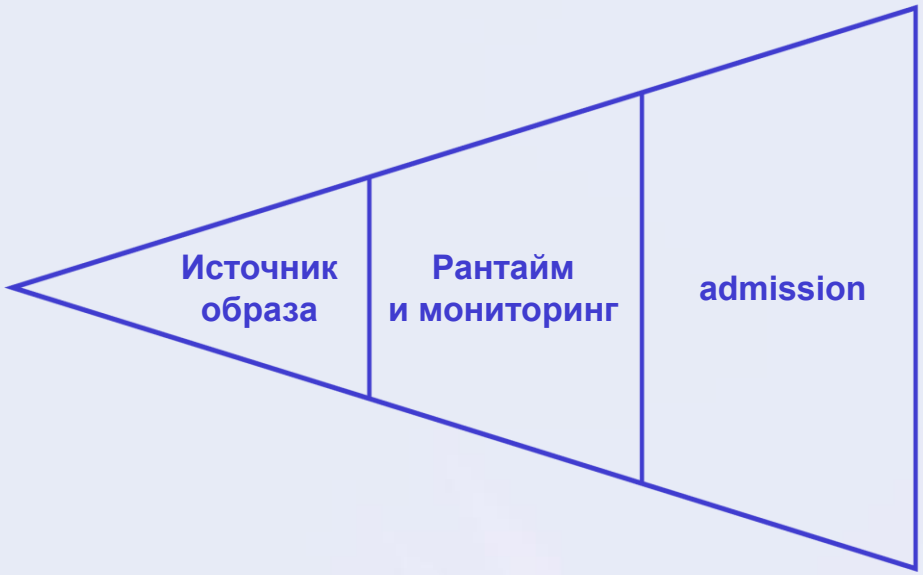
Активные политики: запрет latest-тегов и ограничение источников образов.
Policy Engine проверяет workload на этапе admission до запуска контейнера

Image	Политика	Результат
`alpine:latest`	disallow-latest-tag	❌ denied
`docker.io/randomuser/analytics-helper:1.0.0`	allowed-image-sources	❌ denied
`alpine:3.20`	allowed-image-sources	✅ allowed

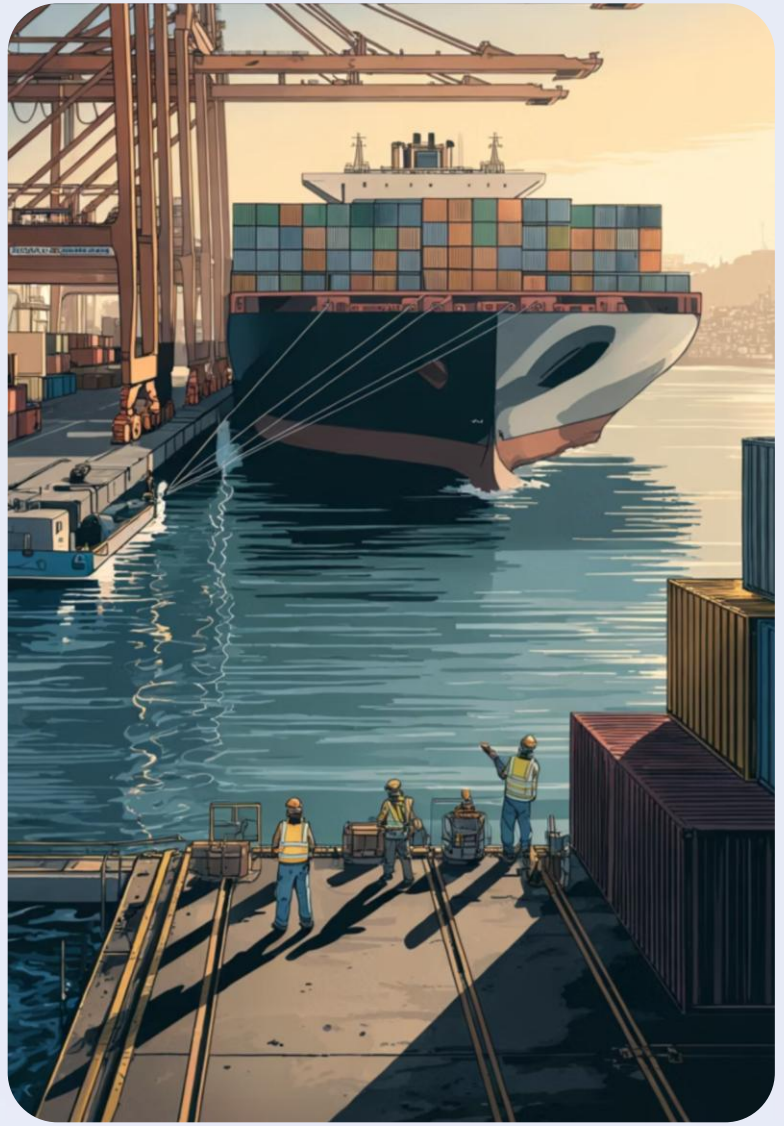
⚠ Kyverno видит риск. Но если подозрительное поведение проявилось внутри уже запущенного контейнера, понадобится runtime layer

Эпизод 9: Что с цепочкой поставок?

Опасный workload не называется miner. Он маскируется под легитимные компоненты: analytics-helper, node-cache, proxy-api, diagnostics.



Защита: запрет latest, доверенные registry, image digest, signing, SBOM и сканирование до запуска



Эпизод 10: RBAC

В namespace identity-demo создан ServiceAccount web-api-sa с минимальными правами. Даже при компрометации workload ущерб ограничен

Действие	Результат
list pods в identity-demo	✅ yes
get pods в identity-demo	✅ yes
list secrets	❌ no
create pods	❌ no
list pods cluster-wide	❌ no

Принцип минимальных привилегий

RBAC не блокирует CPU-нагрузку и не видит процессы. Но он ограничивает то, что скомпрометированный workload может сделать с кластером

ⓘ Минимальные права = минимальный ущерб при взломе

Эпизод 11: ServiceAccount Token

БЕКОН'26

web-api-token-probe

`service account token path exists`

Токен доступен изнутри контейнера — потенциальный вектор атаки на Kubernetes API

web-api-token-disabled

`automountServiceAccountToken: false`

`token path not found` — риск злоупотребления API устранён

Почему это важно

CPU spike — поздний сигнал. Обращение к ServiceAccount token и Kubernetes API может появиться раньше и показывает, что workload пытается понять свои права внутри кластера

⚠ Отключайте automountServiceAccountToken там, где токен не нужен. Это снижает риск без потери функциональности

Эпизод 12: Falco

Shell spawned

```
Warning Shell spawned inside application containerPod=web-api-runtime · Namespace=runtime-demo
```

Token access

```
Warning Suspicious ServiceAccount token accessДоступ к /var/run/secrets/kubernetes.io/serviceaccount/token
```



Falco не знает о NetworkPolicy или admission-политиках. Но он единственный, кто видит runtime-поведение уже запущенного контейнера

Корреляция превращает алерты в расследование

БЕКОН'26

Слой	Что видит	Чего не видит
metrics-server	High CPU (~1000m)	Контекст атаки
NetworkPolicy	Блокировку egress	Процессы внутри
Kyverno	Admission-риск	Runtime-поведение
Pod Security	Опасный Pod spec	Low-priv майнер
RBAC	Blast radius identity	CPU и процессы
Falco	Runtime behavior	Admission и egress

Resource + Network + Admission + Pod spec + Identity + Runtime = Инцидент

Корреляция сигналов превращает изолированные алерты в полноценное расследование

Open Source инструменты

Ни один инструмент не закрывает всю цепочку атаки

1	**Resource visibility** metrics-server, Prometheus, Grafana	Не объясняет причину.
2	**Network / Egress control** NetworkPolicy, Calico, Cilium	На видит процесс внутри контейнера
3	**Admission / Policy-as-Code** Kyverno, OPA Gatekeeper	Не видит уже запущенный контейнер
4	**Supply chain / Image security** Trivy, Grype, Cosign, Syft	Чистый образ может быть скомпрометирован в рантайме
5	**Resource visibility** Falco, Tetragon	Требует качественно написанных правил и понимания работы ядра линукс
6	**Resource visibility** Kubescape, kube-bench, kube-hunter	Оценка состояния, не runtime response



Open source закрывает большую часть сценариев, но требует интеграции, настройки правил и корреляции сигналов

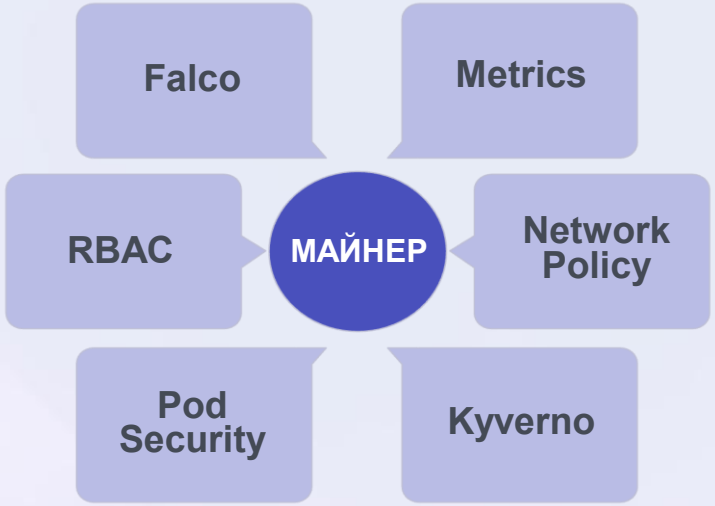
Главный вывод

Криптоджекинг в Kubernetes — это не один контейнер с майнером, а цепочка компрометации

Шесть сигналов по отдельности

- **Metrics / CPU** — показывает симптом
- **NetworkPolicy / Egress** — ограничивает outbound
- **Kyverno / Admission** — снижает supply chain риск
- **Pod Security** — режет опасные спецификации
- **RBAC / Identity** — ограничивает blast radius
- **Falco / Runtime** — видит поведение внутри pod

Результат корреляции



Без корреляции
Шесть разрозненных алертов, каждый без контекста

С корреляцией
Один инцидент с полной цепочкой компрометации

Доверяй, но проверяй

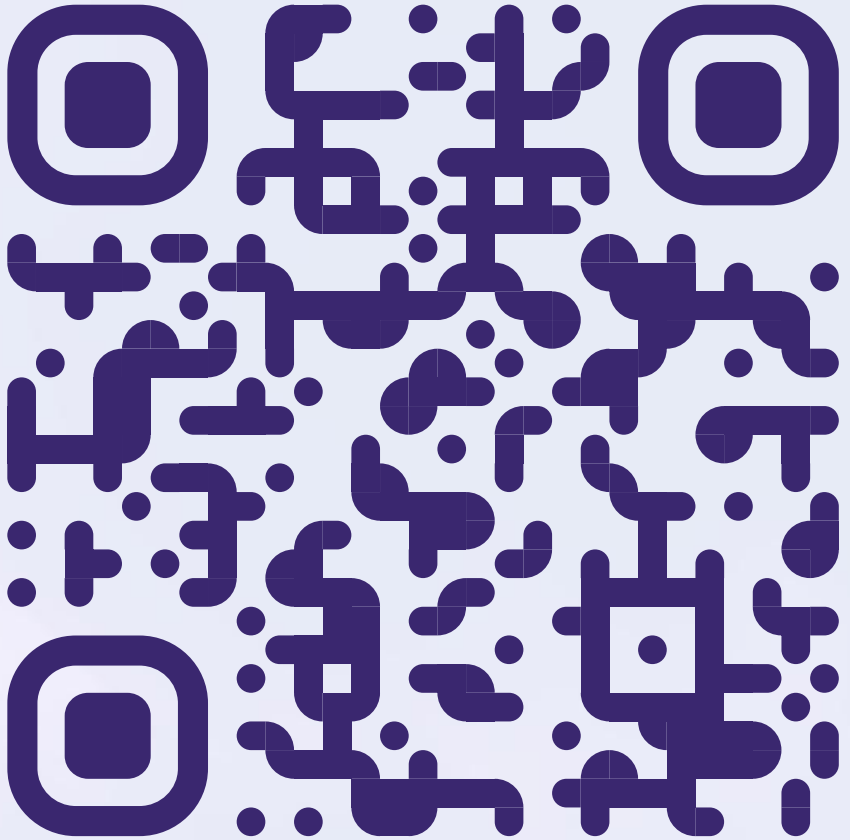
Все ссылки на источники и архив
со скриншотами работы стенда
на всех сценариях



Telegram-канал

Там я публикую:

- Разборы уязвимостей и новых атак
- Практические советы по AppSec/DevSecOps
- Новости из мира кибербезопасности
- Последние исследования в области безопасности IoT



БЕКОН'26

Спасибо

за совместное расследование!

