

K8S SECURITY · CHUYÊN GIA CHIA SẺ

Giải mã kỹ thuật tấn công **Kubernetes** qua lăng kính **MITRE ATT&CK**

LÀM SAO ĐỂ NGĂN CHẶN

Nguyễn Thế Thành

DevOps Mentor



Chúng ta sẽ đi qua những gì

0



Bối cảnh & mục tiêu tấn công

0



Hai kịch bản tấn công thực tế

0



Misconfiguration vs
Zero-day

1

2

3

0



MITRE ATT&CK for Containers

0



Chiến thuật phòng thủ theo
lớp

0



Incident Response Workflow

4

5

6

01

Bối cảnh & mục tiêu tấn công

Vì sao Kubernetes trở thành đích ngắm hàng đầu



BỐI CẢNH

Kubernetes đã ở khắp nơi

Và chính điều đó khiến nó trở thành mục tiêu



82%

người dùng container chạy
Kubernetes trong production

Tăng từ 80% (2024) và 66% (2023) —
CNCF Annual Cloud Native Survey 2025



98%

tổ chức đã áp dụng
cloud native ở mức nào đó



66%

tổ chức chạy GenAI dùng
Kubernetes để điều phối
inference workload



“Khi một nền tảng đạt mức phổ cập đó, nó trở thành **‘chuẩn tấn công’** —
kỹ thuật viết một lần dùng được cho hàng nghìn cluster.”

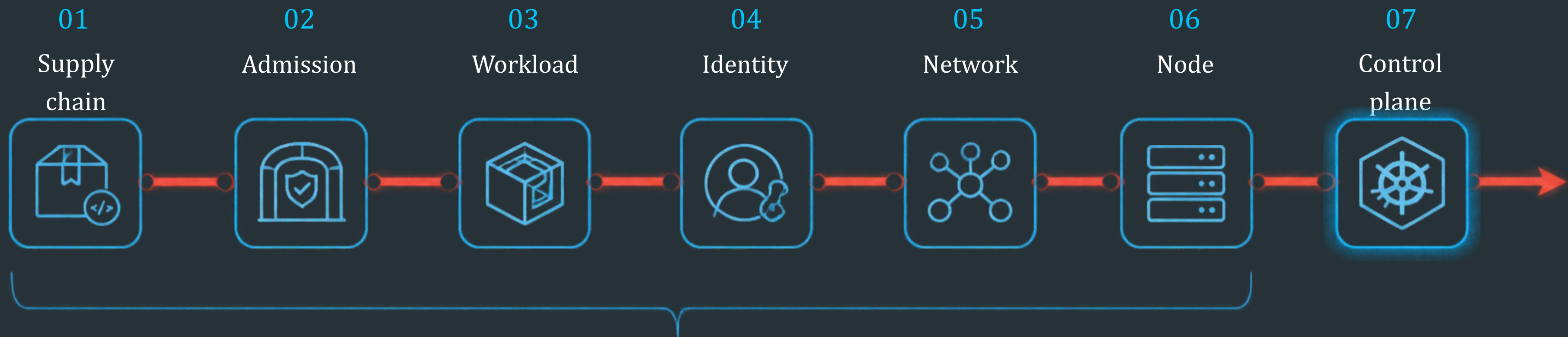
Vì sao Kubernetes là mục tiêu giá trị cao

KUBERNETES CHARACTERISTIC	ATTACKER VALUE
 Control plane điều khiển toàn bộ workload	Chiếm cluster-admin = chiếm mọi ứng dụng cùng lúc
 Mật độ credential cực cao	SA token, Secret, kubeconfig, cloud IAM đều nằm trong etcd
 Mọi thứ đều là API	Chỉ cần token hợp lệ + curl là điều khiển được
 Mặc định thiên về tốc độ, không thiên về an toàn	Pod-to-pod mở hết, SA token tự mount, không NetworkPolicy

BỐI CẢNH

Bản đồ bề mặt tấn công Kubernetes

Khung xuyên suốt bài trình bày — *attacker đi từ trái sang phải*



6 trong 7 lớp là do đội DevOps/platform cấu hình — không phải do Kubernetes quyết định



Đó là lý do phần lớn sự cố K8s là lỗi cấu hình chứ không phải lỗ hổng của chính Kubernetes

Bảo mật **Kubernetes** khác gì hạ tầng truyền thống

01



Vòng đời phù du

Pod sống vài phút — attacker
xoá dấu vết cùng lúc
container biến mất

02



Chia sẻ kernel

Container không phải máy ảo
— **privileged** hoặc **hostPath** là
đường ngắn xuống node

03



Định danh thay cho mạng

Firewall/IP không còn là
biên giới — **RBAC + SA**
token mới là biên giới thật

04



Không còn perimeter

Mặc định mọi pod nói chuyện
được với mọi pod — **lateral**
movement là “bình thường”

BỐI

CẢNH

Hậu quả đo được

State of Kubernetes Security — Red Hat



46%

tổ chức từng **mất doanh thu hoặc khách hàng** vì
sự cố bảo mật liên quan container/Kubernetes



67%

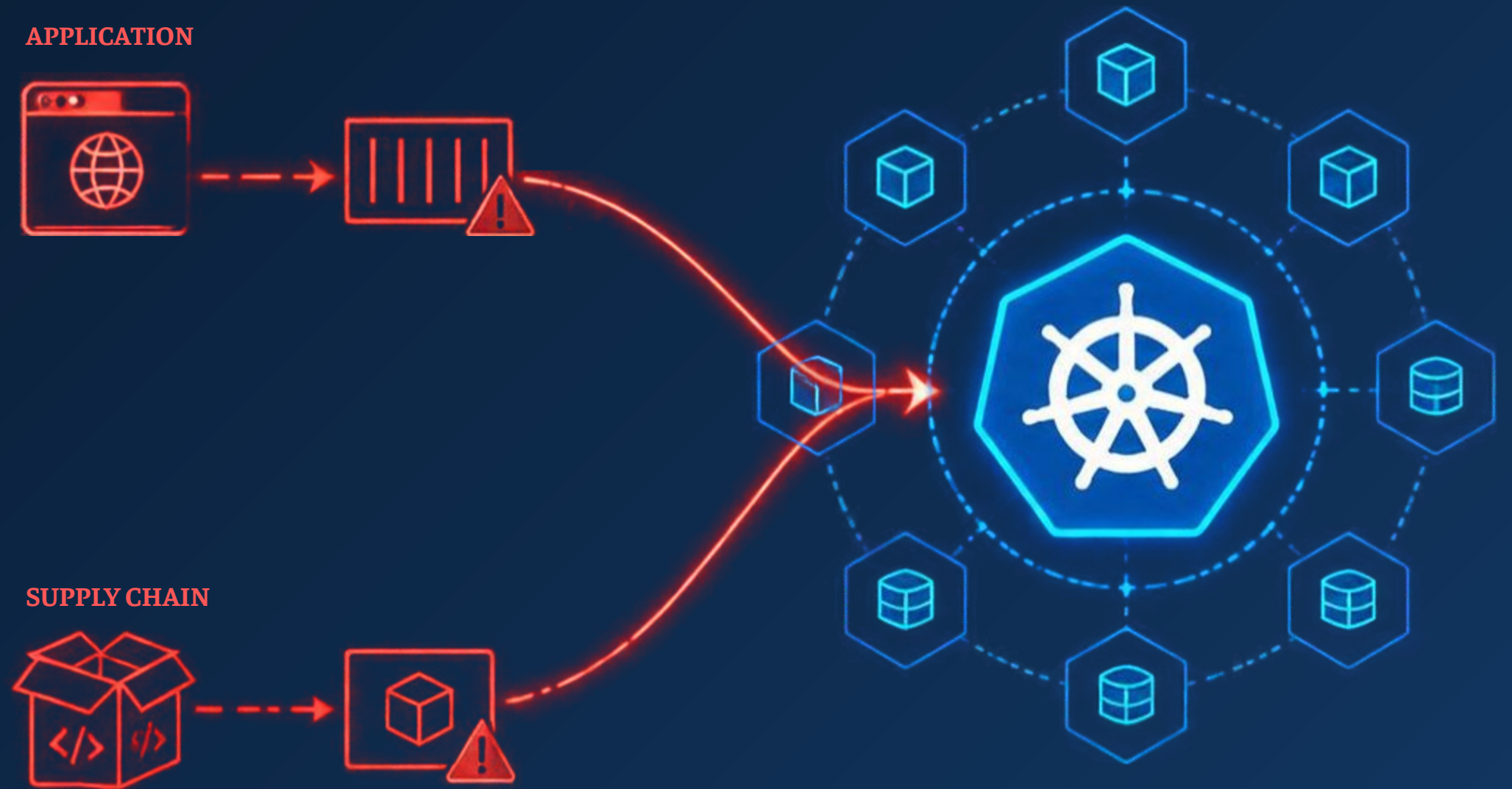
tổ chức đã phải **hoãn hoặc làm chậm triển khai**
vì lo ngại bảo mật

Bảo mật K8s không phải chi phí làm chậm delivery —
chính việc **thiếu nó** mới là thứ đang **làm chậm delivery**.

02

Hai kịch bản tấn công thực tế

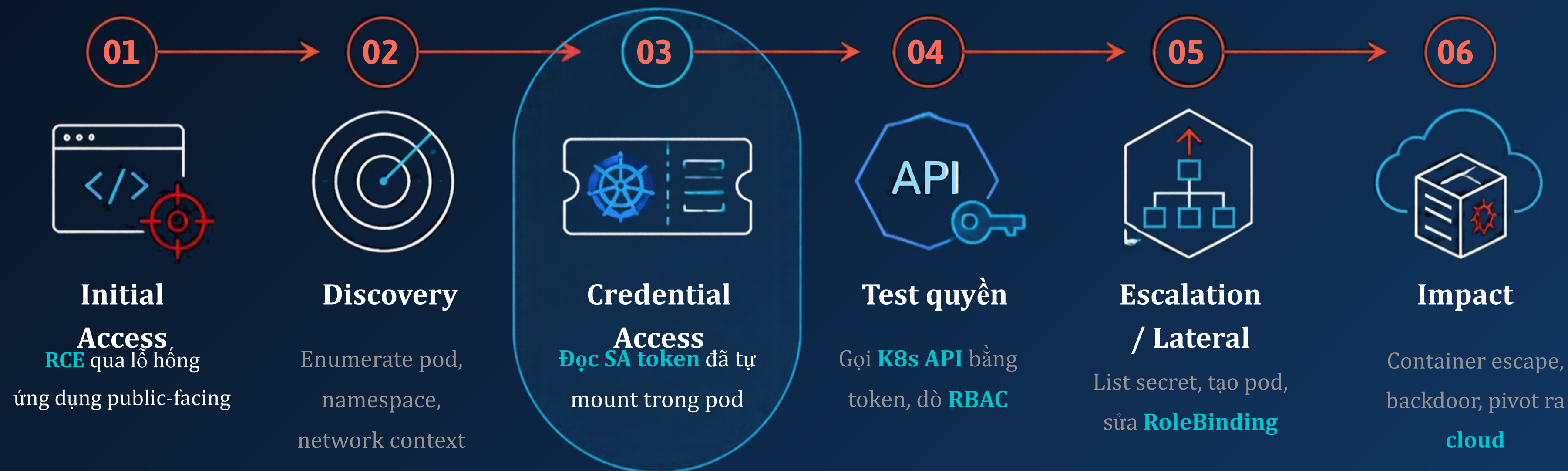
Chiếm cluster qua ứng dụng & supply chain attack



KỊCH BẢN

1 Chiếm cluster qua lỗ hổng ứng dụng

Phần thưởng không phải là container — phần thưởng là **ServiceAccount token**



Mỗi bước đều là **hành vi hợp lệ** của Kubernetes — **không có 'lỗ hổng zero-day'** nào ở đây

KỊCH BẢN 1

Con số & case study thực tế



282 %

tăng trưởng hoạt động
tấn công liên quan K8s
(Unit 42, 4/2026)



22%

môi trường cloud có dấu hiệu
nghĩ ngờ **đánh cắp SA token**
(2025)

CASE STUDY

01



React2Shell — CVE-2025-55182

insecure deserialization trong React Server Components. Attacker thực thi lệnh trong container chỉ **2 ngày** sau khi CVE công bố.

02



Incident sàn crypto — Unit 42, 2025

token của **SA quyền cao** dẫn tới
chiến hạ tầng tài chính và
đánh cắp **hàng triệu USD**.

KỊCH BẢN 2

Supply chain attack vào container

Mã độc đi thẳng vào production qua chính **con đường tin cậy** của bạn

01 TYPOSQUATTING

Microsoft, 5/2026: **14 package** giả
OpenSearch/ElasticSearch thu **AWS creds** +
Vault token

03 WORMABLE PROPAGATION

Shai-Hulud & biến thể — worm sinh ra **hàng**
trăm package

02 COMPROMISED PACKAGE

node-ipc (10M downloads/tuần) bị đẩy **3**
version độc

04 CI/CD CREDENTIAL THEFT

Megalodon: **5.718-commit** workflow đọc / **5.561 repo**
trong **6 giờ**



Điều khiển supply chain attack đặc biệt nguy hiểm

01



01 — Đi qua **đường tin cậy**

không có guardrail nào ở biên phát hiện được vì package/image do chính bạn chủ động cài

02



02 — Chạy ở thời điểm build, với quyền của pipeline

payload chiếm luôn **AWS credential, Vault token, OIDC token** trong CI runner

03



03 — Provenance không còn là bảo chứng tuyệt đối

package độc từng mang chữ ký **Sigstore** hợp lệ và **SLSA Build Level 3**

04



04 — Kế thừa và khuếch đại

một base image độc lan xuống mọi image build từ nó

03

Misconfiguration vs Zero-day



Đâu là rủi ro lớn hơn thực sự



VS



Kẻ thù thật nằm ở đâu

47%

lo ngại nhất về misconfiguration

13%

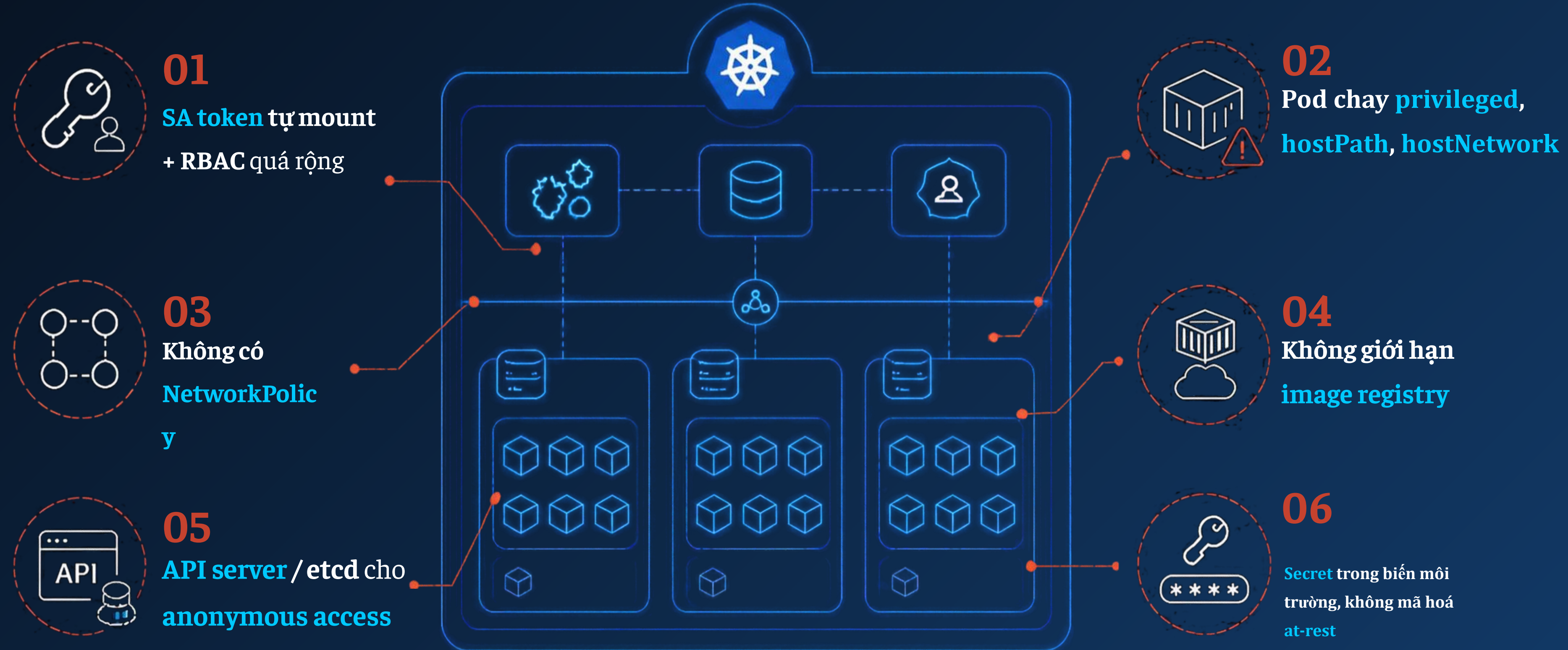
lo ngại nhất về tấn công

Tiêu chí	Misconfig	Zero-day
XÁC SUẤT GẶP	RẤT CAO	THẤP, NGẮN HẠN
CHI PHÍ KHAI THÁC	≈ 0	RẤT CAO
BẠN KIỂM SOÁT ĐƯỢC?	CÓ, 100%	PHẦN LỚN PHỤ THUỘC VENDOR

RỦI RO KỲ VỌNG = XÁC SUẤT × TÁC ĐỘNG

Ở về xác suất, misconfiguration thắng zero-day áp đảo — và đây là về bạn kiểm soát được 100%.

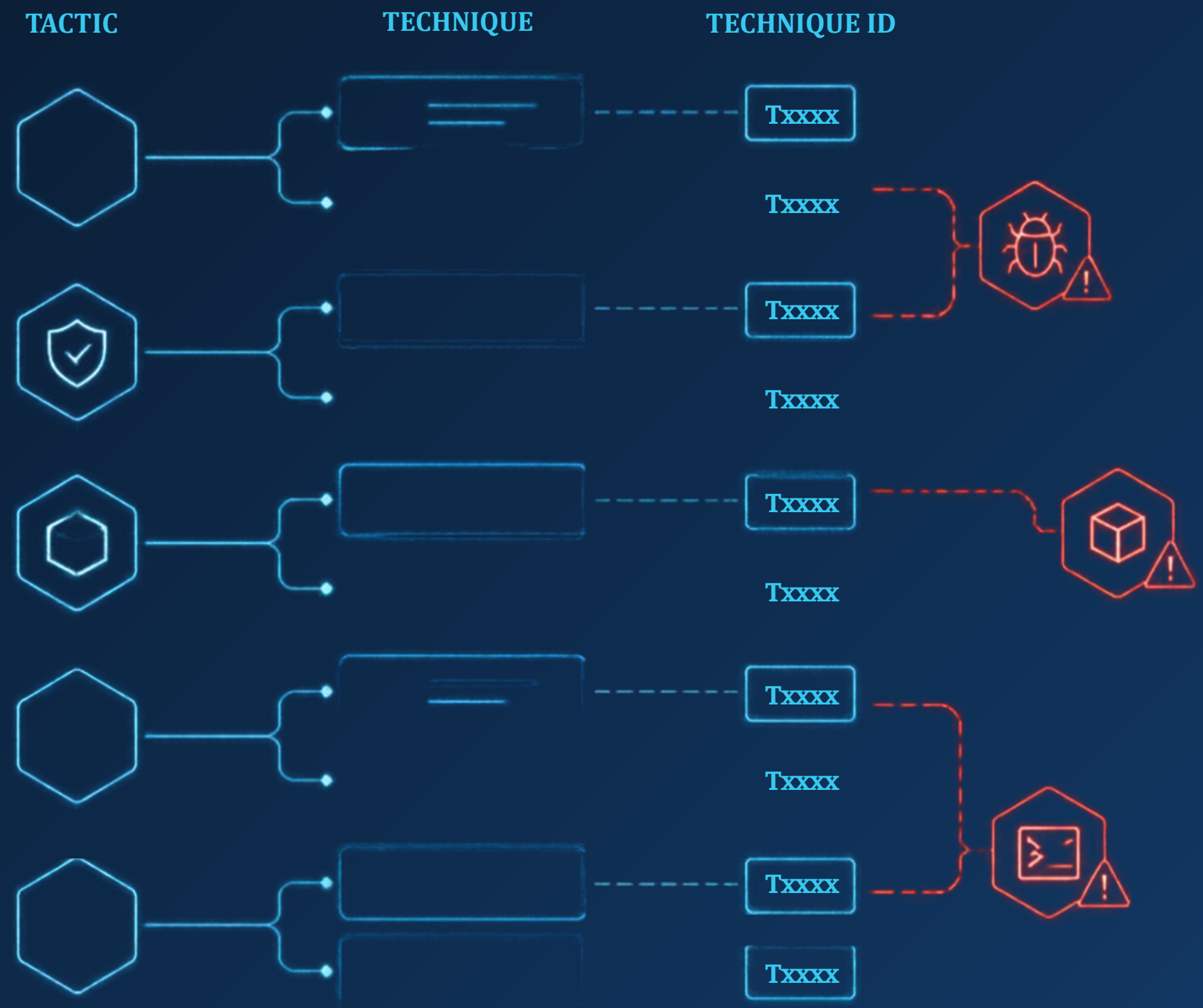
Top misconfiguration phổ biến trong Kubernetes



04

MITRE ATT&CK FOR CONTAINERS

Ngôn ngữ chung để nói về tấn công một cách có hệ thống



Hai khung tham chiếu bổ sung cho nhau

MITRE ATT&CK FOR CONTAINERS		+	THREAT MATRIX FOR K8S
PHẠM VI	Phạm vi: Container nói chung		Phạm vi: Chuyên sâu Kubernetes
TÍNH CHÍNH THỨC	Tính chính thức: Chuẩn công nghiệp, có ID (Txxxx)		Tính chính thức: Knowledge base cộng đồng, GitHub
DÙNG KHI	Dùng khi: Chuẩn hoá detection, trao đổi liên đội		Dùng khi: Đào sâu attack path cụ thể trong K8s

TACTIC = attacker muốn đạt gì · **TECHNIQUE** = họ làm cách nào để đạt

Map hai kịch bản vào MITRE ATT&CK

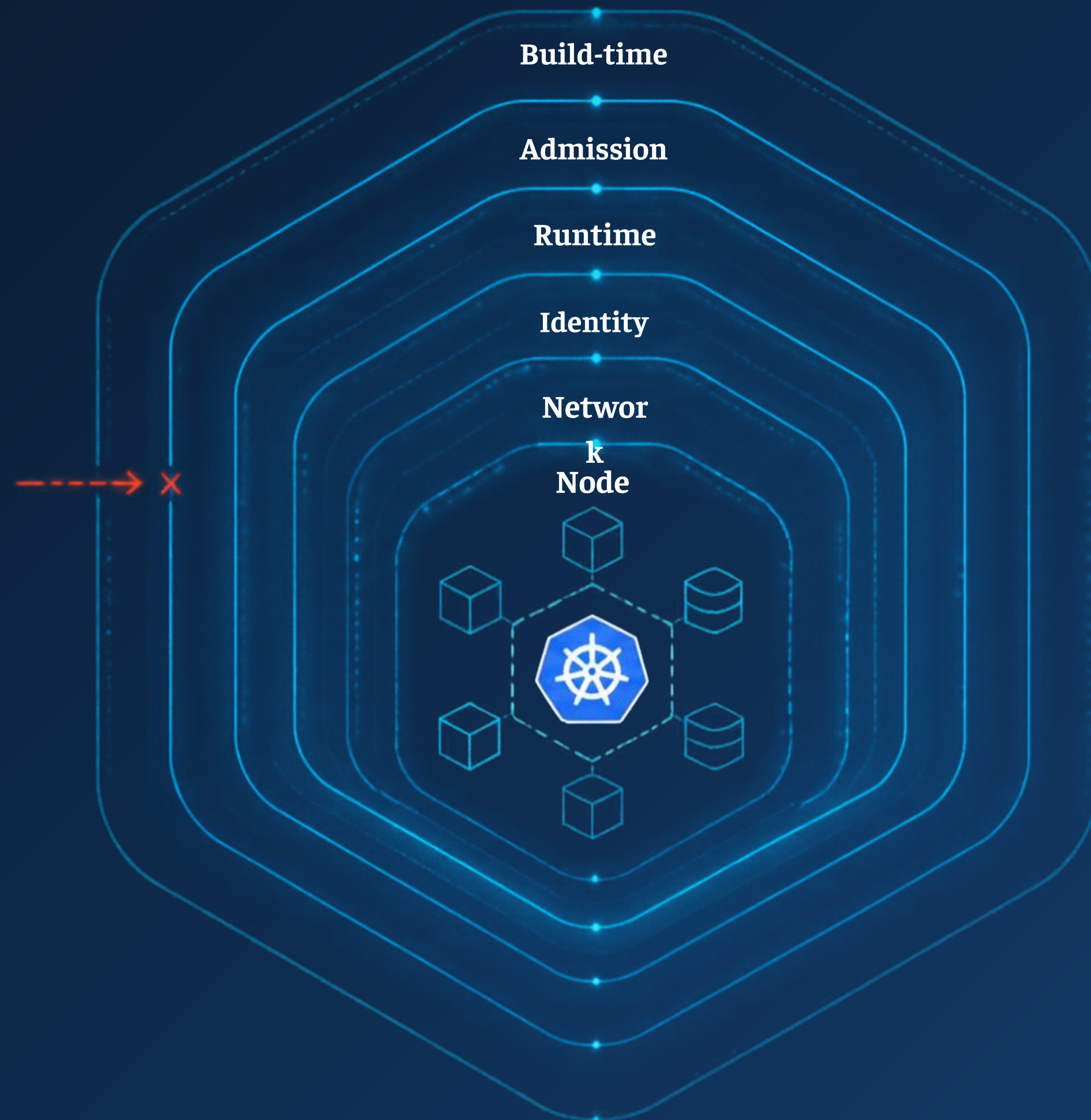
Mỗi bước tấn công đều có **technique ID** cụ thể — và một control tương ứng

	ATTACK BEHAVIOR	MITRE TECHNIQUE	ID
KỊCH BẢN 1	RCE qua ứng dụng public-facing	Exploit Public-Facing Application	T1190
	Đọc SA token đã mount	Steal Application Access Token	
			T1528
	Sửa ClusterRoleBinding để leo quyền	Account Manipulation	
			T1098.006
KỊCH BẢN 2	Đẩy image/package độc vào nguồn tin cậy	Supply Chain Compromise	
			T1195
	Cấy image độc trong registry	Implant Internal Image	
			T1525
	Deploy image độc / container độc	Deploy Container	
			T1610
	Cắm backdoor, cryptomining	Resource Hijacking	
			T1496

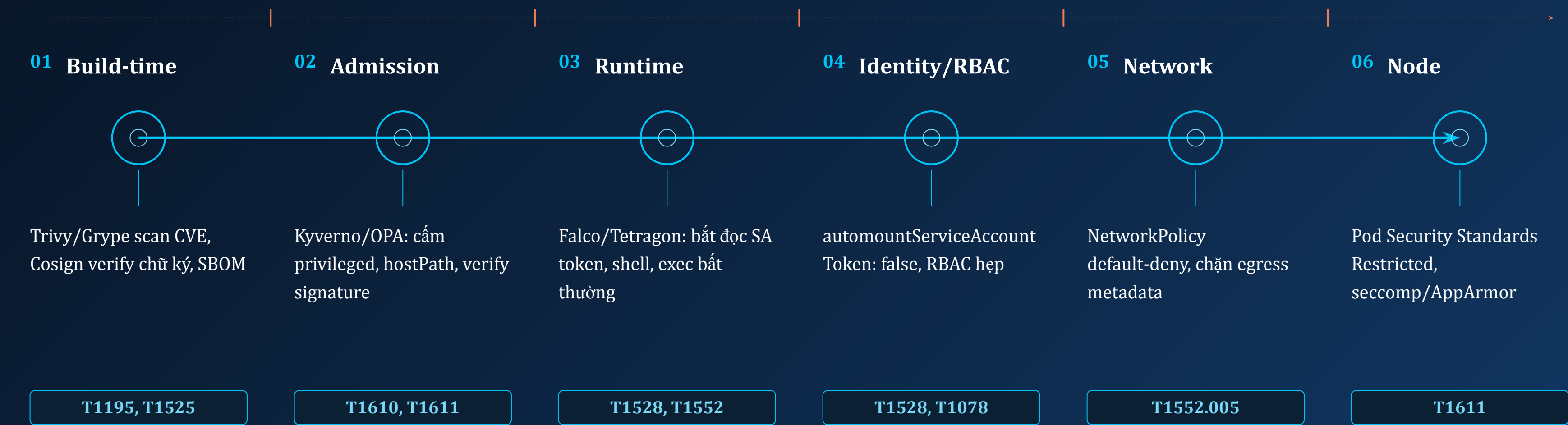
05

Chiến thuật phòng thủ theo lớp

Build-time → Admission → Runtime → Identity → Network →
Node



Defense-in-depth: mỗi lớp khoá một lớp tấn công



Kyverno — chặn ở cửa admission

Admission controller native của K8s, viết policy bằng YAML thuần



- 01

VALIDATE
validate — chặn/audit resource vi phạm
(câm privileged, hostPath)
- 02

MUTATE
mute — ép automountServiceAccountToken: false
- 03

GENERATE
generate — tự tạo default-deny NetworkPolicy
cho namespace mới
- 04

VERIFYIMAGES
verifyimages — chỉ cho chạy image đã ký
Cosign/Sigstore

ADMISSION POLICY

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: disallow-privileged
spec:
  validationFailureAction: Enforce
  rules:
    - name: no-privileged-containers
      validate:
        pattern:
          spec:
            containers:
              - securityContext:
                  privileges: "false"
```

Falco — phát hiện ở runtime

DaemonSet dùng eBPF quan sát syscall — con mắt duy nhất thấy attacker sau khi đã vào



-  eBPF — quan sát syscall
-  CPU • overhead
~1–3%
-  <5 alert nhiều/ngày/cluster •
mục tiêu vận hành

- Driver eBPF gắn kernel, theo dõi mọi syscall
 - chỉ tốn ~1–3% CPU
- Mỗi rule: condition (khi nào bắt), output, priority (8 mức)
- Falcosidekick định tuyến alert → SIEM/ SOAR, Slack, PagerDuty
- Mục tiêu vận hành: dưới 5 alert nhiều/ngày/cluster

RUNTIME RULE

```
- rule: Đọc SA token bởi process lạ
desc: Phát hiện độc token (T1528)
condition: >
  open read and container and
  fd.name startswith
  /var/run/secrets/.../token
output: >
  SA token bị đọc bởi process lạ
  (proc=proc.name pod=tk8s.pod.name)
priority: CRITICAL
tags: [k8s, T1528]
```

Hệ sinh thái công cụ — mỗi tool một vị trí



Lộ trình triển khai 30-60-90 ngày

Audit trước, enforce sau — không bao giờ bật Enforce thẳng trên production



AUDIT FIRST → ENFORCE LATER



Không bao giờ bật **Enforce** thẳng trên production



0-30 DAYS

Nhìn thấy trước đã

- Bật K8s audit log
- Falco ruleset mặc định → SIEM
- Trivy scan trong CI
- Kyverno chế độ Audit



30-60 DAYS

Chặn phần rẻ và chắc

- Chuyển policy nền sang Enforce
- Áp PSS Restricted
- Custom Falco rule cho SA token
- Playbook SOAR đầu tiên



60-90 DAYS

Least privilege + supply chain

- Thu hẹp RBAC, bỏ * trên *
- automount = false mặc định
- Bật Cosign verifyImages
- NetworkPolicy default-deny

06

Incident Response Workflow

Cô lập trước, điều tra sau — vì pod là phù du



Vòng đời IR áp vào đặc thù Kubernetes



NGUYÊN TẮC #1 — **CÔ LẬP**, ĐỪNG GIẾT NGAY
XOÁ POD = MẤT BẰNG CHỨNG + ATTACKER CHỈ CẦN POD MỚI

Runbook cho hai kịch bản tấn công

KỊCH BẢN 1 — TOKEN THEFT

TRIGGER → Falco CRITICAL · “SA token bị đọc”

- 01 **ISOLATE**
Áp NetworkPolicy deny-all lên pod (cắt egress)
- 02 **REVOKE**
Thu hồi token, rotate ServiceAccount
- 03 **INVESTIGATE**
Truy audit log: token đã gọi API gì?
- 04 **PATCH**
Vá lỗ hỏng ứng dụng gốc
- 05 **RECOVER**
Redeploy sạch, rotate mọi secret pod chạm tới

KỊCH BẢN 2 — SUPPLY CHAIN

TRIGGER → Trivy/Falco phát hiện image/package độc

- 01 **BLOCK**
Kyverno chặn digest image độc ở admission
- 02 **ISOLATE**
Cô lập pod đang chạy image nhiễm
- 03 **ROTATE**
Rotate ngay AWS/Vault/OIDC/registry token
- 04 **TRACE**
Truy SBOM: dịch vụ nào khác dùng dependency này?
- 05 **REBUILD**
Xoá image độc khỏi registry, rebuild sạch đã ký

Cảm ơn!

◆ Câu hỏi & thảo luận

