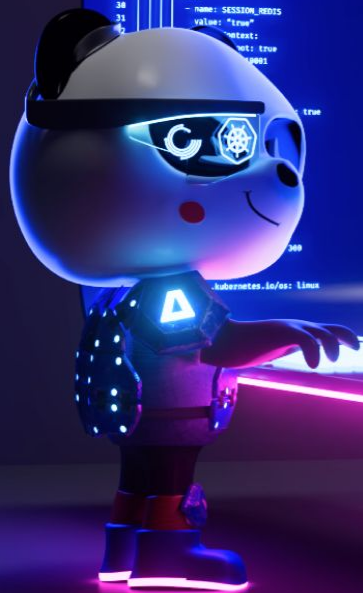


Securing Mission Critical Workloads: NOT Mission Impossible!



ARMO

Creators of  Kubescape

**SCHUBERG
PHILIS**



/whoami

Jonathan Kaftzan

VP Marketing and Biz-Dev at ARMO

Developer Advocate

Crossfit addicted



Jonathan Kaftzan

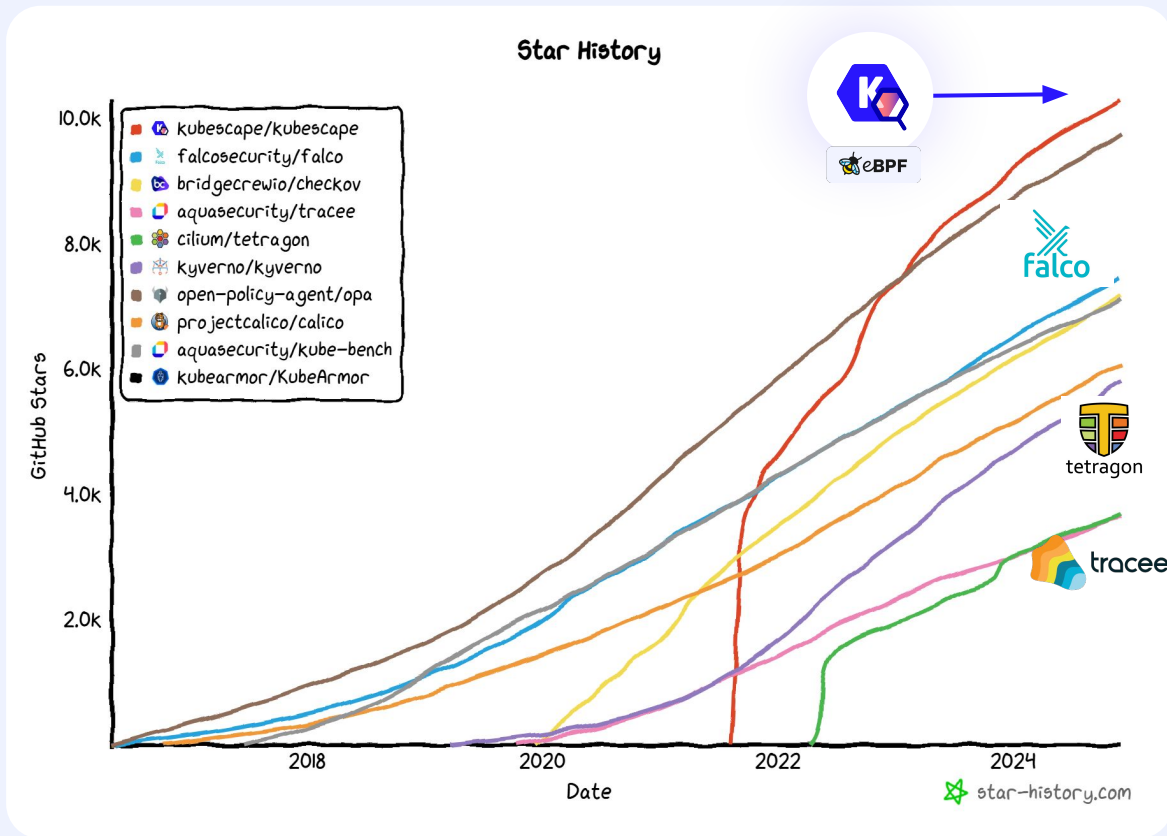


Jonathan Kaftzan



@JKaftzan

/Kubescape: The fastest growing CNCF Cloud Security project



/Kubescape: The fastest growing CNCF Cloud Security project



[About](#)

[Projects](#)

[Training](#)

[Community](#)

[Blog & News](#)

[Join](#)



BLOG / STAFF POST

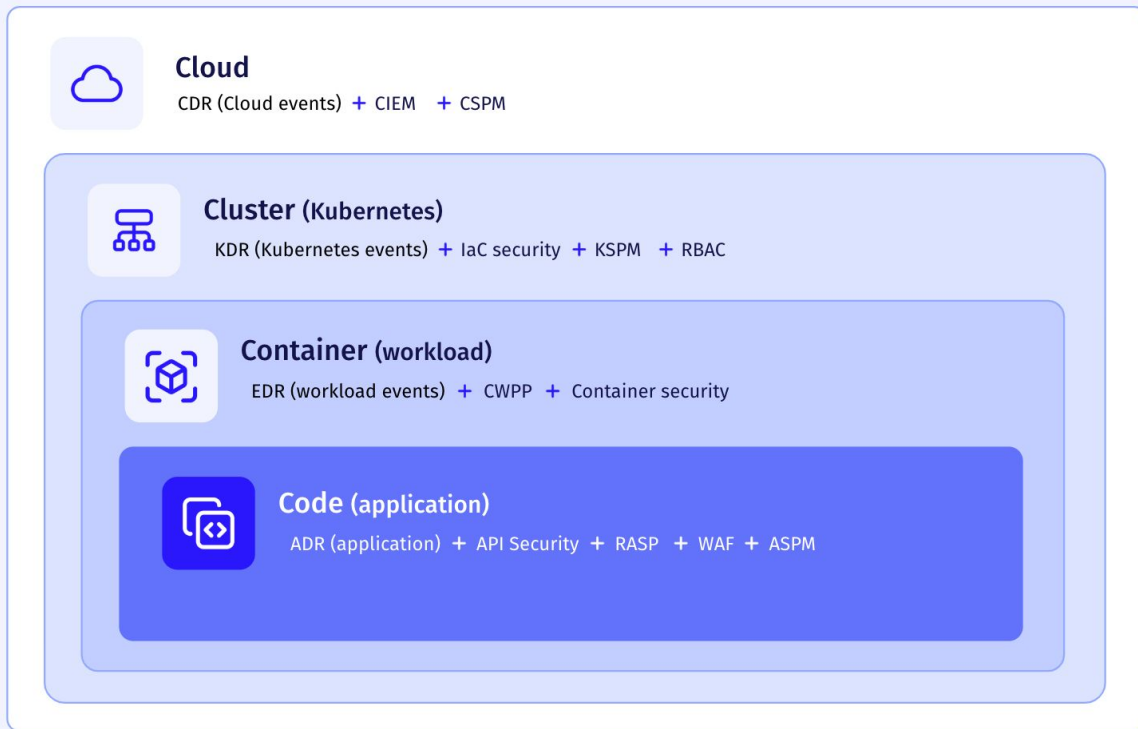
Kubescape becomes a CNCF incubating project



Posted on February 26, 2025

The CNCF Technical Oversight Committee (TOC) has voted to accept **Kubescape** as a CNCF incubating project.

/Why Is Container Security So Hard?

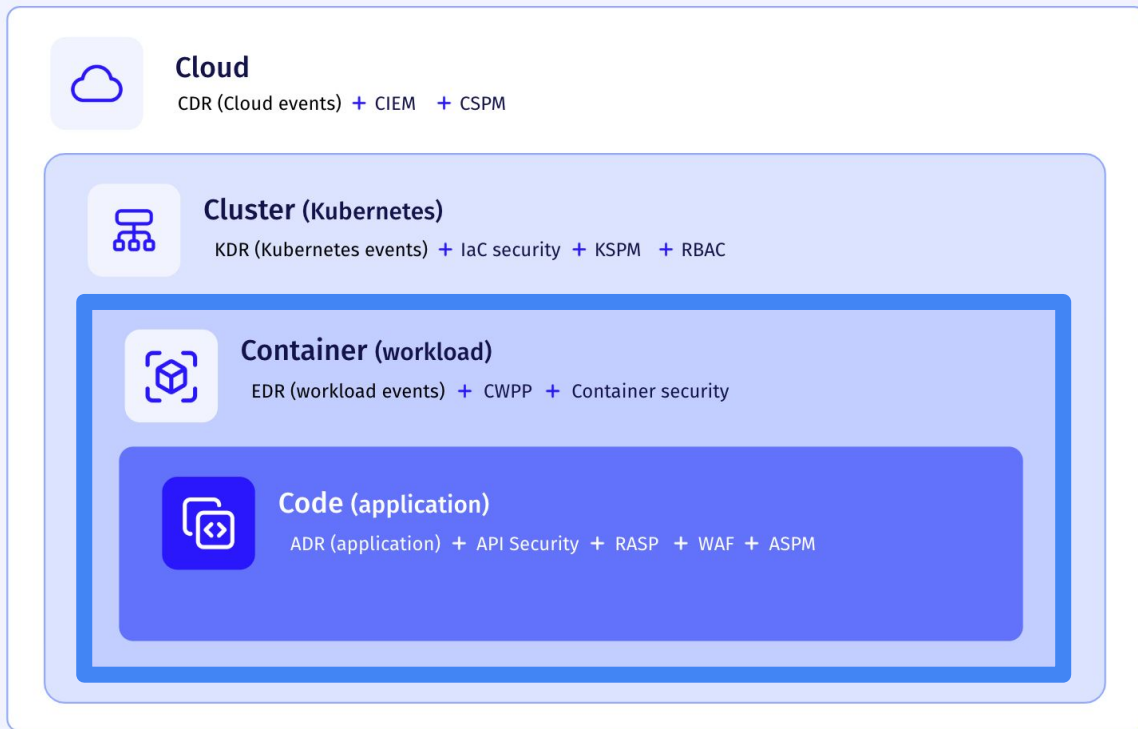


Security is a moving target

Challenges at multiple levels

- Securing your cloud environment
- Securing your container environment
- Running secure containers
- Runtime secure applications

/Why Is Container Security So Hard?



Running secure containers

Common struggles:

- **Building secure images:**
Prevent vulnerabilities before shipping
- **Deploying securely:**
Restrict permissions, control network access
- **Runtime security:**
Detecting threats in real-time

/whoami

Stephen Hoekstra

Mission Critical Engineer at Schuberg Philis

Tech Lead

Security Enthusiast



Stephen Hoekstra



Stephen Hoekstra



github.com/shoekstra

/whoami

Stephen Hoekstra

Mission Critical Engineer at Schuberg Philis

Tech Lead

Security ~~Enthusiast~~ **Sadomasochist**



Stephen Hoekstra



Stephen Hoekstra



github.com/shoekstra

/Schuberg Philis



/Securing mission critical workloads, not:

MISSION: IMPOSSIBLE
MISSION: IMPOSSIBLE

/The Reality of Vulnerability Management

Where and when to scan?



/The Reality of Vulnerability Management

Where and when to scan?



/The Reality of Vulnerability Management



/The Reality of Vulnerability Management



/The Reality of Vulnerability Management

On average ARMO customers have **~13k vulnerabilities**

Over **1k are critical and 2k are high severity**

Actually exploitable? **less than 0.1%**

/Beyond Vulnerability Management: Hardening Deployments

What is hardening?

“In computer security, hardening is the process of securing a system by reducing its attack surface...”

– Wikipedia

/Beyond Vulnerability Management: Hardening Deployments

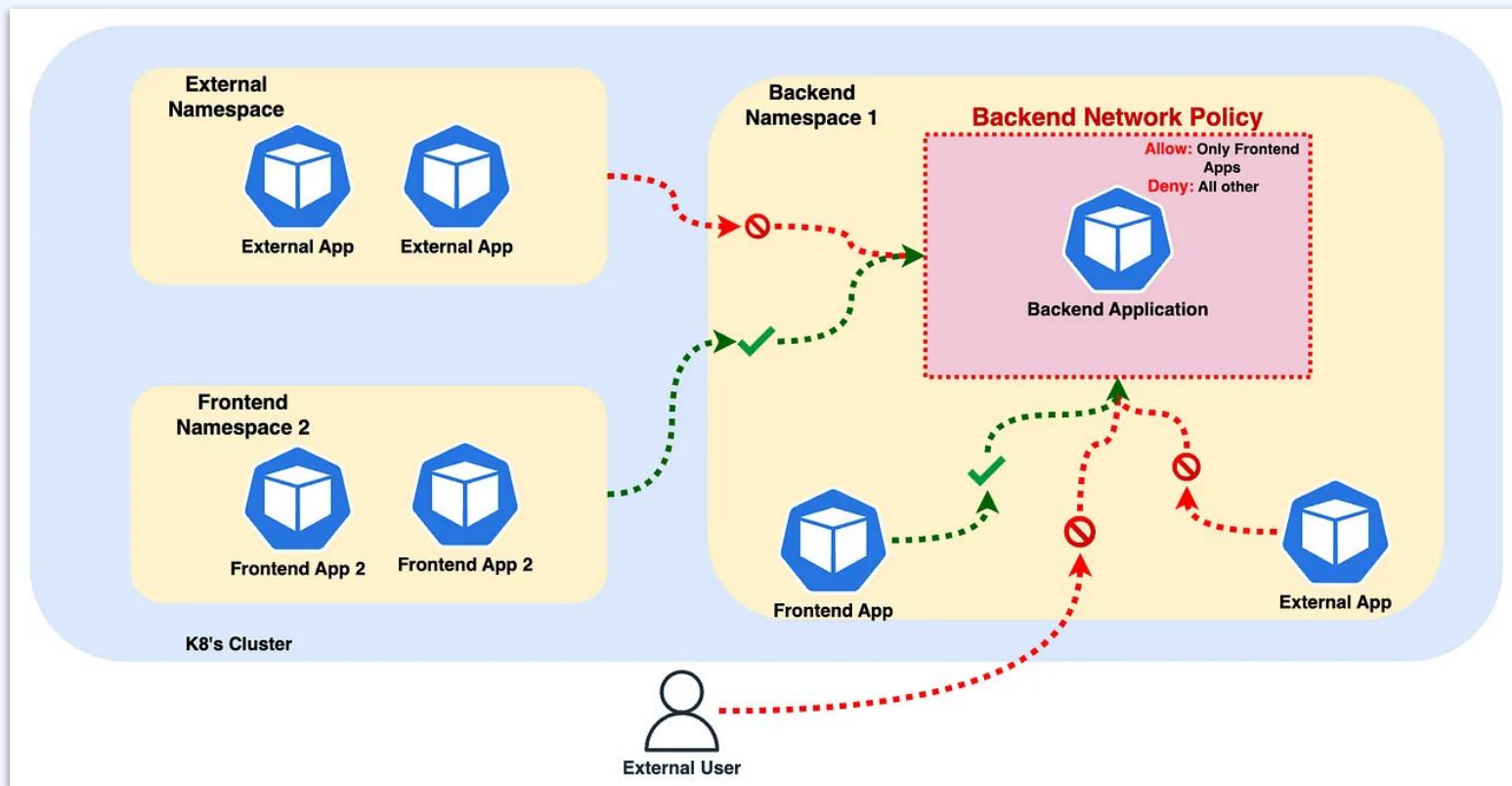
Examples of (Kubernetes) hardening

Specifying a security context	Run as non-root Read-only file systems	Can break existing containers Containers stop when they try to write to disk
Limiting network access	Create Network Policies	Can break applications

/Beyond Vulnerability Management: Hardening Deployments



/Reducing Attack Surface: Network Security



/Finding the right solution

This stuff is hard!

Are there tools out there to help with this?



But open-source projects will only get you so far...

/Finding the right solution

Things to Consider:

- **End-to-End Coverage:** Build, deploy, runtime, and response
- **Signal vs. Noise:** Prioritise real risks, avoid alert fatigue
- **Integration:** Works with your CI/CD, CNI (Cilium), observability stack
- **Enforcement & Hardening:** Prevent privileged containers, restrict syscalls (Tetragon), enforce policies
- **Threat Detection:** Uses modern techniques like eBPF profiling
- **Compliance & Auditing:** Supports industry standards (PCI-DSS, CIS, NIST)

Look for a solution that balances security, usability, and automation

/The ARMO Cloud Runtime Security Offering



Cloud Security

{Kubernetes-first} Cloud Posture

- + CSPM (Agentless scanning)
- + KSPM
- + Runtime-reachability Vulnerability management
- + IaC security
- + Identity security (CIEM) & RBAC



Kubescape



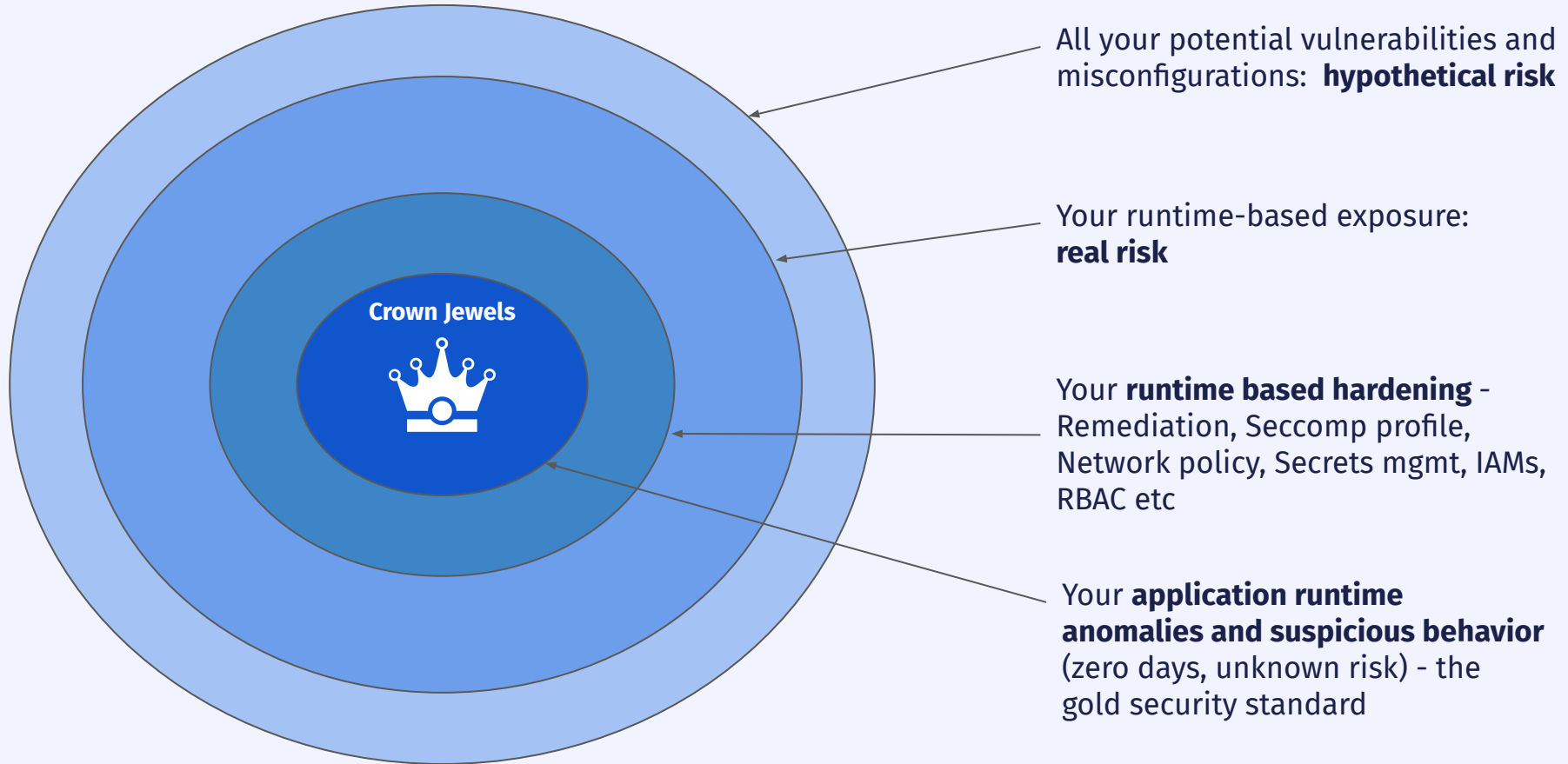
Real Time Protection

Threat Detection & Response (CADR)

- + Cloud Application Detection & Response (CDR, ADR)
- + CWPP
- + API security
- + Container security

Multi-Cloud, On-Premises and Air-Gapped

ARMO Cloud Runtime Security Approach

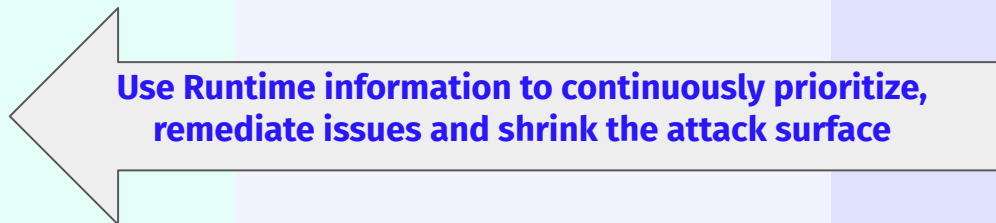


/Posture <-> Runtime Reinforcing Cycle

ΔARMO



**Configuration and
Context
(CSPM / KSPM)**



**Use Runtime information to continuously prioritize,
remediate issues and shrink the attack surface**



**Use Posture and Deep risk context
to adapt runtime security policies and reduce alert fatigue**



**Runtime Information
(eBPF, CDR/KDR/ADR)**

/90% reduction in number of vulnerabilities to manage using runtime context

The dashboard displays a sidebar on the left with navigation options: Posture, Security Risks, Attack Path, Vulnerabilities, Compliance, Network Policies, Seccomp Profiles, RBAC Insights, Threat Detection, Runtime Incidents, Code, Repository Scanning, Registry Scanning, Policies, Risk Acceptance, and Threat Detection.

The main content area features a top navigation bar with tabs for CVEs, Workloads, Images, and SBOM. Below this is a filter bar with buttons for Cluster, Namespace, Workload, CVE ID, Severity, CVSS ≥, In Use: Yes (highlighted with a blue box), Exploitable, and Fixable. To the right of the filter bar is a Risk factor dropdown menu with an 'Add filter' button and a 'Clear all filters' link. A 'Risk spotlight' toggle is also present.

A notification states: "Filters has cut vulnerabilities by 77.9%, now showing only 1,922 out of the initial 8,712".

Two charts are shown: "Vulnerabilities by severity" (a donut chart showing 203 Critical, 782 High, 810 Medium, and 127 Low) and "Vulnerabilities over time" (a stacked bar chart showing the number of vulnerabilities from Oct 20 to Nov 01).

A table titled "Showing 50 of 1922 CVEs" is displayed below the charts. The table has columns for CVE ID, SEVERITY, CVSS, EPSS, EXPLOITABLE, COMPONENT, and FIX VERSION. The first row shows CVE-2024-38513 with a severity of Critical, CVSS of 10.0 v3.1, EPSS of 0.04%, and is not exploitable. The component is github.com/gofiber/fiber/v2.v2.49.2 and the fix version is 2.52.5. The second row shows CVE-2024-41110 with a severity of Critical, CVSS of 9.8 v3.1, EPSS of 0.04%, and is exploitable. The component is github.com/gofiber/fiber/v2.v2.49.2 and the fix version is 25.0.6. The third row shows CVE-2024-41110 with a severity of Critical, CVSS of 9.8 v3.1, EPSS of 0.04%, and is exploitable. The component is github.com/gofiber/fiber/v2.v2.49.2 and the fix version is 25.0.6.

Two callout boxes provide additional context:

- Identifying the in-use SW artifacts and libraries, to eliminate CVEs that are not reachable in the runtime environment** (points to the "In Use: Yes" filter).
- Use risk factor to focus on CVEs in high-risk workloads which are publicly available, privileged or other risk factors** (points to the "Risk factor" dropdown menu).

/Harden critical workloads and prevent attacks with auto-generated network policies and seccomp profiles

The screenshot displays the ARMO console interface. On the left, a sidebar menu includes 'Network Policies' and 'Seccomp Profiles', both highlighted with red circles. The main content area shows a list of workloads, with 'my-release-argo-cd-server' selected. A modal window displays the configuration for this workload, including a 'Seccomp CRD' section with a code editor showing a SeccompProfile configuration. A second modal window shows the generated NetworkPolicy configuration. Blue arrows indicate the flow of information from the workload selection to the Seccomp CRD and then to the NetworkPolicy.

Auto-generated Network policy for every application , to easily deploy into the cluster for security and compliance

Seccomp profile, limit the system calls and capabilities of containers to restrict potential attacker - a highly underutilized linux capability due to complexity

```
1 kind: SeccompProfile
2 apiVersion: sdx.softwarecomposition.kubescape.io/v1beta1
3 metadata:
4   name: my-release-argo-cd-server
5   namespace: default
6   creationTimestamp: null
7 spec:
8   containers:
9     - name: argo-cd-server
10       path: default/Deployment-my-release-argo-cd-server-argocd-server.json
11 spec:
12   defaultAction: SCMP_ACT_ERRNO
13   architectures:
14     - SCMP_ARCH_X86_64
15     - SCMP_ARCH_X86
16     - SCMP_ARCH_X32
17   syscalls:
18     - names:
19       - accept4
20       - access
21       - arch_prctl
```

```
1 + kind: NetworkPolicy
2 + apiVersion: networking.k8s.io/v1
3 + metadata:
4 + name: deployment-prometheus-kube-state-metrics
5 + namespace: default
6 + labels:
7 +   kubescape.io/workload-api-group: apps
8 +   kubescape.io/workload-api-version: v1
9 +   kubescape.io/workload-kind: Deployment
10 +   kubescape.io/workload-name: prometheus-kube-state-metrics
11 +   kubescape.io/workload-namespace: default
12 + annotations:
13 +   generated-by: kubescape
14 + spec:
15 +   podSelector:
16 +     matchLabels:
17 +       app.kubernetes.io/instance: prometheus
18 +       app.kubernetes.io/name: kube-state-metrics
19 +   ingress:
20 +     - ports:
21 +       - protocol: TCP
22 +         port: 8080
23 +   from:
```



Thank you_

ARMO