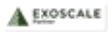
FULLSTACKS

A tale of security, policies, code and reality



Senior DevOps Engineer
@ FullStackS

Organizer / Host
CNCG Graz + CND Austria

I ❤️ 🤷‍♀️ 🐱 🏋️‍♂️ 🍻 🎸 🤘

Daniel Drack
/in/drackthor
daniel@drackthor.me

BSc MA MBA
Kubestronaut, TFA, VA, GitLab, Scrum, Snyk, Exoscale
CNCF Ambassador
Co-Founder & Secretary of Cloud Native Austria

What are we talking about..

Security shouldn't slow you down, it should scale with you.
I'll share how we build "Security as Code" using tools and policy engines like OPA, cnspec, and k8s admission controllers.
We explain and demo how to discuss, develop, implement and practice key cloud-native security pieces for true defense in depth.

Beginning by clarifying what policies and best practices actually are, and where to source them.
From there, we'll explore how to translate high-level security requirements into concrete technical policies, including how to implement, enforce, and maintain them effectively.
Finally, we'll briefly discuss the practical challenges of establishing these practices in real-world environments and sustaining them during day-to-day operations.

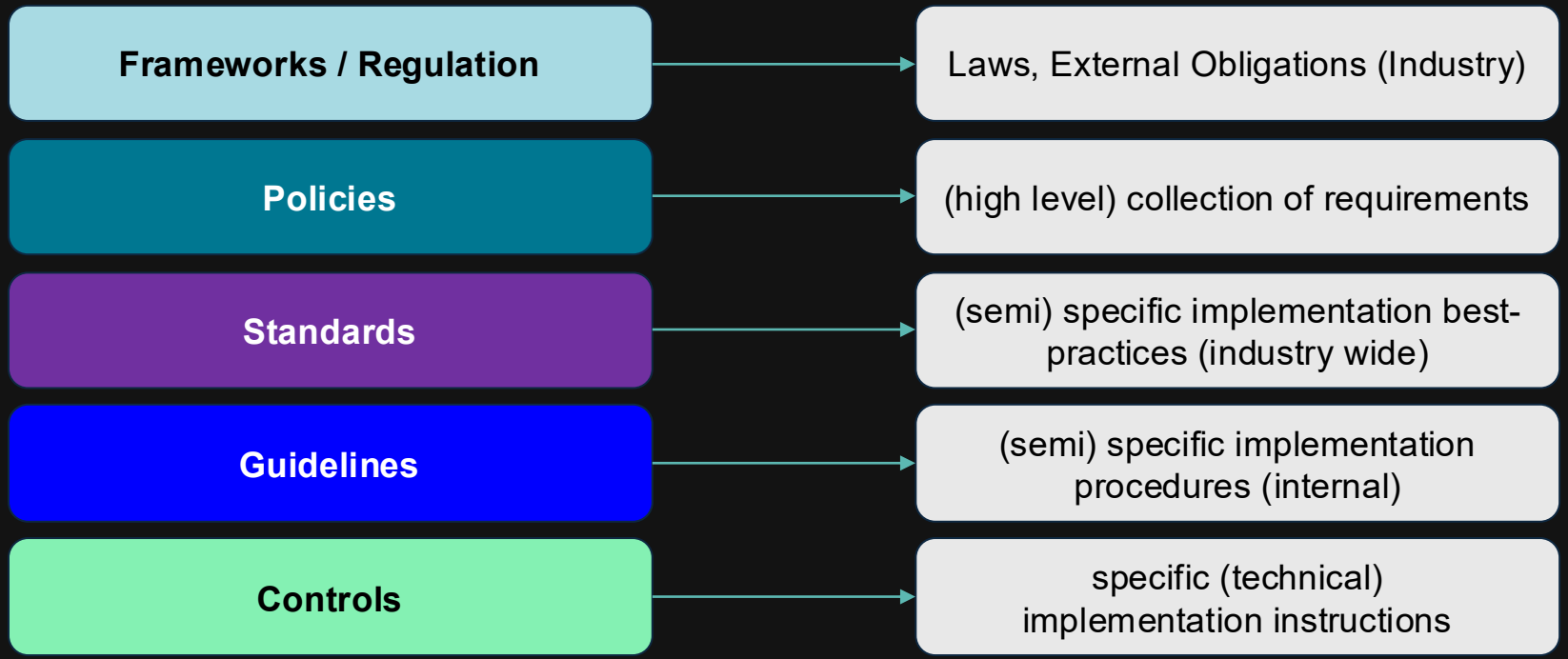
What are we talking about..

"Security as Code" [...] tools and policy engines [...] OPA, cnspec, and k8s admission controllers

[...] defense in depth
[...] discuss, develop, implement and practice

[...] what policies are, and where to source them
[...] translate high-level [...] into policies
[...] practical challenges [...] in real-world

what policies are, and where to source them



NIST CSF - Policy

"It offers a taxonomy of highlevel cybersecurity **outcomes** [..] The CSF does not prescribe how outcomes should be achieved. Rather, it links to online resources [..]"

- https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf

PROTECT (PR): Safeguards to manage the organization's cybersecurity risks are used

- **Identity Management, Authentication, and Access Control (PR.AA):** Access to physical and logical assets is limited to authorized users, services, and hardware and managed commensurate with the assessed risk of unauthorized access
  - **PR.AA-01:** Identities and credentials for authorized users, services, and hardware are managed by the organization
  - **PR.AA-02:** Identities are proofed and bound to credentials based on the context of interactions
  - **PR.AA-03:** Users, services, and hardware are authenticated
  - **PR.AA-04:** Identity assertions are protected, conveyed, and verified

BSI/ISO27k - Standards

"Im IT-Grundschutz-Kompendium werden standardisierte **Sicherheitsanforderungen** für typische Geschäftsprozesse, Anwendungen, IT-Systeme, Kommunikationsverbindungen, Gebäude und Räume in IT-Grundschutz-Bausteinen beschrieben [..]"

- https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/IT-GS-Kompendium/IT_Grundschutz_Kompendium_Edition2023.pdf

SYS.1.6.A17 Ausführung von Containern ohne Privilegien (S)

Die Container-Runtime und alle instanziierten Container SOLLTEN nur von einem nicht-privilegierten System-Account ausgeführt werden, der über keine erweiterten Rechte für den Container-Dienst und das Betriebssystem des Host-Systems verfügt oder diese Rechte erlangen kann. Die Container-Runtime SOLLTE durch zusätzliche Maßnahmen gekapselt werden, etwa durch Verwendung der Virtualisierungserweiterungen von CPUs.

Sofern Container ausnahmsweise Aufgaben des Host-Systems übernehmen sollen, SOLLTEN die Privilegien auf dem Host-System auf das erforderliche Minimum begrenzt werden. Ausnahmen SOLLTEN angemessen dokumentiert werden.

SYS.1.6.A7 Persistenz von Protokollierungsdaten der Container (B)

Die Speicherung der Protokollierungsdaten der Container MUSS außerhalb des Containers, mindestens auf dem Container-Host, erfolgen.

PCI DSS - Standards

" [..] PCI DSS provides a baseline of technical and operational **requirements** designed to protect account data. [..]"

- https://docs-prv.pcisecuritystandards.org/PCI%20DSS/Standard/PCI-DSS-v4_0_1.pdf

Requirements and Testing Procedures		Guidance
5.2 Malicious software (malware) is prevented, or detected and addressed.		
Defined Approach Requirements	**Defined Approach Testing Procedures**	**Purpose** There is a constant stream of attacks targeting newly discovered vulnerabilities in systems previously regarded as secure. Without an anti-malware solution that is updated regularly, new forms of malware can be used to attack systems, disable a network, or compromise data. **Good Practice** It is beneficial for entities to be aware of "zero-day" attacks (those that exploit a previously unknown vulnerability) and consider solutions that focus on behavioral characteristics and will alert and react to unexpected behavior. **Definitions** System components known to be affected by malware have active malware exploits available in the real world (not only theoretical exploits).
5.2.1 An anti-malware solution(s) is deployed on all system components, except for those system components identified in periodic evaluations per Requirement 5.2.3 that concludes the system components are not at risk from malware.	**5.2.1.a** Examine system components to verify that an anti-malware solution(s) is deployed on all system components, except for those determined to not be at risk from malware based on periodic evaluations per Requirement 5.2.3.	
Customized Approach Objective	**5.2.1.b** For any system components without an anti-malware solution, examine the periodic evaluations to verify the component was evaluated and the evaluation concludes that the component is not at risk from malware.	
Automated mechanisms are implemented to prevent systems from becoming an attack vector for malware.		

CIS Benchmarks - Controls

"This document provides **prescriptive guidance** for establishing a secure **configuration** posture for Kubernetes v1.29 - v1.32.[..]"

- https://workbench.cisecurity.org/benchmarks/21709

1.2.1 Ensure that the --anonymous-auth argument is set to false

Audit Procedure

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--anonymous-auth` argument is set to `false`.

Alternative Audit

```
kubectl get pod -nkube-system -lcomponent=kube-apiserver -o=jsonpath='{range .items[*]}{.spec.containers[*].
command} {"\n"}{end}' | grep '\--anonymous-auth' | grep -i false
```

If the exit code is '1', then the control isn't present / failed

Remediation Procedure

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the below parameter.

```
--anonymous-auth=false
```

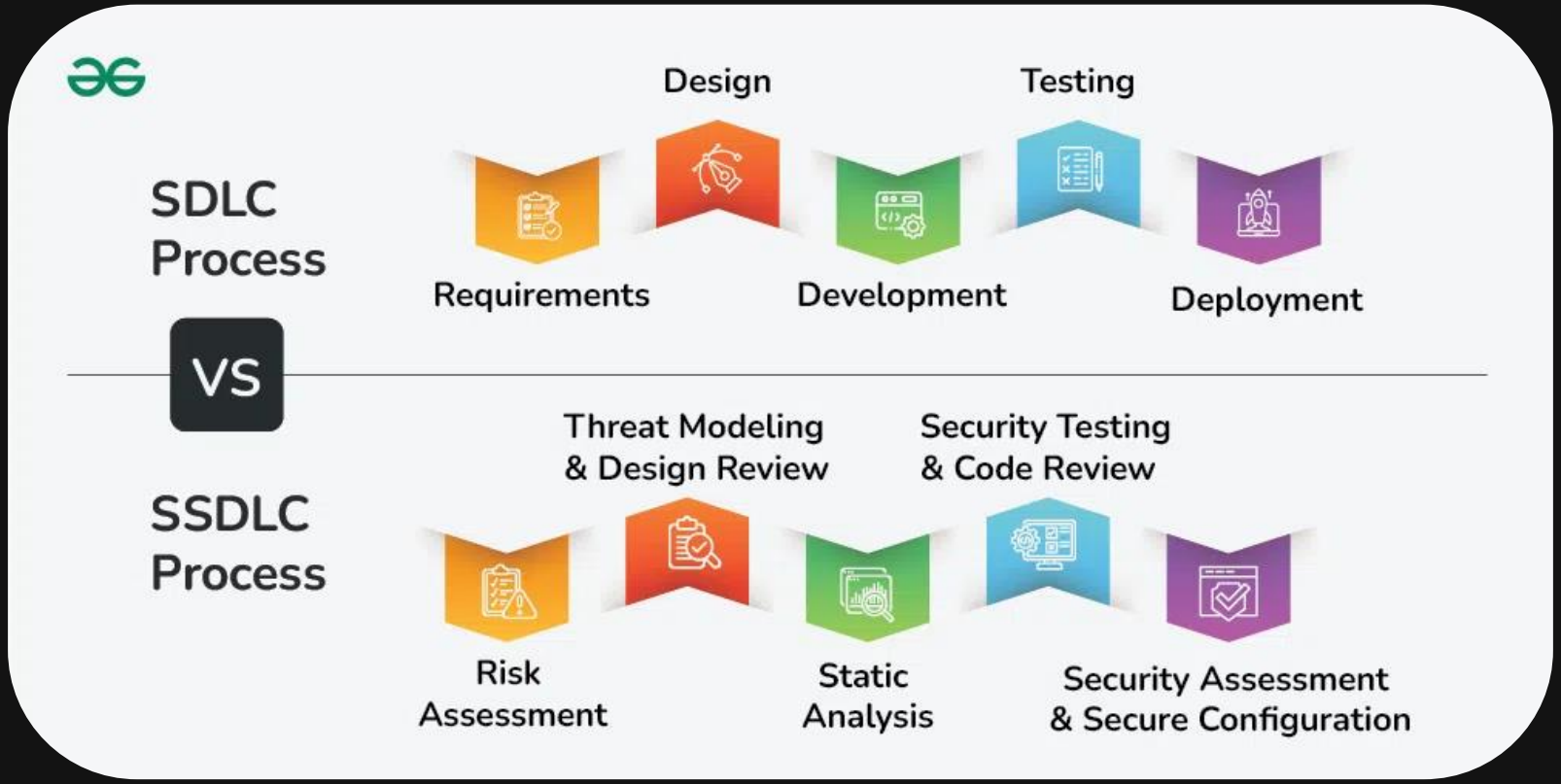
What are we talking about..

"Security as Code" [...] tools and policy engines [...] OPA, cnspec, and k8s admission controllers

[...] defense in depth.
[...] discuss, develop, implement and practice

✅ what policies are, and where to source them
[...] translate high-level [...] into policies
[...] practical challenges [...] in real-world

discuss, develop, implement and practice



Source: https://www.geeksforgeeks.org/ethical-hacking/what-is-secure-software-development-life-cycle-ssdlc/

discuss, develop, implement and practice

Code

Build

Artefact

App

Deployment Manifest

Infrastructure

VM / OS

Platform / k8s

Platform

Repo Push Policies

Vulnerability Policies

License Policies

IaC Policies

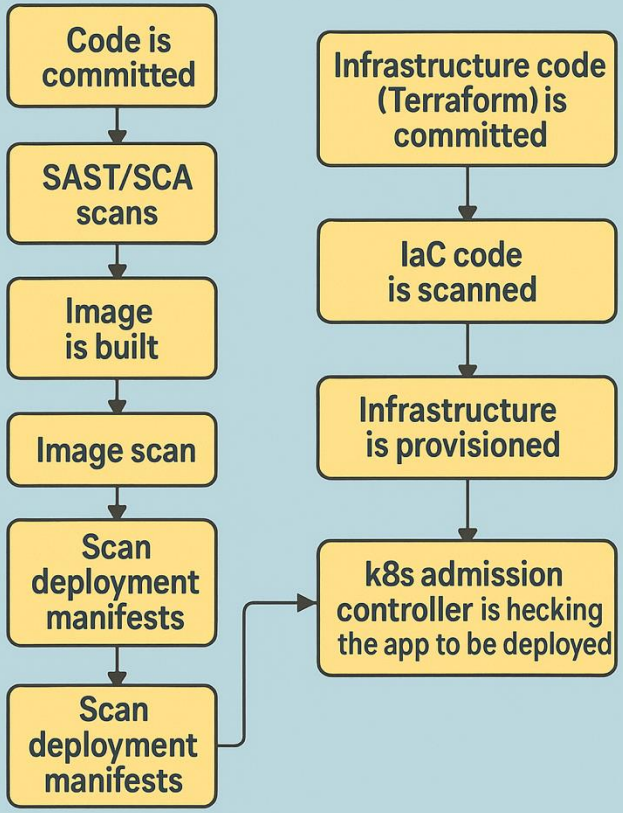
Cost Policies

Security Policies

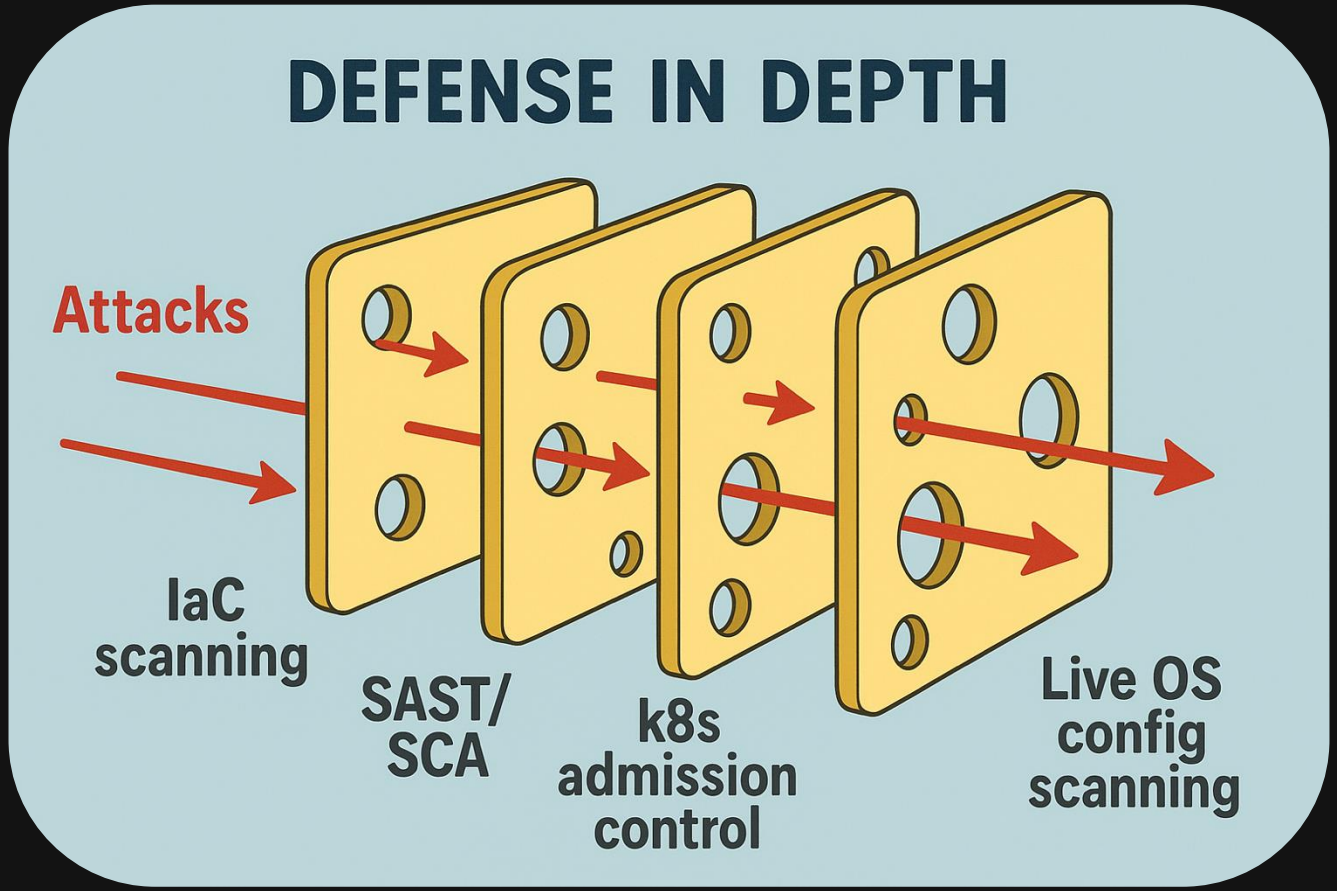
Custom Policies

APPLICATION LIFECYCLE

From aDevops perspective



[...] defense in depth



Repo Push Policies

Vulnerability Policies

License Policies

IaC Policies

Cost Policies

Security Policies

Custom Policies






checkov

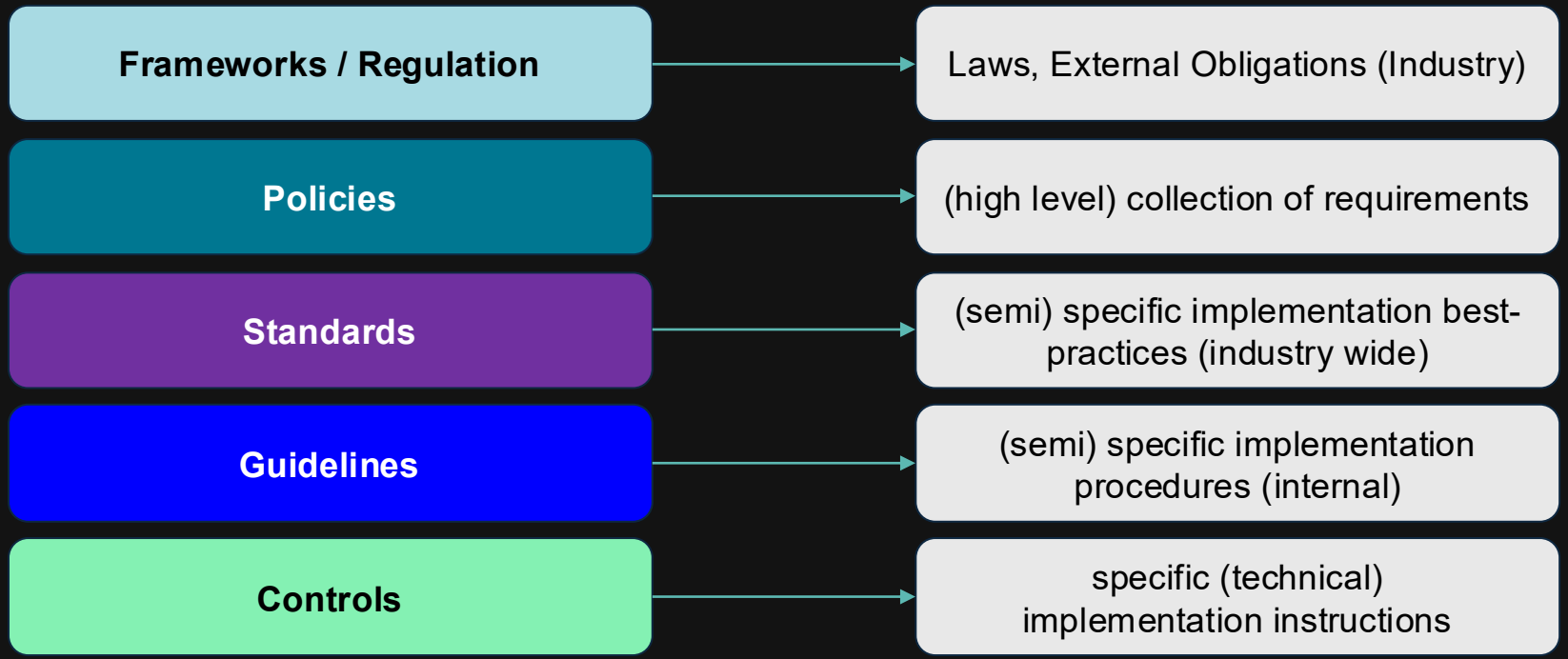

What are we talking about..

"Security as Code" [...] tools and policy engines [...] OPA, cnspec, and k8s admission controllers

✅ defense in depth.
✅ discuss, develop, implement and practice

✅ what policies are, and where to source them
[...] translate high-level [...] into policies
[...] practical challenges [...] in real-world

[...] translate high-level [...] into policies



[...] translate high-level [...] into policies

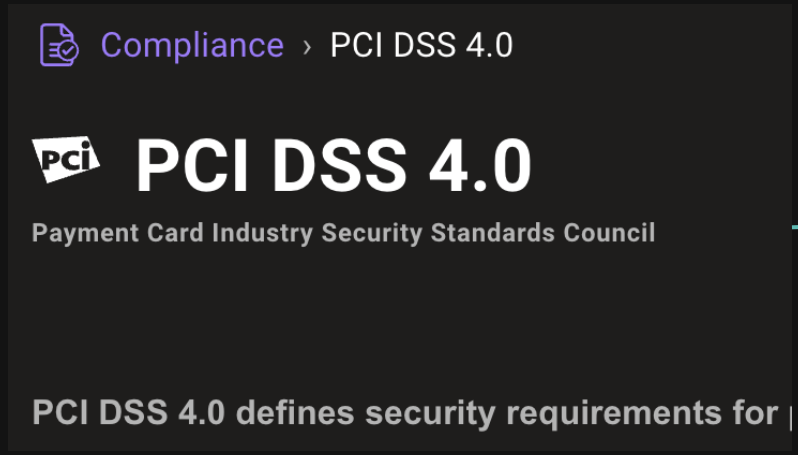
If you want to ensure {PCI/...} compliance as code, you need to map technical checks to {PCI/...} controls.

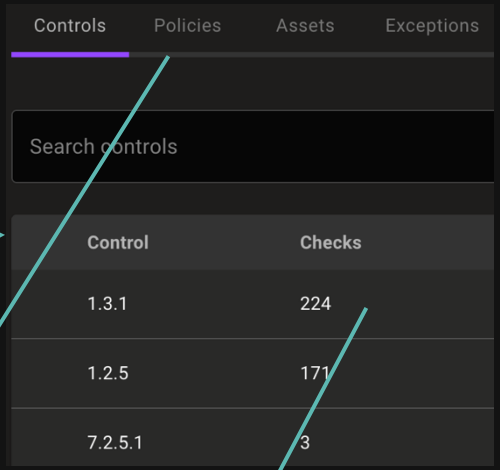
Technical checks must be formulated as code.

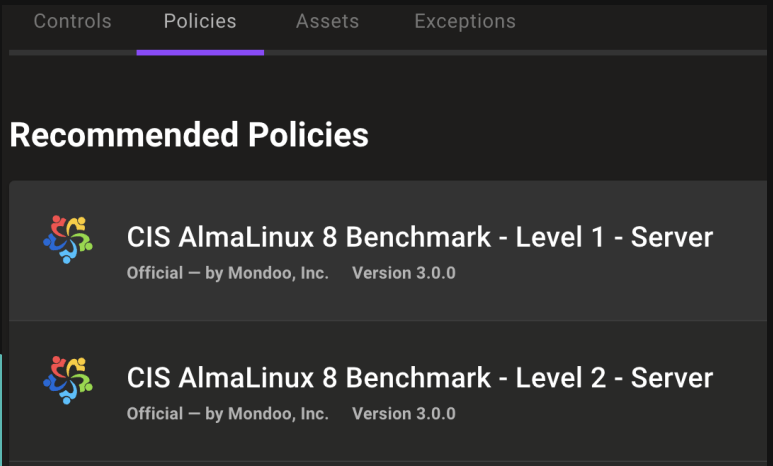
Compliance controls are NOT asset bound – they are tech agnostic.
Technical checks are asset specific.

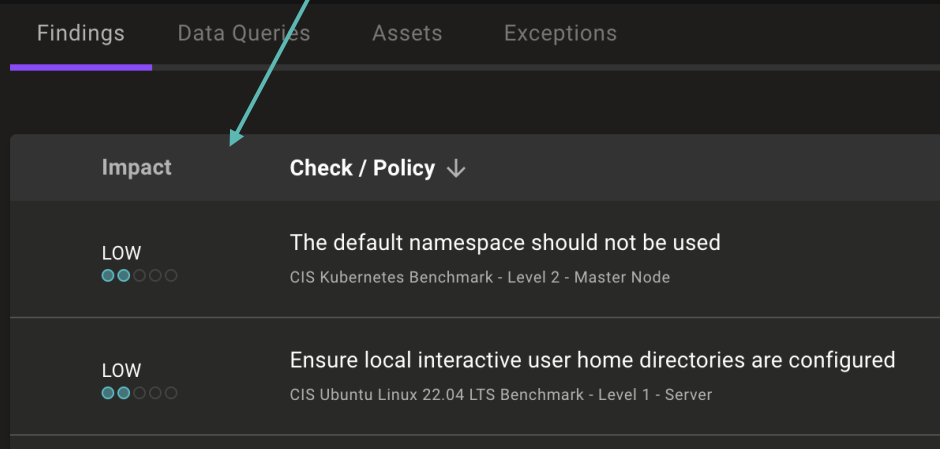
1 Compliance controls → Asset Type A tech check 2.3.1
→ Asset Type B tech check K.L.x
→ Asset Type C tech check 5

[...] translate high-level [...] into policies









[...] translate high-level [...] into policies

Key Part:

Somebody needs to map “high level” framework controls (PCI DSS,..), with low level tech checks (CIS Benchmarks).

Options:
-- github.com/ComplianceAsCode/content
-- Proprietary Vendor Mappings

[…] translate high-level […] into policies

Daniel – kurze Demo.

What are we talking about..

"Security as Code" [...] tools and policy engines [...] OPA, cnspec, and k8s admission controllers

✅ defense in depth.
✅ discuss, develop, implement and practice

✅ what policies are, and where to source them
✅ translate high-level [...] into policies
[...] practical challenges [...] in real-world

[...] practical challenges [...] in real-world

- 👉 Know Your Requirements!
- 👉 Standardize and Streamline your SDLC
- 👉 Report, then Enforce Policies
- 👉 Use Policy Definition Workshops
- 👉 Use Policy Enforcement Sprints
- 👉 Invest Time and Money

Demo Time

OPA to check Terraform code

→ naming conventions for Artifactory Repos

cnspec for OS config checks

→ Ubuntu CIS Benchmark Level 1+2

NeuVector for k8s deployment configs

→ enforce resource limits