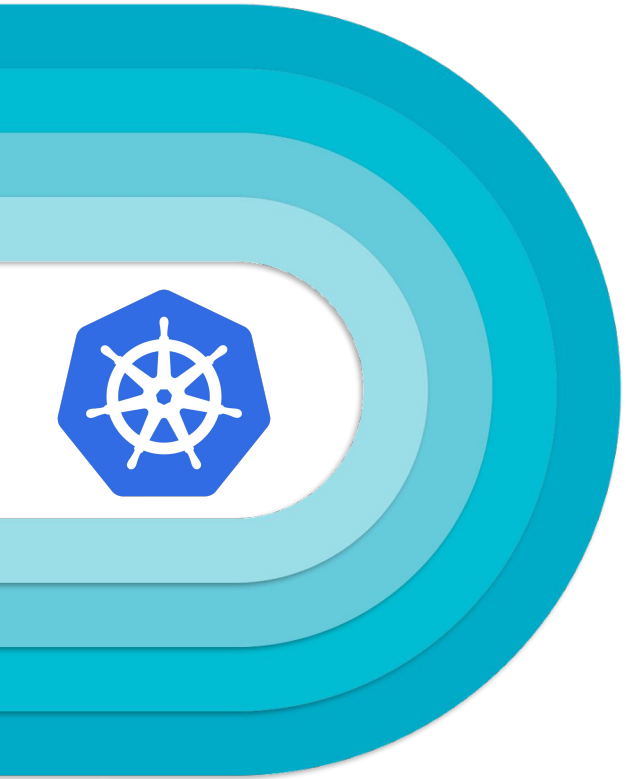
Introduction to Cloud Native Compliance and Benchmarks for DevOps

Ryan O'Daniel Solutions Architect | Sysdig Federal

ryan.odaniel@sysdig.com


Ryan O'Daniel
presents
FUN WITH
compliance
and
benchmarks
REC

Layers of Cloud-Native Security



CSPM

Continuous validation of cloud configuration, items such as S3, IAM, Password Policies, Encryption, etc…

Compliance Frameworks

Continuously validating compliance controls (NIST, PCI, SOC2, etc…) against deployed applications and infrastructure.

Benchmarking

Continuously ensuring nodes supporting applications ar hardened at the Orchestration, Runtime, and OS levels.

Image Scanning

Scanning images at both runtime and build time for vulnerabilities with real-time detection of new violations against compliance and security frameworks.

Runtime Security

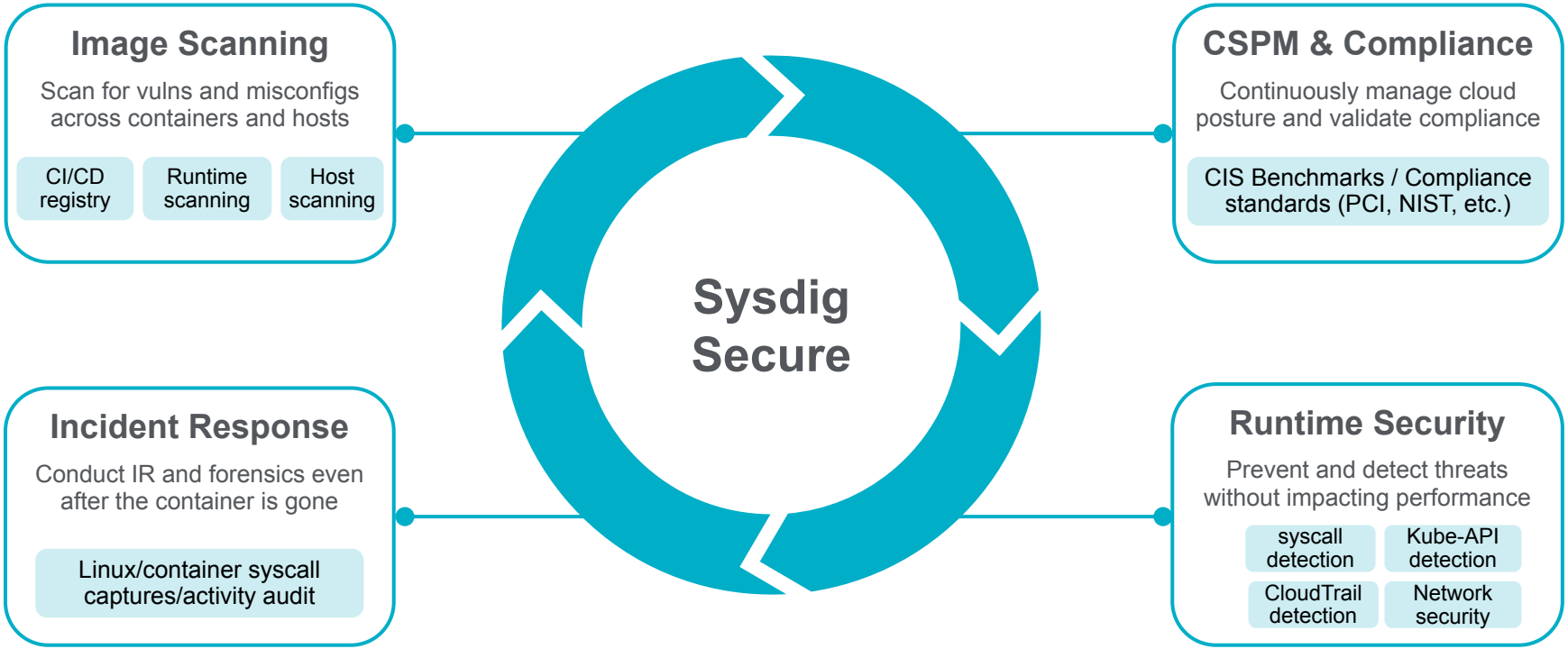
Real-time evaluation of events within cloud infrastructure, from the cloud provider down to the system calls against host kernels.

Forensics

Deep information with context on security events when they occur with correlation between containers, orchestrators, and cloud infrastructure.

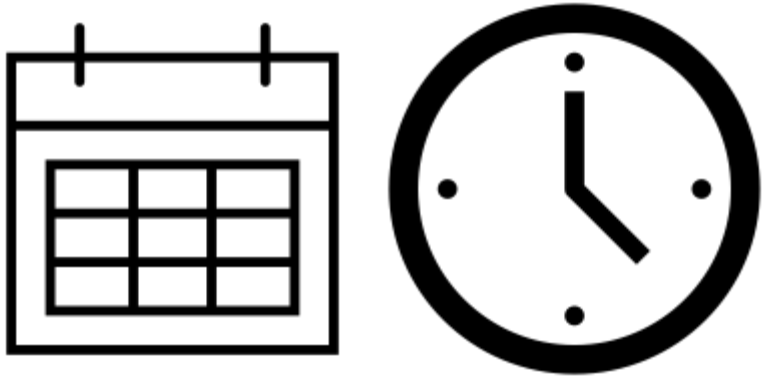
sysdig

Unified Cloud Workload and Cloud Security



sysdig

Old Days of Compliance...

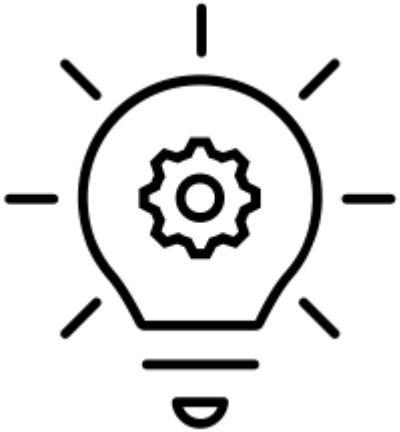


Manual work done ahead of time to prepare for compliance audit...

Then sit with an auditor to validate findings...

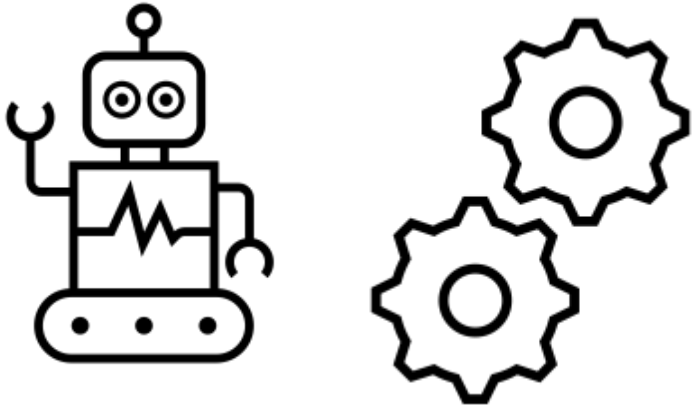
sysdig

How to do Compliance Today…



Capture and present technical controls...

Automatically



sysdig

Benchmarks vs Compliance

Benchmarks

tell you what to do to have a secure infrastructure

Compliance

ask you what you are doing to maintain your secure infrastructure

sysdig

Center for Internet Security®

Confidence in the Connected World®

www.cisecurity.org

CIS **Kubernetes** v1.20 Benchmark v1.0.0
CIS **Docker** Benchmark 1.3.1
CIS Distribution Independent **Linux** Benchmark v2.0.0
CIS **Amazon Web Services** Foundations Benchmark v1.5.0
CIS **Google Cloud** Platform Foundation Benchmark v1.2.0
CIS Microsoft **Azure** Foundations Benchmark v1.3.1

sysdig

CIS_Kubernetes_V1.20_Benchmark_v1.0.0.pdf

1.2 API Server

This section contains recommendations relating to API server configuration flags

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

Profile Applicability:

- Level 1 - Master Node

Description:

Disable anonymous requests to the API server.

Rationale:

When enabled, requests that are not rejected by other configured authentication methods are treated as anonymous requests. These requests are then served by the API server. You should rely on authentication to authorize access and disallow anonymous requests.

If you are using RBAC authorization, it is generally considered reasonable to allow anonymous access to the API Server for health checks and discovery purposes, and hence this recommendation is not scored. However, you should consider whether anonymous discovery is an acceptable risk for your purposes.

Impact:

Anonymous requests will be rejected.

Audit:

Run the following command on the master node:

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

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

Remediation:

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

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

Compliance

National Institute of Standards and Technology
U.S. Department of Commerce

www.nist.gov

- NIST SP 800-53 rev5
  Security and Privacy Controls for Information Systems and Organizations
- NIST SP 800-190
  Application Container Security Guide

sysdig

NIST SP 800-53 rev5

sysdig

NIST.SP.800-190.pdf

4 Countermeasures for Major Risks

This section recommends countermeasures for the major risks identified in Section 3.

4.1 Image Countermeasures

4.1.1 Image vulnerabilities

There is a need for container technology-specific vulnerability management tools and processes. Traditional vulnerability management tools make many assumptions about host durability and app update mechanisms and frequencies that are fundamentally misaligned with a containerized model. These tools are often unable to detect vulnerabilities within containers, leading to a false sense of safety.

Organizations should use tools that take the pipeline-based build approach and immutable nature of containers and images into their design to provide more actionable and reliable results. Key aspects of effective tools and processes include:

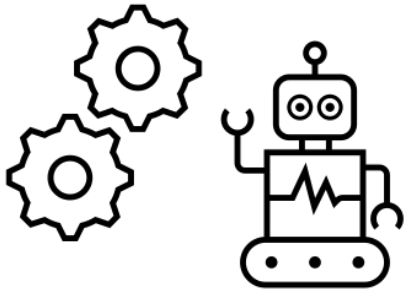
1. Integration with the entire lifecycle of images, from the beginning of the build process, to whatever registries the organization is using, to runtime.
2. Visibility into vulnerabilities at all layers of the image, not just the base layer of the image but also application frameworks and custom software the organization is using. Visibility should be centralized across the organization and provide flexible reporting and monitoring views aligned with organizations' business processes.
3. Policy-driven enforcement; organizations should be able to create "quality gates" at each stage of the build and deployment process to ensure that only images that meet the organization's vulnerability and configuration policies are allowed to progress. For example, organizations should be able to configure a rule in the build process to prevent the progression of images that include vulnerabilities with Common Vulnerability Scoring System (CVSS) [18] ratings above a selected threshold.

sysdig

Demo

Conclusions

- Use benchmarks and common security procedures to increase your security posture
- Document how you are doing that in a compliance report you can show in an audit
- **Automate as much as possible**



sysdig

Questions ?

- Sysdig Blog on NIST 800-53 Compliance for Cloud, Containers, and Kubernetes.
- Sysdig Blog on NIST 800-190 Application Container Security
- Sysdig Secure Capabilities
- 30 Day Free Trial

sysdig

Dig deeper