



ISACA®

Canberra Chapter

May 2021

Using Technology to Help With GRC in Public Cloud and Modern Application Environments

Shain Singh

Cloud/5G Security Architect [APCJ]

shsingh@ieee.org

Who am I?



Shain Singh
Cloud/5G Security Architect @F5

Social

-  <https://linkedin.com/in/shsingh>
-  shsingh@ieee.org
-  <https://twitter.com/shainsingh>
-  <https://github.com/shsingh>
-  <https://shain.io>

Professional Memberships



Why this talk?

Make Security Great Again™

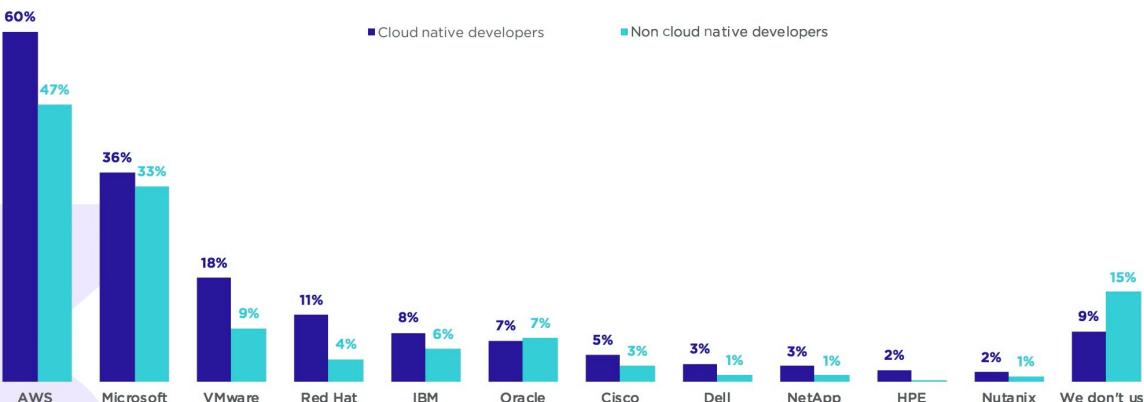
- Blue Teaming should be as fun as Red Teaming
- Create cultural shift in organizations by embracing *DevOps principles*
 - Security should move from a “NO by default” to a “YES with caveats”
 - Meeting developers halfway encourages them to do the same
- Leverage toolsets and methodologies that are becoming common-place for application and infrastructure deployment

Disclaimer

- I am not an expert, I am a curious security practitioner learning how these new technologies can help with raising the bar

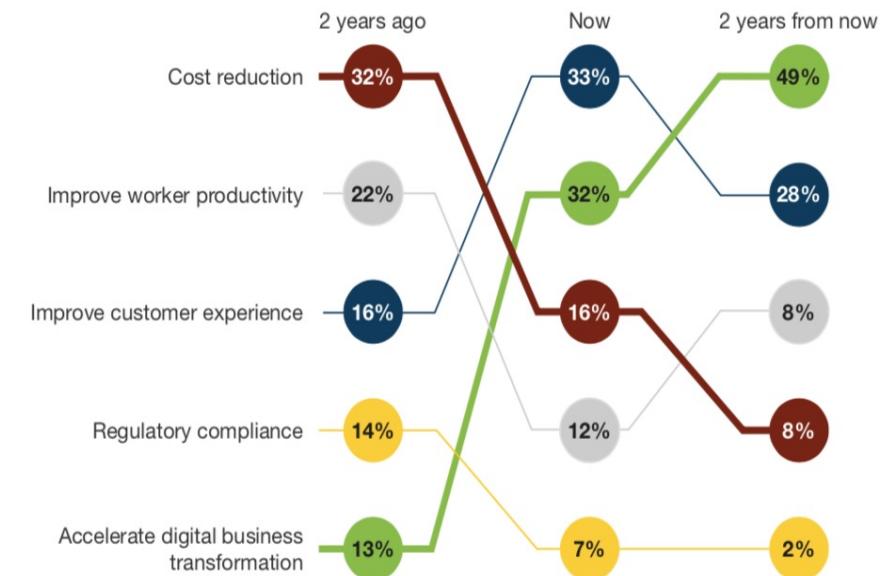
Cloud adoption

Private cloud usage by cloud native and non cloud native developers



[Software Architecture and Design InfoQ Trends Report—April 2020](#)

“What is your primary focus for process improvement efforts?”



[Forrester \[2018\] - The Growing Importance Of Process To Digital Transformation](#)

Container adoption



Honest Status Page
@honest_update

Follow

We replaced our monolith with micro services so that every outage could be more like a murder mystery.

4:10 PM - 7 Oct 2015

3,033 Retweets 2,554 Likes



20

3.0K

2.6K

How the U.S. Air Force Deployed Kubernetes and Istio on an F-16 in 45 days

24 Dec 2019 8:19am, by Tom Krazit



Compliance can assist to set guardrails



CJIS
Criminal Justice
Information Services



DoD SRG
DoD Data Processing



FedRAMP
Government Data
Standards



FERPA
Educational Privacy
Act



FFIEC
Financial Institutions
Regulation



CSA
Cloud Security
Alliance Controls



ISO 9001
Global Quality
Standard



ISO 27001
Security Management
Controls



ISO 27017
Cloud Specific
Controls



ISO 27018
Personal Data
Protection



FIPS
Government Security
Standards



FISMA
Federal Information
Security Management



GxP
Quality Guidelines
and Regulations



HIPAA
Protected Health
Information



HITRUST CSF
Health Information
Trust Alliance
Common Security
Framework



FISC [Japan]
Financial Industry
Information Systems



IRAP [Australia]
Australian Security
Standards



K-ISMS [Korea]
Korean Information
Security



**MTCS Tier 3
[Singapore]**
Multi-Tier Cloud
Security Standard



**OSPAR
[Singapore]**
Outsourcing
Guidelines

Industry standards define deployment patterns



[Cloud Controls Matrix](#)

[Security Guidance For Critical Areas of Focus in Cloud Computing](#)



[Benefits, Risks and Recommendations For Information Security](#)



[Cybersecurity Framework](#)

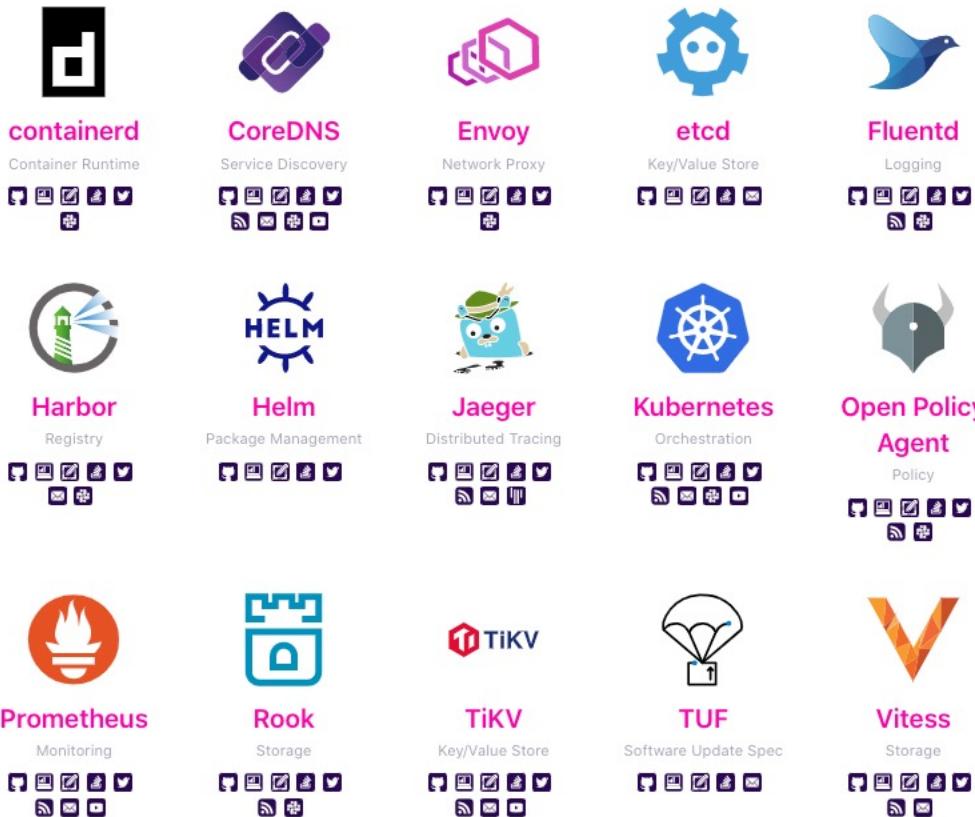


[Secure Cloud Computing Architecture](#)

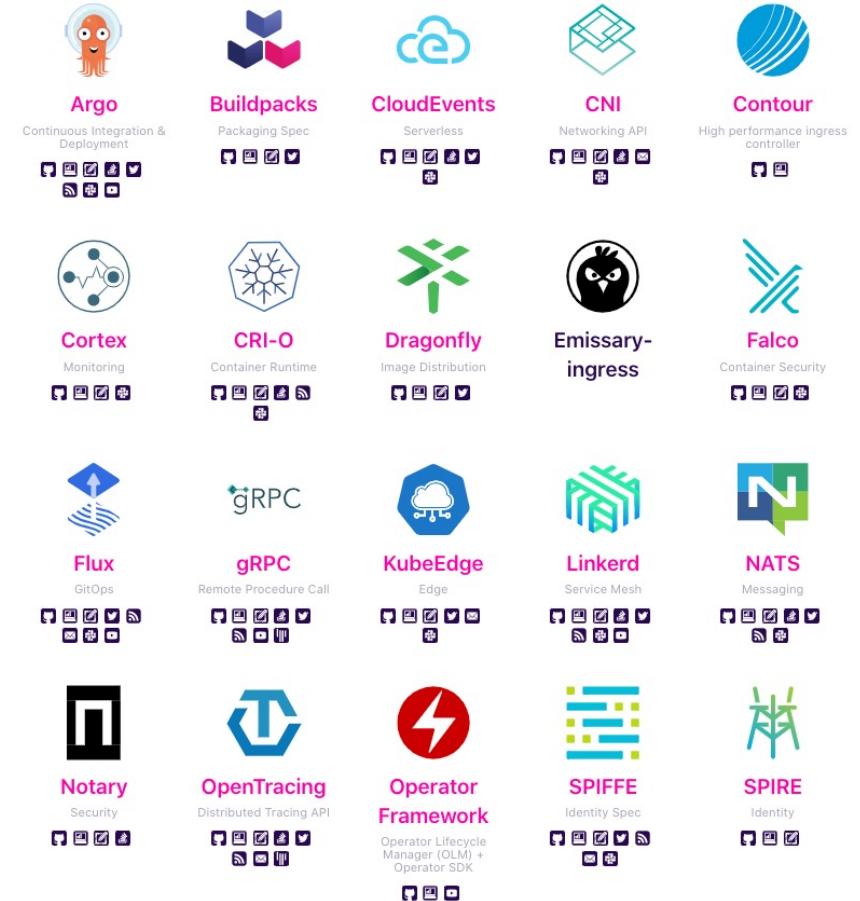


[CIS Benchmarks](#)

Cloud native technologies



Graduated Projects



Incubating Projects

Policy and Controls

Why use a service mesh?

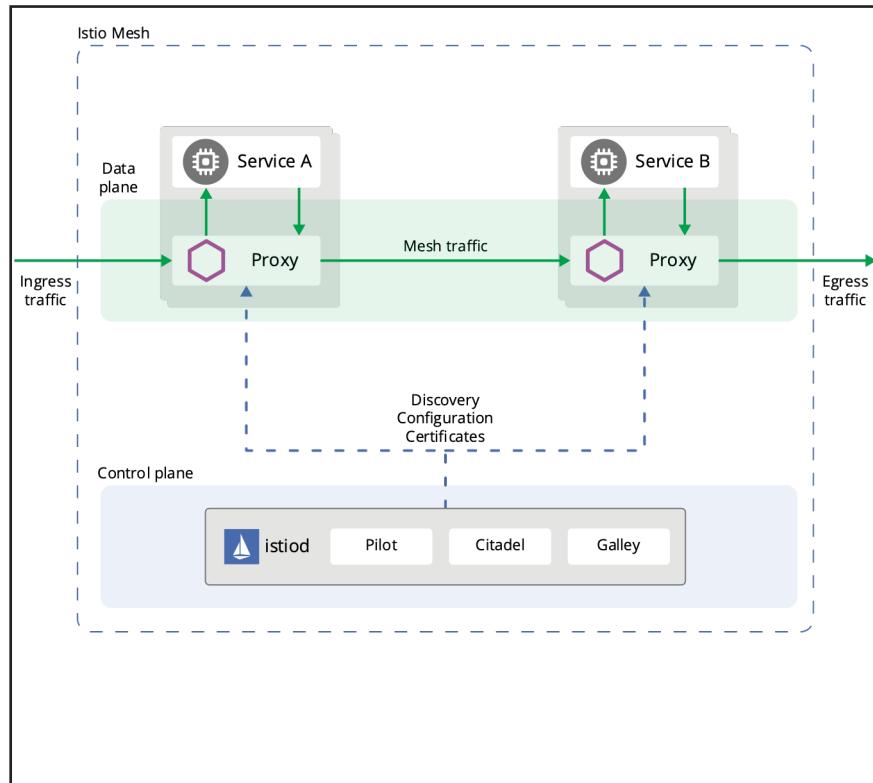
- The distributed cross-domain nature of microservices needs **secure token service (STS), key management and encryption services for authentication and authorization, and secure communication protocols**.
- The ephemeral nature of clustered containers (by which microservices are implemented) calls for secure service discovery.
- The availability requirement calls for:
 - (a) resiliency techniques, such as load balancing, circuit breaking, and throttling
 - (b) continuous monitoring (for the health of the service).
- The **service mesh is the best-known approach that can facilitate specification of these requirements** at a level of abstraction such that it can be uniformly and consistently defined while also being effectively implemented without making changes to individual microservice code.

[NIST SP 800-204A - Building Secure Microservices-based Applications Using Service-Mesh Architecture](#)

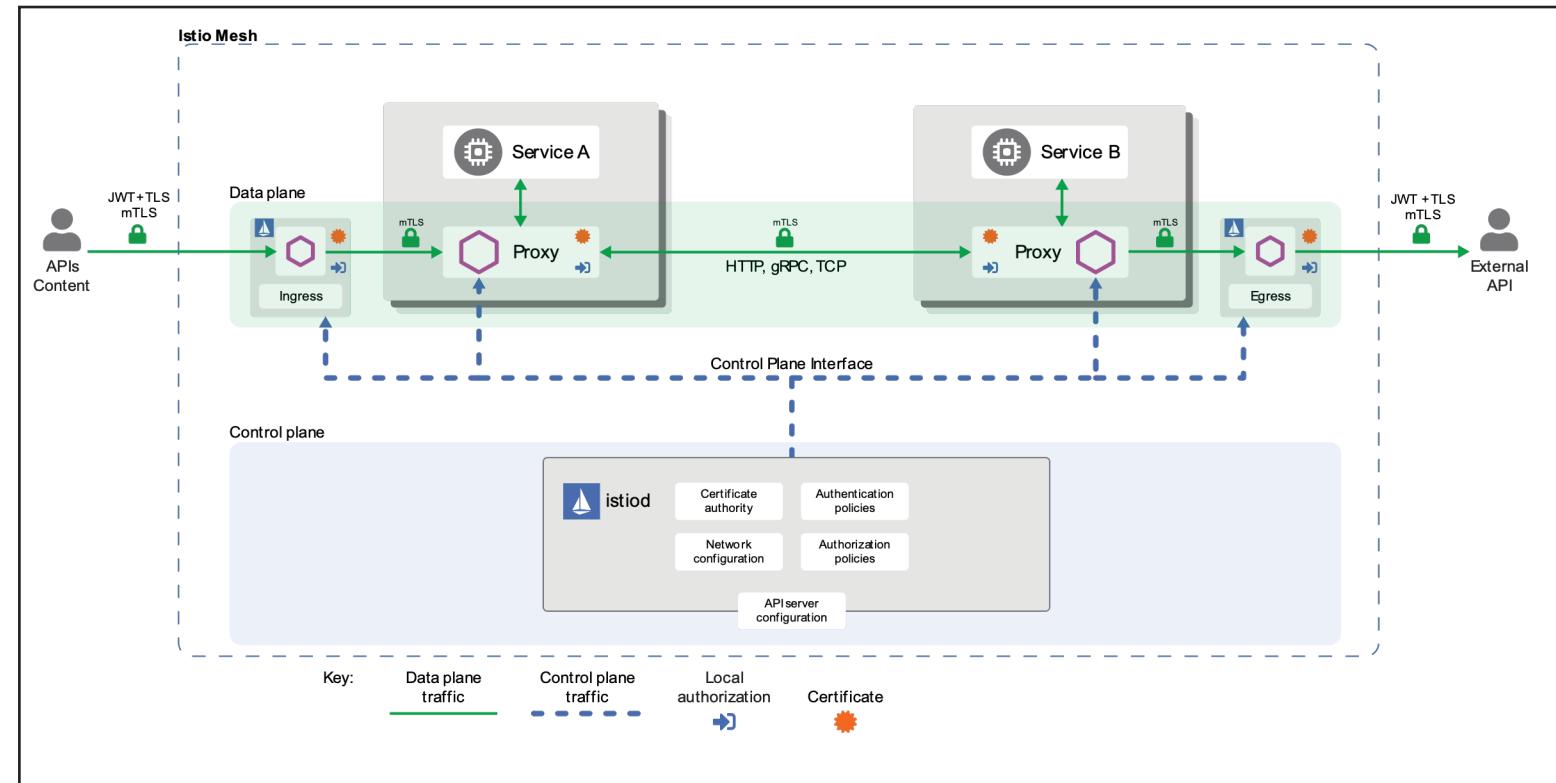
- Deployment architecture in cloud-native applications now consists of **loosely coupled components** (microservices), with all application services provided through a **dedicated infrastructure (service mesh)** independent of the application code.
- Two critical security requirements in this architecture are
 - (a) to build the concept of **zero trust by enabling mutual authentication** in communication between any pair of services
 - (b) a **robust access control mechanism** based on an access control model such as Attribute-based Access Control (ABAC) that can be used to express a wide set of policies and is scalable in terms of user base, objects (resources), and deployment environment.

[NIST SP 800-204B - Attribute-based Access Control for Microservices-based Applications using a Service Mesh](#)

What is a service mesh?



Components



Security Architecture

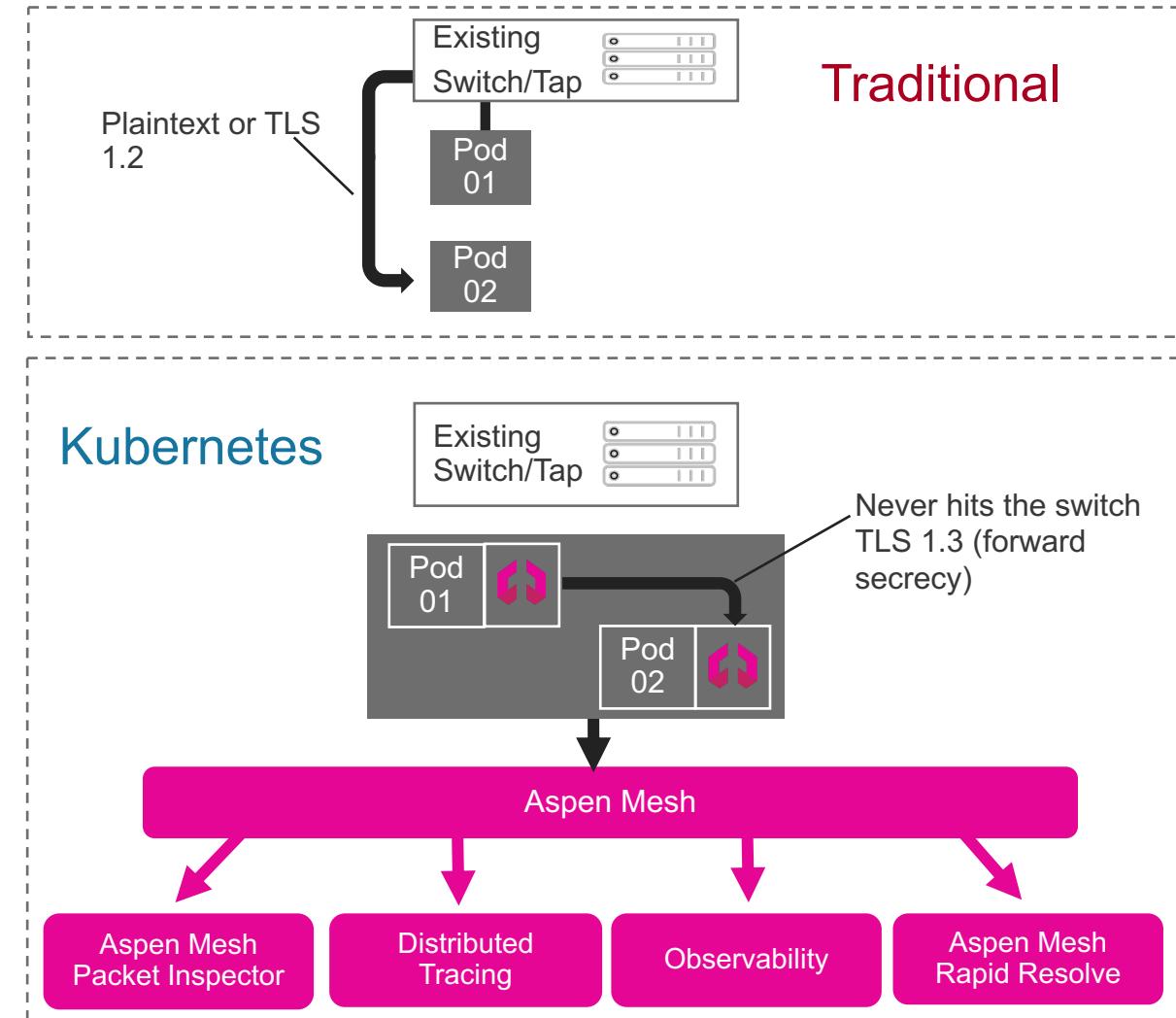
Example – vendor implementation of service mesh

Challenges

- Packet-level inspection of flows in container environment
- Key management and mTLS 1.3 PFS challenges
- Lawful intercept and compliance requirements
- Leverage existing packet broker investment
- Operations troubleshooting, knowledge and training

Solution: Aspen Mesh Packet Inspector

- Inter-service capture at sidecar
- Pre-encryption tapping
- Compatible with TLS 1.3 Forward Secrecy
- Integrates into existing infrastructure & automation
- Scalable and extensible



What are SPIFFE/SPIRE?



- A set of specifications that cover how a workload should retrieve and use its identity
 - SPIFFE ID
 - SPIFFE Verifiable Identity Documents (SVIDs)
 - The SPIFFE Workload API

<https://spiffe.io/docs/latest/spiffe-about/overview/>

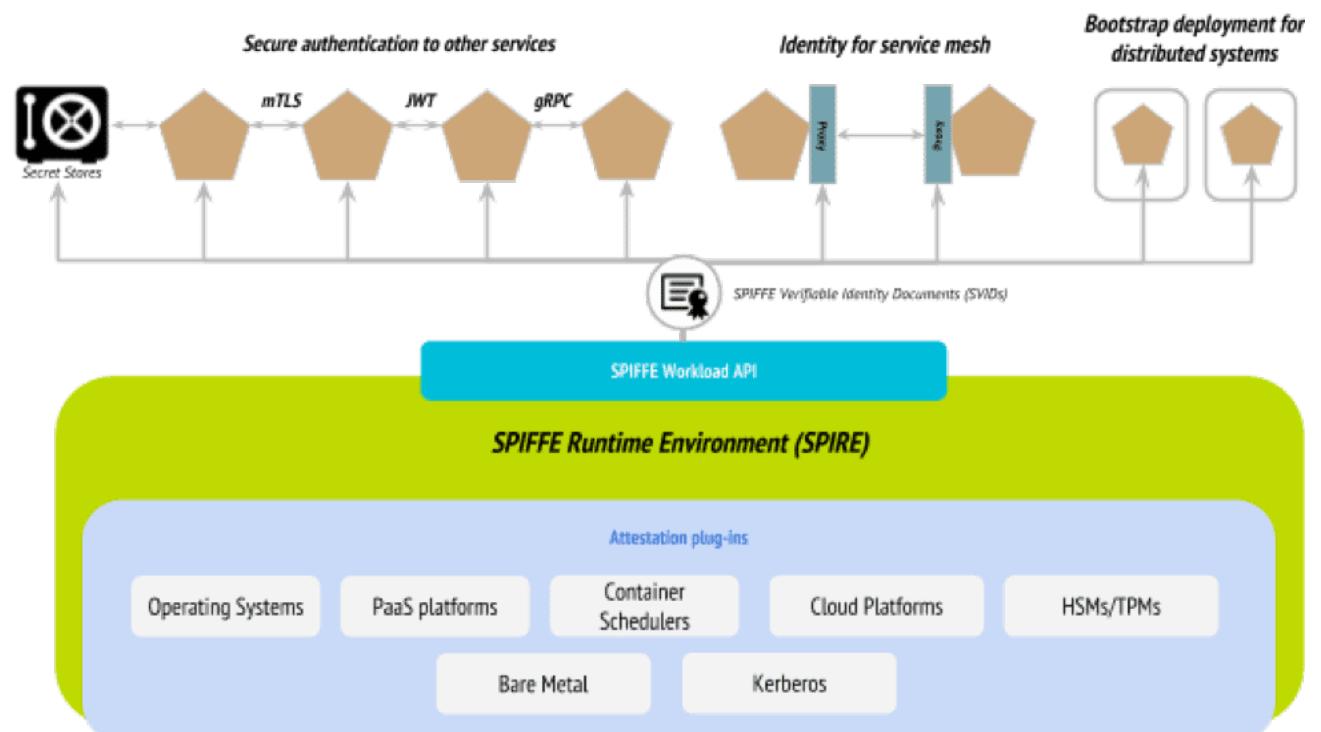
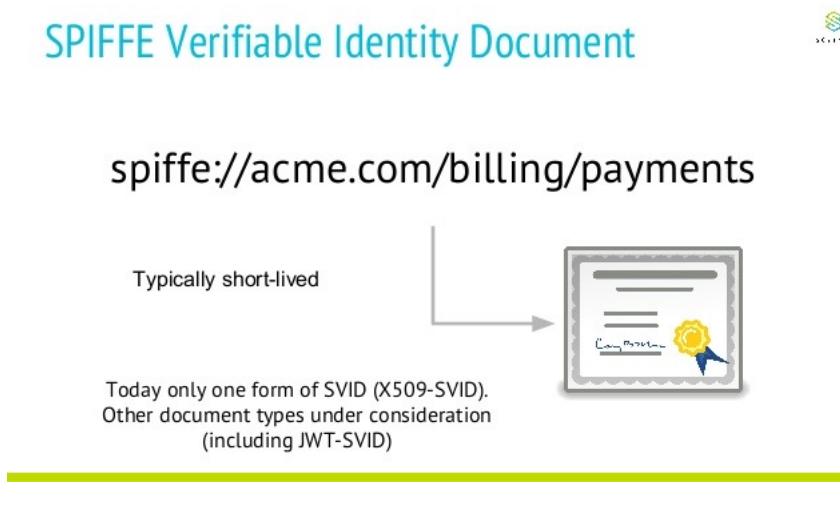


- The SPIFFE Runtime Environment
- Open-source Reference Implementation that applies the SPIFFE Workload API for a variety of platforms and environments
- Highly extensible through plug-ins

<https://spiffe.io/docs/latest/spire-about/>

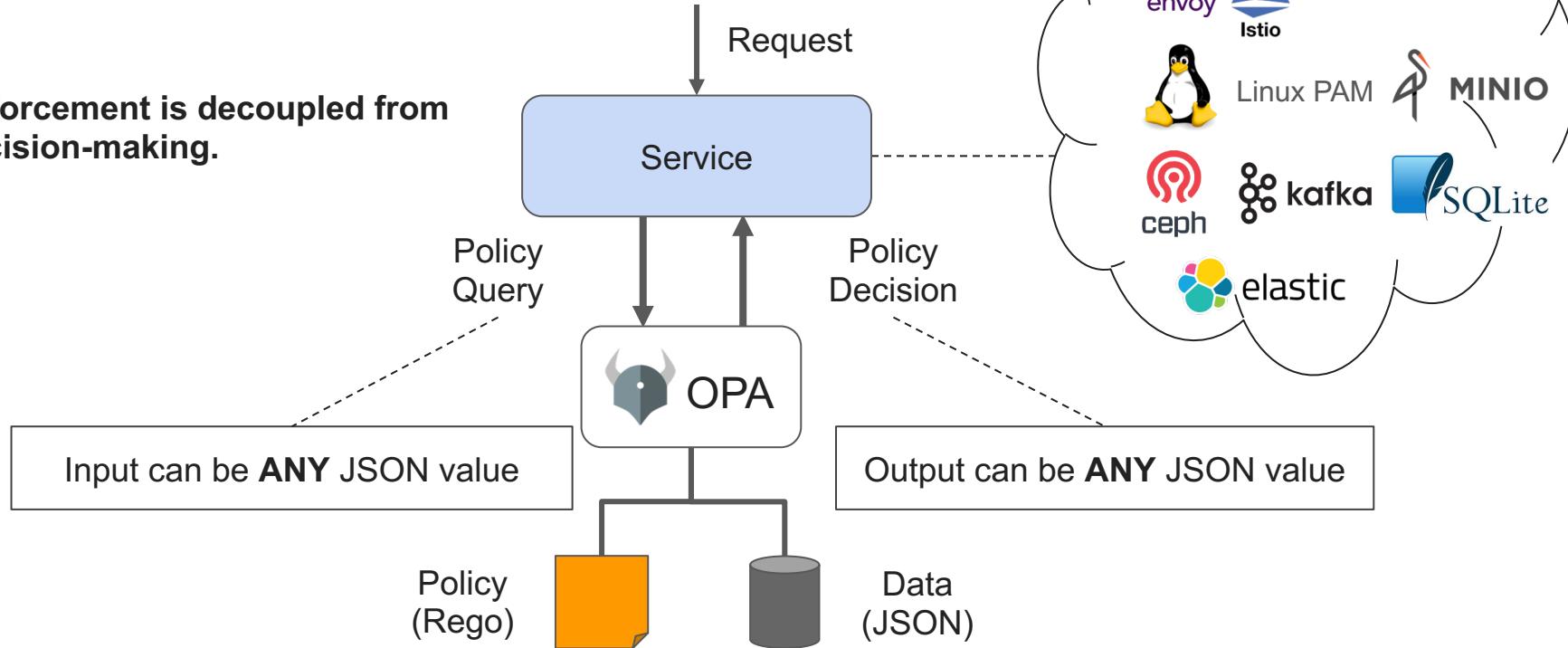
SPIFFE Overview

SPIFFE Verifiable Identity Document



What is OPA?

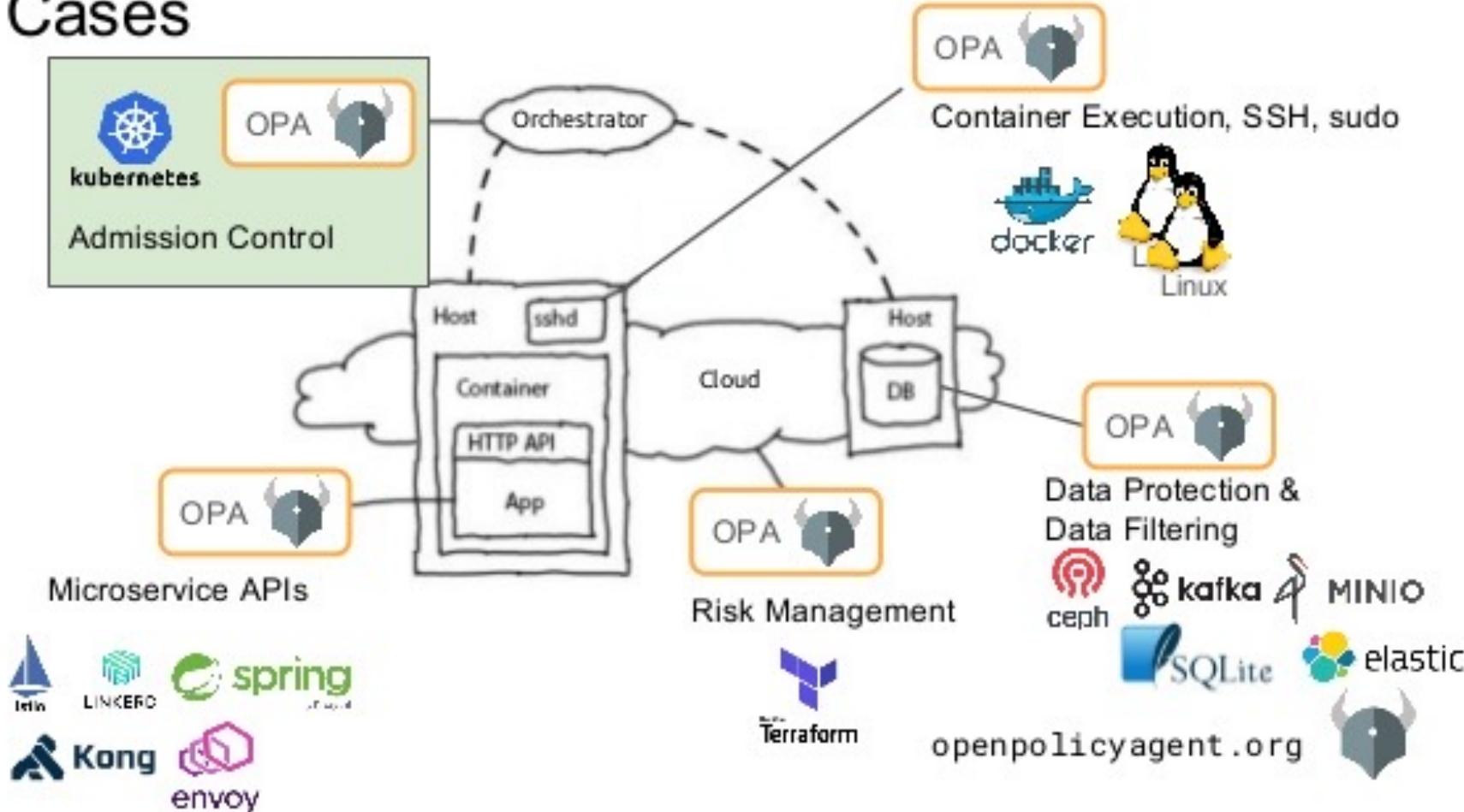
Enforcement is decoupled from decision-making.



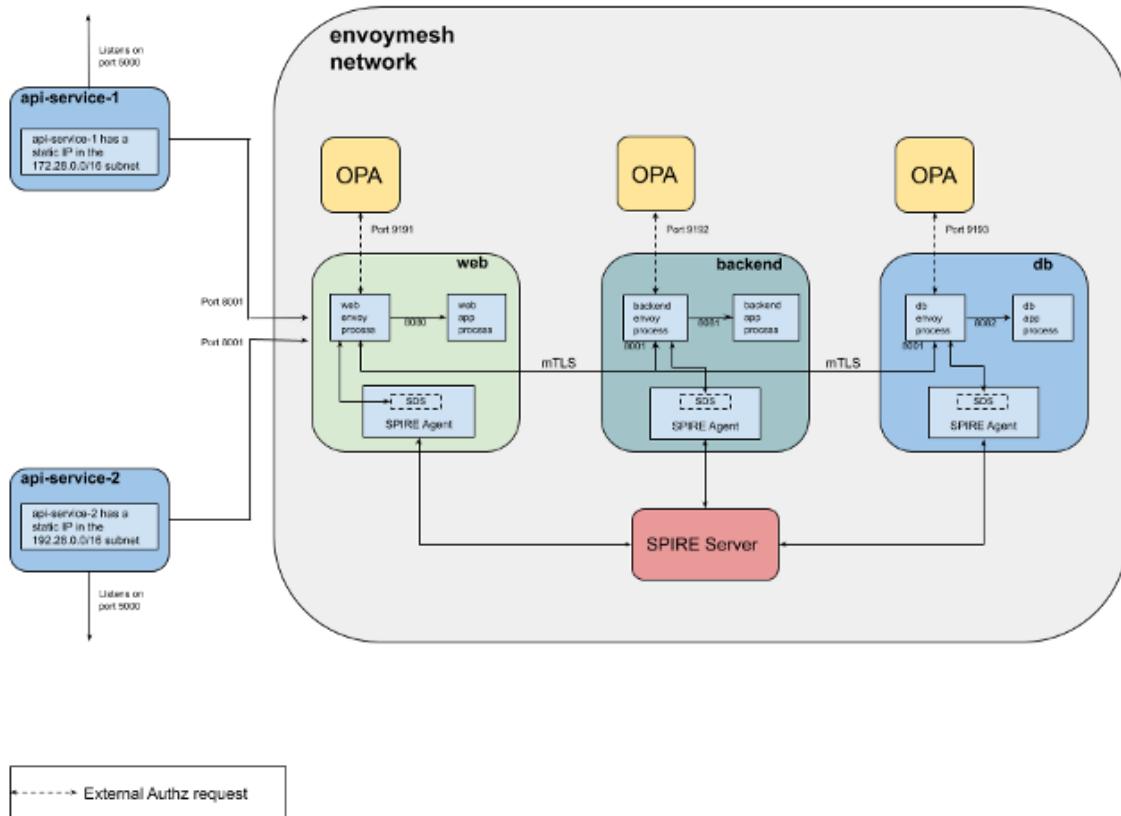
<https://www.openpolicyagent.org/>

OPA Overview

Use Cases



Example – Istio + SPIFFE + OPA



```
package envoy.authz

import input.attributes.request.http as http_request
import input.attributes.source.address as source_address

default allow = false

# allow Backend service to access DB service
allow {
    http_request.path == "/good/db"
    http_request.method == "GET"
    svc_spiffe_id == "spiffe://domain.test/backend-server"
}

svc_spiffe_id = client_id {
    _, _, uri_type_san := split(http_request.headers["x-forwarded-client-cert"], ";")
    _, client_id := split(uri_type_san, "=")
}
```

Compliance as Code

Compliance as Code



<https://www.open-scap.org/>

- Set of open-source tools for security compliance and vulnerability assessment
 - Security Content Automation Protocol (SCAP) is a framework that supports automated configuration, vulnerability and patch checking, technical control compliance activities, and security measurement
 - SCAP standard includes:
 - Extensible Configuration Checklist Description Format (XCCDF)
 - Open Vulnerability and Assessment Language (OVAL)
 - DataStream
 - Asset Reporting Format (ARF)
 - Common Platform Enumeration (CPE)
 - Common Vulnerabilities and Exposures (CVE)
 - Common Weakness Enumeration (CWE)



<https://inspec.io/>

- Open-source testing framework with human- and machine-readable language for specifying compliance, security and policy requirements
- Uses Infrastructure as Code principles to keep compliance in Source Code Management (SCM)
- Tests can be run locally, remotely or as part of CI/CD pipelines for continuous compliance
- Highly extensible and support for large ecosystem of software

Inspec

The MITRE Corporation
Open Source Software from the MITRE Corporation
http://mitre.github.io opensource@mitre.org

Repositories 270 Packages People 15 Projects 1

stig Type Language Sort

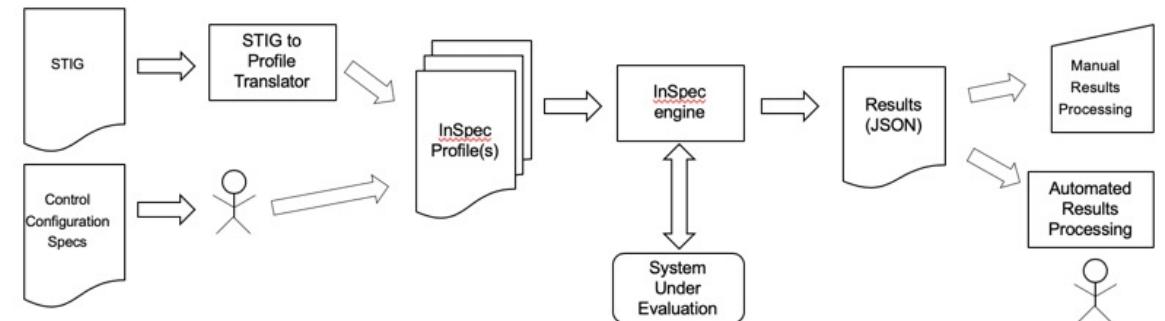
46 results for repositories matching stig sorted by last updated Clear filter

DevSec Hardening Framework
Security + DevOps: Automatic Server Hardening
https://twitter.com/devsecio https://dev-sec.io

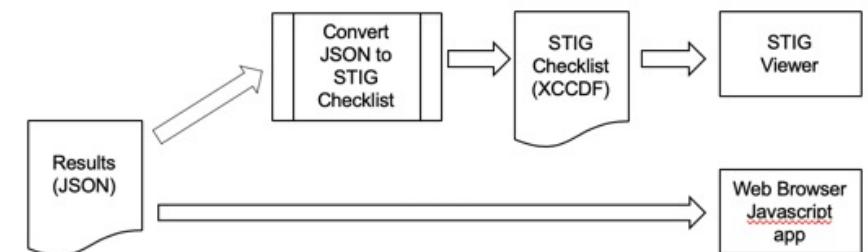
Repositories 47 Packages People 19 Projects 1

Pinned repositories

- ansible-collection-hardening: This Ansible collection provides battle tested hardening for Linux, SSH, nginx, MySQL. (Jinja, 2.2k stars, 423 forks)
- chef-os-hardening: This chef cookbook provides numerous security-related configurations, providing all-round base protection. (Ruby, 389 stars, 134 forks)
- puppet-os-hardening: This puppet module provides numerous security-related configurations, providing all-round base protection. (Puppet, 237 stars, 85 forks)
- linux-baseline: DevSec Linux Baseline - InSpec Profile (Ruby, 547 stars, 131 forks)
- cis-docker-benchmark: CIS Docker Benchmark - InSpec Profile (Ruby, 330 stars, 70 forks)
- cis-kubernetes-benchmark: CIS Kubernetes Benchmark - InSpec Profile (Ruby, 242 stars, 54 forks)



Automating Security Validation Using InSpec



Processing InSpec Results

Example – DevSecOps + Inspec

running #308839490 latest master -> b004c8a0 remove previous sec ci sta... In progress sec-compliance: passed with warnings

passed #308837380 latest master -> b004c8a0 remove previous sec ci sta... 00:06:53 14 minutes ago

Pipeline Needs Jobs 4 Failed Jobs 2 Tests 0

Sec-pre_build	Sec-package	Sec-release	Sec-compliance
sec-source... (green)	sec-os_hard... (orange)	sec-dast_ba... (green)	sec-complia... (orange)

sec-package.gitlab-ci.yml 760 Bytes

Edit Web IDE

```
1 services:
2   - docker:dind
3
4 sec-os_hardening:
5   stage: sec-package
6   image: ansible/galaxy
7   before_script:
8     - mkdir -p ~/.ssh
9     - echo "$DEPLOYMENT_SERVER_SSH_PRIVKEY" | tr -d '\r' > ~/.ssh/id_rsa
10    - chmod 600 ~/.ssh/id_rsa
11    - eval "$(ssh-agent -s)"
12    - ssh-add ~/.ssh/id_rsa
13    - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
14   script:
15     - echo "[prod]" >> inventory.ini
16     - echo "$DEPLOYMENT_SERVER" >> inventory.ini
17     - export ANSIBLE_STDOUT_CALLBACK=json
18     - ansible-galaxy install dev-sec.os-hardening
19     - ansible-playbook -i inventory.ini ansible-hardening.yml > sec-os_hardening-results.json
20   artifacts:
21     paths: [sec-os_hardening-results.json]
22     when: always
23     expire_in: one week
24   allow_failure: true
```

sec-compliance.gitlab-ci.yml 694 Bytes

Edit Web IDE Lock Replace Delete

```
1 services:
2   - docker:dind
3
4 sec-compliance:
5   stage: sec-compliance
6   image:
7     name: chef/inspec
8   only:
9     - "master"
10    environment: production
11   before_script:
12     - mkdir -p ~/.ssh
13     - echo "$DEPLOYMENT_SERVER_SSH_PRIVKEY" | tr -d '\r' > ~/.ssh/id_rsa
14     - chmod 600 ~/.ssh/id_rsa
15     - eval "$(ssh-agent -s)"
16     - ssh-add ~/.ssh/id_rsa
17     - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
18   script:
19     - inspec exec https://github.com/dev-sec/linux-baseline -t ssh://root@$DEPLOYMENT_SERVER -i /id_rsa --chef-license accept --reporter json:/opt/sec-
20   artifacts:
21     paths: [sec-compliance-results.json]
22     when: always
23   allow_failure: true
```

Challenges

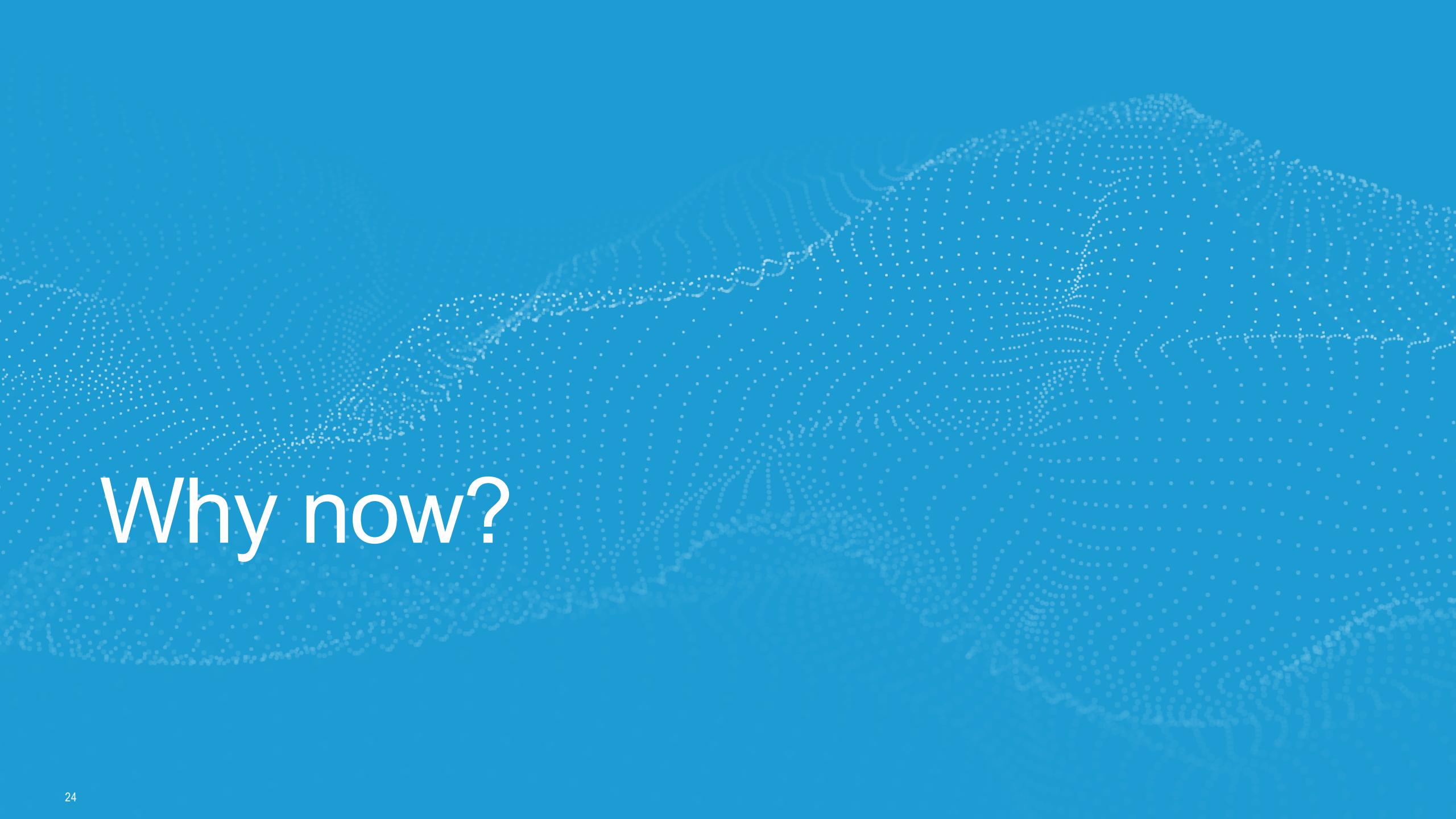
Challenges

Handling exceptions

- Use of third-party software
 - How should vendor software be handled when security and compliance issues are found (e.g. break the build process?)
- What mechanisms to use for alert management from security pipeline processes?
 - Multiple integration scenarios such as Jira for bug/defect tracking and DefectDojo for security violation tracking

Organisational Culture

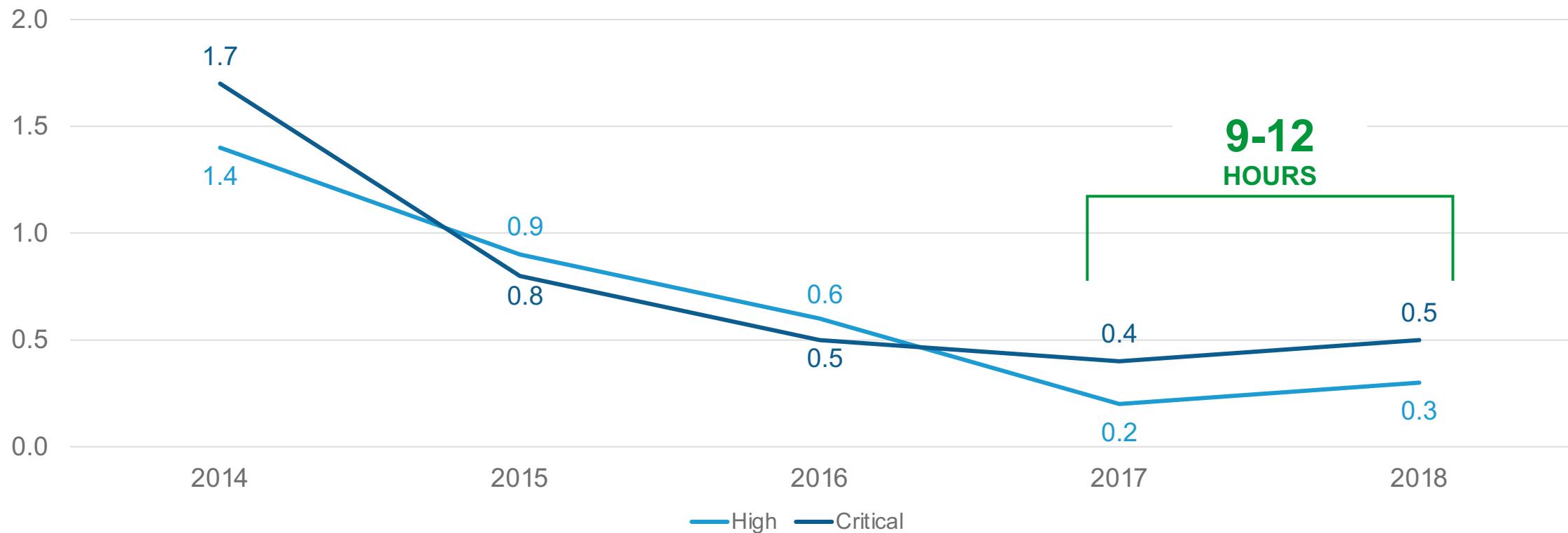
- Moving to a DevSecOps way of working requires significant work
 - People are almost always the hardest to change (DevSecOps involves People, Process and Technology)



Why now?

Protecting against Abuse of Functionality

Average days between “HIGH” AND “CRITICAL” CVEs released



Protecting against Abuse of Intent



The Automated Threat Handbook Web Applications

The Automated Threat Handbook provides actionable information and resources to help defend against automated threats to web applications.

- OAT-020 Account Aggregation
- OAT-019 Account Creation
- OAT-003 Ad Fraud
- OAT-009 CAPTCHA Defeat
- OAT-010 Card Cracking
- OAT-001 Carding
- OAT-012 Cashing Out
- OAT-007 Credential Cracking
- OAT-008 Credential Stuffing
- OAT-021 Denial of Inventory
- OAT-015 Denial of Service
- OAT-006 Expediting
- OAT-004 Fingerprinting
- OAT-018 Footprinting
- OAT-005 Scalping
- OAT-011 Scraping
- OAT-016 Skewing
- OAT-013 Sniping
- OAT-017 Spamming
- OAT-002 Token Cracking
- OAT-014 Vulnerability Scanning

Further Information

References

- [NIST DevSecOps](#)
- [NIST 800-24A Building Secure Microservices-based Applications Using Service-Mesh Architecture](#)
- [NIST 800-24B Attribute-based Access Control for Microservices-based Applications using a Service Mesh](#)

Technical

- [DoD Enterprise DevSecOps Initiative](#)
- [Security Hardening and Baseline profiles](#)
- [MITRE STIG Inspec profiles](#)

Slides available at

<https://oi.shain.io/presentations>