Security as a Platform

A Security-first Approach to Product Innovation

Frank Macreery

Aptible

Who is this for?

Anyone who has ever…

- Spent time collecting evidence to **prove security controls** are in place during an audit
- Spent time **reviewing and fixing misconfigurations** detected by a security posture management (CSPM) product
- Spent time choosing the **right combination of security products** to protect a cloud environment
- **Re-architected a system** to meet a new security or compliance requirement
- Felt like they've been **repeating these same security and compliance tasks** across multiple projects

Status quo: it takes many separate solutions to solve DevOps, Security and Compliance.

Then, they layer in compliance software to ensure they're meeting requirements

STEP 3: COMPLIANCE LAYER

PRISMA BY PALO ALTO NETWORKS · Cloud Conformity · DATADOG · CLOUDVISORY Now a part of FireEye · Lacework

When compliance goals change, the DevOps platform may need to be re-architected, or new security solutions may need to be added

Then, they layer in security point solutions relevant to their architecture and risk profile

STEP 2: SECURITY LAYER

aqua run free · tenable · threat stack · CLOUDFLARE · ALERT LOGIC · CROWDSTRIKE · snyk

First, they buy or build a DevOps platform to deploy code to the cloud

STEP 1: DEVOPS LAYER

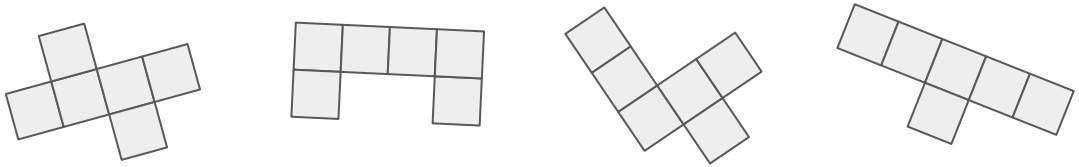
aws · Google Cloud · Azure · HEROKU · kubernetes

When changes are made to the DevOps platform, the security and compliance layers must also be updated, or else unknown vulnerabilities can arise

PROBLEM #1

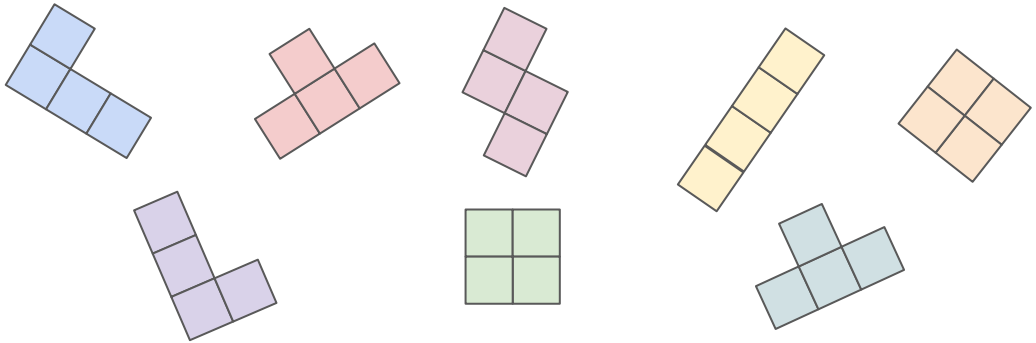
Applying **finite compliance rules** to **infinitely complex infrastructure** doesn't work.

SECURITY AND COMPLIANCE LAYERS



Security posture software (CSPM) defines compliance in terms of **finite configuration rules**...

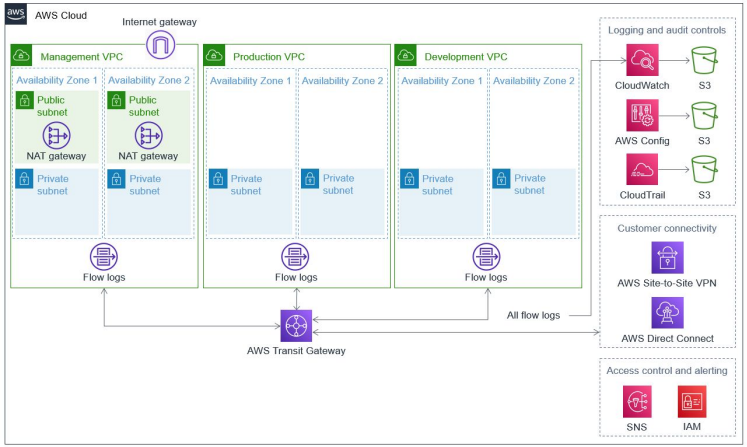
DEVOPS/INFRASTRUCTURE LAYER

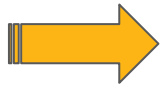


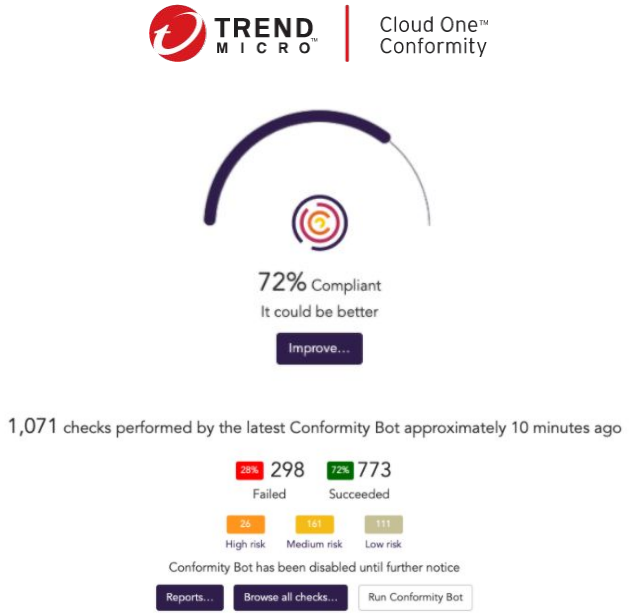
...but infrastructure is architected in complex ways, with **interactions between each unique component** that **change over time**, and much security risk surface area that's **not defined in any configuration.**

This introduces false positives…

AWS HIPAA Reference Architecture

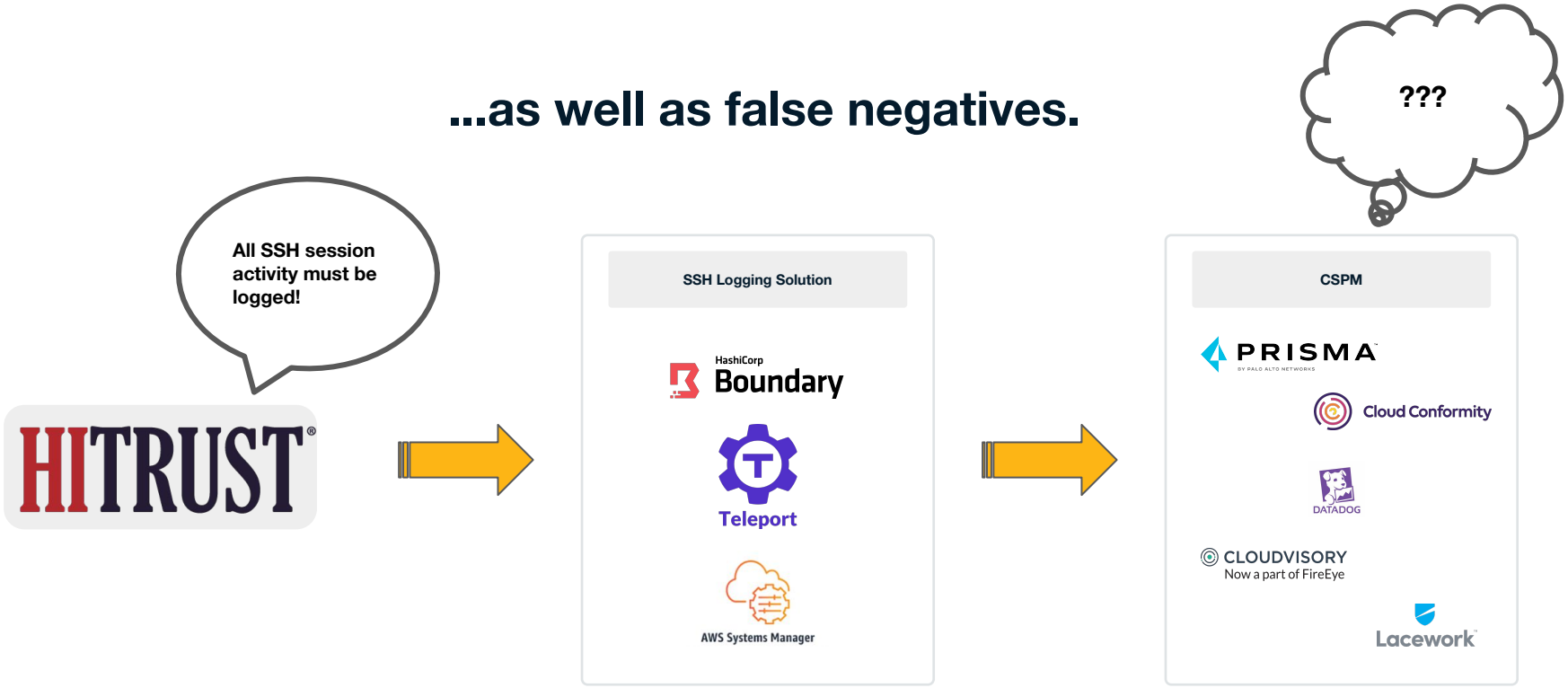






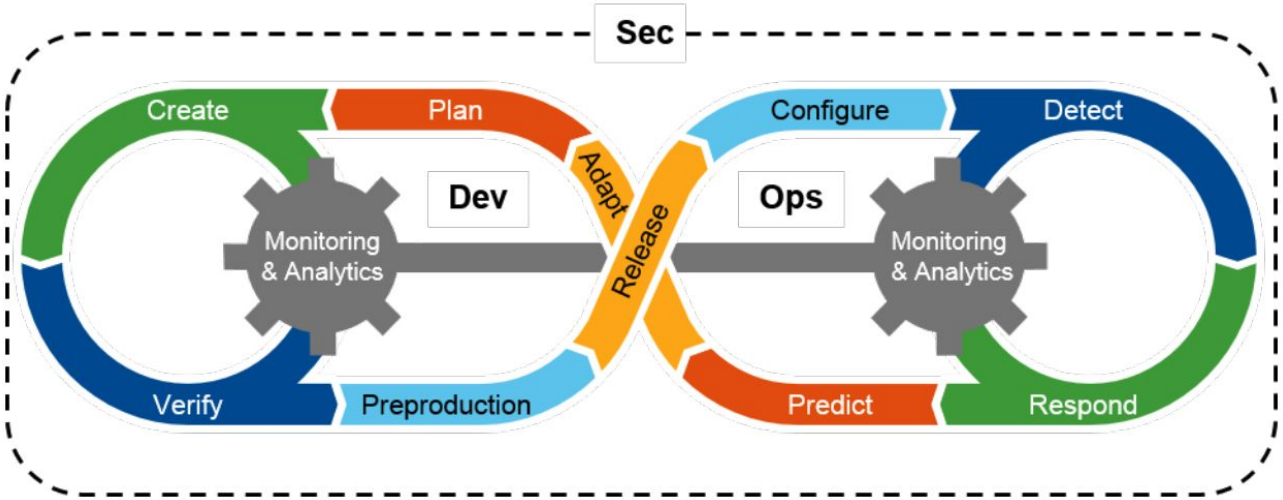
Link: [HIPAA Reference Architecture on the AWS Cloud: Quick Start Reference Deployment]

...as well as false negatives.



PROBLEM #2

The traditional DevSecOps lifecycle is an **unwinnable, infinite loop**.

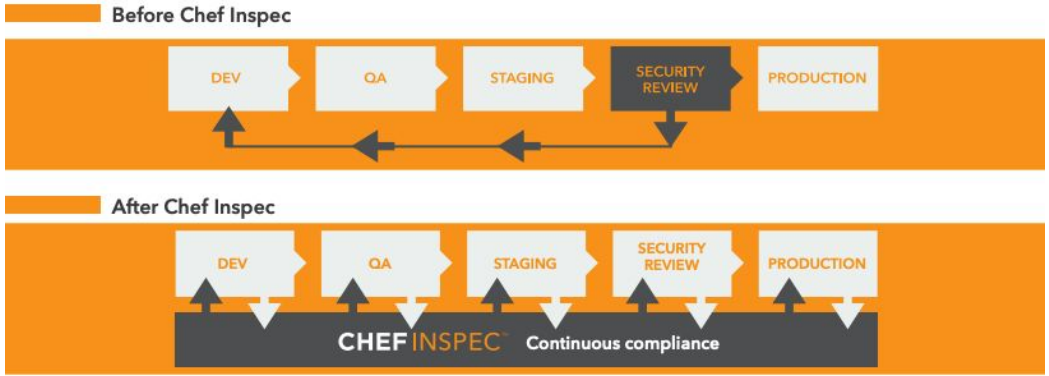


Every new development triggers new work to prevent and detect security incidents

Every new compliance requirement or security incident triggers new development

Source: DevSecOps: How to Seamlessly Integrate Security Into DevOps, Gartner

What about Compliance-as-Code?



Early planning/communication prevents rework

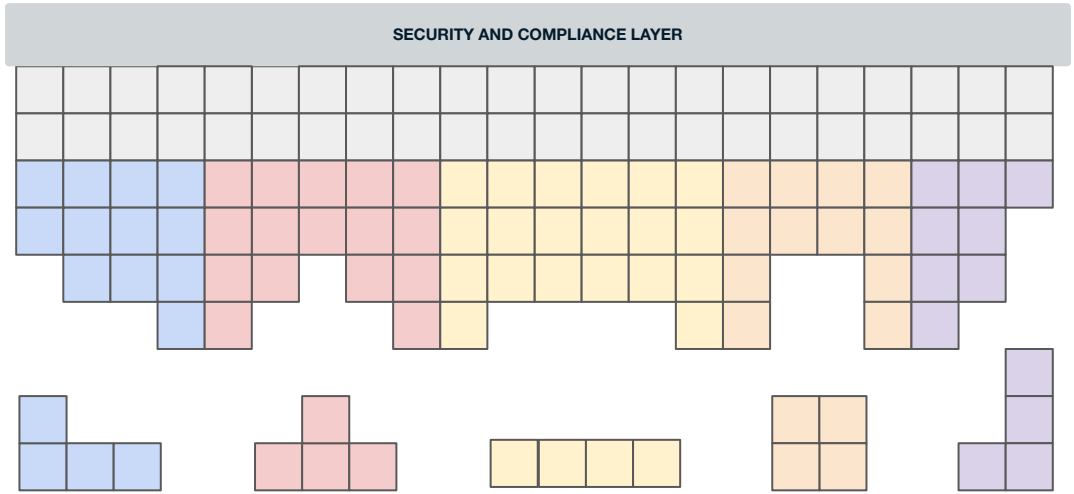
Still, there's an infinite feedback loop between development and security requirements

Source: [Chef InSpec Datasheet]

SOLUTION

Provide **security as a platform**, limiting flexibility to ensure **security upfront**.

A security platform limits the components one can build with, but bakes in security for every component.



Pillars of Security as a Platform

SIMPLIFY	CONTROL	ENFORCE
Only support components you know how to secure	Monitor and alert on any changes made outside of the platform	Compliance layer should enforce, not just monitor

SIMPLIFY

Only support components you know how to secure

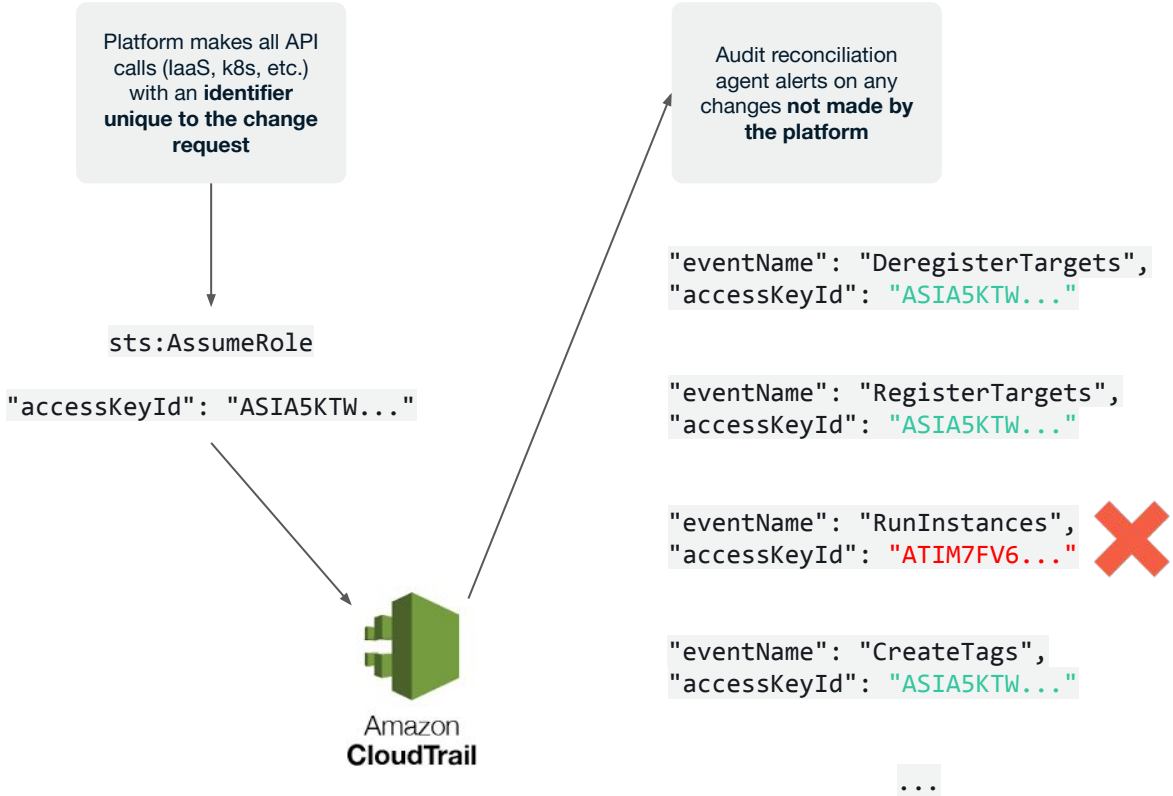
Limit flexibility...

SECURITY/COMPLIANCE REQUIREMENT	PLATFORM COMPONENTS
All app dependencies (npm, Rubygems, etc.) must be **scanned for vulnerabilities** before deployment to production.	Require **Git repos** as inputs for app deployment, as opposed to **pre-built Docker images**. Then, the platform can scan the repo and build + deploy only if no vulnerabilities are found.
All endpoints must be scanned daily by a **dynamic application security tool** (DAST).	Require clients to specify all web-listening ports. **Expose only these ports** via load balancers, firewall rules, etc. and set up automated daily scans.
All production web applications must be **distributed across multiple data centers** or availability zones (AZs) for high availability.	Don't let clients specify application placement. Instead: **accept only a container size and count** (>=2), and schedule containers across AZs.

CONTROL

Monitor and alert on any changes made outside of the platform

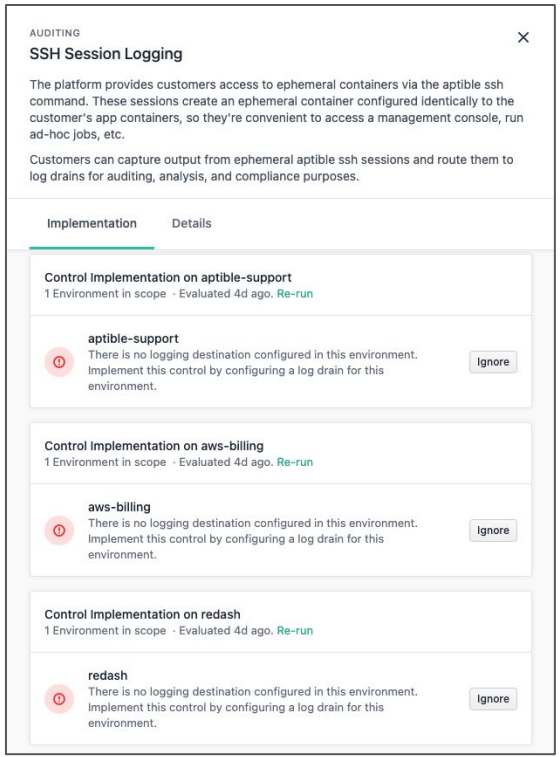
Limit flexibility...

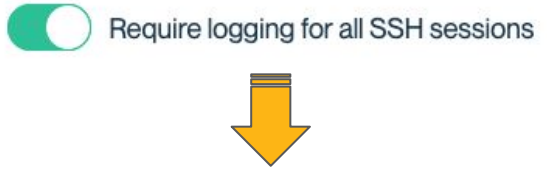


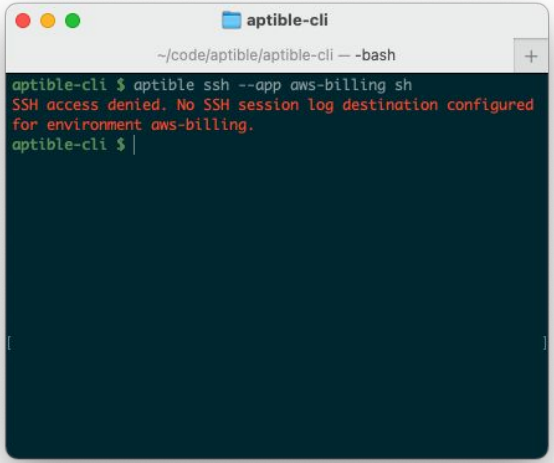
ENFORCE

Compliance layer should enforce, not just monitor

...to provide security upfront







What's the catch?

Building a security platform is hard...

- If the platform doesn't anticipate future compliance requirements, you still have an **infinite loop** between security and development. So the **platform builders must also be experts in compliance** and the common principles underlying all compliance frameworks, present and future.
- Designing components to **solve a general problem** takes more effort than solving a **specific instance** of the problem.

What's the catch?

...but it can be done.

- Larger enterprises are already building internal platforms to align DevOps performance gains with their own unique compliance requirements.[1] Building **one internal platform to service many application teams** ends up saving money and improving security.
- For the rest of us, there's value in developing a **common public security platform**.

Per the *2020 Puppet State of DevOps Report*, 63% of respondents say their company has at least one self-service internal platform.


Aptible