

Clear Skies: Avoiding Security Breaches in AWS

Nick Jones – AWS User Group Malmö, March 2024

aws sts get-caller-identity

Nick Jones – @nojonesuk

- Head of Research @ WithSecure
- Ex-Cloud Security Consulting Lead
- AWS Community Builder
- Focus on:
 - Security automation
 - Attack detection
 - Making security work for engineering teams



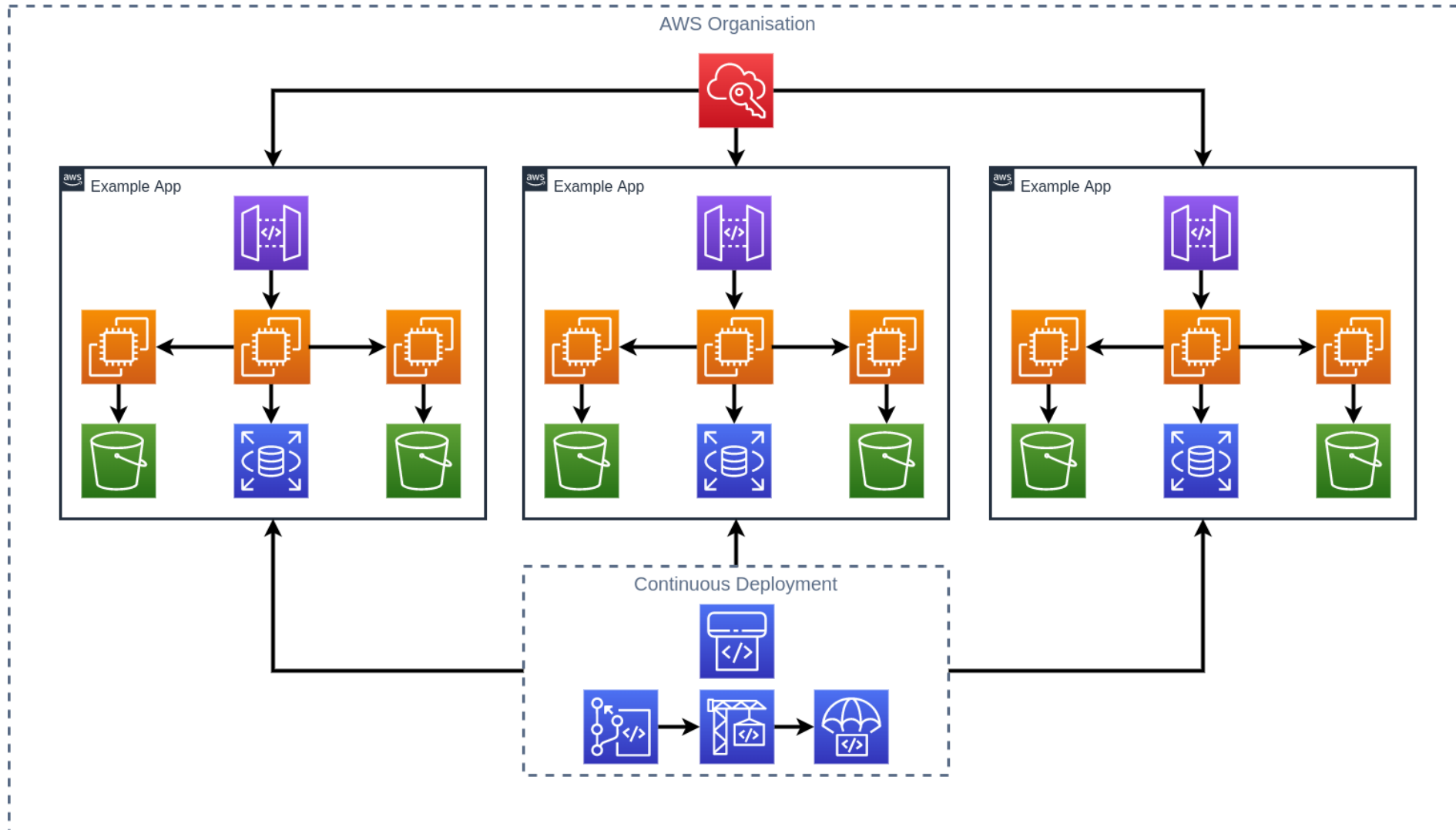
Agenda

- 
- Real World Breach Scenarios
 - Other Interesting Attack Vectors
 - Key Security Controls
 - Security Testing Done Right
 - Collaborating with Security Partners

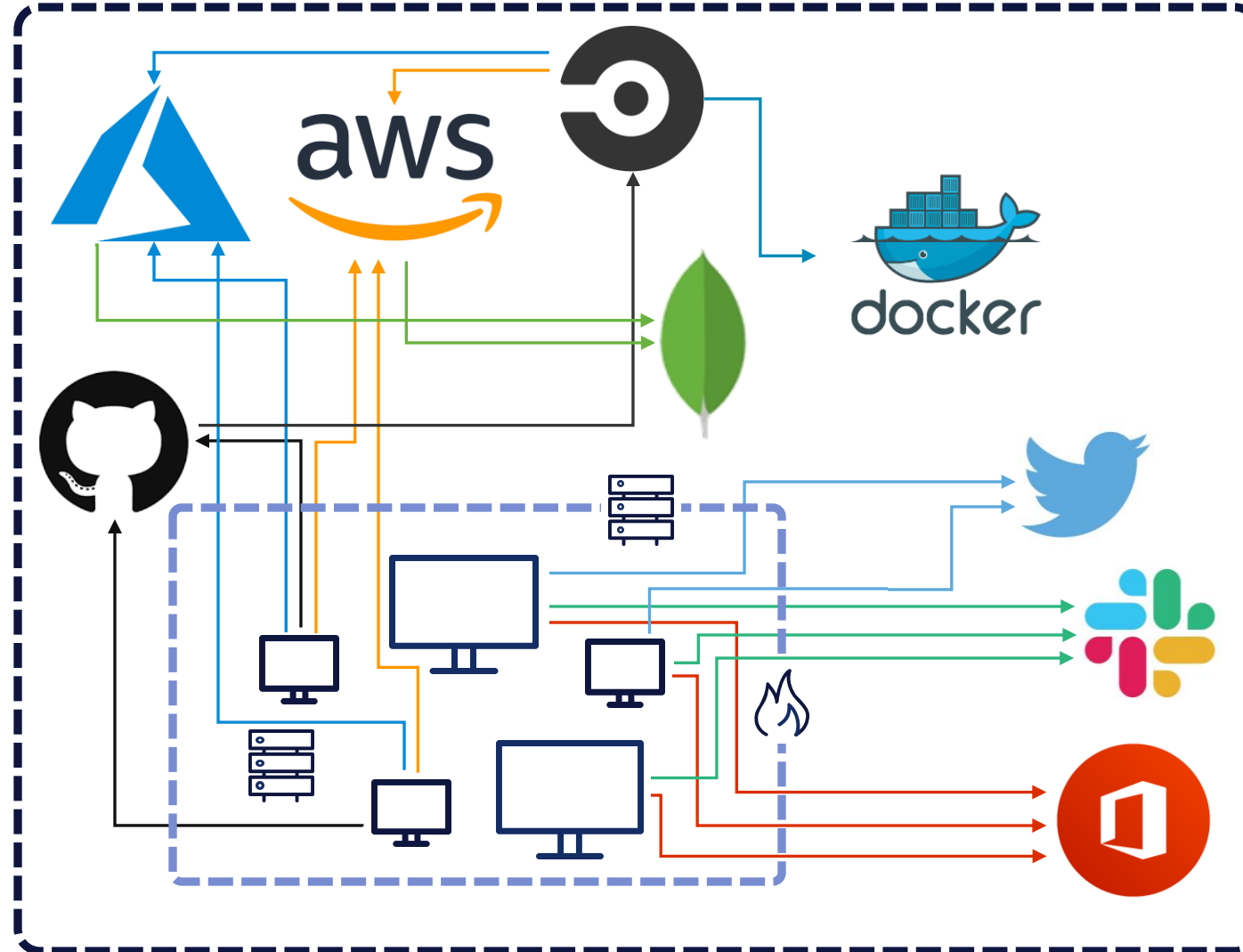
Security's Idea of the Cloud



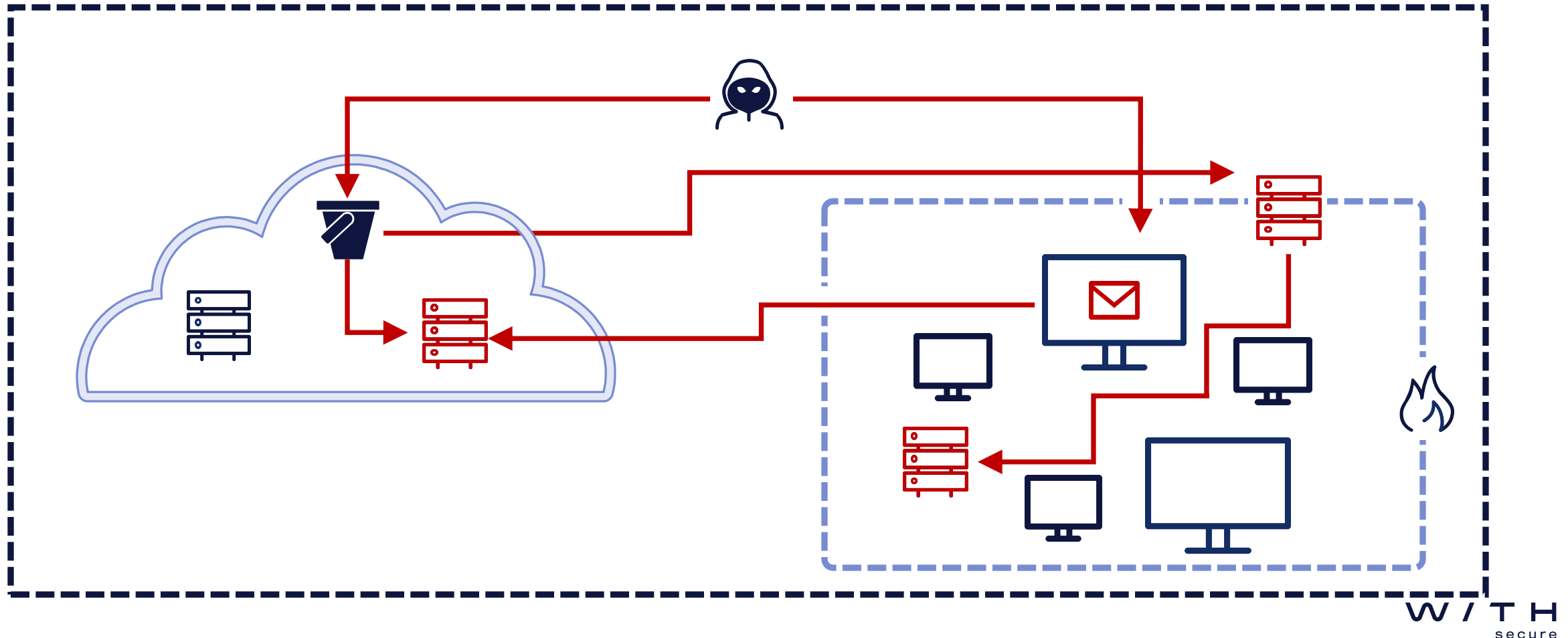
Security Modelling



Enterprise Cloud Adoption



Attackers Target Everything



Real World Breach Scenarios

Breach Dataset

Rami McCarthy's Breach Dataset

- Curated dataset of AWS related security incidents
- <https://github.com/ramimac/aws-customer-security-incidents>

Highlights

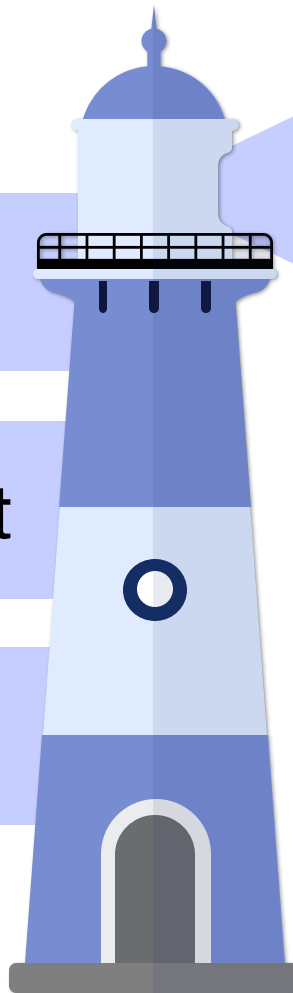
- 45 breaches back to 2014
- 21 incident reports
- Ignores S3 buckets – too many to count!

Inherently Flawed Data

Not all breaches get spotted

Providers hate talking about it

Focus on low hanging fruit



A Note on Cloud Zero Days

Cool but mostly irrelevant

- CloudVulnDB tracking >120 vulns
- One exploited in the wild, no breaches reported
- <https://www.cloudvulndb.org>

Expect this to change

- fwd:cloudsec 2022 keynote from Wiz is a good overview



Open S3 Buckets

The perennial problem

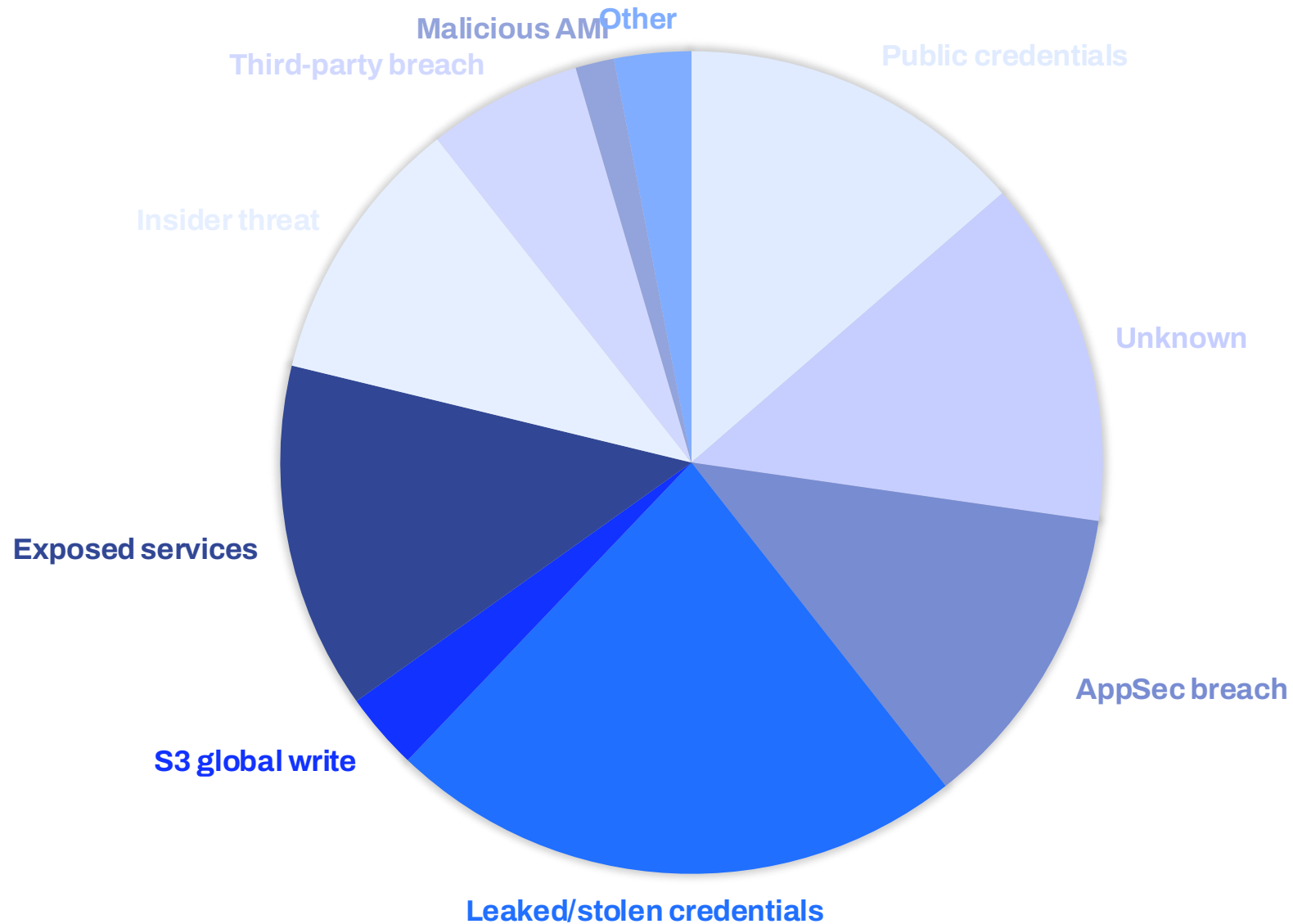
- Biggest source of breaches for years now
- Trivial to find and exploit

Situation is Improving

- AWS providing good options now to prevent
- Enable block public buckets everywhere!

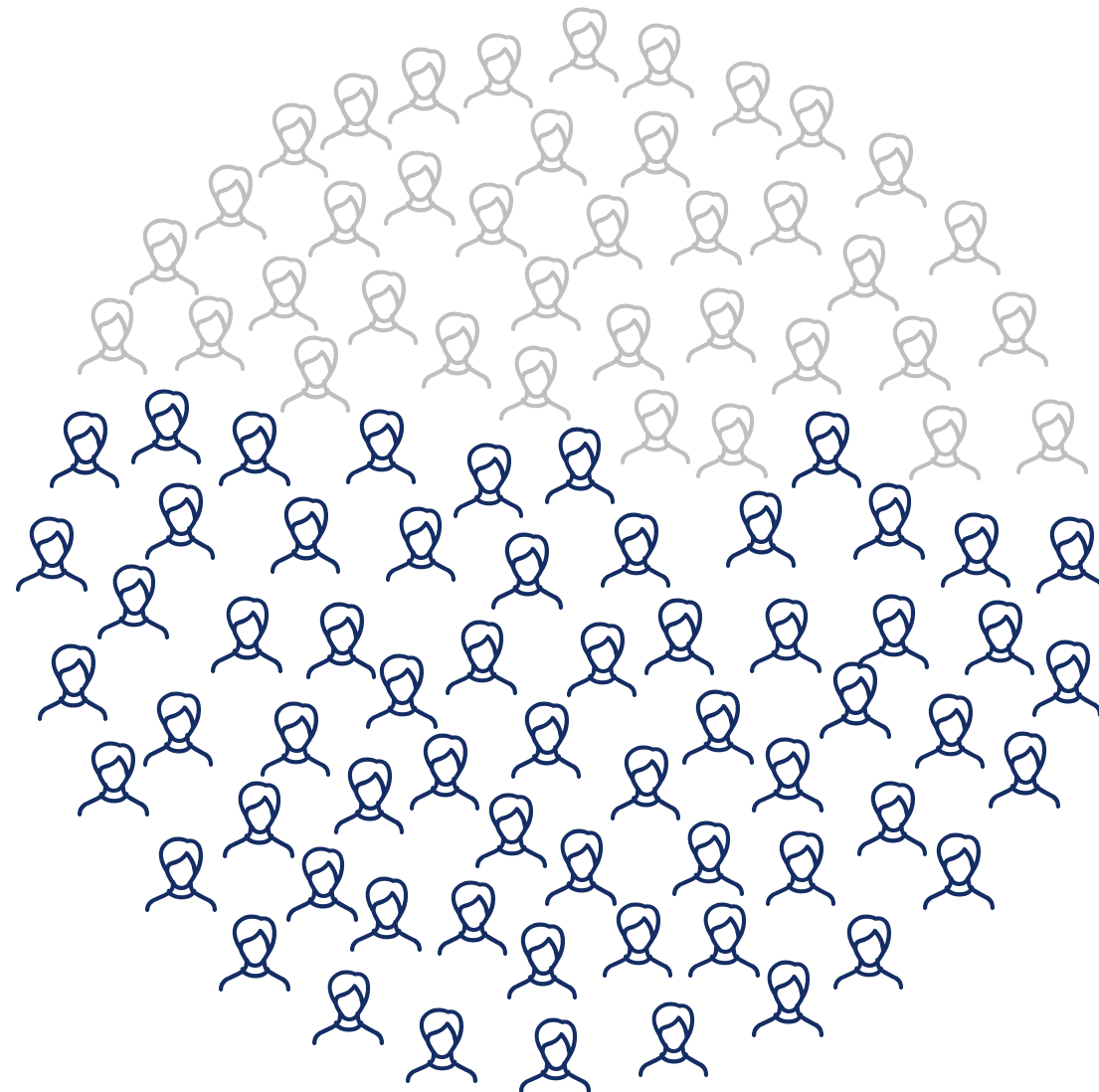


Breach Causes



44%*

Breaches involving IAM users



NUKE IT FROM ORBIT

**ITS THE ONLY WAY TO BE
SURE**

memegenerator.net

Summary

Attackers look for the easiest path

Most attacks are opportunistic

Your org is likely not a priority target

The basics helps stop APTs too

Most get screwed by the basics:

Public S3 buckets

Forgotten accounts

Leaked credentials

Bad leaver handling

IAM Users

You **probably** won't get breached by:

Encryption at rest

Not using the Nitro Enclaves etc

Zero days

AWS Insider threat

Other Interesting Attack Vectors

Cloud Native Management Services

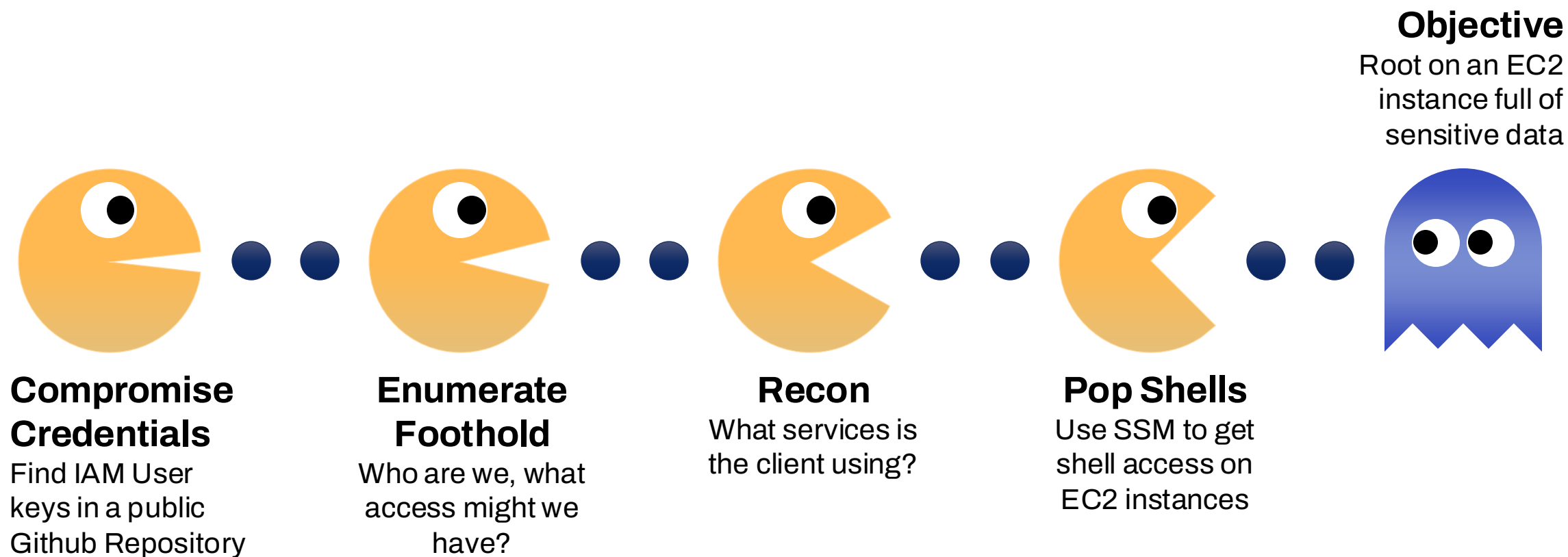
Native SSH/RDP aren't great

- Network level access to manage
- Overhead of separate authentication systems
- Harder to log & audit

SSM & Instance Connect are *mostly* better

- (Usually) easier identity management, fewer networking concerns
- Caveat: It joins two previously separate security domains
- Your IAM configuration needs to be solid!

Cloud-Style Shell Popping!



Cloud Native Phishing

Identity Platforms / SSO

- Okta, Ping, OneLogin, Auth0...
- Single point of access
- Supply chain risk too

Interesting security properties

- MFA, CAPs etc etc
- Often poor session management
- Get the session token, get access to ***everything***

Exploiting Development Workflows

Source Code Management

Everyone uses GitHub or similar to develop and collaborate on their code

CI/CD

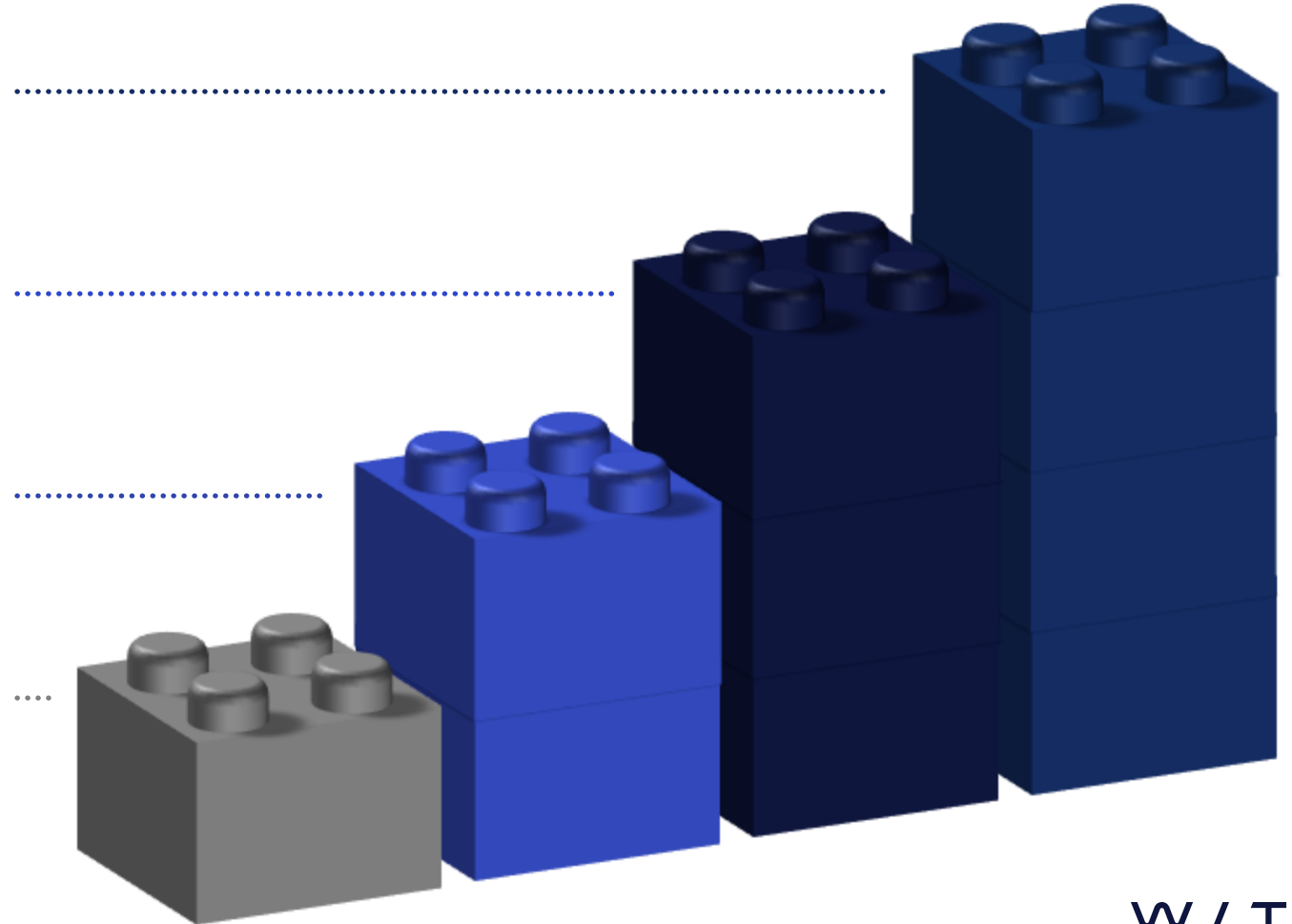
Continuous integration and continuous delivery to automate testing and deployment of cloud workloads

Dev Usability > Security

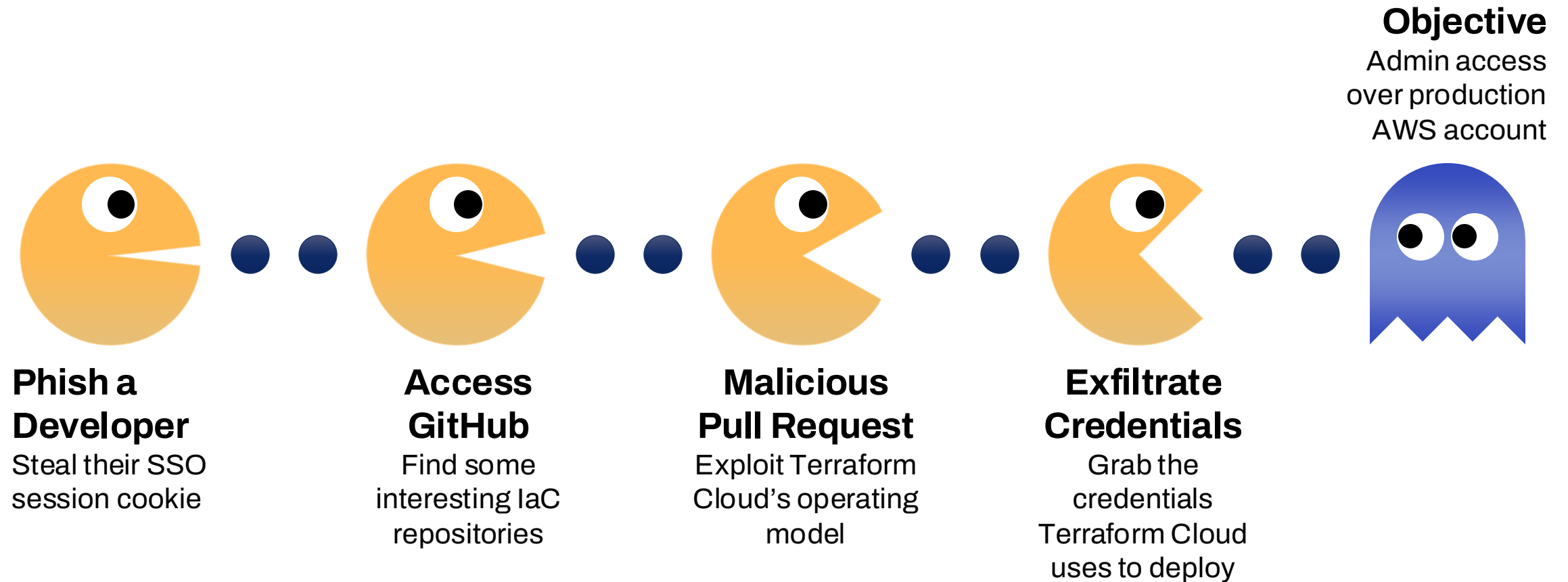
Enabling devs to move at speed often means system architectures and controls are not well hardened

Automatic IaC Deployments

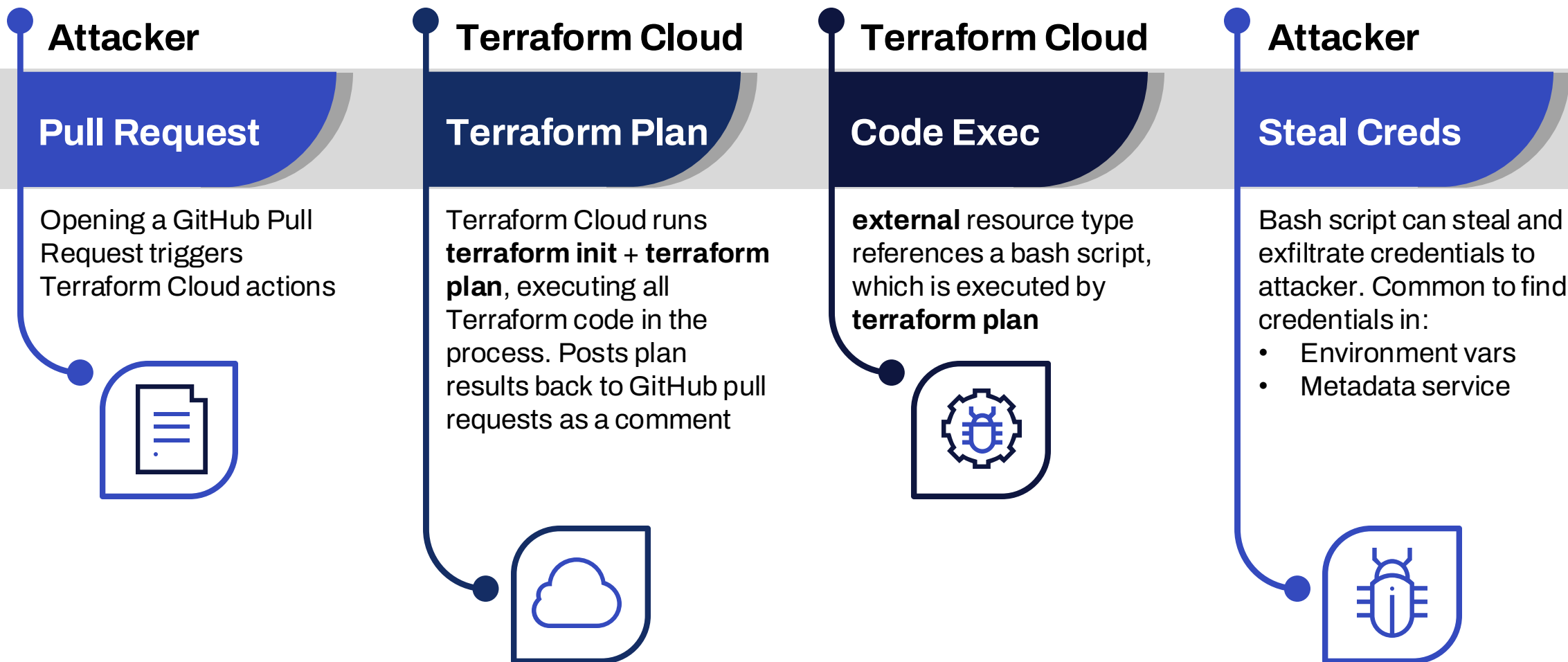
IaC changes often automatically deployed after merging – can we bypass approvals process?



Attack Path 2: DevOoops



Terraform Cloud Exploitation



Pipeline Hardening

01 Code Scan IaC

Analyse IaC for malicious code on pull request before triggering TFC

03 Pipeline Assessments

Treat SCM and CI/CD as crown jewels, threat model and pentest accordingly

02 Four Eyes Checks

Enforce approval on all merges into master

04 Reduce Attack Surface

Standardise tooling, disable unneeded components



Key Security Controls

Strong Identity Controls

Enforce Multi-Factor Authentication (MFA) everywhere

01

Apply principle of not-very-much privilege

02

Eliminate long-lived credentials **(IAM USERS!)**

03

Use provider-backed authentication where possible

04

Automate credential management and rotation

05

Limit Blast Radius

SEPARATE PROJECTS

Use separate accounts/subscriptions/projects for different applications



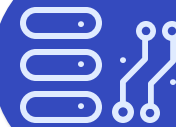
SEGREGATE AT THE NETWORK LEVEL

Enforce strong network boundary controls, avoid VPC peering (especially with third parties)



SEPARATE ENVIRONMENTS

Keep development, QA/test and production environments separated within your cloud's management structure, such as AWS Organisations or Google Organisations



MINIMISE SHARED SERVICE ACCESS

Deploy unique CI/CD pipelines per environment, have monitoring tools reach into the account rather than the accounts writing data out elsewhere



Production Access Control



Secrets Management

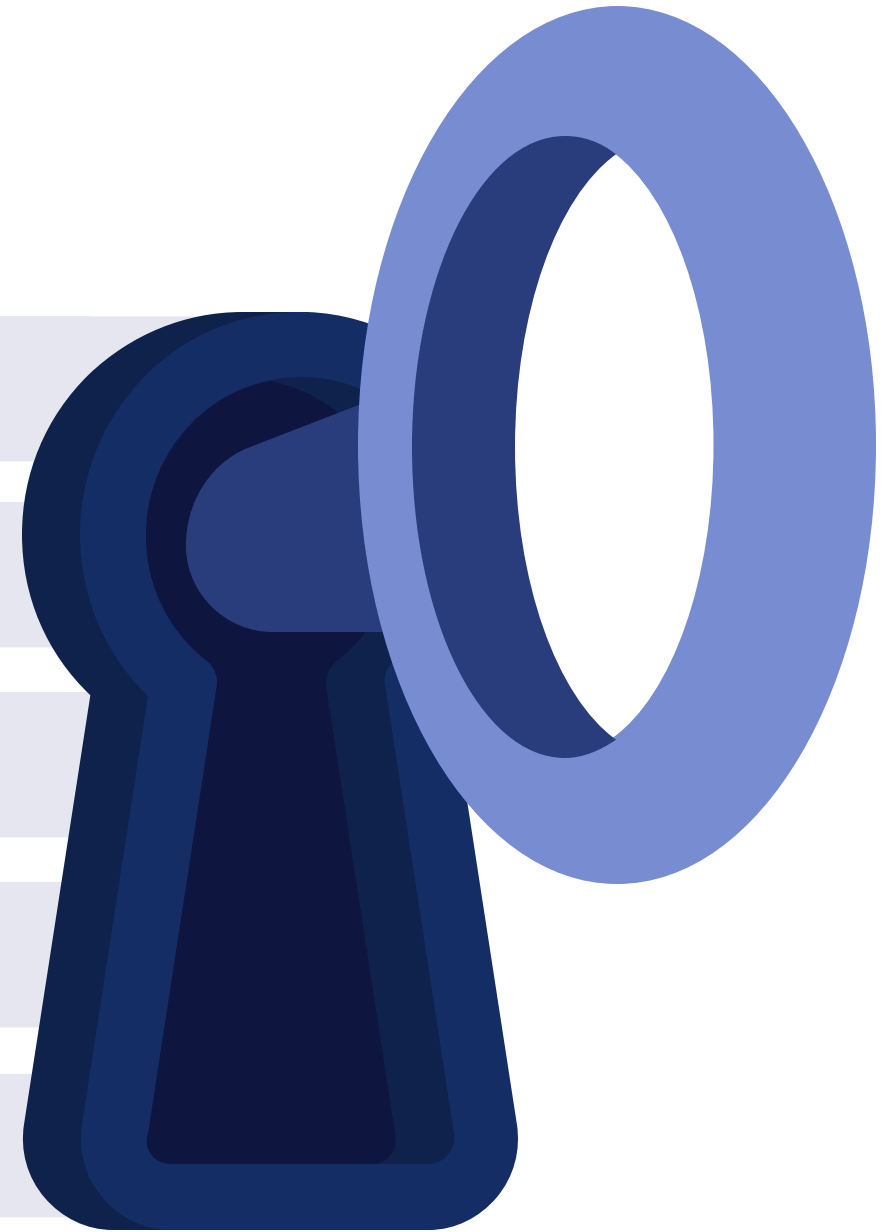
Often the key point of failure

Where do applications store their secrets?

How are credentials shared and rotated?

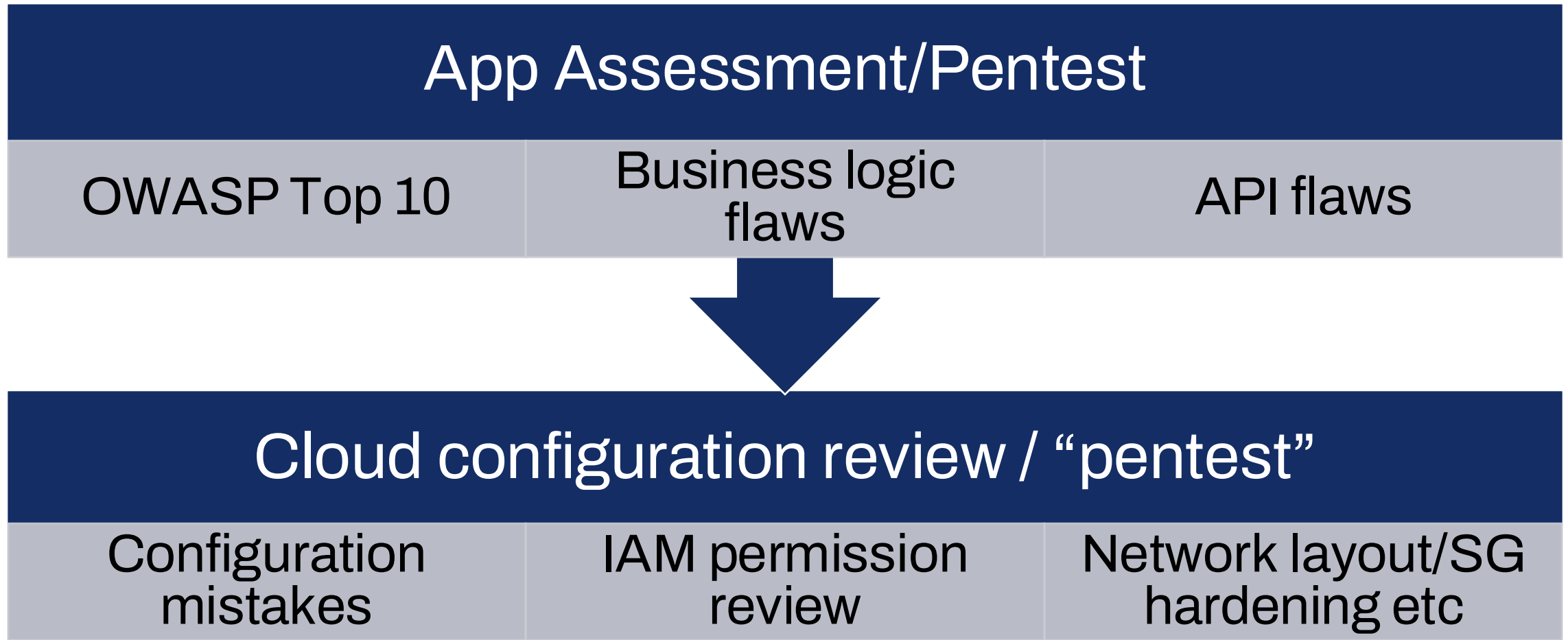
How do you know when secrets are leaked?

Use Secrets Manager / SSM Parameter Store!



Security Testing Done Right

“Penetration Testing” in AWS



“Penetration Testing” Mostly Sucks

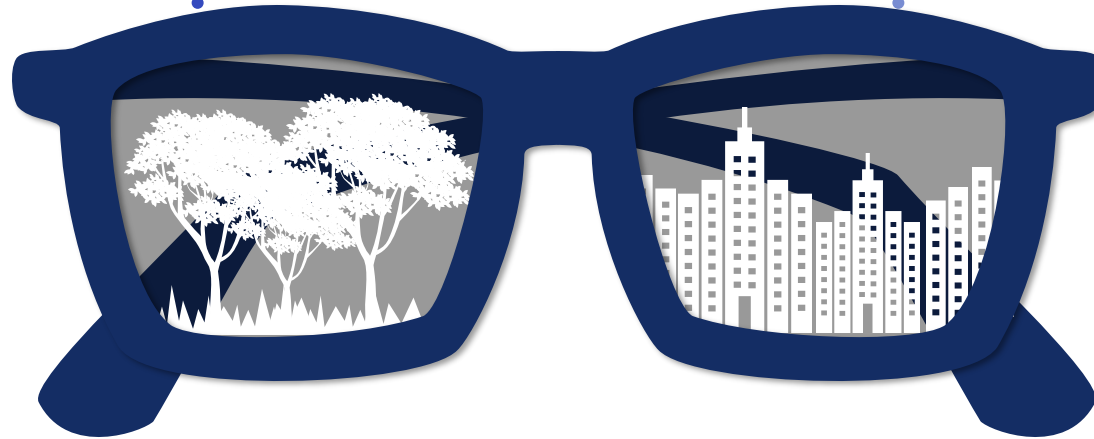
- Driven by audits, not threats
- Cloud engineering moves too fast
- Low return on investment
- Ignores critical supporting systems

What To Do Instead?



Automate

Leverage automation to drive as much security as possible



Assess

Use humans to find the rest, and simulate attackers

Security Automation

02 IaC Scanning

Scan Infrastructure as Code in pipelines

Checkov
TFLint

01 Configuration

Assess resources for configuration issues

Prowler
ScoutSuite



Secrets Scanning 04

Scan repositories for keys, certificates etc

TruffleHog
detect-secrets

IAM 03

Identify IAM misconfigurations

Cloudsplaining
Pmapper
IAMSpy

Human-led reviews



Objective-Driven Assessments

Business targets

- Steal key data/IP
- Move money
- Deploy malicious code to prod

Realistic starting points

- Leaked access keys
- Compromised developer
- Other insider threat
- Application compromise



Don't Buy a Red Team

You likely don't need it

- Adversarial simulation
- All about stealth, validating detection and response
- Depth, not breadth

Red Team = final step

- Confirm/harden attack surface
- Build your attack detection & response capabilities
- Test everything collaboratively
- ... **then** maybe a red team!



Collaborating with Security Partners

If You're Buying Security Testing...

Make it work for you

- Fit their testing and reporting into your workflows
- Push for deep advice and long-term solutions

Find a good partner

- Do they get AWS/Cloud/DevOps?
- Can they show you novel R&D?
- Use engineers to vet providers' technical knowledge

Help Us Help You!

Access

- Give us read access to the AWS accounts
- If you're using IaC, show us that too

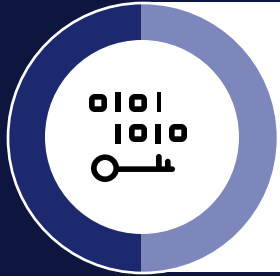
Work with us

- Help us understand what you've built
- Show us problems, help us design solutions
- Stay engaged and **communicate** with testers

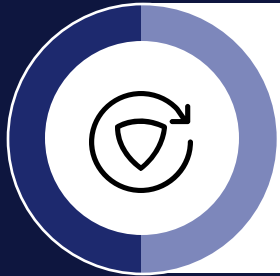
Conclusions



Security of the cloud extends to include a lot of external factors



Focus on IAM (especially users!), secrets management and CI/CD



Leverage automation and be smart about how you use humans

“
If you want to go fast, go alone.
If you want to go far, go together.

-- African Proverb*
”

* <https://medium.com/@johnlatwc/the-githubification-of-infosec-afbdbfaad1d1>

Thanks for listening!

Twitter: @nojonesuk

Blog: www.nojones.net

Community Builders Slack: Nick Jones

W / T H
secure

W / T H[®]
secure