AWS CLOUD SECURITY COMMUNITY DAY

Securing AWS Estates at Scale

Nick Jones, WithSecure

Agenda

- Common Misconceptions
- Real World Breach Scenarios
- What We See
- Key Security Controls

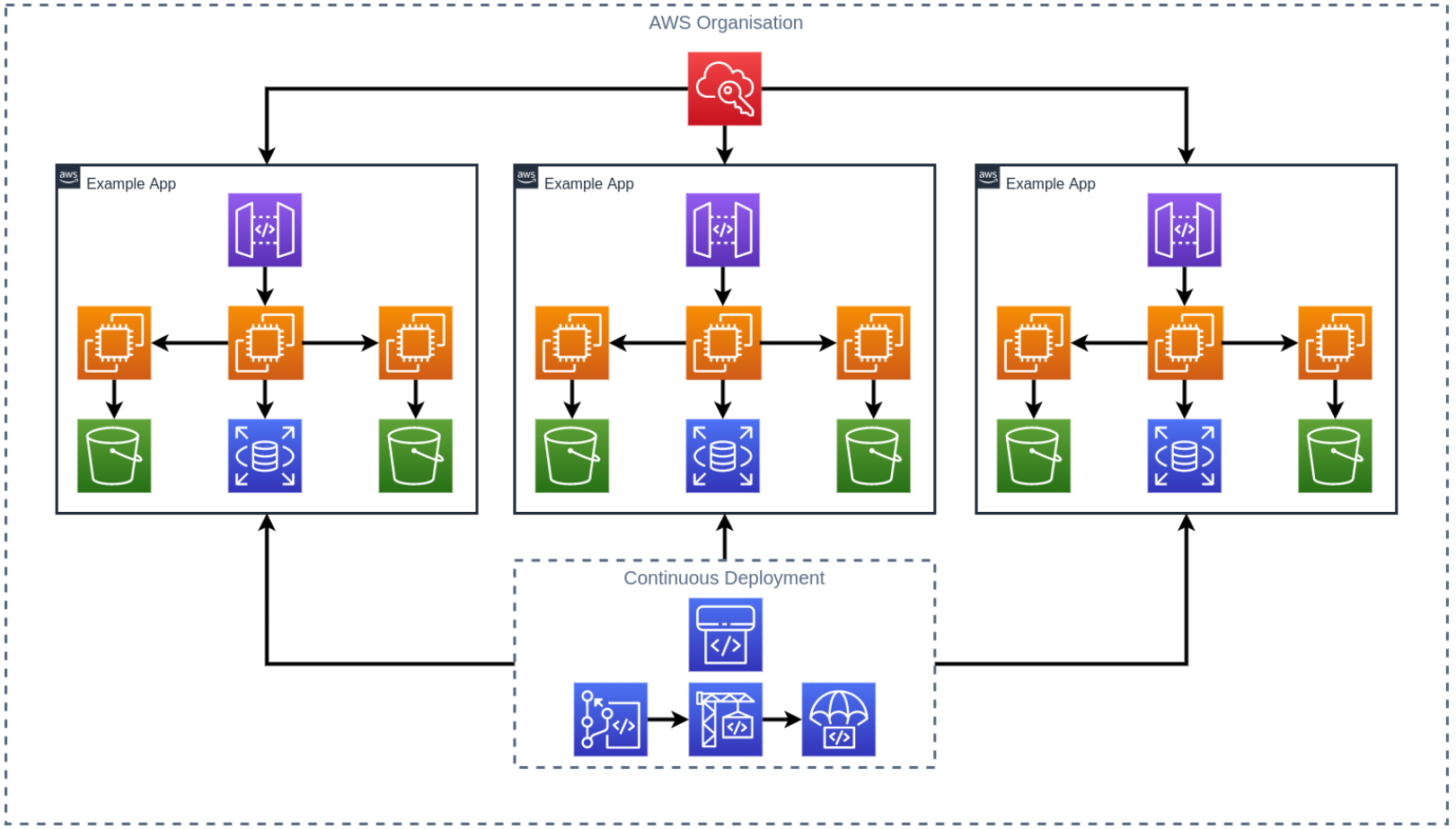
Who Am I?

Nick Jones – @nojonesuk

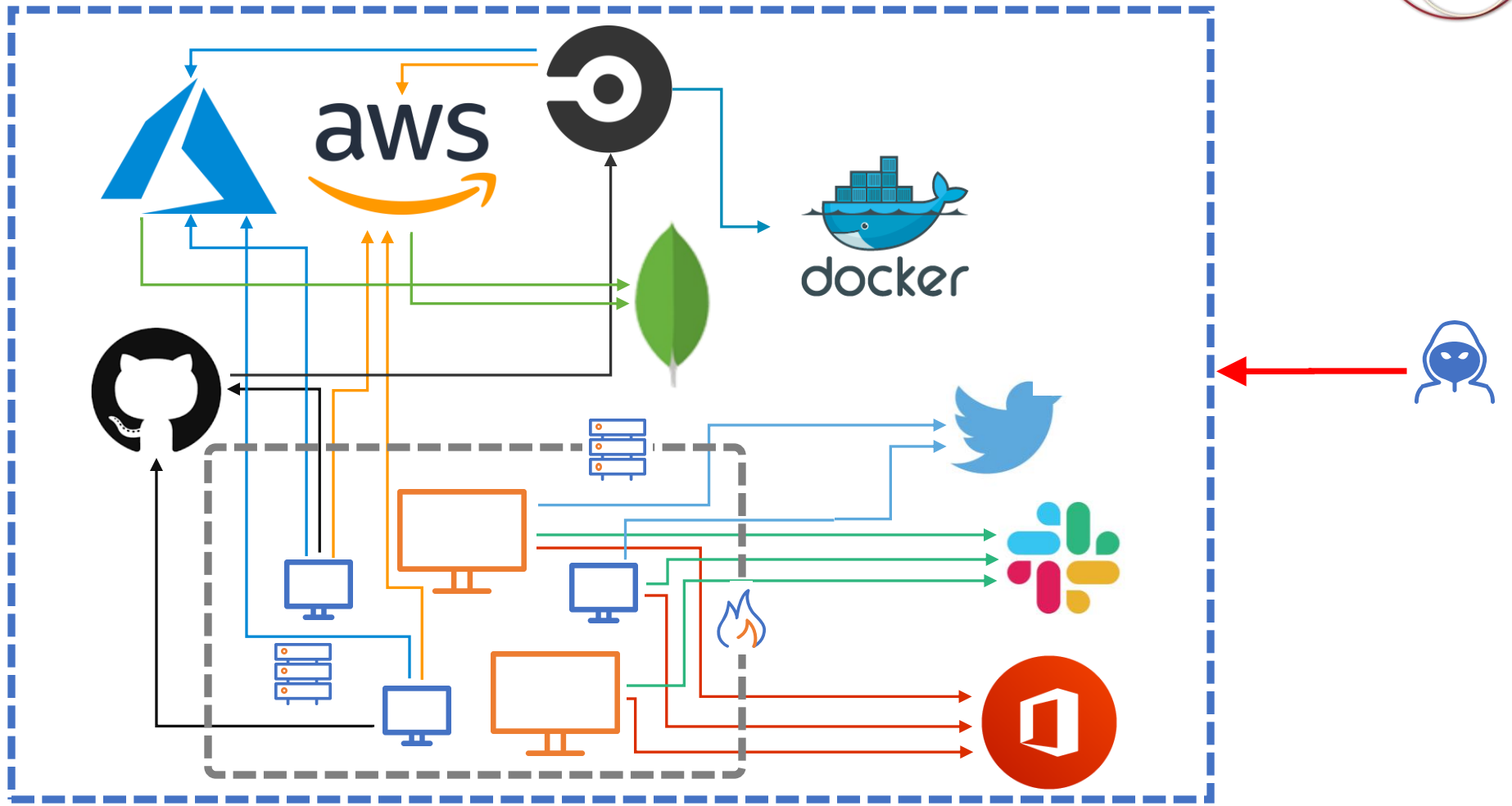
- Principal Consultant
- CloudSec Lead @ WithSecure
- AWS Community Builder
- Previously presented at:
  - fwd:cloudsec
  - RSA Conference
  - Blue Team Con
  - +++

Common Misconceptions

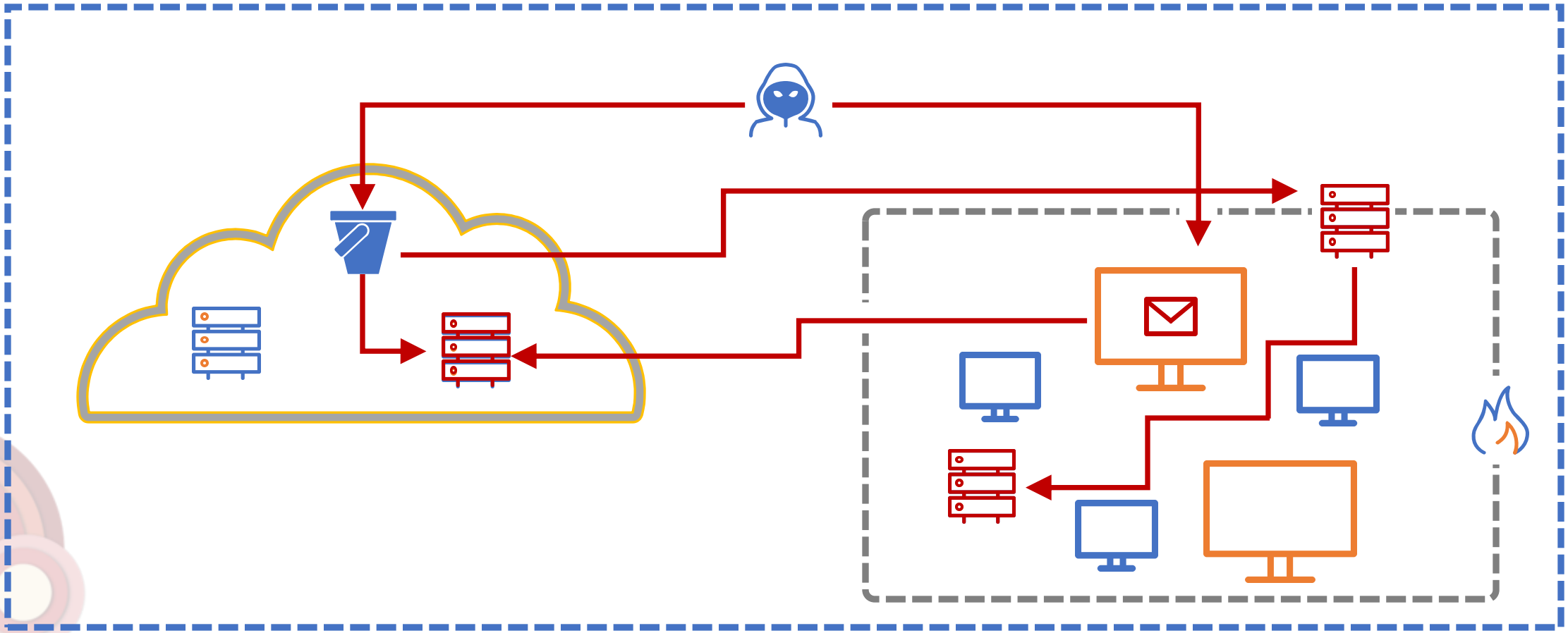
The scope of most people's thinking



The Reality



Attackers don’t just attack the cloud



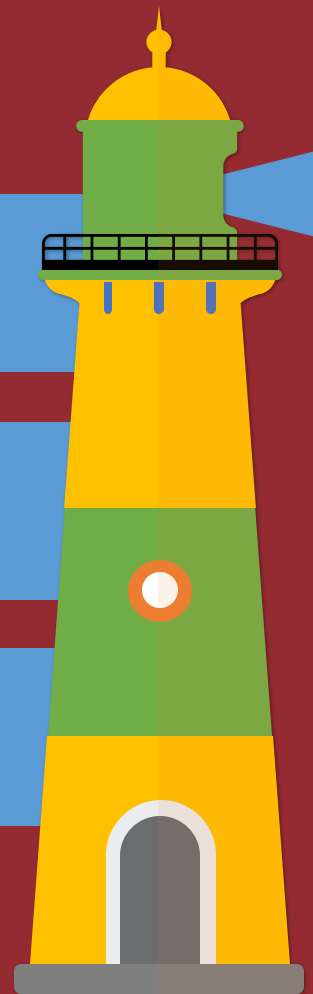
Common Myths Dispelled

Attackers look for path of least resistance	Most people get screwed by the basics:	The following **probably** won't be how you get breached:
• Most attacks are opportunistic • The basics helps stop APTs	• **Public S3 buckets** • Forgotten AWS accounts • Leaked credentials • Admin rights granted to stupid things	• Insufficient/misconfigured encryption at rest • Not using the Nitro Enclaves/AWS ShinyNewSecurityService • Zero days • Insider threat @ AWS

Real World Breach Scenarios

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

- >120 vulns, 1 exploited ITW, no breaches reported
- https://www.cloudvulndb.org

Expect this to change in time

- Israel leading the charge - Wiz, LightSpin, Orca
- fwd:cloudsec 2022 keynote from Wiz is a good overview

Open S3 Buckets

The perennial problem

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

Situation is Improving

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

Photo from https://www.flickr.com/photos/electronicfrontierfoundation/50617066023

What Else are Attackers Doing?

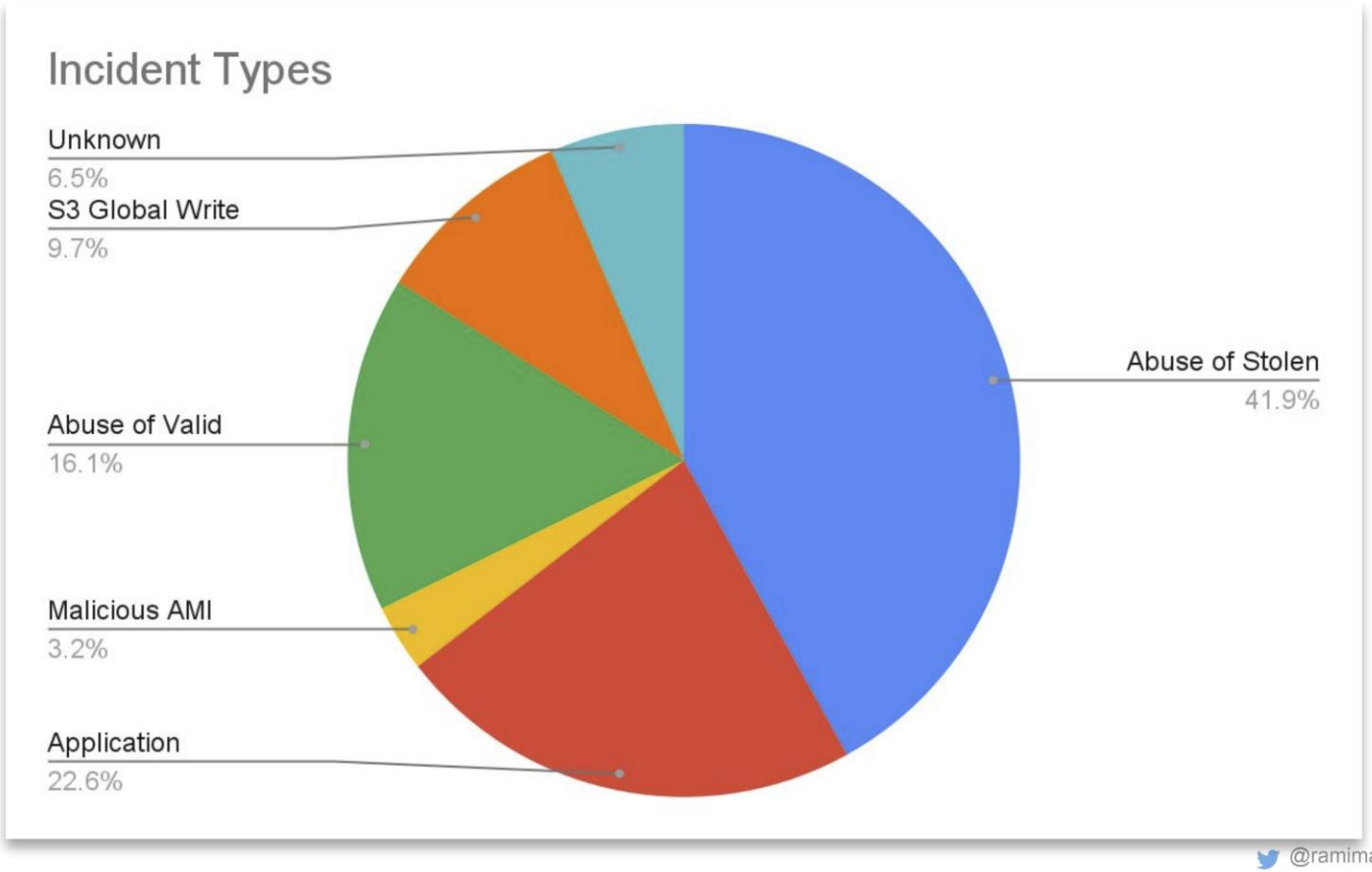


Image from https://speakerdeck.com/ramimac/learning-from-aws-customer-security-incidents-2022?slide=20

What We See

Credential Theft

Most common cloud breach scenario

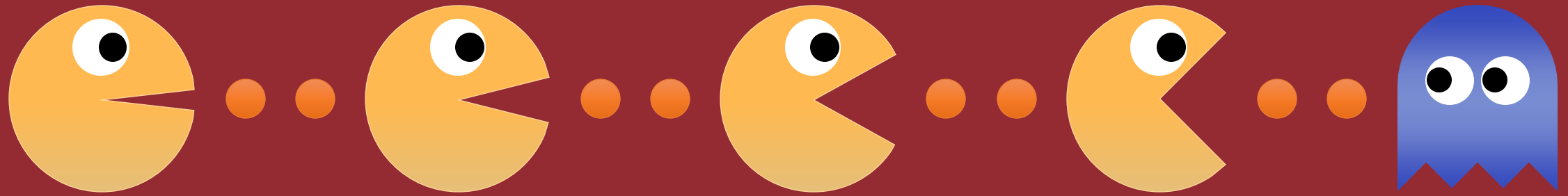
- Verizon DBIRs say ~70% of cloud breaches

Some fun options:

- Credentials in public repositories
- Application Exploitation
- Phishing!

Attack Path 1: Cloud-Style Shell Popping

Objective
Root an EC2 instance full of data



Compromise Credentials
Access Keys in GitHub repository

Enumerate Foothold
Who are we, what access might we have?

Recon
What services is the target using?

Pop Shells
Use our access to get shells on EC2 instances

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 ***everything***

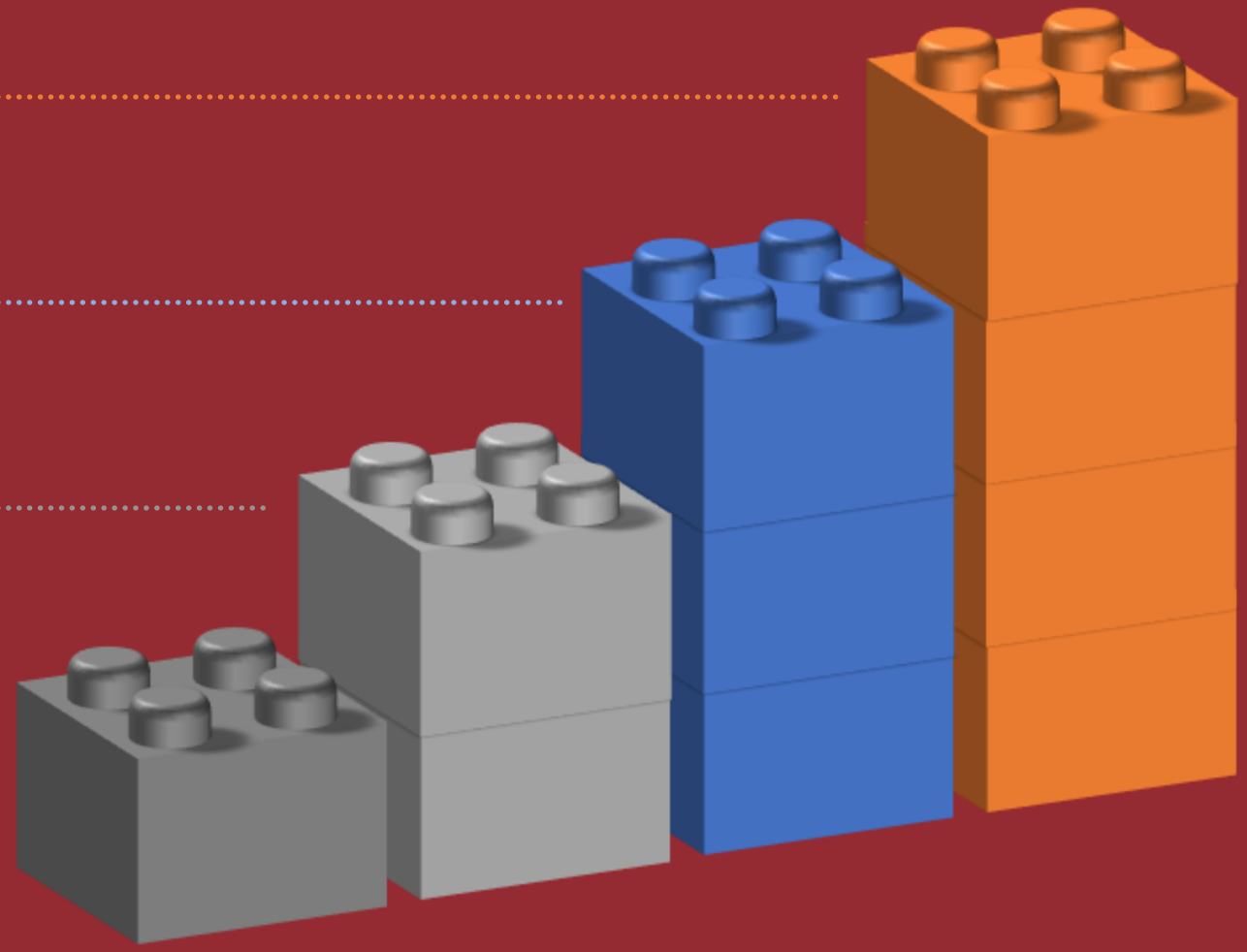
Exploiting Development Workflows

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

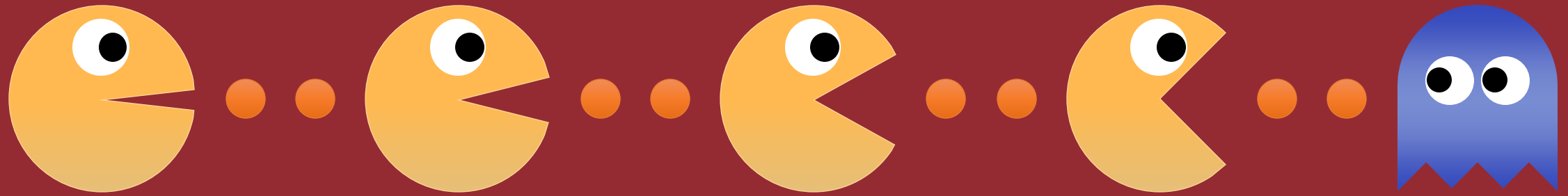
Dev Usability > Security
Enabling devs to move at speed often means system architectures and controls are not well hardened

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



Attack Path 2: DevOooops

Objective
Admin access over production



Phish a Developer
Steal their SSO session cookie

Access GitHub
Find some interesting IaC repositories

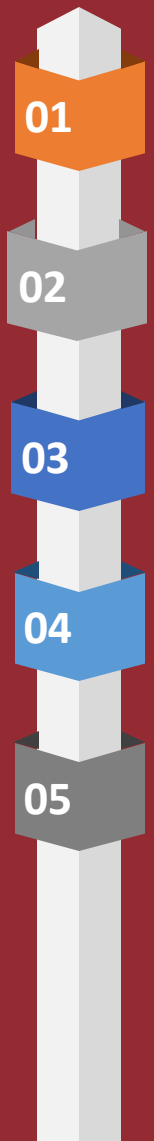
Malicious Pull Request
Exploit Terraform Cloud's operating model

Exfiltrate Credentials
Grab the credentials Terraform Cloud uses to deploy

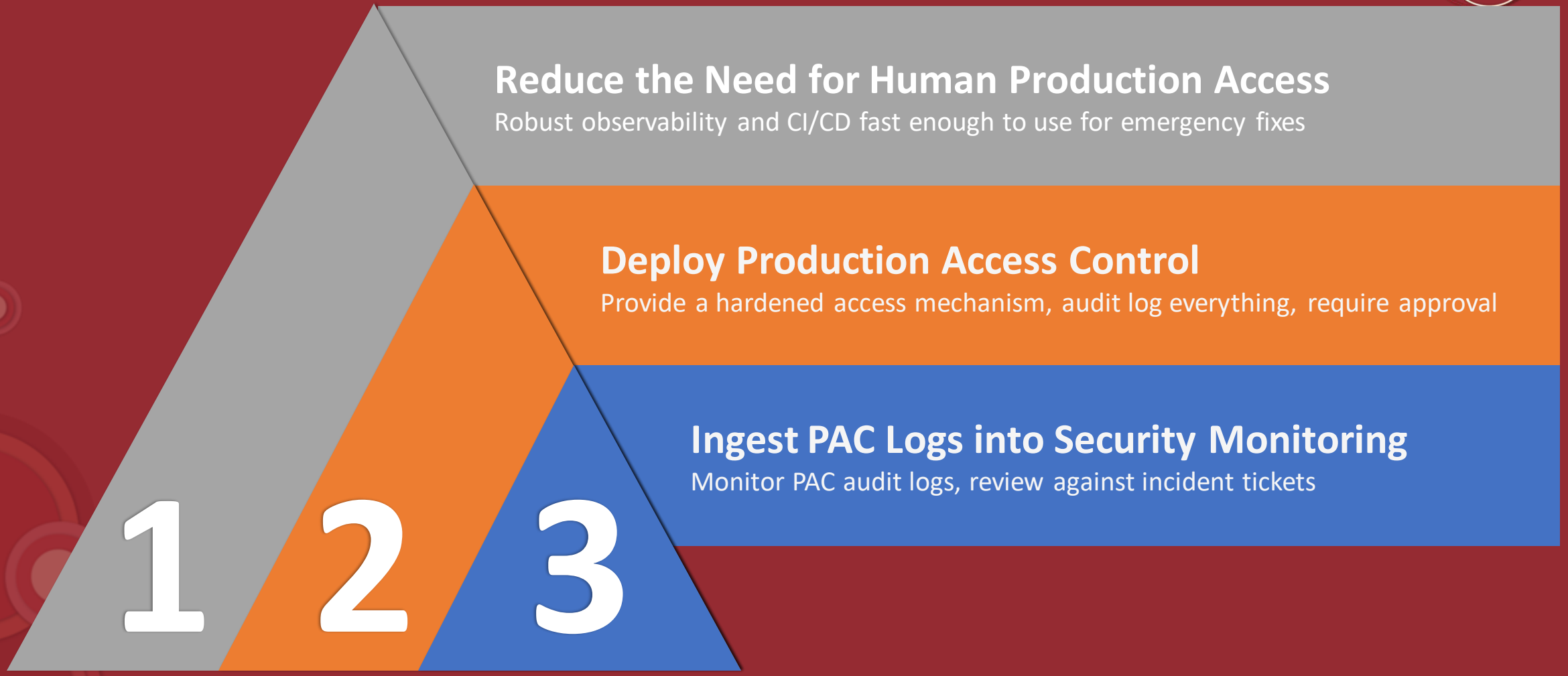
Key Security Controls

Strong Identity Controls

- Multi-Factor Authentication (MFA) everywhere
- Principle of not-very-much privilege
- Eliminate long-lived credentials (**IAM USERS!**)
- Use IAM Roles where possible
- Automate credential management and rotation



Avoid People In Production



Reduce the Need for Human Production Access

Robust observability and CI/CD fast enough to use for emergency fixes

Deploy Production Access Control

Provide a hardened access mechanism, audit log everything, require approval

Ingest PAC Logs into Security Monitoring

Monitor PAC audit logs, review against incident tickets

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?

Secrets management in AWS

Use AWS services to do the heavy lifting

- Secrets Manager
- Systems Manager Parameter Store
- Hashicorp Vault or similar, if used with IAM authentication

Common places to find hardcoded secrets

- **EC2 USER DATA!**
- CloudFormation templates
- App source code
- Environment variables in Lambda configurations
- S3 buckets

Limit blast radius

Separate Projects

Use separate AWS accounts within an Organization

Segregate at the Network Level

Enforce strong network boundary controls

Avoid VPC peering (especially with third parties)

Don’t expose routes between environments

Separate Environments

Keep dev/QA/prod in separate accounts

Run security tools in their own accounts

Log centrally to a logging account

Minimise Shared Service Access

Unique CI/CD pipelines per environment

Pull from central rather than push from environments

Decentralised Security skills

Too much tech for any one person

- Devolve security skills into other teams
- Expect to build a multidisciplinary team

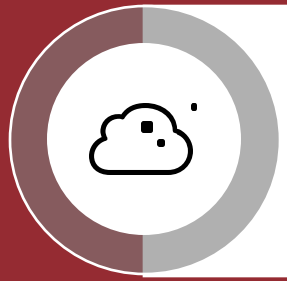
Engineers are the SMEs – work with them!

- Security teams should include automation specialists
- Ex-cloud/devops engineers ideal here

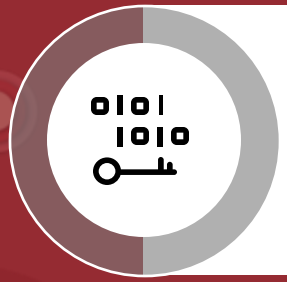
Expect to invest heavily

- Cloud security people are scarce and expensive
- Good tools do not come cheap

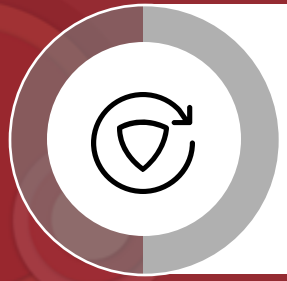
Conclusions



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



Focus on IAM, secrets management, environment segregation and CI/CD



Leverage automation and empower engineers to scale company-wide