Cloud Security Essentials

...

About Me

Dustin Whited
Sr. Cloud Architect @ Dragos
Ex. ScaleSec, Allstate, ISU

Agenda

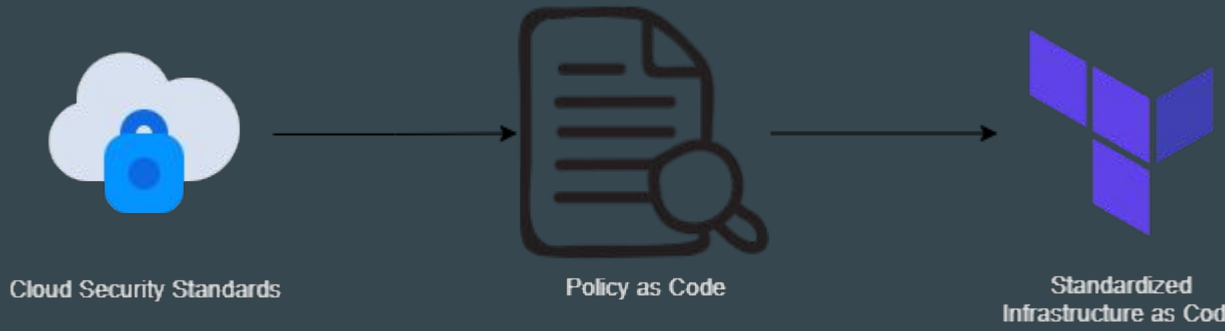
- Standards and the Importance of Infrastructure as Code
- Root
- Get Organizations
- IAM
- Logging

Create Cloud Security Standards

Be opinionated.

AWS.IAM.1 : IAM policies MUST NOT specify “Service:*”

AWS.S3.1 : S3 buckets MUST have private ACLs



Security standards form the baseline policies to validate infrastructure is compliant

Example Terraform Sentinel Policy - All Buckets MUST be private

Policy (Sentinel)

```
nonPrivateS3Buckets = plan.filter_attribute_is_not_value(
  S3Buckets, "acl", "private", true
)
```

Terraform resource

```
resource "aws_s3_bucket" "my_special_bucket" {
  bucket = "s3-website-test.hashicorp.com"
  acl = "public-read"
}
```

Warn

Error

Secure the Root Account

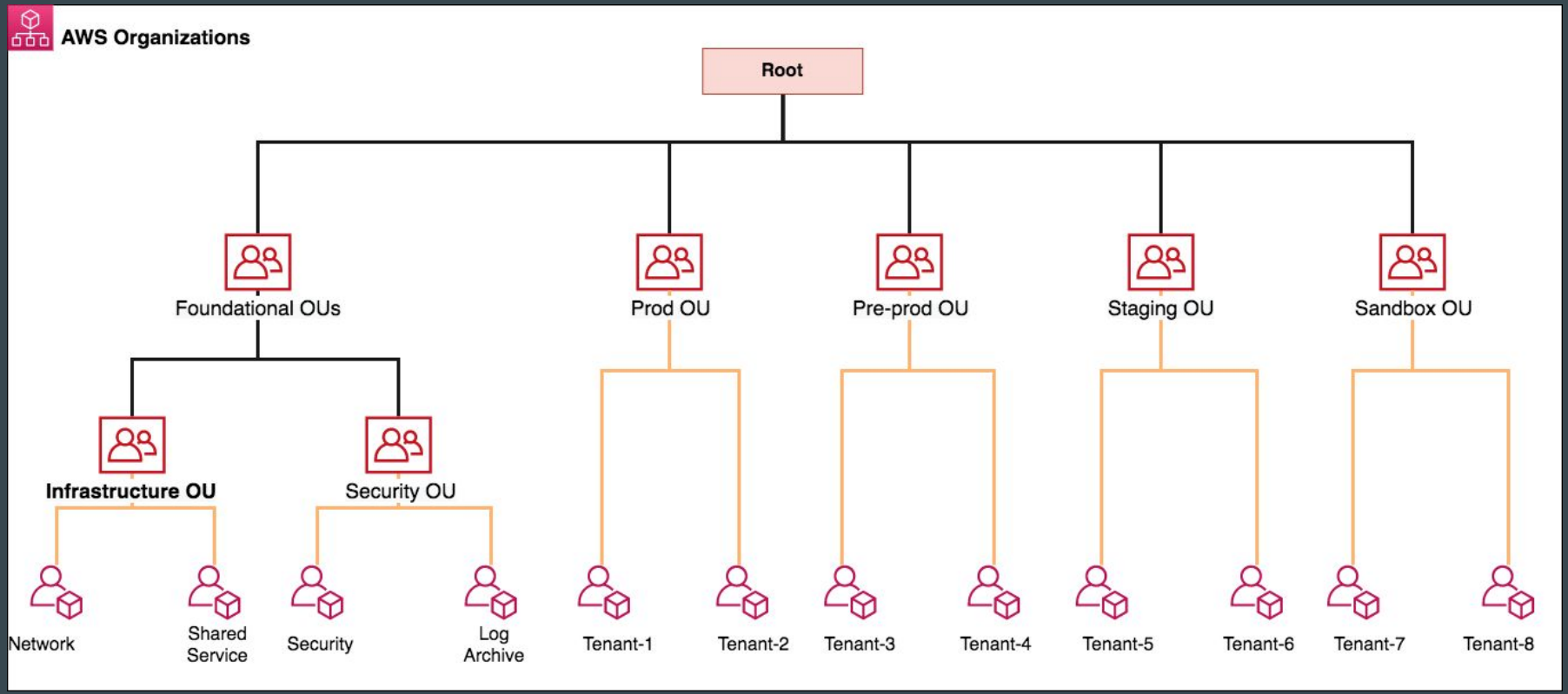
- Turn on MFA
- Keep hard tokens in a secure location
- Use a distribution list for the email
- Don’t use it unnecessarily
- Block the root account with a Service Control Policy (SCP)

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "DenyRootUser",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "StringLike": { "aws:PrincipalArn":
"arn:aws:iam::*:root" }
    }
  }
 }
```

Enable Organizations

- Create a new account to designate as management account
- Delegate security services to sub accounts
- Create Service Control Policies (SCPs) for preventative guardrails
  - Compliance-specific service SCPs: https://github.com/salesforce/aws-allowlister
  - Examples in Terraform: https://github.com/ScaleSec/terraform_aws_scp
- Use Org-Formation or AWS Control Tower to orchestrate
  - Org-Formation = CLI and Infrastructure as Code friendly
  - Control Tower = Console friendly, no APIs, can be brittle

Example Organizational Structure



https://aws.amazon.com/blogs/security/use-aws-firewall-manager-to-deploy-protection-at-scale-in-aws-organizations/

IAM

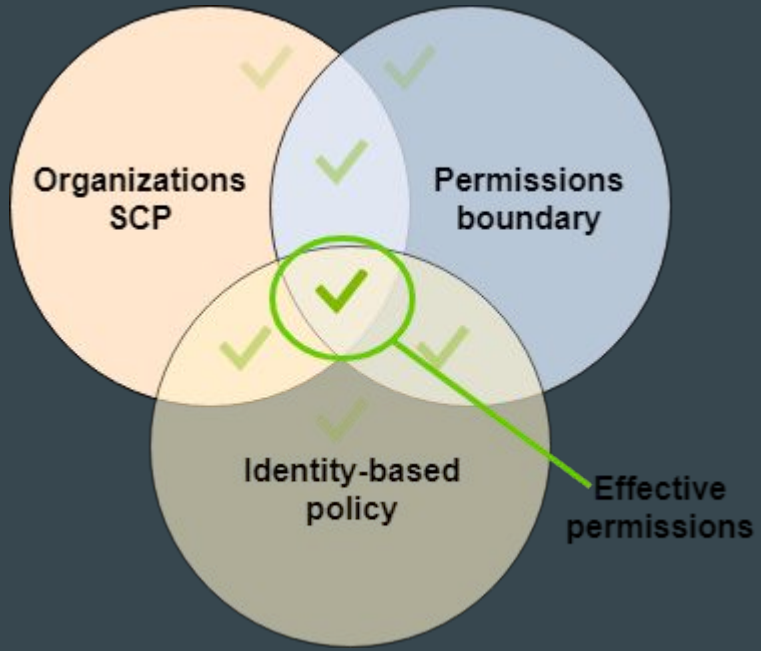
- Centralization of IAM negatively impacts developer velocity
- Results in slightly less permissive policies (usually)
- Not worth it unless there is a dedicated IAM product team

Instead, implement guardrails:

- Service Control Policies to restrict services, regions, and specific actions
- Boundary Policies to limit permissions
- AWS Account to limit blast radius - One workload per account

SCP and Permissions Boundary

- Developer allows role s3:*
- SCP denies s3:ListAllMyBuckets
- Permissions Boundary allows s3:List*



Effective Permissions:

```
s3:ListAccessPoints
s3:ListAccessPointsForObjectLambda
s3:ListBucket
s3:ListBucketMultipartUploads
s3:ListBucketVersions
s3:ListJobs
s3:ListMultiRegionAccessPoints
s3:ListMultipartUploadParts
s3:ListStorageLensConfigurations
```

IAM Users

Delete them. With prejudice.

Why...

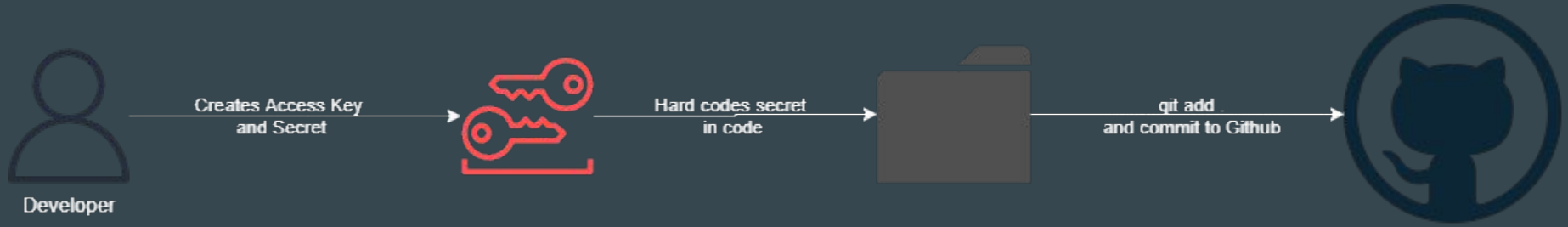
Access keys get left on laptops, hardcoded in application code, and checked into VCS.

There are better ways!

- Role assumption
- OIDC - BitBucket and Github both support OIDC now
- Temporary credentials for human access (AWS SSO)

Github

"85% of [leaks on GitHub] occur on developers' personal repositories." [1]



Scan for secrets pre-commit on the repos you do control
https://github.com/awslabs/git-secrets
https://github.com/OWASP/SEDATED

Github is a great opportunity for a canary!

1. GitGuardian State of Secret Sprawl 2021 https://blog.gitguardian.com/state-of-secrets-sprawl-2021

Canaries

Free service provided by Thinkst http://canarytokens.org/generate

Generates AWS Access Key and Secret ID - sends an email or webhook if they are used.

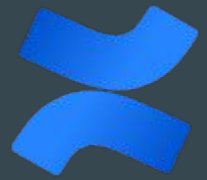
```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Think outside AWS too...

Executive's Laptop

Github

Google Drive



Confluence

Logging

- Create a Log Archive account for dedicated log retention
- Send logs:
  - Enable Organizational CloudTrail(s)
  - GuardDuty Findings
  - Route53 / DNS Query Logs
  - VPC Flow Logs
  - AWS Config
  - WAF logs w/ Firewall Manager