Agenda

- 1st Half
  - Introductions
  - What makes AWS so freakin’ special?
    - Very high level overview of services
  - Top 10 risk mitigations/strategies
- Break
- 2nd Half
  - API’s fo’ life!
    - How does the AWS API work?
  - Build 1: S3 bucket policy monitor
  - Build 2: CloudTrail pipelining

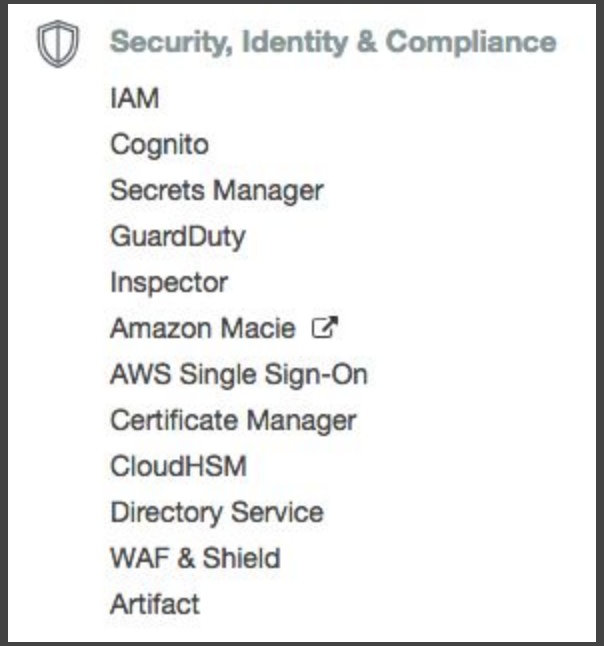
Training Part 1

AWS Security Fundamentals

Let's clear one thing up ...

- Yes, you must trust Amazon
- You make the same type of trust decisions anyway
  - Who owns your datacenter?
  - Who built your hardware?
  - Who owns the connectivity?
- With AWS you are centralizing trust in Amazon, and let's be honest ...
  - They run a datacenter better than I do
  - They build and manage hardware better than I do
  - They can afford better connectivity than me
  - Their economy of scale has benefits across competencies

AWS's Security Services


Security, Identity & Compliance
IAM
Cognito
Secrets Manager
GuardDuty
Inspector
Amazon Macie
AWS Single Sign-On
Certificate Manager
CloudHSM
Directory Service
WAF & Shield
Artifact

Or ... ?

Security, Identity & Compliance

AWS Identity and Access Management (IAM)

Amazon Cloud Directory

Amazon Cognito

Amazon GuardDuty

Amazon Inspector

Amazon Macie

AWS Certificate Manager

AWS CloudHSM

AWS Directory Service

AWS Firewall Manager

AWS Key Management Service

AWS Organizations

AWS Secrets Manager

AWS Single Sign-On

AWS Shield

AWS WAF

AWS Artifact

No, but seriously ...

AWS's Security Services

- IAM - Identity Access Management
  - Relatively complex policy documents... nouns, verbs, conditionals
  - Not necessarily uniformly applied to all services
- VPC - Virtual Private Cloud
  - Network space segmentation
  - You must choose a VPC for most services which require network access
- SecurityGroups
  - Hypervisor level firewalls
  - Inbound / Outbound
  - Applied to most services which require network access
- KMS - Key Management Service
  - Store secrets
- And logging ... we'll cover that later!

Some fundamentals

- Add 2fa to the root account and then never use it again
  - Very few things require the root account
- Turn on (all) CloudTrail and escrow to a bucket in a secure account

Back to the Top 10

1. Insecure use of developer credentials

1. Insecure use of developer credentials

- Minimize the use of IAM users
  - Their credentials are permanent
  - Use 2fa for “humans” if you have to; especially for dangerous operations
- SAML federation for console access
  - Federation links SAML groups to a “role”
- EC2 roles
  - Credentials available via the metadata service
  - Rotate automatically every few hours
- Credential Issuance Systems
  - Hologram - https://github.com/AdRoll/hologram
  - AWS Okta - https://github.com/segmentio/aws-okta

2. Publicly accessible S3 buckets

2. Publicly accessible S3 buckets

- Nothing replaces good hygiene
  - Consider restricting the ability to manage bucket permissions
- Object ACL's are usually a bad idea
  - Difficult to audit
  - Interact with bucket permissions in non-intuitive ways
- Try and group similar data in buckets
- When sharing invest the time to leverage IAM rights correctly
- Auditing tools
  - AWS Trusted Advisor
  - AWS Inspector
  - Security Monkey

3. Improper use of default configurations

3. Improper use of default configurations

- Understand any managed policies before using them
  - Managed policies change
  - Services change
  - ... maybe you shouldn’t use them?
- Avoid “*” permissions for the same reason
- Avoid the “not action”
  - Consider explicit deny statements

4. Access controls do not follow principles of least privilege

4. Access controls do not follow principles of least privilege

- Again ... resist the “*”
- Restrict who can make change to IAM policies
  - Put/Update Policy
  - Versioning
  - Pass Role
- Template your permissions
  - AWS CloudFormation
  - Terraform
  - Use collaboration tools (GIT, etc.)
- “Tuning”
  - Look at your logs and revoke access! https://github.com/Netflix/repokid

5. Misconfigured network constructs

5. Misconfigured network constructs

- VPC SecurityGroups allow ingress and egress filtering
  - Starting with egress filtering is easier than tacking it on (default is unfiltered outbound)
- IP space planning is difficult
  - So try to avoid it
  - Invest the time to have a plan
- Rely on SecurityGroups as network “identity”
  - SecurityGroup per application
  - VPC subnet / AZ
  - Internal and External subnets
- Consider Roles/Responsibilities for SecurityGroup configuration as well as naming and tagging schemes
- Consider restricting access to CIDR range based rules

6. Lack of appropriate logging and monitoring

6. Lack of appropriate logging and monitoring

- CloudTrail
  - ELK is pretty good
  - What about data read/write events?
- S3 Bucket Access
- Guard Duty
- Macie
- Flowlogs
- Pseudo-supported services
  - RDS
  - DynamoDB
  - etc.
- IP address assignment tracking can be VERY hard

7. Lack of inventory management

7. Lack of inventory management

- Cloud makes this easy(ish)!
- Ownership
  - Immutable
    - ASG / IAM Role / SecurityGroup / ImageID
  - Mutable
    - Use tags with orchestration
    - Leverage your discovery system
- Garbage collection
  - FlowLogs can help you understand network usage ... but might be overkill
  - Creation date / image version
  - Beware of outliers (things created via console)
- “Billing” can be a powerful motivator

8. Domain hijacking

8. Domain hijacking

- Some AWS resources have DNS trust relationships
  - S3
  - Elastic Beanstalk
  - Cloudfront
- Audit Tools
  - AWS Investigator
  - Cloud Inquisitor - https://github.com/RiotGames/cloud-inquisitor

9. Lack of a disaster recovery plan

9. Lack of a disaster recovery plan

- Cloud services don’t solve disaster recovery for you
  - Consider region outages
  - Consider total account compromise
- You should be able to recreate your account without existing AWS resources
- Use templates and orchestration
- Some services are harder than others (Dynamo, etc)
- Chaos engineering
  - Ruthlessly enforce service resiliency
- Running disaster exercises is worth it

10. Manual account configuration

10. Manual account configuration

- You aren’t “clouding” if you are using the console
- Orchestration
  - Spinnaker (Netflix)
  - Terraform (Hashicorp)
  - There are others ... AWS CloudFormation
- Track and minimize manually created infrastructure
- Protect the orchestration account / credentials
- Consider CI/CD concepts for deployment
  - Code review / approval
  - Secure Development Lifecycle becomes relevant again!
- Orchestration logs are VERY valuable

Summing things up ...

- Common themes
- Whats next?
  - Training Part 2
  - I hope you like APIs!

Thank you!
Questions?