Pragmatic Cloud Security: The Future is Now

Mike Rothman
@securityincite
About Me

• 25+ year technology professional with the road rash to prove it
• Started as a mainframe programmer
• Got lucky and played around with early LANs
• Got even luckier and drew the short straw with network security
• Ended up looking at cloud security before cloud was a thing
As you scale into the cloud, you must automate security and operations. Some cloud first organizations like Netflix can afford to invest in building automation solutions to achieve operational excellence, but for most organizations there is limited time and budget to invest in automation.

DisruptOps empowers all cloud users to achieve cloud security and operational excellence.

- Independent information security research boutique.
- Pragmatism is religion for us.
- We are security guys - that’s all we do.
- And we know a little bit about the cloud…
  - We have been teaching cloud security for 8 years
  - We wrote the CSA 4.0 guidance
And it’s called the Cloud.

- Migrating to the Cloud isn’t an option, but you can choose your path.
It is not necessary to change. Survival is not mandatory.

(W. Edwards Deming)
“Cloud Security starts with Architecture, and ends with Automation”

– Rich Mogull
Cloud is Fundamentally Different

Abstraction

Automation
Automation is Inherent

The NIST Model (courtesy the CSA)
APIs are Ubiquitous

Cloud Security Alliance
IaaS Reference Model
Cloud Security Must Be Cloud Native

Management Plane  Volatility/Velocity  Distribution/Segregation
Cloud Native == NOT.Legacy

- You build it from scratch
- Redesign without compromise
Building the cloud-native mindset

**Cloud is:**
- Developer Driven
- Abstracted
- Automated
- Massively Scalable
- Volatile
- Ephemeral

**You should:**
- Embrace Automation
- Embrace Failure
- Think Like a Developer
- Kill the Past
- Isolate by Default

{disrupt:Ops}
Cloud Native Security Program Principles

- APIs
- Automation
- Immutability & Isolation
Cloud Native Design Principles

- Fit the infrastructure to the application
- Leverage architecture for security
  - Good design can eliminate common traditional security issues
  - Network attack paths, patch management
- Automate deployment and management with DevOps
  - Required for immutable, cloud disaster recovery, and portability
Architecture + DevOps is cloud security

- Architecture allows you to leverage the shared responsibilities model to reduce your *security management surface* and push them onto a cloud provider that has stronger economic incentives to avoid incidents.
  - You are paying for it anyway, might as well take advantage.
- DevOps provides a **consistency** and **control** impossible with manual application deployments.
- Security can easily **embed** and **automate** into the same toolchains as development and operations.
- Security can steal DevOps techniques to improve security operations. Which is now called **DevSecOps**.
PaaS and "New" Cloud Architectures

- PaaS providers can’t afford a preventable security failure.
  - Including letting things get out of date.
- Many types of PaaS can’t rely on normal networking.
  - Instead you access them via API.
- This creates an opportunity to “air gap” parts of your application.
  - Kill off network attack paths
- To be clear, PaaS doesn’t help with logic flaws
Software Defined Security

- Attackers are automated, security is mostly manual.
  - Our tools have been poor.
- We lack trustable security **automation** and thus need to rely mostly on carbon (that’s you!)
- In cloud, APIs are mandatory. We can write code to automate and orchestrate, even across products and services.
Automation Categories

Guardrails
Continuously assess and enforce operational and security policies
- Fix security group or S3 misconfigurations

Workflows
Streamline and accelerate IT operations and security through automated workflows
- Incident response

Orchestrations
Empower new capabilities through advanced orchestration of infrastructure, operations, and security
- Automatic WAF insertion and configuration
The Principles

- Software Defined Security
- Stateless Security
- Event Driven Security
- Continuous Feedback Loops
The Foundation

Cloud Service Provider

- API and full administrative activity logging
- Events/triggers/rules
- Function as a Service (Serverless)
- Notification service

Cloud Consumer (you)

- Continuous Integration Pipeline
- Version control repository
- Full IAM access to accounts/subscriptions/projects
- Security development team (person)

Critical Capabilities
Building a Guardrail

- Define Your Problem
- Evaluate FOSS/Existing tools
- Determine Tech Stack
- Build Initial Automations (Ops)
- Expand for Scale/Scope
What’s a Guardrail?

- Define and set limits
  - Can be “allow” or “deny”
- Find deviations
  - Assessment or event based
- Evaluate the issue
- Fix/remediate
  - Automatically or manually depending on rules
Example Guardrails

- If you find a public S3 bucket, restrict it to our known network addresses
  - Unless it is approved or tagged
- Don’t allow internal security groups with all ports and protocols open in Prod
  - But allow in Dev
- Require MFA for API access for any user that needs MFA for console access
- Create our baseline IAM policies and roles for all new accounts
  - Based on the environment
- Validate that monitoring and alerting is properly configured
  - And fix if not
- Disable access keys that haven’t been used in 90 days
- Find instances with an IAM role that allows power user or greater access via API
  - Restrict the privileges
- Identify all cross-network peering from accounts we don’t own
  - Then check the security group permissions
What Makes a **Good** Guardrail?

- Accounts for different environments
  - At least Dev vs. Prod
- Handles exceptions
  - And is capable of remembering them
- Understands state and context
- Doesn’t bog down the alert queue
- Can remediate automatically
  - Either completely, or after manual approval
- Ops communications/notifications
- Education, not Blame
Building a Guardrail

1. Define Criteria/Issues
2. Add Filters
3. Set Triggers
4. Add Actions And Targets
Our Guardrail

- **Criteria/Issues**
  - All instances with port 22 open to the 0.0.0.0/0 (the Internet)

- **Filters**
  - Region is us-west-2p (could be VPC/tag/etc)

- **Trigger**
  - Time = every 5 minutes

- **Action**
  - Restrict to known IP range

{disrupt:Ops}
Self-Healing Networks

Change a security group

Event Recorded to CloudTrail

Passed to CloudWatch Log Stream

Triggers an CloudWatch Event

Lambda Function analyzes and reverses

{disrupt:Ops}
Our Event-Driven Guardrail

- **Criteria/Issues**
  - New inbound security group rule added

- **Filters**
  - IAM user, VPC, Tag

- **Trigger**
  - API event (CloudTrail)

- **Action**
  - Reverse + Notify
Expanding to Enterprise Scale

- Hitting all 14 regions simultaneously
- Multiplex
- Central event stream
- Queues/SNS
- AuthN/AuthZ
Building a Workflow

Define Steps
Determine Inputs
Choose Execution Model
Modularize Code

Can be built on Guardrails and support Orchestrations

disrupt:Ops
Our Workflow

- **Steps (Incident Response)**
  - Collect metadata (before we change it)
  - Quarantine on the network and in AWS
  - Snapshot all storage and attach for forensics
  - Analyze

- **Inputs**
  - Instance ID

- **Execution Model**
  - Command line (container or remote)

- **Modularize Code**
  - Classes for analyze vs. respond
  - All methods reusable
Workflow Demo
Workflows Advice

- Workflows are to speed up common, manual tasks
  - Guardrails are for automated enforcement
  - The line between a guardrail action and an Workflows is often thin
- Execution environment matters
  - Lambda vs. containers vs. your laptop
- Use your pipeline
  - Continuous integration servers (Jenkins) make great platforms for repeat automation, not just security testing
- Make a static console
  - E.g. S3 + API Gateway + SQS
Building a Orchestration

1. Locate SDK if available
2. Modularize
3. Consider flow/value
4. Integrate in code

ID apps and APIs

{disrupt:Ops}
Our Orchestration

- Apps/API
  - EC2 + Route 53 + Incapsula
- SDK
  - AWS Ruby + REST client
- Flow/Value
  - ID public web servers -> determine DNS -> check WAF -> add WAF
  - Limit: default AWS domain names
- Modularize
  - Find web instances, ELBs
  - Change DNS, add Incapsula
- Integrate into code
Orchestration Demo

This workflow scans your Amazon deployment to identify web servers (instances and elastic load balancers) not protected with the incapsula Web Application Firewall.
If unprotected servers have domain names managed by Route53, you will have the option to add Incapsula protection.

Press Return to begin:
Complexities

Scaling

Multiple Accounts

Multiple Providers

Circuit Breakers
Architecting For Enterprise Scale

- Collections of scripts/code, not a monolithic application.
- Framework for:
  - Auth
  - Scheduled activities
  - Logging
Where to Start

1. Start with something simple
   ▶ Build it in one account/subscription/project
   ▶ Event + Notification is super easy to start
   ▶ Then go with your first FaaS
   ▶ Desktop first, then FaaS for execution environment

2. Build a library
   ▶ Experiment with execution environments, but standardize quickly

3. Add enterprise scaling capabilities
   ▶ Will depend on your execution environment/model
   ▶ Build it in the cloud and leverage PaaS options

4. Make sure you use CI/CD for long term management
Mike Rothman
@securityincite

Sign up for the {d:Ops} mailing list at disruptops.com
Free trial available in mid-October

mrothman@securosis.com
http://securosis.com/blog

mrothman@disruptops.com
https://disruptops.com/news/blog/