SESSION ID: CSV-R04

PRAGMATIC SECURITY AUTOMATION FOR CLOUD

Rich Mogull
Analyst/VP of Product
Securosis/DisruptOPS
rmogull@disruptops.com
@rmogull
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
The Categories

Guardrails
Continuously assess and enforce operational and security policies

Workflows
Streamline and accelerate IT operations and security through automated workflows

Orchestrations
Empower new capabilities through advanced orchestration of infrastructure, operations, and security

Fix security group or S3 misconfigurations

Incident response

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

Critical Capabilities

Cloud Consumer (you)

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

1. Define Your Problem
2. Eval FOSS/Existing tools
3. Determine Tech Stack
4. Build Initial Automations (Ops)
5. Expand for Scale/Scope
Things We Are Skipping (for time)

- How to configure all the core monitoring/logging
- Setting up IAM and permissions
- The details of implementation on Azure and GCP
  - We will list the core capabilities, but can’t cover all 3 with real examples in 45 minutes
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 Blamification
Building a Guardrail

1. Define Criteria/Issues
2. Set Triggers
3. Add Filters
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

---

**Step 1: Create rule**

Create rules to invoke Targets based on Events happening in your AWS environment.

**Event Source**

Build or customize an Event Pattern or set a Schedule to invoke Targets.

- **Event Pattern**
- **Schedule**
  - Fixed rate of 5 Minutes
  - Cron expression 0/5 * * * ? *

* Required

---

Demo
Our Event-Driven Guardrail

- Criteria/Issues
  - New inbound security group rule added
- Filters
  - IAM user, VPC, Tag
- Trigger
  - API event (CloudTrail)
- Action
  - Reverse + Notify

Demo
Expanding to Enterprise Scale

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

1. Define Steps
2. Determine Inputs
3. Choose Execution Model
4. Modularize Code

Can be built on Guardrails and support Orchestrations
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

---

Demo

```
Enter Instance ID: i-3dbd9f09
Metadata for i-3dbd9f09 appended to ForensicMetadataLog.txt
Quarantining i-3dbd9f09...
i-3dbd9f09 moved to the Quarantine security group from your configuration settings.
Tagging instance with 'IR'...
Instance tagged and IAM restrictions applied.
Identifying attached volumes...
Volume vol-2d3eddb21 identified; creating snapshot
Snapshots complete with description: IR volume vol-2d3eddb21 of instance i-3dbd9f09 at 2014-02-28 11:47:33 -0700
Volume vol-6212f26e identified; creating snapshot
Snapshots complete with description: IR volume vol-6212f26e of instance i-3dbd9f09 at 2014-02-28 11:47:33 -0700
A forensics analysis server is being launched in the background in with the name 'Forensics' and the snapshots attached as volumes starting at /dev/sdf (which may show as /dev/xvdf). Use host key rmogull-oregon for user ec2-user
Press Return to return to the main menu
```
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. Consider flow/value
3. Modularize
4. Integrate in code

ID apps and APIs
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
  - See video

Demo
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

- 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

- Build a library
  - Experiment with execution environments, but standardize quickly

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

- Make sure you use CI/CD for long term management
PRAGMATIC SECURITY AUTOMATION FOR CLOUD

Rich Mogull
Analyst/VP of Product
Securosis/DisruptOPS
rmogull@disruptops.com
@rmogull