Pragmatic Cloud Security Automation

Rich Mogull/Crash/@rmogull
Securosis and DisruptOps

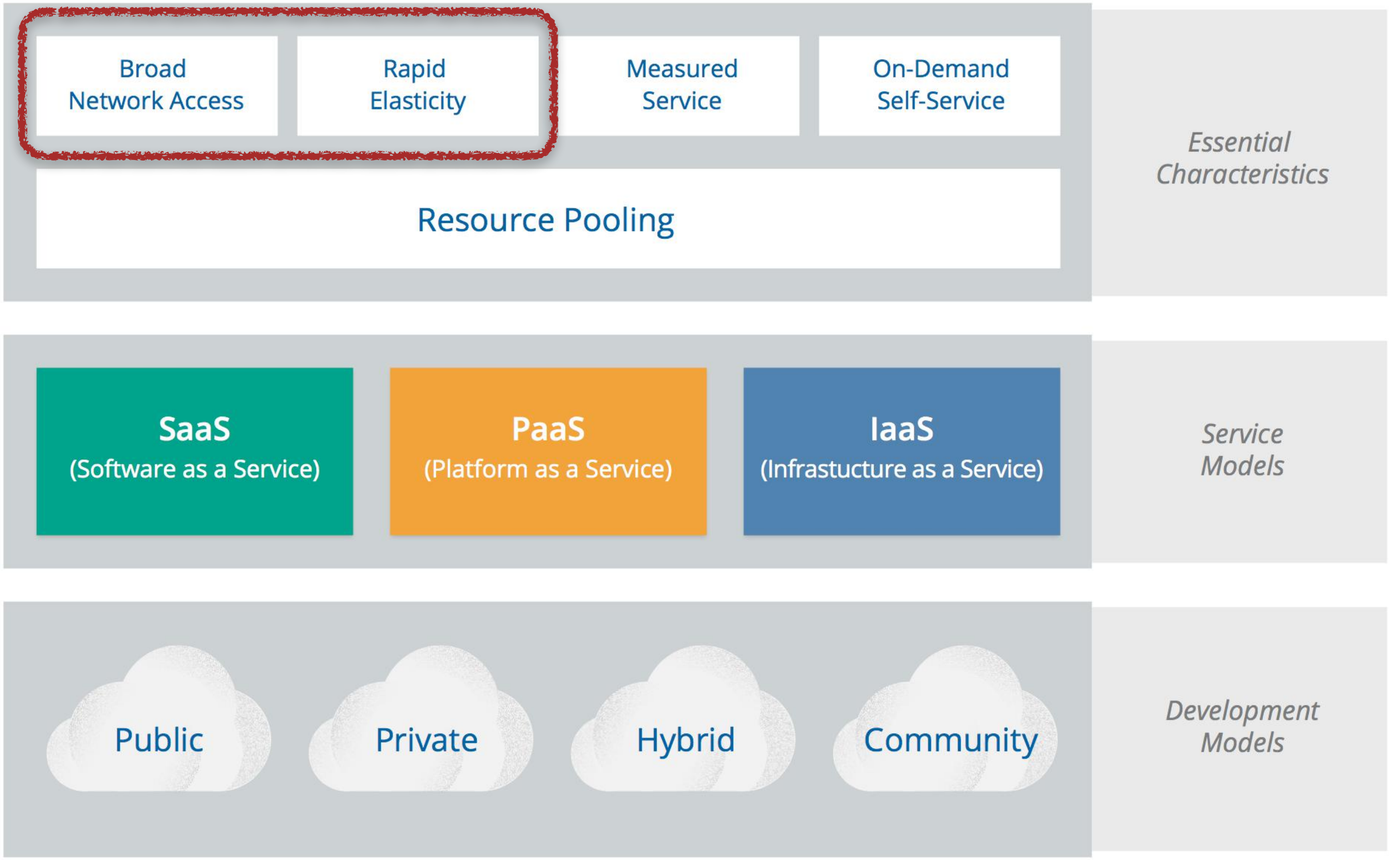
Cloud is Fundamentally Different

Abstraction

Automation

Securosis

Automation is Inherent

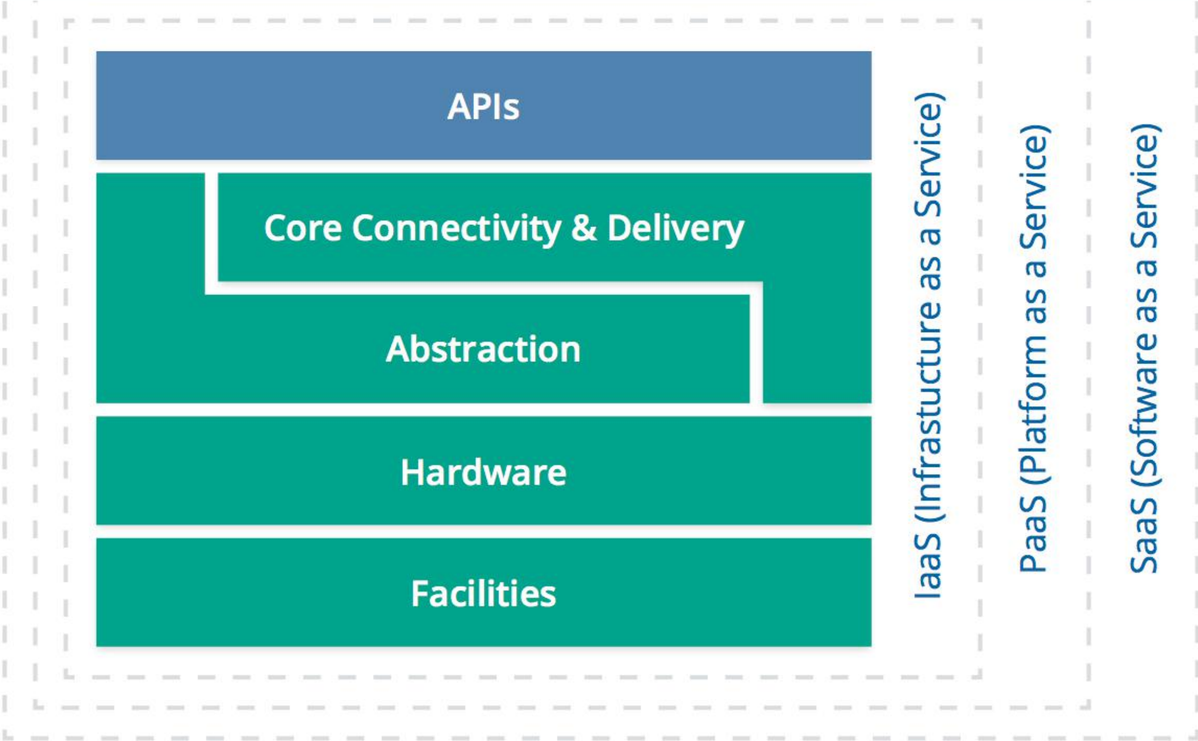


The NIST Model (courtesy the CSA)

Securosis

APIs are Ubiquitous

Cloud Security Alliance
IaaS Reference Model



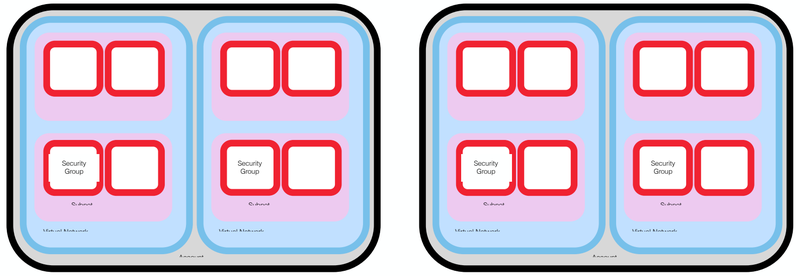
Securosis

Cloud Security Must Be Cloud Native

Management Plane

Volatility/Velocity

Distribution/Segregation



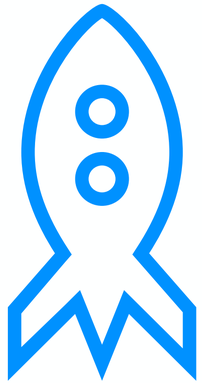
Securosis

The Categories

Guardrails

Continuously assess and enforce operational and security policies

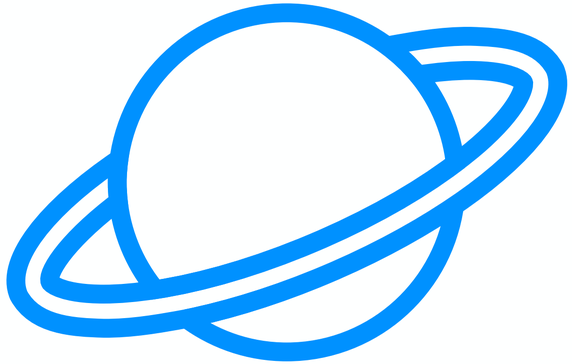
Fix security group or S3 misconfigurations



Workflows

Streamline and accelerate IT operations and security through automated workflows

Incident response



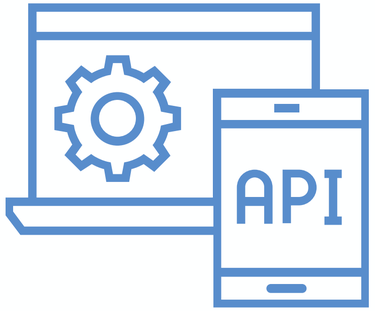
Orchestrations

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

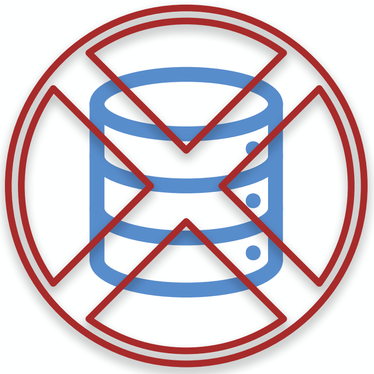
Automatic WAF insertion and configuration

Securosis

The Principles



Software Defined Security



Stateless Security

Event Driven Security

Continuous Feedback Loops

Securosis

The Foundation

Cloud Service Provider

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

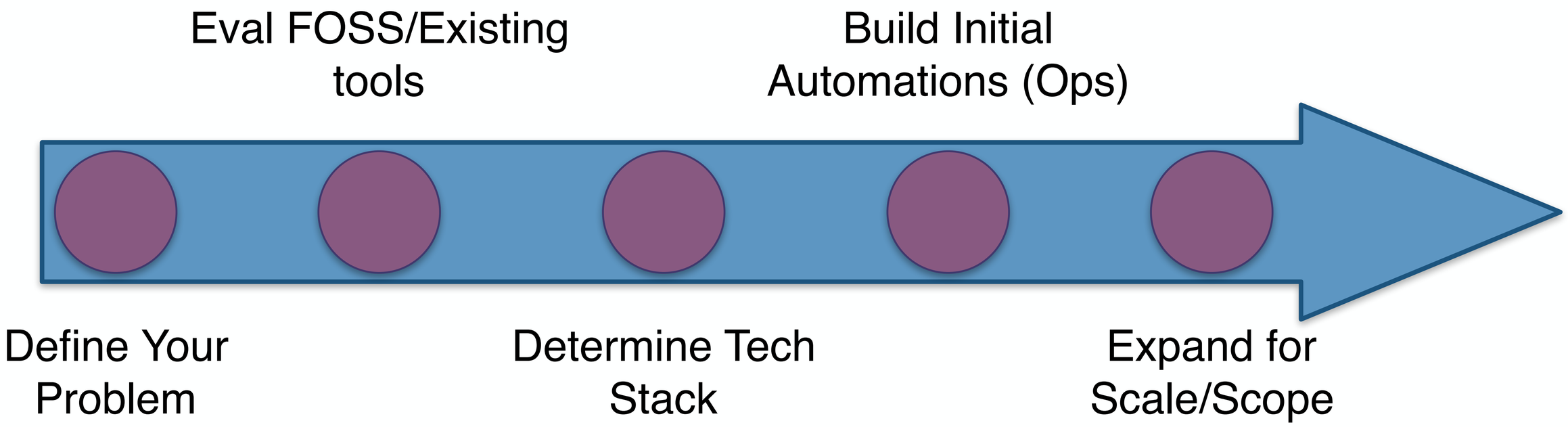
Critical Capabilities

Cloud Consumer (you)

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

Securosis

The Process



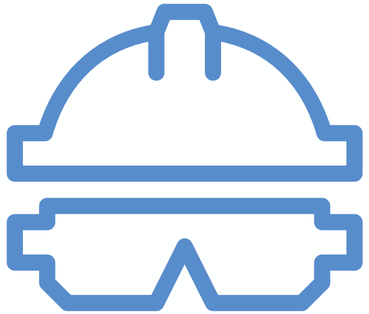
Securosis

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

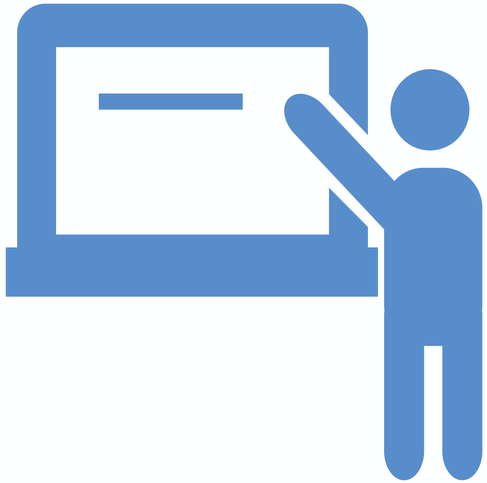


Example Guardrails

- 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

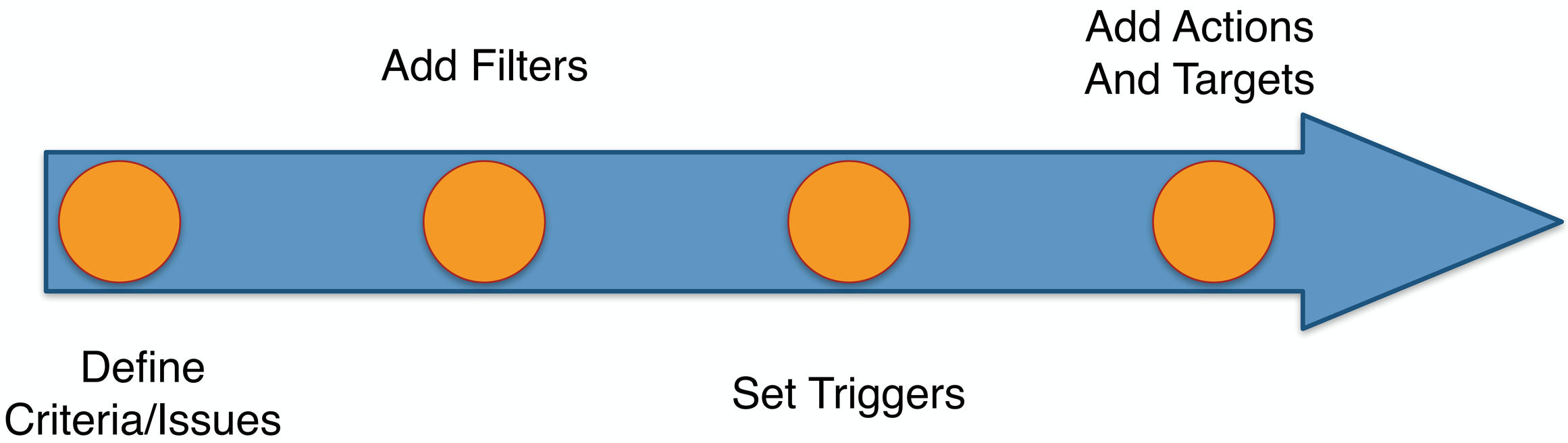
What Makes a ***Good*** Guardrail?



- 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

Securosis

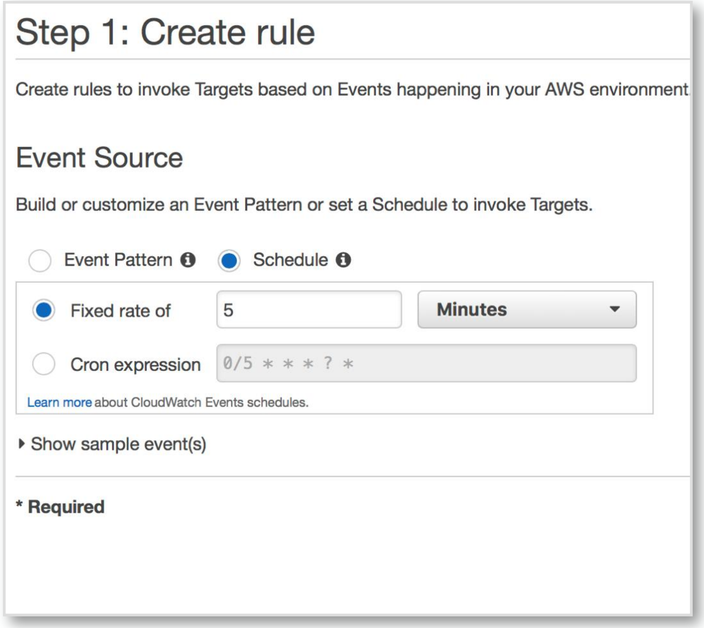
Building a Guardrail



Securosis

Our Guardrail

- Criteria/Issues
  - All instances with port 22 open to the 0.0.0.0/0 (the Internet)
- Filters
  - Region is us-west-2 (could be VPC/tag/etc)
- Trigger
  - Time = every 5 minutes
- Action
  - Restrict to known IP range



Easy. by DisruptOPS

Demo

Securosis

Code Walk Through

- Key aspects:
  - Authentication/authorization via Roles
    - Initializing clients
  - Understanding method and variable scope
  - AWS SDK/JSON navigation
    - Structs > hash > arrays
    - Hidden complexities (e.g. ENIs and security groups)
  - Tips
    - Waiters
    - Managing API limits
    - CLI vs. SDK (—query)

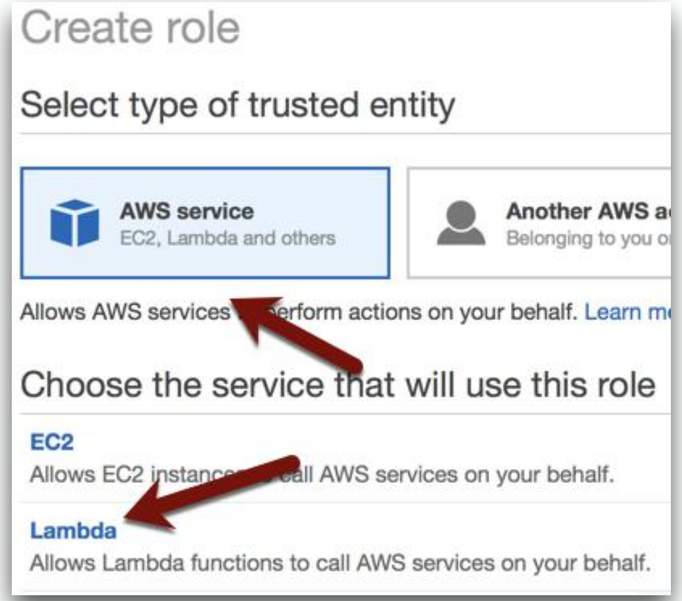
Coding Recommendations

- Language doesn't matter… as long as it supports Lambda
- Understand the AWS credentials hierarchy
  - Hard coded > specified credentials file > default config and credentials files > role
- API limits are a thing. They suck
  - Paginators are your friend when available
- Make sure you understand how to use server side filtering and when it hurts more than it helps

Lab: Create Time-Based Guardrail

- Create a new IAM role to run the Lambda functions for today
  - Give it AdministratorAccess policy only to speed things up
    - NEVER EVER DO THIS IN REAL LIFE!!!
    - (Yes, I've found it in evaluations)
  - Name it lambda_admin

Use the SharedServices account



Lab: Create Notification (or use one from an earlier lab)

- Create a new topic, or pick your existing topic, from SNS
- Make sure you have an active subscription (e.g. SMS or email) to receive the notifications
- Copy and paste the topic ARN to your cheat sheet

Lab: Create the Function

- Create a new Lambda function
  - Name it identify_internet_facing_servers
  - Choose Python 3.x
  - Choose the lambda_admin role
- Paste in the sample code from your student directory
  - If you are a hacker, or ever wanted to be a hacker, figure out how to change to dark mode.
- If you hit an error wait 1-2 minutes and try again, sometimes IAM is slow. Welcome to the cloud!

lambda_function ×

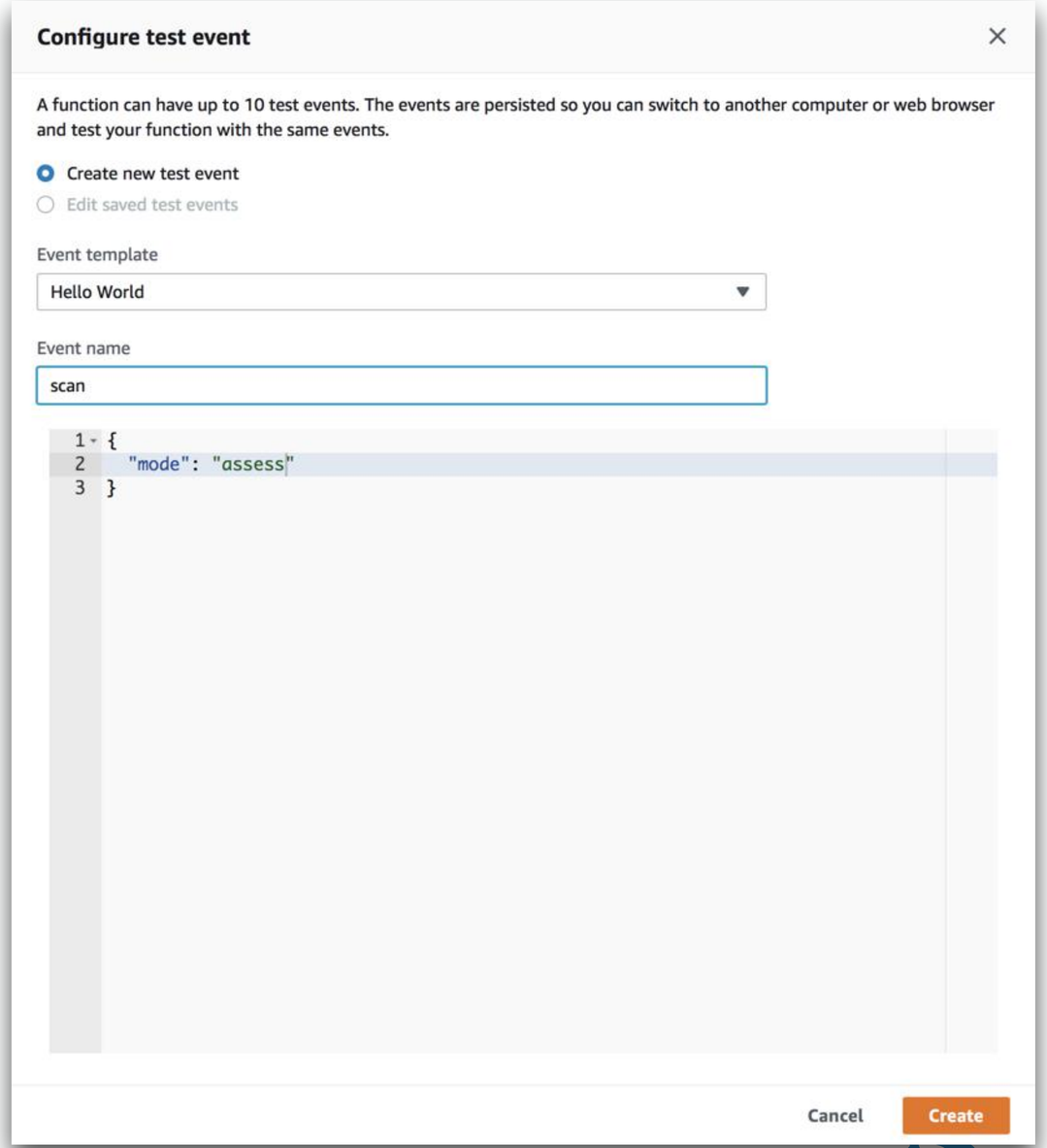
```python
import boto3
import json

# Updates needed- IPV6, an IP range that includes port 22
# Also add fix in case requested IP range is already in there

def update_security_group(event, group, existingIpPermission, port, new_range):
    if "clickType" not in event:
        ec2 = boto3.client('ec2', region_name=event)
    else:
        ec2 = boto3.client('ec2')
    sub_range = []
    for curIP in existingIpPermission['IpRanges']:
        if curIP['CidrIp'] != '0.0.0.0/0':
            sub_range.append(curIP)
    sub_range.append(new_range)
    remove_rule = ec2.revoke_security_group_ingress(
    GroupId=group,
    IpPermissions=[
            {'IpProtocol': 'tcp',
             'FromPort': port,
             'ToPort': port,
             'IpRanges': existingIpPermission['IpRanges']}
               ])
    add_rule = ec2.authorize_security_group_ingress(
    GroupId=group,
```

Lab: Create Test Event

- Create the test event (it's on the top of the Lambda page
- Paste in the sample JSON from your cheat sheet (it's under ### Guardrail)
- Replace with the ARN of your SNS topic
- Update the SNS ARN in the Lambda function: "TargetArn" around line 110



Test

Test

Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution.

```
null
```

Summary

Code SHA-256	jE6w+hOYjz5ZilYuHQ7tc7Tj7ErAN3KBIATR7CO7C5w=	**Request ID**	f4beabf1-83a1-11e8-addc-79cb7b7572de
Duration	1584.43 ms	**Billed duration**	1600 ms
Resources configured	128 MB	**Max memory used**	37 MB

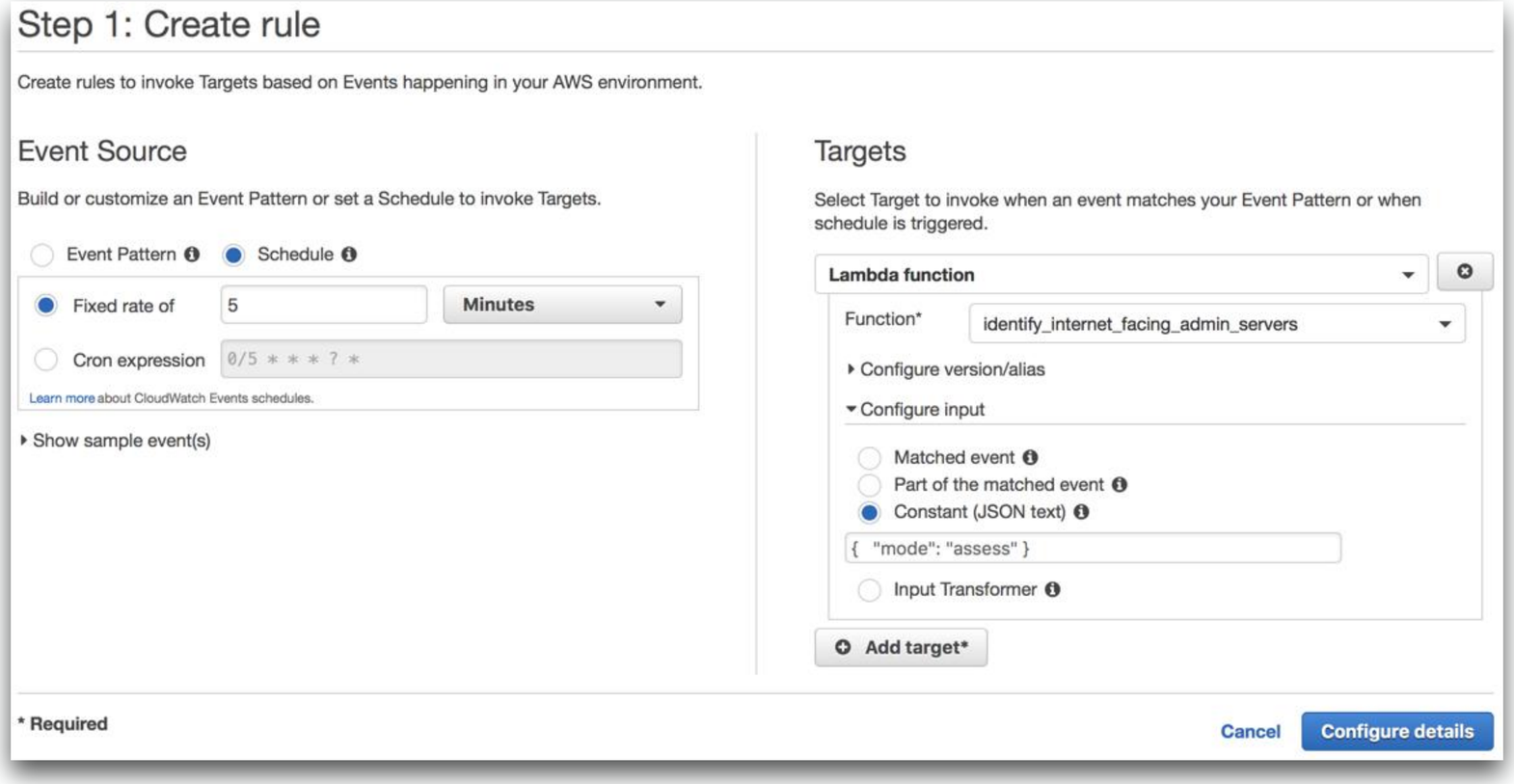
Log output

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.

```
SGs [{"Description": "Temporary group for Packer", "GroupName": "packer_5afcc818-8aae-faa1-8d95-1edc92810fad", "IpPermissions": [{"FromPort": 22,
"IpProtocol": "tcp", "IpRanges": [{"CidrIp": "0.0.0.0/0"}], "Ipv6Ranges": [], "PrefixListIds": [], "ToPort": 22, "UserIdGroupPairs": []}], "OwnerId":
"935440313651", "GroupId": "sg-1a40aa6b", "IpPermissionsEgress": [{"IpProtocol": "-1", "IpRanges": [{"CidrIp": "0.0.0.0/0"}], "Ipv6Ranges": [],
"PrefixListIds": [], "UserIdGroupPairs": []}], "VpcId": "vpc-fbc4a793"}, {"Description": "Temporary group for Packer", "GroupName": "packer_5b19c495-c23d-
0d03-e1e9-501aab880c0c", "IpPermissions": [{"FromPort": 22, "IpProtocol": "tcp", "IpRanges": [{"CidrIp": "0.0.0.0/0"}], "Ipv6Ranges": [], "PrefixListIds":
[], "ToPort": 22, "UserIdGroupPairs": []}], "OwnerId": "935440313651", "GroupId": "sg-893e50f8", "IpPermissionsEgress": [{"IpProtocol": "-1", "IpRanges":
[{"CidrIp": "0.0.0.0/0"}], "Ipv6Ranges": [], "PrefixListIds": [], "UserIdGroupPairs": []}], "VpcId": "vpc-fbc4a793"}]
END RequestId: f4beabf1-83a1-11e8-addc-79cb7b7572de
REPORT RequestId: f4beabf1-83a1-11e8-addc-79cb7b7572de  Duration: 1584.43 ms    Billed Duration: 1600 ms        Memory Size: 128 MB     Max Memory Used:
37 MB
```

Securosis

Lab: Set Schedule



- Create a CloudWatch Rule to run the lambda function every 5 minutes (or sooner if you want)
- Provide the configuration details by pasting in the JSON from your test (e.g. “mode”: “assess”)

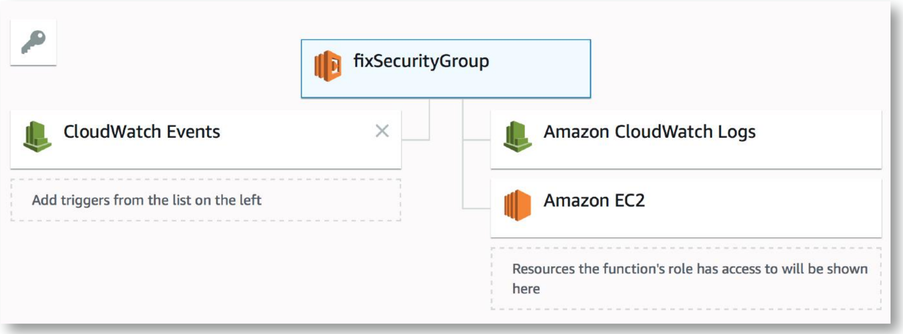
Securosis

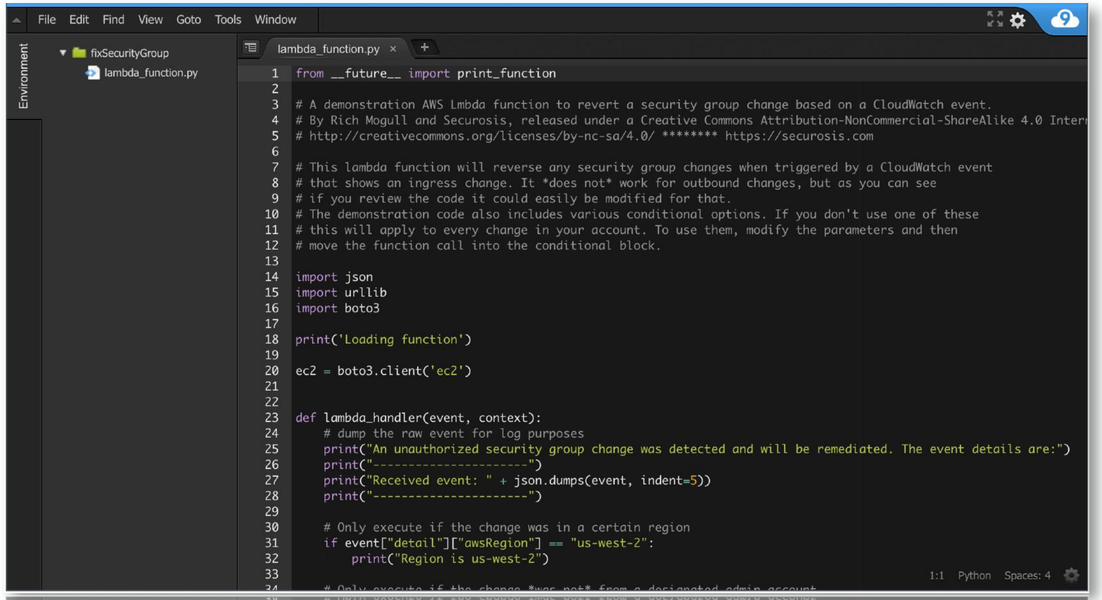
Now try putting it into remediation mode

Securosis

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

Securosis

Self-Healing Infrastructure (yes, for real)

Type	Protocol	Port Range	Source
SSH	TCP	22	68.2.174.98/32
HTTP	TCP	80	0.0.0.0/0

Change a security group

Lambda Function analyzes and reverses

Event Recorded to CloudTrail

Passed to CloudWatch Log Stream

Triggers an CloudWatch Event

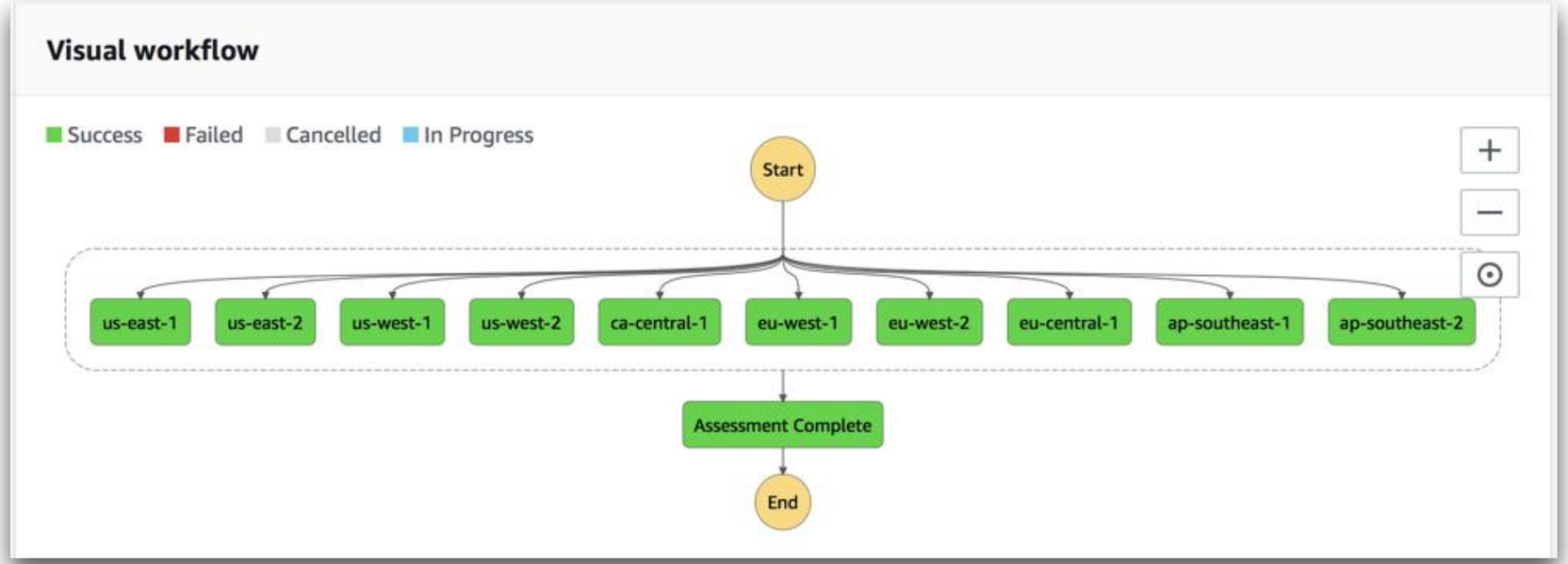
Securosis

Lab: Event-Driven Guardrail

- Create a new lambda function using Python 2.7 and use the same role
- Paste in the content from the revert_security_groups.py file
- Either add the lambda as a second target to your existing alert or create a new CloudWatch rule to trigger this event anytime there is the API call "AuthorizeSecurityGroupIngress"
  - At this point, you should be able to figure this out
  - Pass in the raw event source to the Lambda
- Change a security group to test it
  - This version of the demo code only reverts an ingress authorization. It may also miss certain change operations
  - It does **not** revert IPV6 permissions if your VPC supports it

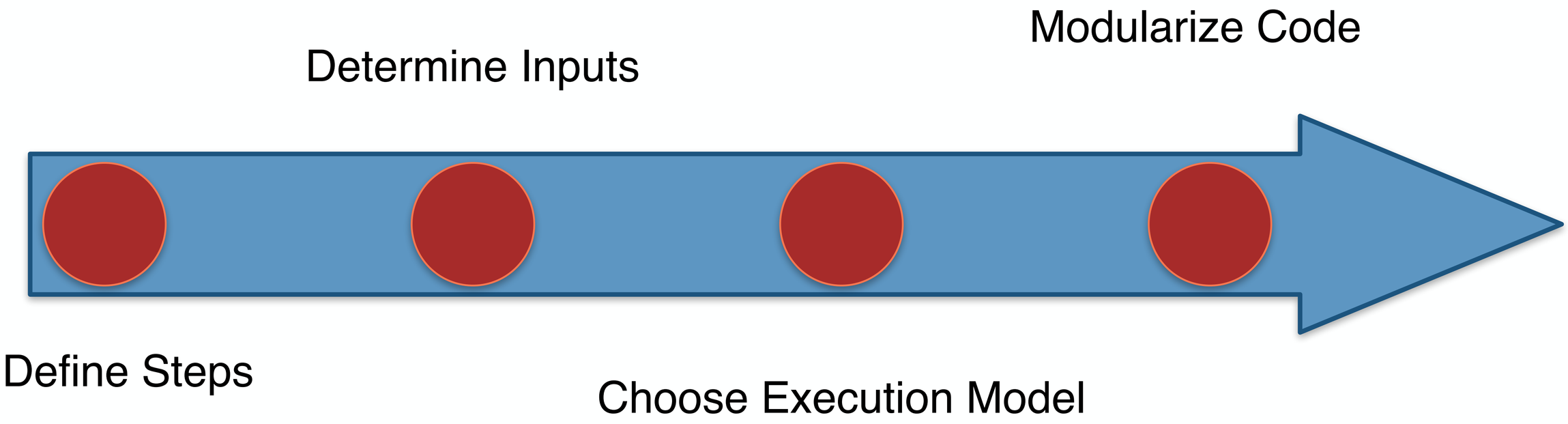
Expanding to Enterprise Scale

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



Securosis

Building a Workflow

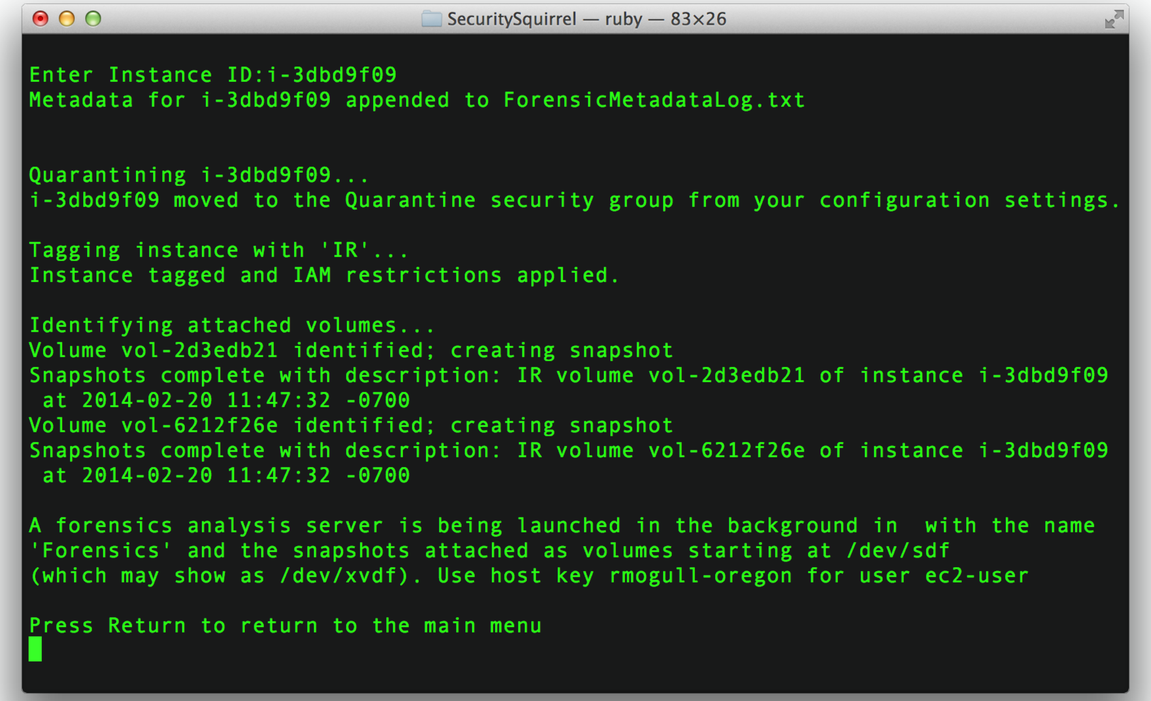


Can be built on Guardrails and support Orchestrations

Securosis

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

Securosis

Lab: Run the Incident Response Workflow

- This is pre-loaded in Admin
  - Launch an instance you can quarantine in your ***default*** *VPC*
    - If you want to use your SecOps VPC you will need to update the code
  - Create a new security group named “quarantine” without any permissions *in the same VPC as your target instance*
  - Log in and `cd ir`
  - `nano config.json`
    - Modify settings for us-west-2 as indicated then save
      - Change the security groups
      - User your SSH key name
      - Update the AMI to ami-082b5a644766e0e6f
  - `ruby ir.rb`

ir.rb Current Limitations

- This is older code we haven't fully updated as better-supported tools are emerging
  - https://threatresponse.cloud
- Everything has to be in the same VPC (target + security groups)
- Requires hard-coding of various IDs
  - These days we code automations to look for required resources, like security groups, then create them if they don't exist
- There is a bunch of in-development code in there that isn't fully functional yet

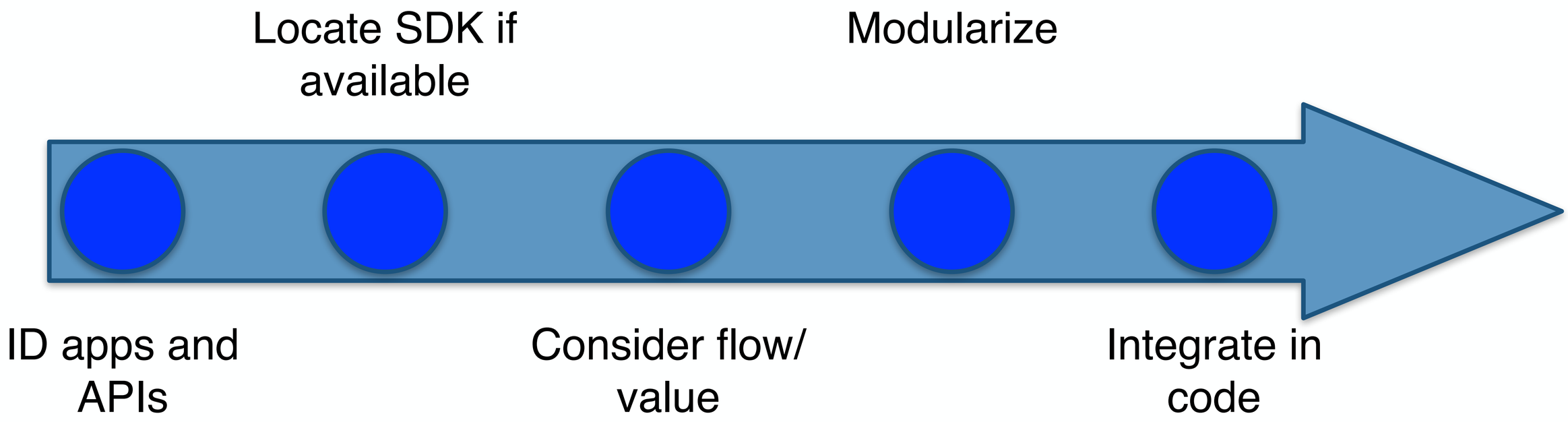
Lab: Add code to stop the instance

- https://docs.aws.amazon.com/sdkforruby/api/index.html

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

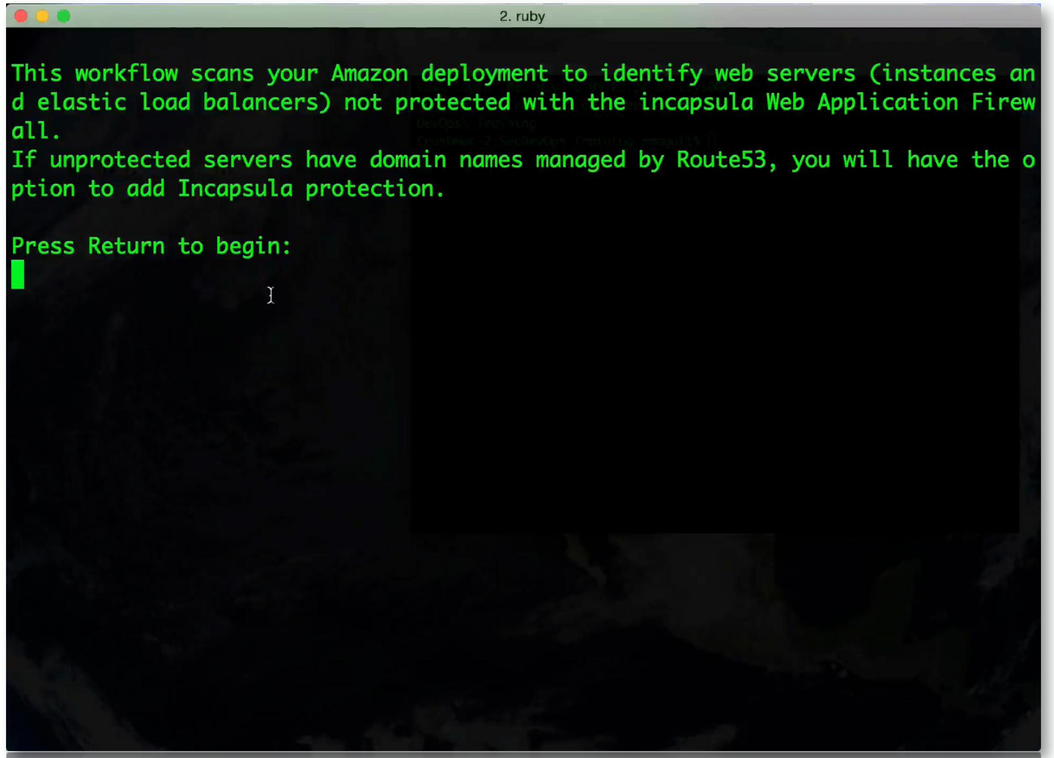
Building an Orchestration



Securosis

Our Orchestration Demo

- 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

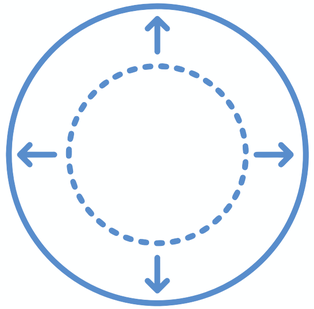


Demo

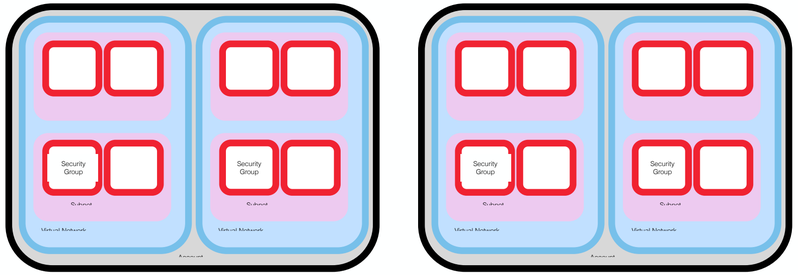
Securosis

Your Student Share directory includes multiple sample lambdas for you to experiment with and modify if you have the time

Complexities

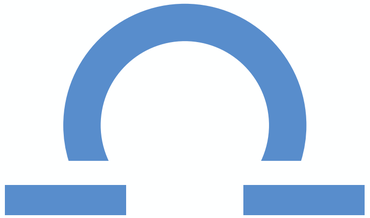


Scaling



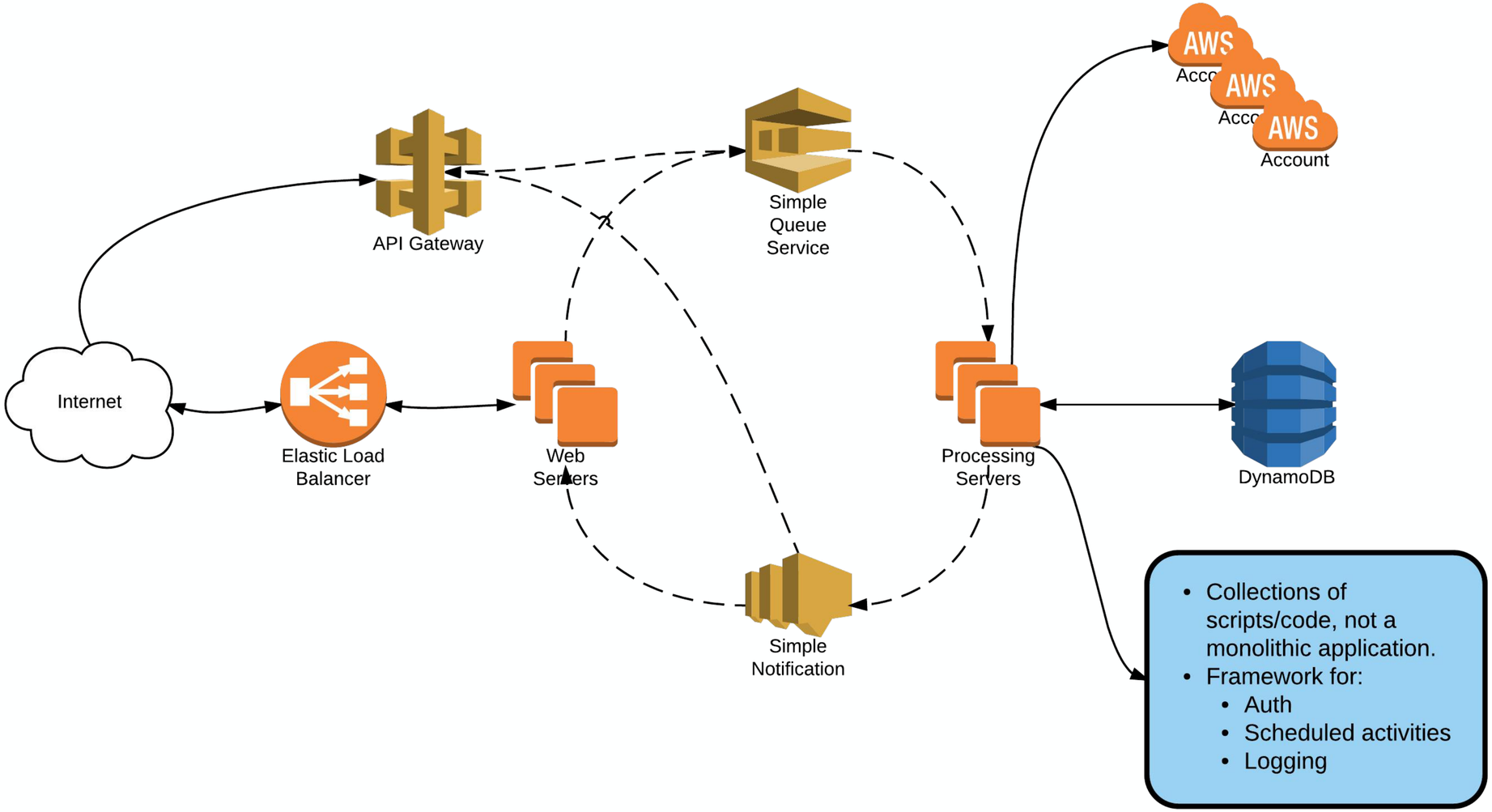
Multiple Accounts

Multiple Providers



Circuit Breakers

Architecting For Enterprise Scale



Securosis

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

Incident Response

Securosis

Key Incident Response Issues

- Real world cloud IR is both better and worse than traditional infrastructure:
  - You still need to manage compromised resources (e.g. instances).
  - You also need to add the cloud management plane to the scope.
  - The cloud provider and you will have different priorities.
  - You may have more or less control, depending on your governance and SaaS vs. IaaS.
    - E.g. you can totally manage the infrastructure remotely with automation, which is an advantage. But in SaaS you might not control much of anything.
    - You have to rely less on network packet capture.
  - Immutable infrastructure is a powerful recovery option.
  - Containment can be much easier.

Securosis

Key Principles

- Know who to call
- Train on your providers of choice
- Write your response procedures and automation code ahead of time
- Don’t rely on manual response
- Use immutable for recovery as often as possible
- **Kill IAM/metastructure access first**
  - Don’t forget that on both the network and with IAM/management plane you may need to **kill active sessions**, not merely revoke access

Background: How to Image an Instance

- Get the instance ID from the EC2 console
- Click on volumes and filter on the instance ID
- Snapshot the volume(s) and record the snapshot ID
- Create a new volume based on the snapshot
  - When you create a new volume you can base it on the snapshot ID
- Attach the new volume to a running instance (and remember the device mapping)
- Log into the running instance and start your forensics

Lab: Incident Response

- This is the capstone lab for this training, leveraging multiple skills.
- You will launch a CloudFormation template to set everything up and launch an attack simulator in 2 accounts
  - That instance will simulate a cloud-native attack on your accounts
  - The activities are all constrained, but represent techniques a real attacker would uses
  - It is also designed to be easy to clean up and allow you to perform a response in the allotted time.
- You must follow all the normal steps in an IR process.

IR Lab Prep

In the SharedServices account

- Do not modify the current account security
  - However, this is where you will deploy any analysis tools to complement the tools already installed
  - Consider using those tools to assess and harden the WebappProduction account

Use the WebappProduction account

- Your instructor will give you a time window to harden the account
- Your objective is to take everything you have learned to prepare the account for the upcoming attack

IR Lab Prep Part 2

- Consider writing an SCP for the Incident Response OU
  - What would you put into an SCP that would help in an incident?
  - Would those changes break the application and is this acceptable?
  - How can you use the SCPs to contain the attack without destroying needed forensics?
- Then, when your instructor tells you
  - Follow the instructions on the next page to start the simulation
  - Run the CloudFormation template in both SharedServices **and** WebappProduction
- Using both accounts will help you better understand the role of your defenses

Lab: Incident Response

- Launch the CloudFormation template on your cheat sheet:
  - us-west-2 as usual
  - Wait 5-ish minutes for it to settle

- **You must!**
  - Follow the IR steps above
  - Contain the attack
  - Determine what happened
- We will provide full cleanup instructions separately

Securosis

IR Lab Constraints and Reality

- This attack simulation is deliberately constrained:
  - It relies on provided admin credentials and skips the hard part of exploitation.
  - It uses all pre-determined resources to ensure we can clean it up.
  - It purposely doesn’t attack certain resources that could either violate terms of service or damage your account.
  - It is designed to fit within our classroom time constraints.
- However:
  - It does demonstrate multiple real-world techniques used by cloud native attackers.
  - It forces you to think in cloud-native response terms.

IR Discussion

- How could an attacker compromise credentials to carry out this attack?
- How could they escalate privileges if they only gain access to lower-level credentials?
- What inherent tools and techniques would prevent the various attacks demonstrated in this lab?
- How could you use automation? Do you think it's required?

What We Covered

- Baseline security, from the account architecture and root account through IAM, monitoring, and network security
- Real-world network architectures and security
- Leveraging DevOps techniques and deployment pipelines for security
- A primer on leveraging cloud-native options for building secure application architectures
- Security automation
- Incident response for cloud