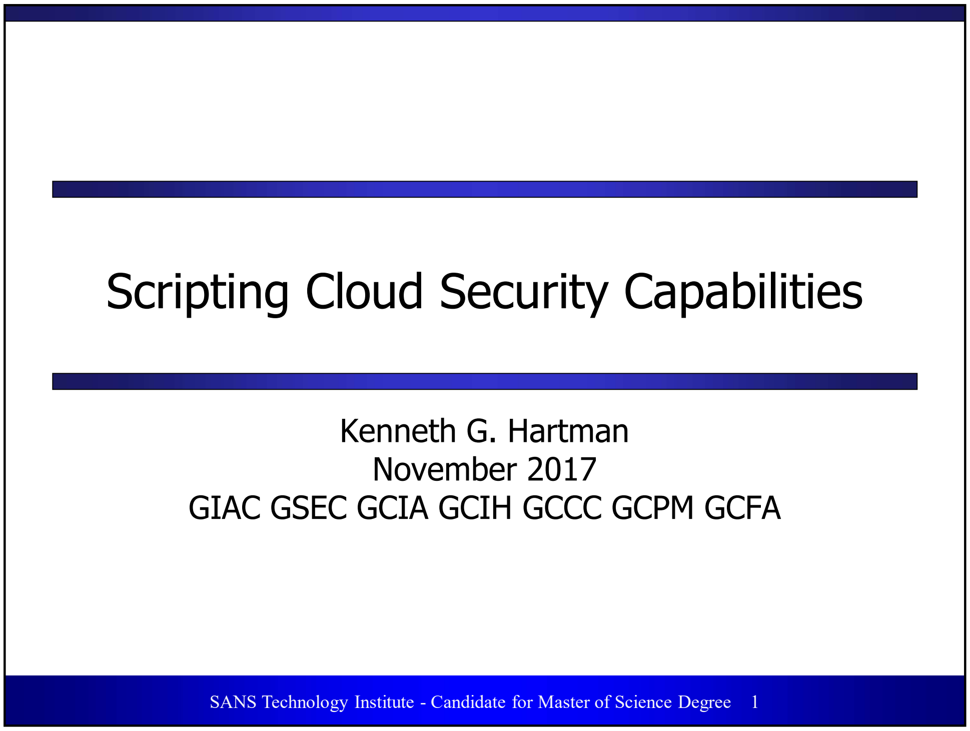
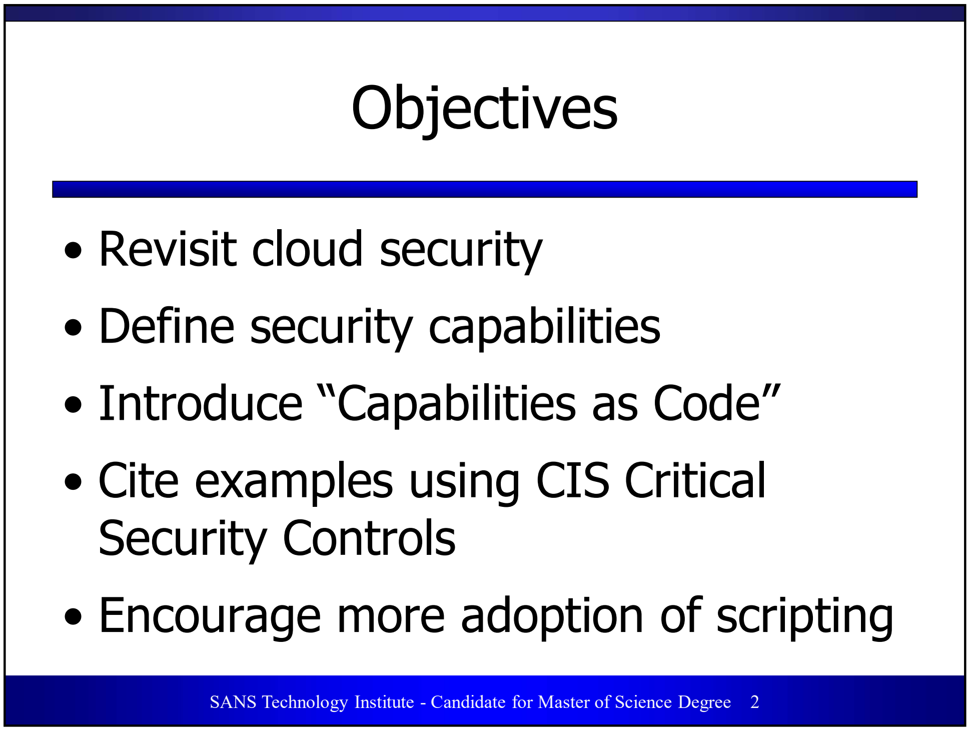


Scripting Cloud Security Capabilities

Kenneth G. Hartman
November 2017
GIAC GSEC GCIA GCIH GCCC GCPM GCFA

Abstract: ITIL defines a capability as the “ability to carry out an activity” and indicates that capabilities are assets that can be intentionally managed and improved in pursuit of the company’s mission. NIST Special Publication 800-53R4 states that a security capability generally results from the selection and implementation of a set of mutually reinforcing security controls. Forward-thinking companies like Google. Microsoft, and Amazon are delivering their cloud services such that they can be consumed by other services via an Application Programming Interface (API). This has given rise to several important concepts such as Software Defined Networking, Orchestration, and Infrastructure as Code. A central theme is that everything that is customized or unique has been reduced so that it can be expressed as version-controlled program code. This allows organizations to encapsulate, inherit, abstract, and reuse their IT capabilities just like other code. Using selected examples from the CIS Critical Security Controls , this presentation will share some concepts, tools and practical experiences of a security engineer using the “capabilities as code” approach to improve the security of his organization’s use of Amazon Web Services.



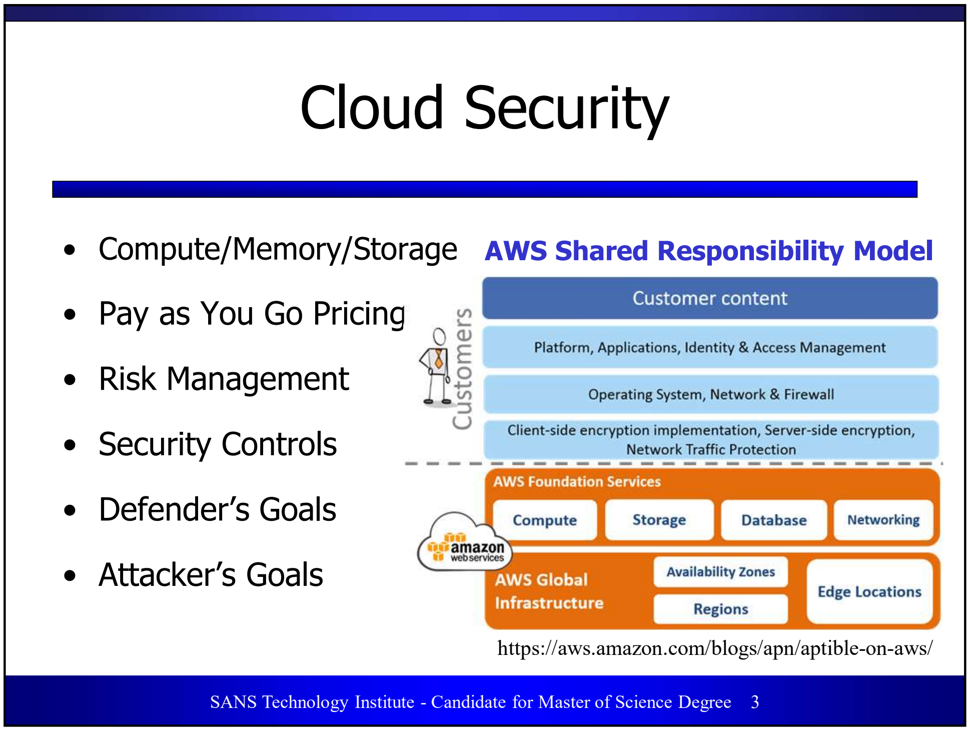
Objectives

- Revisit cloud security
- Define security capabilities
- Introduce "Capabilities as Code"
- Cite examples using CIS Critical Security Controls
- Encourage more adoption of scripting

SANS Technology Institute - Candidate for Master of Science Degree 2

Although this presentation is heavily focused on Amazon Web Services, the concepts apply equally well to Google Cloud Platform and Microsoft Azure as well. My goal with this talk is to encourage you to use the cloud to sharpen your security skills and to do it in such a way as to create a portfolio of capabilities.

In other words, "document your security capabilities as program code and improve with each iteration."



Cloud Security

- Compute/Memory/Storage
- Pay as You Go Pricing
- Risk Management
- Security Controls
- Defender’s Goals
- Attacker’s Goals

AWS Shared Responsibility Model

Customers

Customer content

Platform, Applications, Identity & Access Management

Operating System, Network & Firewall

Client-side encryption implementation, Server-side encryption, Network Traffic Protection

AWS Foundation Services

Compute | Storage | Database | Networking

AWS Global Infrastructure

Availability Zones | Regions | Edge Locations

https://aws.amazon.com/blogs/apn/aptible-on-aws/

SANS Technology Institute - Candidate for Master of Science Degree 3

Compute/Memory/Storage & Pay as you go

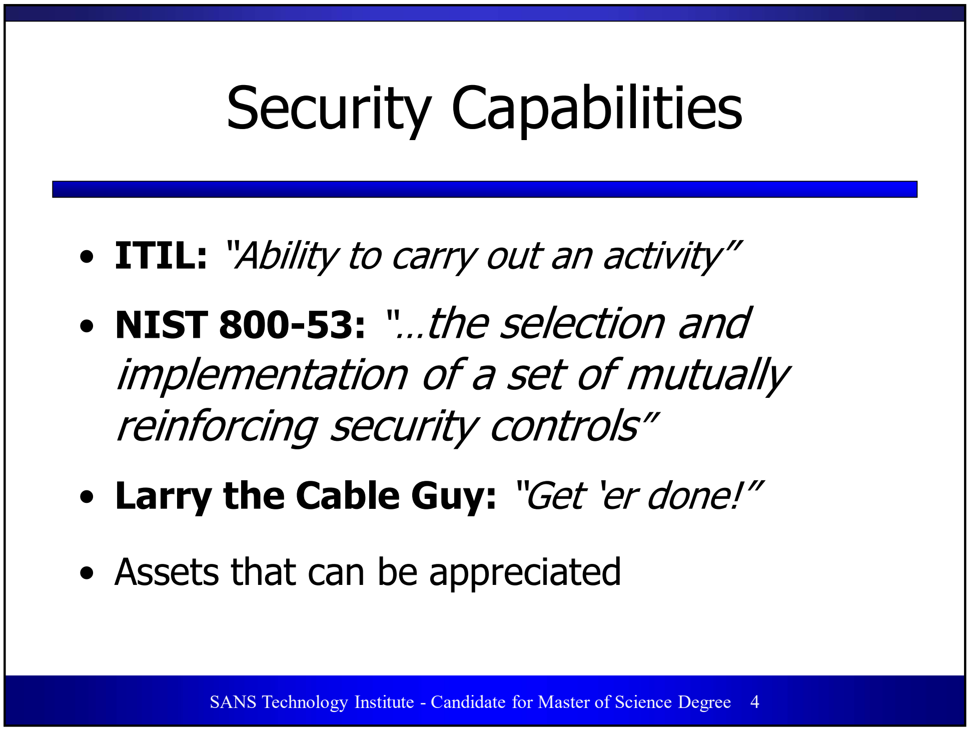
Managing Risks to AIC by treating risks with security controls that can be measured and audited

Defender’s Goal:

- Protect IP, PII, and Customer Data – making sure it is available to authorized users who need to harness this info to the organization’s advantage and prevent unauthorized access from absconding the value of that data for themselves

Attackers Goals:

- Exploit the value of the Data to the benefit of the attacker, or
- Diminish the Value of the data to the detriment of the data owner
- The discussion of who owns PII is outside the scope of this talk



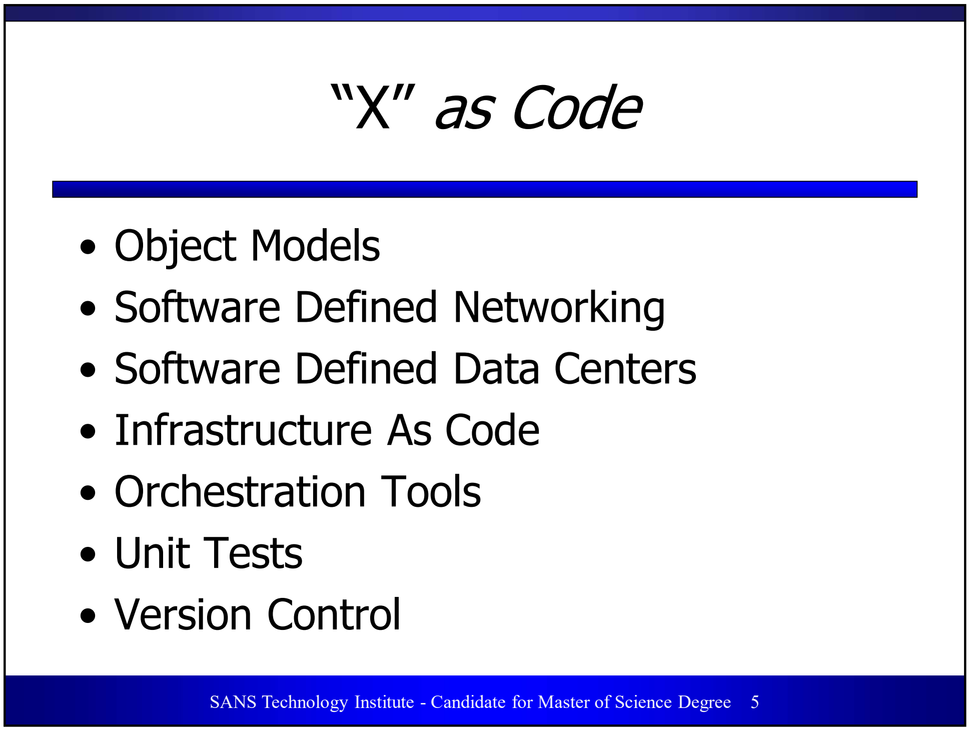
Security Capabilities

- **ITIL:** *"Ability to carry out an activity"*
- **NIST 800-53:** *"…the selection and implementation of a set of mutually reinforcing security controls"*
- **Larry the Cable Guy:** *"Get 'er done!"*
- Assets that can be appreciated

Security capabilities are assets that can be "appreciated." – When security capabilities are documented, they become an asset that can grow in value, especially if they are practiced, refined, and evolve as the business changes. This takes investment, however it is human nature to appreciate things more that require more effort and expense.

"Do cool Shit every day…or die trying" – Tom Peters, in the *Brand You 50*

"Either write something worth reading or do something worth writing." – Benjamin Franklin



“X” *as Code*

- Object Models
- Software Defined Networking
- Software Defined Data Centers
- Infrastructure As Code
- Orchestration Tools
- Unit Tests
- Version Control

SANS Technology Institute - Candidate for Master of Science Degree 5

Expose an API with an Object Model that can be manipulated programmatically (often by Python)

These services are designed with an API and expose an object model that can be manipulated programmatically. It also allows for all of the object-oriented programming constructs to come into play. For example:

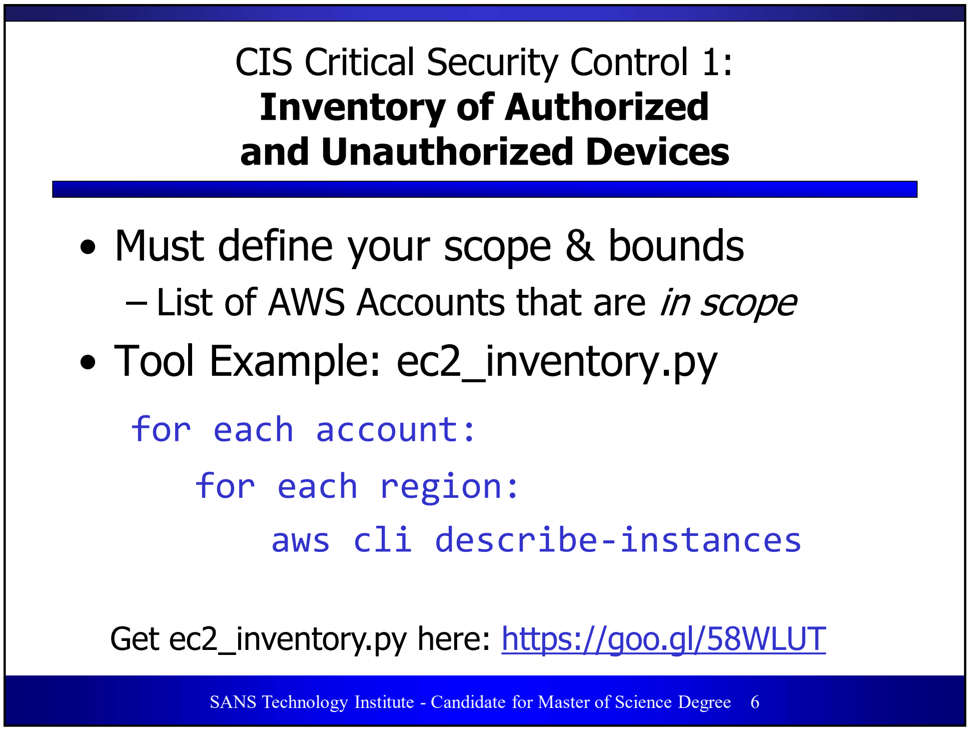
- *Abstraction (Trust Boundaries & Threat Modeling)*
- *Encapsulation*
- *Inheritance & Code Reuse*
- *Polymorphism*

AWS CLI and Boto3

Other Tools: Scapy, GRR/Rekall, Matplotlib

- *SEC573: Automating Information Security with Python*

Artificial Intelligence & Machine Learning



CIS Critical Security Control 1: **Inventory of Authorized and Unauthorized Devices**

- Must define your scope & bounds
  - List of AWS Accounts that are *in scope*
- Tool Example: ec2_inventory.py

```
for each account:
    for each region:
        aws cli describe-instances
```

Get ec2_inventory.py here: https://goo.gl/58WLUT

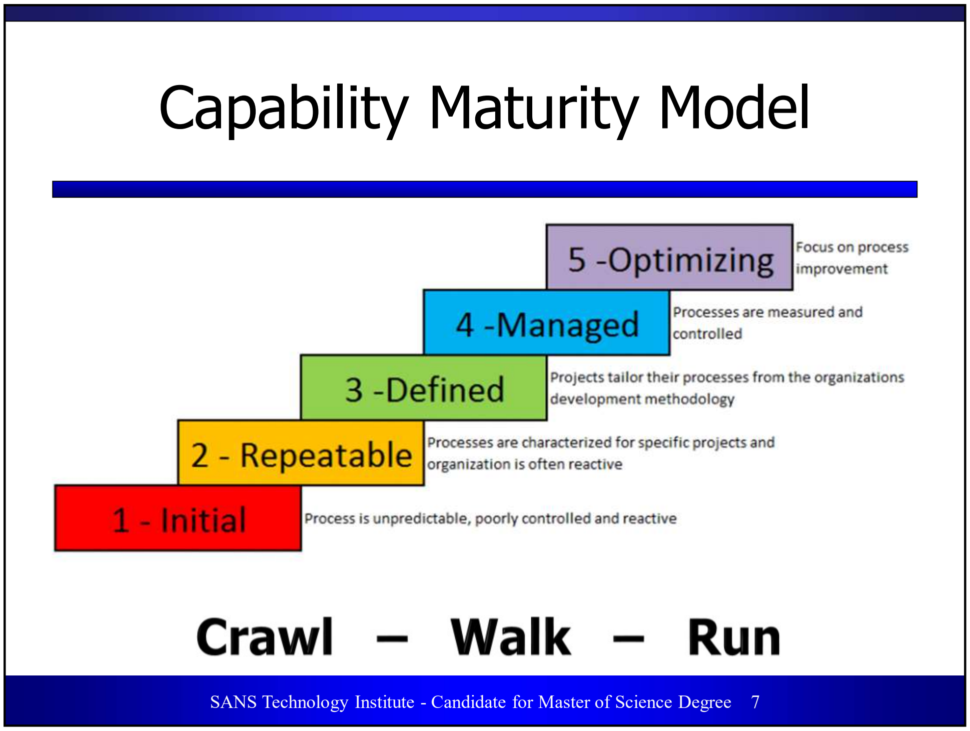
SANS Technology Institute - Candidate for Master of Science Degree 6

Security starts with a clear understanding of the scope of what you are tasked to secure. We use a CSV file that lists the aws accounts that are in our scope. Our program iterates through each account in scope (using the appropriate IAM credentials configured for that account), then within each account, it iterates through each region. Just because all EC2 instances are *supposed to be* in US-EAST-1, we cannot assume that they are.

There are multiple data elements that can be obtained by the describe-instances command. Among other things, we look for two user-defined tags: “ManagedBy” and “KeepUntil.” A different tagging option is to use the ticket number to trace why the instance exists. The ec2_inventory.py script outputs an Excel spreadsheet that shows, the instance name, IP address, Region, Account, Age, in addition to these two tags. We are starting to terminate untagged instances as “unauthorized.

List your other “virtual devices” such as S3 buckets, security groups, VPC’s, elastic load balancers, etc. -- “Know Thyself”

Remember, that your security capabilities have to drive organizational change too. The EC2 Inventory can highlight untagged instances, but most organizations will have to be cajoled into the discipline of tagging their instances. They may not be ready for you to use a Lambda function to start terminating their untagged instances.



Capability Maturity Model

5 -Optimizing — Focus on process improvement

4 -Managed — Processes are measured and controlled

3 -Defined — Projects tailor their processes from the organizations development methodology

2 - Repeatable — Processes are characterized for specific projects and organization is often reactive

1 - Initial — Process is unpredictable, poorly controlled and reactive

Crawl – Walk – Run

SANS Technology Institute - Candidate for Master of Science Degree 7

There is a wealth of great information on the internet on AWS best practices, complete with code examples and toolsets. Some examples are Capital One's Cloud Custodian or Netflix's Security Monkey. Rather than compete or contrast with these approaches, I want to highlight them as mature examples of "capabilities as code." However, they are not **your** capabilities, until you have taken one of them and integrated it, tweaked it and are using it in a "business as usual" manner.

The ec2_inventory.py script that I just discussed is not all that fancy, and there are certainly more impressive examples of code on the internet. But I would argue that your code should reflect your level of process maturity and you may be fighting a losing battle if you are trying to skip levels in the Capability Maturity Model's phased progression.

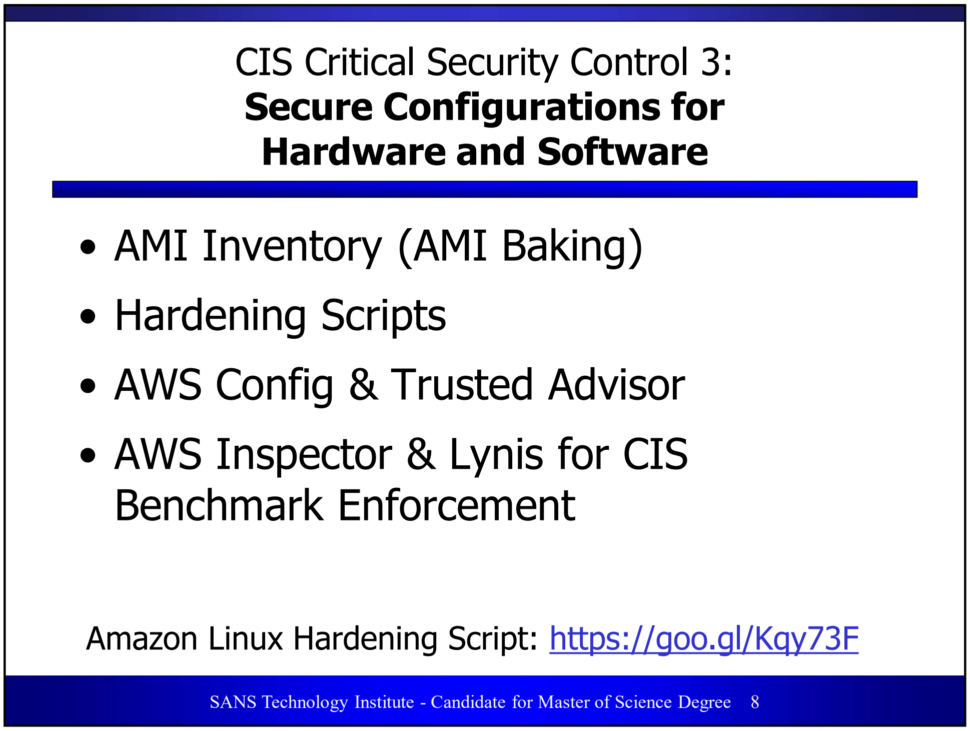
I suspect that the maturity of Cloud Custodian and Security Monkey closely mirrored their respective organizations' cloud security capabilities evolution to maturity.

So while I cover additional code examples, remember the two most important points I am articulating:

- Capabilities can be intentionally matured if documented and integrated into your "business as usual" processes, and
- The best way to document these capabilities is by doing it as continuously maintained program code

Reference:

Software Engineering Institute. (1994). *The capability maturity model: Guidelines for improving the software process*. Reading, Massachusetts: Addison-Wesley.



CIS Critical Security Control 3:
Secure Configurations for Hardware and Software

- AMI Inventory (AMI Baking)
- Hardening Scripts
- AWS Config & Trusted Advisor
- AWS Inspector & Lynis for CIS Benchmark Enforcement

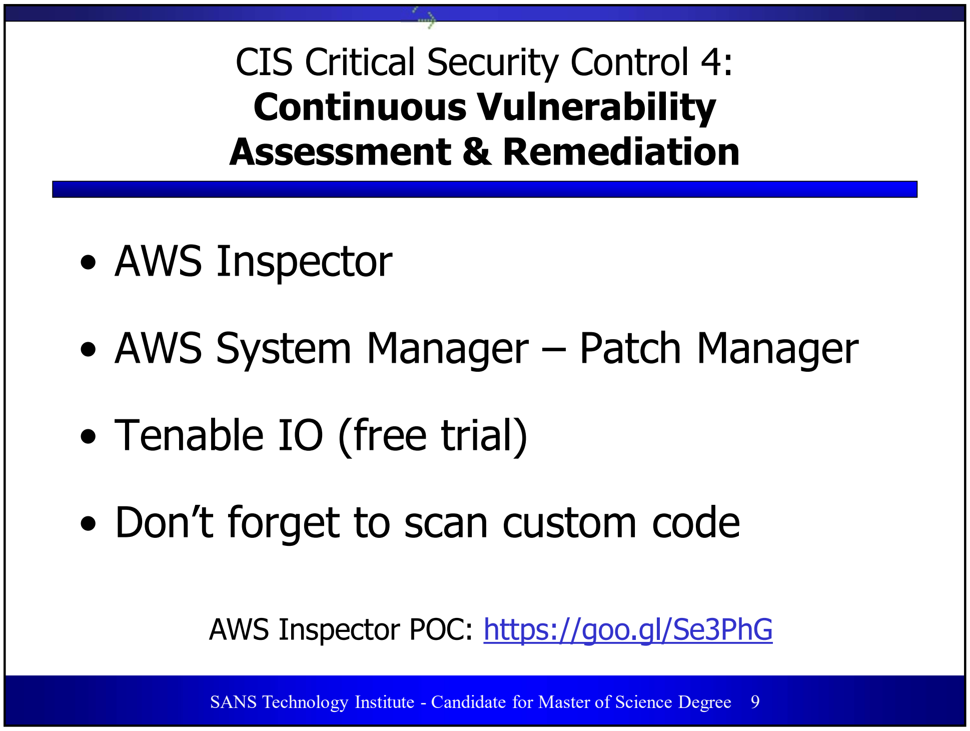
Amazon Linux Hardening Script: https://goo.gl/Kqy73F

I like to use the EC2 Inventory spreadsheet to look at the ages of my company's instances. Very old instances are probably fragile—that's why they have not been recycled. The whole idea of continuous integration (CI) is that you can rapidly provision any instance on a moment's notice. In fact, with Docker, you can get away from patching in place. Instead, just spawn new, patched instances.

AWS lets you create an image (called an AMI) from which to launch EC2 instances. This slide contains a link to a sample EC2 hardening script that you can use to bake a base AMI. Configure your policies so that users can only launch approved and hardened AMI's and not just anything from the AWS Marketplace.

I like to use Lynis to help me make sure that I have the system hardened to meet the CIS Benchmarks. AWS Config is a great tool to use to ensure that your entire AWS environment stays configured as designed, while AWS Trusted Advisor can verify that you meet basic security best practices in your usage of AWS.

I am providing a sample EC2 hardening script because many organizations that I have worked with do not have a defined set of requirements for system hardening, other than to say, "it must meet the CIS Benchmarks." I am arguing that you need to either start from scratch or take script like mine and make it your own. Define your organization's unique hardening requirements and bake those into the hardening script. Iteratively refine the script based on feedback from your risk assessments and incident handling.



CIS Critical Security Control 4: **Continuous Vulnerability Assessment & Remediation**

- AWS Inspector
- AWS System Manager – Patch Manager
- Tenable IO (free trial)
- Don’t forget to scan custom code

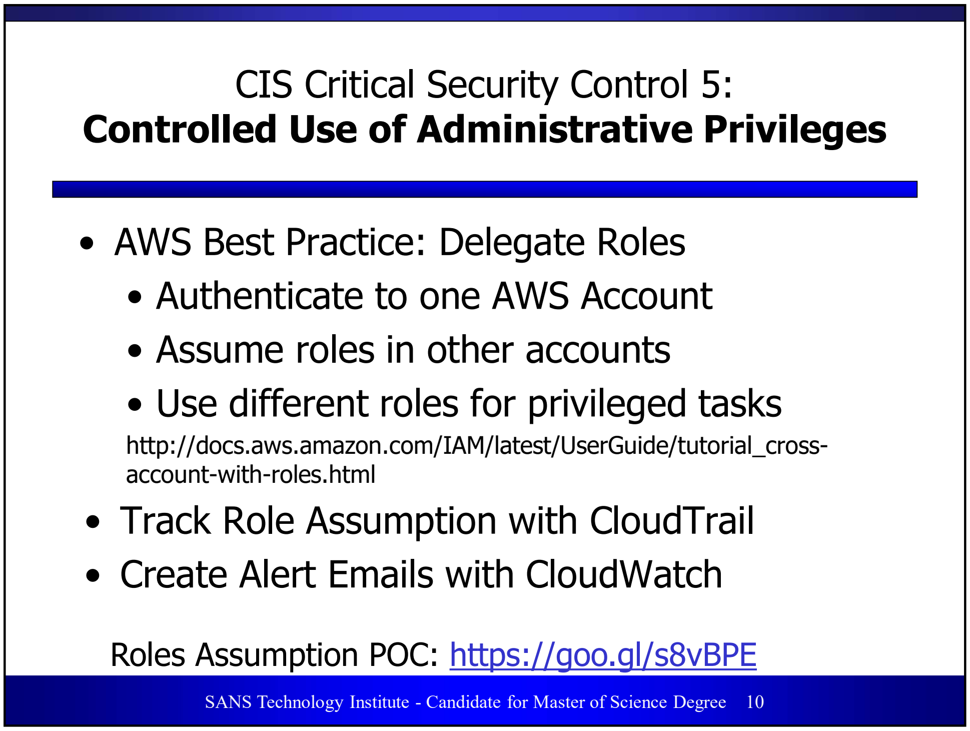
AWS Inspector POC: https://goo.gl/Se3PhG

AWS System Manager is free to use and AWS Inspector comes with a free trial. AWS System Manager has a feature called Patch Manager that will automate the deployment of patches. Tenable IO also has a free trial period.

There are also very good scanners for custom code, and these should also be leveraged and integrated into your CI toolchain. Play with this stuff and script it—to grow your capabilities

This POC Script takes the latest Amazon Linux AMI and launches an EC2 Instance with the AWS Inspector agent installed. It then proceeds to run a vulnerability scan. Last time I ran it, it generated 46 findings. The big takeway for me was:

DO NOT FAIL to run “yum update”



CIS Critical Security Control 5: **Controlled Use of Administrative Privileges**

- AWS Best Practice: Delegate Roles
  - Authenticate to one AWS Account
  - Assume roles in other accounts
  - Use different roles for privileged tasks

  http://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_cross-account-with-roles.html
- Track Role Assumption with CloudTrail
- Create Alert Emails with CloudWatch

Roles Assumption POC: https://goo.gl/s8vBPE

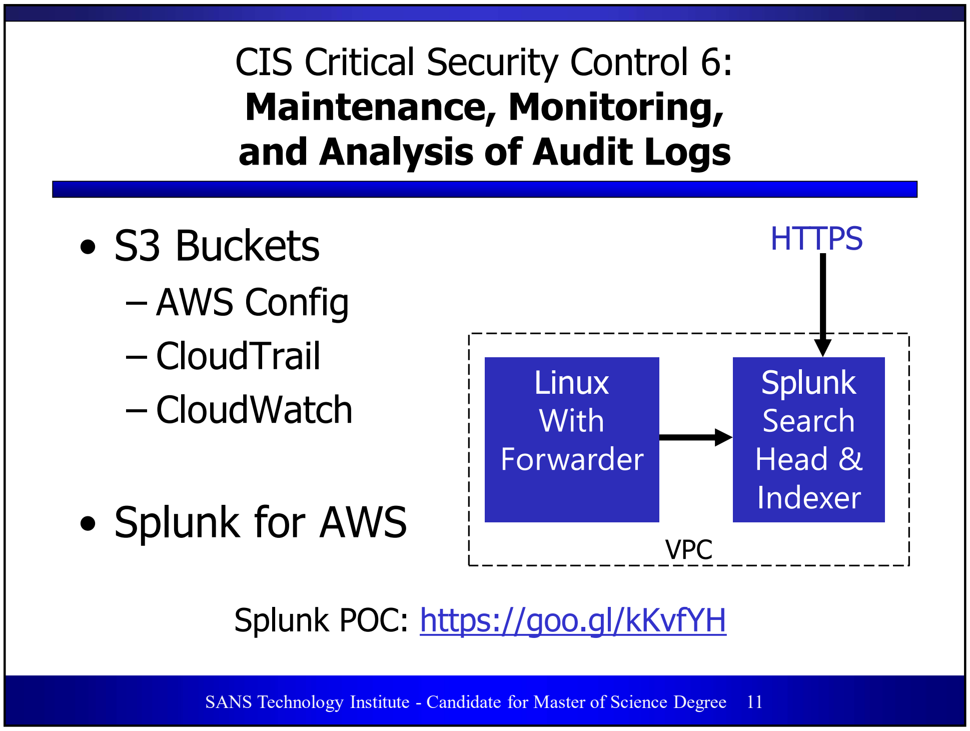
When your organization has multiple AWS accounts, Amazon recommends that you designate one account to be used for user authentication (with MFA) and then configure roles to be assumed in other accounts. This is extremely helpful, because different roles can be configured for different types of administrative tasks and then the role is only assumed when needed (like sudo/su).

You can even configure CloudWatch to send an email alert when certain privileged roles are assumed. Plus, all activity is captured in CloudTrail, assuming you have it enabled.

On this slide, I have provided code for a proof-of-concept that is based on the Amazon Tutorial.

The Amazon tutorials show how to do various tasks via the console. As I work through the examples such as this one, I translate the examples into a POC script using my own sandbox environment.

Amazon and the other Cloud providers have tried to make it very easy to play with their technologies…This is exactly why you should. You need to play with the cloud services in a safe sandbox, so that you know how to secure it. If you don't, others will play with it in your production environment.



CIS Critical Security Control 6:
Maintenance, Monitoring, and Analysis of Audit Logs

- S3 Buckets
  - AWS Config
  - CloudTrail
  - CloudWatch
- Splunk for AWS

HTTPS
Linux With Forwarder
Splunk Search Head & Indexer
VPC

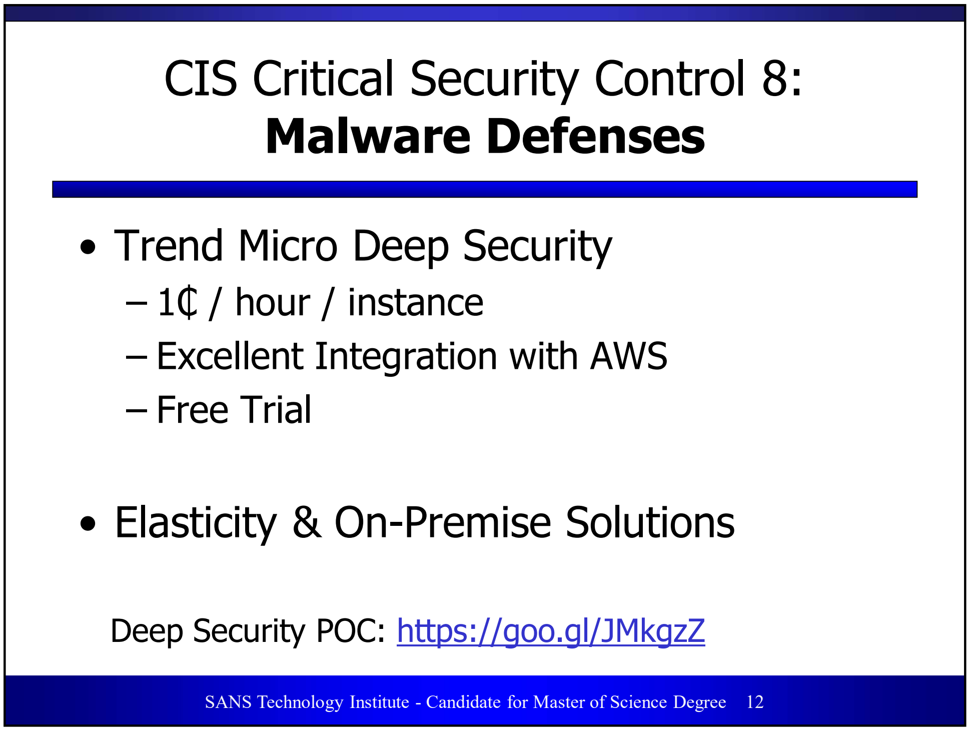
Splunk POC: https://goo.gl/kKvfYH

Amazon stores the logs generated by their various services (Config, CloudTrail, CloudWatch) in S3 buckets. These logs can be queried directly by AWS Athena or sucked into Splunk. Splunk has a *Splunk for AWS App* that facilitates the ingestion and analysis of these logs.

I like to use Splunk for a variety of ad hoc analysis, and even though Splunk is expensive in the enterprise, it has a free license if you stay under 500 MB per day. The great thing about the Cloud, is that you can using a script like mine, spin up a Splunk instance do the analysis, and tear it down when done—multiple times per day.

Remember that to fully meet CSC 6, you also have to analyze and retain the logs from the EC2 instances themselves and not just the AWS logs. Hence the need to install something like the Splunk forwarder.

For me personally, proficiency with Splunk is an important job skill. Therefore, I am constantly improving and documenting my Splunk searches.



CIS Critical Security Control 8: **Malware Defenses**

- Trend Micro Deep Security
  - 1₵ / hour / instance
  - Excellent Integration with AWS
  - Free Trial
- Elasticity & On-Premise Solutions

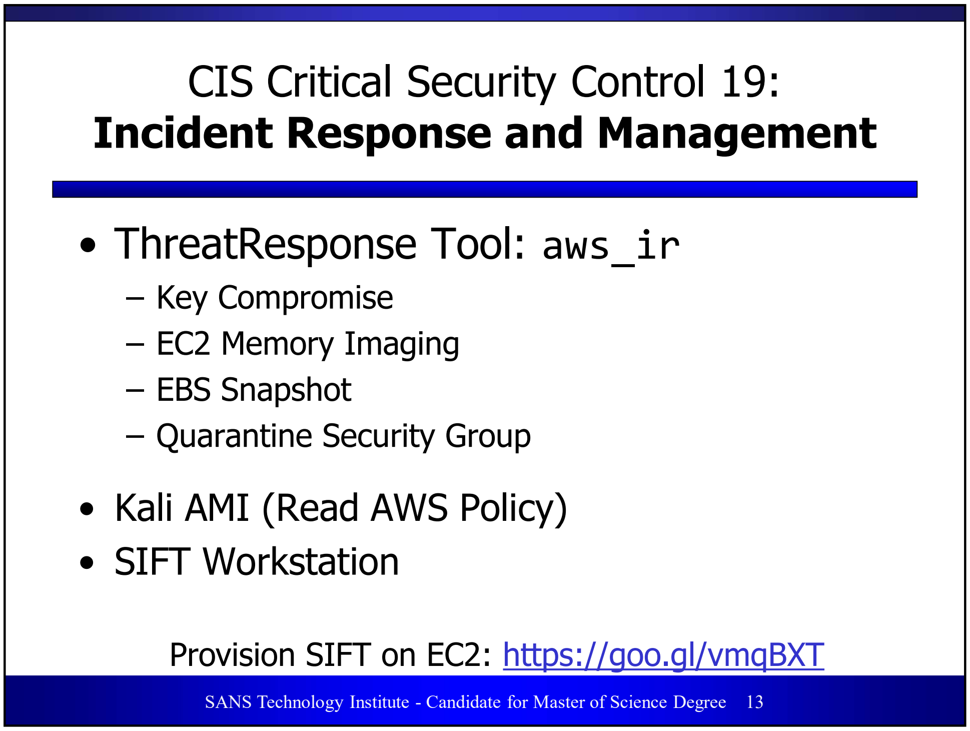
Deep Security POC: https://goo.gl/JMkgzZ

SANS Technology Institute - Candidate for Master of Science Degree 12

You certainly can use on-premise anti-malware solutions in the cloud, but if your EC2 instances are all short-lived, the anti-malware management portal may quickly become cluttered with agents that are no longer phoning home because the instances were terminated. This is where Trend Micro Deep Security shines. During the configuration process, Trend Micro is granted an IAM role so that it is aware of the configuration of your environment and updates the management portal accordingly.

I have no affiliation with Trend Micro. Deep Security is a solution that I and a few other students in the SANS Technology Institute evaluated as part of a group project for ISE 6100. I was impressed with its integration with AWS.

Smart vendors are following Amazon’s lead and offering free trial periods and low-cost elastic pricing so it is very inexpensive to learn their technology. Take advantage of these opportunities.



CIS Critical Security Control 19: **Incident Response and Management**

- ThreatResponse Tool: `aws_ir`
  - Key Compromise
  - EC2 Memory Imaging
  - EBS Snapshot
  - Quarantine Security Group
- Kali AMI (Read AWS Policy)
- SIFT Workstation

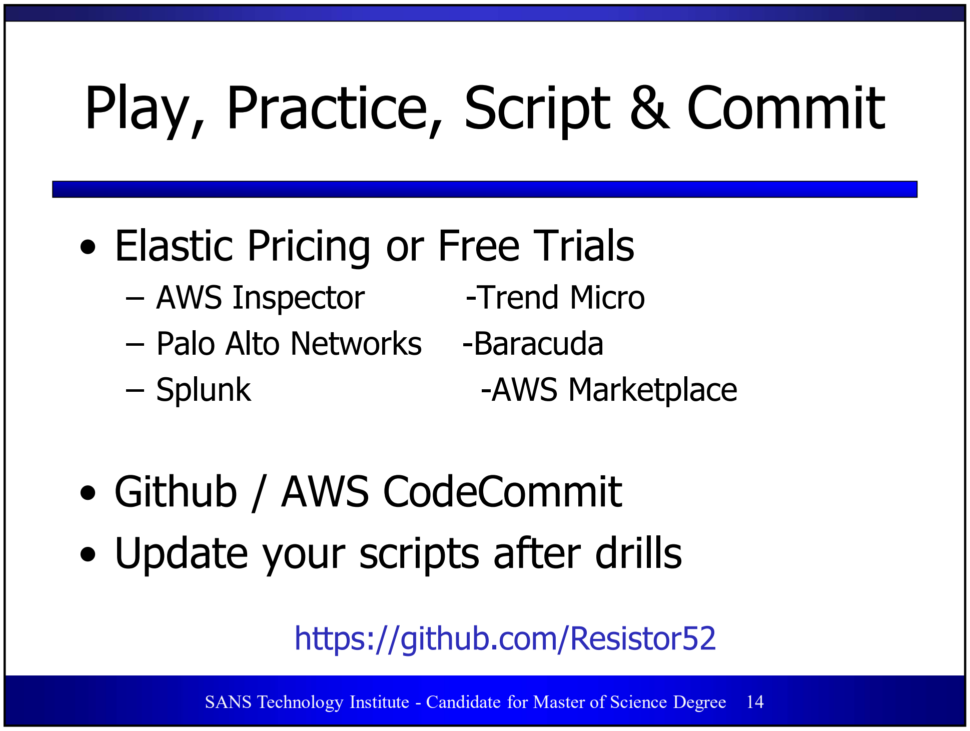
Provision SIFT on EC2: https://goo.gl/vmqBXT

There is an image for Kali in the AWS Marketplace. This makes it easy to spin up a system for an experiment. Just be sure to read the AWS Policy regarding pen testing. That said, the cloud is a great way to practice your pen testing skills. Spin up a kali instance and a second, target instance. You can even soften the target instance, just be sure to put it in a security group that can only be reached by the Kali instance. Script both the attack and the incident response to get a 360 degree view.

The team at ThreatResponse have made some very cool security tools for AWS. The aws_ir script automates the imaging of the EC2 instance memory and creates a forensic clone of the EBS virtual hard drive—all after placing the instance in a security group with a “deny all” ACL for both inbound and outbound traffic.

After attacking yourself, perform the forensics using a virtualized SIFT Workstation. I have provided a script to automate the setup of SIFT and instructions on how to attach the forensic evidence collected by aws_ir.

One of my main points is that be that code reuse is a great practice. After all, why reinvent the wheel? No matter one’s level of maturity, the best way to improve is to translate your capabilities into code. It is not enough to expect to use someone else’s code (for example, aws_ir) in a moment of crisis until you have worked with it and integrated it into your portfolio of security capabilities.



Play, Practice, Script & Commit

- Elastic Pricing or Free Trials
  - AWS Inspector -Trend Micro
  - Palo Alto Networks -Baracuda
  - Splunk -AWS Marketplace
- Github / AWS CodeCommit
- Update your scripts after drills

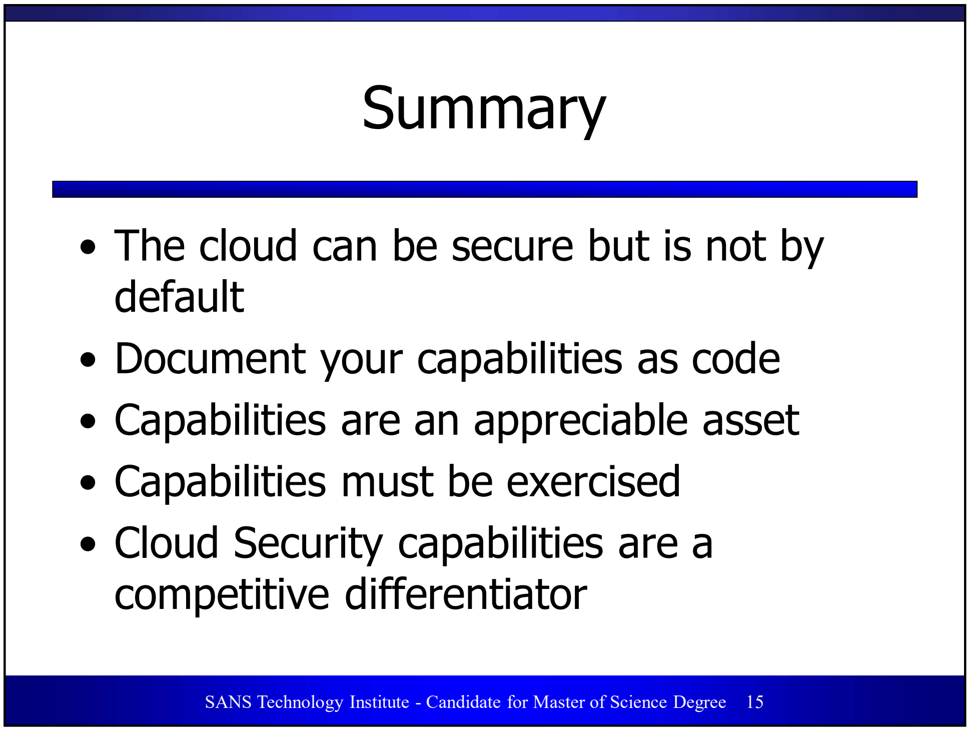
https://github.com/Resistor52

SANS Technology Institute - Candidate for Master of Science Degree 14

There are many security-related service offerings available in the AWS Marketplace. Typically these have a free trial period or have a minimal cost. One of the more expensive ones is PAN at $1.28/hour (after their free trial). But $15 for a day's worth of learning is a great value. Plus, you terminate the instance and the charges stop.

Maybe your code is too sensitive to put in Github's public repository? If so, use AWS CodeCommit. But, do commit them to a code repository of some sort. Take the time to learn how to do this, because the developers in your companies already are. You should understand their technologies and tools.

Take my scripts and make them better. Play with them and share with me what you learn by making a git pull request to my repository.



Summary

- The cloud can be secure but is not by default
- Document your capabilities as code
- Capabilities are an appreciable asset
- Capabilities must be exercised
- Cloud Security capabilities are a competitive differentiator

- The cloud can be secure but is not by default
- IP has a shelf Life but capabilities are an appreciable asset
- Document your security capabilities as code, use version control as you refine them.
- Knowing security best practices are not enough, they must be practiced
- Using someone else’s code is okay, but it needs to be fully integrated until it feels like your own
- Security capabilities ARE a competitive differentiator (for you and your business)