Major Pitfalls to Avoid in Performing Digital Forensics and Incident Response in AWS

And How!

Jonathon Poling

Managing Principal Consultant

Secureworks

~~About~~ Why Me?

- A little bit about me… ← blah blah blah
- Why Me? ← this though
  - Spent WAY TOO MUCH TIME…
    - Reading AWS Docs + Building AWS Cheat Sheets + Hoarding Community Tools + Developing Best Practices
  - AWS SME for Secureworks
  - Developed Secureworks' AWS IR Service Line
  - Help SMB through Fortune 10 Customers…
    - Intelligently Configure/Instrument Their Environments
    - Protect Their Infrastructure and Effectively Respond to Incidents
  - I've done enough dumb things to know how NOT to do them

Why Did I Put This Together?

After years of building Incident Response best practices and responding to a variety of compromises, I realized…

Every organization is doing at least one thing inefficiently and/or ineffectively in performing DFIR in AWS.

So, How Will This Help You?

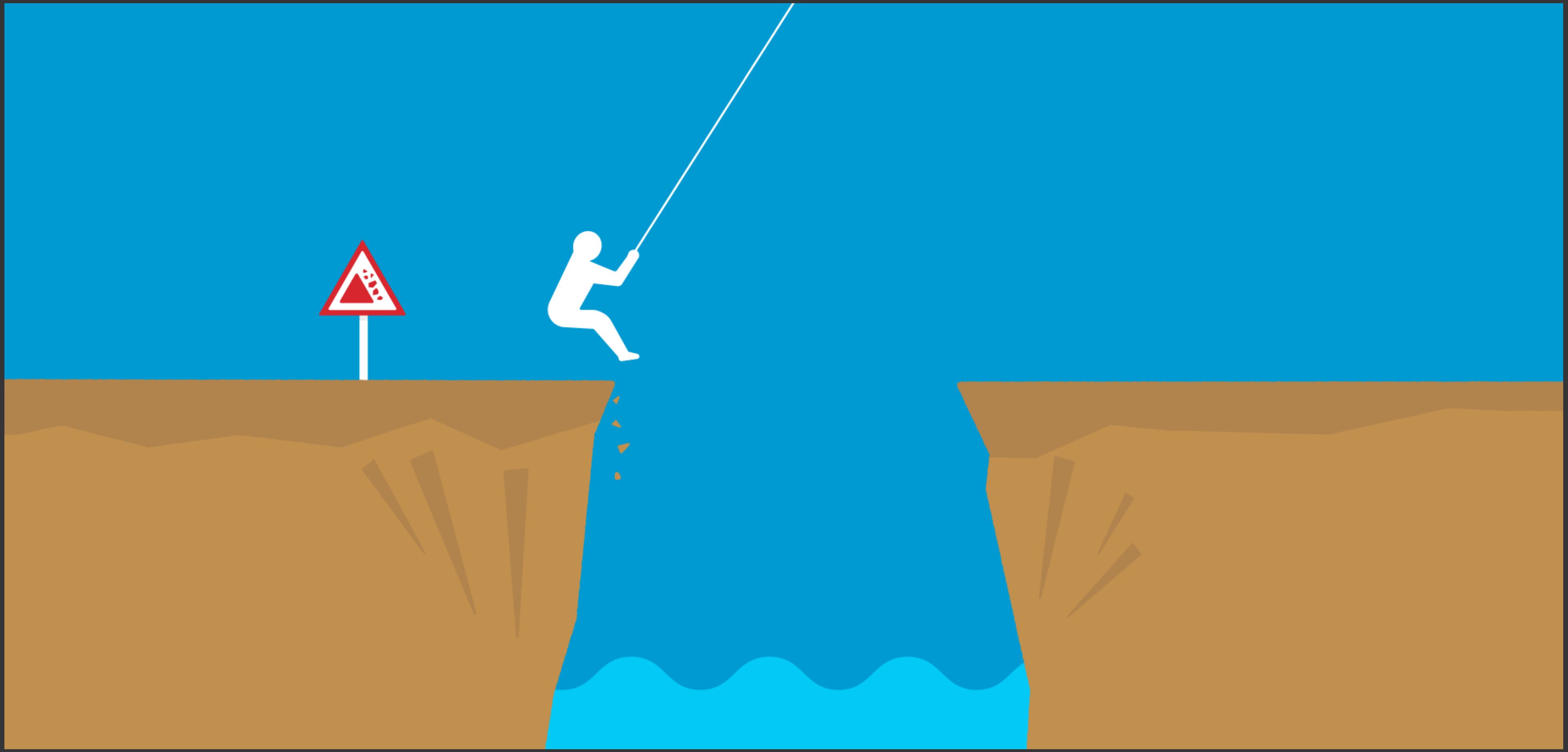
- In this talk, you will learn:
  - The major pitfalls
  - How and why they can occur
  - How to proactively prevent each of them

You will learn exactly how to avoid some of the biggest and most common mistakes that I / others (/ maybe you all?) have made, so that you can start getting better *right now*.

Major Pitfalls We'll Cover

- Insecure credential storage/usage
- Not knowing what to collect (and how)
- Improper log configuration/consolidation
- No Account segregation
- No system isolation/containment methodology
- No cloud-based lab
- Pulling data out of the cloud
- Ineffectively/inefficiently searching logs
- Not staying up to date on AWS

Here We Go…



Insecure Credential Storage/Usage

- People are searching and watching GitHub…
  - https://github.com/dxa4481/truffleHog
    - "Searches through git repositories for high entropy strings and secrets, digging deep into commit history"
  - https://github.com/eth0izzle/shhgit/
    - "Shhgit finds secrets and sensitive files across GitHub code and Gists committed in *near* real time by listening to the GitHub Events API."
  - https://github.com/duo-labs/secret-bridge
    - Duo's recent release, similar to ShhGit (uses GitHub Events API hooks)
- Attackers are looking in your Command Line History, ~/.aws/, and Documents folders for plaintext credentials

Insecure Credential Storage/Usage

So…

How can you protect against (accidental) exposure?

Insecure Credential Storage/Usage

- Proactive Protection
  - Implement local Key/Secrets Management
    - Practices
      - Don’t store credentials in plaintext anywhere!
      - Use/Assume Roles as often as possible
      - Implement and <u>*enforce*</u> (via IAM Policy):
        - MFA
        - Least Privilege Access
    - Tools (Examples)
      - Hashicorp Vault, AWS-Vault, Strongbox, etc.

Insecure Credential Storage/Usage

- Proactive Monitoring/Detection
  - GitHub built-in detection/notification
  - AWS built-in detection/notification
  - AWS GuardDuty
- Proactive Disabling
  - Script/automate the process of:
    - Identifying all activity associated with a given access key
    - Disabling a known compromised Access Key
    - Denying access for all temporary credentials issued before now

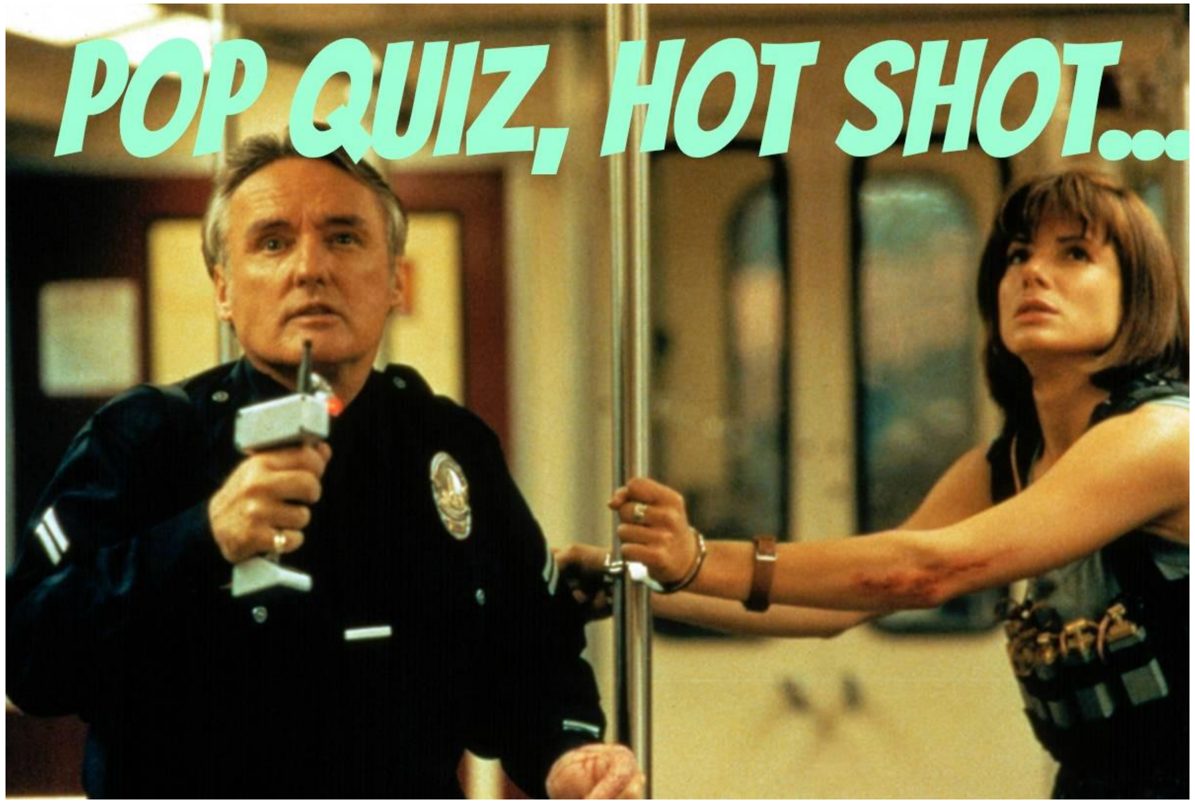
Not Knowing What to Collect (and How)

It's 3am. Your pager goes off. (Yes, the year is 1994)

You receive an alert that hundreds of resources are getting spun up across every region in your account.

What do you collect?

How do you collect it?



Not Knowing What to Collect (and How)

You may have thought (or assumed) that:

- It’s probably very similar to on-premises response
- It can’t be that hard to figure it out as you go
- Given the capabilities to easily start up almost anything within AWS, there’s likely something similar for response

But, let’s think through it now…

Not Knowing What to Collect (and How)

- How are you going to be able to reconstruct what happened?
- What logs are available to help you figure it all out?
- How can you figure out which account(s) were involved?
- How are you going to acquire systems for analysis?
- Do systems need to be shut down in order to image them?
- Does Snapshotting a system acquire a forensically sound image for analysis?
- Will your standard collection/acquisition procedures work?
    - *Are you sure*? Have you actually tested them (at scale)?

Not Knowing What to Collect (and How)

Example of Overestimating the “Ease” of response in AWS:

“Oh, well… we’ve kinda poked around the console and Events History seems to have everything, so we’ll just use that!”

FALSE

Not Knowing What to Collect (and How)

- Using Events History via Console and exporting to CSV MISSES INFORMATION!
  - Critical information for various actions (like Console Login) is completely omitted when you collect CloudTrail logs via Events History output to CSV.
- CloudTrail logs are best collected via the AWS CLI and/or direct copying from S3
  - Situation will determine which may be better/best
- Take the time to sit down with your AWS DFIR folks and map out what you need to collect (and how) before it's 3am and you're scrambling to do it all wrong.

Not Knowing What to Collect (and How)

- A few (not all) focus points for data collection:
  - AWS Data
    - Config – Resource Changes over Time
    - CloudTrail – Account Activity
    - CloudWatch Logs – Account/System Metrics and Stats
      - Also, host-level logging/stats if using AWS Systems Manager and sending host logs to CloudWatch
    - S3 Logs – Bucket-Level Events
      - Doesn’t log object-level events by default!
    - Trusted Advisor – Best Practice Monitoring/Rec’s
    - VPC Flow Logs – Network Activity

Not Knowing What to Collect (and How)

- A few (not all) focus points for data collection (Cont.):
  - System Data
    - Disk Images
      - Build efficient collection techniques (both manual and automated) via Snapshotting and Imaging to S3
    - Memory Images
      - Not straight forward at all
      - Spend time testing, especially if you’re running a *nix fleet

Improper Log Configuration/Consolidation

- Have you configured CloudTrail to collect to S3?
  - How are you going to investigate back further than 90 days?
- Are your CloudTrail logs centralized?
  - How are you going to aggregate and consolidate the logs from every single region into a singular place?
- Do you have system/host-level logging implemented?
  - How are you going to determine what happened on each host?

Improper Log Configuration/Consolidation

- Do you have S3 logging enabled?
  - Do you have Data Event logging enabled?
    - How are you going to see any Get/Put/Delete Object operations?
  - Do you have Server Access Logging enabled?
    - How are you going to see session information, HTTP operation types, and/or user access metadata (user-agent, IP, etc.)?
  - Do you have Versioning enabled?
    - How are you going to restore objects that accidentally (or ransomware-ily) got deleted?

Improper Log Configuration/Consolidation

- Do you have VPC Flow Logs and/or DNS monitoring enabled?
  - If not, how are you going to identify what network connections occurred around the time of compromise?
  - If so, how are you managing and storing the metric ton of data for efficient access/searching later?

Improper Log Configuration/Consolidation

- These all seemed so easy to configure with a single click
- Log configuration and consolidation takes purposeful research, implementation, and testing.
- So, great we all get it. But…

How can I make sure I'm doing all these things (and more)??

Check my Slide Notes ☺

No Account Segregation

Q: If you suspect compromise of a given account, should you be relying on that account to store your investigation data?

No Account Segregation

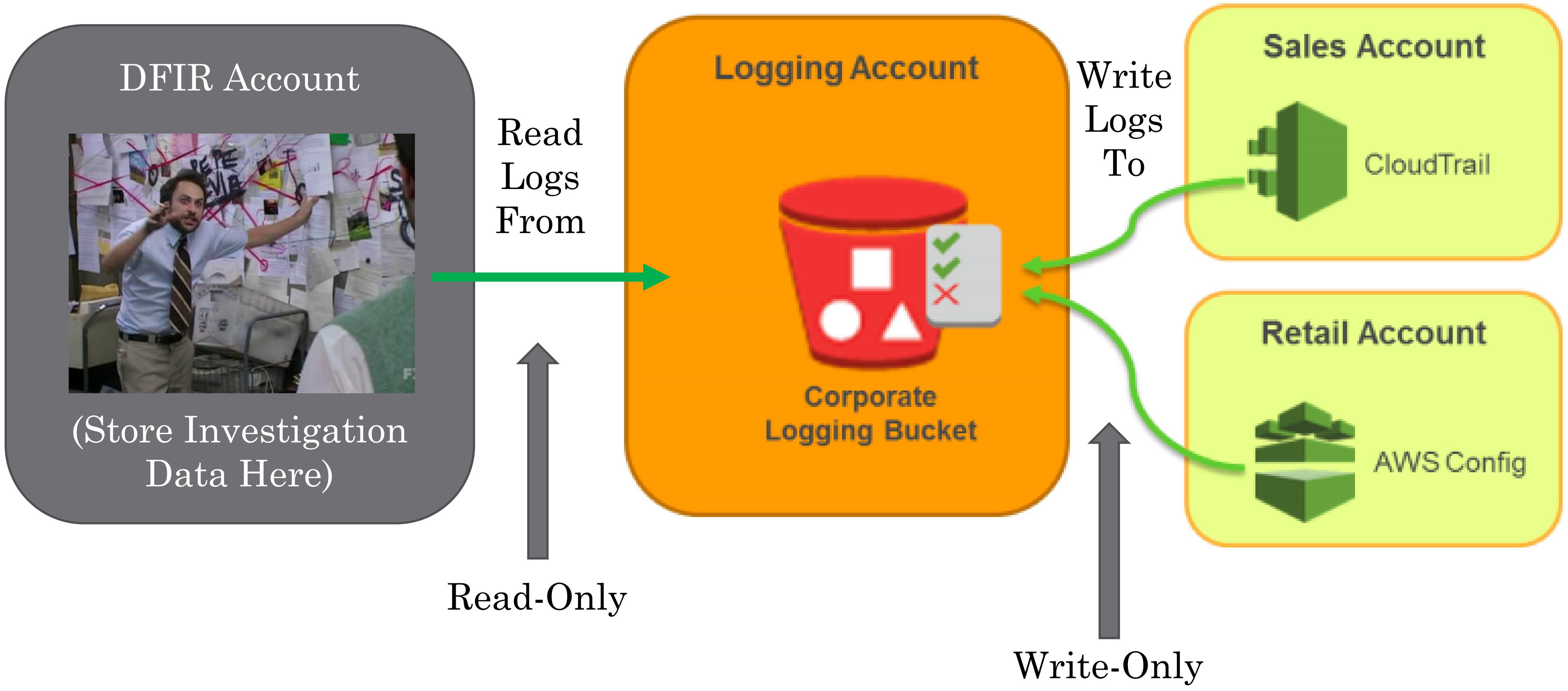
A: Account Segregation is not only critical to limiting compromise, but also for properly accessing, storing, and protecting investigation data.

A multi-account strategy AHEAD OF TIME is a ***must***.

So, how should you structure your account(s)?

No Account Segregation

Below is a good starting point, building on a sample from AWS' Multiple Account Security Strategy Guide.



No Account Segregation

- Take some time to (re)design and (re)architect your accounts NOW so that you aren’t paying for it all later (amidst a compromise)
- Yes, it may be extremely arduous and time-consuming
- Security design is “pay now or pay later” strategy
- The efforts now will pay off ten-fold in the future

No Account Segregation

Don’t be the company whose logs were all deleted by an active attacker.

No System Isolation/Containment Methodology

You just received notice that an Instance is compromised.

Now what?

A. Terminate the Instance and Re-Launch New One?
B. Disable the Network Adapter?
C. Run an A/V Scan on the Endpoint and Wait for Results?
D. Shut it down and take a Snapshot?
E. Delete all the Ingress/Egress Security Group rules?
F. Move the Instance to another VPC?

No System Isolation/Containment Methodology

You just received notice that an Instance is compromised.

Now what?

A. ~~Terminate the Instance and Re-Launch New One?~~

B. ~~Disable the Network Adapter?~~

C. ~~Run an A/V Scan on the Endpoint and Wait for Results?~~

D. Shut it down and take a Snapshot? (Almost Correct – Still loses active network connections and memory)

E. ~~Delete all the Ingress/Egress Security Group rules?~~

F. Move the Instance to another VPC? (Works but only with purposeful setup ahead of time)

No System Isolation/Containment Methodology

So, how do we best prepare for isolation / containment of compromised Instances?

Here is an example setup to get you thinking…

No System Isolation/Containment Methodology

Isolation / Containment Setup (High Level)

- Create Separate "Isolation" Account
  - Ideally, configure all of this in your "Forensics" Account previously discussed
- Create Two Subnets in one VPC
  - A "Compromised" subnet for the compromised Instances
  - An "Analysis" subnet for the forensic Analysis Instances

No System Isolation/Containment Methodology

Isolation / Containment Setup (High Level) – Cont.

- Create (At least) Two Security Groups in the VPC
  - One SG that allows all egress
    - Attaches to Analysis Instances (and/or compromised Instances to allow for network monitoring)
  - One SG that allows only ingress from the analysis Subnet
    - Attaches to compromised Instances (for full isolation)
- Enable Flow Logging on the VPC
  - Or just on the "Compromised" Subnet (your call)

No System Isolation/Containment Methodology

Isolation / Containment Setup (High Level) – Cont.

- Create an S3 Bucket for storing data/artifacts from compromised Instances
  - Enable Object-Level Logging for the S3 Bucket (or all Buckets)
  - Enable Data Encryption, Versioning, and Object Lock (optional)
- Create IAM Role for Compromised Instances
  - Allow the Instance to write (S3:PutOjbect) to pre-configured S3 Bucket
  - Deny Access to any/all other Resources and Actions

No System Isolation/Containment Methodology

Isolation / Containment Setup (High Level) – Cont.

- Create AMI from Instance
  - Can also just take Snapshot(s) of system
- Share AMI (or Snapshots) with "Isolation" (or "Forensics") Account
- Create new Instance from AMI (or Snapshots) within the "Compromised" Subnet
  - Attach previously created IAM Role for compromised Instances
- Monitor the Instance and collect additional data while you perform forensic analysis

No System Isolation/Containment Methodology

What about just using an "Isolation" Security Group that you attach to the compromised Instance?

<u>Pros</u>

- Quick and easy isolation of an Instance

<u>Cons</u>

- Must know list of "safe" IP(s) of system(s) ahead of time from which you'll allow remote connections
- Requires Security Group pre-configured for each VPC in each Region
- Instance still operating in a possibly compromised account

No System Isolation/Containment Methodology

What about just moving the compromised Instance to a separate "Isolation" Subnet/AZ/VPC in the same account?

Pros

- Can properly isolate the Instance to prevent further compromised/propagation across the account

Cons

- Requires shutting the machine down, creating AMI, and re-launching in the new Subnet/AZ/VPC

If we're already doing this, why not just perform the best practice and move it to a separate, dedicated Account instrumented for proper isolation, monitoring, and analysis?

No System Isolation/Containment Methodology

- One size doesn’t necessarily fit all here
- There are a variety of ingenious ways to do this
- Ideal end state is to preserve and collect as much data as possible while limiting the impact of compromise
- But, as with most things Security, it’s a balance of:
  - Maintaining Continuity of Business Operations
  - Mitigating Risk
  - Minimizing Downtime, Damage, and Propagation
  - Maximizing Value of Performing Investigations

No Cloud-Based Lab

- It’s critical to replicate (even better, improve upon) your existing on-premises tooling and capabilities within AWS
- It’s not always straight forward…
  - Licensing
    - How does (or can) it scale?
    - Are there issues with the software running virtually?
    - Does it require a physical dongle?
  - Data Transfer/Access
    - Do any on-premises systems/tools/shares need access into the lab?
    - How are you planning to transfer TB’s (or PB’s) of information in and out?

No Cloud-Based Lab

- It’s not always straight forward… (Cont.)
  - Remote Access
    - How will users remotely connect (AND RELIABLY STAY CONNECTED) into the lab?
    - How will access to data/tools be federated?
  - Chain of Custody
    - How will you store and prevent unauthorized access to data/evidence (with proof)?
  - Maintaining Images
    - Who will be responsible (and how) for maintaining forensic analysis base images/tooling?
- This pitfall also tends to lead to the next one…

Pulling Data Out of the Cloud

Are you downloading data from the cloud to analyze it locally?

Pulling Data Out of the Cloud

STOP IT RIGHT NOW!

- Pulling data out of the Cloud:
  - Introduces multiple throughput bottlenecks
    - Initially transferring the data to S3
    - Downloading data from S3 to local lab
    - (Re)ingesting data into local tools for analysis
  - Adds unnecessary delay to investigations
  - Drastically diminishes the value of the Cloud
- And, odds are, if you're pulling data out of the Cloud for local analysis, you may be hitting the next pitfall…

Ineffectively/Inefficiently Searching Logs

How are you analyzing AWS logs?

- Individually downloading them, aggregating them, and feeding them into an on-premises tool to ingest/search?
- Using a third-party tool to constantly ingest all your logs in the Cloud and then hoping the search interface works when you need it?
- Copying them all to an Instance (or local lab) and then command-line grep'ing (or jq'ing) them?
  - This one is actually valid for smaller log sets
- ….?

Ineffectively/Inefficiently Searching Logs

HARNESS THE POWER OF THE CLOUD FOR ANALYSIS

Some examples of how to harness native Cloud utilities…

- EC2
  - Tons of EC2 Instances dedicated to specific tasks
    - Compute/Memory/Storage-optimized
- Athena
  - Athena is *extremely powerful* for log analysis
  - Query CloudTrail, VPC Flow Logs, and other S3-based logs
- CloudWatch Logs Insights
  - "It plows through massive logs in seconds, and gives you fast, interactive queries and visualizations."

Ineffectively/Inefficiently Searching Logs

- Or, even just quickly stand up an ELK stack for a variety of log analysis
- … the list goes on
- Native Cloud data analysis should be your default method of operation
- Leverage every native tool feasible (obviously, balancing the cost/benefit), and augment with additional tools as necessary

Not Staying Up to Date on AWS

- Static knowledge is not enough!
- You must keep up with the ever-changing landscape of services, changes, and improvements!
- For example, did you know*:
  - You can now enable additional metadata for VPC Flow logs to help you better investigate flows (such as VPC-ID, Subnet-ID, Instance-ID, TCP Flags, and so on)
  - There's already a PoC for abusing VPC Traffic Mirroring
  - AWS Systems Manager can now execute Ansible playbooks directly from GitHub or S3
  - Snap recently authored a blog post on how to create a Cross-Account Support Case Dashboard to aggregate Support Cases across accounts
  - AWS was just acquired by Microsoft

*One of these may be completely false

Not Staying Up to Date on AWS

- So, how can I/we stay on top of all of this stuff when it changes daily?
  - Follow people who live/eat/breathe/break AWS Cloud on Twitter
    - Corey Quinn (@QuinnyPig)
    - Scott Piper (@0xdabbad00)
    - Will Bengtson (@__muscles)
    - Forrest Brazeal (@forrestbrazeal)
    - Spencer Gietzen (@SpenGietz)
    - Rhino Security Labs (@RhinoSecurity)
    - Toni de la Fuente (@ToniBlyx)
    - Andreas Wittig (@andreaswittig)
    - Jerry Hargrove (@awsgeek)
    - Andres Riancho (@AndresRiancho)

…and many more

Not Staying Up to Date on AWS

- So, how can I/we stay on top of all of this stuff when it changes daily? (Cont.)
  - Follow the AWS Security blog (at minimum, if not also the base blogs page for all services)

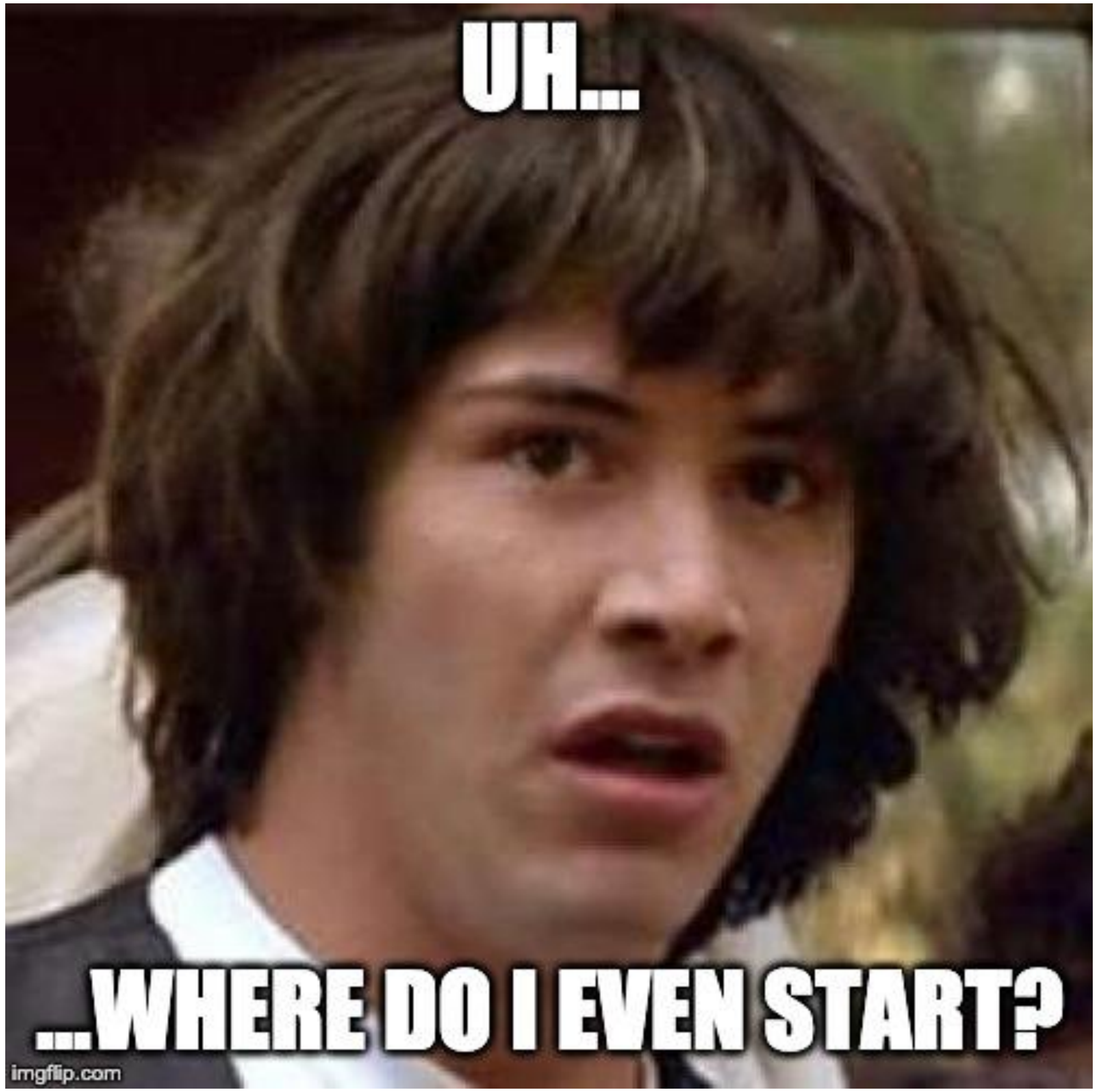
    https://aws.amazon.com/blogs/security/
  - Subscribe to QuinnyPig's "Last Week in AWS" newsletter

    FYI - With much knowledge comes much snark. You have been warned.

    https://www.lastweekinaws.com/
  - Dedicate time for your people to continually test/research/improve

Dedicating someone (or ideally multiple people) to being deep SME's on the AWS services pertinent to DFIR is CRITICAL!

All That Said…



The End

Email: jpoling@secureworks.com
Twitter: @JPoForenso
Blog: https://www.ponderthebits.com