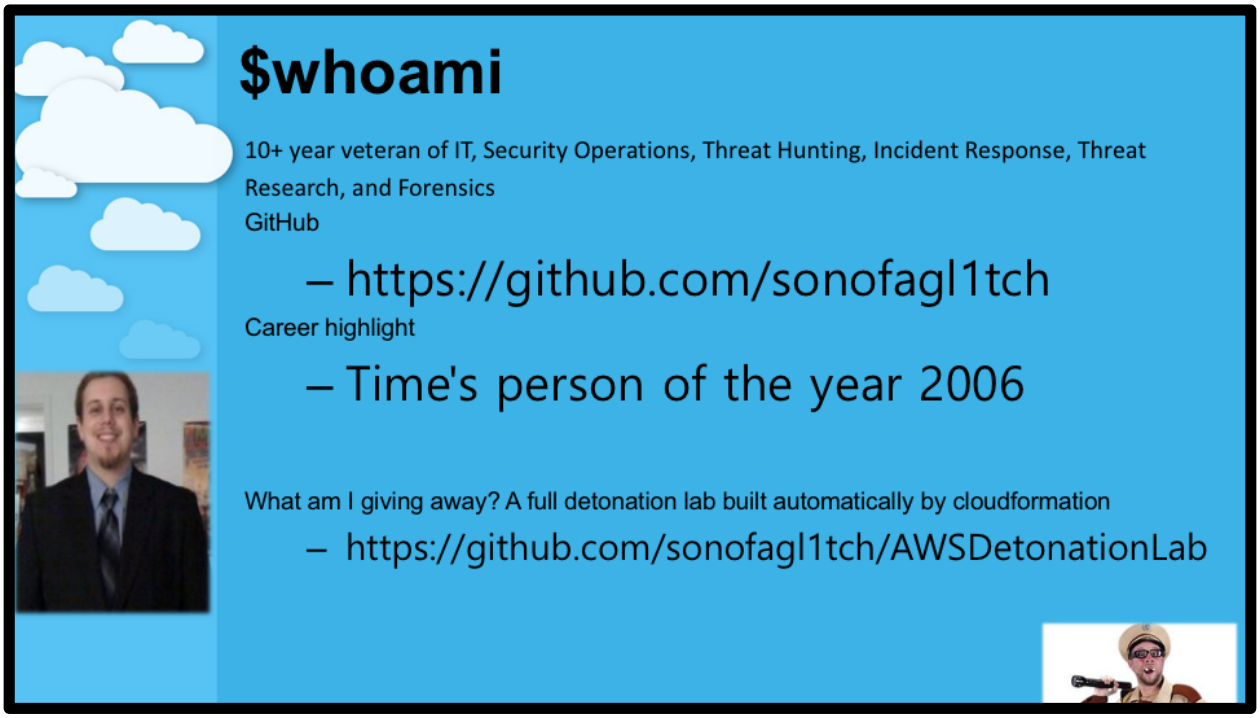
Hello everyone! Let's get started.

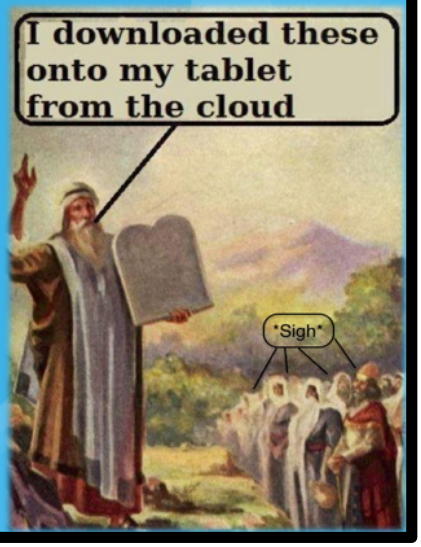


My name is Ryan Nolette and I have been a security practitioner in various capacities for the last 15 or so years.

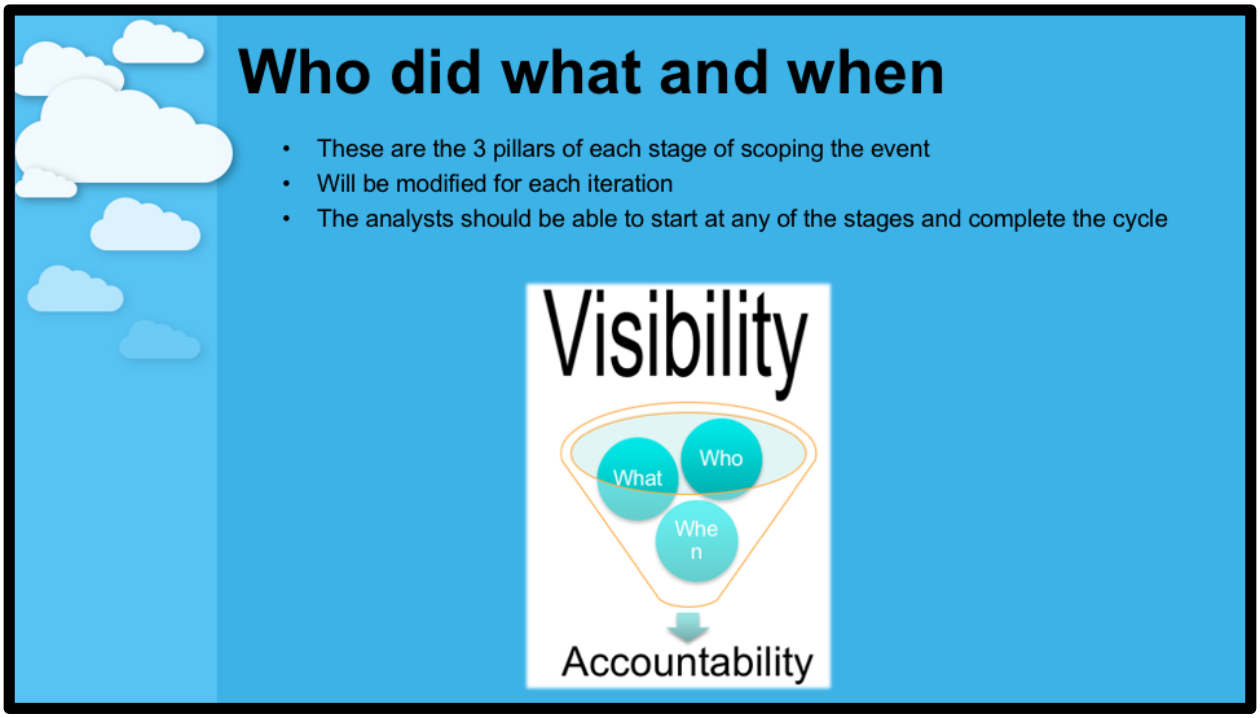
Today I will be walking you through a practical example of detecting and investigating an attack in the cloud as well as giving away a working detonation lab filled with opensource software that you can use to run preconfigured attacks and learn what logging services have visibility into different attacks. Hopefully this will help you learn how these different log sources work together to provide a full picture of an event.

Agenda

- Overview
  - Who did what and when?
  - Common Techniques
  - Common Visibility Tools and Their AWS Equivalent
- Increasing visibility until you have accountability
  - Common Logging
    - Authentication
    - Endpoint
    - Network
    - Vulnerabilities
    - Configuration
- End to end example
  - Detonation Lab
  - Logging Pipelines and Services
- Finding What Matters



In this presentation I am going to cover a brief overview of a few AWS logging sources and how they compare to the most common logging sources in a typical enterprise network and an end to end example using the detonation lab that I am giving away



As part of any investigation, I strive to answer 3 questions, who did what and when. This is a an iterative process and will be repeated for every step in the investigation. I should be able to start with knowledge of any of the 3 and find the other 2.

For example, I start my investigation with an IP address. I have the who. Next I need to find the time scope of when It was used. That is the when. Then I find that this IP address attempted to ssh into my instance 300 times in 10 minutes. This is the what. With these 3 questions answered I can reliably say that this external attacker tried to brute force ssh access into my instance.

I know this is a very simplistic example but I want to very clearly explain these tenants

To expand on these tenants of investigation, I break them into 2 buckets. Visibility and accountability. If you cannot see something, you can not account for it. You must keep increasing visibility until you can account for each event in your environment with 3 simple questions, who did what and when.

Now I know this end goal is next to impossible but that is part of the iterative process, security is a journey, not an end goal.

Common Visibility Tools and Their AWS Equivalent

Traditional Tool	AWS equivalent
IDS/IPS	guardDuty
DLP	Macie
EDR	Cloudwatch + osquery, GRR
Netflow	Cloudwatch + VPCFlow
DNS	Cloudwatch + Route53
Access and authentication auditing	CloudTrail
Active Directory	Directory Service
Identity Management	IAM
Single Sign On	AWS SSO
Vulnerability scanner	Inspector
Configuration Management	AWS config
Logging	Cloudwatch + Firehose + Lambda

So how do I get visibility in the cloud? Simply put, the same way you do in your on prem network. Why complicate things? These AWS tools are the servicewise cloud equivalent of their common on prem twinsies.

Who here has an IPS?
Who here has a Vulnerability scanner?
Who here collects netflow or other network traffic logs?

If you want to know the equivalent to other traditional controls that I haven't mentioned here, find me after the presentation and I will help you find what you want. I didn't have enough room here to mention them all.

Increasing visibility until you have accountability

The process of asking who did what and when and increasing logging and controls until you can answer those questions for every scenario you can think of.

OS hardening

- CIS guidelines audit and hardening scripts.
- Additional logging and hardening scripts created by experience over time.

Logging

- Common logging like auth logs and process creation etc

Authentication

- /var/log/secure
- IAM logs
- IAM roles
- IAM policies

Endpoint

- EDR
- HIDS
- Cloudwatch
- GuardDuty

Network

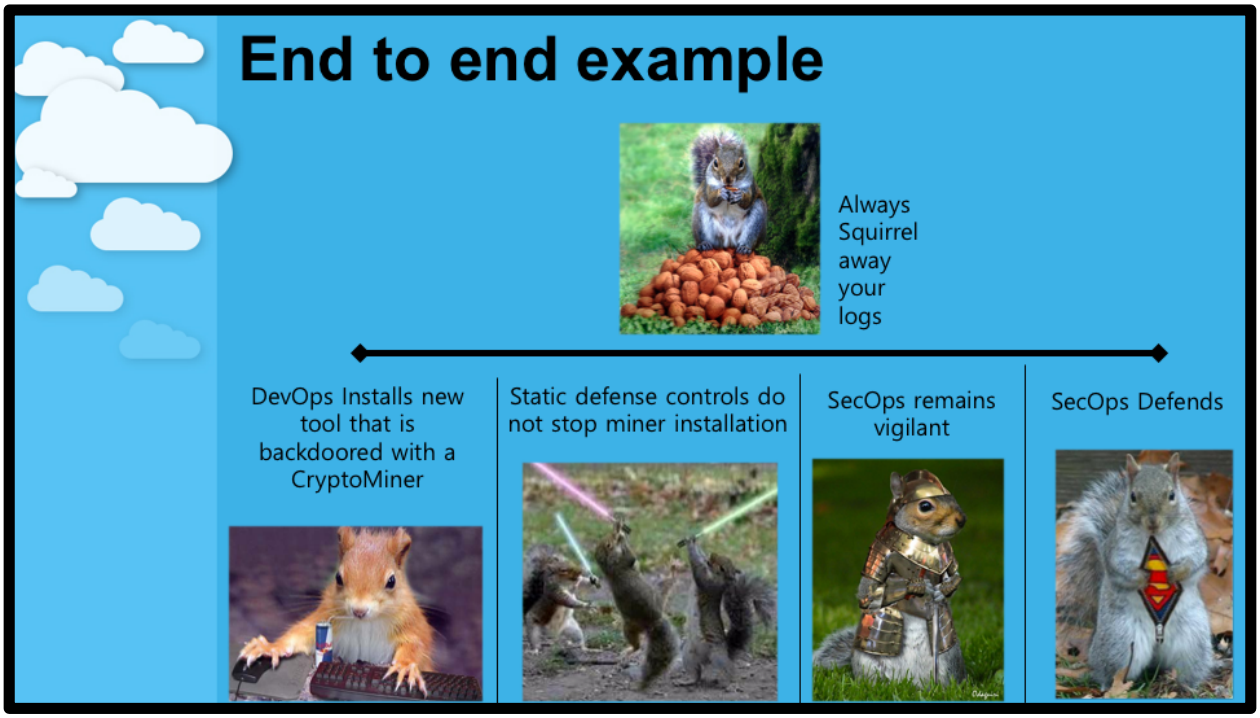
- IDS
- Netstat
- Tcpdump
- Vpc flow logs
- Dns route 53 logs

Vulnerabilities/ Configuration

- Generic vuln scanner
- Inspector
- NVD/CVE usage
- Aws config
- OS hardening
- Applications config

Expanding on our list of equivalent tools I want to show few options for each of the above categories

Take a few ideas, go nuts ☺

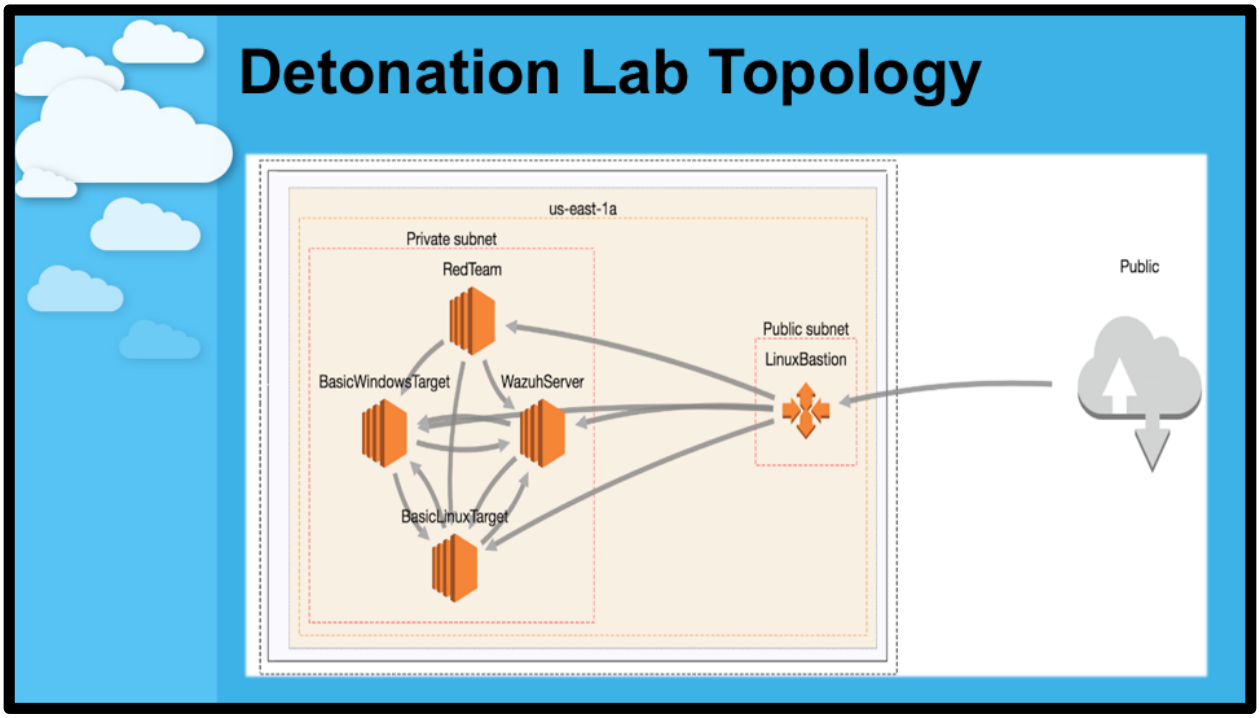


Now let's talk about some practical application of what I have said so far.

The scenario is that a devops engineer has installed a new tool they needed for monitoring their application but accidently installs an RPM that includes a cryptominer.

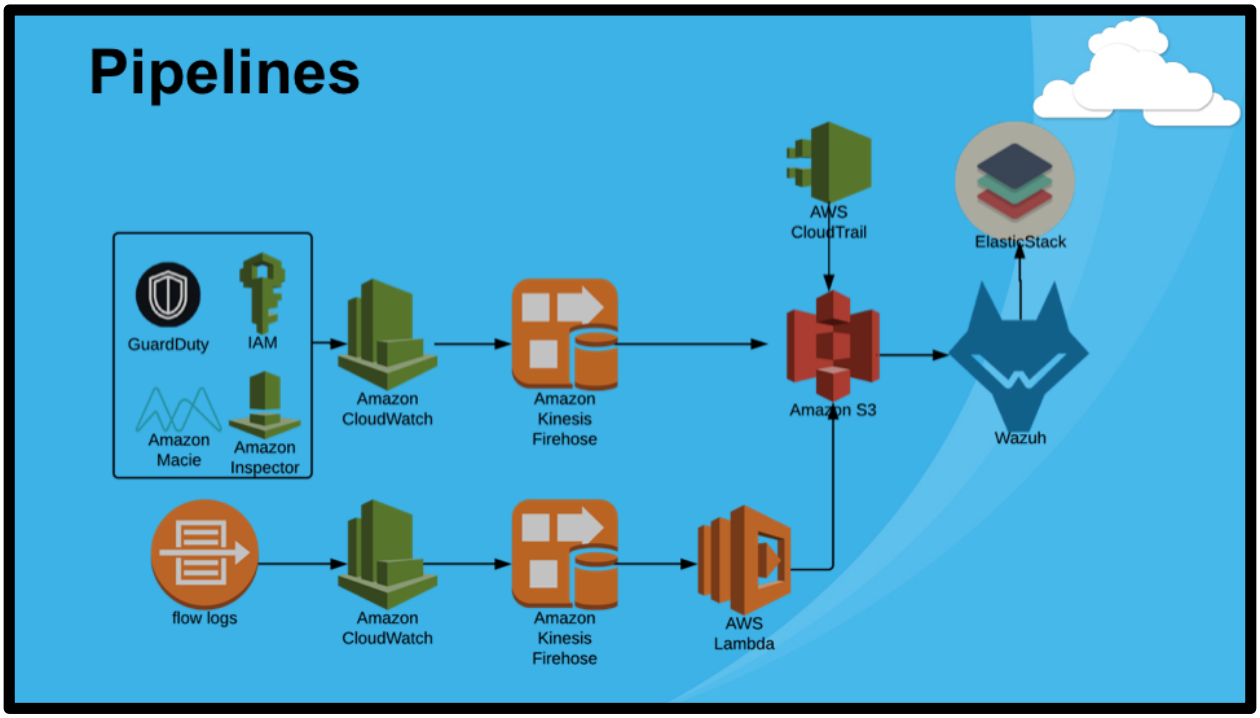
The endpoint doesn't have any whitelisting software on it and the rpm meets all conditions for installing an application from an RPM.

Secops notices a spike in outbound traffic to china and investigates.



Now before we go into the investigation, let's talk about the lab we are working in.

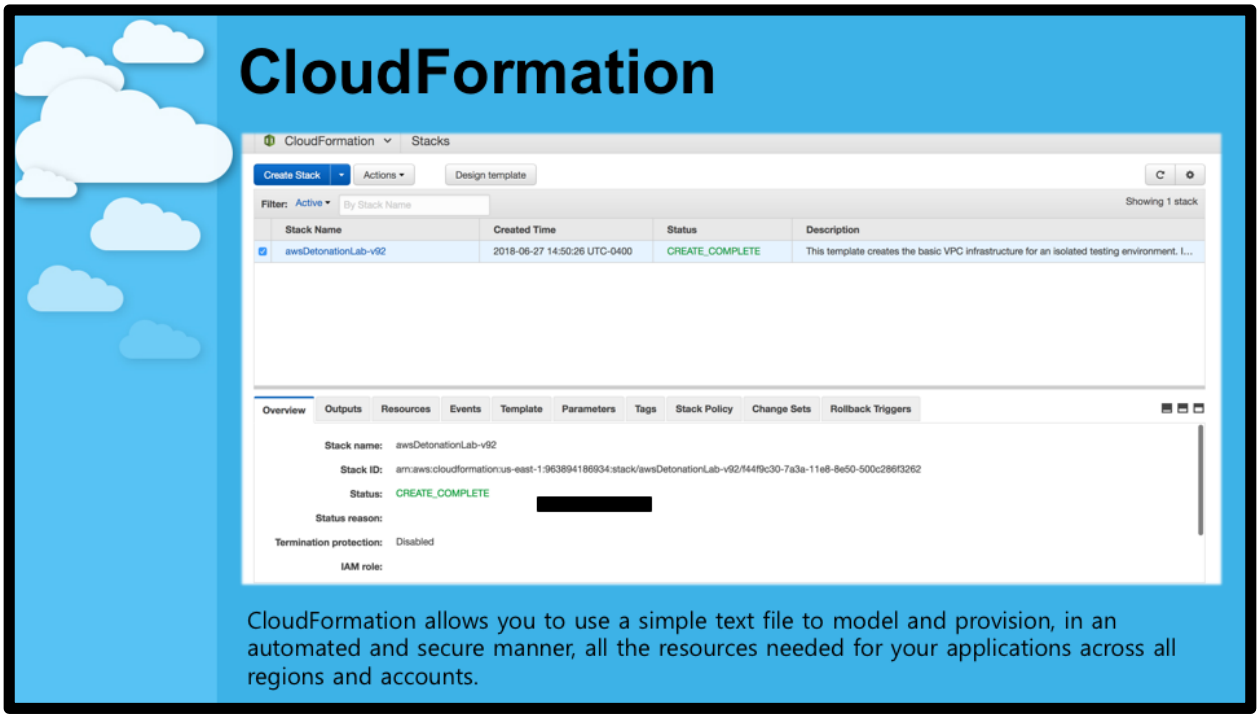
The topology of the detonation lab is 4 hosts (3 linux and 1 windows) behind a bastion host all within their own VPC.



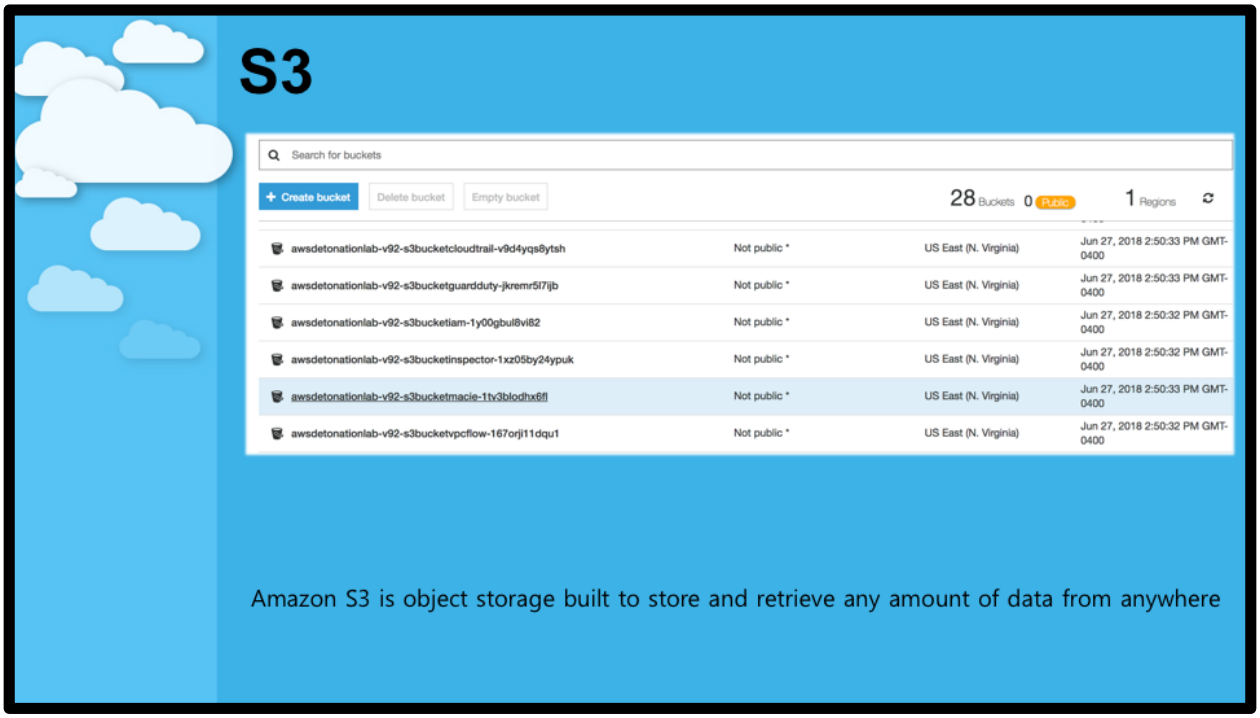
To get logs from various services I am using 3 pipelines. The first pipeline allows for a service to generate logs, trigger an event rule in cloudwatch, sends the logs to kinesis firehose, and writes the logs to an s3 bucket. These logs saved to s3 are then read by a wodle module on the wazuh server and ingested into Kibana.

The second pipelines mimics the first but with the added feature of using lambda to enhance the logs with metadata before writing to s3.

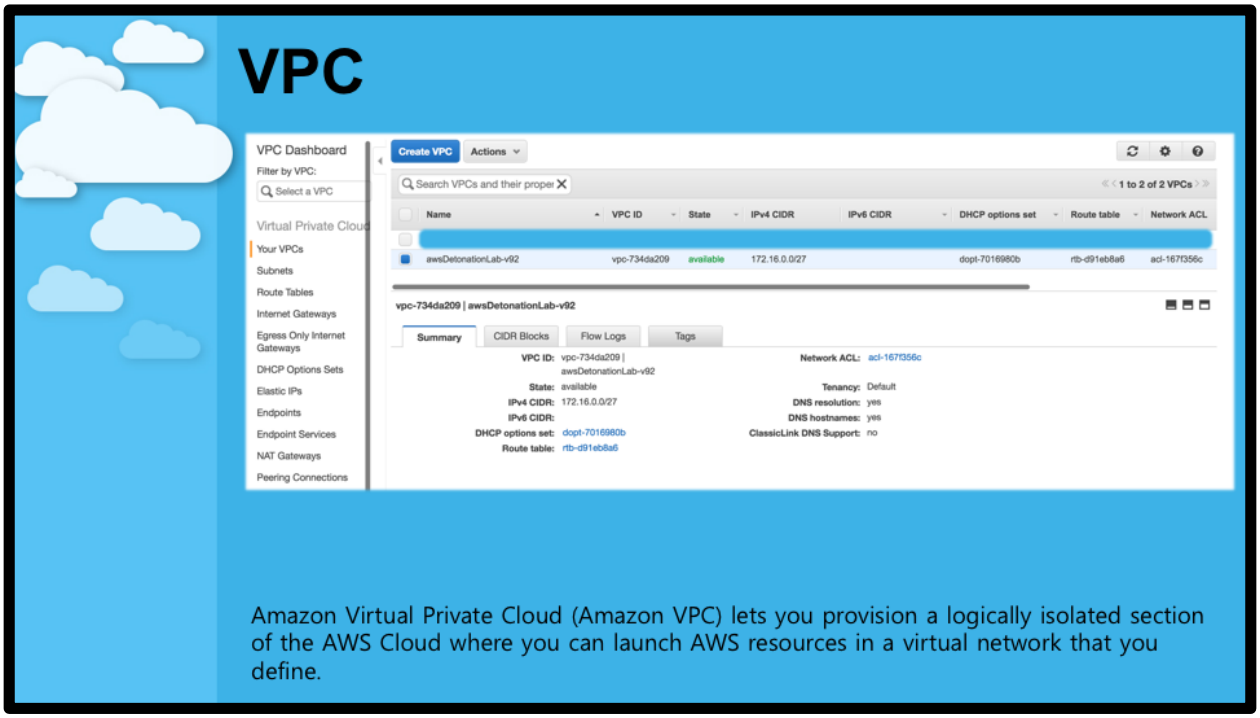
The third pipeline is for the EDR logs which are sent directly to wazuh from the agents.



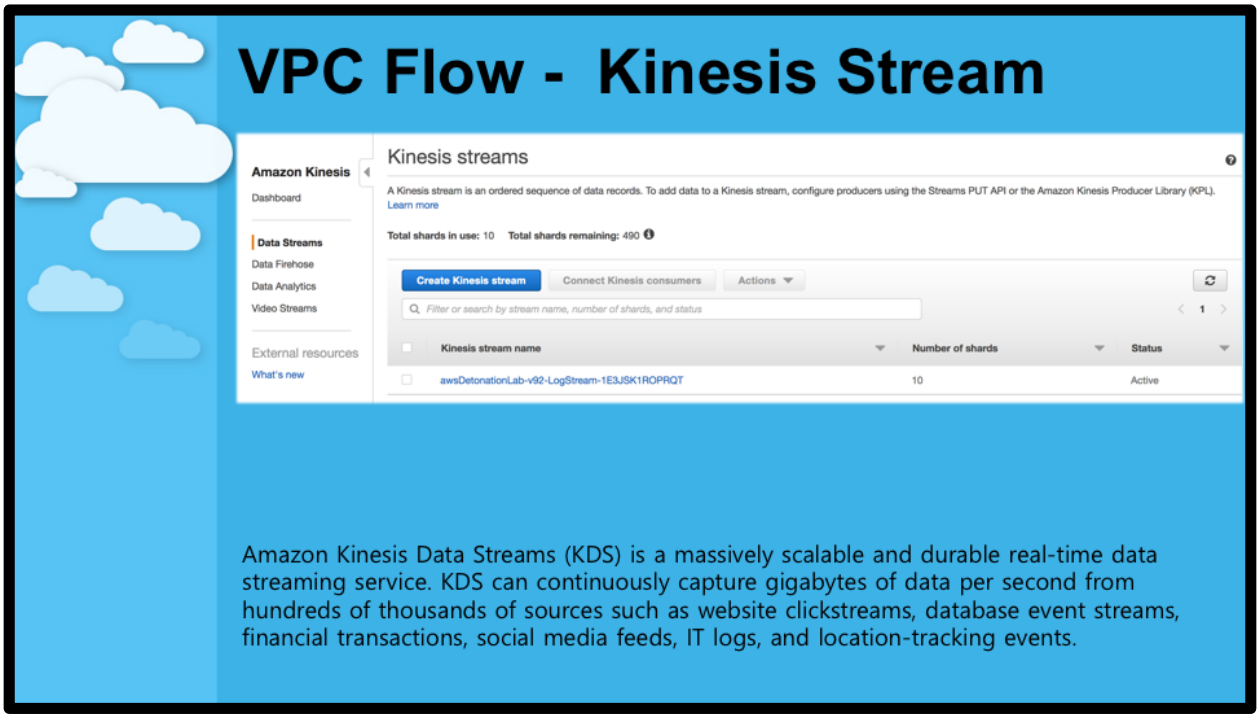
How did I automate the creation of this lab? Cloudformation. This allows me to build a simple or complex template to stand up systems quick. Think of it similar to chef or ansible except this can configure the majority of all aws services including networking and identity management for the entire account.



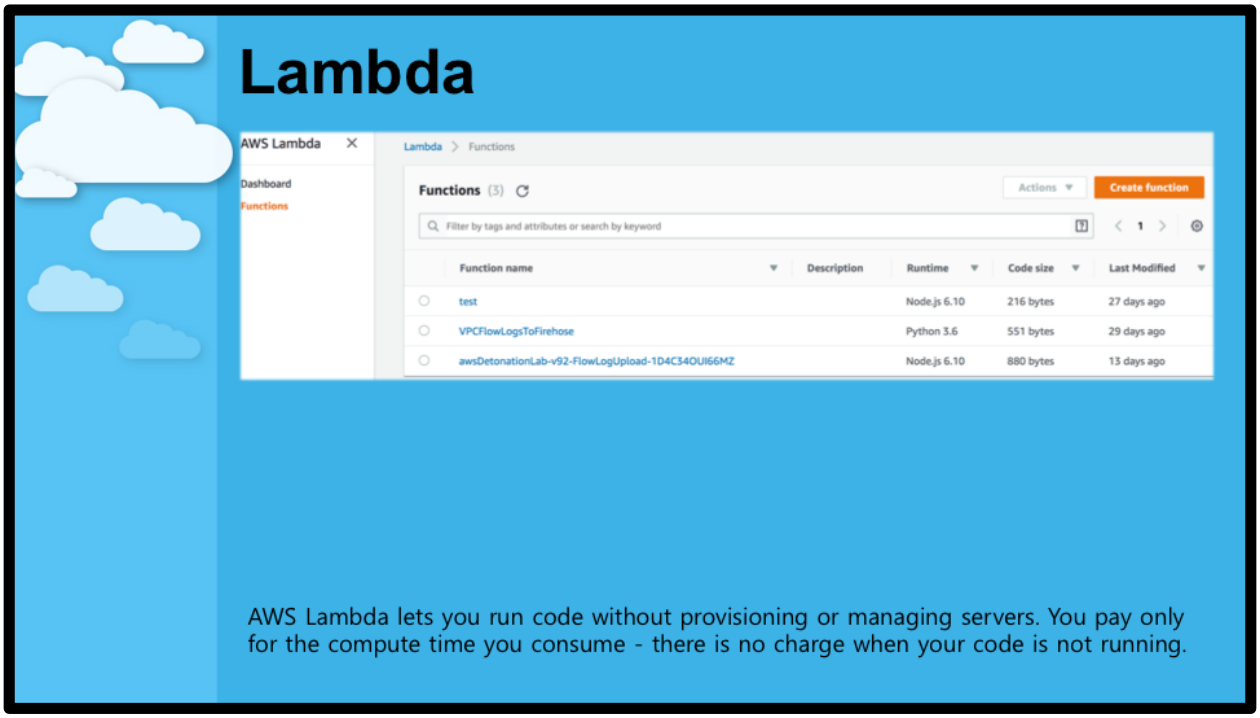
This is my storage.



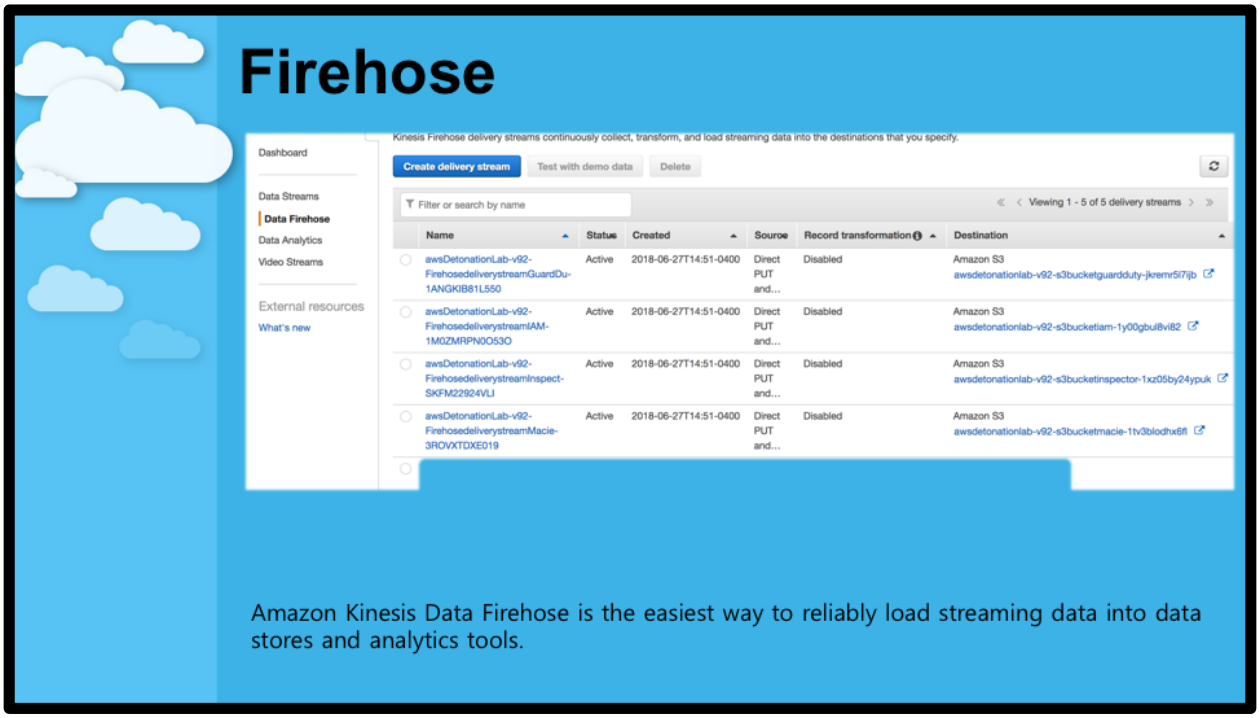
In my lab I am using an isolated VPC to control access to resources for the attacks.



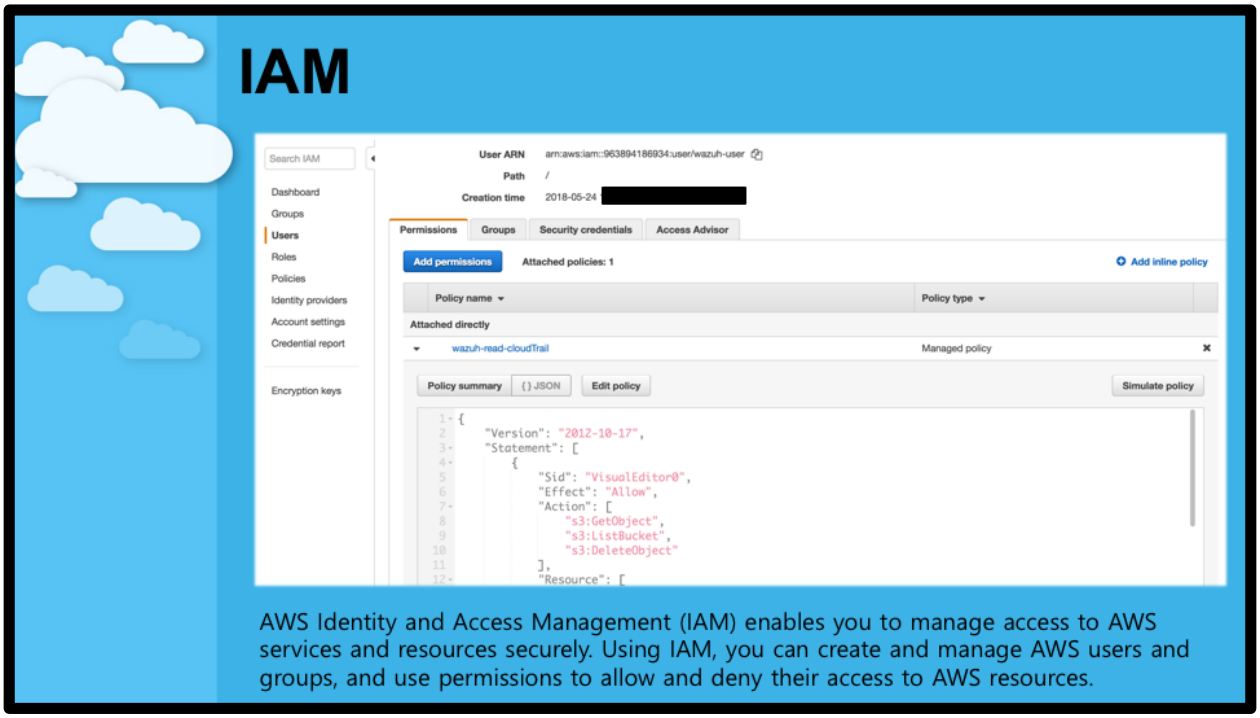
This is the netflow monitoring used for the lab.



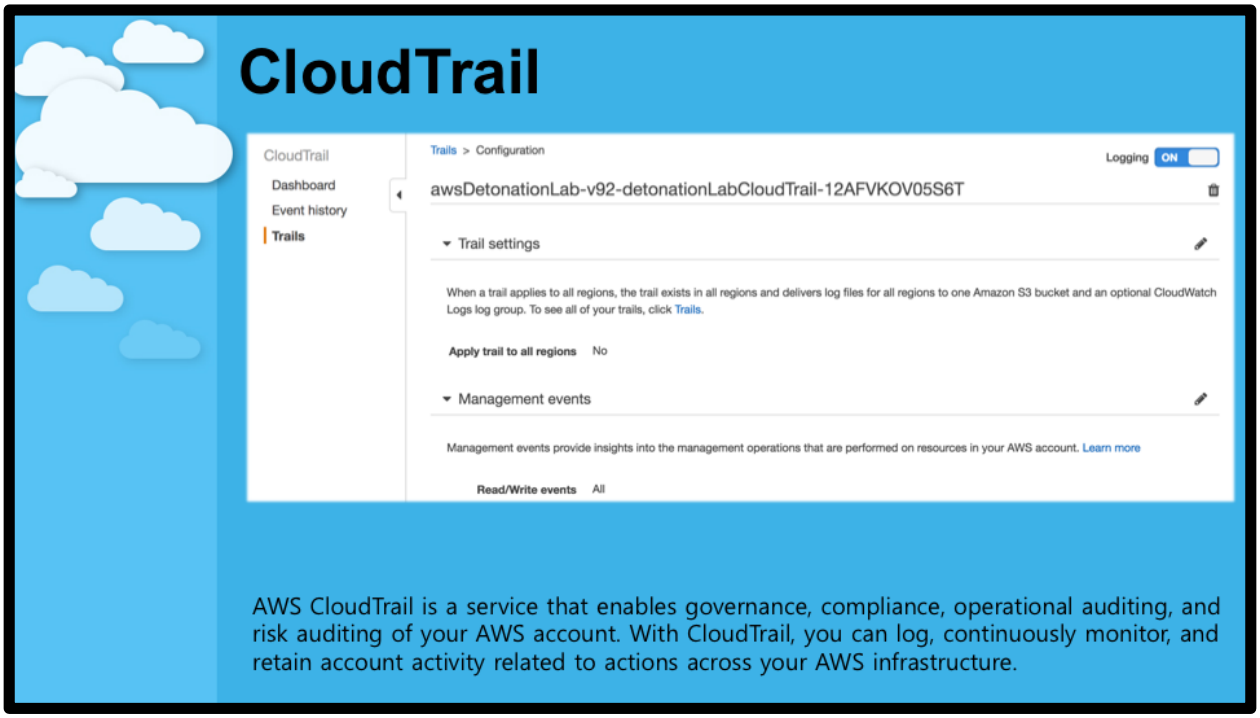
I use lambda to run code against logs I am generating to enhance it with additional details.



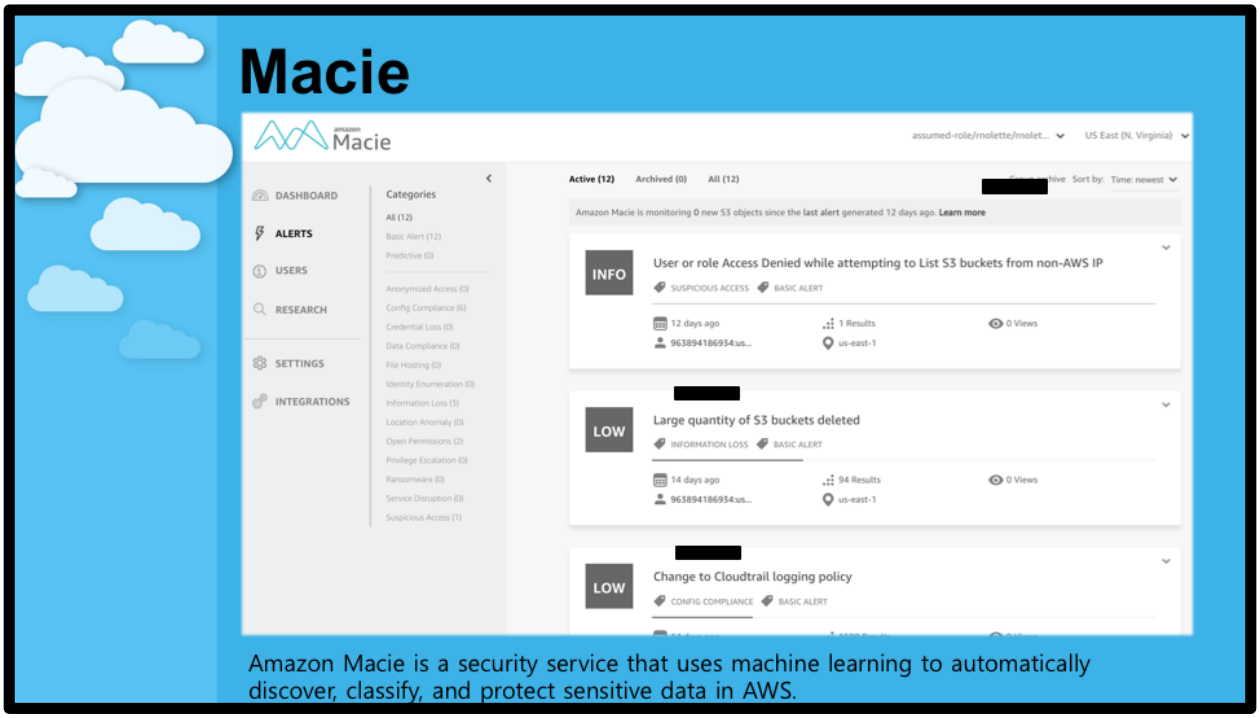
This is how I write the logs to s3



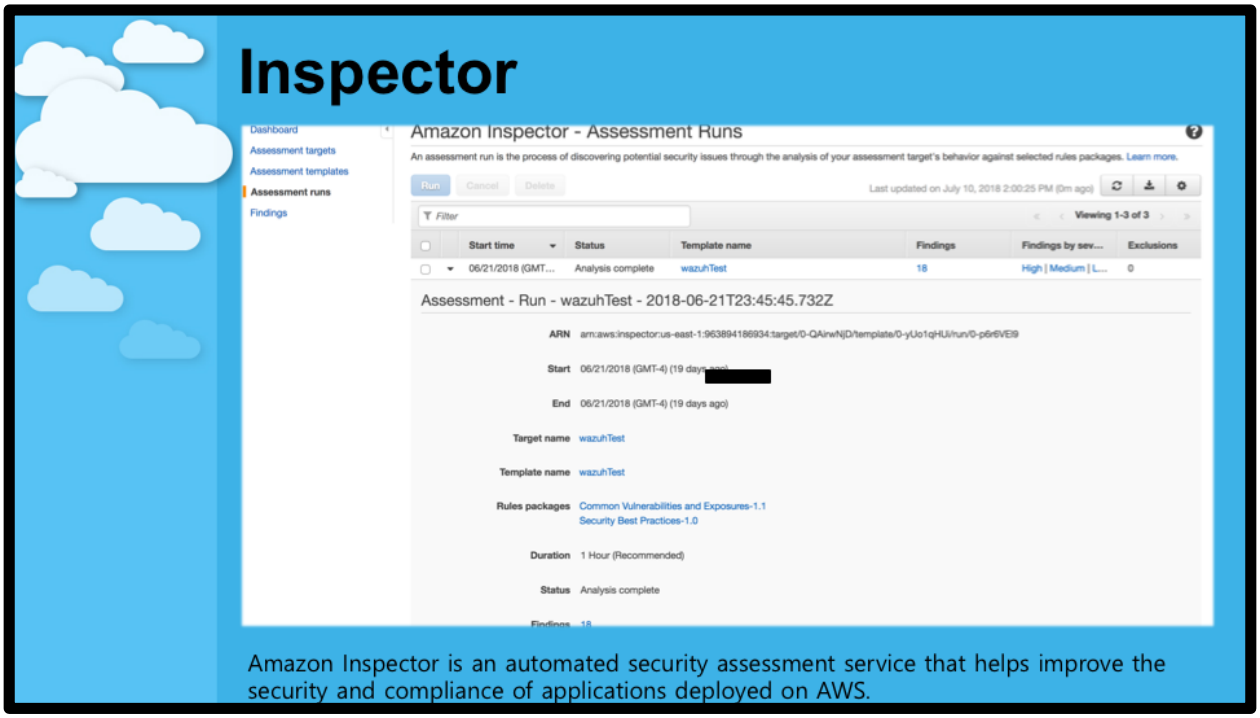
IAM controls the access to all amazon resources for the lab without giving any unneeded priveleges



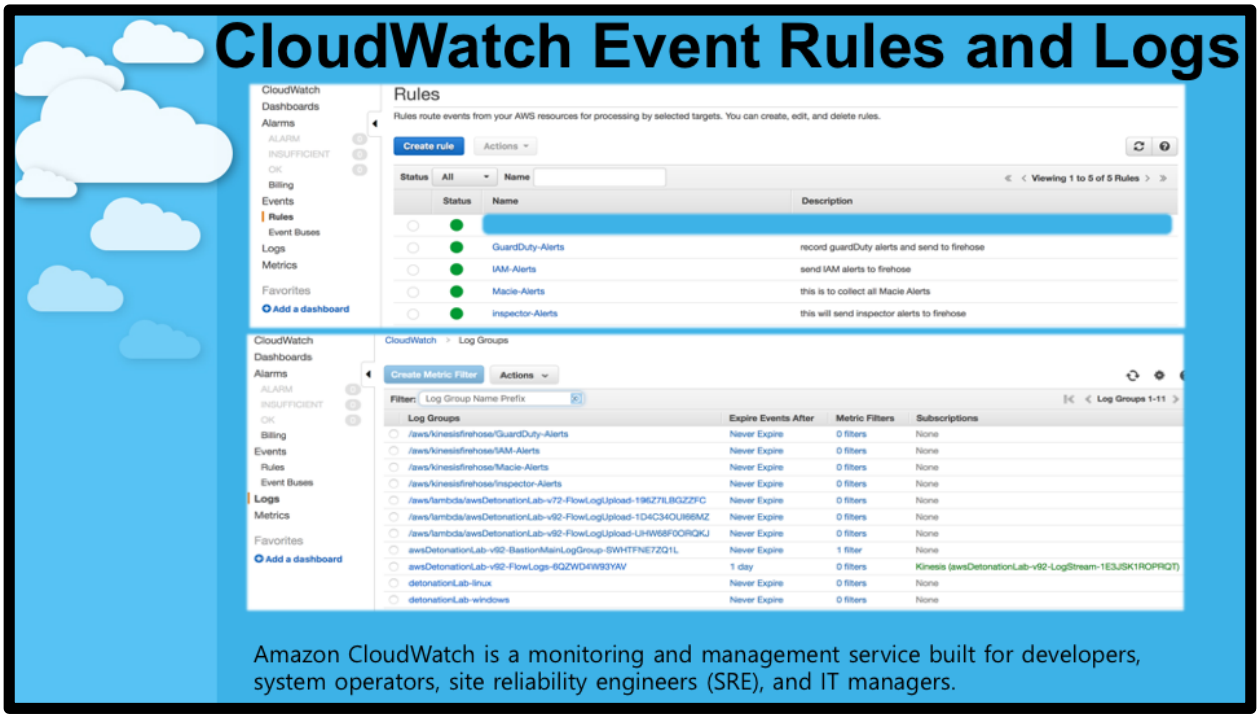
Cloud trail logs all activity to the aws API. Every action take in the console or from the aws cli generates an API action that cloudtrail will monitor.



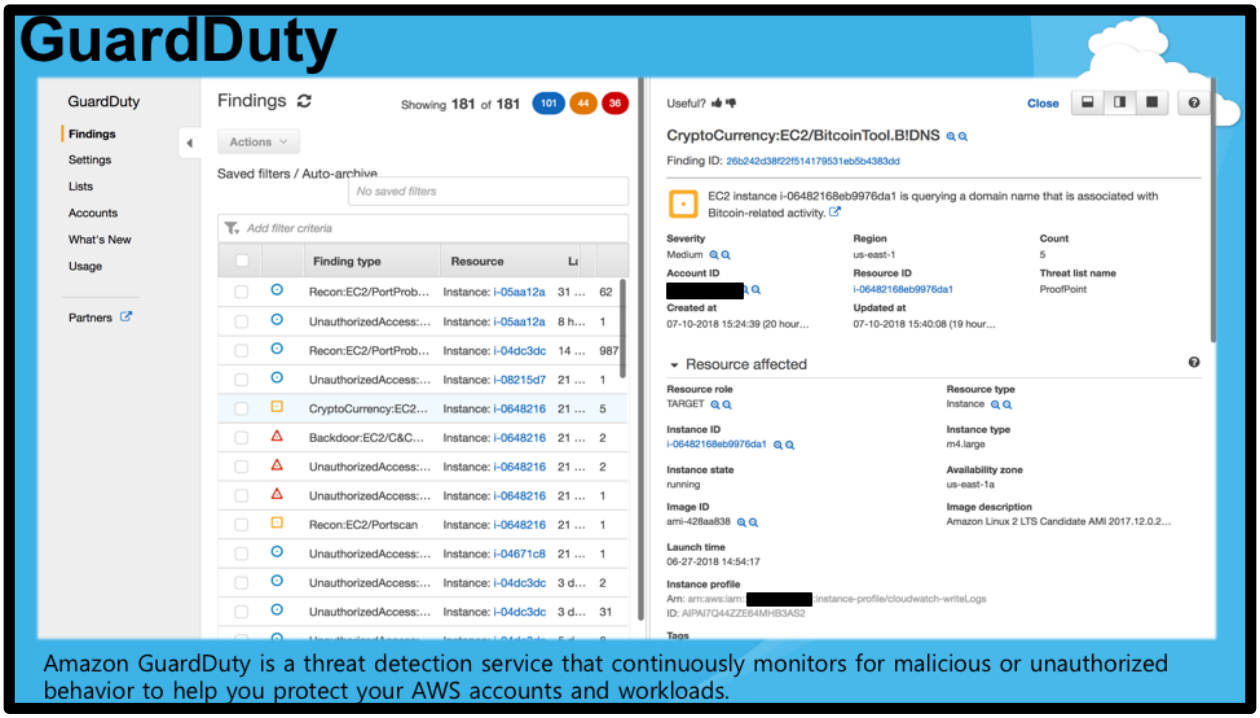
I use macie to monitor the data in my s3 buckets



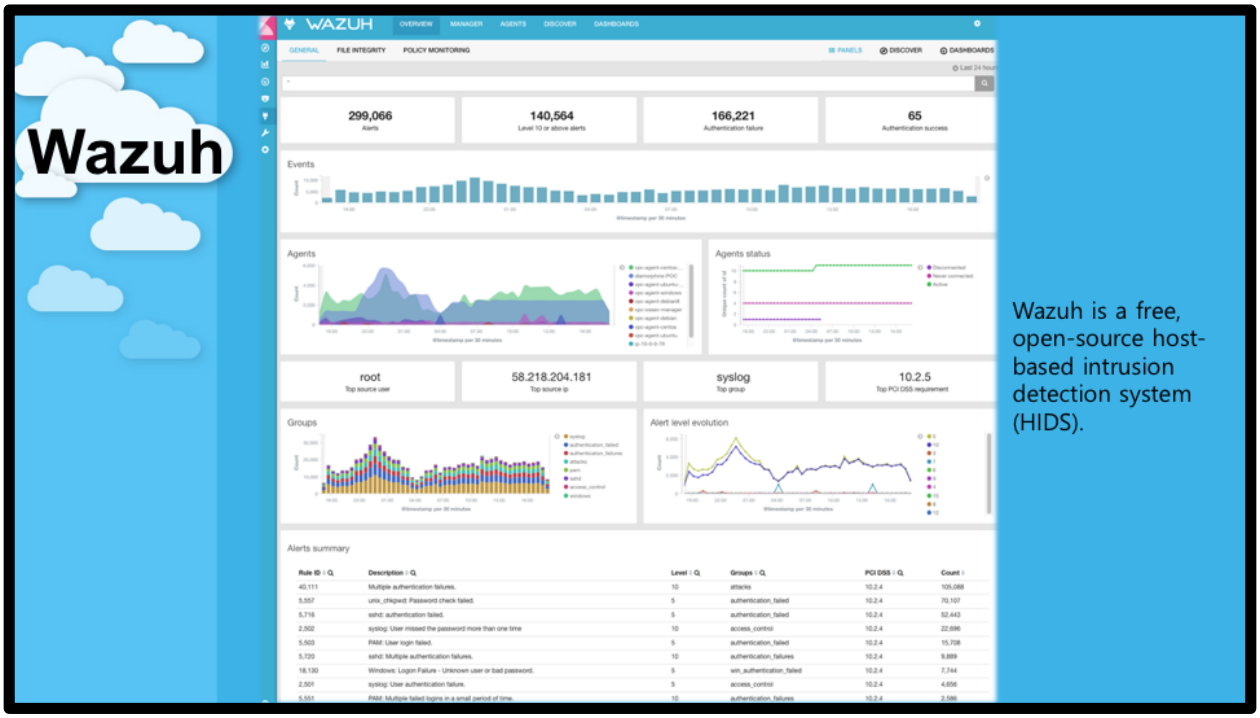
I use inspector as a vulnerability and compliance scanner for my lab



I am using cloudwatch to watch for service events that can be forwarded to a firehose to be written to s3. I also use cloudwatch logs to hold my vpcflow data

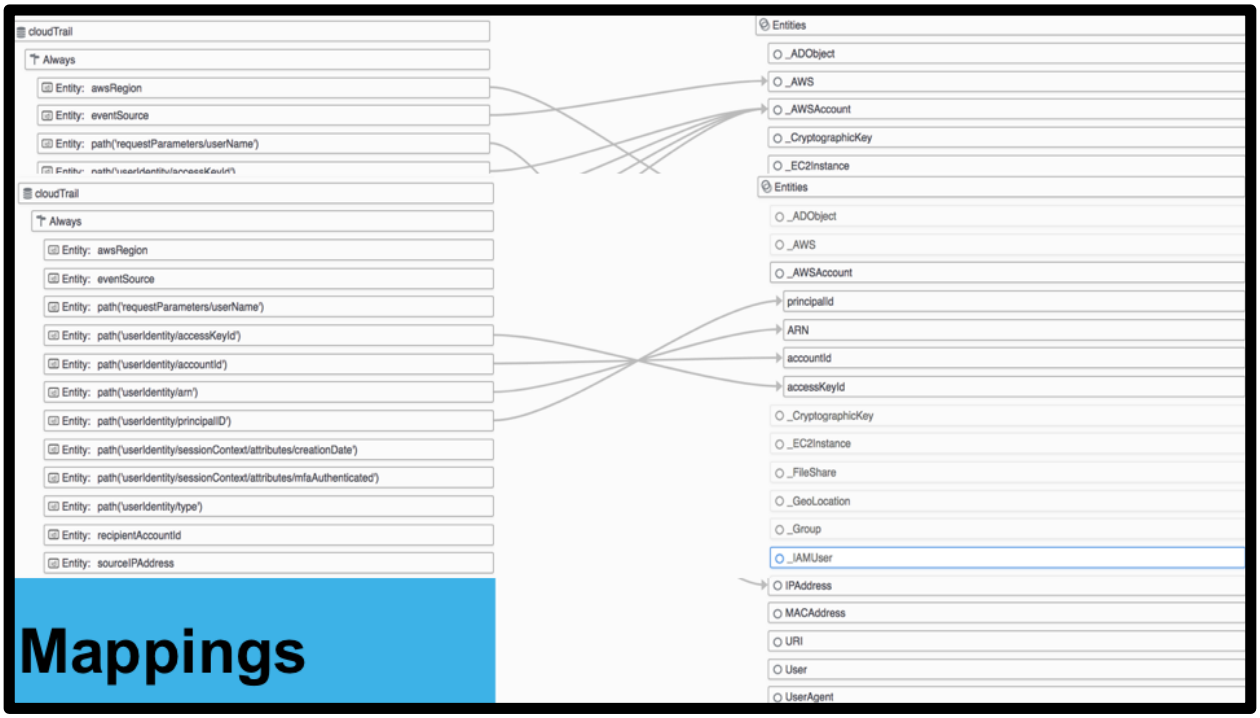


I use Guadduty to alert me on unusual and malicious behaviors like bruteforce attempts, scanning, and credential abuse.



Wazuh is a free, open-source host-based intrusion detection system (HIDS).

I used wazuh as my HIDS for the lab. It is based on OSSEC but has a large number of additional features. I'd like to give a big shoutout to the wazuh team and my contact Marta for working with me to improve the integration of aws service logs into the product. What a great example of the opensource community.

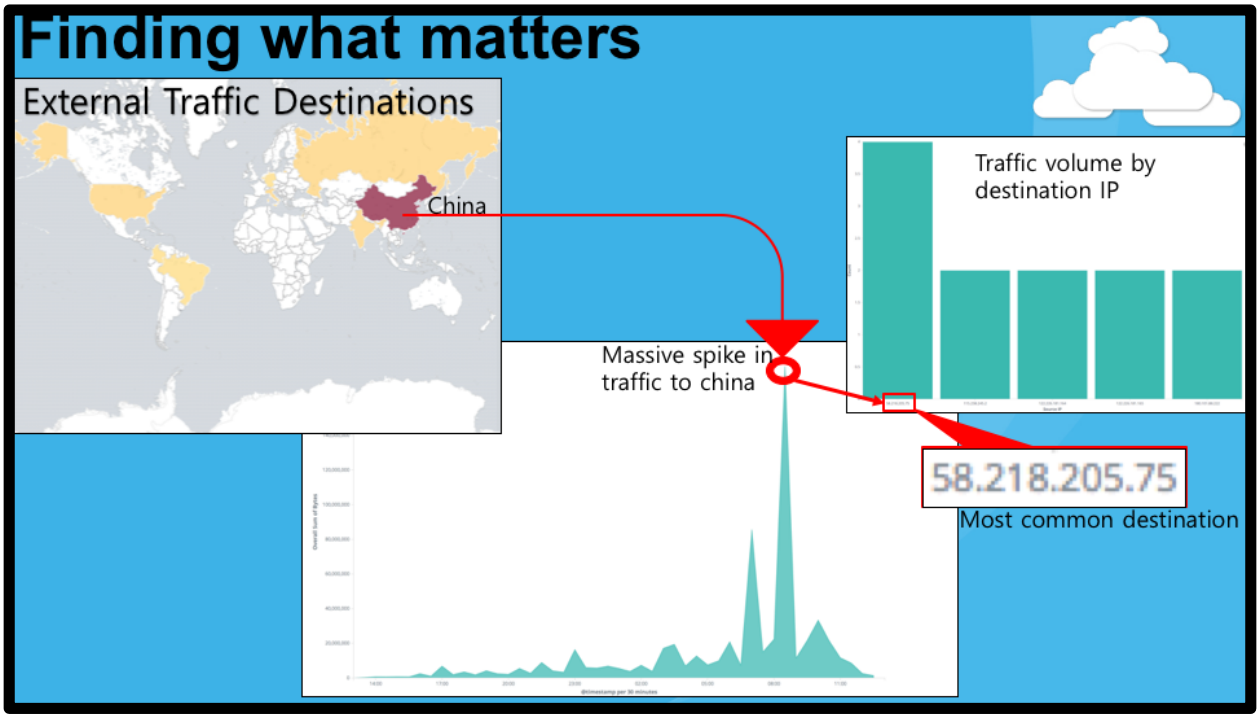


Now that we have all the data getting ingested we need to figure out the best way to correlate the data and start using it to detect and investigate activity

Here I have graphically mapped the datasource cloudtrail to entities and features in a model

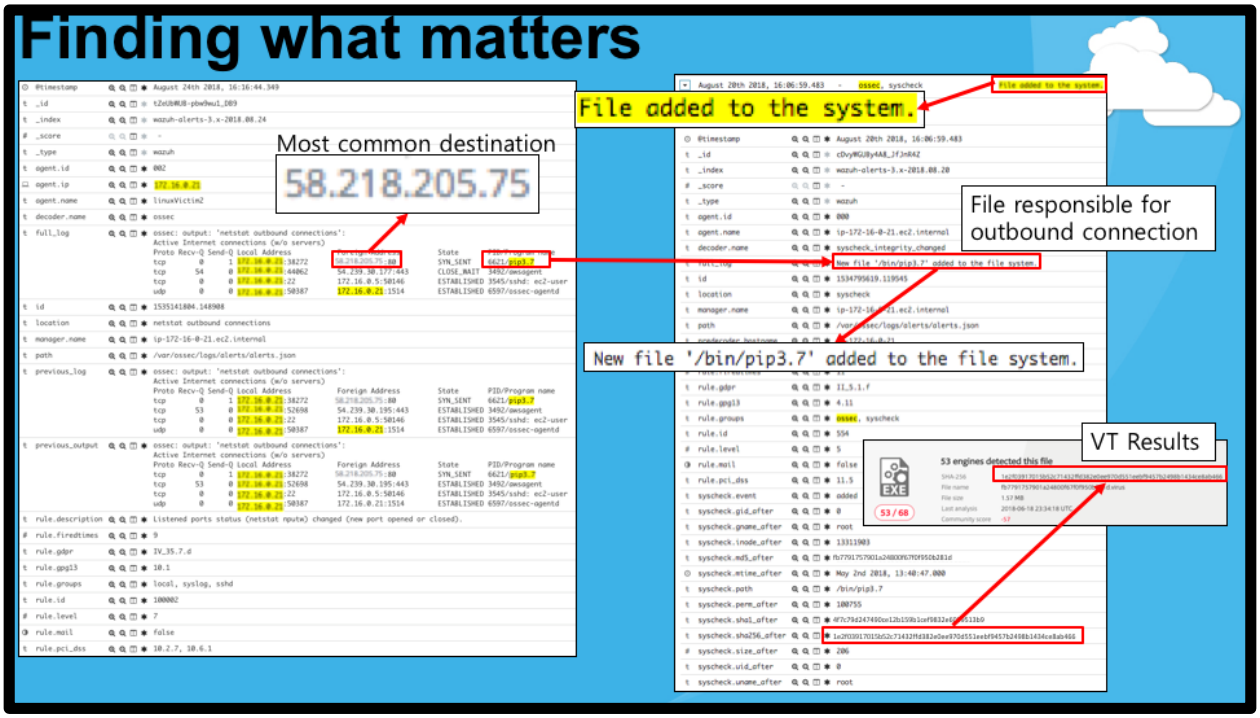
Let's get started with a few simple dashboards. All of these dashboards and more are included in the detonationlab Kibana instance for usage.

Top destination address.
Top talkers by interface.
Most common account ID in guardduty alerts.
Most common cloutrail events



Using some more advanced dashboards have a world heat map for vpc traffic. I noticed that china is now a common destination so I check out a histogram showing volume of traffic and notice that the traffic has suddenly spiked. Pivoting into destination IPs I noticed that a single destination is responsible for the majority of the traffic spike.

From this IP address I check my logs to see if there are any other events involving this destination and I find multiple guardduty alerts for traffic to known C&C server, an ssh brute force attack against one of my instances, and a few other attacks. Let's look at my involved instance for the culprit.



Here I can see a record of the netstat table being changed to involve the known destination IP on that host. It looks like the binary pip3.7 is responsible for the traffic. I do a quick lookup on the hash for the binary on virustotal and find this this binary is actually a known crypto miner.

Ive now done a full stack analysis of a suspicious network event all the way down to the binary responsible for the traffic. What next? Well how did they get in in the first place?

I can go back through all new file alerts and find the user and binary responsible for creating this one. I find that apache was the user writing the file to disk and can assume I have a vulnerable webserver. But how?

How did they get in?

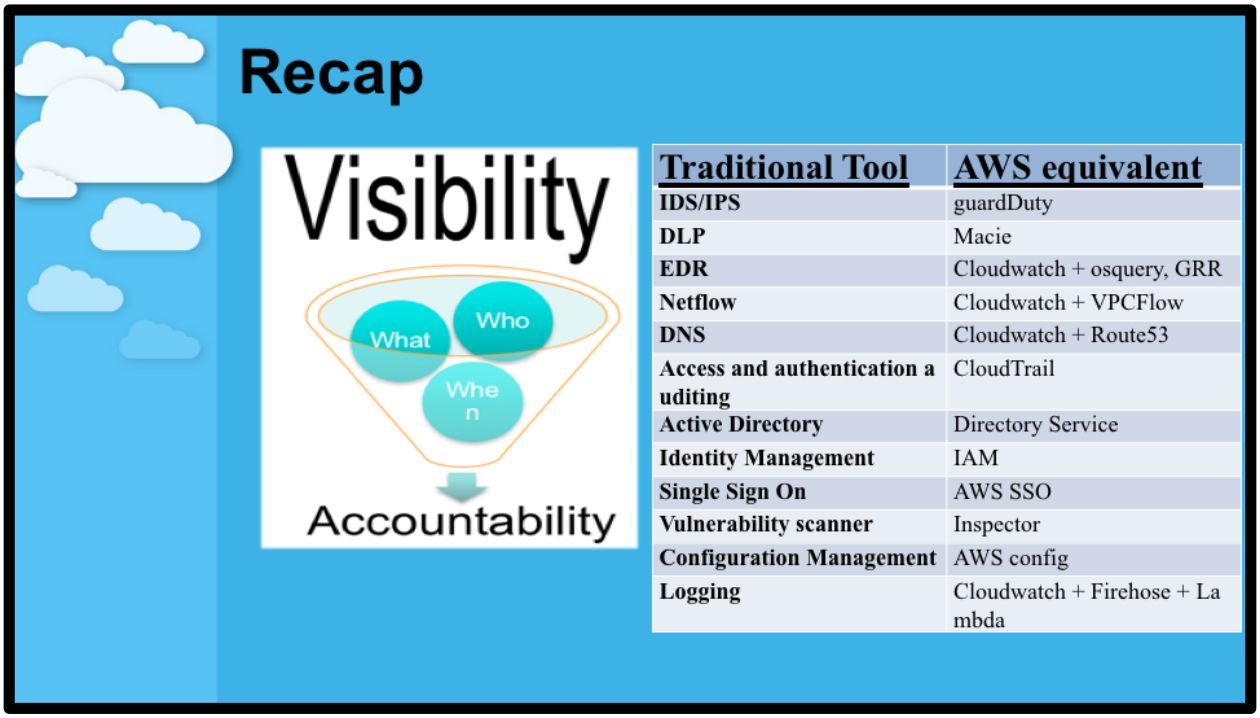
Severity	Date	Finding	Target	Template	Rules Package
High	Yesterday at...	Instance i-063e3b8894815b602 is vulnerable to C...	everything	everything	Common Vulnerabilities and Exposures-1.1

Finding for assessment target 'everything' and template 'everything'

ARN	arn:aws:inspector:us-east-1:963894186934:target/0-ITOJiioL/template/0-S7AjH5cj/run/0-JGN1eNRF/finding/0-jkR0hbLT
Run name	5f44c7fb-3421-133f-1833-5ef4328e05a6_885e45c5-3f8a-9b6d-d70f-194994c58260
Target name	everything
Template name	everything
Start	Yesterday at 10:55 PM (GMT-4) (12 hours ago)
End	Yesterday at 11:56 PM (GMT-4) (11 hours ago)
Status	Analysis complete
Rules package	Common Vulnerabilities and Exposures-1.1
AWS agent ID	i-063e3b8894815b602
Finding	Instance i-063e3b8894815b602 is vulnerable to CVE-2018-10897
Severity	High
Description	A directory traversal issue was found in reposync, a part of yum-utils, where reposync fails to sanitize paths in remote repository configuration files. If an attacker controls a repository, they may be able to copy files outside of the destination directory on the targeted system via path traversal. If reposync is running with heightened privileges on a targeted system, this flaw could potentially result in system compromise via the overwriting of critical system files. Version 1.1.31 and older are believed to be affected.
Recommendation	Use your Operating System's update feature to update package yum-plugin-priorities-0:1.1.31-45.amzn2.0.1, yum-utils-0:1.1.31-45.amzn2.0.1. For more information see https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-10897

Show Details

Using inspector I am able to scan my instance to for common vulnerabilities and found that my server was vulnerable to a known CVE that allows for files to be written to disk as the webserver user



Traditional Tool	AWS equivalent
IDS/IPS	guardDuty
DLP	Macie
EDR	Cloudwatch + osquery, GRR
Netflow	Cloudwatch + VPCFlow
DNS	Cloudwatch + Route53
Access and authentication a uditing	CloudTrail
Active Directory	Directory Service
Identity Management	IAM
Single Sign On	AWS SSO
Vulnerability scanner	Inspector
Configuration Management	AWS config
Logging	Cloudwatch + Firehose + La mbda

For a quick recap, I am able to view activity in my cloud environment very similarly to how I would in my on prem environment. I use these cloud native tools to increase visibility until I am able to find accountability for all actions. Using just tools available from AWS and opensource I can monitor my environment effectively

Last but not least, let's take a look at squirrely attacker personal lifecycle.

They start working hard but not seeing financial returns they need to live how they want to

Then they discover with a few investments they can start making money easily with cryptominers

Next they use those ill gotten gains to buy the only lambo in easter Europe

Which leads to them getting arrested for tax evasion

Which leads to them flipping to act as a witness and end up in the witness protection program

Only to start the cycle again

Thank you all very much for your time today.

And as always remember my motto, flag it, tag it, and bag it.