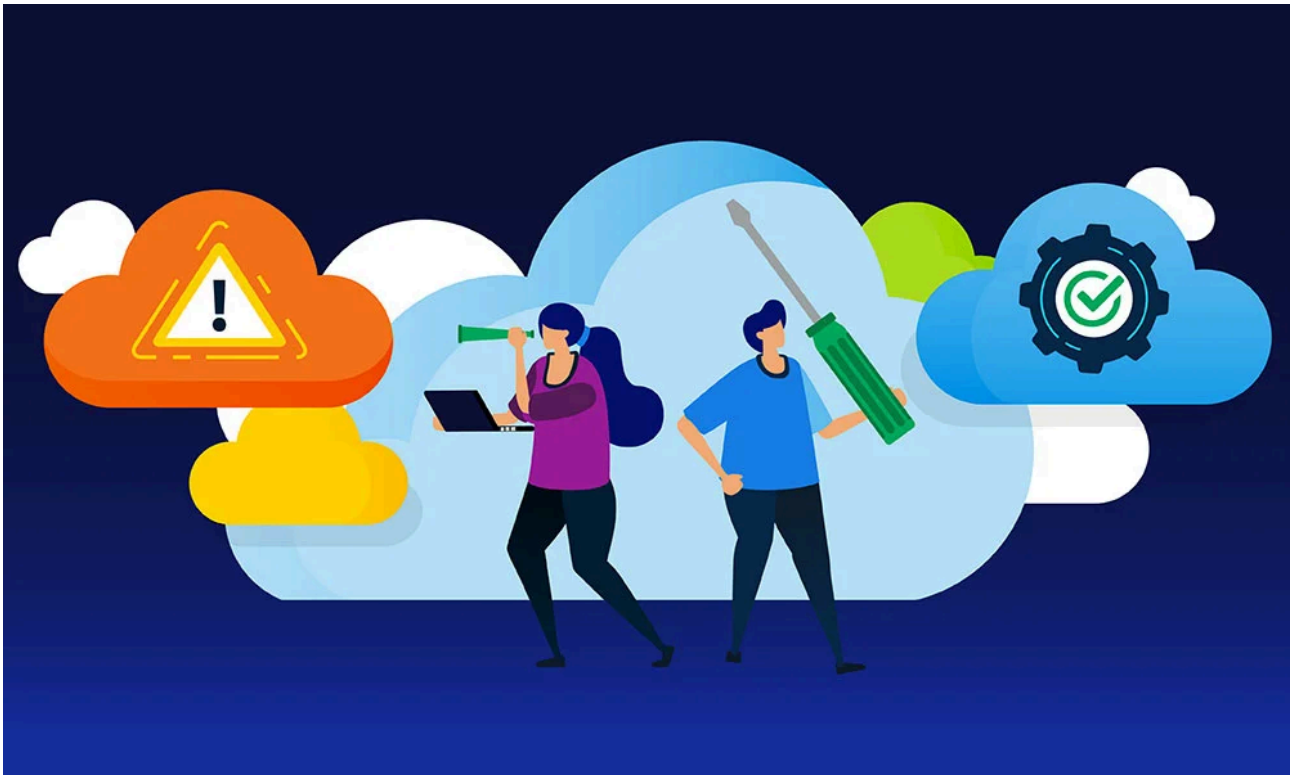
Behind the scenes in the Expel SOC: Alert-to-fix in AWS

By Jon Hencinski, Anthony Randazzo, Sam Lipton, Lori Easterly

Published: 2020-07-28 · Archived: 2026-04-05 20:46:18 UTC



Over the July 4th holiday weekend our SOC spotted a coin-mining attack in a customer's Amazon Web Services (AWS) environment. The attacker compromised the root IAM user access key and used it to enumerate the environment and spin up ten (10) c5.4xlarge EC2s to mine Monero.

While this was *just* a coin miner, it was root key exposure. The situation could have easily gotten out of control pretty quickly. It took our SOC 37 minutes to go from alert-to-fix. That's 37 minutes to triage the initial lead (a custom AWS rule using CloudTrail logs), declare an incident and tell our customer how to stop the attack.

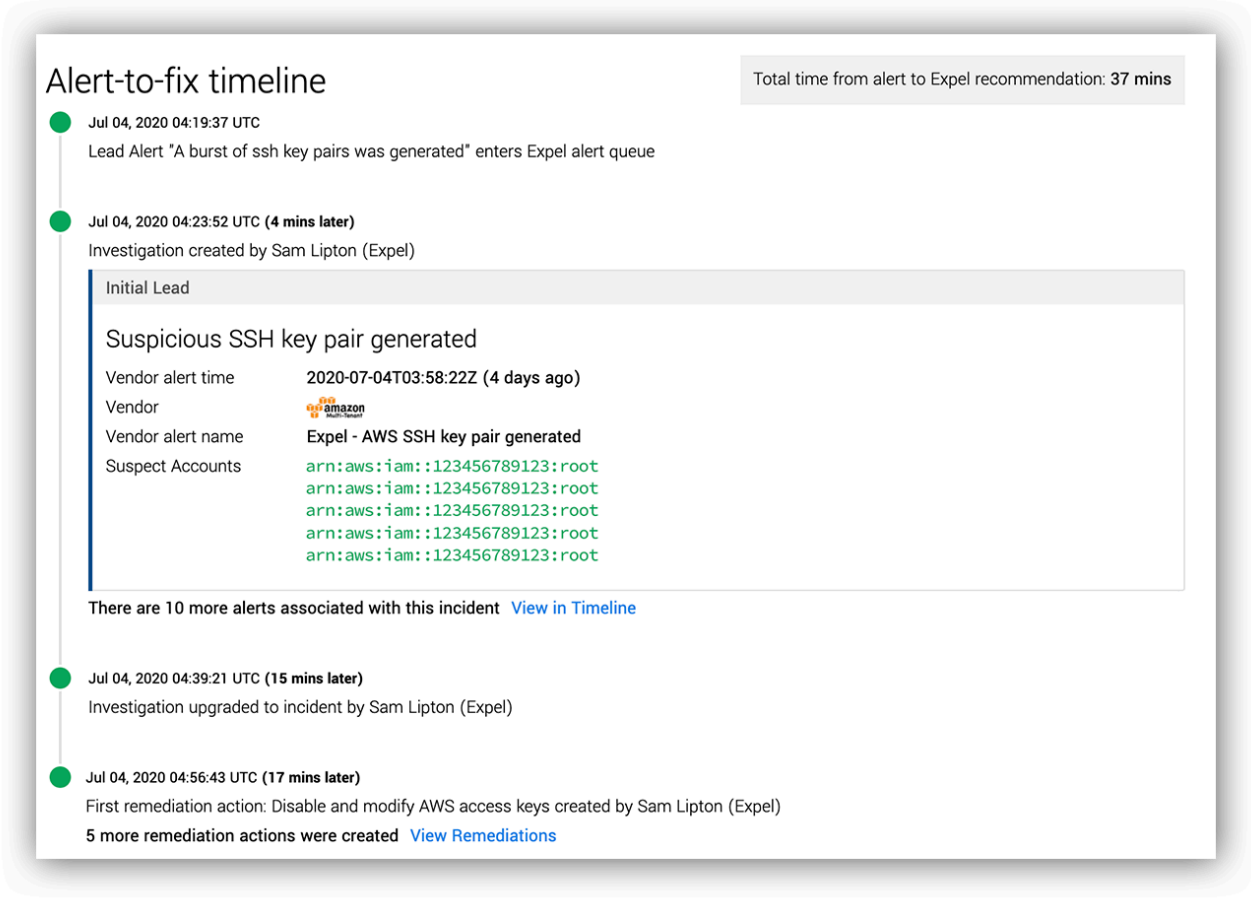
Jon's take: Alert-to-fix in 37 minutes is quite good. Recent industry reporting indicates that most incidents are contained on a time basis measured in days not minutes. Our target is that 75 percent of the time we go from alert-to-fix in less than 30 minutes. Anything above that automatically goes through a review process that we'll talk about more in a bit.

How'd we pull it off so quickly? Teamwork.

We get a lot of questions about what detection and response looks like in AWS, so we thought this would be a great opportunity to take you behind the scenes. In this post we'll walk you through the process from alert-to-fix

in AWS over a holiday weekend. You'll hear from the SOC analysts and Global Response Team who worked on the incident.

Before we tell you how it went down, here's the high level play-by-play:



Triage, investigation and remediation timeline

Now we'll let the team tell the story.

Saturday, July 4, 2020

Initial Lead: 12:19:37 AM ET

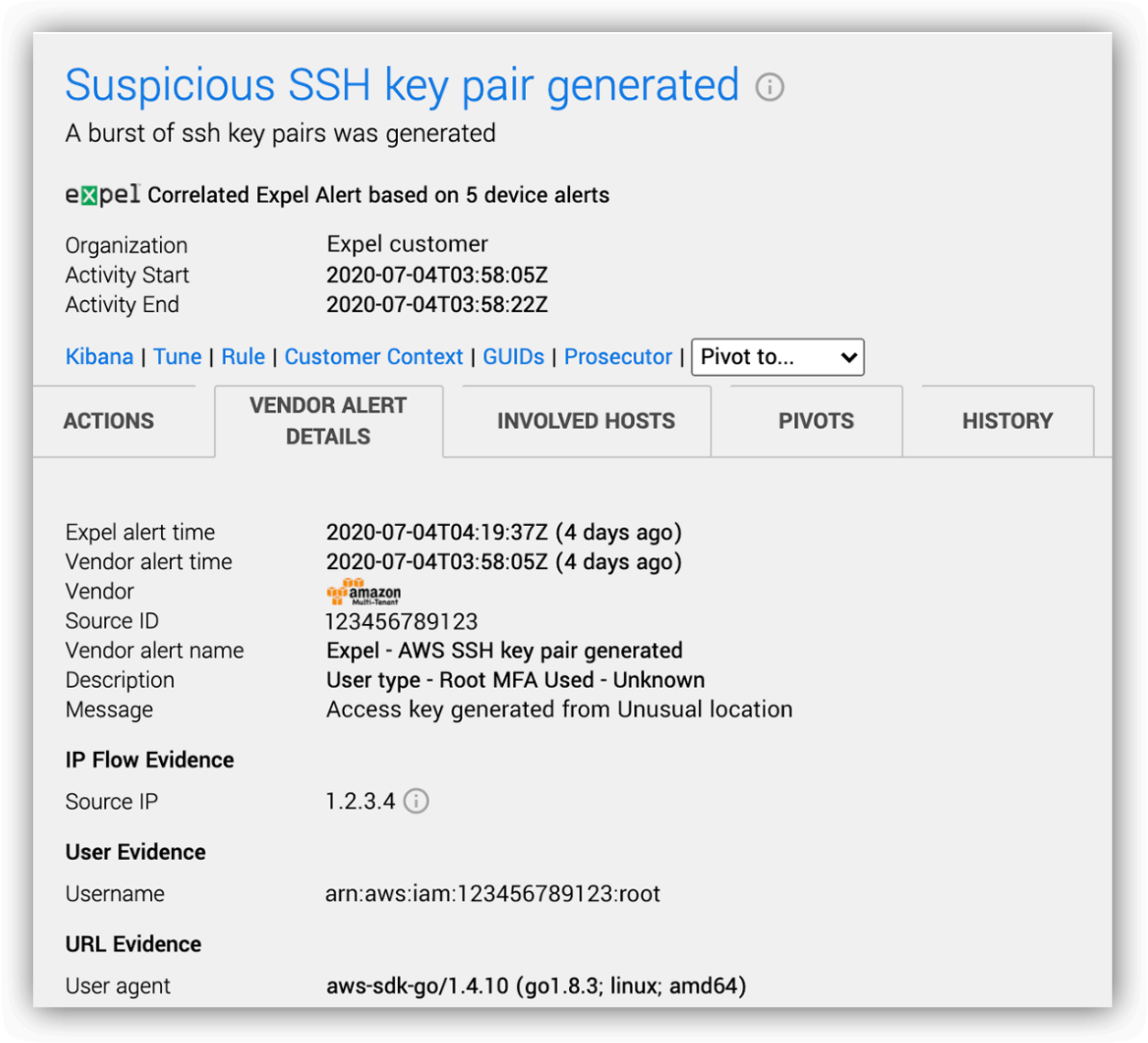
By Sam Lipton and Lori Easterly – SOC analysts

Our shift started at 08:45 pm ET on Friday, July 3. Like many organizations, we've been working fully remotely since the middle of March. We jumped on the Zoom call for shift handoff, reviewed open investigations, weekly alert trending and general info for situational awareness. Things were (seemingly) calm.

We anticipated a quieter shift. On a typical Friday night into Saturday morning, we'll handle about 100 alerts. It's not uncommon for us to spot an incident on Friday evening/Saturday morning, but it's not the norm. It's usually slower on the weekend; there are fewer active users and devices.

Our shift started as we expected, slow and steady. Then suddenly, as is the case in security operations, that all changed.

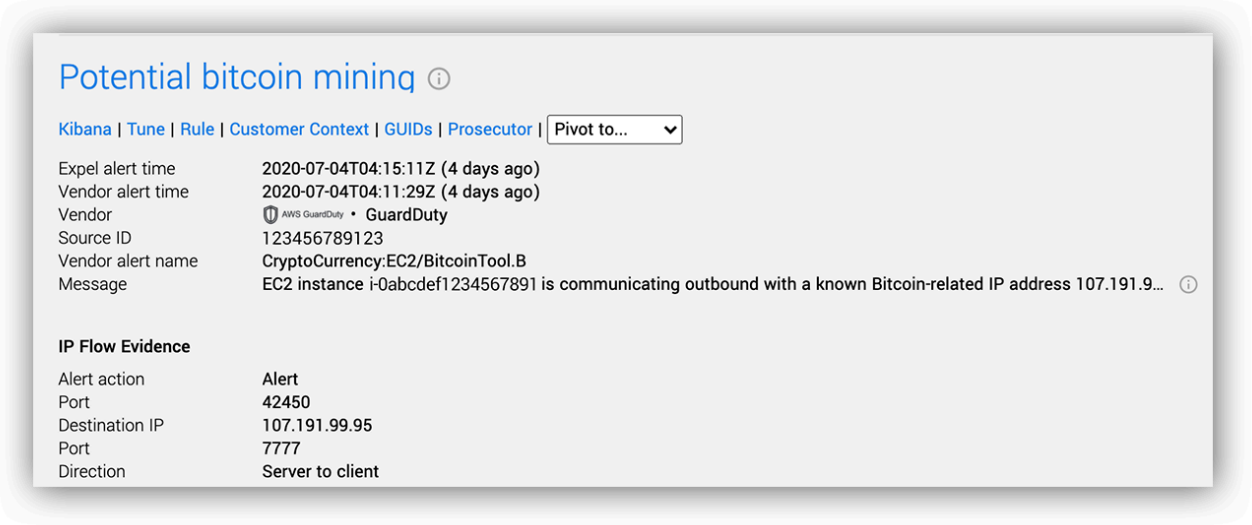
We spotted an AWS alert based on CloudTrail logs that told us that EC2 SSH access keys were generated for the root access key from a suspicious source IP address using the AWS Golang SDK:



Initial lead into the AWS coin-mining incident

The source IP address in question was allocated to a cloud hosting provider that we hadn't previously seen create SSH key pairs via the ImportKeyPair API in this customer's AWS environment (especially from the root account!). The SSH key pair alert was followed shortly thereafter by AWS GuardDuty alerts for an EC2 instance communicating with a cryptocurrency server (monerohash[.]com on TCP port 7777).

We jumped into the SIEM, queried CloudTrail logs and quickly found that the EC2 instances communicating with monerohash[.]com were the same EC2 instances associated with the SSH key pairs that were just detected.



Corroborating AWS GuardDuty alert

As our CTO Peter Silberman says, it was time to buckle up and “pour some Go Fast” on this.

We’ve talked about our Expel robots in a previous post. As a quick refresher, our robot Ruxie (yes– we give our robots names) automates investigative workflows to surface up more details to our analysts. In this event, Ruxie pulled up API calls made by the principal (interesting in this context is mostly anything that isn’t Get*, List*, Describe* and Head*).


AWS Interesting API Calls Enrichment for User arn:aws:iam::123456789123:root

Reason
Robotic Action

Outcome

Api Call	Access Key	IP	UserAgent
CreateSecurityGroup	ABCDEF123456	1.2.3.4	aws-sdk-go/1.4.10 (go1.8.3; linux; amd6
DescribeKeyPairs	ABCDEF123456	1.2.3.4	aws-sdk-go/1.4.10 (go1.8.3; linux; amd6
AuthorizeSecurityGroupIngress	ABCDEF123456	1.2.3.4	aws-sdk-go/1.4.10 (go1.8.3; linux; amd6
CreateTags	ABCDEF123456	1.2.3.4	aws-sdk-go/1.4.10 (go1.8.3; linux; amd6
RunInstances	ABCDEF123456	1.2.3.4	aws-sdk-go/1.4.10 (go1.8.3; linux; amd6
ImportKeyPair	ABCDEF123456	1.2.3.4	aws-sdk-go/1.4.10 (go1.8.3; linux; amd6

AWS alert decision support – Tell me what other interesting API calls this AWS principal made

This made it easy for us to understand what happened:

- The root AWS access key was potentially compromised.
- The root access key was used to access the AWS environment from a cloud hosting environment using the AWS Golang SDK. It was then used to create SSH keys, spin up EC2 instances via the RunInstances API call and created new security groups likely to allow inbound access from the Internet.
- We inferred that the root access key was likely compromised and used to deploy coin miners.

Yep, time to escalate this to an incident, take a deeper look, engage the customer and notify the on-call Global Response Team Incident Handler.

PagerDuty escalation to Global Response Team: 12:37:00 AM ET

Our Global Response Team (GRT) consists of senior and principal-level analysts who serve as incident responders for critical incidents. AWS root key exposure introduces a high level of risk for any customer, so we made the call to engage the GRT on call using PagerDuty. The escalation goes out to a Slack channel that's monitored by the management team to track utilization.

PagerDuty escalation out to the GRT on-call

Incident declaration: 12:39:21 AM ET

A few minutes after the initial lead landed in Expel Workbench – 19 minutes to be exact – we notified the customer that there was a critical security incident in their AWS environment involving the root access key. And that access key was used to spin up new EC2 instances to perform coin mining. Simultaneously, we jumped into our SIEM and queried CloudTrail logs to help answer:

- Did the attacker compromise any other AWS accounts?
- How long has the attacker had access?
- What did the attacker do with the access?
- How did the attacker compromise the root AWS access key?

At 12:56:43 ET we provided the first remediation actions to our customer to help contain the incident in AWS based on what we knew. This included:

- Steps on how to delete and remove the stolen root access key; and
- Instructions on how to terminate EC2 instances spun up by the attacker.

We felt pretty good at this point – we had a good understanding of what happened. The customer acknowledged the critical incident and started working on remediation, while the GRT Incident Handler was inbound to perform a risk assessment.

Alert-to-fix in 37 minutes. Not a bad start to our shift.

Global Response Team enters the chat: 12:42:00 AM ET

By Anthony Randazzo – Global Response Team Lead

I usually keep my phone on silent, but PagerDuty has a vCard that allows you to set an emergency contact. This bypasses your phone's notifications setting so that if you receive a call from this contact, your phone rings (whether it's in silent mode or not).

We call it the SOC "bat phone."

This wasn't the first time I was paged in the middle of the night. I grabbed my phone, saw the PagerDuty icon and answered.

There's a lot of trust in our SOC. I knew immediately that if I was being paged, then the shift analysts were confident that there was something brewing that needed my attention.

I made my way downstairs to my office and hopped on Zoom to get a quick debrief from the analysts about what alerts came in and what they were able to discover through their initial response. Now that I'm finally awake, it's time to surgically determine the full extent of what happened.

As the GRT incident handler, it's important to not only perform a proper technical response to the incident, but also understand the risk. That way, we can thoroughly communicate with our customer at any given time throughout the incident, and continue to do so until we're able to declare that the incident is fully contained.

At this point, we have the answers to most of our investigative questions, courtesy of the SOC shift analysts:

- Did the attacker compromise any other AWS accounts? **There is no evidence of this.**
- How long has the attacker had access? **This access key was not observed in use for the previous 30 days.**
- What did the attacker do with the access? **The attacker generated a bunch of EC2 instances and enabled an ingress rule to SSH in and install CoinMiner malware.**
- How did the attacker compromise the root AWS access key? **We don't know and may never know.**

My biggest concern at this point was communicating to the customer that the access key remediation needs to occur as soon as possible. While this attack was an automated coin miner bot, there was still an unauthorized attacker with an intent of financial gain that had root access to an AWS account containing proprietary and potentially sensitive information lurking somewhere.

There are a lot of "what ifs" floating around in my head. What if the attacker realizes they have a root access key? What if the attacker decides to start copying our customer's EBS volumes or RDS snapshots?

Incident contained: 02:00:00 AM ET

By 2:00 am ET we had the incident fully scoped which meant we understood:

- When the attack started
- How many IAM principals the attacker compromised
- AWS EC2 instances compromised by the attacker
- IP addresses used by the attacker to access AWS (ASN: AS135629)
- Domain and IP address resolutions to coin mining pool (monerohash[.]com:7777)
- And API calls made by the attacker using the root access key

At this point I focused on using what we understood about the attack to deliver complete remediation steps to our customer. This included:

1. A full list of all EC2 instances spun up by the attacker with details on how to terminate them
2. AWS security groups created by the attacker and how to remove them
3. Checking in on the status of the compromised root access key

I provided a summary of everything we knew about the attack to our customer, did one last review of the remediation steps for accuracy and chatted with the SOC over Zoom to make sure we set the team up for success if the attacker came back.

For reference, below are the MITRE ATT&CK Enterprise and Cloud Tactics observed during Expel's response:

MITRE ATT&CK Enterprise and Cloud Tactics observed during Expel's response	
Initial Access	Valid Accounts
Execution	Scripting
Persistence	Valid Accounts, Redundant Access
Command and Control	Uncommonly Used Port

With the incident now under control, I resolved the PagerDuty escalation and called it a morning.

PagerDuty escalation resolution at 2:07am ET

Tuesday, July 7th

By Jon Hencinski – Director of Global Security Operations

Critical incident hot wash: 10:00:00 AM ET

For every critical incident we'll perform a lightweight 15-minute "hot wash." We use this time to come together as a team to reflect and learn. NIST has some opinions on what you should ask, at Expel we mainly focus on asking ourselves:

- How quickly did we detect and respond? Was this within our internal target?
- Did we provide the right remediation actions to our customer?
- Did we follow the process and was it effective?
- Did we fully scope the incident?
- Is any training required?
- Were we effective? If not, what steps do we need to take to improve?

If you're looking for an easy way to get started with a repeatable incident hot wash, steal this:

(ツ)	ಠ_ಠ	(╯°□°)╯ ┻━┻
Things that went well	Things that went 'meh'	Things that went bad

Incident hot wash document template. Steal me!

The bottom line: celebrate what went well and don't be afraid to talk about where you need to improve. Each incident is an opportunity to advance your skills and train your investigative muscle.

Lessons Learned

We were able to help our customer get the situation under control pretty quickly but there were still some really interesting observations:

- It's entirely possible that the root access key was scraped and passed off to the bot to spin up miners right before this was detected. We didn't see any CLI, console or other interactive activity, fortunately.
- The attacker definitely wasn't worried about setting off any sort of billing or performance alarms given the size of these EC2s.
- This was the first time we saw an attacker bring their own SSH key pairs that were uniquely named. Usually we see these generated in the bot automation run via the CreateKeyPair API.
- The CoinMiner was likely installed via SSH remote access (as a part of the bot). We didn't have local EC2 visibility to confirm, but an ingress rule was created in the bot automation to allow SSH from the Internet.
- This was also the first time we'd observed a bot written in the AWS Golang software development kit (SDK). This is interesting because as defenders, it's easy to suppress alert-based on user-agents, particularly SDKs we don't expect to be used in attacks.

We'll apply these lessons learned, continue to improve our ability to spot evil quickly in AWS and mature are response procedures.

While we felt good about taking 37 minutes to go from alert-to-fix in AWS in the early morning hours, especially during a holiday, we don't plan on letting it get to our heads. We hold that highly effective SOCs are the right combination of people, tech and process.

Really great security is a process, there is no end state – the work to improve is never done!

Did you find this behind-the-scenes look into our detection and response process helpful? If so, let us know and we'll plan to continue pulling the curtain back in the future!
