Managed Detection & Response for AWS

By Anthony Randazzo, Britton Manahan, Sam Lipton

Published: 2020-04-28 · Archived: 2026-04-05 23:21:15 UTC

Detection and response in cloud infrastructure is a relatively new frontier. On top of that, there aren't many compromise details publicly available to help shape the detection strategy for anyone running workloads in the cloud.

That's why our team here at Expel is attempting to bridge the gap between theory and practice.

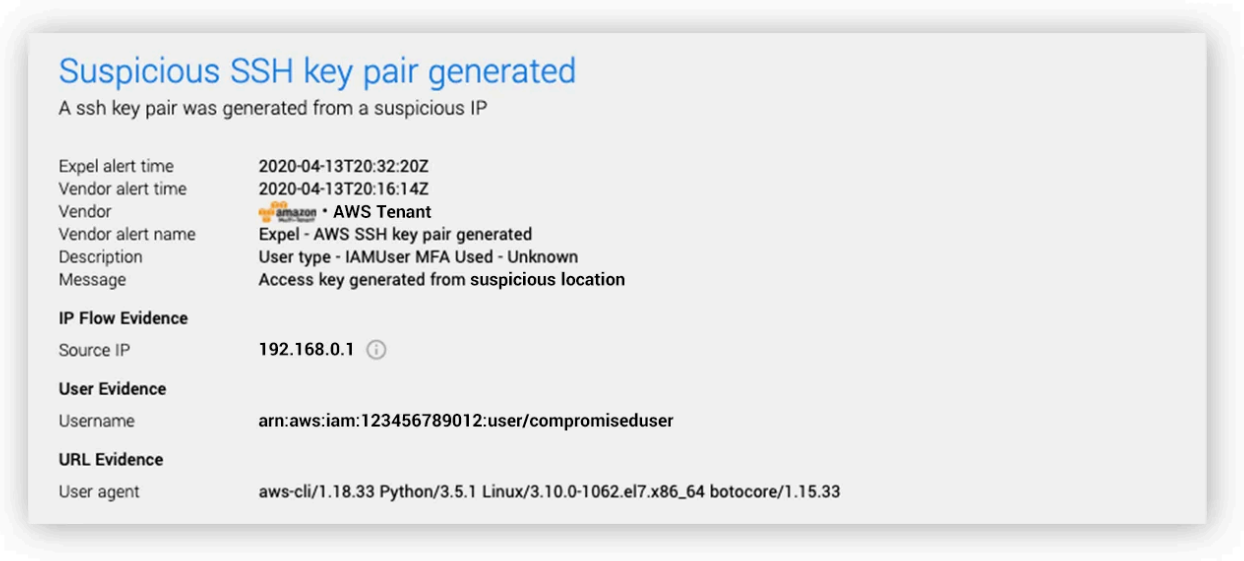
Over the years, we've detected and responded to countless Amazon Web Services (AWS) incidents, ranging from public S3 bucket exposures to compromised EC2 instance credentials and RDS ransomware attacks. Recently, we identified an incident involving the use of compromised AWS access keys.

In this post, we'll walk you through how we caught the problem, what we observed in our response, how we kicked the bad guy out and the lessons we learned along the way.

Compromised AWS access keys: How we caught 'em

We first determined there was something amiss thanks to an Expel detection using CloudTrail logs. Here at Expel, we encourage many of our customers who run on AWS to use Amazon GuardDuty. But we've also taken it upon ourselves to develop detection use cases against CloudTrail logs. GuardDuty does a great job of identifying common attacks, and we've also found CloudTrail logs to be a great source of signal for additional alerting that's more specific to an AWS service or an environment.

It all started with the alert below, telling us that EC2 SSH access keys were being generated (CreateKeyPair/ImportKeyPair) from a suspicious source IP address.



Initial lead Expel alert

How'd we know it was suspicious?

We've created an orchestration framework that allows us to launch actions when certain things happen. In this case, when an alert fired an Expel robot picked it up and added additional information. This robot uses a third-party enrichment service for IPs (in this case, our friends at ipinfo.io). More on our robots here shortly.

Keep in mind that these are not logins to AWS per se. These are authenticated API calls with valid IAM user access keys. API access can be restricted at the IP layer, but it can be a little burdensome to manage in the IAM Policy. As you can see in the alert shown above, there was no MFA enforced for this API call. Again, this was not a login, but you can also enforce MFA for specific API calls through the IAM Policy. We've observed only a few AWS customers using either of these controls.

Another interesting detail from this alert was the use of the AWS Command Line Interface (CLI). This isn't completely out of the norm, but it heightened our suspicion a bit because it's less common than console (UI) or AWS SDK access. Additionally, we found this user hadn't used the AWS CLI in recent history, potentially indicating a new person was using these credentials. The manual creation of an access key was also an atypical action versus leveraging infrastructure as code to manage keys (i.e. CloudFormation or Terraform).

Taking all of these factors into consideration, we knew we had an event worthy of additional investigation.

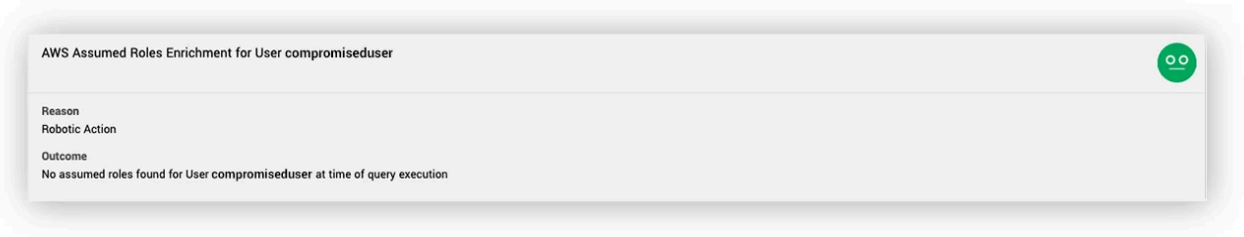
Cue the robots, answer some questions

Our orchestration workflows are critically important – they tackle highly repetitive tasks, that is answer questions an analyst would ask about an alert, on our behalf as soon as the alert fires. We call these workflows our robots.

When we get an AWS alert from a customer's environment, we have three consistent questions we like to answer to help our analysts determine if it's worthy of additional investigation (decision support):

- Did this IAM principal (user, role, etc.) assume any other roles?
- What AWS services does this principal normally interact with?
- What interesting API calls has this principal made?

So, when the initial lead alert for the SSH key generation came in, we quickly understood that role assumption was not in play for this compromise. If the user had assumed roles, it would have been key to identity and include them in the investigation. Instead, we saw the image below:



Expel AWS AssumeRole Robot

Once we knew access was limited to this IAM user, we wanted to know what AWS services this principal generally interacts with. Understanding this helps us spot outlier activity that's considered unusual for that principal. Seeing the very limited API calls to other services further indicated that something nefarious might be going on.

Expel AWS Service Interaction Robot

Finally, we wanted to see what interesting API calls the principal made. From a detection perspective, we define interesting API calls in this context to be mostly anything that isn't Get*, List*, Describe* and Head*. This enrichment returned 344 calls to the AuthorizeSecurityGroupIngress API from the AWS CLI user-agent. This is really the tipping point for considering this a security incident.

Expel AWS Interesting API Robot

How we responded

After we spotted the attack, we needed to scope this incident and provide the measures for containment. We framed our response by asking the primary investigative questions. Our initial response was going to be limited to determining what happened in the AWS control plane (API). CloudTrail was our huckleberry for answering most of our questions.

- What credentials did the attacker have access to?
- How long has the attacker had access?
- What did the attacker do with the access?
- How did the attacker get access?

What credentials did the attacker have access to?

By querying historical CloudTrail events for signs of this attacker, Expel was able to identify that they had access to a total of eight different IAM user access keys, and was active from two different IPs. If we recall from earlier, we were able to use our robot to determine that no successful AssumeRole calls were made, limiting our response to these IAM users.

How long has the attacker had access?

CloudTrail indicated that most of the access keys had not been used by anyone else in the past 30 days thus we can infer that the attacker likely discovered the keys recently.

What did the attacker do with the access?

Based on observed API activity, the attacker had a keen interest in S3, EC2 and RDS services as we observed **ListBuckets**, **DescribeInstances** and **DescribeDBInstances** calls for each access key, indicating an attempt to see which of these resources was available to the compromised IAM user.

As soon as the attacker identified a key with considerable permissions, **DescribeSecurityGroups** was called to determine the level of application tier access (firewall access) into the victim's AWS environment. Once these groups were enumerated, the attacker "backdoored" all of the security groups with a utility similar to aws_pwn's backdoor_all_security_group script. This allowed for any TCP/IP access into the victim's environment.

Additional **AuthorizeSecurityGroupIngress** calls were made for specific ingress rules for port 5432 (postgresql) and port 1253, amounting to hundreds of unique Security Group rules created. These enabled the attacker to gain network access to the environment and created additional risks by exposing many AWS service instances (EC2, RDS, etc.) to the internet.

A subsequent **DescribeInstances** call identified available EC2 instances to the IAM user. The attacker then created a SSH key pair (our initial lead alert for **CreateKeyPair**) for an existing EC2 instance. This instance was not running at the time so the attacker turned it on via a **RunInstances** call. Ultimately, this series of actions resulted in command line access to the targeted EC2 instance, at which point visibility can be a challenge without additional OS logging or security products to investigate instance activity.

How did the attacker get credentials?

While frustrating, it's not always feasible to identify the root cause of an incident for a variety of reasons. For example, sometimes the technology simply doesn't produce the data necessary to determine the root cause. In this case, using the tech we had available to us, we weren't able to determine how the attacker gained credentials, but we have the following suspicions:

- Given multiple credentials were compromised, it's likely they were found in a public repository such as git, an exposed database or somewhere similar.
- It's also possible credentials were lifted from developer machines directly, for example the AWS credentials file.

We attempted to confirm these, but couldn't get to an answer in this case. Though unfortunate, it offers an opportunity to work with the victim to improve visibility.

For reference, below are the Mitre ATT&CK Cloud Tactics observed during Expel's response.

Initial Access	Valid Accounts
Persistence	Valid Accounts, Redundant Access
Privilege Escalation	Valid Accounts
Defensive Evasion	Valid Accounts
Discovery	Account Discovery

Cloud Security Threat Containment

By thoroughly scoping the attacker's activities, we were able to deliver clear remediation steps. This included:

- Deleting the compromised access keys for the eight involved IAM user accounts;
- Snapshotting (additional forensic evidence) and rebuilding the compromised EC2 instance;
- Deleting the SSH keys generated by the attacker;

- And deleting the hundreds of Security Group ingress rules created by the attacker.

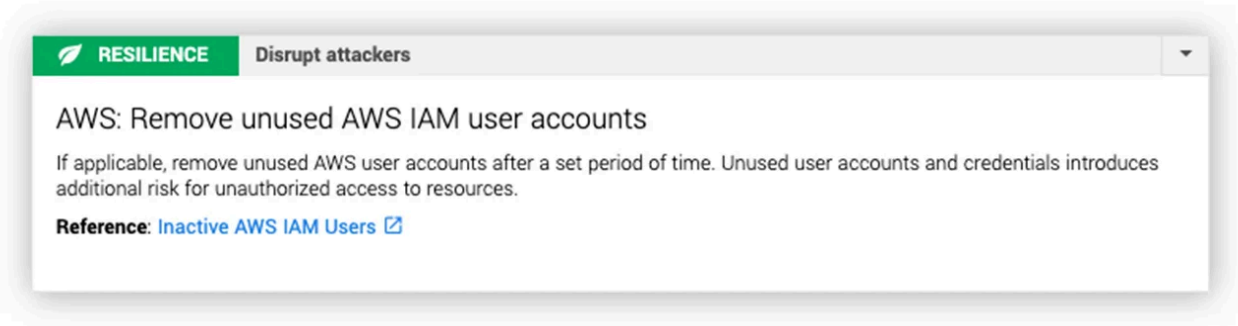
Resilience: Helping our customer improve their security posture

When we say incident response isn't complete without fixing the root of the problem – we mean it.

One of the many things that makes us different at Expel is that we don't just throw alerts over the fence. That would only be sort of helpful to our customers and puts us in a position where we'd have to tackle the same issue on another day … and likely on many more days after that.

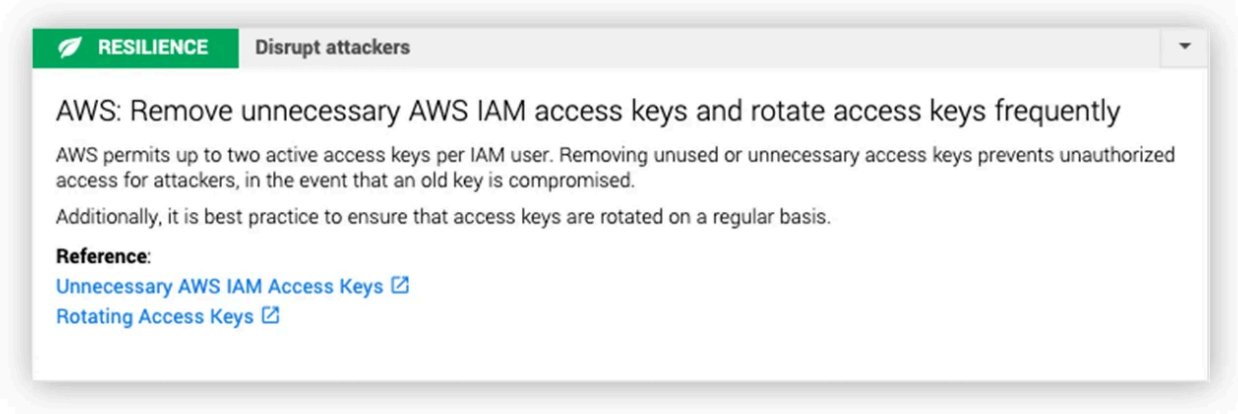
We're all about efficiency here. That's why we provide clear recommendations for how to solve issues and what actions a customer can take to prevent these kinds of attacks in the future. Everybody wins (except for the bad guys).

While we weren't certain how the access keys were compromised in the first place, below are the resilience recommendations we gave our customer once the issue was resolved.



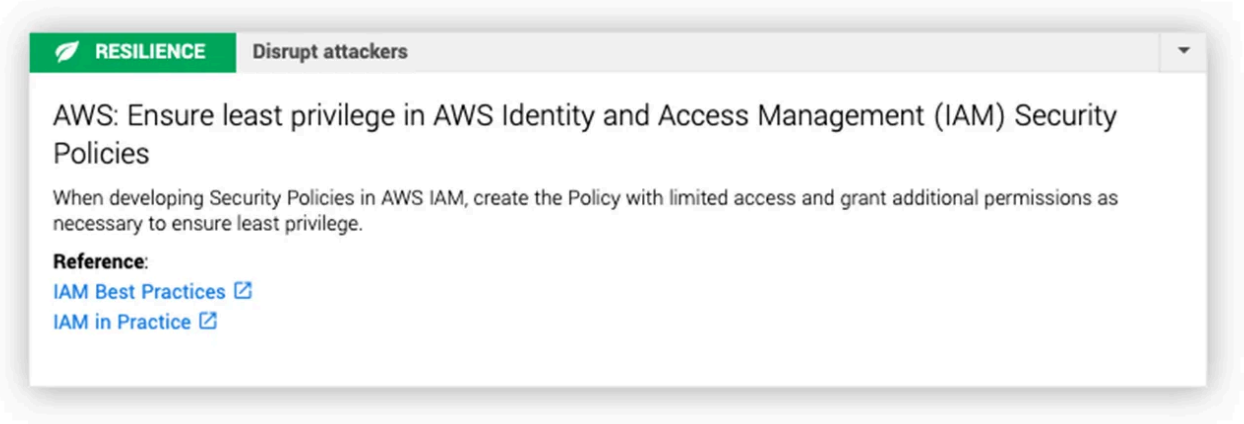
Expel AWS Resilience (1)

- If the IAM user is unused, then it probably doesn't need to remain active in your account. We made this recommendation because these access keys hadn't been in use by anyone other than the attacker in the previous 30 days.



Expel AWS Resilience (2)

- Since the access keys for this IAM principal were at least 30 days old given that no activity occurred from a legitimate user, it was time to do some tidying up, so to speak. If you need that user, rotate the access keys on a regular basis.



Expel AWS Resilience (3)

- We noticed that this IAM user had far too many EC2 permissions and thought this resilience measure was in order. We also shared that it would be far safer to delegate those EC2 permissions with an IAM role.

Lessons learned

Fortunately, we were able to disrupt this attack before there was any serious damage, but it highlighted the very real fact that cloud infrastructure – whether you're running workloads on AWS or somewhere else – is a prime target for attackers. As with every incident, we took some time to talk through what we discovered through this investigation and are sharing our key lessons here.

- **AWS customers must architect better security "in" the cloud.** That is, create greater visibility into the EC2 and other infrastructure identified in the shared responsibility model.
- You can't find evil if the analysts don't know what to look for **– train, train some more, and then when you're done training, train again.** Special thanks to Scott Piper (@0xdabbad00) and Rhino Security Labs (@RhinoSecurity) for their contributions to AWS security research.
- **While security in the cloud is still relatively in its infancy, the same can be said for the attacker behaviors –** much of what we observed here and in the past were elementary attack patterns.
- **There are additional automated enrichment opportunities.** We've started working on a new AWS robotic workflow to summarize historical API usage data for the IAM principal and will compare it to the access parameters of the alert.

Be on the lookout for an additional blog post in the future for our automated AWS alert enrichments. Until then, check out our other blogs to learn more about how we leverage AWS cloud security for our customers, along with tips and tricks for ramping up your own org's security when it comes to cloud.
