Incident report: From CLI to console, chasing an attacker in AWS

expel.com/blog/incident-report-from-cli-to-console-chasing-an-attacker-in-aws/

April 5, 2022

Recently, our SOC detected unauthorized access into one of our customer's Amazon Web Services (AWS) environments. The attacker used a long-term access key to gain initial access. Once they got in, they were able to abuse the AWS Identity and Access Management (IAM) service to escalate privileges to administrative roles and create two new users and access keys — creating a foothold in their environment. However, we stopped them before the attacker was able to get any further.

In this post, we'll walk you through how we spotted unauthorized access, the investigative steps we took to understand what the attacker did in AWS, and share our lessons learned and key takeaways from the incident.

Quick background

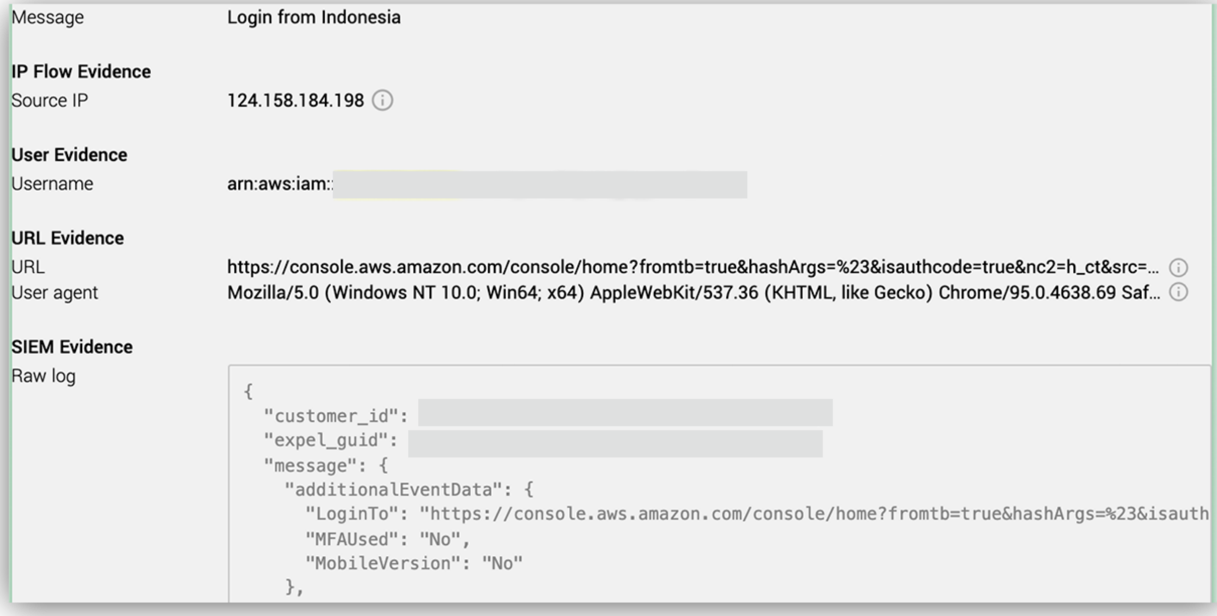
Before we tell you how it went down, for this customer we're ingesting AWS CloudTrail logs and applying our own custom detections. This customer did not have AWS GuardDuty enabled for their monitored AWS accounts — nor did we have any visibility beyond the AWS control plane.

Our initial lead

Our first clue into the incident was an AWS alert based on CloudTrail logs for a console login from an IAM user originating from an atypical country.

From the AWS alert (screenshot below), our SOC was able to extract the following details about the console login:

- The authentication originated from Indonesia
- The type of AWS account was an IAM user
- Multi-factor authentication (MFA) was not used for the authentication



Message Login from Indonesia

IP Flow Evidence

Source IP 124.158.184.198

User Evidence

Username arn:aws:iam::[redacted]

URL Evidence

URL https://console.aws.amazon.com/console/home?fromtb=true&hashArgs=%23&isauthcode=true&nc2=h_ct&src=...

User agent Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Saf...

SIEM Evidence

Raw log

```
{
  "customer_id": [redacted]
  "expel_guid": [redacted]
  "message": {
    "additionalEventData": {
      "LoginTo": "https://console.aws.amazon.com/console/home?fromtb=true&hashArgs=%23&isauth
      "MFAUsed": "No",
      "MobileVersion": "No"
    },
```

The details about the AWS console login prompted our SOC to ask the following questions:

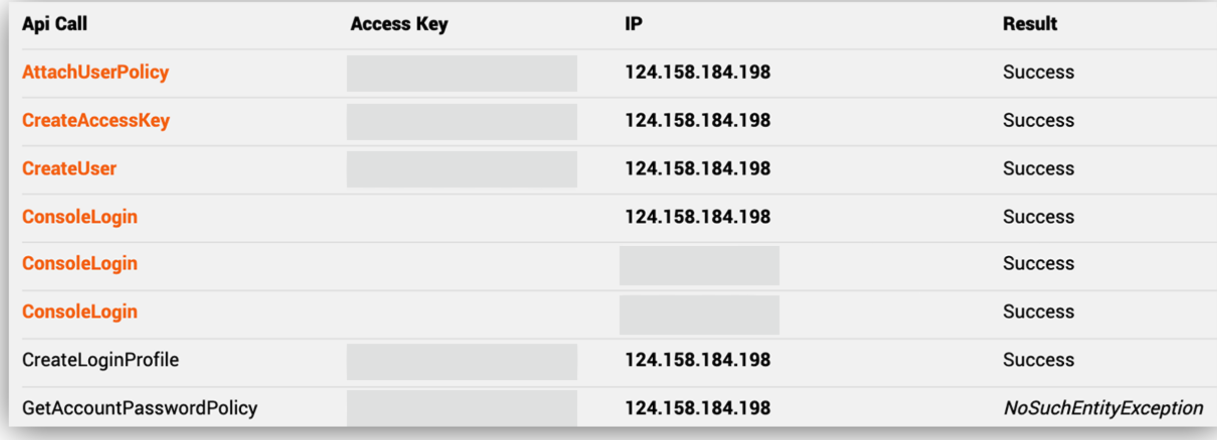
- Does this IAM user typically login from Indonesia?
- Why is this IAM user authenticating to the console directly and not via an identity provider?
- Why wasn't MFA used?

These questions combined certainly raised our suspicion.

SOC pro-tip: IAM accounts typically have long-term access keys associated with them. You'll see these long-term access keys start with "AKIA." Most of the AWS incidents we detect were the result of a publicly exposed long-term access key.

The first step in our investigative process was to understand what happened after the successful AWS console login. We used one of our bots, Ruxie™, to gain some more insight. As a quick refresher, our robot Ruxie (yes – we give our robots names) automates investigative workflows to surface up more details to our analysts.

We used Ruxie to list the most recent interesting API calls from the AWS account in the initial lead (interesting in this context is mostly anything that isn't Get*, List*, Describe*, and Head*). API calls that are sometimes associated with attacker activities in AWS are highlighted in orange. We highlight these actions in orange to provide a visual cue to our analysts that something may be amiss here.



Api Call	Access Key	IP	Result
AttachUserPolicy	[redacted]	124.158.184.198	Success
CreateAccessKey	[redacted]	124.158.184.198	Success
CreateUser	[redacted]	124.158.184.198	Success
ConsoleLogin		124.158.184.198	Success
ConsoleLogin		[redacted]	Success
ConsoleLogin		[redacted]	Success
CreateLoginProfile	[redacted]	124.158.184.198	Success
GetAccountPasswordPolicy	[redacted]	124.158.184.198	*NoSuchEntityException*

The list of API calls returned by Ruxie showed us that the source IP address associated with the atypical console login also issued API calls to CreateUser, CreateAccessKey, and AttachUserPolicy. For the AWS defenders out there, it's important to note that AWS accounts are assigned temporary access keys when authenticating to the AWS console (these access keys typically start with "ASIA").

Now this activity really has our attention. But why?

The CreateUser API call is used to create a new IAM user. The CreateAccessKey API call creates a new long-term access key for a specific user. AttachUserPolicy attaches a specified policy to a specified user.

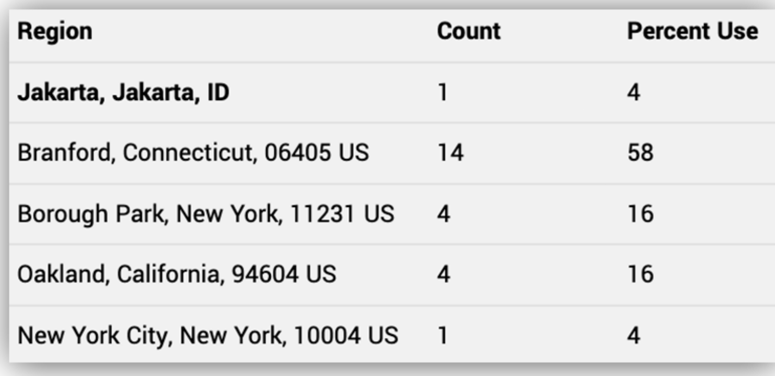
Therefore, an attacker can use these API calls to:

- Create a new IAM user
- Create a new long-term access key for the user
- Attach a highly privileged policy to the user for elevated access

This series of API calls can give an attacker persistent and elevated access in an AWS environment — yep, this activity certainly has our attention.

The next question our SOC asked was, "where does this IAM user typically authenticate from?" Ruxie to the rescue. Ruxie provided our analysts with a list of login activity by region and frequency for the IAM user listed in our lead alert.

Bottom line? Logins from Indonesia are highly suspicious.



Region	Count	Percent Use
Jakarta, Jakarta, ID	1	4
Branford, Connecticut, 06405 US	14	58
Borough Park, New York, 11231 US	4	16
Oakland, California, 94604 US	4	16
New York City, New York, 10004 US	1	4

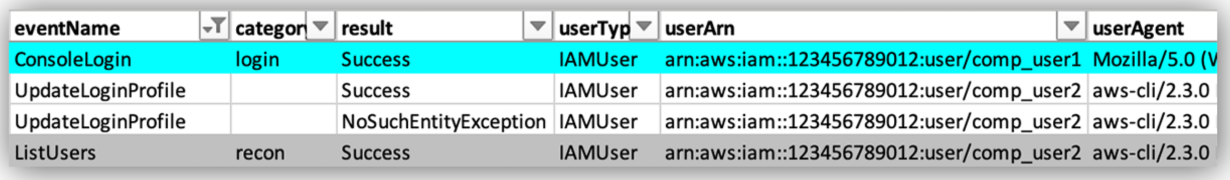
Recently authenticated regions Ruxie action for the source IAM user
OK, so at this point we have an IAM user logging in to the AWS console, from a location we've confirmed is atypical, and that account is issuing API calls to create new users, long-term access keys, and to attach highly privileged policies to users for elevated access.

At this point our SOC declared an incident, issued a recommendation to our customer to reset the (**arn:aws:iam::123456789012:user/comp_user1**) account credentials, and in parallel began working to answer:

- How did the attacker obtain AWS console creds?
- What else did the attacker do in AWS?

How did the attacker obtain AWS console creds?

The next step in our investigative process was to get a more detailed timeline of all AWS API calls issued by the source IP address associated with the lead alert for the console login. This investigative step helps us better understand the actions performed by the attacker.



eventName	categor	result	userTyp	userArn	userAgent
ConsoleLogin	login	Success	IAMUser	arn:aws:iam::123456789012:user/comp_user1	Mozilla/5.0 (V
UpdateLoginProfile		Success	IAMUser	arn:aws:iam::123456789012:user/comp_user2	aws-cli/2.3.0
UpdateLoginProfile		NoSuchEntityException	IAMUser	arn:aws:iam::123456789012:user/comp_user2	aws-cli/2.3.0
ListUsers	recon	Success	IAMUser	arn:aws:iam::123456789012:user/comp_user2	aws-cli/2.3.0

API calls in AWS triage timeline
When examining the detailed timeline of API calls, the attacker first issued a ListUsers API call using the **arn:aws:iam::123456789012:user/comp_user2** IAM user from the AWS command line interface (aws-cli). Note that this is a different IAM user from the lead alert. We inferred the **arn:aws:iam::123456789012:user/comp_user2** IAM user was also compromised since the API call originated from the same IP address recorded in the console login. More on this in a second.

Next, the attacker issued two calls to the UpdateLoginProfile API; one for the **arn:aws:iam::123456789012:user/comp_user1** IAM user (succeeded) and one for the **arn:aws:iam::123456789012:user/comp_user2** IAM user (failed with the NoSuchEntityException reason).
Finally, CloudTrail logs recorded a ConsoleLogin from the **arn:aws:iam::123456789012:user/comp_user1** account.

So as a quick recap, our investigation revealed the attacker took the following steps to gain access to the AWS console:

- Used the AWS access keys for **arn:aws:iam::123456789012:user/comp_user2** to issue a ListUsers API call.
  The results of the ListUsers API call returned **arn:aws:iam::123456789012:user/comp_user1** in the results.
- Issued an API call to UpdateLoginProfile to change the AWS console password for the **arn:aws:iam::123456789012:user/comp_user1** account.
- Authenticated into the AWS console using the **arn:aws:iam::123456789012:user/comp_user1** account.

It’s our working theory that the AWS access keys for the **arn:aws:iam::123456789012:user/comp_user2** account were discovered by the attacker in a publicly available code repository.

What else did the attacker do in AWS?

SOC pro-tip: At this point in our investigation, CloudTrail logs indicate the attacker has access to the AWS console for this customer. Therefore, we’re expecting to see attacker activity recorded from different IP addresses associated with AWS. (Don’t exclude these in your search!)

Immediately following the authentication to the AWS console, the attacker issued the following API calls:

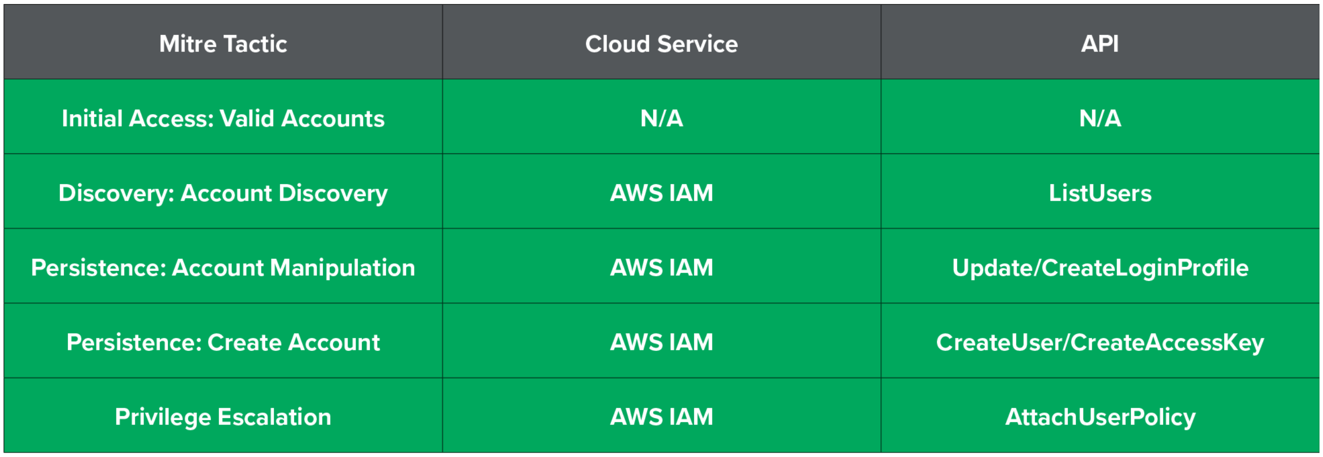
- CreateUser
- CreateAccessKey
- AttachUserPolicy
- CreateLoginProfile

This allowed the attacker to create two new IAM users with accompanying long-term access keys and attached a policy that allowed one of the new users to create and change their AWS console password. For one of the newly created IAM user accounts, the attacker issued an AttachUserPolicy API call to attach an AdministratorAccess policy to the newly created IAM user — allowing the attacker to elevate their privileges in AWS.

Lastly, the attacker used one of the newly created IAM users to make a call to the RequestServiceQuotaIncrease API in order to increase the EC2 quota. It's our opinion that this action was taken in preparation for starting multiple large EC2 instances for cryptocurrency mining. It is at this point that we worked with the customer to begin remediation.

Crisis averted!

Here's the final play-by-play of the actions taken by the attacker in AWS mapped to the MITRE ATT&CK framework:



Mitre Tactic	Cloud Service	API
Initial Access: Valid Accounts	N/A	N/A
Discovery: Account Discovery	AWS IAM	ListUsers
Persistence: Account Manipulation	AWS IAM	Update/CreateLoginProfile
Persistence: Create Account	AWS IAM	CreateUser/CreateAccessKey
Privilege Escalation	AWS IAM	AttachUserPolicy

MITRE tactics mapped to AWS API calls

You can check out additional AWS API calls we've seen associated with attacker activity in our AWS mind map.

Remediation in AWS

With the scope of the compromise understood, we provided our customer with the following remediation recommendations:

1. Delete the two created accounts and accompanying access keys for **arn:aws:iam::123456789012:user/created_user1** and **arn:aws:iam::123456789012:user/created_user2**
2. Deactivate the long term access keys associated with the **arn:aws:iam::123456789012:user/comp_user1** and **arn:aws:iam::123456789012:user/comp_user2** IAM users
3. Reset the AWS console password for the **arn:aws:iam::123456789012:user/comp_user1** and **arn:aws:iam::123456789012:user/comp_user2** IAM users

We also recommended that the customer ensure AWS access keys are not being accidentally released to the public and implement least privilege with regards to AWS users. Additionally, we recommended the customer implement MFA for IAM user AWS console authentications.

Lessons learned

What stood out the most in this incident was the attacker's technique to use an exposed long-term access key to gain access to the AWS console — potentially for ease of access and persistence. Simply put, the attacker wants to go from the aws-cli to the AWS console. To achieve this, the attacker issued API calls UpdateLoginProfile or CreateLoginProfile from an IAM user. We now have a detection based on CloudTrail logs that alerts our SOC anytime we see these specific API calls originating from an IAM user where the User-Agent is the aws-cli. We've added some additional logic to reduce benign noise associated with AWS IP addresses.

We're also exploring additional detections based on CloudTrail logs where we see newly created IAM users issue API calls to RequestServiceQuotaIncrease to increase the EC2 quota. This could be a signal of potential unauthorized activity in EC2.

While we detected unauthorized activity early in the attack lifecycle, we're a learning organization and always looking for opportunities to improve our detection and response capabilities in AWS.