Case Study: The DangerDev@protonmail.me Investigation

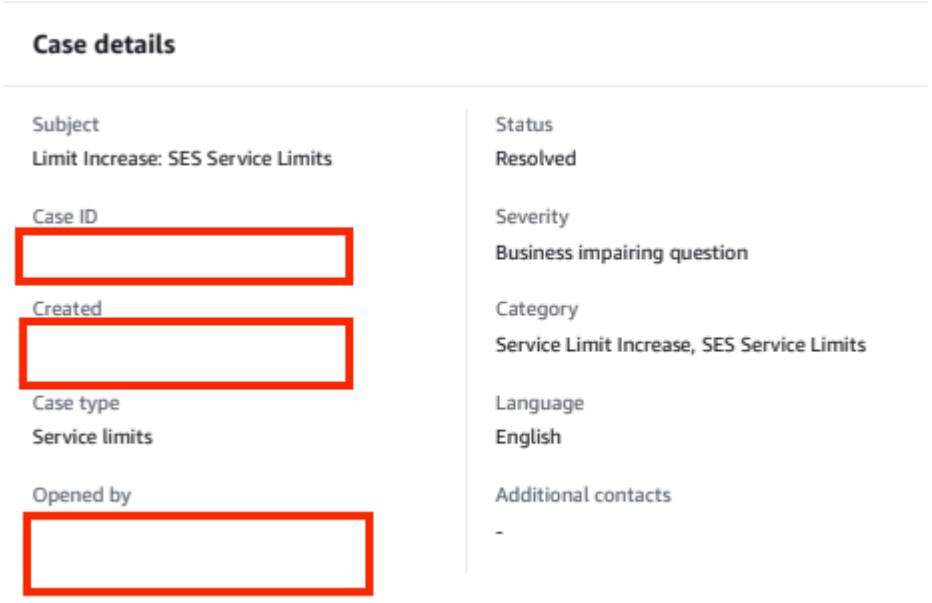
Archived: 2026-04-05 23:06:21 UTC

An AWS incident response story

Recently we worked on a very interesting incident response case in a customer's AWS environment. We would like to share the story of this case in more detail in this blog including the techniques used by the threat actor (TA). In this blog, we want to share a detailed story of this case, including the techniques used by the TA. We hope that this is helpful for people protecting AWS accounts around the world.

Background

It all started on a Friday afternoon when we got a call asking for support with an ongoing AWS incident. The trigger for the incident was a suspicious support case that was created within one of the AWS accounts of our client. The support case wasn't raised by the client themselves and the reason it triggered an alert from AWS to the client is because it was a request to increase Simple Email Service (SES) sending limits. However our client wasn't using SES….



Due to client confidentiality we have censored certain information.

SES is a popular target for attackers as it can be abused to send out phishing and spam campaigns at massive rates and from a trusted sender (Amazon). We talked about it before in an incident write-up there's also an excellent blog on SES abuse by Permiso Security.

Reading Guide

Please consider that cloud attack techniques are challenging to categorize into specific MITRE phases due to their multipurpose nature and the ambiguity of threat actor intent.

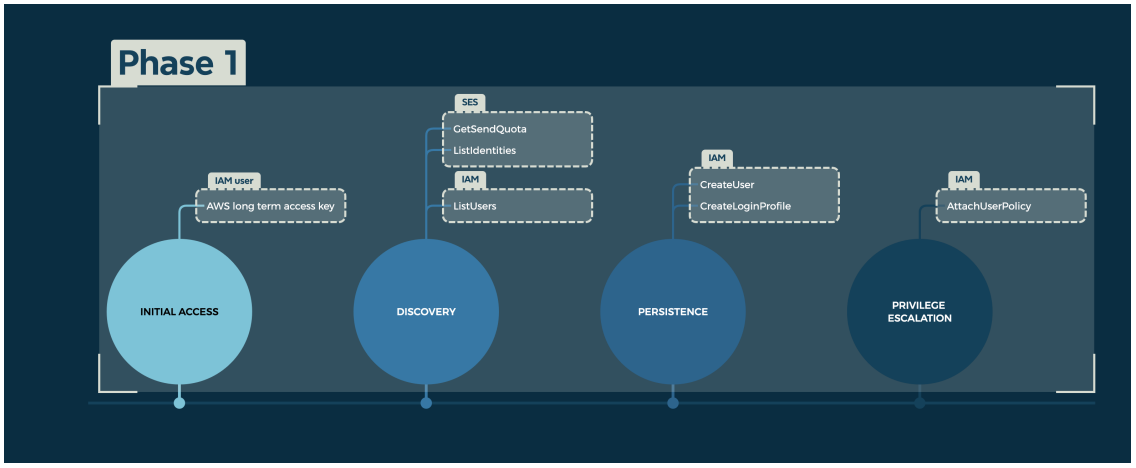
For instance, an attacker's actions related to user activities may fit into the persistence phase, but could also serve to evade defenses by blending in with specific usernames and roles. Additionally, threat actors often move between phases, such as transitioning from traditional persistence activities to subsequent discovery activities with newly created users. The story is written in chronological order.

Incident overview

The malicious activity took places over the course of a month. Within that month there were 3 distinguished phases of activity. We have separated this write-up in the three phases of the attack and used the MITRE ATT&CK framework techniques to categorize our findings.

Phase 1

The support case that triggered the incident response was opened by an IAM user called DangerDev@protonmail.me. According to our client this was not a legitimate user, time to figure out how this all happened. (**Tip**: Right-click open in new tab for high quality)



Initial access

Access to the environment was achieved through an accidentally exposed long term access key belonging to an IAM user. The access key belonged to a user with Administrator Access. We cannot provide additional details as to where the access key was stored.

Discovery

What do you do if you just received a fresh access key to start your hacking adventure with?

SES discovery

If you think the answer is discovering and enumerating SES you are correct. Repeated calls were made towards SES with GetSendQuota and ListIdentities.These calls are used to get an idea on how much emails can be sent at once and which emails and domains are registered to send emails.

The SES activity occurred on two separate days and was the only observed activity in the first two weeks, which was interesting. In other engagements we have seen that after discovery of an active access key the TA will launch whatever they can immediately before they get kicked out.

User discovery

After approximately two weeks the TA came back and ran ListUsers to list the IAM users in the AWS account.

The discovery commands were most likely automated based on the user-agent and the frequency of the calls performed.

An observation for this phase is that the TA isn't running the typical enumeration commands like GetCallerIdentity and ListAttachedUserPolicies. This could be because calls like that could trigger detection.

Persistence

After all of the above it's time for our main character to make an entrance. A CreateUser call was made for a new account DangerDev@protonmail.me.


```
{ [-]
    awsRegion: us-east-1
    eventCategory: Management
    eventID: 482c4578-44e1-4522-92d9-6521ad24bcb9
    eventName: CreateUser
    eventSource: iam.amazonaws.com
    eventTime: [redacted]
    eventType: AwsApiCall
    eventVersion: 1.08
    managementEvent: true
    readOnly: false
    recipientAccountId: [redacted]
    requestID: 495a83e6-b393-4223-bf73-dbbc0e12d13c
    requestParameters: { [-]
      userName: DangerDev@protonmail.me
    }
    responseElements: { [+]
    }
    sourceIPAddress: 107.151.188.91
```

After this action the TA performed a `CreateLoginProfile` call which is used to give a user the ability to login through the AWS management console.


```
{ [-]
   awsRegion: us-east-1
   eventCategory: Management
   eventID: 4c7f92b6-bbdf-4078-88fc-2934164dc9df
   eventName: CreateLoginProfile
   eventSource: iam.amazonaws.com
   eventTime: [redacted]
   eventType: AwsApiCall
   eventVersion: 1.08
   managementEvent: true
   readOnly: false
   recipientAccountId: [redacted]
   requestID: 93cf59cd-993a-43e2-85a2-d14ee20e8fc8
   requestParameters: { [-]
     passwordResetRequired: false
     userName: DangerDev@protonmail.me
   }
   responseElements: { [+]
   }
   sourceIPAddress: 107.151.188.91
```


Privilege Escalation

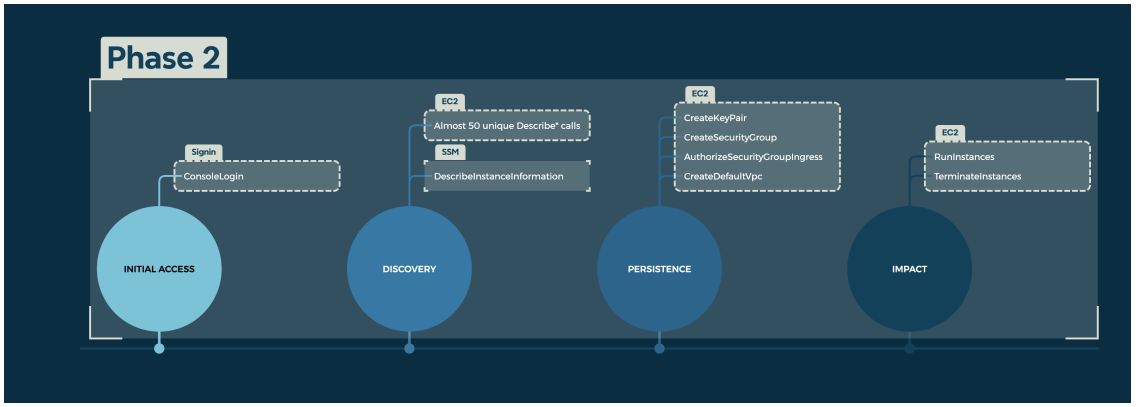
The `AdministratorAccess` policy was attached to the newly created account with `AttachUserPolicy`. This policy is an AWS managed policy that provides full access to AWS services and resources.

```
{ [-]
    awsRegion: us-east-1
    eventCategory: Management
    eventID: a6640cfc-0c4a-4a2f-b745-f59a43f2a33f
    eventName: AttachUserPolicy
    eventSource: iam.amazonaws.com
    eventTime: [redacted]
    eventType: AwsApiCall
    eventVersion: 1.08
    managementEvent: true
    readOnly: false
    recipientAccountId: [redacted]
    requestID: 443fb542-1500-40f8-8372-d2f11af74ef5
    requestParameters: { [-]
      policyArn: arn:aws:iam::aws:policy/AdministratorAccess
      userName: DangerDev@protonmail.me
    }
    responseElements: null
    sourceIPAddress: 107.151.188.91
```


This activity marked the end of the activity with the original IAM user and most of the subsequent activity was performed with the DangerDev user and some new identities will enter this story.

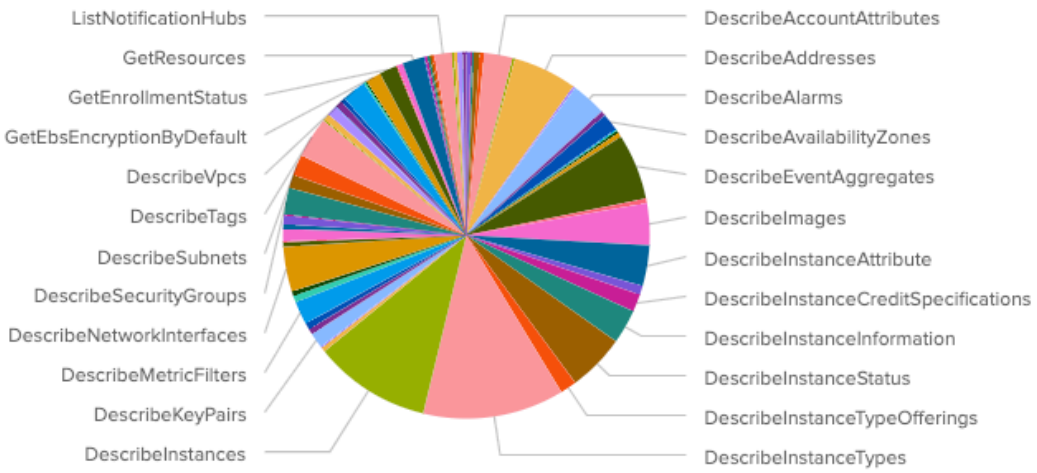
Phase 2

In the second phase of the attack that lasted approximately one week the TA was mostly testing out their access and what kind of activities they could perform within the environment.



Discovery

With the newly created account the TA performed additional discovery activities which were probably closer aligned with their malicious intentions. The following calls were made in less than an hour.



The majority of this activity relates to EC2 instances:

Using the `sessionCredentialFromConsole` field we can identify activity performed through the AWS management console. Which is quite interesting as it's less likely this activity was scripted. This example shows a `DescribeSecurityGroups` event which lists all security groups in the account, which is a nice bridge to the next phase of the attack.

Persistence

After the discovery activity the TA performed the following actions in a timespan of 30 minutes:

- CreateKeyPair
- CreateSecurityGroup
- CreateDefaultVpc
- AuthorizeSecurityGroupIngress
- RunInstances

In short what happened was that the TA launched a EC2 instance and as part of that process a VPC with a security group was created. The TA modified the security group to allow for external RDP access as shown below:

We will discuss the EC2 instance creation next, because the TA did something interesting.

Impact

What we saw was that the TA created a test instance first with instance type t2.micro, which is one of the smallest instances and definitely not suitable for crypto mining. It seems the TA wanted to test if they could successfully launch and access their EC2 machine, because shortly after the instance was terminated by the TA.

	eventName ▲
03:22:01	RunInstances
03:40:43	TerminateInstances

After this test it was time for the heavy hitters. The TA launched three instances with instance type p3.16xlarge.

	eventName ⇕	requestParameters.instanceType ⇕
03:38:32	RunInstances	p3.16xlarge
03:43:26	RunInstances	p3.16xlarge
03:46:08	RunInstances	p3.16xlarge

This instance type is much better for crypto mining as it has a GPU with 128GB and 64 vCPUs. However, there was a problem with launching the instance, due to account limits.


```
{ [-]
   awsRegion: us-east-1
   errorCode: Client.VcpuLimitExceeded
   errorMessage: You have requested more vCPU capacity than your current vCPU limit of 64 allows for the instance bucket that the specified instance type
belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit.
   eventCategory: Management
   eventID: a0f0b3d0-564f-4540-87b8-8cecd19788be
   eventName: RunInstances
   eventSource: ec2.amazonaws.com
   eventTime: [redacted]
   eventType: AwsApiCall
   eventVersion: 1.08
   managementEvent: true
   readOnly: false
   recipientAccountId: [redacted]
   requestID: b44a1ae4-1767-4a4d-8c89-51280717babf
   requestParameters: { [+]
   }
   responseElements: null
   sessionCredentialFromConsole: true
   sourceIPAddress: 107.151.188.91
   userAgent: AWS Internal
   userIdentity: { [+]
   }
}
```


The other machine launched successfully but was terminated after approximately one hour. After this activity it stayed mostly quiet for another two weeks.

Phase 3

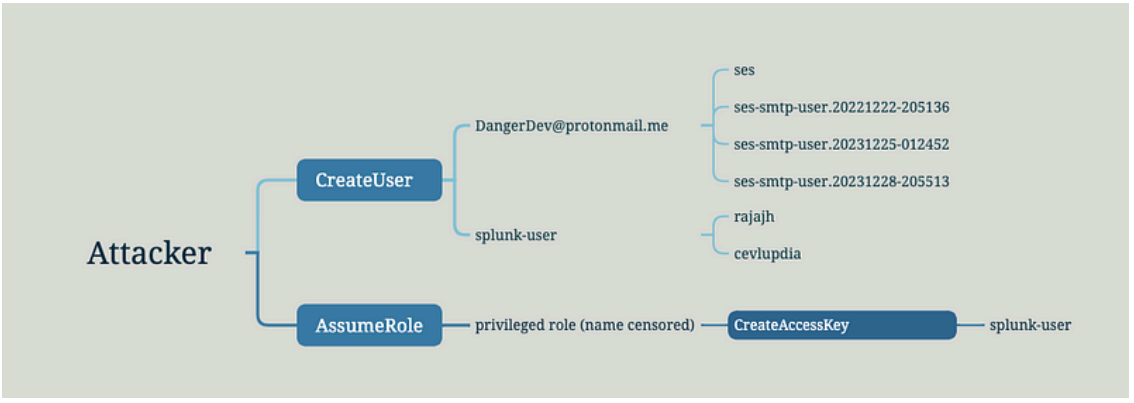
The bulk of the activity took place in the last phase of the attack and some of the actions ultimately led to discovery of the attack.

Persistence & Defense Evasion

The majority of the activities is related to user and role creation or modification. The TA used an interesting technique to achieve persistent access into the AWS account.

User creation

Over the course of this attack the TA performed a number of activities related to users and roles. The graphic below is intended to show you the activities that were performed.



The TA manually created a user account called ses.

The username `ses` is interesting, because it mimics accounts automatically created when using SES. They can be identified because they all follow the official name convention `ses-smtp-user.<date-time>` and in the event you can see it's invoked by SES console

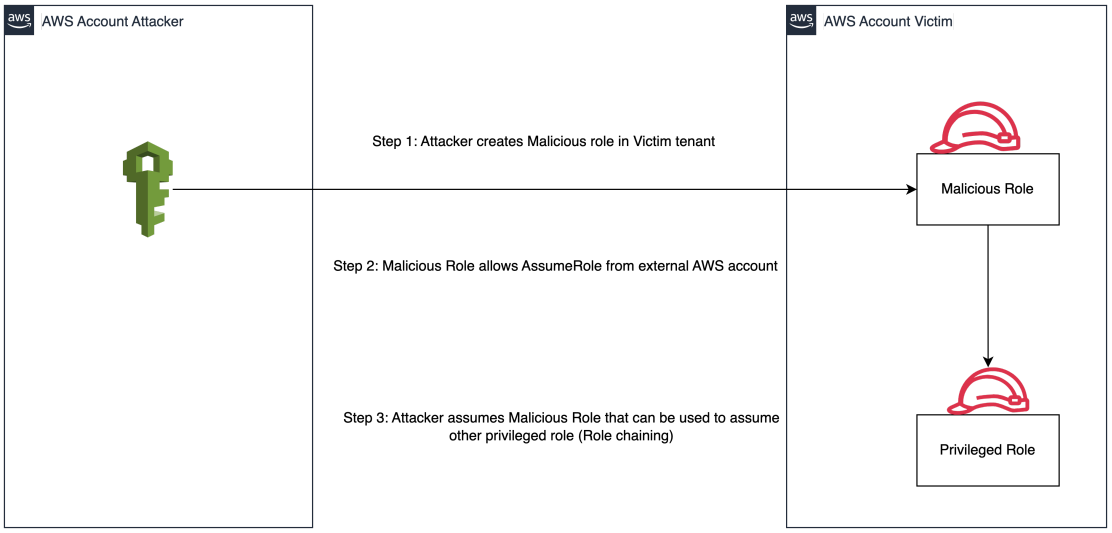
Therefore the creation of an account with the name [ses] might also be an attempt to evade detection.

Additionally the TA created access keys for existing accounts, due to confidentiality we can't name the accounts in question. We've added an example of this activity for the [ses] account below.

```
{ [-]
    awsRegion: us-east-1
    eventCategory: Management
    eventID: d98052fd-e94d-4da0-9347-0a766aa1b28c
    eventName: CreateAccessKey
    eventSource: iam.amazonaws.com
    eventTime: [redacted]
    eventType: AwsApiCall
    eventVersion: 1.08
    managementEvent: true
    readOnly: false
    recipientAccountId: [redacted]
    requestID: 7149baa3-dd5b-4da4-a3cd-c83574fe3deb
    requestParameters: { [-]
      userName: ses
    }
    responseElements: { [+]
    }
    sessionCredentialFromConsole: true
    sourceIPAddress: 112.215.253.179
    userAgent: AWS Internal
    userIdentity: { [+]
    }
}
```


Role creation

One of the more interesting actions observed in this attack is the creation of a role that allows identities from an external AWS account to assume a privileged role in the victim tenant. Which sounds quite complicated, but this is how it works:



And what it looks like in CloudTrail:


```
{ [-]
   awsRegion: us-east-1
   eventCategory: Management
   eventID: 390058d2-70f6-46f5-9e31-6e906e5250b3
   eventName: CreateRole
   eventSource: iam.amazonaws.com
   eventTime: 2
   eventType: AwsApiCall
   eventVersion: 1.08
   managementEvent: true
   readOnly: false
   recipientAccountId:
   requestID: c10ecd8e-59d2-44e6-b428-45cf5110a91f
   requestParameters: { [-]
     assumeRolePolicyDocument: {"Version":"2012-10-17","Statement":[{"Effect":"Allow","Action":"sts:AssumeRole","Principal":
{"AWS":"265857590823"},"Condition":{}}]}
     description:
     roleName: AWSLanding-Zones-ConfigRecorderRoles
   }
   responseElements: { [+]
   }
   sessionCredentialFromConsole: true
   sourceIPAddress: 140.213.103.106
   userAgent: AWS Internal
   userIdentity: { [+]
   }
}
```


Notice the external AWS account ID and also the roleName. The roleName is AWSLanding-Zones-ConfigRecorderRoles which was very similar to an existing role name.

The second role has the same purpose, but for a different external AWS account it also has a name very similar to an existing role.

AWSeservedSSO_* vs. AWS**R**eservedSSO_*

```
{ [-]
   awsRegion: us-east-1
   eventCategory: Management
   eventID: 97bbcce9-b1a5-4992-90c4-859a00f33325
   eventName: CreateRole
   eventSource: iam.amazonaws.com
   eventTime: [redacted]
   eventType: AwsApiCall
   eventVersion: 1.08
   managementEvent: true
   readOnly: false
   recipientAccountId: [redacted]
   requestID: 9c61e7a5-c64c-4609-b8d0-b5f816bf7917
   requestParameters: { [-]
     assumeRolePolicyDocument: {"Version":"2012-10-17","Statement":[{"Effect":"Allow","Action":"sts:AssumeRole","Principal":
{"AWS":"671050157472"},"Condition":{}}]}
     description:
     roleName: AWSeservedSSO_PowerUserAccess_[redacted]
   }
   responseElements: { [+]
   }
   sessionCredentialFromConsole: true
   sourceIPAddress: 140.213.98.125
   userAgent: AWS Internal
   userIdentity: { [+]
   }
}
```


After the creation of both roles, an AssumeRole event was observed as shown below. The TA assumed the role from their own account (671050157472).


```
{ [-]
   awsRegion: us-east-1
   eventCategory: Management
   eventID: 915e711e-f583-39e9-8b58-bbecf72ab4da
   eventName: AssumeRole
   eventSource: sts.amazonaws.com
   eventTime: [redacted]
   eventType: AwsApiCall
   eventVersion: 1.08
   managementEvent: true
   readOnly: true
   recipientAccountId: [redacted]
   requestID: 14cd039e-4a8b-4c5f-b4f6-b60ed5a6455e
   requestParameters: { [-]
     roleArn: arn:aws:iam::[redacted]:role/AWSeservedSSO_PowerUserAccess_a8f6ee25857da56d
     roleSessionName: forroles
   }
   resources: [ [+]
   ]
   responseElements: { [+]
   }
   sharedEventID: abc6dd46-5841-498e-af71-d6c356f0185b
   sourceIPAddress: AWS Internal
   userAgent: AWS Internal
   userIdentity: { [-]
     accountId: 671050157472
     invokedBy: AWS Internal
     principalId: AIDAZYPNUZWQDPJYWT6YM
     type: AWSAccount
   }
}
```

If you see any of these accounts in your environment please reach out to us or start your incident response process as they're confirmed malicious by AWS:

- 265857590823
- 671050157472

We haven't seen this type of attack technique before, it's a pretty clever way of establishing backdoor access that doesn't require an IAM user inside the victim account.

Privilege escalation

In addition to the aforementioned activities such as the creation of users and roles with privileged access the TA performed activities that could be classified as attempts to escalate privileges.

Using AttachRolePolicy the TA added the AWS managed AdministratorAccess policy to the AssumeRole that allows for external access:


```
{ [-]
   awsRegion: us-east-1
   eventCategory: Management
   eventID: f9f95c3d-214b-4508-a119-8bd55add7b30
   eventName: AttachRolePolicy
   eventSource: iam.amazonaws.com
   eventTime: [redacted]
   eventType: AwsApiCall
   eventVersion: 1.08
   managementEvent: true
   readOnly: false
   recipientAccountId: [redacted]
   requestID: e3672db7-e0a4-48d9-a13b-c797b7624512
   requestParameters: { [-]
      policyArn: arn:aws:iam::aws:policy/AdministratorAccess
      roleName: AWSeservedSSO_PowerUserAccess_a8f6ee25857da56d
   }
   responseElements: null
   sessionCredentialFromConsole: true
   sourceIPAddress: 140.213.98.125
   userAgent: AWS Internal
   userIdentity: { [+]
   }
}
```


Another interesting event is UpdateLoginProfile where the TA uses the initially compromised account to update the console password of another account.

Discovery

In this phase the TA also performed discovery activities such as:

- ListBuckets
- ListGroupsForUser
- ListInstanceProfiles
- ListSSHPublicKeys
- **SimulatePrincipalPolicy**

Most of the above actions are pretty self-explanatory, however we want to highlight SimulatePrincipalPolicy. As this is a technique not reported on before by anyone (or at least we couldn't find it).

So how does this work, let's start with the event that is generated first..

The AWS Policy Simulator allows users to test an existing policy recorded in the policySourceArn field against a set of actions recorded in the actionNames field. This helps answer the question can I perform action X with policy Y.

The fact that the TA used this service to test certain actions tells us that they were interested in actions related to the SSM service and AWS Secrets Manager.

Defense evasion

Interestingly enough the TA actor put quite some effort into hiding their traces:

- Removing IAM users with DeleteUser
- Cleaning up policies with DetachUserPolicy and DeleteUserPolicy
- Deactivating long term access keys with UpdateAccessKey
- Cleaning up long term access keys with DeleteAccessKey
- Inspecting GuardDuty findings with ListFindingsand GetFindings
- Creating a LightSail instance upon discovery with CreateInstances

It was interesting to see that the TA modified users and access keys during the attack before they were discovered. It shows that they wanted to stay under the radar for a little longer. We would like to focus on the GuardDuty and LightSail actions as these are less commonly observed and offer some great insights.

GuardDuty

Looking at the GuardDuty related activities, we believe it was one action performed by the TA that resulted in the events below:

	eventName	userAgent
10:54:36	ListDetectors	RDS Console, aws-internal/3 aws-sdk-java/1.11.
10:54:37	GetDetector	RDS Console, aws-internal/3 aws-sdk-java/1.11.
10:54:37	ListFindings	RDS Console, aws-internal/3 aws-sdk-java/1.11.
10:54:37	GetFindings	RDS Console, aws-internal/3 aws-sdk-java/1.11.

What was interesting is the user-agent being Amazon Relational Database Service (RDS) console, it seems that the TA accessed GuardDuty from the RDS console. This is also visible in the ListFindings event which is filtered for findings related to the RDS resources.

LightSail

For those who primarily use EC2 or ECS for compute there's another compute resource in AWS that threat actors target. It's often overlooked as it isn't part of the regular AWS compute offering nor does it integrate with IAM. It's called LightSail and it's basically a virtual private server offering.

What happened is that our client started removing access for the TA actor. However, at this point they didn't yet know that the TA created an access key for another user. So when the TA noticed access was lost to their account they quickly used another account to create a LightSail instance, this resulted in an error because the account wasn't verified.

```
{ [-]
   awsRegion: us-west-2
   errorCode: InvalidInputException
   errorMessage: We're sorry, your AWS account is pending verification. Please try again later.
   eventCategory: Management
   eventID: 56bd17bb-d2de-417f-a0af-f49e7590cb70
   eventName: CreateInstances
   eventSource: lightsail.amazonaws.com
   eventTime: [redacted]
   eventType: AwsApiCall
   eventVersion: 1.08
   managementEvent: true
   readOnly: false
   recipientAccountId: [redacted]
   requestID: e01eef2c-ca9b-4352-a78b-4644e1bed7f6
   requestParameters: { [+]
   }
   responseElements: null
   sessionCredentialFromConsole: true
   sourceIPAddress: 114.122.132.171
   userAgent: AWS Internal
   userIdentity: { [+]
   }
}
```


Approximately an hour later the request went through successfully and the TA was able to launch a LightSail instance.

```
{ [-]
    awsRegion: us-east-1
    eventCategory: Management
    eventID: 4ae7dfd3-c677-4ce5-a386-c248803f6a07
    eventName: CreateInstances
    eventSource: lightsail.amazonaws.com
    eventTime: 1[redacted]
    eventType: AwsApiCall
    eventVersion: 1.08
    managementEvent: true
    readOnly: false
    recipientAccountId: [redacted]
    requestID: dc14c33b-c26c-4918-bd01-7f8392f1a38a
    requestParameters: { [-]
      availabilityZone: us-east-1a
      blueprintId: windows_server_2019/1
      bundleId: 2xlarge_win_3_0
      instanceNames: [ [+]
      ]
      keyPairName: LightsailDefaultKeyPair
      tags: [ [+]
      ]
      userData:
    }
    responseElements: { [+]
    }
    sessionCredentialFromConsole: true
    sourceIPAddress: 114.122.132.171
    userAgent: AWS Internal
    userIdentity: { [+]
    }
```


The TA accessed the associated RDP settings. Access was soon revoked as the instance was deleted within an hour. We were not able to investigate the LightSail to determine what happened within that hour.

Impact

No story is complete without some impact, in this case there was quite a few things the TA did which gave us some insights into their objectives. Roughly speaking we can categorize these into three actions:

1. Cryptomining
2. Phishing and spam through SES
3. Setting up fake domains for spear phishing and scams

Cryptomining

Luckily it didn't last long, because this activity closely preceded the initial discovery of the incident. But the TA actor did create several powerful and expensive instances in the AWS account. All of the below instanceTypes have GPU's enabled and significant CPU power.

eventName	requestParameters.instanceType
RunInstances	g4dn.8xlarge
RunInstances	p3.8xlarge
RunInstances	p3.16xlarge
RunInstances	p3.16xlarge
RunInstances	p3.16xlarge
RunInstances	p3.16xlarge

The instances weren't available for investigation and no VPC flow logs were available to perform further analysis.

To access the machines the TA created new inbound rules to allow traffic on port 22:

eventName	fromPort	ipRange
AuthorizeSecurityGroupIngress	22	0.0.0.0/0
AuthorizeSecurityGroupIngress	22	0.0.0.0/0
AuthorizeSecurityGroupIngress	22	0.0.0.0/0

Phishing and spam through SES

The TA was mostly interested in SES for further malicious activity. We can't share too many details on the emails that were sent out. However they were mostly aimed at individuals to phish for cryptocurrency exchange credentials and general spam.

What is interesting and what ultimately led to discovery of the incident is that the AWS Trust & Safety team communicated with the TA through a support case. Here's what happened:

1. TA requests increase in SES sending quoata (to send more spam)
2. AWS requests more details
3. TA responds in the support case
4. Quota increased by AWS

As part of the SES abuse the TA created the following identities with CreateEmailIdentity.

requestParameters.emailIdentity	responseElements.identityType
address-csv.com	DOMAIN
daihuku.me	DOMAIN
	DOMAIN
fundmade.jp	DOMAIN
goku.link	DOMAIN
in-telligent.com	DOMAIN
insale.jp	DOMAIN
invoice@ec-japan.jp	EMAIL_ADDRESS
mark@shift-strategies.ca	EMAIL_ADDRESS
ondemand.gift	DOMAIN
postly.jp	DOMAIN

Most of the domains are targeting Japanese websites, which is pretty interesting in itself in regards to attribution.

Fake domains mimicking PayPal

The TA also knew his way inside Amazon Route 53, four domains were created with `RegisterDomain`.

eventName	requestParameters.domainNameInfo.domainName.name
RegisterDomain	3dlntl-paypal.com
RegisterDomain	3dlntlverify.com
RegisterDomain	3dlntlpaypalcard.com
RegisterDomain	paypal-lntl.com

There's no need to guess what these domains were intended to be used for. The domains were short lived and quickly taken offline, which brings us to the end of the malicious activity observed.

The threat actor

We noticed that there's not a lot of threat intelligence on cloud threat actors. While we were completing this write-up Datadog published an excellent incident write-up which has some overlapping indicators for the ASN and IP addresses used by the TA. We've added all the IOCs below, if you're working for an organization in the TI field and have more information please do reach out.

With that in mind, some observations from our side on the TA:

- They know their way around AWS they go beyond the basic EC2 abuse and have skills to setup (more advanced) persistence methods;
- Uses a combination of automation scripts and hands-on keyboard activity. For example the testing and enumeration of AWS access keys was definitely automated as we kept seeing the original access keys being used to make calls on set times. However they also performed lots of activity through the management console;
- Some OPSEC was observed, as an example the TA didn't use the `GetCallerIdentity` command. Which is commonly observed in attacks as it's basically a `whoami` for AWS environments. Additionally we didn't see many failed API calls, which is often an indicator for a less skilled TA trying every possible action;

- Financially motivated, ultimately their goal seemed to be to perform (spear)phishing for financial gain through PayPal lures and cryptocurrency phishing;
- The TA was mostly using Indonesian based IP addresses outside of commercial VPN solutions.

Conclusion

We hope you made it this far, we know it's quite the read, but we believe this is a story worth sharing with all the technical details. There are lessons to be learned from this incident, which we will save for another blog post.

Last but definitely not least, we want to thank our client for allowing us to write this story. Your willingness to share this information will help others.

Indicators of Compromise

	Type	**Indicator**
	IAM user	DangerDev@protonmail.me
	IAM user	rajajh
	IAM user	cevlupdia
	IAM user	ses
	IP Address	45.141.215.56
	IP Address	37.19.205.154
	IP Address	212.102.51.243
	IP Address	212.102.51.242
	IP Address	172.86.96.166
	IP Address	140.213.52.30
	IP Address	140.213.51.83
	IP Address	140.213.49.11
	IP Address	140.213.47.147
	IP Address	140.213.45.145
	IP Address	140.213.22.46
	IP Address	140.213.22.245
	IP Address	140.213.105.43

	Type	Indicator
	IP Address	140.213.104.41
	IP Address	140.213.104.172
	IP Address	140.213.104.12
	IP Address	140.213.103.218
	IP Address	140.213.103.218
	IP Address	140.213.103.106
	IP Address	140.213.103.106
	IP Address	140.213.102.80
	IP Address	140.213.102.107
	IP Address	140.213.101.161
	IP Address	140.213.100.197
	IP Address	140.213.100.13
	IP Address	138.199.53.239
	IP Address	138.199.22.105
	IP Address	114.122.132.171
	IP Address	112.215.253.179
	IP Address	112.215.208.219
	IP Address	108.181.27.205
	IP Address	107.151.188.91
	IP Address	104.28.250.136
	IP Address	104.28.250.135
	IP Address	104.28.218.136
	IP Address	140.213.99.249
	IP Address	140.213.98.193
	IP Address	140.213.98.125
	IP Address	140.213.51.240

	Type	Indicator
	IP Address	140.213.49.33
	IP Address	140.213.49.247
	IP Address	140.213.47.253
	IP Address	140.213.47.116
	IP Address	140.213.45.86
	IP Address	140.213.45.43
	IP Address	140.213.45.223
	IP Address	140.213.45.192
	IP Address	140.213.45.148
	IP Address	140.213.43.96
	IP Address	140.213.43.91
	IP Address	140.213.43.75
	IP Address	140.213.43.62
	IP Address	140.213.43.30
	IP Address	140.213.43.213
	IP Address	140.213.41.83
	IP Address	140.213.39.93
	IP Address	140.213.39.220
	IP Address	140.213.37.94
	IP Address	140.213.37.56
	IP Address	140.213.37.52
	IP Address	140.213.37.206
	IP Address	140.213.37.190
	IP Address	140.213.24.82
	IP Address	140.213.24.101
	IP Address	140.213.22.36

	Type	Indicator
	IP Address	140.213.22.201
	IP Address	140.213.22.16
	IP Address	140.213.22.143
	IP Address	140.213.18.44
	IP Address	140.213.18.249
	IP Address	140.213.18.201
	IP Address	140.213.18.169
	IP Address	140.213.18.150
	IP Address	140.213.18.137
	IP Address	140.213.18.112
	IP Address	140.213.16.70
	IP Address	140.213.16.17
	IP Address	140.213.16.130
	IP Address	140.213.105.30
	IP Address	140.213.105.3
	IP Address	140.213.105.223
	IP Address	140.213.105.188
	IP Address	140.213.105.137
	IP Address	140.213.105.118
	IP Address	140.213.104.222
	IP Address	140.213.104.217
	IP Address	140.213.104.203
	IP Address	140.213.104.162
	IP Address	140.213.103.222
	IP Address	140.213.103.210
	IP Address	140.213.103.17

	Type	Indicator
	IP Address	140.213.102.5
	IP Address	140.213.102.224
	IP Address	140.213.102.193
	IP Address	140.213.102.172
	IP Address	140.213.101.86
	IP Address	140.213.101.227
	IP Address	140.213.101.20
	IP Address	140.213.101.179
	IP Address	140.213.101.17
	IP Address	140.213.101.128
	IP Address	140.213.100.76
	IP Address	140.213.100.44
	IP Address	140.213.100.152
	IP Address	140.213.100.136
	IP Address	140.213.100.131
	IP Address	140.213.100.127
	IP Address	112.215.210.73
	IP Address	112.215.210.187
	IP Address	112.215.210.145
	IP Address	112.215.209.144
	IP Address	112.215.209.131
	IP Address	112.215.208.204
	IP Address	112.215.208.135
	IP Address	104.28.218.135
	Domain	congtyxaydungvuhiep.com
	Domain	login.3dlntl-paypal.com

	Type	Indicator
	Domain	paypal-lntl.com
	Domain	3dlntl-paypal.com
	Domain	3dlntlverify.com
	Domain	3dlntlpaypalcard.com
	Domain	lntl-paypal.com
	Domain	login.paypal-lntl.com
