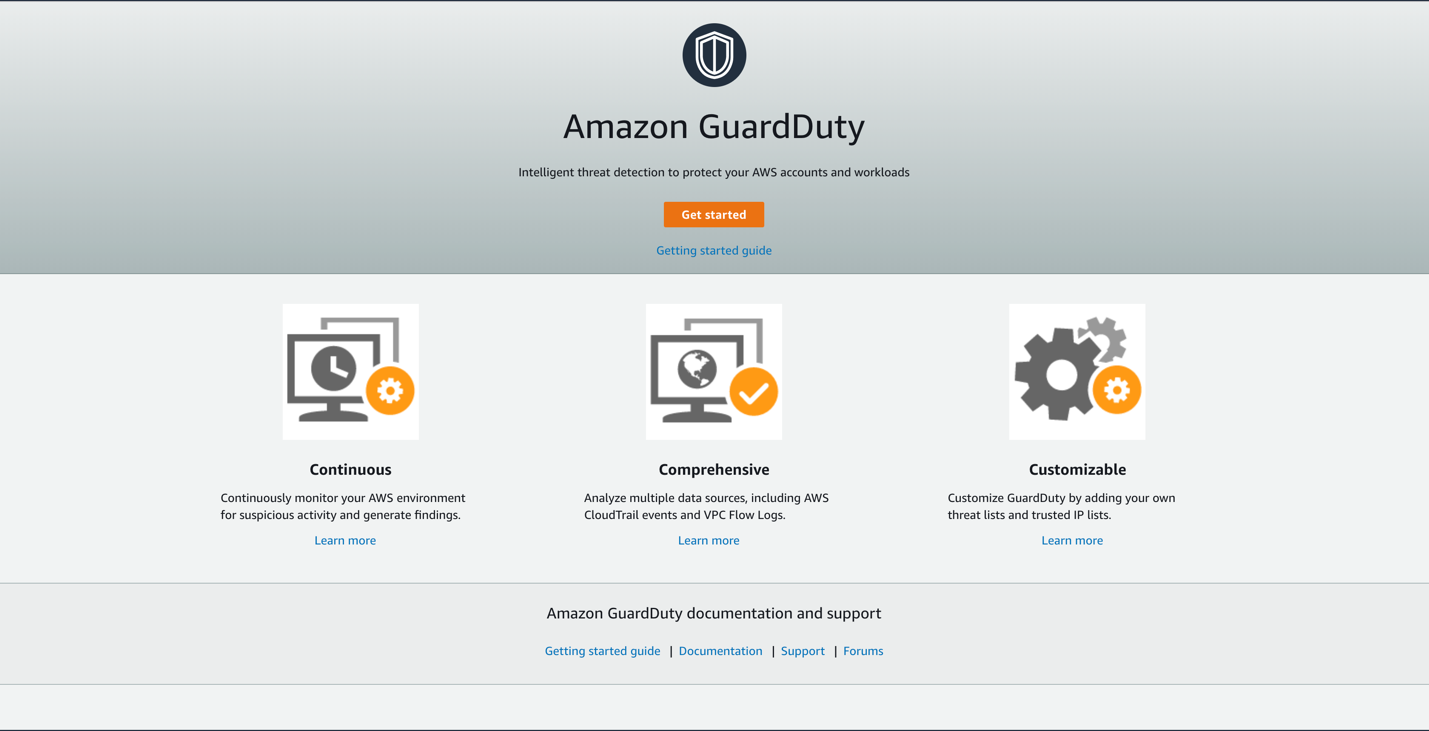
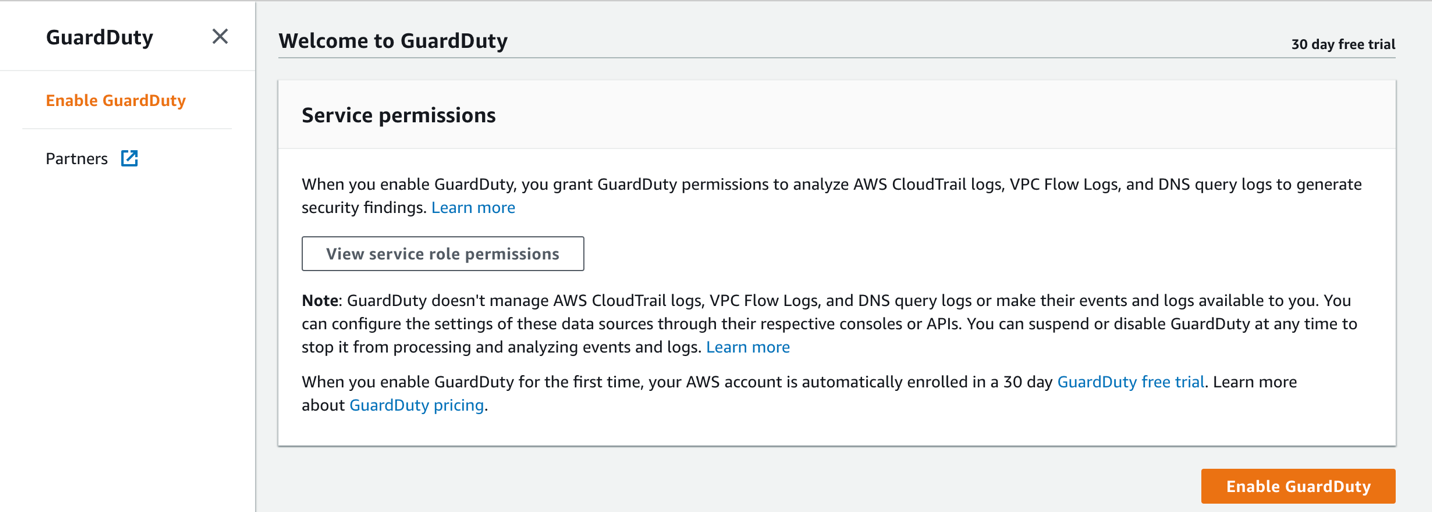
Getting Hands on with Amazon GuardDuty

## **Overview**

This lab walks you through a scenario covering threat detection and remediation using [Amazon GuardDuty](https://aws.amazon.com/guardduty/); a managed threat detection service. The scenario simulates an attack that spans a few threat vectors, representing just a small sample of the threats that GuardDuty is able to detect.

In addition, you will look at how to view and analyze GuardDuty findings, how to send alerts based on the findings, and, finally, how to remediate findings.

## **Deploy the environment**

1. Let's enable GuardDuty, in just 2 clicks! Please use the **us-east-1 (Virginia)** region for this workshop. Navigate to the [GuardDuty Console](https://us-east-1.console.aws.amazon.com/guardduty/home?) and click **Get Started**. 
2. On the next screen click **Enable GuardDuty**. 
3. Click Deploy to AWS to launch the CloudFormation stack to setup the lab environment. Please note we are using the US East 1 (Virgina): [Deploy in us-west-2](https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/new?stackName=GuardDuty-Hands-On&templateURL=https://sa-security-specialist-workshops-us-west-2.s3-us-west-2.amazonaws.com/guardduty-hands-on/amazon-guard-duty-revamped-v1.yml)
4. Click the **Deploy to AWS** button above. This will automatically take you to the console to run the template.
5. Click **Next** on the **Specify Template** page.
6. Under **Parameters** Enter your **EmailAddress** to receive the notifications during the lab, on the **Specify stack details** page. Then click **Next**
7. Scroll down and click **Next** on the **Configure stack options** page.
8. Finally, acknowledge that the template will create IAM roles under **Capabilities** and click **Create stack**.

This will bring you back to the CloudFormation console. You can refresh the page to see the stack starting to create. Before moving on, make sure the stack is in a **CREATE\_COMPLETE**.

1. You should see an email with subject **AWS Notification - Subscription Confirmation**. Please click on **Confirm Subscription** which will ensure you receive notifications during the lab.

While we wait for our CloudFormation stack to complete, take 5 minutes to read about GuardDuty **data sources** and **findings**. Then check back to confirm the CloudFormation build has completed, move onto to **What is created?** section below.

## **How GuardDuty Works**

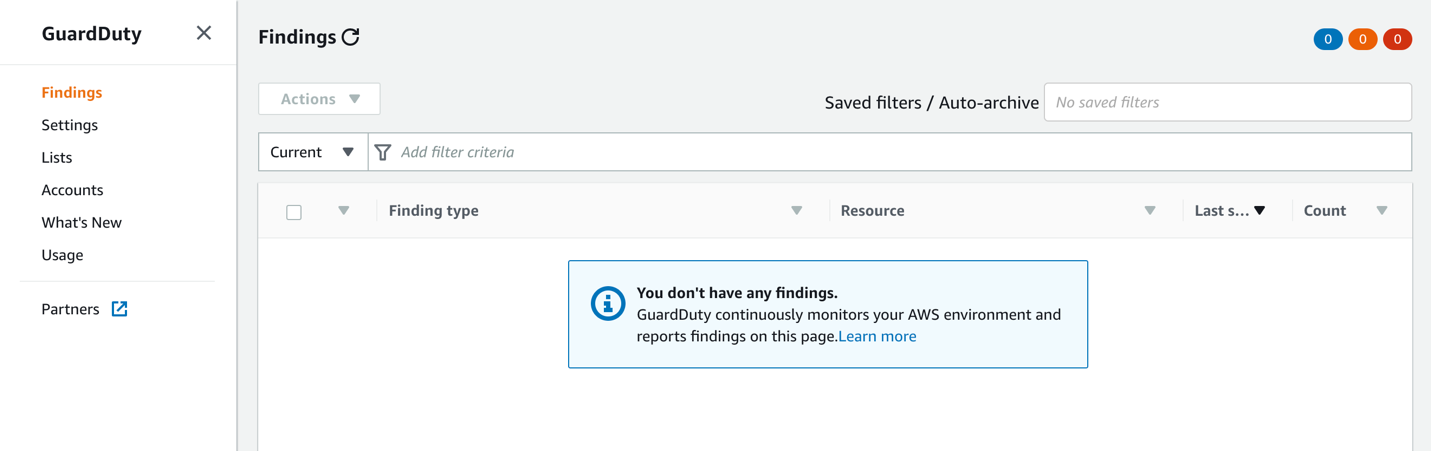
### **Data Sources**

From the moment you enable GuardDuty it begins analyzing all of the VPC Flow Logs, CloudTrail logs, and DNS Logs in that region. DNS Logs are generated from the default AWS DNS resolvers used for your VPCs and are not an available data source to customers.

If you are using a 3rd party DNS resolver or if you set up your own DNS resolvers, then GuardDuty cannot access, process, and identify threats from that data source.

GuardDuty accesses all of these [data sources](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_data-sources.html) without any of them having to be enabled; although it is a best practice to enable CloudTrail and VPC Flow Logs for your own analysis. GuardDuty is a regional service so in order for the service to monitor these data sources in other regions you will need to enable it in those regions. You can accomplish this by following the same steps above and enabling it through the console but most customers are using the APIs to programmatically enable it in all regions and across multiple accounts. Regardless of the number of VPCs, IAM users, or other AWS resources in your account, there is no impact to your resources because all of the processing is being done within the managed service.

The pricing for GuardDuty is based on the quantity of AWS CloudTrail Events analyzed and the volume of Amazon VPC Flow Log and DNS Log data analyzed (per GB). Each region in an AWS Account has a free 30-day trial to better forecast what the cost of the service will be.



### **Findings**

Now that GuardDuty is enabled it is actively monitoring the three data sources for malicious or unauthorized behavior as it relates to your EC2 instances and AWS IAM Principals. You should be taken directly to the **Findings** tab which will show finding details as GuardDuty detects them. After deploying the scenario, you will start to see GuardDuty findings being detected. Each finding is broken down into the following format to allow for a concise yet readable description of potential security issues.

**ThreatPurpose : ResourceTypeAffected / ThreatFamilyName . ThreatFamilyVariant ! Artifact**

Click [here](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-format.html) for a description of each part.

The more advanced behavioral and machine learning detections require a baseline (7 - 14 days) to be established so GuardDuty is able to learn the regular behavior and identity anomalies. An example of a finding that requires a baseline would be if an EC2 instance started communicating with a remote host on an unusual port or an IAM User who has no prior history of modifying Route Tables starts making modifications. All of the findings generated in these scenarios will be based on signatures, so the findings will be detected 10 minutes after the completion of the CloudFormation stack. The delay is due to the amount of time it takes for the information about a threat to appear in one of the data sources and the amount of time it takes for GuardDuty to access and analyze that particular data source.

Click [here](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-active.html) for a complete list of current GuardDuty finding types.

GuardDuty sends notifications based on Amazon CloudWatch Events when any change in the findings takes place. These notifications are sent within 5 minutes of the finding. All subsequent occurrences of an existing finding will have the same ID as the original finding and notifications will be sent every 6 hours after the initial notification. This is to eliminate alert fatigue due to the same finding.

The initial findings will begin to show up in GuardDuty 10 minutes after the CloudFormation stack creation completes.

### **What is created?**

The CloudFormation template will create the following resources:

* Three [Amazon EC2](https://aws.amazon.com/ec2/) Instances (and supporting network infrastructure)
  + Two Instances that contain the name “Compromised Instance”
  + One instance that contains the name “Malicious Instance”
* [AWS IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html) For EC2 which will have permissions to SSM Parameter Store and DynamoDB
* One [Amazon SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/GettingStarted.html) so you will be able to receive notifications
* Three [AWS CloudWatch Event](https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/WhatIsCloudWatchEvents.html) rules for triggering the appropriate notification or remediation
* Two [AWS Lambda](https://aws.amazon.com/lambda/) functions that will be used for remediating findings and will have permissions to modify Security Groups and revoke active IAM Role sessions (on only the IAM Role associated with this scenario)
* [AWS Systems Manager Parameter Store](https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-paramstore.html) value for storing a fake database password.

Make sure the CloudFormation stack is in a **CREATE\_COMPLETE** status before moving on.

# Compromised EC2 Instance

## **Scene simulation**

After an uneventful yet unnecessarily long commute to work, you arrived at the office on Monday morning. You grabbed a cup of coffee, sat down in your cube, opened up your laptop and begin to go through your emails. Soon after you begin though you start receiving emails indicating that GuardDuty has detected new threats. You don’t yet know the extent of the threats but you quickly begin to investigate. Now the good news is that your coworker Alice has already set up some hooks for specific findings so that they will be automatically remediated.

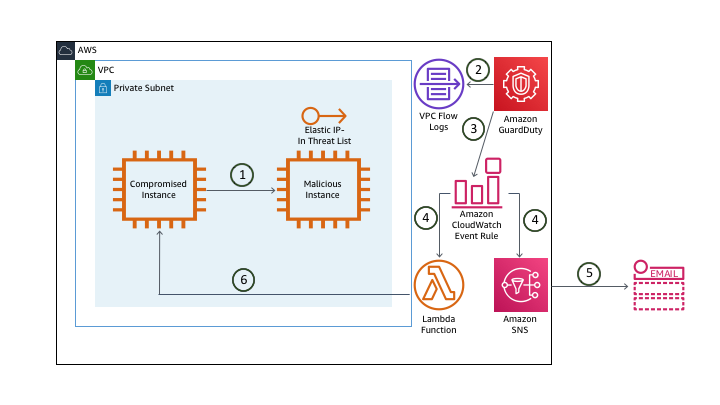
The first email you receive from GuardDuty indicates that one of your EC2 instances might be compromised:

GuardDuty Finding | ID: 1xx: The EC2 instance i-xxxxxxxxx may be compromised and should be investigated

Shortly after the first email, you receive a second email indicating that the same GuardDuty finding has been remediated:

GuardDuty Remediation | ID: 1xx: GuardDuty discovered an EC2 instance (Instance ID: i-xxx) that is communicating outbound with an IP Address on a threat list that you uploaded. All security groups have been removed and it has been isolated. Please follow up with any additional remediation actions.

## **Architecture Overview**



1. The **compromised instance** pings the EIP of the **malicious instance**. That EIP is in a custom threat list.
2. GuardDuty is monitoring the VPC Flow Logs (in addition to CloudTrail and DNS Logs) and analyzing this based on threat lists, machine learning, baselines, etc.
3. GuardDuty generates a finding and sends this to the GuardDuty console and CloudWatch Events.
4. The CloudWatch Event rule triggers an SNS topic and a Lambda function.
5. SNS sends you an e-mail with the finding information.
6. A Lambda function isolates the **compromised instance**.

When Alice setup the hook for notifications she only included certain information about the finding because she had also setup a Lambda function to automatically isolate the instance and send out the details of the remediation. Since the finding has been remediated you decide you still want to take a closer look at the setup Alice currently has in place.

## **Investigation**

### **Browse to the GuardDuty console to investigate**

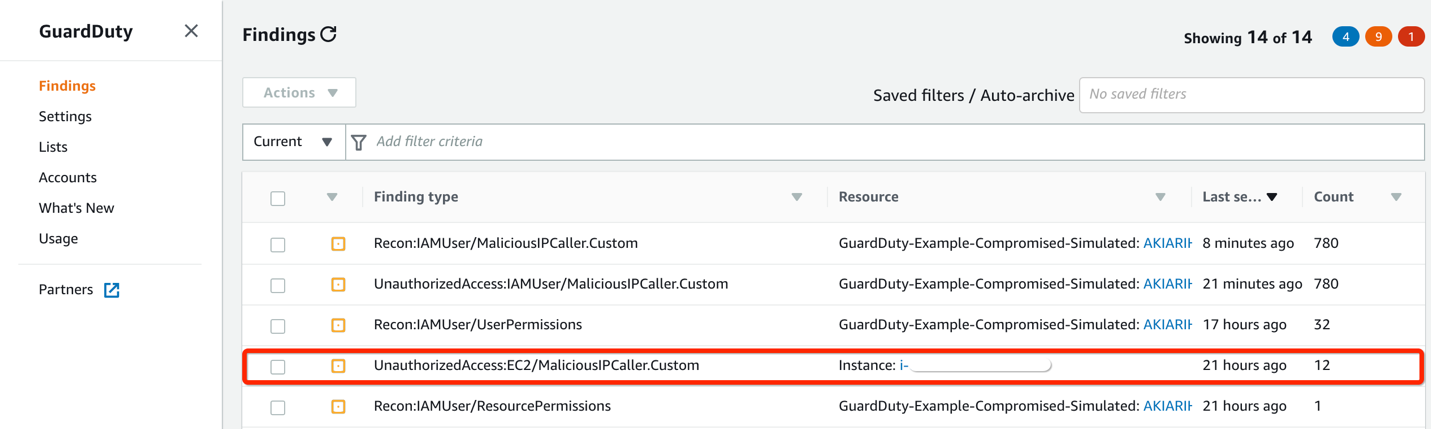
Although you can view the GuardDuty findings in the console, most customers aggregate all findings across their regions and accounts to a central security information and event management (SIEM) system for analysis and remediation. A common approach for aggregating these findings is to setup GuardDuty in a [Master/Member](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_accounts.html) structure and then use a workflow including CloudWatch Event Rules and Lambda Functions to push findings to your SIEM or a centralized logging framework. There are also partner solutions that publish Lambda Function Blueprints to make it easier to consolidate findings.

1. Navigate to the [GuardDuty Console](https://us-east-1.console.aws.amazon.com/guardduty/home?) (us-east-1).

If there is nothing displayed click the refresh button.

1. A finding should show up with the type **UnauthorizedAccess:EC2/MaliciousIPCaller.Custom**.

Based on the format you reviewed earlier can you determine the security issue by the finding type?



The quick view of the finding shows a severity symbol, the finding type, the affected resource, the last time the finding was detected, and a count of the subsequent occurrences of an existing finding.

Findings are available in the service for 90 days.

The finding type indicates that an EC2 instance in your environment is communicating outbound to an IP address included on a [custom threat list](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_upload_lists.html). Click on **Lists** in the left navigation to view the custom threat list Alice added.

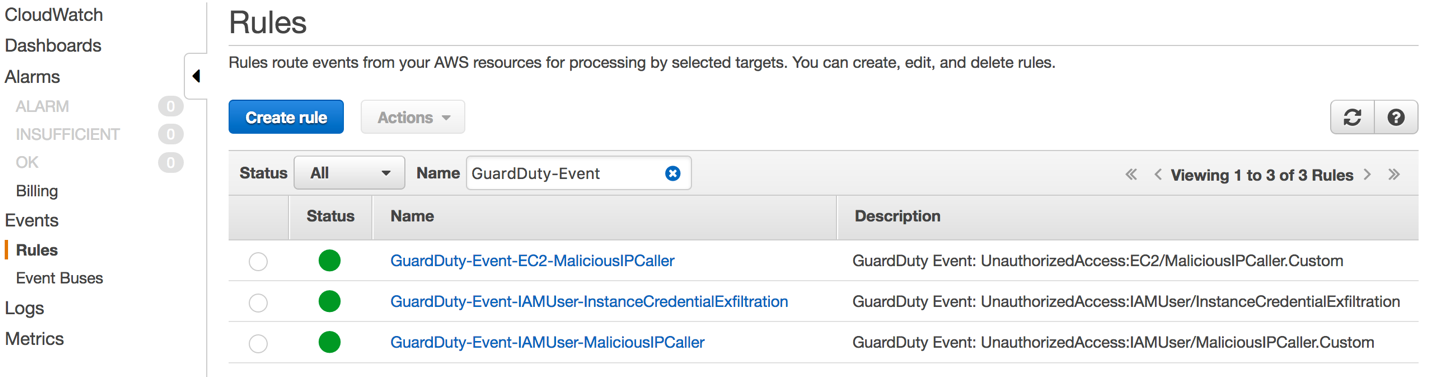
GuardDuty uses managed threat intelligence provided by AWS Security and third-party providers, such as ProofPoint and CrowdStike. You can expand the monitoring scope of GuardDuty by configuring it to use your own custom trusted IP lists and threat lists. If you setup a Master/Member GuardDuty structure, users from the Master GuardDuty account can manage trusted IP lists and threats lists and they are inherited by the member accounts. Users from the member accounts are not able to modify the lists.

### **View the CloudWatch Event rule**

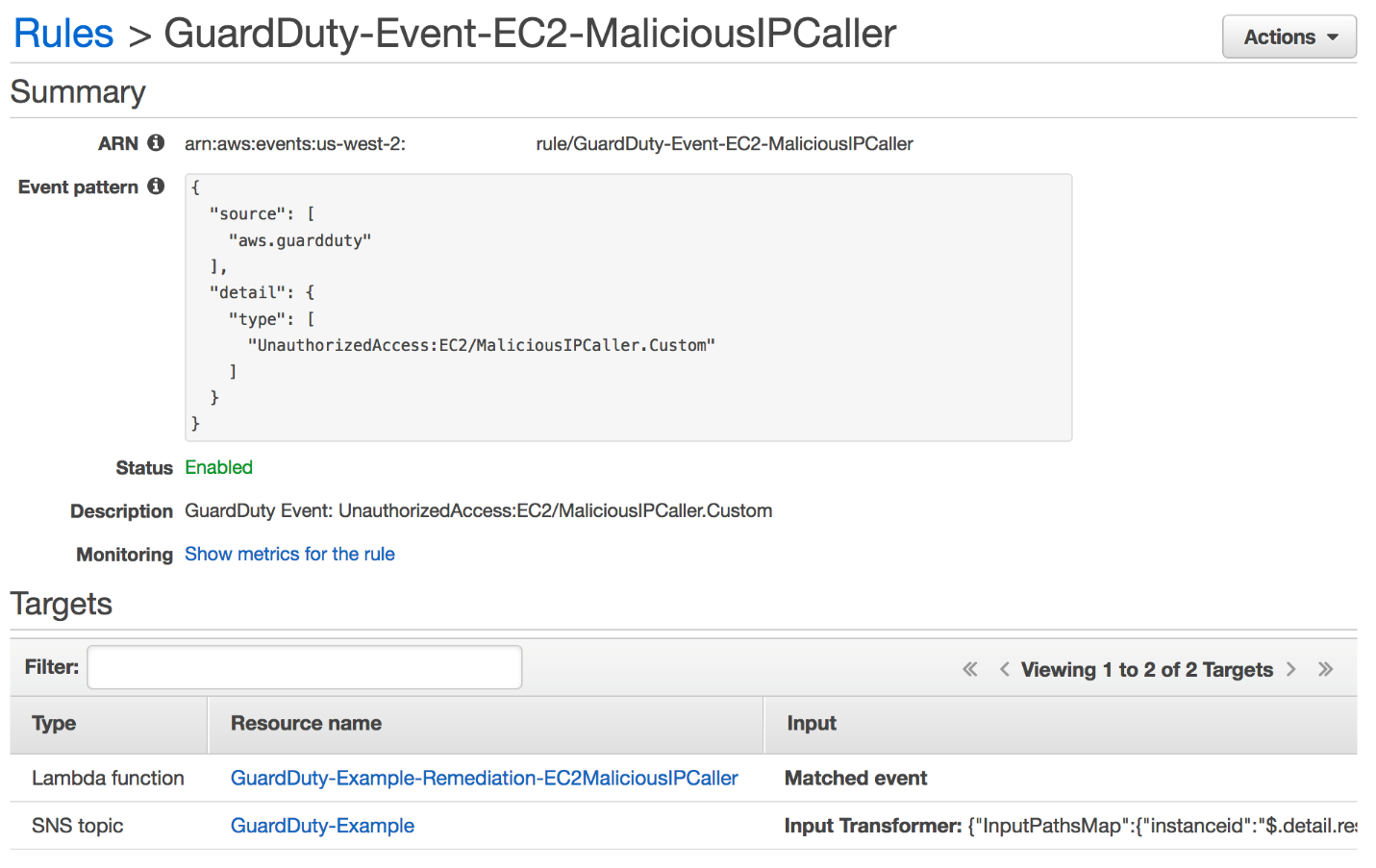
Alice used CloudWatch Event Rules to send the email you received about the findings and also to take remediations steps. Examine the CloudWatch Events console to understand what Alice configured and to see how the remediation was triggered.

1. Navigate to the [CloudWatch Console](https://us-east-1.console.aws.amazon.com/cloudwatch/home?) (us-east-1) and on the left navigation, under the **Events** section, click **Rules**.

You will see three Rules in the list that were created by the CloudFormation template. All of these begin with the prefix “GuardDuty-Event."



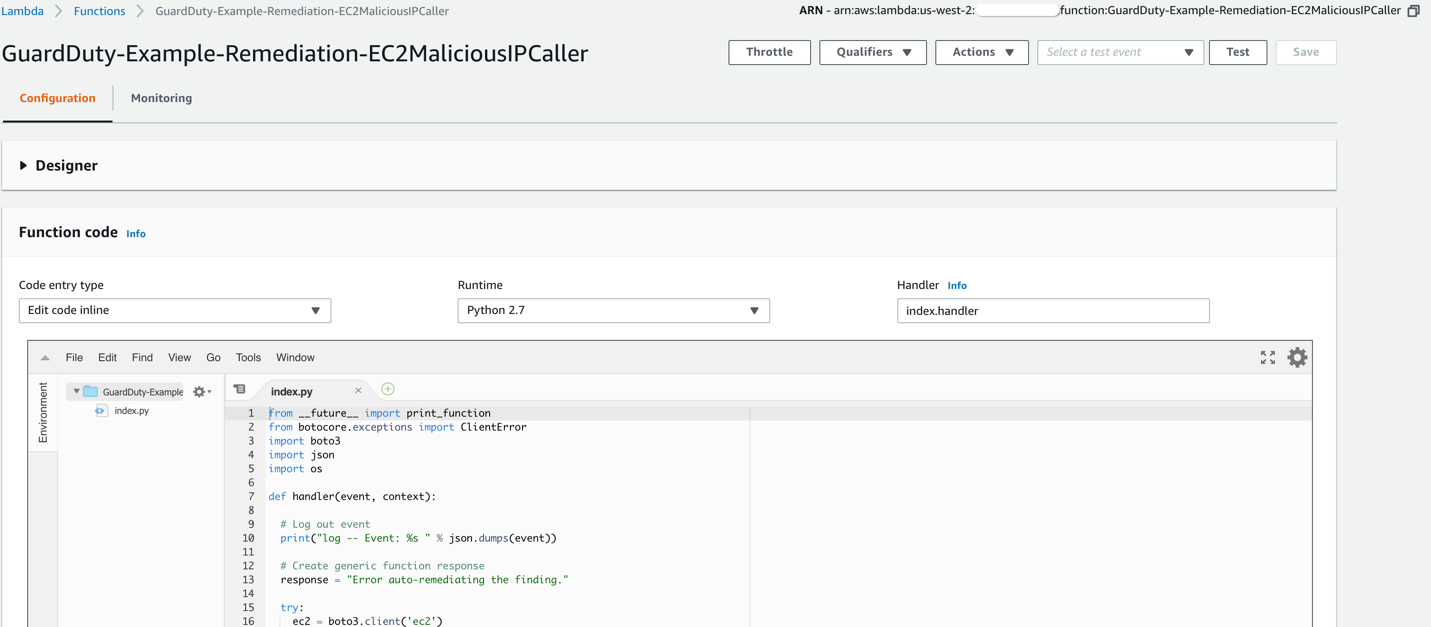
1. Click on the rule named **GuardDuty-Event-EC2-MaliciousIPCaller**.



Under the **Targets** section you will see two entries, one for a Lambda function and one for an SNS Topic. The CloudWatch Event Rule publishes the finding to the SNS Topic which in turn sends out an email notification. Rather than sending the entire JSON event you can see how Alice customized the email by using an [**input transformer**](https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/CloudWatch-Events-Input-Transformer-Tutorial.html). You can use the input transformer feature of CloudWatch Events to customize the text that is taken from an event before it is input to the target of a rule.

### **View the Remediation Lambda function**

The Lambda function is what handles the remediation logic for this finding. Alice setup the Lambda function to remove the compromised instance from its current security group and add it to one with no ingress or egress rules so that the instance is isolated from the network. Click the **Resource Name** for the Lambda function in the Targets section to evaluate the remediation logic.



Collapse the **Designer** tab and scroll down to view the code for this function (walking through the code logic is outside the scope of this scenario). You can also click the **Monitoring** tab and view the invocation details for this function.

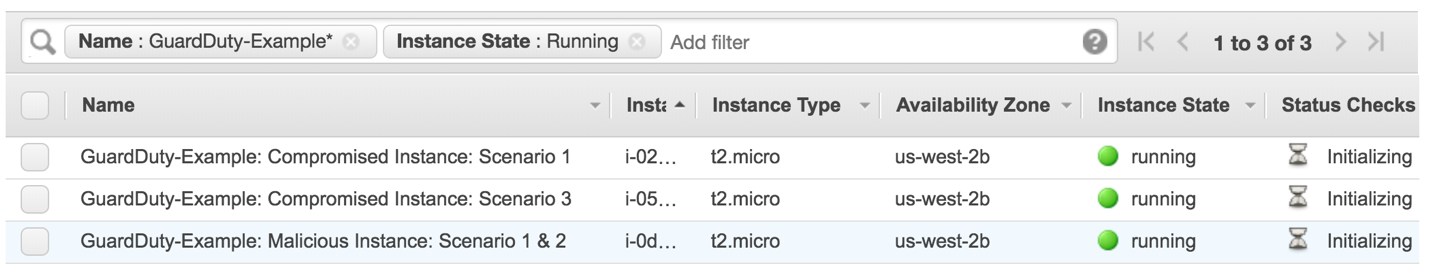
What permissions does the Lambda Function need to perform the remediation?

### **Verify that the remediation was successful**

Next, double check the effects of the remediation to ensure the instance is isolated. At this point you have the instance ID of the compromised instance from the email notifications and the name of the isolation security group name from reviewing the Lambda Function code.

1. Browse to the [EC2 console](https://us-east-1.console.aws.amazon.com/ec2/v2)(us-east-1) and click **Running Instances**.

You should see three instances with names that begin with **GuardDuty-Example**.



1. Click on the instance with the instance ID you saw in the GuardDuty finding or email notifications.

**GuardDuty-Example: Compromised Instance: Scenario 1**.

1. After reviewing the remediation Lambda Function you know that the instance should now have the Security Group with a name that includes **ForensicSecurityGroup**. Under the **Description** tab verify the instance has this security group.

Initially, all three of the instances launched by the CloudFormation template were in the Security Group with a name that includes **TargetSecurityGroup**. The Lambda function removed the TargetSecurityGroup from the instance and added the ForensicsSecurityGroup to isolate the instance.

1. Click on the **ForensicSecurityGroup** and view the ingress and egress rules.

## **Questions**

**Which data source did GuardDuty use to identify this threat?**

**Will isolating the instance have any effect on an application running on the instance?**

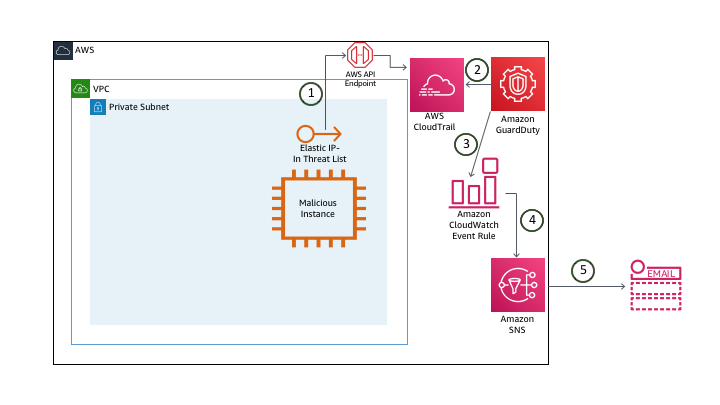
**How could you add more detail to the email notifications?**

# Compromised IAM credentials (simulated)

You have completed the examination of the first attack, confirmed it was properly remediated, and then sat back to take your first sip of coffee for the day when you notice an additional email about new findings. The first of the new findings indicates that an API call was made using AWS IAM credentials from an IP address on a custom threat list.

**None of your personal IAM credentials have actually been compromised or exposed in any way.**

## **Architecture Overview**



1. The **malicious instance** makes API calls. The EIP on the instance is in a custom threat list. API calls are logged in CloudTrail
2. GuardDuty is monitoring the CloudTrail Logs (in addition to VPC Flow Logs and DNS Logs) and analyzing this based on threat list, machine learning, baselines, etc.
3. GuardDuty generates a finding and sends this to the GuardDuty console and CloudWatch Events.
4. The CloudWatch Event rule triggers an SNS topic.
5. SNS sends you an e-mail with the finding information.

## **Investigation**

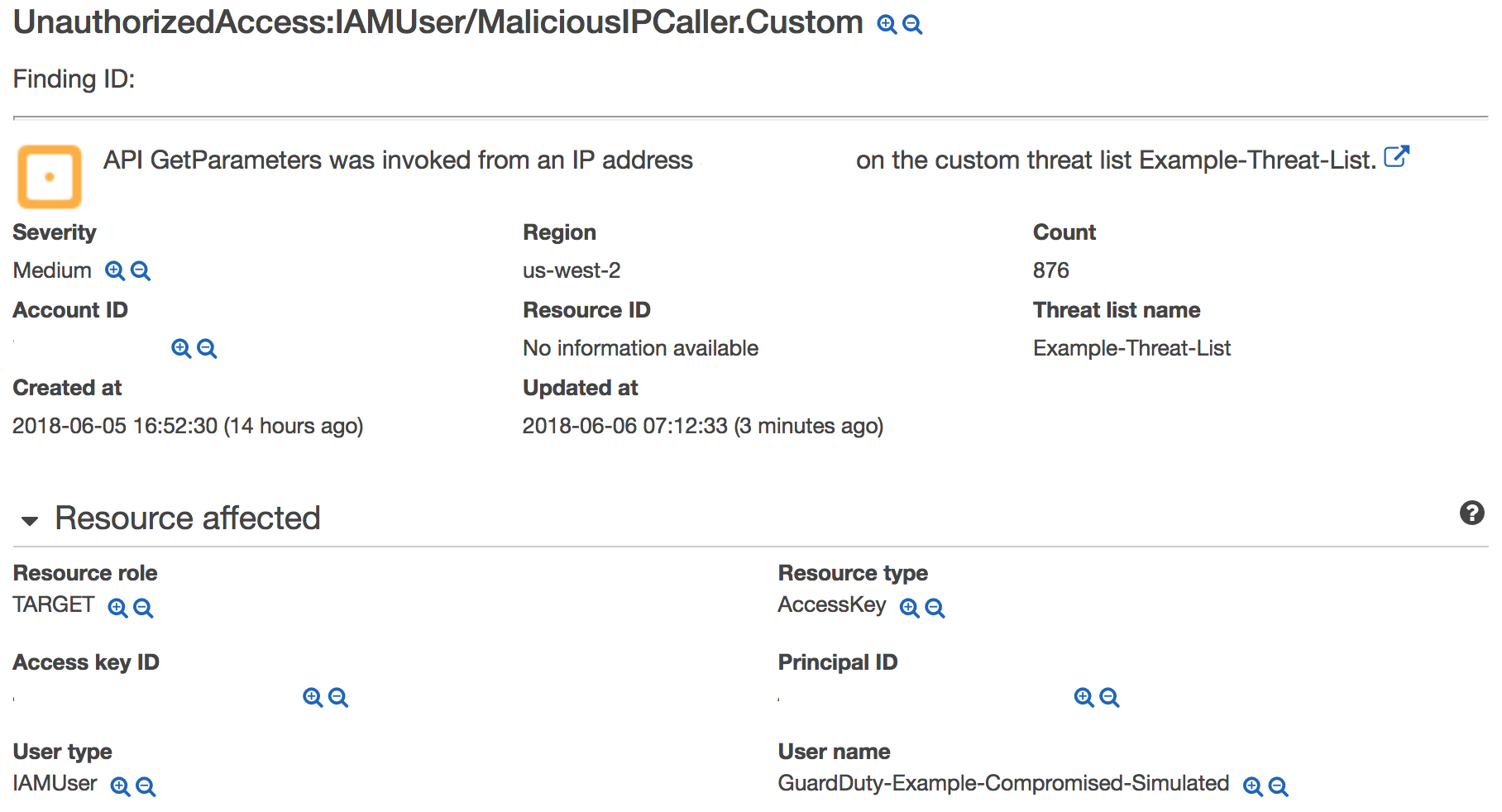
### **Browse to the GuardDuty console to investigate**

To view the findings:

1. Navigate to the [GuardDuty Console](https://us-east-1.console.aws.amazon.com/guardduty/home?) (us-east-1).
2. Click the Refresh icon to refresh the GuardDuty console. You should now see additional findings that are related to **Recon:IAMUser** and **UnauthorizedAccess:IAMUser**.

Based on the format you reviewed earlier can you determine the security issues by the finding type?

1. Click on the **UnauthorizedAccess:IAMUser/MaliciousIPCaller.Custom** finding to view the full details.



You can see the finding details include information about what happened, what AWS resources were involved in the suspicious activity, when this activity took place, and other additional information. Under **Resource Affected**, find the **User Name** associated with this finding.

This finding indicates that the IAM credentials (of the user you found above) are possibly compromised because API calls using those credentials are being made from an IP address on a custom threat list.

What actions did this AWS IAM User take? You can see under **Action** and then **API** that a **GetParameters** API call was made but how can you view the rest of the actions made by this user over the past hour or day? GuardDuty is able to analyze large volumes of data and identity true threats in your environment but from an investigation and remediation stand point it is still important to correlate other data to understand the full scope of the threat. In this case an analyst would use the details in this finding to pinpoint historical user activity in CloudTrail.

### **View the CloudWatch Event rule**

1. Navigate to the [CloudWatch console](https://us-east-1.console.aws.amazon.com/cloudwatch/home?) and on the left navigation, under the **Events** section, click **Rules**.
2. Click on the rule that Alice configured for this particular finding (**GuardDuty-Event-IAMUser-MaliciousIPCaller**).
3. Under the **Targets** section, you will see a rule for an SNS Topic. Turns out Alice did not set up a Lambda function to remediate this threat because the decision by the security team was to manually investigate and remediate this particular type of finding.

Since GuardDuty integrates with CloudWatch Events you have the flexibility to put in place full or partial automated remediation workflows. These could be custom Lambda Functions that you build out or maybe even [partner](https://aws.amazon.com/guardduty/resources/partners/) solutions. You can also configure other AWS Resources as targets in your CloudWatch Event Rules such as SSM Run Commands or Step Functions state machines. For some finding types you may choose to have only notification workflows and require manual remediation steps. As you design these workflows it is important to evaluate the workloads running in your environments to see what effects a remediation could have.

### **Manually remediate the finding**

Since Alice did not setup a remediation for this finding, you have to manually remediate this. While the security team is analyzing the previous activity of this user to better understand the scope of the compromise, you need to disable the access key associated with the user to prevent any more unauthorized actions.

1. Browse to the [AWS IAM](https://console.aws.amazon.com/iam/home?region=us-east-1) console.
2. Click **Users** in the left navigation.
3. Click on the user you identified in the GuardDuty finding and email notifications (**GuardDuty-Example-Compromised-Simulated**).
4. Click the **Security Credentials** tab.
5. Under **Access Keys**, find the Access Key ID you identified in the finding and click **Make Inactive**.

## **Questions**

**Which data source did GuardDuty use to identity this threat?**

**What permissions did the user have?**

**Why would the security team decide against setting up an automated remediation?**

# IAM Role credential exfiltration

### **Generate a finding manually**

All of the simulated attacks and findings are generated automatically in the CloudFormation template except for one; which requires you to take some manual steps. To produce the final finding, you will need to copy the IAM temporary security credentials from the EC2 instance and manually make API calls from your laptop.

**API calls need to come from outside the AWS network or they will not generate findings**

#### Retrieve the IAM temporary security credentials using AWS Systems Manager

To simulate this last and final attack you will need to retrieve the IAM temporary security credentials generated by the IAM Role for EC2. You can either SSH directly to the instance and query the metadata or follow the steps below to use [AWS Systems Manager Session Manager](https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager.html) (an SSM agent was automatically started on the instance at launch).

1. Go to [Managed Instances](https://us-east-1.console.aws.amazon.com/systems-manager/managed-instances?region=us-east-1) within the **AWS Systems Manager** console (us-east-1).

You should see an instance named **GuardDuty-Example: Compromised Instance: Scenario 3** with a ping status of **Online**.

1. To see the keys currently active on the instance, click on **Session Manager** on the left navigation and then click **Start Session**.
2. To see the credentials currently active on the instance, click on the radio button next to **Compromised Instance: Scenario 3** and click **Start Session**.
3. Run the following command in the shell:

curl http://169.254.169.254/latest/meta-data/iam/security-credentials/GuardDuty-Example-EC2-Compromised

1. Make note of the **AccessKeyID**, **SecretAccessKey**, and **Token**.

#### Create a new AWS CLI profile on your laptop to use the IAM temporary credentials

Now that you have retrieved the IAM temporary security credentials you will need to add them to an AWS CLI profile. There are a number of ways to do this, but below are some commands to help get you started:

If you do not have the AWS command line installed, you can find instructions for your OS here:

https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html

From a command prompt, run the following commands (replace the variables with your credentials):

aws configure set profile.badbob.region us-east-1

aws configure set profile.badbob.aws\_access\_key\_id <access\_key>

aws configure set profile.badbob.aws\_secret\_access\_key <secret\_key>

aws configure set profile.badbob.aws\_session\_token <session\_token>

If you view your local aws credentials file, you should now see an [badbob] profile with the stolen IAM temporary credentials.

#### Run commands using the IAM temporary credentials

Now that you have your named profile you can use it to make API calls. Use the commands below to query different services to see what you have access to.

**Don't be surprised if you see some access denied responses, it is intended**

**Do you have any IAM permissions:**

aws iam get-user --profile badbob

aws iam create-user --user-name Chuck --profile badbob

**What about DynamoDB?**

aws dynamodb list-tables --profile badbob

aws dynamodb describe-table --table-name GuardDuty-Example-Customer-DB --profile badbob

**Can you query the data?**

aws dynamodb scan --table-name GuardDuty-Example-Customer-DB --profile badbob

aws dynamodb put-item --table-name GuardDuty-Example-Customer-DB --item '{"name":{"S":"Joshua Tree"},"state":{"S":"Michigan"},"website":{"S":"https://www.nps.gov/yell/index.htm"}}' --profile badbob

aws dynamodb scan --table-name GuardDuty-Example-Customer-DB --profile badbob

aws dynamodb delete-table --table-name GuardDuty-Example-Customer-DB --profile badbob

aws dynamodb list-tables --profile badbob

**Do you have access to Systems Manager Parameter Store?**

aws ssm describe-parameters --profile badbob

aws ssm get-parameters --names "gd\_prod\_dbpwd\_sample" --profile badbob

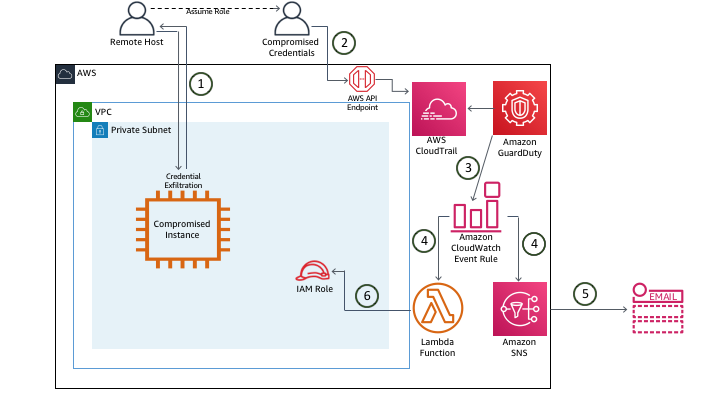
aws ssm get-parameters --names "gd\_prod\_dbpwd\_sample" --with-decryption --profile badbob

aws ssm delete-parameter --name "gd\_prod\_dbpwd\_sample" --profile badbob

After manually remediating the previous GuardDuty finding, you have finally finished your first cup of coffee when an email notification comes in alerting you to yet another finding. You finish reading the first email and then a minute or so later you see the relevant remediation email, meaning Alice has already put in place a remediation for this finding. The other findings you looked at dealt with EC2 instances and AWS IAM credentials separately, but this finding appears to be related to an AWS IAM Role associated with an EC2 instance. You decide to take a closer look at the finding and remediation.

**None of your personal IAM credentials have actually been compromised or exposed in any way.**

## **Architecture Overview**



1. The remote host accesses the **compromised instance** and exfiltrates the IAM role credentials via the metadata.
2. The remote host sets up a CLI user to make API calls to the AWS account to which the credentials belong.
3. GuardDuty generates a finding and sends this to the GuardDuty console and CloudWatch Events.
4. The CloudWatch Event rule triggers an SNS topic and a Lambda function.
5. SNS sends you an e-mail with the finding information.
6. A Lambda function attaches a policy to the role revoking all active sessions.

## **Investigation**

### **Browse to the GuardDuty console to investigate**

To view the findings:

1. Navigate to the [GuardDuty console](https://us-east-1.console.aws.amazon.com/guardduty/home?) (us-east-1) and then, in the navigation pane on the left, choose **Current**.
2. Click the Refresh icon to refresh the GuardDuty console. You should see a finding with the type **UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration**.
3. Click on the **UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration** finding to view the full details.

Looking at the finding details you can see that this is actually a **High Severity** finding. This finding informs you of attempts to run AWS API operations from a host outside of EC2, using temporary AWS credentials that were created on an EC2 instance in your AWS account. This means your EC2 instance has been compromised, and the temporary credentials from the instance have been exfiltrated to a remote host outside of AWS.

You will notice that each GuardDuty finding has an assigned severity level (low, medium, or high) that can help you determine your response to a potential security issue that is highlighted by the finding. These severity levels are preset by AWS but we have seen customers modify these values in their automation workflows to better align the risk of that finding in the context of their environment and requirements.

### **View the CloudWatch Event rule**

1. Navigate to the [CloudWatch console](https://us-east-1.console.aws.amazon.com/cloudwatch/home?) (us-east-1) and on the left navigation, under the **Events** section, click **Rules**.
2. Click on the rule that Alice configured for this particular finding (**GuardDuty-Event-IAMUser-InstanceCredentialExfiltration**).

Take a closer look at the **Event Pattern**. The pattern Alice setup for all the rules specifies particular findings.

Like Alice, you can create CloudWatch Event Rules that are triggered for particular findings but you can also create a rule that is triggered based on any GuardDuty finding in order to have a centralized workflow. Below is an example of an Event Pattern that would trigger for any GuardDuty finding:

{

"detail-type": [

"GuardDuty Finding"

],

"source": [

"aws.guardduty"

]

}

### **View the remediation Lambda function**

Alice also set up a remediation for this threat. Look through the Lambda Function code to better understand the remediation.

Go to the [Lambda console](https://us-east-1.console.aws.amazon.com/lambda/home?) (us-east-1) and review the function named **GuardDuty-Example-Remediation-InstanceCredentialExfiltration**.

The Lambda Function retrieves the Role name from the finding details and then attaches an IAM policy that revokes all active sessions for the role.

What permissions does the Lambda Function need to perform the remediation? Is there a risk associated with this level of permissions?

### **Verify that the remediation was successful**

To verify that the **InstanceCredentialExfiltration** finding was remediated, you can run one of the CLI commands you ran earlier.

aws dynamodb list-tables --profile badbob

You should see a response that states that there is an explicit deny for that action. Go view the Role to evaluate the policy that was attached.

1. Browse to the [AWS IAM](https://console.aws.amazon.com/iam/home?region=us-east-1) console.
2. Click **Roles** in the left navigation.
3. Click on the Role you identified in the GuardDuty finding and email notifications (**GuardDuty-Example-EC2-Compromised**).
4. Click the **Permissions** tab.
5. Click on the **RevokeOldSessions** Policy.

## **Questions**

**What are the risks involved with this remediation?**

**What other EC2 instances are using this Role?**

# Summary

By walking through these scenarios you generated, analyzed, and remediated all of the following threats in your environment:

* [UnauthorizedAccess:EC2/MaliciousIPCaller.Custom](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_unauthorized.html#unauthorized8)
* [Recon:IAMUser/MaliciousIPCaller.Custom](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_recon.html#recon2)
* [UnauthorizedAccess:IAMUser/MaliciousIPCaller.Custom](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_unauthorized.html#unauthorized2)
* [UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration](https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_unauthorized.html#unauthorized11)

Now that you understand the different components of the GuardDuty service and how it integrates with other AWS services, you can explore ways of using it across your own environment.