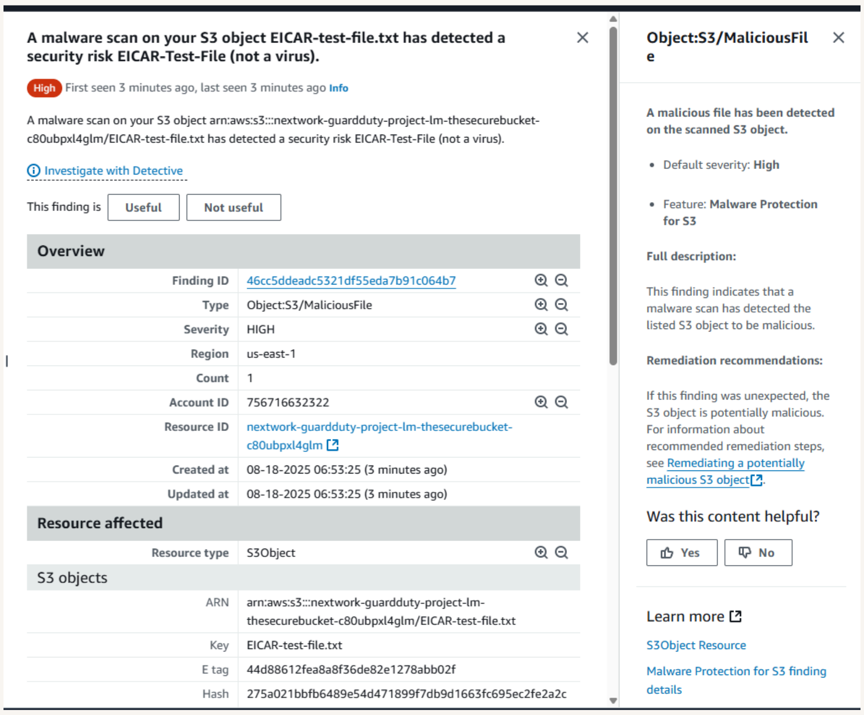
nextwork.org

# Threat Detection with GuardDuty

L Louis Moyo



**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 3 minutes ago, last seen 3 minutes ago Info

A malware scan on your S3 object arn:aws:s3:::nextwork-guardduty-project-lm-thesecurebucket-c80ubpxl4glm/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 | 46cc5ddeadc5321df55eda7b91c064b7 |
| Type | Object:S3/MaliciousFile |
| Severity | HIGH |
| Region | us-east-1 |
| Count | 1 |
| Account ID | 756716632322 |
| Resource ID | nextwork-guardduty-project-lm-thesecurebucket-c80ubpxl4glm |
| Created at | 08-18-2025 06:53:25 (3 minutes ago) |
| Updated at | 08-18-2025 06:53:25 (3 minutes ago) |

**Resource affected**

| | |
|---|---|
| Resource type | S3Object |

S3 objects

| | |
|---|---|
| ARN | arn:aws:s3:::nextwork-guardduty-project-lm-thesecurebucket-c80ubpxl4glm/EICAR-test-file.txt |
| Key | EICAR-test-file.txt |
| E tag | 44d88612fea8a8f36de82e1278abb02f |
| Hash | 275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c |

**Object:S3/MaliciousFile**

**A malicious file has been detected on the scanned S3 object.**

- Default severity: **High**
- Feature: **Malware Protection for S3**

**Full description:**

This finding indicates that a malware scan has detected the listed S3 object to be malicious.

**Remediation recommendations:**

If this finding was unexpected, the S3 object is potentially malicious. For information about recommended remediation steps, see Remediating a potentially malicious S3 object.

Was this content helpful?

Yes No

Learn more

S3Object Resource

Malware Protection for S3 finding details

# Introducing Today's Project!

## Tools and concepts

The services I used were Amazon S3, GuardDuty Malware Protection, AWS CloudFormation, and CloudShell. Key concepts I learnt include how GuardDuty scans S3 buckets for malware using test files like EICAR, how CloudFormation automates resource creation (and the importance of deleting stacks afterwards), and how to securely manage and remove sensitive files like credentials.json from CloudShell. I also learnt how to troubleshoot failed resource deletions, clean up S3 versioned objects, and why it's critical to clean up all project resources to avoid unnecessary costs.

## Project reflection

This project took me approximately a few hours to complete. The most challenging part was troubleshooting the DELETE_FAILED S3 bucket issue, since I had to manually empty the bucket before deletion. It was most rewarding to see GuardDuty detect the EICAR test file successfully, confirming that malware protection was functioning.

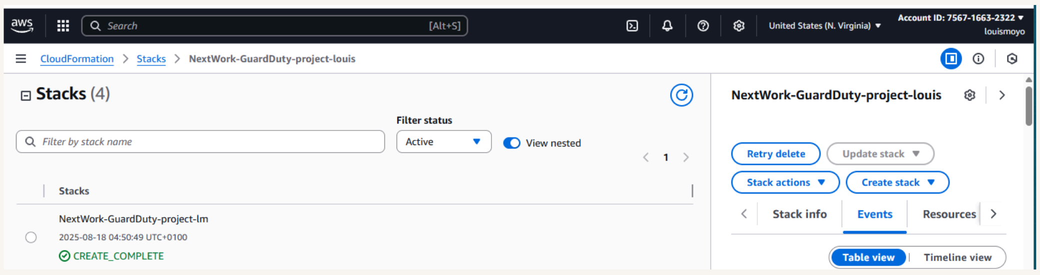
I did this project today to gain practical experience with GuardDuty Malware Protection and AWS clean-up best practices. Yes, this project met my goals because it gave me hands-on security testing, better understanding of CloudFormation, and confidence in managing AWS resources securely.

## Project Setup

To set up for this project, I deployed a CloudFormation template that launches a vulnerable web application environment for security testing. The three main components are: Web app infrastructure – an EC2 instance hosting the OWASP Juice Shop, along with supporting networking resources such as a new VPC, subnets, route tables, internet gateway, load balancer, and CloudFront distribution to make the app accessible. S3 storage – a bucket containing a sensitive file (important-information.txt) to simulate valuable data that attackers may try to steal. GuardDuty monitoring – AWS GuardDuty enabled to detect suspicious behaviour, attacks, or anomalies in the environment so I can observe how threats are identified in real time.

The web app deployed is called the OWASP Juice Shop. To practice my GuardDuty skills, I will simulate real-world attacks against this intentionally vulnerable app - such as stealing credentials and sensitive data - and then use GuardDuty to detect, analyse, and understand the alerts it generates in response to those threats.

GuardDuty is AWS's managed threat detection service that uses machine learning, anomaly detection, and threat intelligence to identify potential attacks or malicious activity in your AWS environment. In this project, it will continuously monitor the vulnerable OWASP Juice Shop app, analyze traffic and account activity, and generate findings when we simulate real-world attacks. This lets me see how GuardDuty detects threats in practice and how security teams can use those alerts to respond.



CloudFormation > Stacks > NextWork-GuardDuty-project-louis

Stacks (4)

Filter by stack name

Filter status: Active

View nested

Stacks

NextWork-GuardDuty-project-lm
2025-08-18 04:50:49 UTC+0100
CREATE_COMPLETE

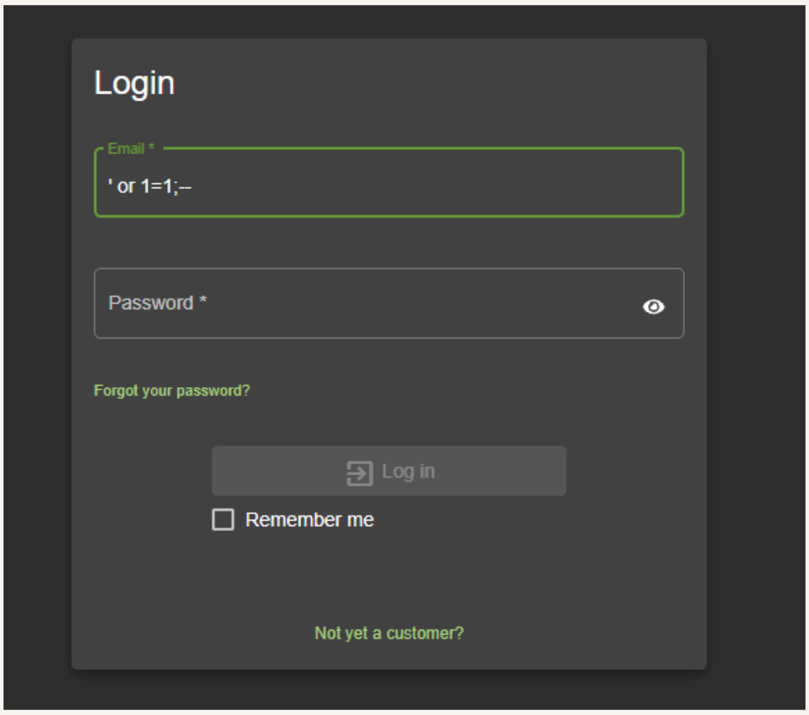
NextWork-GuardDuty-project-louis

Louis Moyo
NextWork Student

## SQL Injection

The first attack I performed on the web app is SQL injection, which means inserting malicious SQL code into an input field (like a username or password box) to trick the database into running unintended queries. SQL injection is a security risk because it can let attackers bypass authentication, view or steal sensitive data, or even take control of the entire database.

My SQL injection attack involved entering crafted text such as ' or 1=1;-- into the login form’s username field. This means the database query always evaluates as true, which bypasses normal login checks and lets me access the admin portal without knowing valid credentials.

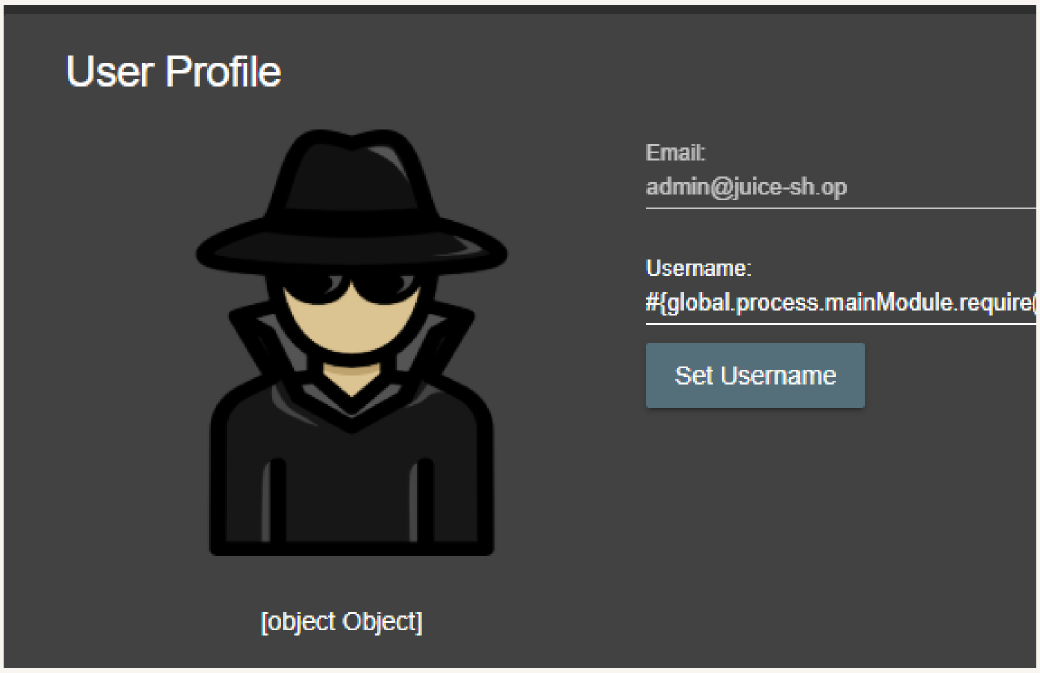

Login

Email *
' or 1=1;–

## Command Injection

Next, I used command injection, which is a type of attack where an attacker tricks a vulnerable application into running system-level commands instead of just handling input as plain text. The Juice Shop web app is vulnerable to this because it doesn't properly validate or sanitise user input in fields like the username. Instead of storing my text safely, it executes the malicious code I enter, allowing me to run commands on the underlying EC2 server and steal IAM credentials from the instance metadata service.

To run command injection, I pasted malicious JavaScript into the Username field of the admin profile page. The script will force the EC2 server hosting the web app to call the AWS instance metadata service, grab the IAM security credentials, and save them into a publicly accessible credentials.json file. This works because the app didn't sanitise the input and executed my code instead of treating it as harmless text.


User Profile

Email:
admin@juice-sh.op

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

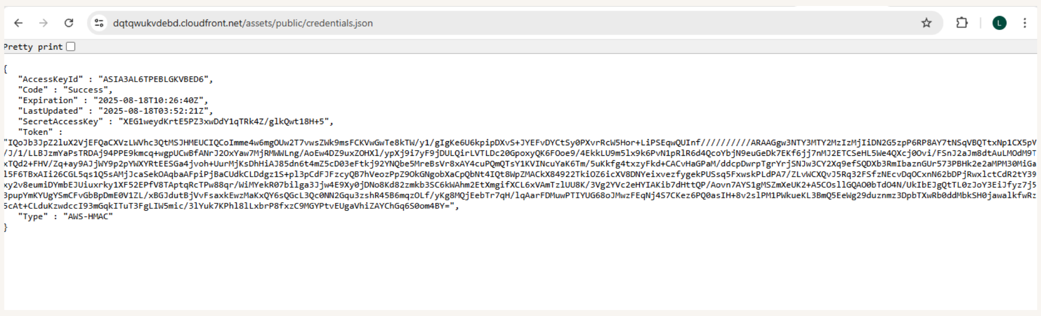
[object Object]

L

**Louis Moyo**
NextWork Student

## Attack Verification

To verify the attack's success, I navigated to the /assets/public/credentials.json URL in my deployed Juice Shop app. The credentials page showed me the stolen AWS access keys, secret key, and session token, along with their expiration time. Seeing these values confirmed that my command injection successfully extracted the EC2 instance’s IAM credentials and made them publicly accessible.



dqtqwukvdebd.cloudfront.net/assets/public/credentials.json

Pretty print ☐

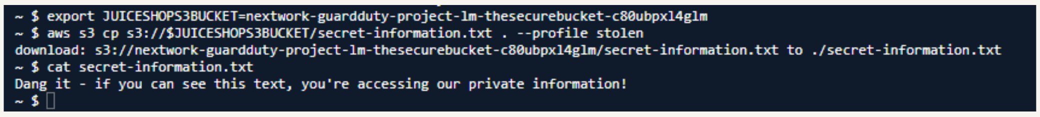
```
{
   "AccessKeyId" : "ASIA3AL6TPEBLGKVBED6",
   "Code" : "Success",
   "Expiration" : "2025-08-18T10:26:40Z",
   "LastUpdated" : "2025-08-18T03:52:21Z",
   "SecretAccessKey" : "XEG1weydKrtE5PZ3xwDdY1qTRk4Z/glkQwt18H+5",
   "Token" :
"IQoJb3JpZ2luX2VjEFQaCXVzLWVhc3QtMSJHMEUCIQCoImme4w6mgOUw2T7vwsZWk9msFCKVwGwTe8kTW/y1/gIgKe6U6kpipDXvS+JYEFvDYCtSy0PXvrRcW5Hor+LiPSEqwQUInf//////////ARAAGgw3NTY3MTY2MzIzMjIiDN2G5zpP6RP8AY7tNSqVBQTtxNp1CX5pV
/J/1/LLBJzmYaPsTRDAj94PPE9kmcq+wgpUCwBfANrJ2OxYaw7MjRMWWLng/AoEw4DZ9uxZOHXl/ypXj9i7yF9jDULQirLVTLDc20GpoxyQK6FOoe9/4EkkLU9m5lx9k6PvN1pRlR6d4QcoYbjN9euGeDk7EKf6jj7nMJ2ETCSeHL5We4QXcj0Ovi/FSnJ2aJm8dtAuLMOdM9T
xTQd2+FHV/Zq+ay9AJjWY9p2pYWXYRtEESGa4jvoh+UurMjKsDhHiAJ85dn6t4mZ5cD03eFtkj92YNQbe5MreBsVr8xAY4cuPQmQTsY1KVINcuYaK6Tm/5uKkfg4txzyFkd+CACvHaGPaM/ddcpDwrpTgrYrjSNJw3CY2Xq9efSQDXb3RmIbaznGUr573PBHk2e2aMPM30MiGa
l5F6TBxAIi26CGL5qs1Q5sAMjJcaSekOAqbaAFpiPjBaCUdkCLDdgz1S+pl3pCdFJFzcyQB7hVeozPpZ9OkGNgobXaCpQbNt4IQt8WpZMACkX84922TkiOZ6icXV8DNYeixvezfygekPUSsq5FxwskPLdPA7/ZLvWCXQvJ5Rq32FSfzNEcvDqOCxnN62bDPjRwxlctCdR2tY39
xy2v8eumiDYmbEJUiuxrky1XF52EPfV8TAptqRcTPw88qr/WiMYekR07bilga3Jjw4E9Xy0jDNo8Kd82zmkb3SC6kWAhm2EtXmgifXCL6xVAmTzlUU8K/3Vg2YVc2eHYIAKib7dHttQP/Aovn7AYS1gMSZmXeUK2+A5COsllGQAO0bTdO4N/UkIbEJgQtTL0zJoY3EiJfyz7j5
3pupYmKYUgYSmCFvGbBpDmE0V1ZL/xBGJdutBjVvFsaxkEwzMaKxQY6sQGcL3Qc0NN2Gqu3zshR45B6mqzOLf/yKg8MQjEebTr7qH/lqAarFDMuwPTIYUG68oJMwzFEqNj4S7CKez6PQ0asIH+8v2slPM1PWkueKL3BmQ5EeWg29duznmz3DpbTXwRb0ddMbkSH0jawalkfwRz
5cAt+CLduKzwdccI93mGqkITuT3FgLIW5mic/3lYuk7KPhl8lLxbrP8fxzC9MGYPtvEUgaVhiZAYChGq6S0om4BY=",
   "Type" : "AWS-HMAC"
}
```

## Using CloudShell for Advanced Attacks

The attack continues in CloudShell, because it lets me simulate an external hacker using the stolen IAM credentials outside of the EC2 instance they belong to. By running commands in CloudShell with those keys, GuardDuty sees unusual behaviour —credentials meant for one server being used elsewhere—which is exactly the kind of suspicious activity it's designed to detect.

In CloudShell, I used wget to download the credentials.json file from the Juice Shop app into my session, giving me a local copy of the stolen AWS credentials. Next, I ran a command using cat and jq to display the contents of the file in a clear, formatted way, so I could easily read the AccessKeyId, SecretAccessKey, and SessionToken that were extracted during the attack.

I then set up a profile, called stolen, to use the AWS access keys I stole from the vulnerable web app and simulate how an attacker would operate with them. I had to create a new profile because the default CloudShell profile uses my own account's permissions, and I needed a separate profile to clearly test and demonstrate what could be done with the compromised credentials.



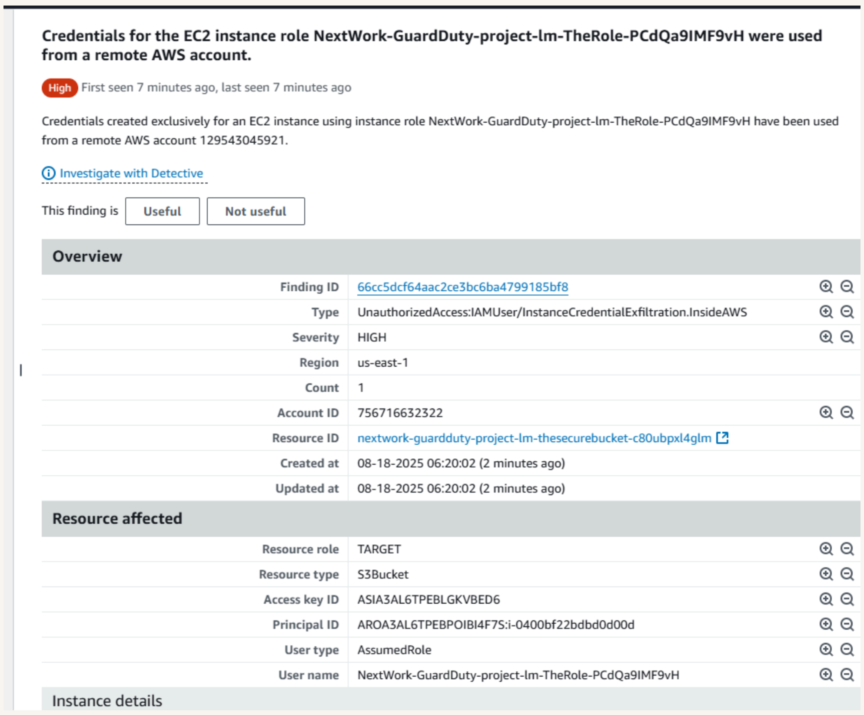
```
~ $ export JUICESHOPS3BUCKET=nextwork-guardduty-project-lm-thesecurebucket-c80ubpxl4glm
~ $ aws s3 cp s3://$JUICESHOPS3BUCKET/secret-information.txt . --profile stolen
download: s3://nextwork-guardduty-project-lm-thesecurebucket-c80ubpxl4glm/secret-information.txt to ./secret-information.txt
~ $ 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 a few minutes. Findings are not instant but typically appear quickly enough to provide near real-time visibility into suspicious or malicious activity in the AWS environment.

GuardDuty's finding was called UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS, which means the EC2 instance role's credentials were stolen and then used from another AWS account, indicating credential theft. Anomaly detection was used because GuardDuty identified that the credentials were being accessed in an unusual and suspicious way outside their expected context.

GuardDuty's detailed finding reported that credentials created exclusively for the EC2 instance role NextWork-GuardDuty-project-Im-TheRole-PCdQa9IMF9vH were used from a remote AWS account. This indicated possible credential exfiltration, flagged as UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS, with a severity of HIGH.



**Credentials for the EC2 instance role NextWork-GuardDuty-project-lm-TheRole-PCdQa9IMF9vH were used from a remote AWS account.**

High First seen 7 minutes ago, last seen 7 minutes ago

Credentials created exclusively for an EC2 instance using instance role NextWork-GuardDuty-project-lm-TheRole-PCdQa9IMF9vH have been used from a remote AWS account 129543045921.

Investigate with Detective

This finding is Useful Not useful

**Overview**

| | |
|---|---|
| Finding ID | 66cc5dcf64aac2ce3bc6ba4799185bf8 |
| Type | UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS |
| Severity | HIGH |
| Region | us-east-1 |
| Count | 1 |
| Account ID | 756716632322 |
| Resource ID | nextwork-guardduty-project-lm-thesecurebucket-c80ubpxl4glm |
| Created at | 08-18-2025 06:20:02 (2 minutes ago) |
| Updated at | 08-18-2025 06:20:02 (2 minutes ago) |

**Resource affected**

| | |
|---|---|
| Resource role | TARGET |
| Resource type | S3Bucket |
| Access key ID | ASIA3AL6TPEBLGKVBED6 |
| Principal ID | AROA3AL6TPEBPOIBI4F7S:i-0400bf22bdbd0d00d |
| User type | AssumedRole |
| User name | NextWork-GuardDuty-project-lm-TheRole-PCdQa9IMF9vH |

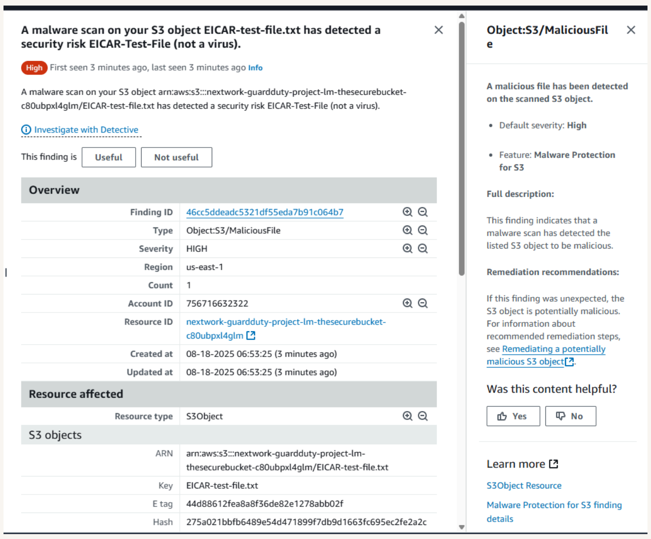
Instance details

## Extra: Malware Protection

For my project extension, I enabled GuardDuty Malware Protection for S3. Malware is malicious software designed to harm systems, steal data, or disrupt operations. By enabling this feature, GuardDuty can automatically scan my S3 bucket objects for threats, helping me detect and respond quickly to potential infections in stored files.

To test Malware Protection, I uploaded an EICAR test file into my protected S3 bucket. The uploaded file won't actually cause damage because it’s a harmless text file designed only to trigger antivirus and malware detection systems, helping confirm that GuardDuty’s Malware Protection is working correctly.

Once I uploaded the file, GuardDuty instantly triggered a malware detection finding for the EICAR test file. This verified that Malware Protection for S3 was active and correctly scanning new objects in the bucket, confirming that GuardDuty can identify and alert on malicious files as soon as they are uploaded.



**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 3 minutes ago, last seen 3 minutes ago Info

A malware scan on your S3 object arn:aws:s3:::nextwork-guardduty-project-lm-thesecurebucket-c80ubpxl4glm/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 | 46cc5ddeadc5321df55eda7b91c064b7 |
| Type | Object:S3/MaliciousFile |
| Severity | HIGH |
| Region | us-east-1 |
| Count | 1 |
| Account ID | 756716632322 |
| Resource ID | nextwork-guardduty-project-lm-thesecurebucket-c80ubpxl4glm |
| Created at | 08-18-2025 06:53:25 (3 minutes ago) |
| Updated at | 08-18-2025 06:53:25 (3 minutes ago) |

**Resource affected**

| | |
|---|---|
| Resource type | S3Object |

S3 objects

| | |
|---|---|
| ARN | arn:aws:s3:::nextwork-guardduty-project-lm-thesecurebucket-c80ubpxl4glm/EICAR-test-file.txt |
| Key | EICAR-test-file.txt |
| E tag | 44d88612fea8a8f36de82e1278abb02f |
| Hash | 275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c |

**Object:S3/MaliciousFile**

**A malicious file has been detected on the scanned S3 object.**

- Default severity: **High**
- Feature: **Malware Protection for S3**

**Full description:**

This finding indicates that a malware scan has detected the listed S3 object to be malicious.

**Remediation recommendations:**

If this finding was unexpected, the S3 object is potentially malicious. For information about recommended remediation steps, see Remediating a potentially malicious S3 object.

Was this content helpful?

Yes No

Learn more

S3Object Resource

Malware Protection for S3 finding details