



[nextwork.org](https://nextwork.org)

# Threat Detection with GuardDuty



Peter Delgado

GuardDuty

Summary

Findings

Malware scans

Protection plans

S3 Protection

EKS Protection

Extended Threat Detection

Runtime Monitoring

Malware Protection

RDS Protection

Lambda Protection

Accounts

Usage

Suppression rules

Settings

Lists

What's New

Partners

Security Hub

Findings (2)

Create suppression rule

Actions

Saved filters

Apply saved filters

Filter findings

Status

Current

Threat type

All findings

Title

The API GetEnvironmentStatus w

Credentials for the EC2 instance s

4PetrVNB5I0w were used from a

Credentials for the EC2 instance role NextWork-GuardDuty-project-peter-TheRole-4PetrVNB5I0w were used from a remote AWS account.

First seen 19 minutes ago, last seen 19 minutes ago

Credentials created exclusively for an EC2 instance using instance role NextWork-GuardDuty-project-peter-TheRole-4PetrVNB5I0w have been used from a remote AWS account 122545045921.

Investigate with Detective

This finding is Useful Not useful

Overview

Finding ID 7ecdec57050815519e8e4a111fedd12

Type UnauthorizedAccessIAMUser/InstanceCredentialExfiltration.insideAWS

Severity HIGH

Region us-east-1

Count 1

Account ID 910445377331

Resource ID nextwork-guardduty-project-peter-thesecurebucket-uzgid120dbs

Created at 01-20-2025 17:33:44 (5 minutes ago)

Updated at 01-20-2025 17:33:44 (5 minutes ago)

Resource affected

Resource role TARGET

Resource type S3Bucket

Access key ID ASIASHEWFGP8SXHOORIV

Principal ID ARDASH6WFGP8QREJ3CY7A3-07eb952ee9094e0df

User type AssumedRole

User name NextWork-GuardDuty-project-peter-TheRole-4PetrVNB5I0w

Instance details

Instance ID i-07eb952ee9094e0df

Instance type t2.micro

Instance state running

Availability zone us-east-1a





**Peter Delgado**  
NextWork Student

[nextwork.org](https://nextwork.org)

# Introducing Today's Project!

## Tools and concepts

The services we used were GuardDuty, CloudFormation, S3, and CloudShell. Key concepts learned were SQL, Command injections, using Linux commands like wget, cat, and jq, and malware protection

## Project reflection

too long.

I did this project to extend my knowledge and to learn how to use security resources on AWS and how they work together to protect a website.





## Project Setup

To set up for this project, we deployed a CloudFormation template that launches ten insecure web app (OWASP Juice Shop). The three main components are the web app infrastructure, an S3 bucket, and GuardDuty protecting our environment

The web app deployed is called WASP Juice Shop. To practice our GuardDuty skills, we will attack the Juice Shop, and then visit the GuardDuty console to detect and analyze its findings - does it pick up on our attacks on our web app?

AWS GuardDuty is a threat detection service, which means it helps you find potential security risks or attacks in your apps and AWS environment. It uses machine learning to look for unusual activity in your AWS account, like your network traffic and CloudTrail activity logs. If it finds something suspicious, it will alert you so you can investigate.





**Peter Delgado**  
NextWork Student

[nextwork.org](https://nextwork.org)

NextWork-GuardDuty-project-peter

Stack info | Events | Resources | Outputs | Parameters | Template | Change sets | Git sync

**Overview**

Stack ID: [arn:aws:cloudformation:us-east-1:910445327331:stack/NextWork-GuardDuty-project-peter/13a4d00-1629-11f0-96f6-d62612715871](#)

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	2026-01-20 12:54:33 UTC-0500
		Updated time	-
Deleted time	-	Drift status	NOT_CHECKED
Last drift check time	-	Termination protection	Deactivated
IAM role	-		

**Latest operations**

Operation 1

Operation ID: [#5d93758-acc4-4150-b9b2-4322a933ab99](#)

Operation type





# SQL Injection

SQL injection is a web vulnerability that happens when an attacker can insert malicious SQL code into an application's database queries. This can let them bypass authentication, steal data, or even change the database structure.

I typed in SQL injection ' or 1=1;-- into the email field of the login page. This means the login query will always evaluate to true, meaning our database will see this as a valid email and password and allow the person to login

The screenshot shows the OWASP Juice Shop login page. The header includes the site logo, name, and navigation links. The login form has fields for email and password. The email field contains the SQL injection payload ' or 1=1;--. The password field is empty. Below the fields are links for 'Forgot your password?' and a 'Log in' button. A 'Remember me' checkbox is also present. At the bottom of the form, there is a link for 'Not yet a customer?'. A cookie consent banner is visible at the bottom right of the page.

OWASP Juice Shop

Search Account EN

Login

Email \*

' or 1=1;--

Password \*

Forgot your password?

Log in

Remember me

Not yet a customer?

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

Me want it!





# Command Injection

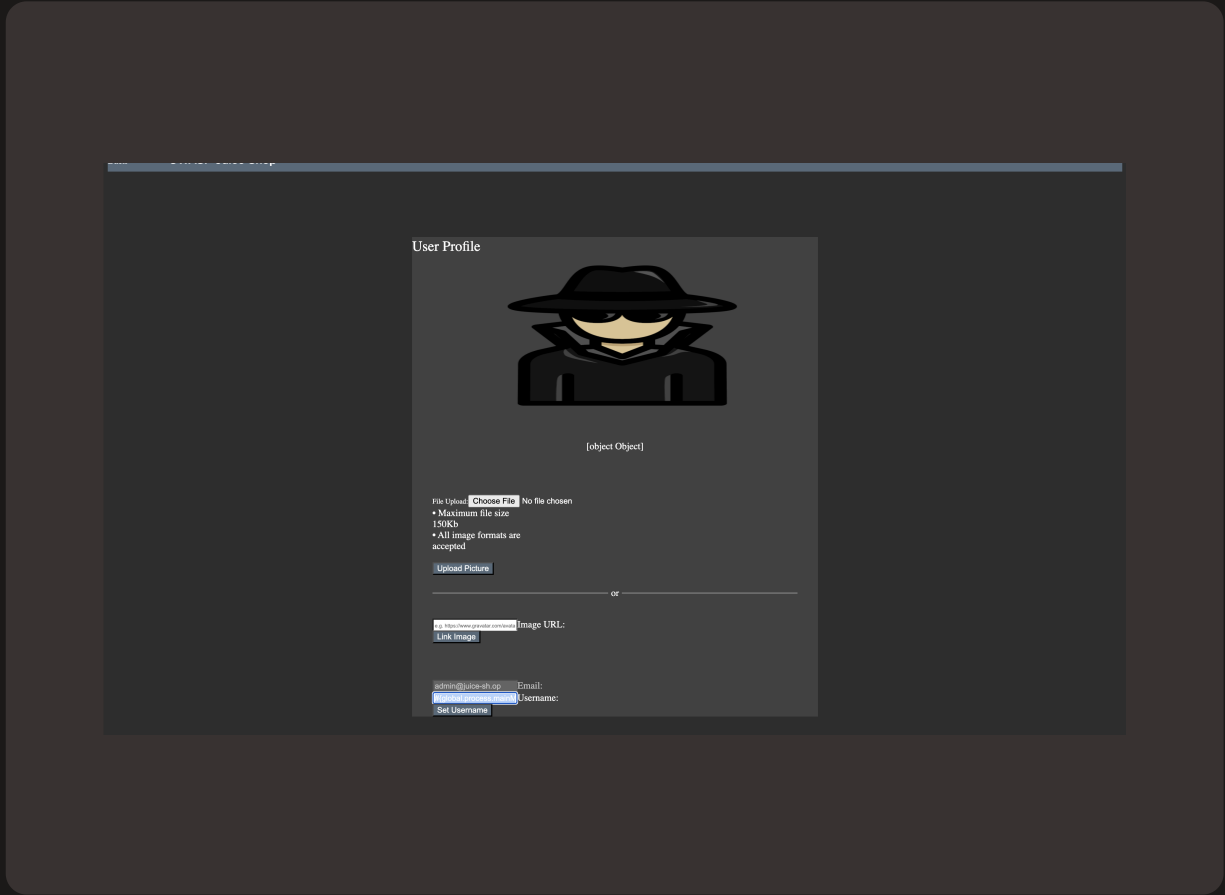
Command injection is a piece of code. All together, these instructions fetch highly sensitive information (IAM credentials to your AWS environment) and store them in a place where anyone on the internet can access them! The Juice Shop web app is vulnerable because it is not set up to sanitize code, meaning it won't scan the username to make sure it's not a script instead of a plain text username.

To run command injection, I pasted the code into the username field. The script will fetch highly sensitive information (IAM credentials to your AWS environment) and store them in a place where anyone on the internet can access them!





nextwork.org







## Attack Verification

To verify the attack's success, we visited the publicly exposed credentials file. This page showed us access keys that represent our EC2 instance's access to our AWS environment. We can use keys to get the same level of access.

```
Pretty-print
{
  "AccessKeyId": "ASIASHEWFGRRSXHQRIV",
  "Code": "Success",
  "Expiration": "2026-01-21T01:02:22Z",
  "LastUpdated": "2026-01-20T18:35:43Z",
  "SecretAccessKey": "eoaT00K08S0m/a2vqg8BVZ-whYR1G9TFznn3n",
  "Token": "
\"Do3b3p2Z1uX2VjED0v////////vEaCYZLWhc3QMS3HMEUCH+1e7Y/24ZY+nul8n031z8foMY1QRx0Hzw/7BRrA1EAp+FYt03+60m6fX08sl115/v9AMy1twt3b5uQU/EqQU1TP/////////AAAGvSHTA8NDUJmJc2MzE0Bd1UAJub1L/9CpEv1qYBURK+66/xSBMTx
9vW/qMaTH2Z7mKQ0aX0/00wJ7jXsA0SVe0TVp4/ZZ108KwT2LAb6J2a1c28rTXf4kvzFUG4HLS5G3x0R4tc305J2k0UwE2jYp0sk2uzEP1L4K0bzZp0E01PEFG0av0PyW0YJ30Mc9YL0y0BhF183/Pr8h0Zd++Xy0w186gic0x/cd0Yvha0Yq0B0z0b1L2yrtJ1k1R
F31vY0h0TUN5yQ0B0R0g0p0h0c1K1P0Q0N0A0a0p0R03S0M0C0vF0d0q0d1lYronT02vX/v76R0M0A0p0400y01n1L5p0y0X2k0C0CW0BvF0L0r00N0w17p0N0Y0W0c0w0ALV2AS0L0A0f0h0N0Z0p0u0SE0d0Sp0p0H0g0v0B0R0M0e0v0E03FJ0S007010
U0w0Y0P018t0P0M0K0A0N0a0r0g0g0T0C5/g0P3f1CZ0r0u0v0g02B0t0K0f0u0N0p011C0ag0P0a0M0e0JUL7a0Kf+/AU09v0h2Lq0X0b07K0k0W09J0K03g0w0n0QJ11v7m0x0M27/atbP0W0R1S0g07zjh6Z0H0M0p0KLh0K0d0eL1W0g0V0g0B0u0b0FEU0Z0K0T0K0C90a1p0K0AN0v7Z
m0K0W0g0J0H0W0M0E1104W031p022070d0b0g0d0g00F7A010M0A0K0a0p0v0V030Q0Z0L0g0u1C0P0M0Y0A0p0Y0J0M0E11A0f0d0d0L0q0e0P0W0K010H0m0M010Z0v0m0M01L0K0E20Z0T0L0D0P0f0Q0U0K0u0L0W0J0W0m0m0S0J0u0J0G0Z0K0h0W0m020v0y0K0N
g0h020R0Z0n0370W0F0E05J0N0M07H0P0h0r0v0p0R0b020124P0j050B0t0L0d0c0B0d0f0M0q0Te0f0L2m0S080p0S0R0B0w0B40f050B0P0V0J0N0V1Z0p01v0c2C0g0H0U0TJ100z0Z0F15C0X0a07f0R0J0C0A0g0r0y0p0u0N0R01x0t0nd0g0z0M+2g0m0F0M0G0Z0P0T0A0Z0q0p0K112CAJ050K0a0Z0w0y0g
01p0Y070H0B0a0E0\"
  \"type\": \"AWS-IMAC\"
}
```





## Using CloudShell for Advanced Attacks

In this step, we're trying to get access to private data using stolen credentials. To do this, you have to keep the hacker hat on and try to hack into your own AWS environment. We'll use CloudShell to run commands that a hacker would run to steal data inside your AWS environment.

In CloudShell, we used `wget` to download the exposed credentials file into our CloudShell environment. Next, we ran a command using `cat` and `jq` to read the downloaded file and format it nicely so the credentials (in JSON) are easy to understand.

We then set up a new profile using all of the stolen credentials. We had to create a new profile because the hacker doesn't inherently have access to the victim's AWS environment. We'll need to use the profile to switch permission settings.



[illegible]





## GuardDuty's Findings

GuardDuty can take up to 15 minutes to detect that something suspicious has happened.

GuardDuty's finding was called  
UnauthorizedAccess:|AMUser/InstanceCredentialExfiltration.InsideAWS, which means credentials belonging to my EC2 instance were being used in another account. Anomaly detection was used because this was unusual behavior.

GuardDuty's finding was called  
UnauthorizedAccess:|AMUser/InstanceCredentialExfiltration.InsideAWS, which means credentials belonging to my EC2 instance were being used in another account. Anomaly detection was used because this was unusual behavior.





GuardDuty > Findings

GuardDuty

Summary

Findings

Malware scans

Protection plans

S3 Protection

EKS Protection

Extended Threat Detection New

Runtime Monitoring

Malware Protection

RDS Protection

Lambda Protection

Accounts

Usage

Suppression rules New

Settings

Lists

What's New

Partners

Security Hub

Findings (2)

Create suppression rule

Actions

Saved filters

Apply saved filters

Filter findings

Status

Current

Threat type

All findings

Title

The API GetEnvironmentStatus w

Credentials for the EC2 instance r

4PetrVNB5l0w were used from a

Credentials for the EC2 instance role NextWork-GuardDuty-project-peter-TheRole-4PetrVNB5l0w were used from a remote AWS account.

First seen 19 minutes ago, last seen 19 minutes ago

Credentials created exclusively for an EC2 instance using instance role NextWork-GuardDuty-project-peter-TheRole-4PetrVNB5l0w have been used from a remote AWS account 129543045921.

Investigate with Detective

This finding is

Useful

Not useful

Overview

Finding ID	7ecdeec57050815519e8e4a111fdd12
Type	UnauthorizedAccessIAMUser/InstanceCredentialExfiltration.InsideAWS
Severity	HIGH
Region	us-east-1
Count	1
Account ID	910445327331
Resource ID	nextwork-guardduty-project-peter-theseurebucket-uzpidd120dbe
Created at	01-20-2026 17:33:44 (5 minutes ago)
Updated at	01-20-2026 17:33:44 (5 minutes ago)

Resource affected

Resource role	TARGET
Resource type	S3Bucket
Access key ID	ASIASH6WFGPRQREJ3CY7A-i-07eb952ee9094e0bf
Principal ID	ARQASH6WFGPRQREJ3CY7A-i-07eb952ee9094e0bf
User type	AssumedRole
User name	NextWork-GuardDuty-project-peter-TheRole-4PetrVNB5l0w

Instance details

Instance ID	i-07eb952ee9094e0bf
Instance type	t2.micro
Instance state	running
Availability zone	us-east-1a





## Extra: Malware Protection

For our project extension, we enabled Malware Protection for S3. Malware is a file that contains threats, e.g., opening the file will cause a data breach or the deletion of resources.

GuardDuty's detailed finding reported that an S3 bucket was affected; the action that was done using the stolen credentials was GetObject, and the EC2 instance whose credentials were leaked. The IP address + location of the actor was also available.

Start your answer with 'Once I uploaded the file, GuardDuty instantly triggered a finding called object:S3/MaliciousFile. This verified that GuardDuty could successfully detect malware. It also mentioned that the threat type is EICAR-Test-File (not a virus)





GuardDuty > Findings

GuardDuty

Summary

Findings

Malware scans

Protection plans

S3 Protection

EKS Protection

Extended Threat Detection New

Runtime Monitoring

Malware Protection

RDS Protection

Lambda Protection

Accounts

Usage

Suppression rules New

Settings

Lists

What's New

Partners LA

Security Hub LA New

Findings (2) Info

Create suppression rule

Actions

Saved filters

Apply saved filters

Filter findings

State

Current

Threat type

All findings

Title

The API GetEnvironmentStatus w

Credentials for the EC2 instance : 4PetrVNB5I0w were used from a

Credentials for the EC2 instance role NextWork-GuardDuty-project-peter-TheRole-4PetrVNB5I0w were used from a remote AWS account.

New First seen 19 minutes ago, last seen 19 minutes ago

Credentials created exclusively for an EC2 instance using instance role NextWork-GuardDuty-project-peter-TheRole-4PetrVNB5I0w have been used from a remote AWS account 123543045921.

Investigate with Detective

This finding is Useful Not useful

Overview

Finding ID	7ee0ee57050815519e8e4a111fedd12	🔍
Type	UnauthorizedAccess/IAMUser/InstanceCredentialExfiltration.insideAWS	🔍
Severity	HIGH	🔍
Region	us-east-1	🔍
Count	1	
Account ID	910445327331	🔍
Resource ID	nextwork-guardduty-project-peter-thesebucket-uzpid120dbe <span>LA</span>	
Created at	01-20-2026 17:33:44 (5 minutes ago)	
Updated at	01-20-2026 17:33:44 (5 minutes ago)	

Resource affected

Resource role	TARGET	🔍
Resource type	S3Bucket	🔍
Access key ID	ASIASH6WFGPR5XHOORIV	🔍
Principal ID	ARIASH6WFGPRQREJ3CY7A3-07eb952ee9094e0bf	🔍
User type	AssumedRole	🔍
User name	NextWork-GuardDuty-project-peter-TheRole-4PetrVNB5I0w	🔍

Instance details

Instance ID	i-07eb952ee9094e0bf	🔍
Instance type	t2.micro	
Instance state	running	
Availability zone	us-east-1a	