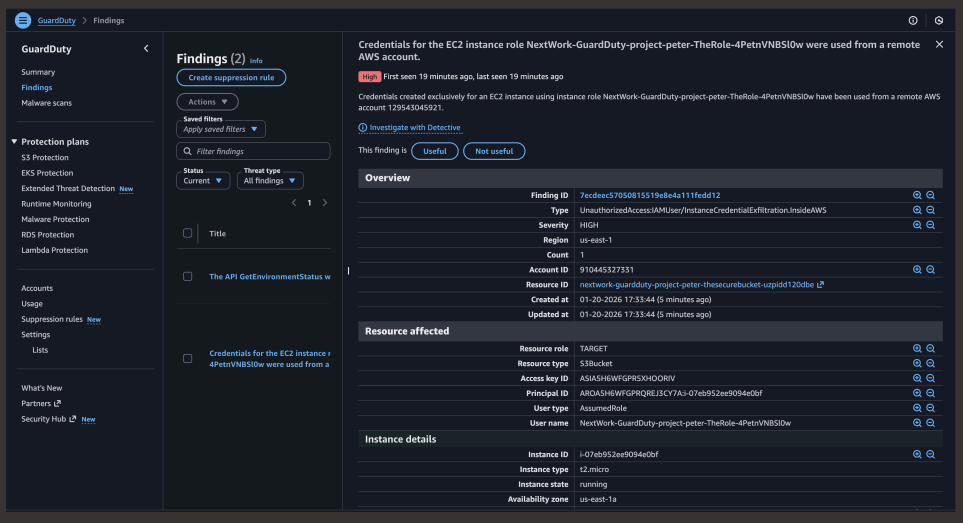
nextwork.org

Threat Detection with GuardDuty

Peter Delgado



Peter Delgado
NextWork Student

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.

Peter Delgado
NextWork Student

Project Setup

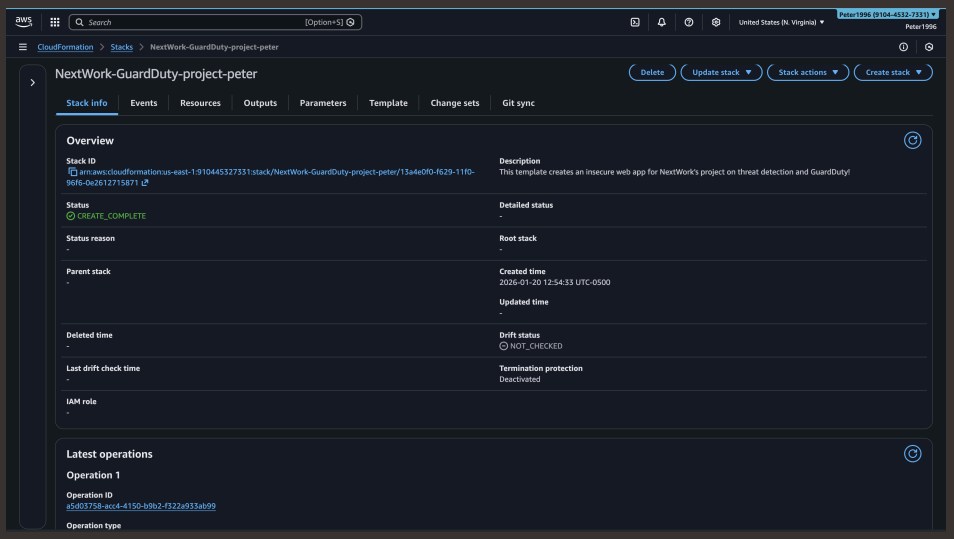
To set up for this project, we deployed a CloudFormation template that launches ten insecure web app (OWASP Juice Shop. The three main components are the web app infrastructure, an S3 bucket, and GuardDuty protecting our environment

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



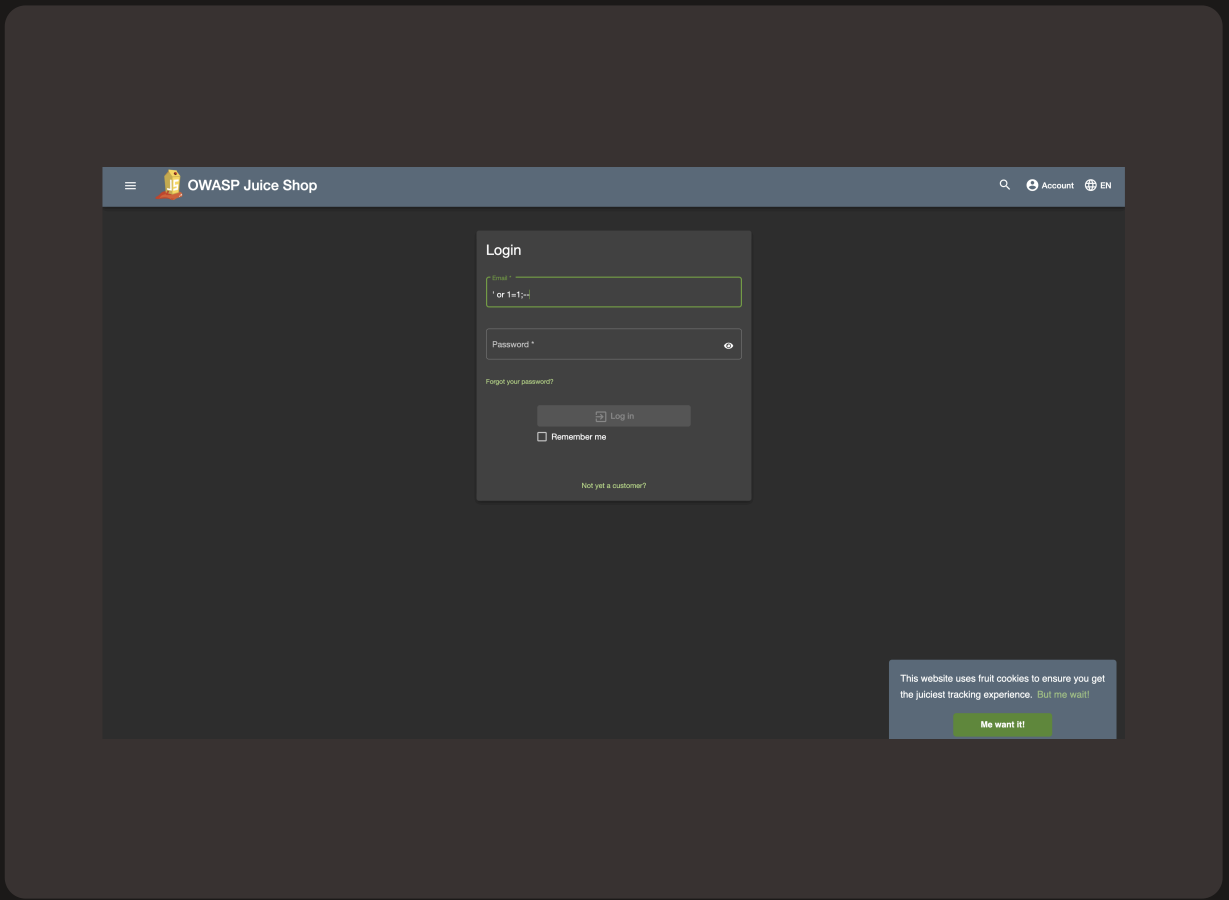
Peter Delgado
NextWork Student

nextwork.org

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



Peter Delgado
NextWork Student

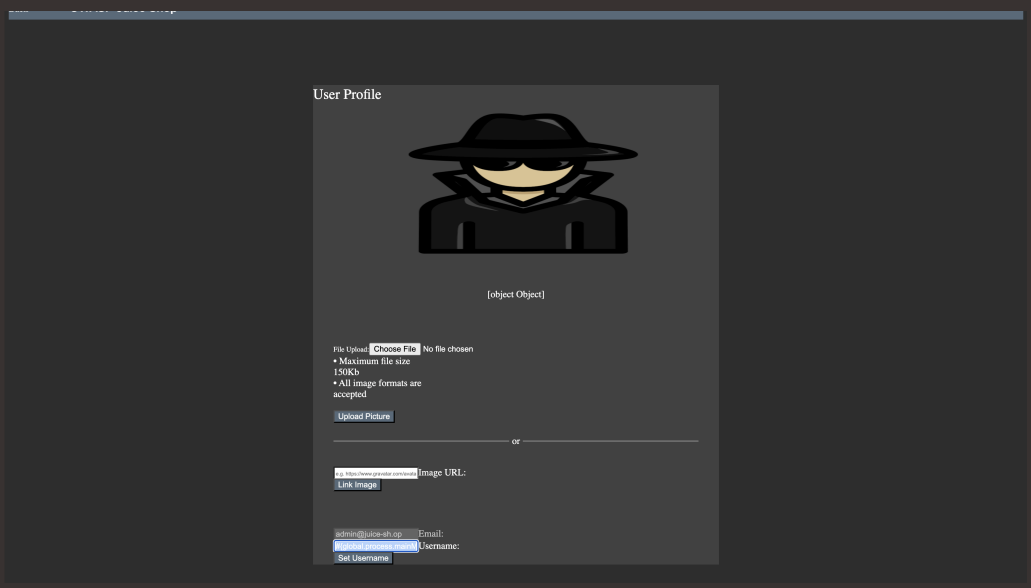
Command Injection

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!

Peter Delgado
NextWork Student

nextwork.org

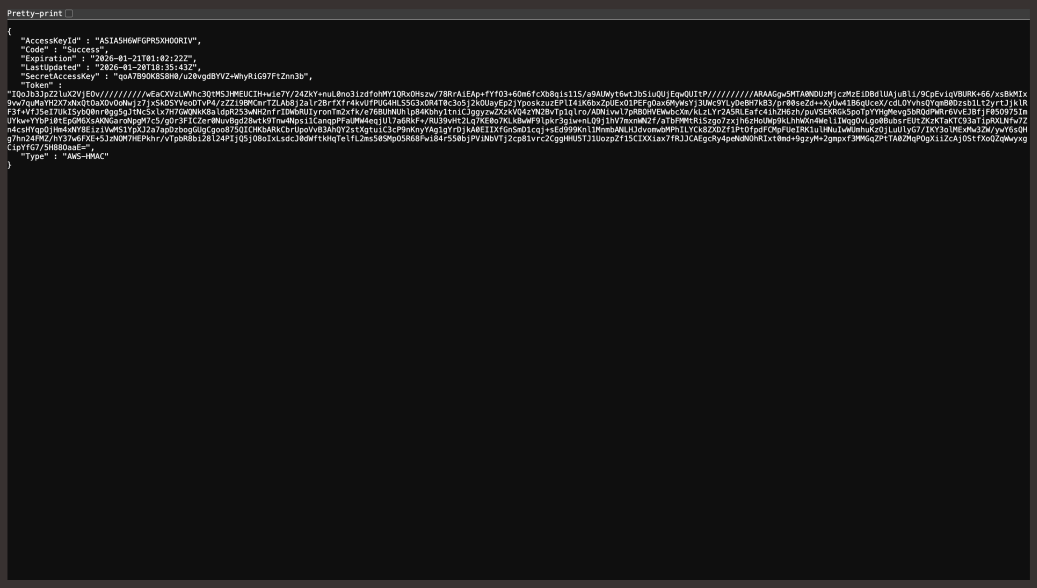


Peter Delgado
NextWork Student

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.



Peter Delgado
NextWork Student

Using CloudShell for Advanced Attacks

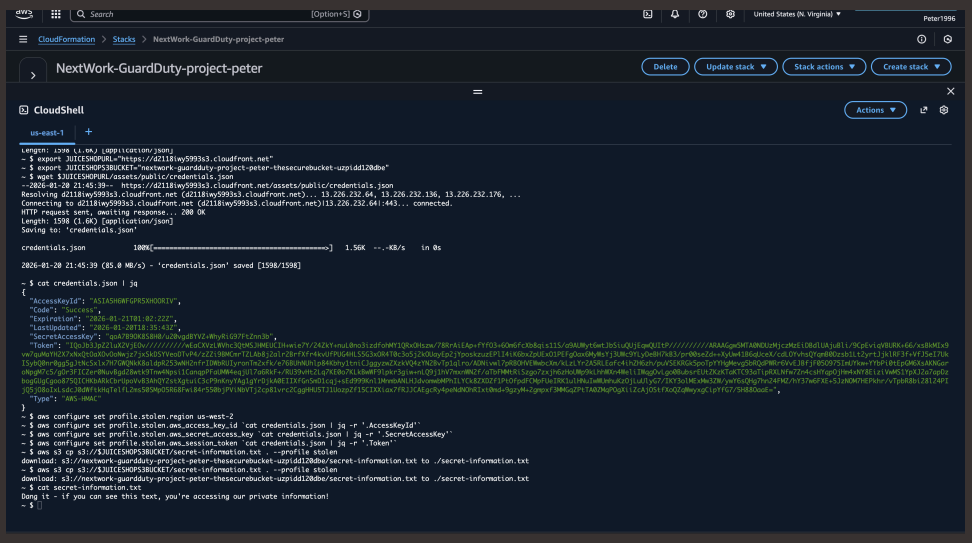
In this step, we're trying to get access to private data using stolen credentials. To do this, you have to keep the hacker hat on and try to hack into your own AWS environment. We'll use CloudShell to run commands that a hacker would run to steal data inside your AWS environment.

In CloudShell, we used wget to download the exposed credentials file into our CloudShell environment. Next, we ran a command using cat and ja 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.

Peter Delgado
NextWork Student

nextwork.org



Peter Delgado
NextWork Student

GuardDuty's Findings

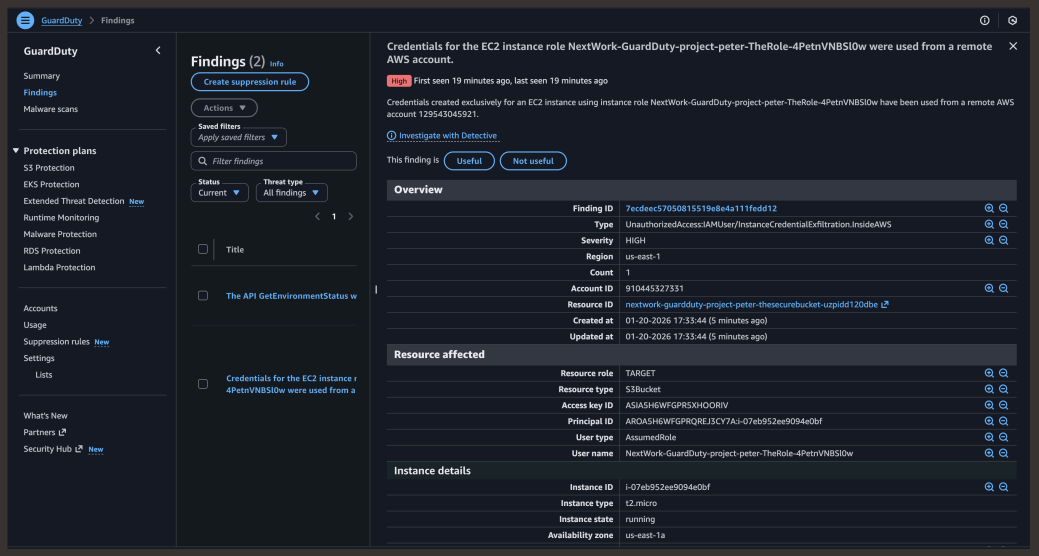
GuardDuty can take up to 15 minutes to detect that something suspicious has happened.

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.

Peter Delgado
NextWork Student

nextwork.org



Peter Delgado
NextWork Student

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)

Peter Delgado
NextWork Student

nextwork.org