

Cybersecurity Cohort Capstone: Technical Documentation (Sprints 1-5)

Project Goal

This project aims to establish a scalable and automated Security Operations Center (SOC) on AWS to centralize security monitoring, incident detection, and response capabilities. Secondary goals include improving security posture and reducing response times to security incidents.

Target Audience

This SOC platform is designed for security operations professionals, IT security teams, and security analysts within an organization.

Success Metrics

- Reduce alert fatigue by at least 30% within six months of SOC deployment.
- Increase incident response times by 20%.
- Achieve an average incident response time of less than 15 minutes for critical security events.
- Improve security visibility with centralized monitoring and reporting capabilities.
- Streamline incident response workflows through automation playbooks and improve overall response efficiency.

Architecture Design

Data Sources

- Network Security Devices (pfSense firewall)
- Cloud Infrastructure Logs (VPC Flow Logs, DNS Logs, CloudTrail Logs)

Log Ingestion and Preprocessing

- Amazon Kinesis Firehose: Collects logs from various sources and prepares them for further analysis by Splunk Cloud.

Log Management and Analysis

- Splunk Cloud (SIEM): Analyzes logs for security events, anomalies, and potential threats. Integrates with EOpen threat intelligence feed.

Threat Detection and Vulnerability Assessment

- Amazon GuardDuty: Monitors your AWS environment for malicious activity.
- Amazon Inspector: Scans EC2 instances for vulnerabilities and exposures.

Security Orchestration, Automation, and Response (SOAR)

- AWS Lambda (Automated Responses): Creates custom functions for repetitive tasks (e.g., quarantining infected instances).
- AWS Security Hub (Optional): Central repository for security findings from various AWS services.

Security Visualization

- Kibana (integrated with Splunk Cloud): Provides real-time dashboards and reports for security posture awareness.

Threat Intelligence Feed

- Emerging Threats Open (ETOpen) - Open Source Tool: Provides valuable threat intelligence data to Splunk Cloud.

IAM Roles with Least Privilege

- IAM ensures secure access control by assigning roles with least privilege to users and resources.

How They Work Together

1. **Log Collection:** Logs are generated from various sources and collected by Kinesis Firehose.
2. **Log Preprocessing and Delivery:** Kinesis Firehose preprocesses logs and delivers them to Splunk Cloud.
3. **Log Analysis and Threat Detection:** Splunk Cloud analyzes logs and leverages threat intelligence to identify potential threats.
4. **Alert Generation:** If suspicious activity is detected, Splunk Cloud generates security alerts.
5. **Investigation and Response:** Security analysts investigate alerts and determine appropriate responses.
6. **Automation (Optional):** AWS Lambda functions can be triggered for automated responses to high-severity alerts.
7. **Centralized Findings (Optional):** Security Hub can act as a central repository for findings, aiding in comprehensive investigation.
8. **Security Visualization:** Kibana provides real-time visualizations of security data for situational awareness.

Alternative: AWS OpenSearch Service

OpenSearch is an open-source alternative to Splunk Cloud for log analysis and works well with Kibana for visualization.

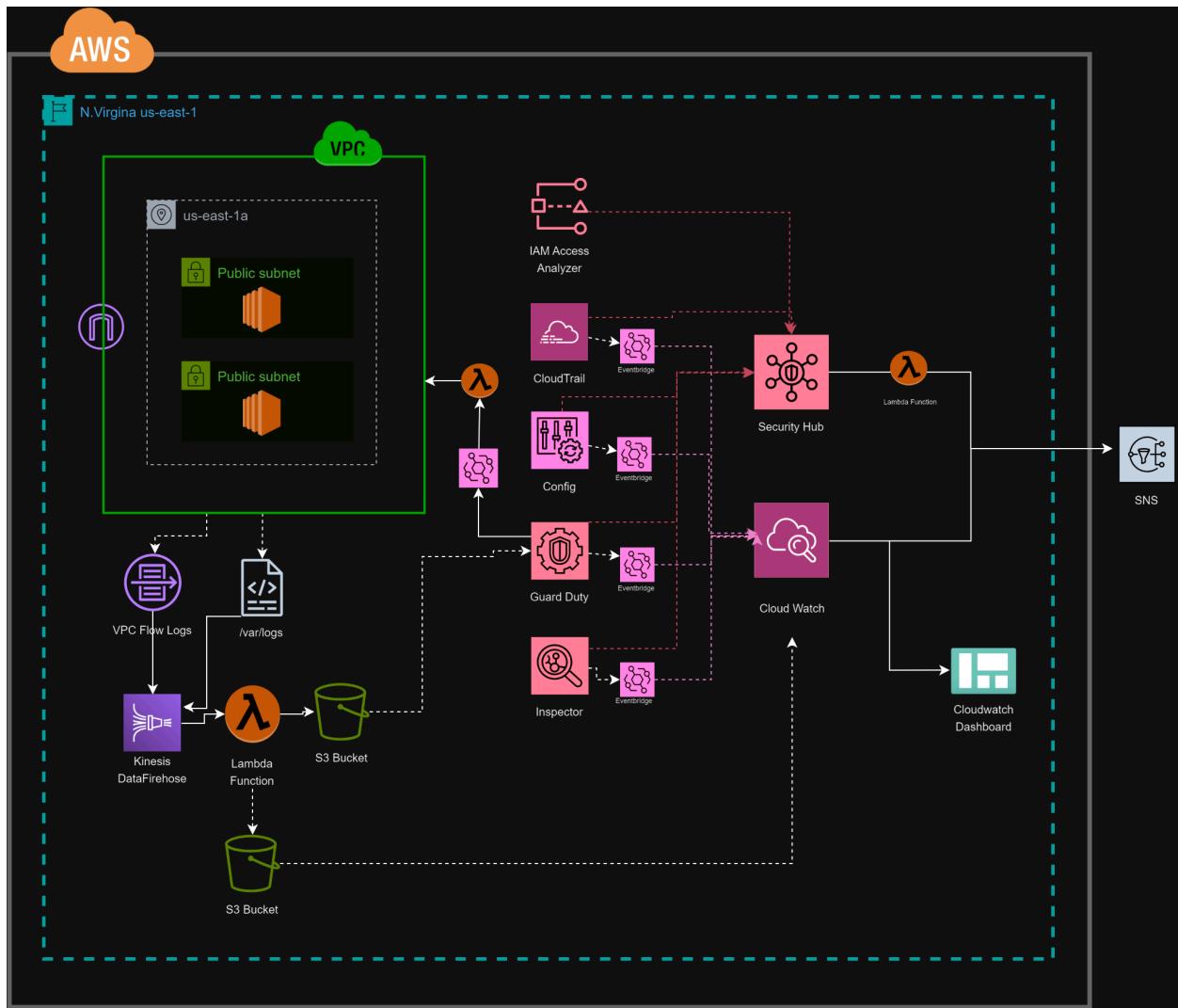
Research Highlights

- Optimizing data ingestion for threat coverage is crucial for effective SOC operations.
- Accurate data intake is essential for successful SOC functionality.
- Transitioning from basic log collection to true threat detection and response is important.

Additional Resources

- AWS re:Inforce 2022: Using AWS security services to build your cloud security operations baseline.
- SOCGUIDE: <https://github.com/cyb3rxp/awesome-soc>

Network Diagram



IAM Policies and Configuration

- IAM policies define access controls for users within the SOC, adhering to the principle of least privilege for security.
- Role-Based Access Control (RBAC) is implemented to assign permissions based on job responsibilities.

Security Data Ingestion Pipelines

This section outlines the security data ingestion pipelines for the SOC platform, covering data sources, ingestion architecture, automation and scalability, error handling, log management and normalization, and access control.

- **Data Source Identification:**

- Cloud Service Logs (VPC Flow Logs, GuardDuty Findings)
- Network Security Appliance Logs (Firewalls, IPS/IDS, WAF)

- **Data Ingestion Architecture:**

- Log Forwarding Mechanisms (Amazon EventBridge, CloudTrail)
- Data Ingestion Services (Amazon Kinesis Data Streams)

Testing and Validation

This sprint focuses on testing and validating the functionalities developed in previous stages.

Testing Scenarios and Playbook Validation

- Develop test scenarios simulating security incidents. These scenarios could involve injecting test data into CloudTrail logs or VPC Flow Logs to trigger playbooks and validate their functionality.
- Ensure playbooks function as intended and produce the desired outcomes during simulated attacks.

Security of Playbooks

- Implement access controls and permissions within the SIEM solution to restrict unauthorized modification of playbooks. Regularly review and update playbooks to ensure they remain effective and don't introduce vulnerabilities.

Threat Detection & Alert Correlation

SIEM Solution and Data Sources

- **SIEM Solution:** We'll leverage Amazon CloudWatch Logs as the central log management solution. CloudTrail, VPC Flow Logs, and Inspector findings can all be integrated with CloudWatch Logs for centralized analysis.
- **Data Sources:** Data sources include CloudTrail logs (API activity), VPC Flow Logs (network traffic), and Inspector findings (vulnerability scans).

Normalization and Enrichment

- CloudWatch Logs offers log parsing capabilities to normalize logs from different sources into a common format.
- Additional enrichment might involve integrating threat intelligence feeds to add context to log data.

Correlation Rules and Threat Detection

- Develop rules within CloudWatch Logs that analyze log data from various sources to identify potential threats. These rules might look for specific patterns in CloudTrail logs indicating suspicious activity or correlate network traffic logs with potential vulnerabilities identified by Inspector.
- Techniques could include pattern matching (e.g., identifying known malicious commands in CloudTrail logs), statistical analysis (e.g., detecting unusual spikes in network traffic), or simple rule-based correlations.
- Integrate threat intelligence feeds from reputable providers to enrich correlation rules with the latest indicators of compromise (IoCs). This helps identify emerging threats and suspicious activity patterns.

- Continuously monitor and refine correlation rules to reduce false positives and improve detection accuracy. Analyze false positives to understand the root cause and adjust rules accordingly.

Alert Prioritization and Triage

- Prioritize alerts based on a combination of factors like asset criticality (e.g., high-risk servers), threat level (e.g., potential data breach), and confidence score (e.g., high confidence based on multiple indicators).
- Alerts triggered by correlation rules can be routed to basic automation within the SIEM solution or integrated with a future SOAR platform for more complex workflows.
- Define high-fidelity alert scenarios (e.g., unauthorized login attempt from suspicious IP for a critical account) that trigger automated responses and notify security personnel for further investigation.

By incorporating these testing and validation steps, you can ensure your SOC platform effectively detects and responds to security threats.

Networking in AWS

VPC settings

Resources to create [Info](#)
Create only the VPC resource or the VPC and other networking resources.

VPC only VPC and more

Name tag - *optional*
Creates a tag with a key of 'Name' and a value that you specify.
SOC-VPC-01

IPv4 CIDR block [Info](#)
 IPv4 CIDR manual input IPAM-allocated IPv4 CIDR block

IPv4 CIDR
10.0.0.0/16
CIDR block size must be between /16 and /28.

IPv6 CIDR block [Info](#)
 No IPv6 CIDR block IPAM-allocated IPv6 CIDR block Amazon-provided IPv6 CIDR block IPv6 CIDR owned by me

Tenancy [Info](#)
Default

Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Step 1: Create a VPC

1. Log in to the AWS Management Console.

2. Navigate to the VPC Dashboard:

- Go to the "Services" dropdown and select "VPC".

3. Create a VPC:

- Click on "Your VPCs" in the left-hand menu.
- Click the "Create VPC" button.
- Fill in the details:
 - Name tag: Provide a name for your VPC.
 - IPv4 CIDR block: Enter the IPv4 CIDR block for your VPC (e.g., `10.0.0.0/16`).
 - IPv6 CIDR block: Optional, can be left empty unless required.
 - Tenancy: Default.
- Click "Create VPC".

Create subnet [Info](#)

VPC

VPC ID
Create subnets in this VPC.
vpc-027b385b18e3946b8 (SOC-VPC-01)

Associated VPC CIDRs

IPv4 CIDRs
10.0.0.0/16

Step 2: Create Subnets

1. Navigate to the Subnets section:

- Click on "Subnets" in the left-hand menu.

2. Create Subnet 1:

- Click the "Create subnet" button.
 - Select your VPC from the drop-down.
 - Fill in the details for Subnet 1:
 - Name tag: Provide a name for the first subnet.
 - Availability Zone: Select an availability zone.
 - IPv4 CIDR block: Enter the CIDR block for Subnet 1 (e.g., `10.0.1.0/24`).
 - Click "Create subnet".
3. Create Subnet 2:
- Click the "Create subnet" button again.
 - Select your VPC from the drop-down.
 - Fill in the details for Subnet 2:
 - Name tag: Provide a name for the second subnet.
 - Availability Zone: Select a different availability zone (optional).
 - IPv4 CIDR block: Enter the CIDR block for Subnet 2 (e.g., `10.0.2.0/24`).
 - Click "Create subnet".

Subnet settings

Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 2

Subnet name

Create a tag with a key of 'Name' and a value that you specify.

The name can be up to 256 characters long.

Availability Zone [Info](#)

Choose the zone in which your subnet will reside, or let Amazon choose one for you.



IPv4 VPC CIDR block [Info](#)

Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.



IPv4 subnet CIDR block

256 IPs



▼ Tags - optional

Key

Value - optional



[Remove](#)

[Add new tag](#)

You can add 49 more tags.

Subnet 2 of 2

Subnet name

Create a tag with a key of 'Name' and a value that you specify.

The name can be up to 256 characters long.

Availability Zone [Info](#)

Choose the zone in which your subnet will reside, or let Amazon choose one for you.



IPv4 VPC CIDR block [Info](#)

Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.



IPv4 subnet CIDR block

256 IPs



▼ Tags - optional

Key

Value - optional



You can add 49 more tags.

Step 3: Create a Route Table

1. Navigate to the Route Tables section:

- Click on "Route Tables" in the left-hand menu.

2. Create a Route Table:

- Click the "Create route table" button.
- Select your VPC from the drop-down.
- Fill in the details:

- Name tag: Provide a name for the route table.
- Click "Create route table".

VPC > Route tables > Create route table

Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.

SOC-routetable

VPC
The VPC to use for this route table.

vpc-0c0d497df62299ae4 (SOC-VPC-01)

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - <i>optional</i>	Remove
<input type="text" value="Name"/> X	<input type="text" value="SOC-routetable"/> X	Remove

Add new tag

You can add 49 more tags.

Cancel Create route table

Step 4: Create the Internet Gateway (IGW)

1. Navigate to the Internet Gateways section:
 - Click on "Internet Gateways" in the left-hand menu.
2. Create an Internet Gateway:
 - Click the "Create internet gateway" button.
 - Fill in the details:

- Name tag: Provide a name for the IGW.
- Click "Create internet gateway".

VPC > Internet gateways > Create internet gateway

Create internet gateway Info

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings

Name tag
Creates a tag with a key of 'Name' and a value that you specify.

Tags - optional
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional	Remove
<input type="text" value="Name"/> X	<input type="text" value="SOC-InternetGateway"/> X	Remove

Add new tag
You can add 49 more tags.

Cancel **Create internet gateway**

VPC > Internet gateways > Attach to VPC (igw-0c0cc96e5ddc302cc)

Attach to VPC (igw-0c0cc96e5ddc302cc) Info

VPC
Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs
Attach the internet gateway to this VPC.
 X

▶ AWS Command Line Interface command

Cancel **Attach internet gateway**

Step 5: Attach the Gateway to the VPC

1. Attach the IGW:

- Select the IGW you just created.
- Click the "Actions" button and choose "Attach to VPC".
- Select your VPC from the drop-down.
- Click "Attach internet gateway".

The screenshot shows the 'Edit routes' page for a route table. It has a table with columns: Destination, Target, Status, and Propagated. A new row is being added with 'Destination' set to '10.0.0.0/16', 'Target' set to 'local', and 'Status' set to 'Active'. Below the table are 'Add route' and 'Remove' buttons. At the bottom are 'Cancel', 'Preview', and 'Save changes' buttons.

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
Q 0.0.0.0/0 X	Internet Gateway	-	No
Q igw-0c0cc96e5ddc302cc X			

Step 6: Add the IGW Route to Your Route Table

1. Modify the Route Table:

- Select the route table you created.
- Click the "Routes" tab.
- Click the "Edit routes" button.
- Click "Add route".
- Fill in the details:
 - Destination: `0.0.0.0/0` (this means all traffic).
 - Target: Select your Internet Gateway.
- Click "Save routes".

The screenshot shows the 'Edit subnet associations' page for a route table. It has two sections: 'Available subnets (2/2)' and 'Selected subnets'. Under 'Available subnets', there is a table with columns: Name, Subnet ID, IPv4 CIDR, IPv6 CIDR, and Route table ID. Two subnets are listed: 'Computer-01' and 'Computer-02'. Under 'Selected subnets', two subnets are selected: 'subnet-02e9e7a53ad765159 / Computer-01' and 'subnet-079dc1990ab406b3d / Computer-02'. At the bottom are 'Cancel' and 'Save associations' buttons.

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
Computer-01	subnet-02e9e7a53ad765159	10.0.1.0/24	-	Main (rtb-04818bb6b663fe8d5)
Computer-02	subnet-079dc1990ab406b3d	10.0.2.0/24	-	Main (rtb-04818bb6b663fe8d5)

Step 7: Associate Both Subnets to the Route Table

1. Associate Subnet 1:

- Select the route table you created.
- Click the "Subnet associations" tab.
- Click the "Edit subnet associations" button.

- Select Subnet 1 from the list.
 - Click "Save associations".
2. Associate Subnet 2:
- Repeat the steps above for Subnet 2.

You have now created a VPC with two subnets, set up an internet gateway, created a route table, and associated your subnets with the route table. This setup allows your subnets to route traffic through the internet gateway, providing internet access to resources within those subnets.

This outlines the security data ingestion pipelines for the SOC platform, covering data sources, ingestion architecture, automation and scalability, error handling, log management and normalization, and access control.

1. Security Data Ingestion Pipelines (Matthew Escalera, Pedro Gomez)

1.1 Data Source Identification

The SOC platform will ingest security data from a variety of sources to provide a comprehensive view of the security posture. Here's a list of potential data sources:

- **Cloud Service Logs:**
 - VPC Flow Logs: Records network traffic flowing in and out of VPCs.
 - GuardDuty Findings: For collecting logs.
- **Network Security Appliance Logs:**
 - Firewalls: Logs security events like blocked traffic and access attempts.
 - Intrusion Prevention/Detection Systems (IPS/IDS): Alerts on suspicious network activity.
 - Web Application Firewalls (WAF): Logs attempts to exploit web application vulnerabilities.

1.2 Data Ingestion Architecture

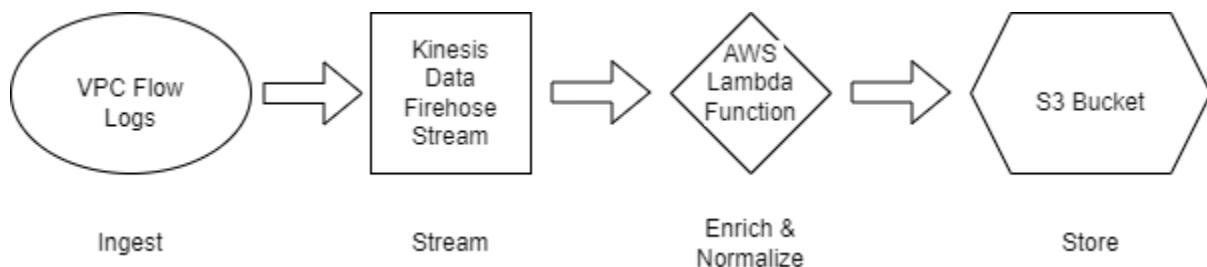
The data ingestion architecture will leverage various services to collect, transform, and deliver security data to the log management platform. Here's a breakdown of the key components:

- **Log Forwarding Mechanisms:**
 - Amazon EventBridge: A standard protocol for sending logs from devices to a central server, used for Aws config, Inspector and Guard Duty.

- Cloud Trail made to native settings to send logs to CloudWatch, VPC flow logs has the same thing.
- **Data Ingestion Services:**
 - Amazon Kinesis Data Streams: Real-time streaming data for high-volume log ingestion.
 - AWS Lambda: Used for converting information coming from VPC flow logs and parsing and enriching it into a more readable log.
- **Data Transformation and Enrichment:**
 - AWS Lambda Functions: Serverless functions for real-time data processing and transformation tasks.

1.3 Data ingestion pipeline architecture (sample)

VPC FLOW LOGS → Kinesis Data Firehose Stream → AWS Lambda Function (data is being enriched normalized here) → S3 bucket (where the results are being stored).



1.3 Error Handling and Monitoring

```

import logging

logger = logging.getLogger()

logger.setLevel(logging.INFO)

def lambda_handler(event, context):

    logger.info('Lambda execution started...')

    try:
        # Your Lambda function logic here
        logger.info('Function logic executed successfully.')
    except Exception as e:
        logger.error(f'An error occurred: {e}')

    return {
        'statusCode': 200,
        'body': 'Function executed successfully!'
    }

```

```
except Exception as e:

    # Log the error
    logger.error(f'Error: {str(e)}')

    # Return a custom error response

    return {
        'statusCode': 500,
        'body': 'Internal Server Error'
    }

finally:
    logger.info('Lambda execution completed.')
```

The code above can be of use when Lambda is acting unexpectedly. You can use detailed logging with the 'logger' module. This can be used with CloudWatch Logs to examine logs and identify the root cause of the unexpected behavior. When using detailed logging, this can provide a trail of events within the Lambda function.

2. Log Management & Normalization (Giovanni Garcia Flores, Mario Register)

2.1 Log Management Platform

AWS Cloud Watch

We decided to use AWS Cloud Watch as our log management platform.

We currently have these log groups for these services:

AWS Guard Duty

AWS Inspector

AWS Config

AWS VPC Flow Logs

AWS CloudTrail

There are also logs being collected for both our Lambda and Kinesis Data Firehose services running our log data normalization and enrichment service.

The screenshot shows the AWS CloudWatch Log groups page. The left sidebar includes sections for Favorites and recent dashboards, Alarms, Logs (with Log groups selected), Metrics, X-Ray traces, Events, Application Signals, Network monitoring, and Insights. The main content area displays a table titled 'Log groups (7)'. The table has columns for Log group, Log class, Anomaly detection, Data protection, Sensitive data, Retention, and Metric filters. Log groups listed include '/aws/events/aws_config_events_logs', '/aws/events/aws_inspector_log', '/aws/events/guarddutyfindingevent', '/aws/kinesisfirehose/PUT-S3-OuCQ', '/aws/lambda/ParseAndEnrichVPCFlowLogs', 'aws-cloudtrail-logs-637423261198-253f8f90', and 'vpc-flow-logs-cw'. A 'Create log group' button is at the top right of the table.

We used a mix of Amazon Eventbridge rules and native AWS logging services within the application to send their logs to Cloudwatch.

2.2 Log Normalization and Enrichment

We created a Kinesis Data Firehose stream that would intake logs which would then be transformed by AWS Lambda to further normalize and enrich the data into a more readable and accessible type of data.

The screenshot shows the AWS Amazon Data Firehose stream details page. The left sidebar includes sections for Firehose streams, Resources (What's new, Developer guide, API reference), and a Test with demo data section. The main content area displays 'Firehose stream details' for a stream named 'arn:aws:firehose:us-east-1:637423261198:deliverystream/PUT-S3-OuCQ'. It shows the status as Active, source as Direct PUT, destination as Amazon S3, data transformation as Enabled, and creation time as May 29, 2024 at 15:18 EDT. The 'Configuration' tab is selected, showing a 'Transform and convert records' section with options to configure AWS Lambda functions for transform and convert records. The Lambda function 'ParseAndEnrichVPCFlowLogs' is listed with version '\$LATEST'. Other tabs include Monitoring and Destination error logs.

When creating the stream, an appropriate IAM role with the proper permissions to collect the logs and to be able to use the Lambda function is needed.

An IAM role also needs to be created for the Lambda function itself to be able to intake and change the logs but to also store them in an S3 bucket.

The Lambda function itself would also need to have a custom Python code to be able to do the data enrichment and normalization itself.

The screenshot shows the AWS Lambda function editor with the following details:

- Region:** N. Virginia
- Function Name:** lambda_function
- File:** lambda_function.py
- Code Content:** The code is a Python script for processing log entries. It includes imports for base64, json, and AWS Lambda context. The `lambda_handler` function takes an event and context, extracts records from the event, and processes them through `parse_vpc_flow_log` and `map_to_ecs` functions. The results are then encoded and appended to the output list. Finally, the function returns the processed records.
- Environment:** The environment variables are listed as follows:
 - ParseAndEnrichVPC: true
- Code Properties:** The code is written in Python, and the editor supports spaces for indentation.

For demonstration, we used VPC Flow logs as an example. We created an EC2 to be able to simulate network traffic for new logs to be created and monitored.

Checking the monitor tab will tell us if the code has been executed or not.

#	Timestamp	RequestId	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsedInMB
► 1	2024-05-20T23:27:58.542Z	93F30479-c1cd8fb0824caf806c278c94aa1315	2024/05/29/{\$LATEST} cb0d8fb0824caf806c278c94aa1315	2.42	3.0	128.0	36.0
► 2	2024-05-20T23:26:02.475Z	a039e640c-1223-42ea-b125-78a44d18012f	2024/05/29/{\$LATEST} cb0d8fb0824caf806c278c94aa1315	19.79	20.0	128.0	36.0
► 3	2024-05-20T23:24:58.193Z	fc077fd0-f02e-4c79-0115-5657888ab598	2024/05/29/{\$LATEST} cb0d8fb0824caf806c278c94aa1315	2.1	3.0	128.0	36.0
► 4	2024-05-20T23:22:59.195Z	a3c39db4-a742-45f1-af136-8fc0253d0be2	2024/05/29/{\$LATEST} cb0d8fb0824caf806c278c94aa1315	9.75	10.0	128.0	36.0
► 5	2024-05-20T23:20:58.454Z	5b346c48-2021-4a86-9e47-33e0977c2522	2024/05/29/{\$LATEST} cb0d8fb0824caf806c278c94aa1315	11.34	12.0	128.0	36.0
► 6	2024-05-20T23:18:58.194Z	67130470-c157-4848-90e4-d035ca7ecf49	2024/05/29/{\$LATEST} cb0d8fb0824caf806c278c94aa1315	8.35	9.0	128.0	36.0
► 7	2024-05-20T23:16:59.453Z	82d27a05-d08d-4865-b2d4-95c5ade7084d	2024/05/29/{\$LATEST} cb0d8fb0824caf806c278c94aa1315	5.89	6.0	128.0	36.0
► 8	2024-05-20T23:14:57.712Z	8b6158b0-01df-4e45-9f57-f5ce20b8bc2f	2024/05/29/{\$LATEST} cb0d8fb0824caf806c278c94aa1315	15.52	16.0	128.0	36.0
► 9	2024-05-20T23:12:58.953Z	f67d8c7a-04b0-4f5b-99c4-b1f6b008e854	2024/05/29/{\$LATEST} cb0d8fb0824caf806c278c94aa1315	7.45	8.0	128.0	36.0

Most expensive invocations in GB-seconds (memory assigned * billed duration)							
#	Timestamp	RequestId	LogStream	BilledDurationInMS	MemorySetInMB	BilledDurationInGBSeconds	
► 1	2024-05-20T23:35:59.980Z	42ab05a4-9a76-4d0e-a5c2-4063b606b2f8	2024/05/29/{\$LATEST} 3bb339d757d4966ba4d54f8eb652f28	25.0	128	0.003125	
► 2	2024-05-20T23:14:59.961Z	469029ab-f2e4-439c-9968-28bd028a8f	2024/05/29/{\$LATEST} 3bb339d757d4966ba4d54f8eb652f28	22.0	128	0.00275	
► 3	2024-05-20T22:11:59.818Z	25488584-be09-4b99-ad23-7ddca0ad3a70	2024/05/29/{\$LATEST} 3bb339d757d4966ba4d54f8eb652f28	21.0	128	0.002625	
► 4	2024-05-20T22:29:00.197Z	f90c10ca-0c5f-4459-9e23-202d325523b3	2024/05/29/{\$LATEST} 3bb339d757d4966ba4d54f8eb652f28	21.0	128	0.002625	
► 5	2024-05-20T22:11:22:59.699Z	b608994a-bc94-4c81-aeaf-8473c6632328	2024/05/29/{\$LATEST} 3bb339d757d4966ba4d54f8eb652f28	20.0	128	0.0025	
► 6	2024-05-20T22:03:58.480Z	1ceca51b-817d-4c48-8864-d018a033b56	2024/05/29/{\$LATEST} 3bb339d757d4966ba4d54f8eb652f28	20.0	128	0.0025	
► 7	2024-05-20T22:38:09.977Z	a7f42506-8c6c-4b12-8a03-0e03aefc930	2024/05/29/{\$LATEST} 3bb339d757d4966ba4d54f8eb652f28	20.0	128	0.0025	
► 8	2024-05-20T22:04:56.958Z	1328797c-3e8b-43b7-8a22-9a3215ad699	2024/05/29/{\$LATEST} 3bb339d757d4966ba4d54f8eb652f28	20.0	128	0.0025	
► 9	2024-05-20T22:31:59.477Z	abf3c0e6-80a6-43d3-ab6b-cfa12752a2b	2024/05/29/{\$LATEST} 3bb339d757d4966ba4d54f8eb652f28	20.0	128	0.0025	

We can check on the logs and we can see there was a successful run.

The screenshot shows the AWS CloudWatch Log Events interface. The left sidebar has sections for Favorites and recent, Alarms, Logs (Log groups, Log Anomalies, Live Tail, Logs Insights), Metrics, X-Ray traces, Events (Rules, Event Buses), Application Signals, Network monitoring, and Insights. The main area is titled "Log events" and shows a list of log entries. The first entry is "INIT_START Runtime Version: python:3.12.v27 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:598ba0ea2c7b29b2fa9d54c92ef1ffdfb6c6fe22f81dcfa...". Subsequent entries show "START", "END", and "REPORT" requests for the same Lambda function, with details like RequestId, Duration, Billed Duration, Memory Size, and Max Memory Used.

We then go into the bucket where the data has been transformed and delivered to.

The screenshot shows the AWS Amazon S3 Objects interface. The left sidebar has sections for Buckets (Amazon S3, Buckets). The main area shows a bucket named "lambdaresults1234". Under the "Objects" tab, there are two objects: "2024/" and "processing-failed/". Both are listed as Folders. There are buttons for Actions (Copy S3 URI, Copy URL, Download, Open, Delete, Create folder, Upload) and a search bar for "Find objects by prefix".

We then look out for the most recent file and download it, then open it. We personally used Visual Studio Code to view the file, here is the result.

Creating a GuardDuty Threat List and Integrating with Security Hub

This guide outlines the steps to create a GuardDuty threat list, test it with EC2 instances, and integrate the findings into Security Hub for improved security visibility:

Step 1: Create a GuardDuty Threat List

1. Prepare the Threat List:

- Create a CSV file named `threatlist.csv` containing the IP addresses you want to monitor.
- Example:

```
192.0.2.0/24  
198.51.100.0/24
```

2. Upload the Threat List to an S3 Bucket:

- Upload the `threatlist.csv` file to an S3 bucket in your AWS account.

3. Enable GuardDuty and Create a Threat List:

- If not already enabled, navigate to the AWS Management Console and enable GuardDuty.
- In the GuardDuty console, under "Lists," create a new Threat List.
- Provide the S3 URI of the uploaded CSV file when creating the Threat List.

Step 2: Create and Configure EC2 Instances

1. Launch Two EC2 Instances:

- Launch two EC2 instances in your AWS account, ensuring they are in the same region where GuardDuty is enabled.

2. Simulate Malicious Activity:

- Connect to one of the EC2 instances.
- Simulate traffic to one of the IP addresses in your threat list. You can use `curl` to make an HTTP request:

- Bash

```
curl http://192.0.2.1
```

- - (Replace `192.0.2.1` with the actual IP address from your list)

Step 3: Monitor GuardDuty Findings

1. Check GuardDuty Findings:

- After simulating activity, navigate to the GuardDuty console.
- Under "Findings," check for new findings related to the simulated activity. GuardDuty should detect connections to the IP addresses in your threat list.

Step 4: Integrate GuardDuty Findings into Security Hub

1. Enable Security Hub:

- If not already enabled, navigate to the Security Hub console and enable Security Hub.

2. Enable GuardDuty Integration:

- In the Security Hub console, go to the "Integrations" section.
- Enable the GuardDuty integration if it's not already enabled.

3. View Findings in Security Hub:

- After enabling the integration, navigate to the "Findings" section in Security Hub.
- You should see GuardDuty findings listed along with other security findings.

Step 5: Automate Responses (Optional)

This step outlines an optional approach to automate responses to GuardDuty findings:

1. Set Up Automated Responses with AWS Lambda:

- You can configure automated responses using AWS Lambda and CloudWatch Events.

2. Create a CloudWatch Event Rule:

- Go to the CloudWatch console and create a new rule with the following event pattern to match GuardDuty findings:

- JSON

```
{  
  "source": ["aws.guardduty"],  
  "detail-type": ["GuardDuty Finding"]  
}
```

-
-

- Set the target of the rule to the Lambda function that you want to invoke when a GuardDuty finding triggers the event.

3. Create a Lambda Function:

- Create a Lambda function to handle GuardDuty findings. The function can take actions like stopping an instance or sending notifications based on the finding details.

```
import boto3  
  
  
def lambda_handler(event, context):  
    ec2 = boto3.client('ec2')  
  
    guardduty = boto3.client('guardduty')  
  
    finding = event['detail']  
  
    instance_id =  
    finding['resource']['instanceDetails']['instanceId']
```

```
try:

    response = ec2.stop_instances(InstanceIds=[instance_id])

    print(f"Stopped instance: {instance_id}")

except Exception as e:

    print(f"Error stopping instance: {str(e)}")
```

Important Notes:

- Ensure the Security Hub integration with GuardDuty is enabled for findings to appear.
- Consider testing the Lambda function independently before integrating it with GuardDuty findings to ensure proper functionality.
- Remember to follow AWS best practices for security and access control when setting up these services.

By following these steps, you can create a GuardDuty threat list, monitor suspicious activity, and integrate the findings with Security Hub for a centralized view of your security posture. Optionally, you can further enhance your security by automating responses using AWS Lambda and CloudWatch Events.

Technical Documentation (Focused on SIEM with Limited SOAR Integration)

SOAR Tool Selection and Integration (Future Consideration)

- **Chosen Tool (Placeholder):** While a full SOAR suite isn't implemented here, consider integrating a SOAR platform like Palo Alto Networks Cortex XSOAR or MacAfee MVISION SOAR in the future. These platforms offer comprehensive SOAR features and integrate with various AWS services.
- **Integration Process (Future Consideration):** Integration with a chosen SOAR platform would involve establishing secure communication channels between the SOAR tool and your AWS services. This might involve API integrations or dedicated connectors provided by the SOAR vendor. Security Hub could play a role here, as it aggregates security findings from AWS services and can be integrated with some SOAR platforms.
- **Challenges (Future Consideration):** Potential challenges include managing API keys and access controls for secure communication, ensuring data format compatibility between platforms, and potential limitations in the free tiers of some SOAR solutions.

Incident Response Playbooks (Limited with SIEM)

- **Overview:** While a full SOAR suite isn't implemented here, we can leverage basic automation within the SIEM solution for some incident response tasks. Playbooks might be limited in scope but can still be beneficial.
- **Incident Types:** Playbooks could address common security incidents like suspicious login attempts detected by CloudTrail, potential data exfiltration identified through VPC Flow Logs, or vulnerabilities discovered by Inspector.

- **Workflow and Automation:** Playbooks could trigger automated actions like user account lockout for suspicious logins, network traffic filtering for potential exfiltration attempts, or automated isolation of vulnerable instances.
- **Security Best Practices:** Playbooks should adhere to best practices like NIST Cybersecurity Framework or MITRE ATT&CK. Leverage these frameworks to define response procedures based on the identified incident type.

Playbook Testing and Validation

- **Testing and Validation:** Develop test scenarios simulating security incidents. These scenarios could involve injecting test data into CloudTrail logs or VPC Flow Logs to trigger playbooks and validate their functionality.
- **Security of Playbooks:** Implement access controls and permissions within the SIEM solution to restrict unauthorized modification of playbooks. Regularly review and update playbooks to ensure they remain effective and don't introduce vulnerabilities.

Threat Detection & Alert Correlation

SIEM Solution and Data Sources

- **SIEM Solution:** We'll leverage Amazon CloudWatch Logs as the central log management solution. CloudTrail, VPC Flow Logs, and Inspector findings can all be integrated with CloudWatch Logs for centralized analysis.
- **Data Sources:** Data sources include CloudTrail logs (API activity), VPC Flow Logs (network traffic), and Inspector findings (vulnerability scans).
- **Normalization and Enrichment:** CloudWatch Logs offers log parsing capabilities to normalize logs from different sources into a common format.

Additional enrichment might involve integrating threat intelligence feeds to add context to log data.

Correlation Rules and Threat Detection

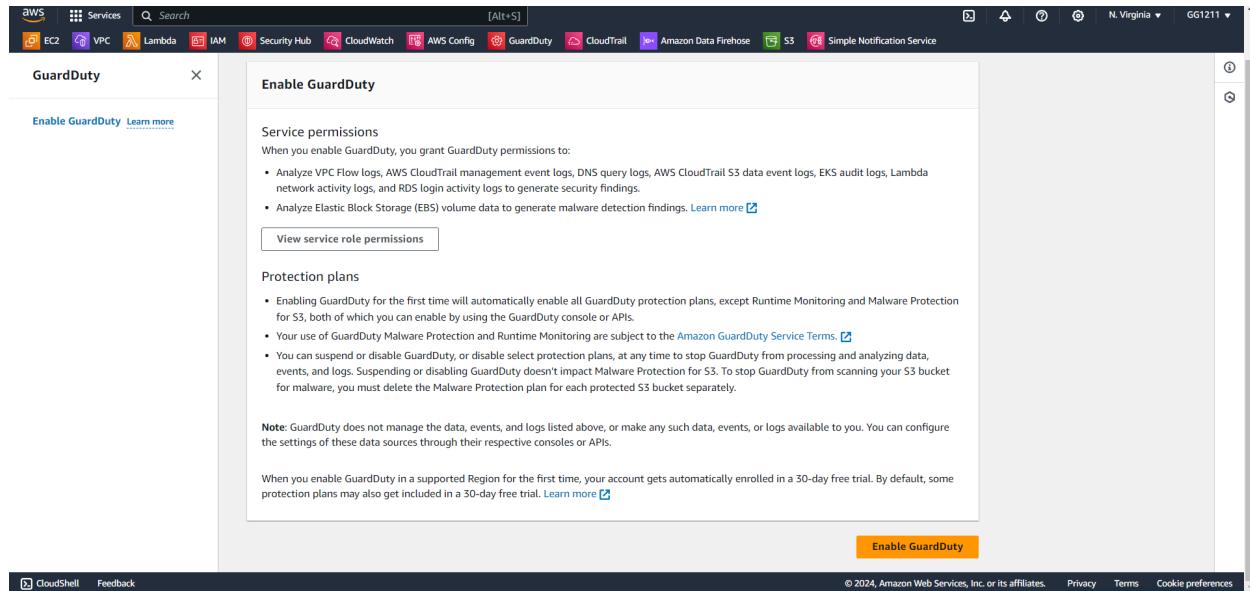
- **Correlation Rules:** Develop rules within CloudWatch Logs that analyze log data from various sources to identify potential threats. These rules might look for specific patterns in CloudTrail logs indicating suspicious activity or correlate network traffic logs with potential vulnerabilities identified by Inspector.
- **Techniques:** Techniques could include pattern matching (e.g., identifying known malicious commands in CloudTrail logs), statistical analysis (e.g., detecting unusual spikes in network traffic), or simple rule-based correlations.
- **Threat Intelligence and IoCs:** Integrate threat intelligence feeds from reputable providers to enrich correlation rules with the latest indicators of compromise (IoCs). This helps identify emerging threats and suspicious activity patterns.
- **Tuning and Refinement:** Continuously monitor and refine correlation rules to reduce false positives and improve detection accuracy. Analyze false positives to understand the root cause and adjust rules accordingly.

Alert Prioritization and Triage

- **Prioritization:** Prioritize alerts based on a combination of factors like asset criticality (e.g., high-risk servers), threat level (e.g., potential data breach), and confidence score (e.g., high confidence based on multiple indicators).
- **Integration with Playbooks:** Alerts triggered by correlation rules can be routed to basic automation within the SIEM solution or integrated with a future SOAR platform for more complex workflows.
- **High-Fidelity Alerts (Example):** A high-fidelity alert might be triggered if CloudTrail logs show a successful login attempt from an unauthorized IP address

for a critical server account. This could initiate automated playbook actions to lock the compromised account and notify security personnel for further investigation.

Enable Guard Duty:



Go and create an EC2

The screenshot shows the AWS EC2 console interface. At the top, there's a navigation bar with various services like EC2, VPC, Lambda, IAM, Security Hub, CloudWatch, AWS Config, GuardDuty, CloudTrail, Amazon Data Firehose, S3, and Simple Notification Service. The main area is titled 'Summary' and shows 'Number of instances' set to 1. Below this, the 'Software Image (AMI)' section is selected, showing 'Canonical, Ubuntu, 24.04 LTS, ...read more' with the ID 'ami-04b70fa74e45c3917'. The 'Virtual server type (instance type)' is set to 't2.micro'. Under 'Firewall (security group)', it says 'New security group'. In the 'Storage (volumes)' section, it shows '1 volume(s) - 8 GiB'. On the right, there are 'Cancel', 'Launch Instance' (which is orange), and 'Review commands' buttons. At the bottom left, there's a 'Description' section with a text input field containing the text 'Amazon Machine Image (AMI)'. The bottom of the screen has a footer with links for CloudShell, Feedback, and various AWS terms like Privacy, Terms, and Cookie preferences.

Create a new keypair, we will be using this for all our instances.

Create key pair



Key pair name

Key pairs allow you to connect to your instance securely.

KP01

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA

RSA encrypted private and public key pair

ED25519

ED25519 encrypted private and public key pair

Private key file format

.pem

For use with OpenSSH

.ppk

For use with PuTTY

 When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#) 

Cancel

Create key pair

These are the next settings

Screenshot of the AWS EC2 instance creation wizard. The instance type is selected as t2.micro. The software image (AMI) is Canonical, Ubuntu, 24.04 LTS. The virtual server type is t2.micro. A new security group is being created named "Malicious Actor SG". The security group will be added to all network interfaces. The description is also "Malicious Actor SG". The summary shows 1 instance.

These are the Security Group rules for this instance:

Screenshot of the AWS EC2 Security Groups page. The security group "sg-0cd82bb627f0e2210 - Malicious Actor SG" is selected. The details show the security group name, ID, owner, and VPC ID. The Outbound rules section shows three entries: one ICMP rule (All ICMP - IPv4, ICMP, All, 0.0.0.0/0) and two TCP rules (HTTP and HTTPS, both port 80, 0.0.0.0/0). The Outbound rules table is as follows:

Name	Security group rule...	IP version	Type	Protocol	Port range	Destination	Description
-	sgr-0cb834efabfac25	IPv4	All ICMP - IPv4	ICMP	All	0.0.0.0/0	-
-	sgr-019f75d374b04da...	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-0814a8cf04b567971	IPv4	HTTPS	TCP	443	0.0.0.0/0	-

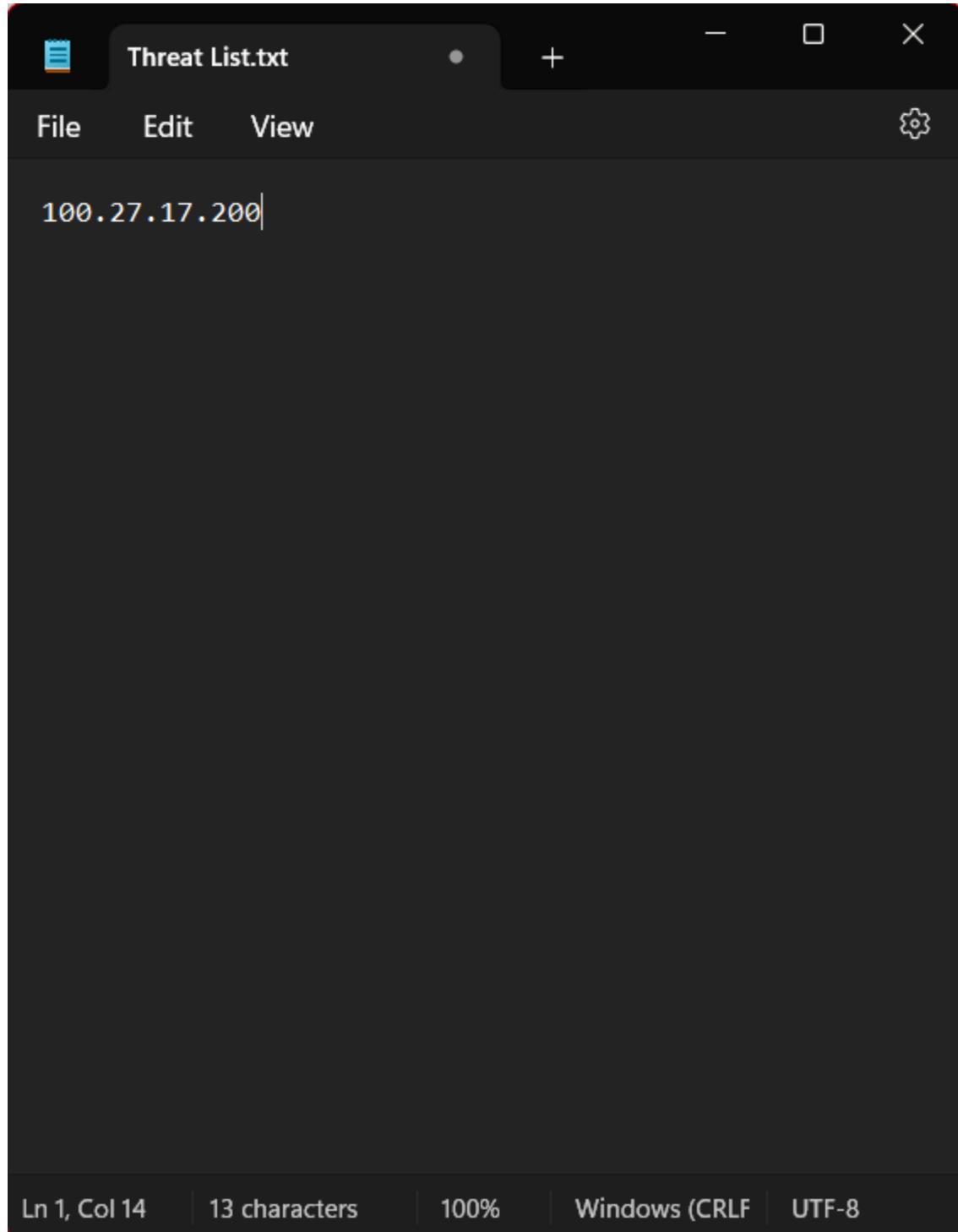
Screenshot of the AWS EC2 Security Groups page. The security group "sg-0cd82bb627f0e2210 - Malicious Actor SG" is selected. The details show the security group name, ID, owner, and VPC ID. The Inbound rules section shows four entries: one ICMP rule (All ICMP - IPv4, ICMP, All, 0.0.0.0/0) and three TCP rules (HTTP, SSH, and HTTPS, all port 80, 22, and 443 respectively, 0.0.0.0/0). The Inbound rules table is as follows:

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0498cd18122067...	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-0de15b04b65c9c1...	IPv4	All ICMP - IPv4	ICMP	All	0.0.0.0/0	-
-	sgr-0093be00c9b71fc29	IPv4	SSH	TCP	22	0.0.0.0/0	-
-	sgr-09d6a4374a664d...	IPv4	HTTPS	TCP	443	0.0.0.0/0	-

Keep a note of its public ip address

Public IPv4 address
🔗 100.27.17.200 | open address ↗

Put that address within a text editor and save it.



Now we will create an ec2 instance for our vulnerable machine. You can keep it to free tier like the last machine.

Name: Vulnerable_Machine

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Quick Start

Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux Enterprise Server

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-04b70fa74e45c3917 (64-bit (x86)) / ami-04b70fa74e45c3917 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description: Canonical, Ubuntu, 24.04 LTS, amd64 noble image build on 2024-04-23

Architecture: 64-bit (x86)

AMI ID: ami-04b70fa74e45c3917

Browse more AMIs

Storage (volumes): 1 volume(s) - 8 GiB

Cancel Launch instance Review commands

You can use the same keypair.

Key pair (login)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

KP01

Create new key pair

Here are its network settings

Network settings

VPC - required

vpc-00d4d497df62299ae4 (SOC-VPC-01)
10.0.0.0/16

Subnet

subnet-079dc1990ab406b3d Computer-02
VPC: vpc-00d4d497df62299ae4 Owner: 637423609702 Availability Zone: us-east-1a IP addresses available: 251 CIDR: 10.0.2.0/24

Create new subnet

Auto-assign public IP

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Security group name - required

Vulnerable_MachineSG

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _./@!#\$%^&*();{}~`

Description - required

Vulnerable_MachineSG

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type: ssh

Protocol: TCP

Port range: 22

Summary

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 24.04 LTS, ami-04b70fa74e45c3917

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Cancel Launch instance Review commands

Its Security Group rules

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
sgr-004e9831628923...	IPv4	HTTP	TCP	80	0.0.0.0/0	-	
sgr-0ff41e1d54a8dd475	IPv4	All ICMP - IPv4	ICMP	All	0.0.0.0/0	-	
sgr-0b9580936f938cb7c	IPv4	SSH	TCP	22	0.0.0.0/0	-	
sgr-0be5bc505fb74b5f	IPv4	HTTPS	TCP	443	0.0.0.0/0	-	

Name	Security group rule...	IP version	Type	Protocol	Port range	Destination	Description
sgr-020b5b740ee01ab...	IPv4	HTTPS	TCP	443	0.0.0.0/0	-	
sgr-0824572704d149...	IPv4	All ICMP - IPv4	ICMP	All	0.0.0.0/0	-	
sgr-09a917d829fb4fd1	IPv4	HTTP	TCP	80	0.0.0.0/0	-	

Thats about it, now lets go to making an S3 bucket to store the sample threat list we saved on the notepad.

The screenshot shows the AWS S3 'Create bucket' wizard. The 'General configuration' tab is selected, displaying fields for AWS Region (US East (N. Virginia) us-east-1), Bucket type (General purpose selected), Bucket name (Threat_List), and Object Ownership (ACLS disabled recommended selected). The 'Object Ownership' tab is also visible. The 'Default encryption' tab is selected in the second screenshot, showing Server-side encryption options (Server-side encryption with Amazon S3 managed keys (SSE-S3) selected), Bucket Key settings (Bucket Key is disabled), and Advanced settings. A note at the bottom indicates that after creating the bucket, files can be uploaded and additional settings configured.

Insert a bucket policy to allow Guard Duty to grab the information from the list. Here is a sample code.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "guardduty.amazonaws.com"
      },
      "Action": "s3:GetObject",
    }
  ]
}
```

```

    "Resource": "arn:aws:s3:::your-bucket-name/threatlist.txt"
}
]
}

```

Input that into the policy permissions.

Finally upload the text file to the s3 bucket and save it.

The screenshot shows two screenshots of the AWS S3 console side-by-side.

Left Screenshot (Upload Process):

- Header: AWS Services Search [Alt+S] N. Virginia GG1211
- Breadcrumbs: Amazon S3 > Buckets > threat-list-01 > Upload
- Section: Upload Info
- Description: Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)
- Area: Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.
- Table: Files and folders (1 Total, 0 B)

Files and folders (1 Total, 0 B)		
All files and folders in this table will be uploaded.		
<input type="text"/> Find by name		
Name	Folder	Type
Threat List.txt	-	text/plain
- Section: Destination Info
- Destination: s3://threat-list-01

Right Screenshot (Bucket Contents):

- Header: AWS Services Search [Alt+S] N. Virginia GG1211
- Breadcrumbs: Amazon S3 > Buckets > threat-list-01
- Section: threat-list-01 Info
- Tab: Objects Info Object URL Copied
- Description: Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)
- Table: Objects (1)

Objects (1) <small>Info</small>						
<input type="button"/> Copy S3 URI <input type="button"/> Copy URL <input type="button"/> Download <input type="button"/> Open <input type="button"/> Delete <input type="button"/> Actions <input type="button"/> Create folder <input type="button"/> Upload						
<input type="text"/> Find objects by prefix						
Name	Type	Last modified	Size	Storage class		
Threat List.txt	txt	June 26, 2024, 14:47:58 (UTC-04:00)	0 B	Standard		

Copy the url.

Add a threat IP list



GuardDuty generates findings based on threat IP lists. [Learn more](#)

List name

Location

For example, <https://s3.amazonaws.com/bucket-name/file.txt>

Format

Use TXT for files that contain simple IP lists.

By adding this list, you accept and agree to the GuardDuty service terms, including those related to third-party threat intelligence, and direct GuardDuty to read data from this resource.

I agree

[Cancel](#)

[Add list](#)

Go to guard duty and then to lists to add the s3 bucket to it. Make sure you add it to the threat ip. You will need the url of the bucket in order to add it to the guard duty threat list.

The screenshot shows the AWS GuardDuty List management interface. A green success message at the top says "Successfully created the list." Below it, the "List management" section displays a table for "Trusted IP lists". The table has columns for List name, List file URL, Format, Tags, and Status. It shows one entry: "No trusted IP list" with "No trusted IP list to display" and a button "Add a trusted IP list". To the right of this table is a context menu with options: Activate (which is highlighted in blue), Edit, Add tags, Delete, Actions ▾, and Add a threat IP list.

And finally just activate it.

The screenshot shows the AWS Security Hub setup page. It includes sections for "Security standards" and "AWS Integrations". In the "Security standards" section, there is a list of checkboxes for various security standards, with "Enable AWS Foundational Security Best Practices v1.0.0" checked. In the "AWS Integrations" section, there is a note about enabling Security Hub to import findings from other AWS services. At the bottom, there are "Cancel" and "Enable Security Hub" buttons.

Optionally you can activate security hub now for extra insights into the sample attack we are about to make. NOTE you will need Config set up before hand.

SOAR AUTOMATION SETUP

For the SOAR example, we will set up an event bridge rule and use lambda to create a remediation action.

Go to IAM and create a custom role for Lambda so it can have permission to change the ec2 instance state to listen to guard duty.

Select trusted entity

Trusted entity type

- AWS service
- AWS account
- Web identity
- SAML 2.0 federation
- Custom trust policy

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Lambda

Choose a use case for the specified service.
Use case
 Lambda

Make sure on the next step to add EC2 full access permission and guard duty read only access and lambda basic execution role

Step 2: Add permissions

Policy name	Type	Attached as
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonGuardDutyReadOnlyAccess	AWS managed	Permissions policy
AWSLambdaBasicExecutionRole	AWS managed	Permissions policy

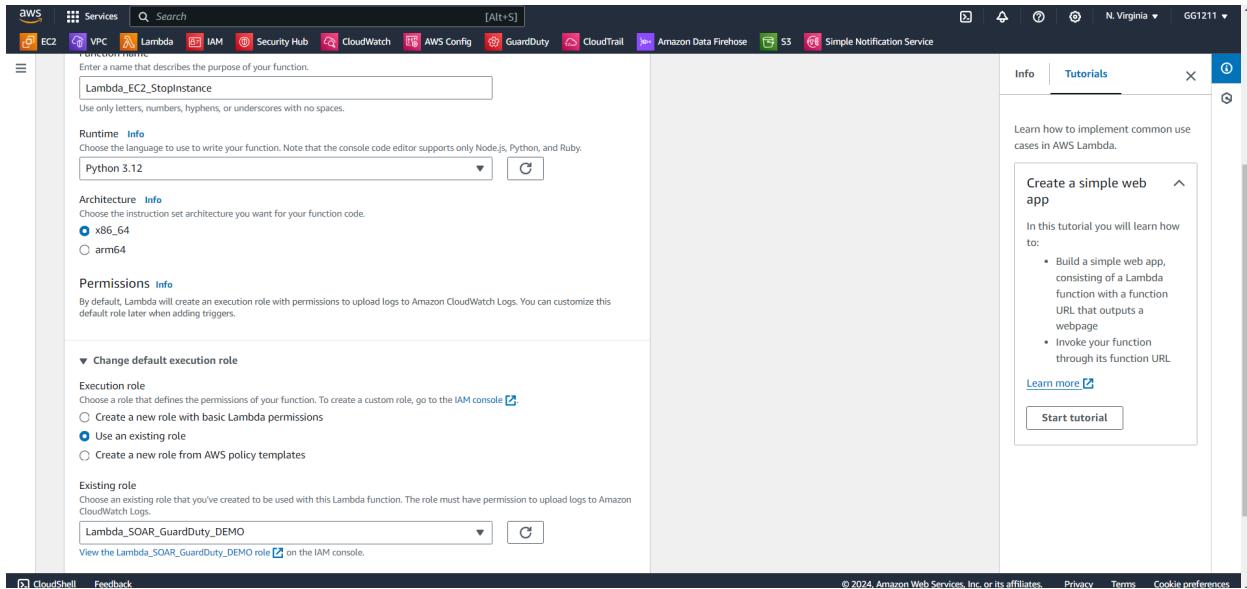
Step 3: Add tags

Add tags - optional

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add new tag



We will then create a new lambda function using these settings and make sure to select the IAM role we made and select it.

Once created input this code

```
import boto3

def lambda_handler(event, context):
    ec2 = boto3.client('ec2')
    guardduty = boto3.client('guardduty')

    finding = event['detail']
    instance_id = finding['resource']['instanceDetails']['instanceId']

    try:
        response = ec2.stop_instances(InstanceIds=[instance_id])
        print(f"Stopped instance: {instance_id}")
    except Exception as e:
        print(f"Error stopping instance: {str(e)}")
```

The screenshot shows the AWS Lambda console. A success message at the top states: "Successfully created the function Lambda_EC2_StopInstance. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the "Code" tab is selected, showing the code source in a code editor. The code is as follows:

```

1 import boto3
2
3 def lambda_handler(event, context):
4     ec2 = boto3.client('ec2')
5     guardduty = boto3.client('guardduty')
6
7     finding = event['detail']
8     instance_id = finding['resource']['instanceDetails']['instanceId']
9
10    try:
11        response = ec2.stop_instances(InstanceIds=[instance_id])
12        print(f"Stopped instance: {instance_id}")
13    except Exception as e:
14        print(f"Error stopping instance: {str(e)}")
15

```

The "Code properties" section shows the package size is 200 bytes, the SHA256 hash is `SHA256 hash: #L_HApeEDF...`, and it was last modified on June 26, 2024, at 07:35 PM EDT.

A sidebar on the right titled "Create a simple web app" provides a tutorial on how to implement common use cases in AWS Lambda, including building a simple web app with a Lambda function and a function URL.

Click deploy and you should be good.

The screenshot shows the Amazon EventBridge Rules page. The left sidebar includes sections for Developer resources (Learn, Sandbox, Quick starts), Buses (Event buses, Rules, Global endpoints, Archives, Replays), Pipes (Pipes), Scheduler (Schedules, Schedule groups), Integration (Partner event sources, API destinations), and CloudShell/Feedback.

The main content area displays the "Rules" section, which is currently empty. It includes a "Select event bus" dropdown set to "default" and a "Rules" table with a single entry: "No rules". A "Create rule" button is located at the bottom of the rules table.

Next go to cloud watch and then rules to access eventbridge.

The screenshot shows the AWS EventBridge console interface. At the top, there's a navigation bar with various AWS services like EC2, VPC, Lambda, IAM, Security Hub, CloudWatch, AWS Config, GuardDuty, CloudTrail, Amazon Data Firehose, S3, and Simple Notification Service. Below the navigation bar, there's a section for creating an event pattern. It includes three options: 'Use schema' (using an Amazon EventBridge schema to generate a pattern), 'Use pattern form' (using a template provided by EventBridge to create an event pattern, which is currently selected), and 'Custom pattern (JSON editor)' (writing an event pattern in JSON). The 'Event pattern form' section contains fields for 'Event source' (set to 'AWS services' and 'GuardDuty'), 'Event type' (set to 'GuardDuty Finding'), and an 'Event pattern' JSON editor containing the following code:

```

1 {
2   "source": ["aws.guardduty"],
3   "detail-type": ["GuardDuty Finding"]
4 }

```

Below the JSON editor are buttons for 'Copy', 'Test pattern', and 'Edit pattern'. At the bottom of the page, there are 'Cancel', 'Previous', and 'Next' buttons.

The screenshot shows the continuation of the event rule creation process. On the left, a sidebar lists steps: Step 2 'Build event pattern' (completed), Step 3 'Select target(s)' (active), Step 4 - optional 'Configure tags' (not completed), and Step 5 'Review and create' (not completed). The main area is titled 'Target 1'. It shows the target type selected as 'AWS service' and the target configuration for a 'Lambda function' named 'Lambda_EC2_Stopinstance'. There are also sections for 'Configure version/alias' and 'Additional settings'. At the bottom, there are buttons for 'Add another target', 'Cancel', 'Skip to Review and create', 'Previous', and 'Next' (which is highlighted).

Direct the rule to the lambda function we created and finally create the rule.

DEMO

SSH into both your vulnerable and malicious machine. ping or curl the vulnerable machine from the malicious actor machine and vice versa. It will take a minute for Guard Duty to capture the event and for the lambda function to start executing.

```

ubuntu@ip-10-0-1-133:~$ ping 44.211.53.186
PING 44.211.53.186 (44.211.53.186) 56(84) bytes of data.
64 bytes from 44.211.53.186: icmp_seq=1 ttl=63 time=0.606 ms
64 bytes from 44.211.53.186: icmp_seq=2 ttl=63 time=0.684 ms
64 bytes from 44.211.53.186: icmp_seq=3 ttl=63 time=0.655 ms
64 bytes from 44.211.53.186: icmp_seq=4 ttl=63 time=0.648 ms
64 bytes from 44.211.53.186: icmp_seq=5 ttl=63 time=0.619 ms
...
C
--- 44.211.53.186 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4130ms
rtt min/avg/max/mdev = 0.606/0.642/0.684/0.027 ms
ubuntu@ip-10-0-1-133:~$ 

C
--- 44.197.245.31 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.579/0.623/0.681/0.036 ms
root@ip-10-0-1-133:/home/ubuntu# sudo nmap 44.197.245.31
Starting Nmap 7.94 ( https://nmap.org ) at 2024-06-26 20:41 UTC
Nmap can report for ec2-44-197-245-31.compute-1.amazonaws.com (44.197.245.31)
Host is up (0.00078s latency).
TCP filtered ports: 998
No service shown
PORT      STATE    SERVICE
80/tcp    closed   http
443/tcp   closed   https

```

Portscan attack and ping command from malicious machine to vulnerable machine.

The screenshot shows the AWS GuardDuty interface. On the left, there's a sidebar with navigation links like Summary, Findings, Usage, and Protection plans. The main area displays a 'Findings' table with three entries. One entry is highlighted, showing details about an EC2 instance performing a port scan. The 'Overview' section provides a summary of the finding, including the account ID, resource ID, creation date, and update date.

Severity	Finding type
Low	Policy:IAMUser/RootCredentialUsage
Medium	UnauthorizedAccess:EC2/MaliciousPCaller.Custom
Medium	Recon:EC2/Portscan

Recon:EC2/Portscan

Finding ID: ccc82b246573d7e191778c653518d7ec

Medium The EC2 instance i-05195f8f2295be27e is performing outbound port scans against remote host 44.197.245.31.

Investigate with Detective

This finding is Useful Not useful

Overview

Severity	MEDIUM
Region	us-east-1
Count	1
Account ID	637423609702
Resource ID	i-05195f8f2295be27e
Created at	06-26-2024 16:42:09 (3 minutes ago)
Updated at	06-26-2024 16:42:09 (3 minutes ago)

Malware scan

The screenshot shows the AWS GuardDuty console. On the left, the navigation pane includes links for Summary, Findings (which is selected), Usage, EC2 malware scans, Protection plans (S3, EKS, Runtime Monitoring, Malware Protection for EC2, Malware Protection for S3, RDS, Lambda), Accounts, Settings, Lists, What's New, Partners, and CloudShell.

The main content area is titled "Findings Info". It shows a table with one row:

Severity	Finding type
<input checked="" type="checkbox"/> Medium	UnauthorizedAccess:EC2/MaliciousIPCaller.Custom
<input type="checkbox"/> Low	PolicyIAMUser/RootCredentialUsage

To the right, a detailed view of the first finding is shown:

UnauthorizedAccess:EC2/MaliciousIPCaller.Custom
 Finding ID: [aa82b14683f4abd4d5d65202f9be8b8](#)

Medium The EC2 instance i-0cdb8204df2e2bb0a is communicating with the IP address 3.237.200.121 on the custom threat list Threat_List_DEMO.info

Investigate with Detective

This finding is [Useful](#) [Not useful](#)

Overview

Severity	MEDIUM
Region	us-east-1
Count	1
Account ID	637423609702
Resource ID	i-0cdb8204df2e2bb0a
Created at	06-26-2024 16:07:13 (a few seconds ago)
Updated at	06-26-2024 16:07:13 (a few seconds ago)

© 2024 Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The threat was detected on guard duty.

The screenshot shows the AWS CloudWatch Logs console. The left sidebar includes sections for Favorites and recent dashboards, Alarms (0 in alarm), Logs (Log groups, Log Anomalies, Live Tail, Logs Insights, Contributor Insights), Metrics, X-Ray traces, Events (Rules, Event Buses), and Application Signals.

The main area shows log events for the log group /aws/lambda/Lambda_EC2_StopInstance. The table has columns for Timestamp and Message.

Timestamp	Message
2024-06-26T20:10:02.584Z	INIT_START Runtime Version: python:3.12.v28 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:7776dee4c90fff4d4bf833b78d94e5b4148d14d044b867a1...
2024-06-26T20:10:02.877Z	START RequestId: c595612a-f327-42e1-a781-b5eacfbaabf03 Version: \$LATEST
2024-06-26T20:10:05.944Z	2024-06-26T20:10:05.944Z c595612a-f327-42e1-a781-b5eacfbaabf03 Task timed out after 3.07 seconds
2024-06-26T20:10:05.944Z	END RequestId: c595612a-f327-42e1-a781-b5eacfbaabf03
2024-06-26T20:10:05.944Z	REPORT RequestId: c595612a-f327-42e1-a781-b5eacfbaabf03 Duration: 3067.26 ms Billed Duration: 3000 ms Memory Size: 128 MB Max Memory Used: 89 MB Init...
2024-06-26T20:10:05.984Z	INIT_START Runtime Version: python:3.12.v28 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:7776dee4c90fff4d4bf833b78d94e5b4148d14d044b867a1...
2024-06-26T20:11:11.127Z	START RequestId: c595612a-f327-42e1-a781-b5eacfbaabf03 Version: \$LATEST
2024-06-26T20:11:14.163Z	2024-06-26T20:11:14.163Z c595612a-f327-42e1-a781-b5eacfbaabf03 Task timed out after 3.04 seconds
2024-06-26T20:11:14.163Z	END RequestId: c595612a-f327-42e1-a781-b5eacfbaabf03
2024-06-26T20:11:14.163Z	REPORT RequestId: c595612a-f327-42e1-a781-b5eacfbaabf03 Duration: 3035.44 ms Billed Duration: 3000 ms Memory Size: 128 MB Max Memory Used: 43 MB
2024-06-26T20:11:14.189Z	INIT_START Runtime Version: python:3.12.v28 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:7776dee4c90fff4d4bf833b78d94e5b4148d14d044b867a1...

The cloud watch log group for the lambda function shows a successful execution

Looking back at the vulnerable machine, the instance has stopped.

Monitoring Capabilities

1. CloudWatch Logs:

- **Usage:** Capture and store log files from Amazon EC2 instances, AWS Lambda functions, and other services.
- **Integration:** Easily integrate with other AWS services such as CloudTrail, VPC Flow Logs, and AWS WAF for comprehensive log analysis.

2. CloudWatch Metrics:

- **Usage:** Monitor metrics related to AWS resources and applications.
- **Integration:** Automatically collects metrics from AWS services like EC2, S3, DynamoDB, Lambda, etc.

3. Dashboards and Alarms:

- **Dashboards:** Create customizable dashboards to visualize operational data and security metrics in real-time.
- **Alarms:** Set up alarms based on thresholds for metric values, allowing proactive monitoring and alerting.

Implementing Security Monitoring with CloudWatch

1. Enable CloudTrail Integration:

- Use CloudWatch Logs to collect and analyze AWS CloudTrail logs. This helps you track API calls and resource changes across your AWS environment.

2. VPC Flow Logs:

- Store VPC Flow Logs in CloudWatch Logs for detailed analysis of network traffic patterns and potential security incidents.
- 3. **Custom Metrics and Logs:**
 - Create custom metrics and logs to monitor specific security-related events or application performance metrics.

Advantages of Using CloudWatch Exclusively

1. **Centralized Monitoring:**
 - CloudWatch provides a single interface to monitor logs and metrics across your AWS resources, simplifying management and troubleshooting.
2. **Integration with AWS Services:**
 - Seamless integration with various AWS services ensures comprehensive visibility into your cloud infrastructure.
3. **Scalability and Reliability:**
 - Built to handle large-scale environments with high reliability and scalability, ensuring that you can monitor and manage even complex architectures effectively.
4. **Cost Efficiency:**
 - CloudWatch offers flexible pricing based on the volume of logs ingested and metrics monitored. Utilizing CloudWatch exclusively can potentially simplify cost management compared to using multiple monitoring solutions.

Considerations

1. **Data Retention and Management:**
 - Plan for data retention policies in CloudWatch to ensure compliance and manage storage costs effectively.
2. **Complex Use Cases:**
 - For advanced analytics or specific use cases requiring machine learning insights or deeper analysis, consider additional AWS services or third-party tools that complement CloudWatch.

Conclusion

Using Amazon CloudWatch exclusively for your AWS Security Operations Center can be a cost-effective and efficient solution, especially if you prefer a unified platform for monitoring and analyzing logs and metrics across your AWS environment. Evaluate your specific requirements, scalability needs, and integration preferences to determine if CloudWatch meets all your monitoring and security operation needs or if additional tools or services are necessary.

References

- **Optimizing your SOC's threat coverage and data value:** <https://m.youtube.com/watch?v=YD6QCEVDw20> (This video by Securonix discusses optimizing data ingestion for threat coverage and highlights the importance of accurate data intake for effective SOC operations.)
- **Live Demo: Get Faster Time-to-Value Across Your SOC Workflow with Hunters SOC Platform:** <https://m.youtube.com/watch?v=YD6QCEVDw20> (This Hunters SOC Platform demo showcases how their platform connects to various data sources and emphasizes the importance of data readiness for analysis.)
- **Upgrade your SOC from log collection to true threat detection & response:** <https://m.youtube.com/watch?v=YD6QCEVDw20> (This Microsoft and CyberProof discussion focuses on transitioning from basic log collection to true threat detection and response. While not directly addressing ingestion requirements, it highlights the importance of comprehensive data collection for effective SOC functionality.)
- **AWS re:Inforce 2022 - Using AWS security services to build your cloud security operations baseline -**
 ▶ AWS re:Inforce 2022 - Using AWS security services to build your cloud sec...
(An AWS conference that explains GuardDuty and Security Hub in detail as it relates to building a cloud security operation).
- **SOC GUIDE** <https://github.com/cyb3rxp/awesome-soc/blob/main/README.md>
 - AWS Security Best Practices and Beginner Tools
 ▶ AWS Summit DC 2021: Security best practices, common mistakes, and solutions
 - VPC Flow Log Tutorial
<https://docs.aws.amazon.com/vpc/latest/tgw/flow-logs-cwl.html>
 - GaurdDuty Configuration Stuff
https://repost.aws/questions/QUBu_mmiH7QWKGq-MmZ_p2bQ/best-method-to-send-guardduty-logs-to-opensearch
 - Guard duty events to cloud watch
<https://docs.logz.io/docs/shipping/aws/guardduty/>

- Explanation of why Sec Hub is not a siem solution by itself

➡️ Getting Hands on with Amazon GuardDuty - AWS Virtual Workshop

- Guard Duty automated response architecture

➡️ Amazon GuardDuty Deep Dive

- Exporting cloudwatch logs to s3

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/S3ExportTasksConsole.html#ExportSingleAccount>