A

Project Report On

"Zero Trust Security and Real-Time Threat Detection in AWS Cloud"

Prepared by

Dahiya Mandeepsinh (D24AIML081)

Under the guidance of

Mr. Deep Mendha

A Report Submitted to
Charotar University of Science and Technology for

Partial Fulfillment of the Curriculum Requirements of

Bachelor of Technology

in Artificial Intelligence and Machine Learning

(5th Semester Software Group Project-II –AIML305)

Submitted at

Department of Artificial Intelligence and Machine Learning

Chandubhai S. Patel Institute of Technology

At: Changa, Dist.: Anand – 388421

November 2025

CANDIDATE'S DECLARATION

I hereby declare that the project entitled **" Zero Trust Security and Real - Time Threat Detection in AWS Cloud "** is my own work conducted under the guidance of **Mr. Deep Mendha** .

I further declare that to the best of my knowledge, the project for B. Tech does not contain any part of the work, which has been submitted for the award of any degree either in this University or in other University without proper citation.

Dahiya Mandeepsinh
(D24AIML081)

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

This is to certify that the report entitled "**Zero Trust Security and Real-Time Threat Detection in AWS Cloud**" is a bonafied work carried out by **Dahiya Mandeepsinh (D24AIML081)** under the guidance and supervision of **Prof. Deep Mendha** for the subject **Software Group Project -II (AIML305)** of 5th Semester of Bachelor of Technology in **Artificial Intelligence and Machine Learning** at Faculty of Technology & Engineering – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate himself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Under supervision of,

Mr. Deep Mendha
Designation
Department of Artificial Intelligence and
Machine Learning
CSPIT, Changa, Gujarat.

Dr. Nirav Bhatt
Head of Department
Department of Artificial Intelligence and
Machine Learning
CSPIT, Changa, Gujarat.

Chandubhai S Patel Institute of Technology

At: Changa, Ta. Petlad, Dist. Anand, PIN: 388 421. Gujarat

Author Guidelines for the Preparation of Contributions to Springer Computer Science Proceedings

Dahiya Mandeepsinh

d24aiml081@charusat.edu.in

Abstract. This project implements a zero-trust inspired monitoring and detection stack using native AWS threat detection (Amazon GuardDuty) together with a host-level / network IDS (Suricata) deployed on an EC2 instance. The goal is to detect reconnaissance activity, suspicious outbound portscans, and improper credential usage while demonstrating how host-based and cloud-native detections complement each other. The implementation includes an Apache web server running on a monitored EC2 instance, Suricata for live packet inspection, GuardDuty findings analysis, and recommendations for automated remediation (Security Groups, IAM best practices, Lambda responses, and centralized logging).

Keywords: Zero Trust, GuardDuty, Suricata, EC2, IDS/IPS, portscan, cloud security, AWS, incident detection, VPC flow logs

1. INTRODUCTION

The rapid growth of cloud computing has fundamentally transformed how organizations deploy and manage their applications and workloads. While the cloud offers unmatched flexibility, scalability, and accessibility, it also introduces new categories of security risks. Unlike traditional on-premises systems, cloud resources are often publicly reachable, shared across multiple tenants, and dynamic in nature. This diverse environment requires security mechanisms that are not only strong but also continuously adaptive.

To address these challenges, the Zero-Trust Security Model has emerged as an industry standard. Instead of assuming that internal systems are trustworthy, Zero Trust operates on the belief that no device, user, or network segment should be trusted by default. Verification and monitoring must occur at every stage of communication. In other words, “Never trust—always verify.”

In line with this philosophy, the present project aims to implement a small-scale Zero-Trustinspired monitoring environment using a combination of:

- AWS GuardDuty, a cloud-native threat detection service
- Suricata, an open-source network intrusion detection system (IDS)
- Apache Web Server deployed on an EC2 instance

The goal of this project is not only to build and evaluate a threat detection pipeline but also to explore how cloud-native and host-level security tools complement each other. The project simulates real-world attack scenarios—such as port scanning and improper credential use—and analyzes how these incidents are detected, interpreted, and mitigated.

This report documents the entire process, starting from the problem identification, system design, implementation, findings, analysis, and eventual recommendations for strengthening the overall environment. The intention is to demonstrate how cloud resources can be secured effectively using layered detection mechanisms aligned with Zero-Trust principles.

2. PROBLEM STATEMENT AND OBJECTIVES

2.1 Problem Statement

Although cloud infrastructure has become the backbone of many organizations, it is also a frequent target for attackers who employ automated tools to discover vulnerable hosts. Instances that are exposed to the Internet are particularly at risk and commonly face threats such as:

- Port scans
- Credential misuse
- Unauthorized outbound communication
- Malware infiltration
- Misconfigured identity or network settings

Traditional perimeter-based security approaches are no longer sufficient in these dynamic, distributed environments. Attackers routinely scan cloud address ranges for exposed services and misconfigurations. At the same time, careless handling of highly privileged credentials (for example, root account keys or long-lived IAM access keys) increases the risk of privilege escalation and account takeover.

Hence, the central question addressed by this work is:

How can we monitor, detect, and respond to suspicious activities within a cloud environment using a Zero-Trust aligned approach that combines cloud-native telemetry with host-level intrusion detection?

To explore this, the project deploys an Apache web server on an AWS EC2 instance, installs and configures Suricata as a host-level IDS, and enables AWS GuardDuty for cloud-level monitoring. Findings from these tools are collected and analyzed, and mapped to concrete mitigation measures.

2.2 Objectives of the Project

To address the problem described above, the project establishes the following objectives:

1. Deploy a vulnerable but monitored workload.
   Launch an EC2 instance running an Apache web server to simulate a public-facing service that could be targeted by attackers.
2. Enable cloud-native threat detection. Activate AWS GuardDuty to monitor:
   - API activity via CloudTrail,
   - Network behavior via VPC Flow Logs, and
   - DNS request patterns.
3. Install and configure Suricata.
   Configure Suricata on the EC2 host to provide packet inspection, rule-based detection, and EVE-JSON logging for suspicious traffic.
4. Generate and capture realistic threats.
   Simulate incidents such as port scans and the use of root credentials so that GuardDuty and Suricata generate representative findings.
5. Analyze and interpret detection results.
   Evaluate alerts from GuardDuty and Suricata to determine the trigger, the rationale behind the alert, and the associated risks.
6. Recommend remediation and hardening.

Produce a prioritized action plan covering immediate, short-term, and long-term remediation, including identity controls, network segmentation, and automation.

7. Document thoroughly.
   Produce a detailed technical report that mirrors professional cybersecurity documentation and captures implementation, findings, and recommendations.

The final deliverable is a full security evaluation and a Zero-Trust detection environment implemented using AWS Free Tier resources and open-source tooling.

3. LITERATURE REVIEW / RELATED WORK

Modern cybersecurity frameworks emphasize continuous monitoring, behavioral analytics, and intelligent detection techniques. In cloud environments, conventional security tools alone are often insufficient because cloud infrastructure functions differently from traditional on-premise networks. Consequently, researchers and industry practitioners increasingly recommend hybrid monitoring architectures that combine both host-level and cloud-level visibility to achieve stronger security coverage.

3.1 Zero-Trust Security Model

The Zero-Trust Security Model, originally popularized by Forrester Research and later standardized through NIST guidelines, shifts away from the long-standing assumption that internal networks can be trusted by default. Instead, Zero Trust promotes a defensive philosophy built on the following core principles:

- Never trust internal networks blindly
- Validate every connection request
- Enforce least-privilege access
- Continuously monitor user and device behavior

These principles align particularly well with cloud environments, where traditional perimeterbased security is insufficient due to dynamic scaling, distributed workloads, and internet exposure. Research consistently shows that Zero-Trust frameworks reduce the impact of credential misuse, lateral movement, and misconfigured access permissions.

3.2 Cloud-Native Threat Detection (AWS GuardDuty)

AWS GuardDuty is an intelligent, cloud-native threat detection service that incorporates machine learning, anomaly detection algorithms, and external threat intelligence feeds to identify suspicious activities. GuardDuty autonomously analyzes key AWS telemetry sources, including:

- CloudTrail Logs – API activity and identity-related events
- VPC Flow Logs – network traffic metadata
- DNS Query Logs – domain lookups and potential command-and-control behavior

Common threats detected by GuardDuty include port scanning, brute-force login attempts, compromised IAM credentials, anomalous outbound communication, and malware command-andcontrol traffic. Studies highlight that cloud-native detection tools like GuardDuty are essential for identifying attack patterns that host-based IDS tools may overlook, particularly those involving AWS service misuse, credential compromise, or API-level anomalies.

3.3 Host-Based Intrusion Detection (Suricata)

Suricata is a widely adopted open-source intrusion detection and prevention engine known for its deep packet inspection capabilities, multi-threaded performance, and automatic protocol detection. It supports flexible logging formats such as EVE-JSON and relies on communitymaintained rule sets (e.g., Emerging Threats) to detect known attack signatures. Suricata specializes in monitoring:

- Port scans
- Exploit attempts
- Suspicious packet sequences
- Malware-related network patterns
- Protocol and traffic anomalies

When deployed alongside cloud-native tools, Suricata provides packet-level visibility that complements GuardDuty's higher-level behavioral and API-based analytics. This combination creates a multi-layered detection strategy consistent with Zero-Trust architecture recommendations found in both academic and industry literature.

3.4 Gaps Identified in Existing Approaches

Despite their strengths, neither GuardDuty nor Suricata is sufficient on its own. GuardDuty provides excellent visibility into AWS account activity and network trends, but it cannot inspect packet payloads. Conversely, Suricata inspects packets in detail but does not have visibility into AWS API usage, IAM behavior, or service-level anomalies.

Therefore, combining cloud-native detection (GuardDuty) with host-level packet inspection (Suricata) results in a much more comprehensive security posture. This hybrid model bridges critical visibility gaps and aligns closely with modern Zero-Trust principles that demand verification at every layer of the infrastructure.

4. SYSTEM DESIGN AND ARCHITECTURE

Designing a secure cloud environment requires a strategic blend of cloud-native services and host-level security controls. The architecture implemented in this project follows the core philosophy of Zero Trust — no component is inherently trusted, and every activity must be continuously validated.

To achieve this, the entire system is designed around three major pillars:

- A monitored workload (Apache on EC2)
- Cloud-level threat detection (AWS GuardDuty)
- Host-level intrusion detection (Suricata IDS)

Each of these components plays a unique role in detecting different categories of threats.

4.1 Overview of the Architecture

The overall system consists of the following major components:

1. AWS EC2 Instance
Runs Ubuntu Server (free-tier eligible) and hosts an Apache web server. Suricata is installed on this instance to inspect inbound and outbound network packets.

2. AWS GuardDuty
A fully managed threat detection service that analyzes CloudTrail logs, VPC Flow Logs, and DNS logs for suspicious patterns.

3. Security Group Configuration
The EC2 instance is placed behind a security group allowing only SSH and HTTP access.

4. Suricata IDS
Suricata performs packet inspection, detects port scans and exploit attempts, and logs events in JSON format.

This architecture collectively supports a layered Zero-Trust detection model.

4.2 Architecture Diagram (Detailed Description)

Although the diagram is not physically included here, the architecture can be described as the following flow:

This representation highlights how user traffic flows through the system, how Suricata inspects packets, and how GuardDuty analyzes cloud-level activity.

4.3 Why Two IDS Systems? (GuardDuty + Suricata)

Running both GuardDuty and Suricata may seem redundant, but they serve fundamentally different purposes:

- GuardDuty detects cloud-level anomalies such as risky API usage, suspicious DNS activity, and unusual network behavior.
- Suricata inspects raw packet payloads and detects signature-based intrusions and protocol anomalies.

Using both systems provides multi-layered detection similar to enterprise SOC environments.

4.4 Data Flow & Monitoring Approach

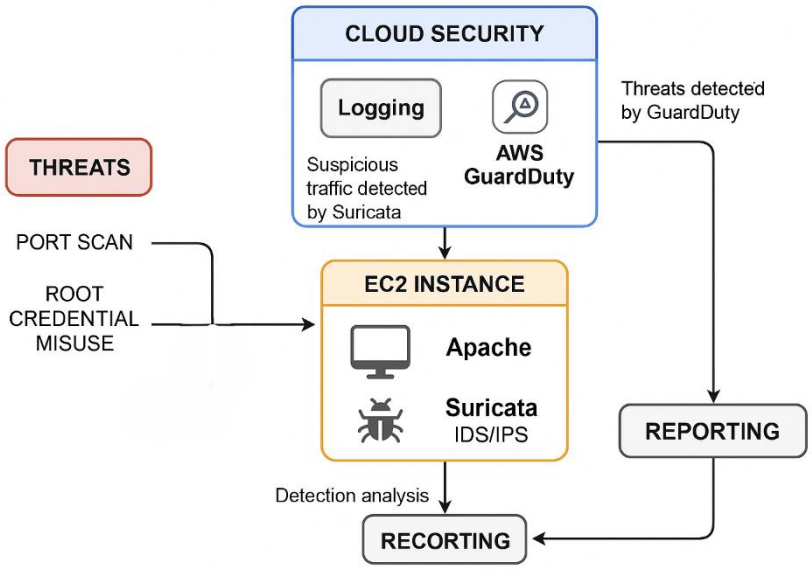
The data flow across the system follows these stages:

1. User connects to EC2 instance
Apache serves content while Suricata simultaneously inspects the incoming packets.

2. EC2 instance communication
Any outbound or inbound activity is captured in VPC Flow Logs, which GuardDuty analyzes for anomalies.

3. Suspicious behavior detected
GuardDuty or Suricata generates alerts based on threat indicators.

This combined setup mimics a realistic cloud security monitoring pipeline aligned with ZeroTrust principles.



5. IMPLEMENTATION

This section documents each step taken to build, configure, and validate the Zero-Trust monitoring environment. Screenshots provided in the dataset correlate directly with these steps.

5.1 Environment Setup

Step 1: EC2 Instance Setup and Apache Installation

A free-tier eligible Ubuntu EC2 instance was launched from the AWS Management Console. A public IP address was assigned, and a Security Group was configured to allow SSH (port 22) and HTTP (port 80). The instance was accessed via SSH, and Apache was installed using:

sudo apt update sudo apt install apache2 -y

The Apache service was then enabled and started:

sudo systemctl enable apache2

sudo systemctl start apache2

Running systemctl status apache2 confirmed that the service was active. Accessing the public IP in a browser displayed the Apache default page, indicating a successful setup.

5.2 Suricata Installation & Configuration

Suricata was installed as the host-level IDS on the EC2 instance using:

sudo apt install suricata –y

The IDS service was enabled and started:

sudo systemctl enable suricata

sudo systemctl start suricata

To verify that Suricata was running, the following command was used:

sudo systemctl status suricata

A runtime test was performed using verbose mode to observe initialization and traffic monitoring:

sudo suricata -i eth0 –v

The output confirmed that Suricata detected the network interface and initialized capture threads. However, the message:

SC_ERR_NO_RULES - No rule files loaded.

indicated that Suricata did not yet have rule sets installed, meaning signature-based attack detection was not active at this stage.

5.3 AWS GuardDuty Activation & Findings

AWS GuardDuty was enabled to provide cloud-level threat detection. Once activated, GuardDuty began analyzing CloudTrail logs, VPC Flow Logs, and DNS query logs.

After initial deployment and simulated activity, GuardDuty produced findings that included:

- Recon:EC2/Portscan (Medium Severity) – indicating outbound port scanning from the EC2 instance.
- Policy:IAMUser/RootCredentialUsage (Low Severity) – indicating that an API call was made using root credentials.

These findings matched the simulated threats and confirmed that GuardDuty was functioning correctly. Screenshots such as GuardDuty Findings.png, Findings Screenshot.png, and pie.png visually represent the detection results.

5.4 Web Service Proof (Apache)

To verify the environment, the public IP address of the EC2 instance (e.g., 98.81.232.134) was accessed through a browser. The default Apache page was displayed, confirming:

- The Apache server was installed correctly
- HTTP traffic was successfully reaching the instance
- Security Group rules were configured properly
- The workload was reachable for simulated threat activity

This step validated that the monitored workload was functioning as intended and was capable of receiving traffic for analysis by both Suricata and GuardDuty.

6. FINDINGS ANALYSIS

Once GuardDuty and Suricata were deployed and the simulated suspicious activities were carried out, both systems started generating findings. This section analyzes those findings in depth, explaining why they occurred, what they signify, and what risks they introduce.

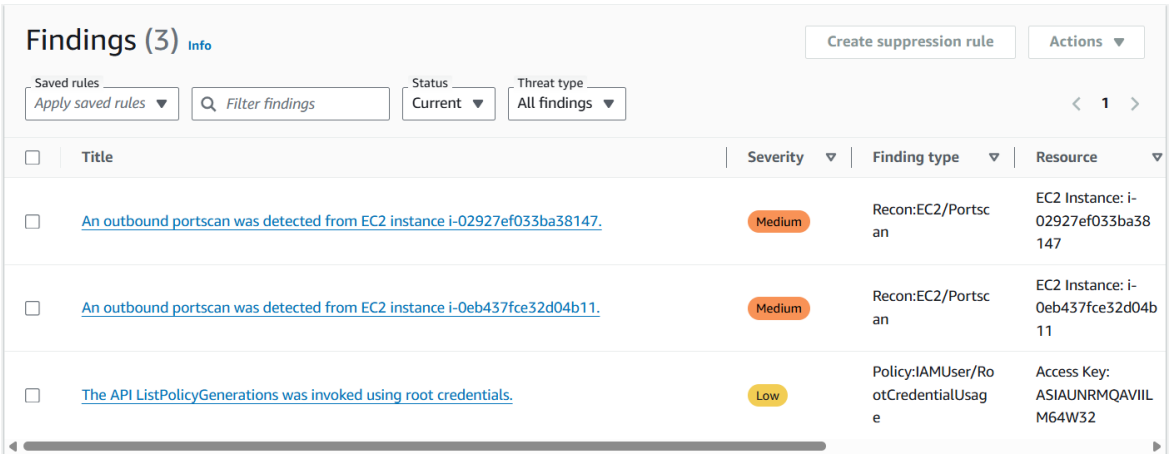
6.1 Analysis of GuardDuty Findings

AWS GuardDuty produced two major categories of findings during the project:

6.1.1 Recon:EC2/Portscan (Medium Severity)

This type of alert is triggered when GuardDuty detects that an EC2 instance is scanning external IP addresses or services. Port scanning is often the first step attackers use to discover running services, open ports, vulnerable applications, or misconfigured firewalls.

In this project, the portscan alert was generated intentionally to test GuardDuty's monitoring capabilities.



6.1.2 Policy:IAMUser/RootCredentialUsage (Low Severity But Critical)

This alert indicates that the AWS root user (or a root user access key) was used to call an API — in this case, "ListPolicyGenerations." Even though GuardDuty marks this as low severity, it represents one of the most critical security misconfigurations from a cloud-security standpoint.

Root credentials have unrestricted permissions and can:

- Delete the entire AWS account
- Remove security controls
- Disable logging
- Create new IAM users with admin rights

6.2 Analysis of Suricata Observations

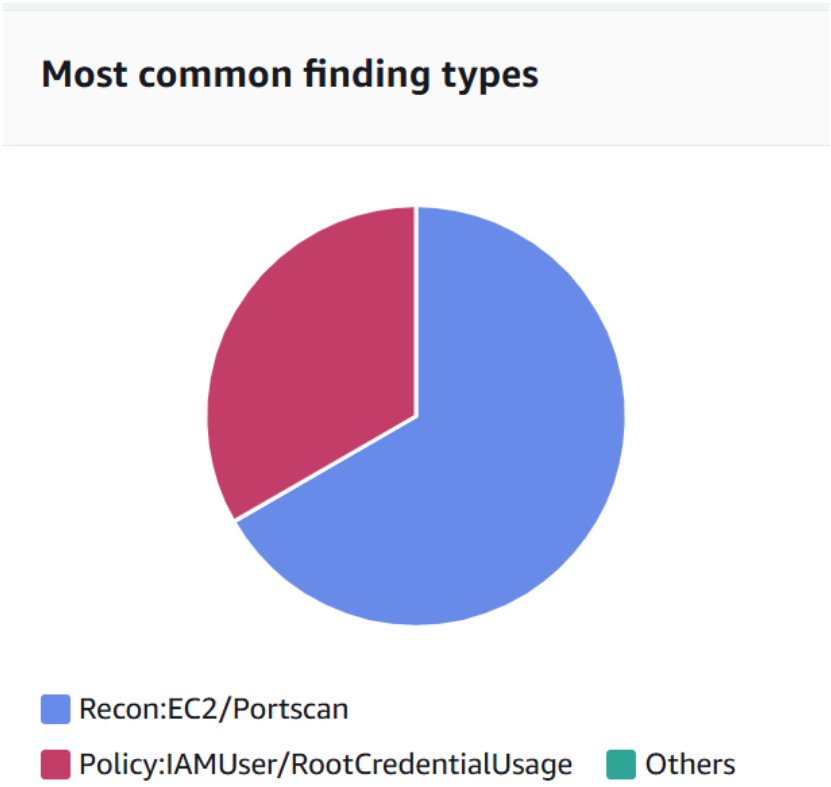
Suricata was installed and successfully ran on the EC2 instance, but an important detail was observed during runtime testing.

The Suricata output displayed the message:

SC_ERR_NO_RULES - No rule files loaded.

This message indicates that although Suricata was active and monitoring the network interface, it lacked detection rule sets. Without rules, Suricata cannot detect:

- Port scans
- Brute-force attempts
- Malware signatures
- Suspicious payloads



7. REMEDIATION AND HARDENING PLAN

In a real-world environment, responding to security incidents is a structured and time-sensitive process. A strong remediation plan ensures early containment, root cause identification, recovery, and future prevention. The remediation strategy for this project is divided into Immediate, Short-Term, Medium-Term, and Long-Term phases.

7.1 Immediate Remediation (0–2 Hours)

1. Contain Suspicious Portscan Activity

If the outbound port scanning was not intentional, it may indicate a compromised EC2 instance, unauthorized remote access, or malware attempting to spread.

Actions include:

- Remove outbound access from the instance's Security Group.
- Move the instance into a quarantine group that denies all outbound traffic.
- Allow SSH only from admin IP for investigation.
- Stop the instance if necessary to prevent further damage.

2. Investigate Root Credential Usage

GuardDuty detected the use of root credentials, which is a high-risk security issue.

Immediate steps:

- Verify who used the root credentials.
- Review CloudTrail logs for unusual API calls.
- Ensure no unexpected IAM users, roles, or policies were created.

3. Secure the Root Account

- Delete any root access keys.
- Enable root MFA immediately.
- Store root credentials securely offline.
- Avoid using root for any operational tasks.

4. Preserve Evidence for Forensics

- Take AMI and EBS snapshots.
- Export GuardDuty findings.
- Collect Suricata logs and OS-level logs.

7.2 Short-Term Remediation (Within 24 Hours)

1. Install Suricata Rule Sets

Suricata requires rule sets for attack detection. Recommended action:

sudo suricata-update sudo systemctl restart suricata

2. Enable Centralized Logging

Enable CloudTrail, VPC Flow Logs, and Route 53 DNS logging. Send Suricata EVE JSON logs to CloudWatch or a SIEM for correlation.

3. IAM Hardening

- Identify over-privileged IAM roles.
- Apply least privilege access principles.
- Enforce MFA for all accounts.
- Rotate long-term access keys.
- Disable unused accounts.

7.3 Medium-Term Remediation (1–3 Days)

1. Automated GuardDuty Responses

Use Lambda + EventBridge to automate responses:

- Quarantine instances showing Recon events.
- Disable IAM keys flagged by GuardDuty.
- Notify administrators via SNS alerts.

2. Improve Network Segmentation

- Use public subnets only for load balancers.
- Move EC2 workloads to private subnets.
- Apply outbound egress restrictions.

3. Host Hardening

- Run OS updates using apt upgrade.
- Disable unused services and ports.
- Enable key-only SSH authentication.

7.4 Long-Term Remediation (1–4 Weeks)

1. Build Incident Response Playbooks

Document repeatable processes for:

- Recon alert handling
- IAM anomaly response
- Isolation workflow
- Evidence collection

2. Implement Full Zero-Trust Architecture

- Continuous identity verification
- Device posture checks
- Least privilege enforcement
- Network segmentation

3. Adopt Keyless AWS Infrastructure• Use IAM Roles instead of long-term access keys.

- Prefer STS temporary session credentials.
- Use session-based authentication mechanisms.

8. TESTING AND VALIDATION

Testing validates whether the monitoring and detection systems are functioning properly.

8.1 Apache Functionality Testing

The Apache default webpage was successfully accessed using the EC2 public IP. This proves:

- Web server is correctly installed
- Port 80 rules are configured properly
- The instance is publicly reachable

This is important because attackers typically target public endpoints.

8.2 GuardDuty Detection Testing

Intentional activities triggered:

- Outbound portscan
- Root credential usage

GuardDuty quickly detected both, proving:

- Threat intelligence sources are active
- Analysis of API logs and VPC Flow Logs works flawlessly
- The system is effective for cloud-level threat monitoring

8.3 Suricata Testing

Two test procedures were performed:

1. Service ValidationThe command:

systemctl status suricata

showed that Suricata is:

- Active
- Running
- Listening on the correct interface

2. Runtime Test

Running Suricata in verbose mode:

suricata -i eth0 -v

displayed initialization logs and thread creation. However, the warning:

No rule files loaded

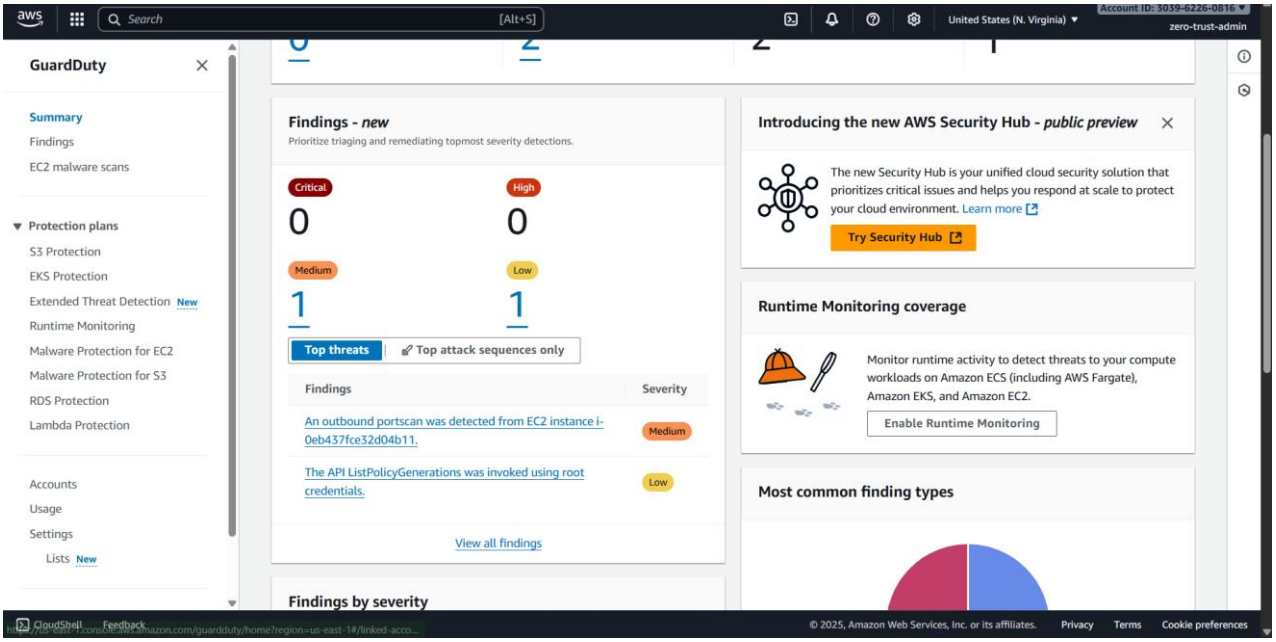
indicated rule sets need to be installed to activate detection.

8.4 Further Validation Needed

To fully validate the IDS pipeline:

- Install ET rules
- Perform a controlled Nmap scan
- Verify alerts appear in /var/log/suricata/eve.json

This final step confirms Suricata's detection capability.



9. RESULTS AND DISCUSSION

After deploying the monitoring components and simulating security events, the system produced multiple useful insights. This section interprets the outcomes of GuardDuty, Suricata, and Apache testing, examining how effectively each component contributed to the overall ZeroTrust monitoring approach.

9.1 GuardDuty Detection Results

GuardDuty successfully detected two major types of suspicious activity: outbound port scanning and root account API usage. These findings indicate that GuardDuty's log analysis and threat intelligence mechanisms functioned correctly in identifying real anomalies. Port scanning represented reconnaissance-level behavior, whereas root credential usage highlighted identityrelated risks. Both detections validate the effectiveness of GuardDuty's monitoring pipeline.

9.2 Suricata Results

Suricata installation and initialization were successful. It recognized the network interface, initialized threads, and began monitoring traffic. Although Suricata did not generate signaturebased detections due to missing rule sets, its engine and logging system functioned properly. Once rules are added, Suricata can detect malware traffic, exploit attempts, and suspicious packet-level behavior, complementing GuardDuty's cloud-level detections.

9.3 Apache Server Results

The Apache web server hosted on the EC2 instance responded successfully to both local and external HTTP requests. Accessing the public IP confirmed that:

- Apache was installed and running correctly
- Security group rules allowed public HTTP access
- The instance was reachable from the internet

This validated the environment for testing, as attackers typically target public-facing services when attempting reconnaissance or exploits.

9.4 Overall System Behavior

When analyzing the results holistically, several strengths and limitations were identified.

Strengths Observed:

- GuardDuty produced accurate and timely detections
- Suricata validated host-level monitoring functionality
- Apache provided a realistic attack surface for testing
- Combined tools enabled multi-layered Zero-Trust monitoring

Limitations Identified:

- Suricata required rule sets for complete detection capabilities
- Automated remediation workflows were not implemented at this stage
- Centralized log correlation (SIEM) was not yet deployed

10. CONCLUSION AND FUTURE SCOPE

10.1 Conclusion

This project successfully implements a foundational Zero-Trust threat detection environment by combining AWS GuardDuty with Suricata IDS on an EC2 instance. Through structured deployment, active monitoring, and simulated threat behaviors, the system was able to detect and highlight suspicious activities such as port scanning and root account misuse.

The major outcomes of the project include:

- Deployment of a realistic test environment An Apache web server on EC2 provided an accessible endpoint that attackers typically target.
- Successful integration of cloud-native detection (GuardDuty) GuardDuty immediately detected risky activities using AWS's intelligent threat analysis pipeline.
- Successful deployment of host-based detection (Suricata) Suricata proved its operational readiness for packet-level monitoring.
- Real-world insights into cloud monitoring challenges The project highlighted the crucial differences between cloud API-layer threats and network-layer threats.
- Development of a layered security vision The project showed that neither cloud-native nor host-based solutions alone are enough—together, they form a powerful detection framework.

Overall, this project demonstrates a practical, beginner-friendly, yet powerful implementation of Zero-Trust principles using free-tier resources. It proves that sophisticated cloud security monitoring can be achieved even in resource-constrained environments.

10.2 Future Scope

To expand this project into a fully professional-grade Zero-Trust security solution, several enhancements can be considered:

1. Implement Automated Incident Response

Using AWS Lambda and EventBridge rules, GuardDuty alerts can trigger:

- Automatic instance isolation
- Disabling IAM keys
- Removing suspicious outbound egress rules
- Sending alerts to administrators2. Integrate a Centralized SIEM

Implement:

- AWS CloudWatch Logs
- OpenSearch/Kibana
- Grafana dashboards

This allows correlation of Suricata logs, GuardDuty alerts, VPC Flow Logs, and authentication events.

2. Expand Suricata Capabilities

- Install Emerging Threats (ET) rules
- Enable IPS (Intrusion Prevention) mode
- Create custom detection rules4. Implement Network Segmentation

Use:

- Public and private subnets
- NAT gateways
- Internal load balancers

Integrate Identity Zero-Trust

- Single Sign-On
- Device posture checks
- Just-in-time role elevation

3. Infrastructure as Code (IaC)

Use Terraform or CloudFormation to automate deployment of:

a. EC2
b. Suricata configuration
c. Logging infrastructure
d. GuardDuty settings
e.

4. Add Machine Learning–Based Anomaly Detection

Tools like Amazon Detective or custom ML models can provide deeper insights into user and network behavior.

5. Build Red Team Automation

Simulate:

a. Brute-force attacks
b. Packet floods
c. Exploitation attempts
d. Privilege escalationto continuously test and improve the security posture.

REFERENCES

1. Amazon Web Services. *AWS GuardDuty – Documentation and User Guide*. Available at: https://docs.aws.amazon.com/guardduty/latest/ug/what-is-gua rdduty.html

2. Amazon Web Services. *Amazon EC2 Documentation*. Available at: https://docs.a ws.amazon.com/ec2/

3. AWS CloudTrail Documentation. Available at: https://docs.aws.amazon.co m/cloudtrail/

4. AWS VPC Flow Logs – User Guide. Available at: https://docs.aws.amazon. com/vpc/latest/userguide/flow-logs.html

5. AWS DNS Logging Documentation. Available at: https://docs.aws.amazon. com/route53/

6. Suricata Official Documentation — OISF. Available at: https://suricata.io/do cumentation/

7. Emerging Threats (ET) Ruleset for Suricata. Available at: https://rules.emergi ngthreats.net/

8. NIST SP 800-207. *Zero Trust Architecture*. Available at: https://csrc.nist.go v/publications/detail/sp/800-207/final

9. Forrester Research. *Zero Trust Extended (ZTX) Framework*. Available at: https: //www.forrester.com/report/Zero-Trust-eXtended-Ecosystem/R ES137229

10. Apache HTTP Server Documentation. Available at: https://httpd.apache.org /

11. Amazon Detective – Behavior Analysis for AWS Security. Available at: https://aw s.amazon.com/detective/

12. AWS Well-Architected Framework – Security Pillar. Available at: https://docs.a ws.amazon.com/wellarchitected/latest/security-pillar/intro.