

Simulated Threat Detection with AWS GuardDuty

Objective

To demonstrate applied knowledge of AWS GuardDuty by simulating cloud-based threat detection, interpreting critical alerts, and analyzing real-world security scenarios in a controlled environment.

Actions Taken

1. Enabled AWS GuardDuty:

Activated GuardDuty in a fresh AWS Free Tier environment to monitor for potential security threats across EC2, IAM, S3, and RDS services.

2. Simulated Threat Data:

Used GuardDuty's built-in "Generate sample findings" feature to simulate common attack scenarios without risking live infrastructure or data.

The screenshot displays the AWS GuardDuty console interface. On the left, a sidebar contains navigation links for Summary, Findings, EC2 malware scans, Protection plans (S3, EKS, Extended Threat Detection, Runtime Monitoring, Malware Protection for EC2/S3, RDS, Lambda), Accounts, Usage, Settings, Lists, What's New, and Partners. The main panel is titled 'Findings (363)' and includes filters for Saved rules, Status (Current), and Threat type (All findings). A table lists several findings with columns for Title, Severity, Finding type, Resource, and Count. The findings include root credential usage, malicious file execution, DNS rebinding attacks, Denial of Service (DoS) attempts, process injection, and domain name queries.

Title	Severity	Finding type	Resource	Count
The API GetEnvironmentStatus was invoked using root credentials.	Low	Policy:IAMUser/RootCredentialUsage	Access Key: ASIAQ45XXNQGVHKO3IJC	112
The API ListAttachedRolePolicies was invoked using root credentials.	Low	Policy:IAMUser/RootCredentialUsage	Access Key: ASIAQ45XXNQGVVNC GCM2	16
[SAMPLE] A resource of type EKSCluster has executed malicious file.	High	Execution:Runtime/MaliciousFileExecuted	EKS Cluster: GeneratedFindingEKSClusterName	1
[SAMPLE] The EC2 instance i-99999999 may be the target of a DNS rebinding attack.	High	UnauthorizedAccess:Runtime/MetadataDNSRebind	ECS Cluster: GeneratedFindingECSClusterName	1
[SAMPLE] The EC2 instance i-99999999 is behaving in a manner that may indicate it is being used to perform a Denial of Service (DoS) attack using an unusual protocol.	High	Backdoor:EC2/DenialOfService.UnusualProtocol	EC2 Instance: i-99999999	1
[SAMPLE] Process injection via proc was detected in EC2 instance i-99999999.	High	DefenseEvasion:Runtime/ProcessInjection.Process	ECS Cluster: GeneratedFindingECSClusterName	1
[SAMPLE] A domain name related to known abused domains was queried by EC2 instance i-99999999.	Medium	Impact:Runtime/AbuseDomainRequest.Reputation	Kubernetes Cluster:	1
[SAMPLE] A Kubernetes API commonly used in Discovery tactics invoked from a known malicious IP address.	Medium	Discovery:Kubernetes/MaliciousIPCaller	EKS Cluster: GeneratedFindingEKSClusterName	1

3. Analyzed Multiple Threat Categories:

- Reconnaissance: Simulated port scanning and metadata scraping attempts
- Unauthorized Access: Sample alerts for brute-force login attempts
- Defense Evasion: Process injection and malware behavior detection
- Credential Abuse: Use of root account and malicious IP login attempts

4. Reviewed Severity and Recommendations:

Classified alerts by severity and reviewed GuardDuty's integrated remediation guidance and service-specific risks.

Deep Dive: High Severity Finding

Finding: DefenseEvasion\Runtime/ProcessInjection.Proc

<input type="checkbox"/>	[SAMPLE] Process injection via proc was detected in EC2 instance i-99999999.	High	DefenseEvasion:Runtime/ProcessInjection.Proc	ECS Cluster: GeneratedFindingECSClusterName	1
--------------------------	--	------	--	---	---

Severity: 8/8 (Critical)

Summary: Simulated detection of a process injection attempt within an EC2 instance, where one process attempts to modify another's memory via the `/proc` filesystem, a common tactic used for evading defenses, escalating privileges, or disabling security agents.

Why It Matters

Process injection is a high-risk evasion technique. If successful, attackers can hijack legitimate processes to run malicious code, bypass security controls, and maintain persistence on the instance, potentially compromising your AWS environment undetected.

How I Would Investigate

- Review VPC Flow Logs for unusual outbound traffic or connections from the compromised instance.
- Examine CloudTrail logs for unusual API activity tied to the instance or its IAM role.

- Snapshot the EC2 instance disk and memory for forensic analysis (use tools like Volatility).
- Audit the ECS cluster tasks and container images for unauthorized changes or rogue deployments.
- Check for recent process creations, hidden processes, or unexpected open ports on the instance.

GuardDuty Findings

Findings (363)

Create suppression rule

Actions

Saved rules: Apply saved rules

Filter findings

Status: Current Threat type: All findings

Process injection via proc was detected in EC2 instance i-999999999.

High First seen 2 hours ago, last seen 2 hours ago

The process GeneratedFindingProcessName in EC2 instance i-99999999 has modified memory files under /proc.

[Investigate with Detective](#)

Investigate with Detective

Detective visualizes the CloudTrail, VPC flow data and EKS audit logs for the resources affected by this finding.

- GuardDuty finding** [036ad4cb659846f1b374ff42ec28cddb](#)
- EC2 instance** [i-999999999](#)
- AWS account** [062125468685](#)
- IP address** [198.51.100.0](#)

Task details

ARN	arn:aws:ecs:us-east-1:123456789012:cluster/sample-cluster
Status	ACTIVE
ARN	arn:aws:ecs:us-east-1:123456789012:task/sample-cluster/12345678-1234-1234-1234-123456789012

How I Would Remediate

- Immediately quarantine the EC2 instance (**i-999999999**) by applying a restrictive security group or isolating it from the network.
- Terminate the instance after collecting forensic evidence.
- Rotate any associated IAM roles and credentials.

- Patch and redeploy a clean, hardened instance from a trusted AMI.
- Enforce runtime security tools (e.g., AWS Inspector, Falco).
- Tighten IAM permissions following least privilege principles.
- Set up GuardDuty-to-SNS or EventBridge alerts for automated response on similar findings.

Key Takeaways

- Learned how AWS GuardDuty detects sophisticated runtime evasion tactics like process injection without requiring an agent.
- Gained hands-on skills in analyzing defense evasion scenarios using AWS-native services and logs.
- Developed a structured investigation and remediation process for critical cloud runtime threats.
- Strengthened incident response readiness for advanced attack techniques targeting EC2 and containerized workloads.

As part of this exercise, I also exported the full set of simulated GuardDuty findings in .json format for offline analysis and potential integration with third-party SIEM or SOAR platforms. This artifact demonstrates my familiarity with handling threat data in structured formats, enabling advanced analysis, reporting, and automated response workflows.

This simulation reinforced my capability to operationalize AWS-native security service, interpret critical findings, and craft actionable response strategies under realistic cloud attack scenarios. Beyond technical execution, it highlights my proactive approach to security engineering, combining detection, investigation, and remediation into a cohesive incident response workflow.