Sprint 2: Core Architecture & SIEM Integration

Team Name: Cyber Shawties LLC

Project Title: Cloud-Native SIEM with Secure Governance

Course: Cybersecurity Capstone - Phase 3

Sprint: 2 (Core Architecture Build)

Submission Date: October 8, 2025

---

1. Infrastructure as Code (IaC)

Our project follows an **IaC-first principle**, ensuring all cloud infrastructure is provisioned through code for repeatability, auditability, and version control.

- **IaC Tool: Terraform v1.13.3** was selected for its multi-cloud capabilities, robust module ecosystem, and native integration with AWS services.
- **GitHub Repository:** All Terraform code is stored in a central GitHub repository, using a `main`/`develop`/`feature` branch structure to manage changes.
- **Core Infrastructure Provisioned:**
    - **Networking (VPC):** A custom VPC (`kms-siem-vpc`) was created with a CIDR block of `$10.0.0.0/16$`. This includes:
        - A public subnet (`$10.0.1.0/24$`) for resources requiring internet access.
        - A private subnet (`$10.0.2.0/24$`) for backend resources like the Wazuh manager.
        - An Internet Gateway (`kms-siem-igw`) and NAT Gateway (`kms-siem-nat`) to manage traffic flow.
    - **IAM:** A series of least-privilege IAM roles were created, including a `securityopsrole` for security analysis and a `wazuh-log-reader` for SIEM integration. All policies are scoped to specific resources and actions, avoiding wildcards (`*`) wherever possible.
    - **S3 Buckets:** Logging buckets (e.g., `mindbodysecure-logs`) were created to store CloudTrail, AWS Config, and other operational logs. These buckets are hardened with strict security settings.

---

2. Updated Topology & Architecture Diagrams

The architecture is designed for security, resilience, and observability, aligning with the AWS Well-Architected Framework. It establishes a secure foundation for ingesting and analyzing sensitive security data.

Diagram Description

The architecture consists of a foundational network layer, a robust encryption and governance framework, and an agentless SIEM integration for observability.

1. **Data Sources (AWS):** Services like CloudTrail, AWS Config, and CloudWatch generate logs and metrics based on account activity.
2. **Log Aggregation (S3):** All logs are delivered to a centralized and hardened S3 bucket (`mindbodysecure-logs`) for durable storage.
3. **SIEM Ingestion (Wazuh):** The Wazuh manager, running in the private subnet, uses its `aws-s3` wodle to periodically poll the S3 bucket and ingest new CloudTrail logs.
4. **Alerting & Monitoring (CloudWatch & SNS):** CloudWatch Alarms monitor for specific events (e.g., Root Account Usage, Unauthorized API calls) and trigger an SNS topic, which sends notifications via HTTPS to the Wazuh SIEM for correlation.
5. **Analysis & Visualization (Wazuh Dashboard):** Analysts use the Wazuh Dashboard (powered by OpenSearch) to view alerts, investigate events, and monitor compliance.

AWS Well-Architected Framework Alignment

- **Security Pillar:** Implemented a multi-key KMS strategy for data encryption, enforced least-privilege IAM roles, hardened S3 buckets, and enabled detective controls like GuardDuty and CloudTrail.
- **Reliability Pillar:** Utilized managed AWS services like S3 and SNS for high availability. The infrastructure is defined in Terraform, allowing for consistent and automated deployments and recovery.
- **Cost Optimization Pillar:** Leveraged an open-source SIEM (Wazuh) to avoid per-GB ingestion costs associated with commercial tools. S3 lifecycle policies can be used to move older logs to cheaper storage tiers.

---

3. SIEM & Security Tool Integration

This sprint focused on integrating AWS native security services with our open-source SIEM, Wazuh, to create a comprehensive, multi-layered security monitoring solution.

AWS Services Configured

🔐 KMS Hardening & Configuration

A multi-key encryption strategy was implemented to ensure data isolation and compliance. Key policies are configured to enforce a strict separation of duties between Key Administrators and Key Users. All customer-managed keys are set for automatic annual rotation.

The following steps were taken to encrypt CloudWatch log groups with a customer-managed KMS key:

1. **Create KMS Key:** A symmetric KMS key was created for encrypting log data.

2. **Apply Key Policy:** The key policy was updated to grant usage permissions to the CloudWatch Logs service (`logs.<REGION>.amazonaws.com`) and CloudTrail (`cloudtrail.amazonaws.com`).
3. **Associate Key with Log Group:** The KMS key was attached to the target CloudWatch log group using the AWS CLI to encrypt all future log data.
4. Bash

```
aws logs associate-kms-key \
  --log-group-name "<YOUR_LOG_GROUP_NAME>" \
  --kms-key-id "<YOUR_KMS_KEY_ARN>"
```

5. **Verify Encryption:** The configuration was verified by checking the log group's properties.

🪣 S3 Hardening

All S3 buckets, especially those containing logs, were hardened using a comprehensive checklist to prevent unauthorized access and ensure data integrity.

1. **Block Public Access:** All public access settings were enabled at both the account and bucket level to prevent accidental exposure.
2. **Enable Encryption:** Server-Side Encryption with KMS (SSE-KMS) was enforced as the default for all logging buckets.
3. **Enable Versioning & Logging:** Bucket versioning and MFA Delete were enabled to protect against accidental deletion, and server access logging was directed to a separate, secure logging bucket.
4. **Apply Least Privilege:** Bucket policies were configured to deny insecure connections (`aws:SecureTransport: false`) and restrict access to specific IAM roles and VPC endpoints.
5. **Enable Continuous Compliance:** AWS Config rules like `s3-bucket-public-read-prohibited` were enabled to continuously audit S3 settings for compliance.

Third-Party Tool Integration

🛡️ Wazuh (Agentless SIEM)

Wazuh was configured for agentless monitoring to ingest AWS logs directly, reducing management overhead on EC2 instances.

1. **Enable AWS S3 Module:** The `aws-s3` wodle in Wazuh was configured to connect to the `mindbodysecure-logs` S3 bucket to pull CloudTrail logs. The `ossec.conf` file was updated as follows:

2. XML

```
<wodle name="aws-s3">
  <disabled>no</disabled>
  <interval>10m</interval>
  <run_on_start>yes</run_on_start>
  <bucket type="cloudtrail">
    <name>mindbodysecure-logs</name>
    <aws_profile>default</aws_profile>
  </bucket>
</wodle>
```

3.
4.
5. **Map CloudTrail Decoders:** Wazuh's built-in decoders for AWS were verified to ensure that incoming JSON-formatted CloudTrail logs are correctly parsed.
6. **Create Custom Rules:** Custom rules were written in `/var/ossec/etc/rules/local_rules.xml` to generate specific alerts for important AWS events.
7. XML

```
<group name="aws,cloudtrail">
  <rule id="100300" level="5">
    <decoded_as>json</decoded_as>
    <field name="eventSource">cloudtrail.amazonaws.com</field>
    <description>Amazon CloudTrail event detected</description>
  </rule>


  <rule id="100301" level="7">
    <if_sid>100300</if_sid>
    <field name="eventName">ConsoleLogin</field>

    <description>Console login detected in CloudTrail logs</description>

  </rule>

</group>
```

8.
9.
10. **Restart & Verify:** After restarting the Wazuh manager, logs were checked to confirm successful connection and ingestion from the S3 bucket.

---

4. End-to-End Workflow Validation

The following test cases were executed to validate the complete workflow from event generation in AWS to alerting in the Wazuh SIEM.

Test Case 1: CloudTrail to Wazuh Log Ingestion

- **Objective:** Validate that a standard AWS API action is logged by CloudTrail, ingested by Wazuh, and triggers an alert.
- **Steps:**
    1. An S3 bucket named `test-bucket-cloudguardians` was created via the AWS Console.
    2. Waited for CloudTrail to deliver the event log to the `mindbodysecure-logs` S3 bucket.
    3. Waited for the Wazuh manager's next 10-minute polling interval to ingest the new log file.
- **Expected Result:** An alert for the `CreateBucket` event should appear in the Wazuh dashboard, containing details like the user identity, source IP, and bucket name.
- **Actual Result:** The `CreateBucket` event appeared in the Wazuh dashboard approximately 6 minutes after the action was performed. The alert details matched the expected output.
- **Status: PASSED** ✅

Test Case 2: CloudWatch Alarm to Wazuh Alert

- **Objective:** Validate that a CloudWatch alarm based on a metric filter correctly triggers an SNS notification that is ingested by Wazuh.
- **Steps:**
    1. A CloudWatch alarm was configured to monitor for "Unauthorized API Calls" in CloudTrail logs.
    2. An action was performed using an IAM user with insufficient permissions to trigger the alarm.

3. The alarm transitioned to the IN ALARM state.

- **Expected Result:** The alarm should trigger the SNS topic, which sends a notification to the Wazuh HTTPS endpoint, generating a high-severity alert in the Wazuh dashboard.
- **Actual Result:** The alarm fired as expected, and an alert appeared in Wazuh within 2 minutes of the unauthorized action.
- **Status: PASSED** ✅

---

5. Team Member Contributions (Sprint 2)

- **Team Lead:** Coordinated sprint goals, managed the project backlog in Jira, and facilitated daily stand-ups.
- **Infrastructure Engineer:** Authored the Terraform modules for the VPC, subnets, gateways, and S3 buckets. Configured the S3 backend for Terraform state management.
- **Security Analyst / SIEM Engineer:** Deployed and configured the Wazuh manager. Developed the agentless integration with AWS CloudTrail, wrote custom detection rules, and configured the CloudWatch-to-Wazuh alerting pipeline.
- **Developer:** Created the IAM roles and policies using Terraform, including the securityopsrole. Implemented the KMS multi-key strategy and associated keys with logging services.
- **Documentation Owner:** Authored the Sprint 2 report, created architecture diagrams, and documented the S3 hardening and Wazuh integration playbooks.

Sources

- Technical Paper: Implementation of a Secure Governance and Access Control Architecture for a Cloud-Native SIEM Platform
- Phase 3: CloudWatch + Wazuh SIEM Integration Playbook
- Universal Capstone Project Rubric and Deliverables — Student Guidev3
- Sprint 2 - Example Submission
- KMS Key Strategy Summary
- Role Overview: securityopsrole
- IAM Hardening Checklist
- IAM Audit Checklist (Step-by-Step)