Cloud Security Monitoring in GCP

#3 Applications and Data Security Monitoring Lab Guide

1. Lab Overview

Deploy a vulnerable application and a Wazuh SIEM stack, then use logs, traces, and infrastructure telemetry to detect malicious behavior and data access anomalies.

This lab applies the concepts for:

- Secure logging patterns
- Data access monitoring
- Alert design
- SIEM correlation workflows

Objectives:

- Design security-relevant application logs
- Route logs into a SIEM (Wazuh)
- Detect application-layer attacks
- Detect anomalous data access
- Understand how SIEM correlation works across layers

2. Deploy a vulnerable Application

Steps

1. Review Terraform components under `3.1applications_vuln_app` resources for the vulnerable application deployment:
    a. Infrastructure Components:
        i. Google Cloud Function - serverless API
        ii. API Gateway - Public-facing REST API with OpenAPI specification
        iii. Cloud Storage Bucket - For storing uploaded images
        iv. Firestore Database - For storing metadata
        v. Identity Platform - For user authentication (Firebase Auth)
        vi. Service Account - With appropriate IAM permissions
    b. Available API endpoints:
        i. GET /health
        ii. POST /auth/audit
        iii. GET /profile

iv. POST /images/upload
v. GET /images/{imageId}
vi. DELETE /images/{imageId}
vii. GET /export
viii. GET /admin

2. Understand the logging mechanism and structure for the App
    a. Core Log Fields in the logging interface
    b. stdout/stderr to Cloud Logging
3. Enable required APIs

```shell
PROJECT_ID=<your_project_id>
gcloud auth login
gcloud config set project $PROJECT_ID
gcloud services enable \
  compute.googleapis.com \
  identitytoolkit.googleapis.com \
  apigateway.googleapis.com \
  cloudfunctions.googleapis.com \
  cloudbuild.googleapis.com \
  run.googleapis.com \
  artifactregistry.googleapis.com \
  logging.googleapis.com \
  storage.googleapis.com \
  firestore.googleapis.com \
  cloudresourcemanager.googleapis.com
```

4. Create terraform.tfvars from terraform.tfvars.example, update and deploy Terraform resources

```shell
terraform init
terraform plan
terraform apply
```

5. Using the test-api.sh script, test the API endpoints

```shell
chmod +x test-api.sh
./test-api.sh YOUR_API_KEY YOUR_GATEWAY_URL
```

6. Explore the logs generated by the tests

3. Prepare log routing infrastructure for Wazuh

Steps:

1. Create Pub/Sub infrastructure

```SQL
#Create a topic
gcloud pubsub topics create wazuh-gcp-logs
#Create a subscription
gcloud pubsub subscriptions create wazuh-gcp-logs-sub \
  --topic wazuh-gcp-logs
```

2. Create a Cloud Logging sink:
    a. Make sure to enable Cloud Storage data access and Audit Logs
    b. Filter cloud_function gcs_bucket and cloudaudit as filters
    c. Configure sink destination for pubsub.googleapis.com/projects/<PROJECT_ID>/topics/wazuh-gcp-logs
3. Grant permissions for Wazuh
    a. Grant permissions to wazuh-gcp-logs
    b. Create a service account for Wazuh
    c. Grant minimal permissions
    d. Create key (clearly not the secure way)

```Shell
#a
gcloud pubsub topics add-iam-policy-binding wazuh-gcp-logs \
  --member="serviceAccount:<sink-sa>" \
  --role="roles/pubsub.publisher"
#b
gcloud iam service-accounts create wazuh-log-reader
#c
gcloud projects add-iam-policy-binding <PROJECT_ID> \

--member="serviceAccount:wazuh-log-reader@<PROJECT_ID>.iam.gserviceaccount.com"
\
```

```
  --role="roles/pubsub.subscriber"
#d
gcloud iam service-accounts keys create wazuh.json \
  --iam-account=wazuh-log-reader@<PROJECT_ID>.iam.gserviceaccount.com
```

4. Deploy Wazuh

Steps:

1. Clone lab resources repository and Review Terraform components under `3applications_wazuh`, and update the terraform variables file

```
Shell
git clone https://github.com/0x74696D/security_monitoring_labs
cd 3applications
terraform plan
```

2. Deploy Wazuh

```
Shell
terraform apply
```

3. Use ssh port forwarding to access Wazuh dashboard

```
Shell
gcloud compute ssh wazuh-aio --zone us-central1-a --project $PROJECT_ID -- -L
5601:localhost:5601
```

4. Navigate to Wazuh dashboard http://localhost:5601/app/login? And login using admin/<your_configured_password>
5. Under discover, explore indexed logs under wazuh-alerts-* index.
6. Configure custom logs indexing

```shell
chmod +x configure_index.py
python3 configure_index.py --password <your_configured_password>
```

5. Implement detection

Steps:

1. Password-based login attacks
    a. Identify the login endpoint (firebase logs) and familiarize yourself with a successful login attempt and unsuccessful one.
    b. Using Alerting → Alerts, with create a monitor to:
        i. Detect brute force attacks
        ii. Detect credential stuffing attack
        iii. Detect Successful Login After Brute Force
2. Detect Direct Data Access
    a. Identify profile images access requests. Familiarize yourself with app access vs direct access logs
    b. Create an alert to detect Direct Data Access to the bucket.

[Optional]
Create detection for:

- IDOR (Insecure Direct Object Reference) - Image Download
- Unauthorized Access Attempts
- Admin Endpoint Access Attempt
- Weak Token Validation Bypass

Lab Resources:

- GKE Audit logging
- Wazuh documentation
- Monitoring Google Cloud Pub/Sub
- OWASP WWW Attacks
- Wazuh Query Language