**LAB08 cloud application security coding**

| Class | CT201H [M01-M04] |
| --- | --- |
| Student ID | B2111933 |
| Name | TruongDangTrucLam |
| Email address | lamb2111933@student.ctu.edu.vn |
| Browser | Chrome |

**System environment for developing**

| Resources | Sender(attacker) | Receiver(victim) | Homepage |
| --- | --- | --- | --- |
| OS |  |  |  |
| IP address |  |  |  |
| URL |  |  |  |
| Web browser |  |  |  |
| CSS language |  |  |  |
| Web server |  |  |  |
| Web application |  |  |  |
| DB server script |  |  |  |
| Others |  |  |  |

**Select one model and exercise for cloud security**

**[Model A]**

1. Survey the **Method 1 model for developing your project**

2. Exercise **or** Explain **Method 1**

**[Model B]**

**How to develop cloud security application on your current system(PC, notebook) and install on cloud server**

**Prerequisites:**

* Python3 installed on your local system.
* A cloud server (e.g., AWS EC2, GCP Compute Engine, Azure VM) with a suitable operating system (e.g., Ubuntu, CentOS).
* Basic understanding of cloud infrastructure and networking.

**1. Create a Local Project Directory:**

$ mkdir cloud\_security\_app  
 $ cd cloud\_security\_app

**2. Initialize a Python Project:**

Create a new Python file (e.g., main.py) within the project directory.

**3. Install Required Libraries:**

Use pip to install the necessary libraries for your cloud security application. For example, you might need libraries like boto3 for AWS, google-cloud-storage for GCP, or azure-storage-blob for Azure.

Add the following line to your main.py file to install the required libraries:

!pip install boto3 google-cloud-storage azure-storage-blob

**4. Develop Cloud Security Logic:**

Write the Python code in main.py to implement the desired cloud security functionality. This might involve:

* + - **Authentication and Authorization:** Implement mechanisms to authenticate users and authorize their access to cloud resources.
    - **Resource Monitoring:** Monitor cloud resources for anomalies, vulnerabilities, or unauthorized access.
    - **Data Encryption:** Encrypt sensitive data stored in the cloud.
    - **Access Control:** Implement fine-grained access control policies for cloud resources.
    - **Vulnerability Scanning:** Scan cloud resources for vulnerabilities and take appropriate actions.
    - **Incident Response:** Define procedures for handling security incidents and breaches.

Here's a basic example of how you might implement resource monitoring using the boto3 library for AWS:

import boto3

def monitor\_ec2\_instances():

ec2\_client = boto3.client('ec2')

response = ec2\_client.describe\_instances()

for reservation in response['Reservations']:

for instance in reservation['Instances']:

instance\_id = instance['InstanceId']

instance\_state = instance['State']['Name']

print(f"Instance ID: {instance\_id}, State: {instance\_state}")

monitor\_ec2\_instances()

**5. Test and Refine:**

Run your main.py file locally to test the functionality and make any necessary adjustments.

**6. Package the Application:**

If you want to distribute your application, you can create a package using tools like setuptools. This will bundle your code and dependencies into a distributable format.

**7. Deploy to Cloud Server:**

**7.1. Create an AWS Account and IAM Role:**

* If you don't already have one, create an AWS account.
* Create an IAM role with the necessary permissions to interact with EC2 instances. Assign the AmazonEC2FullAccess policy to this role.

**7.2. Upload the Script to S3:**

* Create an S3 bucket and upload your Python script to it.
* Make sure the script is publicly accessible or grant the IAM role you created in step 1 read-only access to the script.

**7.3. Create a Lambda Function:**

* In the AWS Lambda console, create a new function.
* Choose "Python" as the runtime and select the appropriate version (e.g., Python 3.9).
* Configure the function's name, description, and timeout.
* For the role, select the IAM role you created in step 1.
* In the function's code editor, paste the contents of your Python script.

**7.4. Deploy the Lambda Function:**

* Click the "Save" button to deploy the Lambda function.

**7.5. Test the Function:**

* Use the Lambda console to test the function manually or configure a test event.
* Verify that the function is successfully monitoring EC2 instances and printing the desired output.