**LAB09 security application coding using complex AI**

| 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

[Model A]

1. Survey how to code security application program using complex AI
2. Explain your security coding process

[Model B]

1. **Code simplified cloud security application code using complex AI**

I will generate code for AWS:

import boto3

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Initialize AWS clients

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

s3\_client = boto3.client('s3')

# Function to fetch and process EC2 instance data

def fetch\_ec2\_data():

response = ec2\_client.describe\_instances()

instances = response['Reservations'][0]['Instances']

# Extract relevant data (adjust as needed)

data = []

for instance in instances:

instance\_id = instance['InstanceId']

instance\_type = instance['InstanceType']

launch\_time = instance['LaunchTime']

public\_ip = instance['PublicIpAddress']

private\_ip = instance['PrivateIpAddress']

security\_groups = instance['SecurityGroups']

data.append([instance\_id, instance\_type, launch\_time, public\_ip, private\_ip, security\_groups])

return data

# Function to preprocess data and create features

def preprocess\_data(data):

# Convert data to NumPy array

data = np.array(data)

# Encode categorical features (adjust as needed)

encoded\_data = np.zeros((len(data), len(data[0])))

for i, row in enumerate(data):

encoded\_data[i, 0] = instance\_type\_mapping[row[1]] # Example mapping

encoded\_data[i, 1] = launch\_time\_mapping[row[2]] # Example mapping

return encoded\_data

# Function to create and train the AI model

def train\_model(X\_train, y\_train):

model = Sequential()

model.add(Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=10, batch\_size=32)

return model

# Function to predict anomalies

def predict\_anomalies(model, X\_test):

predictions = model.predict(X\_test)

anomalies = np.where(predictions > 0.5)[0]

return anomalies

# Main execution

if \_\_name\_\_ == '\_\_main\_\_':

data = fetch\_ec2\_data()

preprocessed\_data = preprocess\_data(data)

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(preprocessed\_data, labels, test\_size=0.2)

model = train\_model(X\_train, y\_train)

# Predict anomalies on the testing set

anomalies = predict\_anomalies(model, X\_test)

# Print or log the anomalies

print("Detected anomalies:", anomalies)

# Generate CSV file

csv\_data = pd.DataFrame(data, columns=['Instance ID', 'Instance Type', 'Launch Time', 'Public IP', 'Private IP', 'Security Groups'])

csv\_data.to\_csv('ec2\_data.csv', index=False)

1. **Modify and improve the logic**

import boto3

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow.python.keras.models import Sequential

from tensorflow.python.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Initialize AWS clients

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

s3\_client = boto3.client('s3')

# Define instance type mapping (adjust as needed)

instance\_type\_mapping = {

't2.micro': 0,

't2.small': 1,

'm4.large': 2,

# ... other instance types

}

# Define launch time mapping (adjust as needed)

launch\_time\_mapping = {

'2023-01-01': 0,

'2023-02-01': 1,

'2023-03-01': 2,

# ... other launch times

}

# Define labels (adjust as needed)

labels = [

0, 1, 0, 1, 0, 1, 0, 1, 0, 1,

0, 1, 0, 1, 0, 1, 0, 1, 0, 1,

# ... other labels

]

# Function to fetch and process EC2 instance data

def fetch\_ec2\_data():

response = ec2\_client.describe\_instances()

instances = response['Reservations'][0]['Instances']

# Extract relevant data (adjust as needed)

data = []

for instance in instances:

instance\_id = instance['InstanceId']

instance\_type = instance['InstanceType']

launch\_time = instance['LaunchTime']

public\_ip = instance['PublicIpAddress']

private\_ip = instance['PrivateIpAddress']

security\_groups = instance['SecurityGroups']

data.append([instance\_id, instance\_type, launch\_time, public\_ip, private\_ip, security\_groups])

return data

# Function to preprocess data and create features

def preprocess\_data(data):

# Convert data to NumPy array

data = np.array(data)

# Encode categorical features (adjust as needed)

encoded\_data = np.zeros((len(data), len(data[0])))

for i, row in enumerate(data):

encoded\_data[i, 0] = instance\_type\_mapping[row[1]] # Example mapping

encoded\_data[i, 1] = launch\_time\_mapping[row[2]] # Example mapping

# ... other encodings

return encoded\_data

# Function to create and train the AI model

def train\_model(X\_train, y\_train):

model = Sequential()

model.add(Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=10, batch\_size=32)

return model

# Function to predict anomalies

def predict\_anomalies(model, X\_test):

predictions = model.predict(X\_test)

anomalies = np.where(predictions > 0.5)[0]

return anomalies

# Main execution

if \_\_name\_\_ == '\_\_main\_\_':

data = fetch\_ec2\_data()

preprocessed\_data = preprocess\_data(data)

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(preprocessed\_data, labels, test\_size=0.2)

model = train\_model(X\_train, y\_train)

# Predict anomalies on the testing set

anomalies = predict\_anomalies(model, X\_test)

# Print or log the anomalies

print("Detected anomalies:", anomalies)

# Generate CSV file

csv\_data = pd.DataFrame(data, columns=['Instance ID', 'Instance Type', 'Launch Time', 'Public IP', 'Private IP', 'Security Groups'])

csv\_data.to\_csv('ec2\_data.csv', index=False)

1. **Execute and explain your code**

I apologize for the inconvenience, but I do not currently have an AWS account.