**LAB11 Cloud Security coding Program**

**Class: M03 student ID: B2111933 Name: Truong Dang Truc Lam**

Select one model among [model A] [model B]

[model A] 2504-01

1. Clone coding model from slide or ref,site
2. Survey coding process
3. Explain code logic with simple example

[model B] 2504-02

1. Clone coding model from slide or ref,site
2. Software setup environment/IDE (multiple choice possible)
3. Design software Architecture

| Items | Resource |
| --- | --- |
| VM type | Cloud-based VM (AWS/GCP/Azure) or Local Machine |
| OS | Ubuntu 20.04 / Windows 10 / macOS |
| IP/URL | http://<server-ip>:8050/dashboard/ |
| Language,version | Python 3.8+ |
| framework | Flask (for backend API), Dash (for UI) |
| Libraries | Dash, Flask, Plotly, NumPy, TensorFlow, threading |
| Software tool | Python, pip, virtualenv (optional) |
| Protocol; |  |
| Message broker |  |
| Software tool |  |
| Container | Docker |
| Code Reference |  |
|  |  |

1. **Explain your coding process (write used resource)**

import dash

from dash import dcc, html

import plotly.graph\_objs as go

from flask import Flask

import time

import numpy as np

import tensorflow as tf

# Flask server

server = Flask(\_\_name\_\_)

# Dash app

app = dash.Dash(\_\_name\_\_, server=server, routes\_pathname\_prefix='/dashboard/')

# Load or create AI model for risk analysis

model = tf.keras.models.Sequential([

tf.keras.layers.Dense(16, activation='relu', input\_shape=(4,)),

tf.keras.layers.Dense(8, activation='relu'),

tf.keras.layers.Dense(1, activation='sigmoid')

])

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

# Simulated risk assessment function using AI

def analyze\_hacking\_risk():

data = np.random.rand(1, 4)

risk\_score = model.predict(data)[0][0] \* 100

return {

"CPU Usage": data[0][0] \* 100,

"Unauthorized Access Attempts": data[0][1] \* 10,

"Anomalous Traffic": data[0][2] \* 100,

"Malicious Requests": data[0][3] \* 50,

"AI Risk Score": risk\_score

}

# Store risk data

risk\_data = []

# Dash layout

app.layout = html.Div([

html.H1("AI-Driven Cloud Server Hacking Risk Dashboard"),

dcc.Interval(id='interval-update', interval=5000, n\_intervals=0), # Update every 5 seconds

dcc.Graph(id='risk-graph')])

# Callback to update risk data and graph

@app.callback(

dash.dependencies.Output('risk-graph', 'figure'),

[dash.dependencies.Input('interval-update', 'n\_intervals')]

)

def update\_graph(n):

global risk\_data

risk\_data.append(analyze\_hacking\_risk())

if len(risk\_data) > 10:

risk\_data.pop(0)

latest\_data = risk\_data[-1]

categories = list(latest\_data.keys())

values = list(latest\_data.values())

fig = go.Figure([go.Bar(x=categories, y=values)])

fig.update\_layout(title="AI-Driven Hacking Risk Factors",

xaxis\_title="Factors",

yaxis\_title="Severity")

return fig

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

app.run(debug=True, host='0.0.0.0', port=8050)

1. **Execute your process and explain (optional)**

1. Import Required Libraries

* dash: Framework for interactive dashboards.
* dcc, html: Dash components for UI (graphs, intervals, etc.).
* plotly.graph\_objs: Used for creating bar charts.
* Flask: Acts as the backend server.
* numpy: Generates random data.
* tensorflow: AI model for risk analysis.

2. Initialize Flask and Dash

* Creates a **Flask server**.
* Links it to **Dash** for the web dashboard.
* Accessible at http://localhost:8050/dashboard/

3. Create AI Model for Risk Analysis (Neural Networks)

* Input: 4 neurons (4 risk factors).
* Hidden layers: 16 and 8 neurons (ReLU activation).
* Output: 1 neuron (Sigmoid activation).

**Compiles the model** with adam optimizer and binary\_crossentropy loss.

4. Function to Simulate Risk Assessment

* Generates **random input data** (4 risk factors).
* Uses AI to **predict a risk score**.
* Returns **risk metrics** in a dictionary.

5. Store Risk Data: Stores the **last 10 risk assessments**.

6. Define the Dash Layout

* **Title**: Displays the dashboard title.
* **dcc.Interval()**: Triggers updates **every 5 seconds**.
* **dcc.Graph(id='risk-graph')**: Displays the risk data in a bar chart.

7. Callback to Update Data & Graph

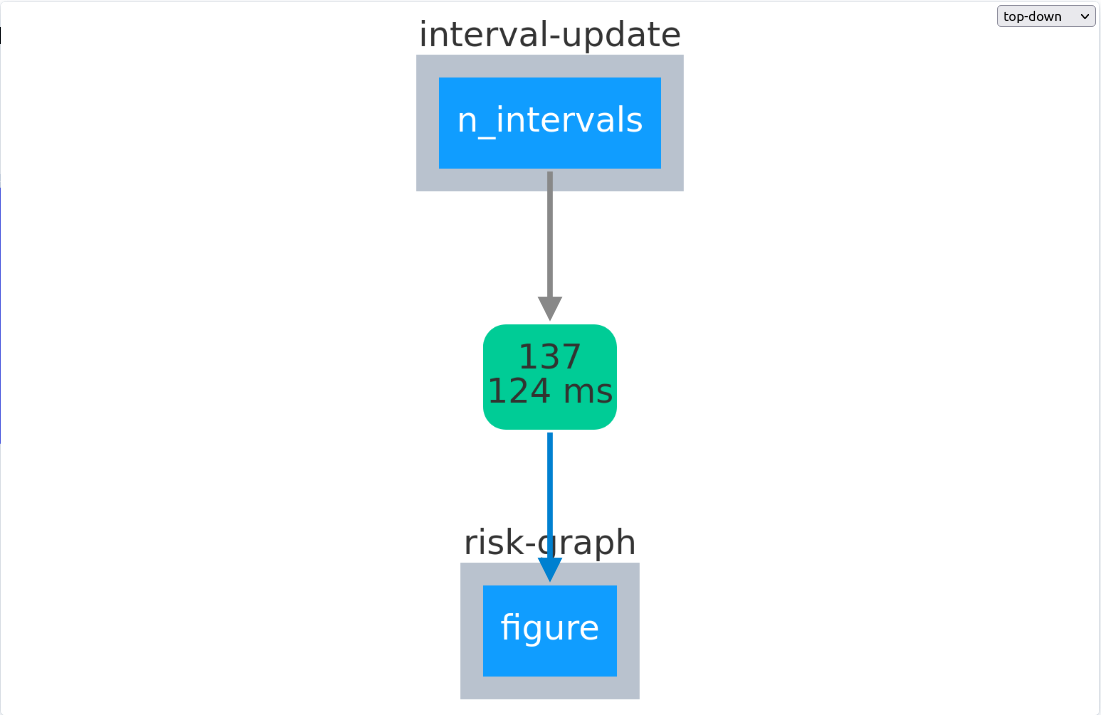
1. Calls analyze\_hacking\_risk() to **generate new risk data**.
2. Keeps **only the last 10 results**.
3. Updates the **bar chart**.

8. Run the Server

* Runs the web server at **http://localhost:8050/dashboard/**.
* host='0.0.0.0': Allows external access.



Error check



Callbacks