**LAB10 Coding Security Visualization**

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

1. **Select one coding model in slide 2503-27, 2503-28 or search from ref,site**

I choose AI driven code for IoT server security status using Dash

**2. Software setup environment/IDE (multiple choice possible)**

Ubuntu, VSCode, Python 3.x

**3. Design software Architecture**

| Items | Resource |  |  |  |
| --- | --- | --- | --- | --- |
| VM type |  |  |  |  |
| OS | Ubuntu 20.04 |  |  |  |
| IP/URL | Localhost |  |  |  |
| Language,version | Python 3.x |  |  |  |
| framework | Dash |  |  |  |
| Libraries | dash, plotly, numpy, pandas |  |  |  |
| Software tool |  |  |  |  |
| Protocol; | MQTT, CoAP |  |  |  |
| Message broker | Mosquitto |  |  |  |
| Software tool |  |  |  |  |
| Container | Docker |  |  |  |
| Code Reference | Python Dash-based AI-Driven IoT Security Dashboard |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

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

With the help from ChatGPT:

import dash

from dash import dcc, html

from dash.dependencies import Input, Output

import plotly.graph\_objs as go

import pandas as pd

import numpy as np

import time

# Initialize the Dash app

app = dash.Dash(\_\_name\_\_)

# Sample IoT server data (simulated)

servers = [f"IoT\_Server\_{i}" for i in range(1, 11)]

def generate\_security\_data():

""" Simulates AI-based security risk scores with additional metrics."""

return pd.DataFrame({

"Server": servers,

"Risk Score": np.random.randint(10, 100, size=len(servers)),

"Anomalies Detected": np.random.randint(0, 15, size=len(servers)),

"CPU Load (%)": np.random.uniform(20, 95, size=len(servers)),

"Network Traffic (MB)": np.random.uniform(100, 1000, size=len(servers))

})

# Layout of the dashboard

app.layout = html.Div([

html.H1("AI-Driven IoT Server Security Dashboard", style={'textAlign': 'center'}),

dcc.Interval(id='interval-component', interval=5000, n\_intervals=0), # Auto refresh every 5 sec

dcc.Graph(id='security-status-graph'),

])

@app.callback(

Output('security-status-graph', 'figure'),

[Input('interval-component', 'n\_intervals')]

)

def update\_graph(n):

df = generate\_security\_data()

print("Generated Data:") # Debug log

print(df)

fig = go.Figure()

fig.add\_trace(go.Bar(x=df['Server'], y=df['Risk Score'], name='Risk Score', marker\_color='red'))

fig.add\_trace(go.Scatter(x=df['Server'], y=df['Anomalies Detected'], name='Anomalies', mode='markers',marker=dict(size=10, color='blue')))

fig.add\_trace(go.Scatter(x=df['Server'], y=df['CPU Load (%)'], name='CPU Load (%)', mode='lines',line=dict(color='green')))

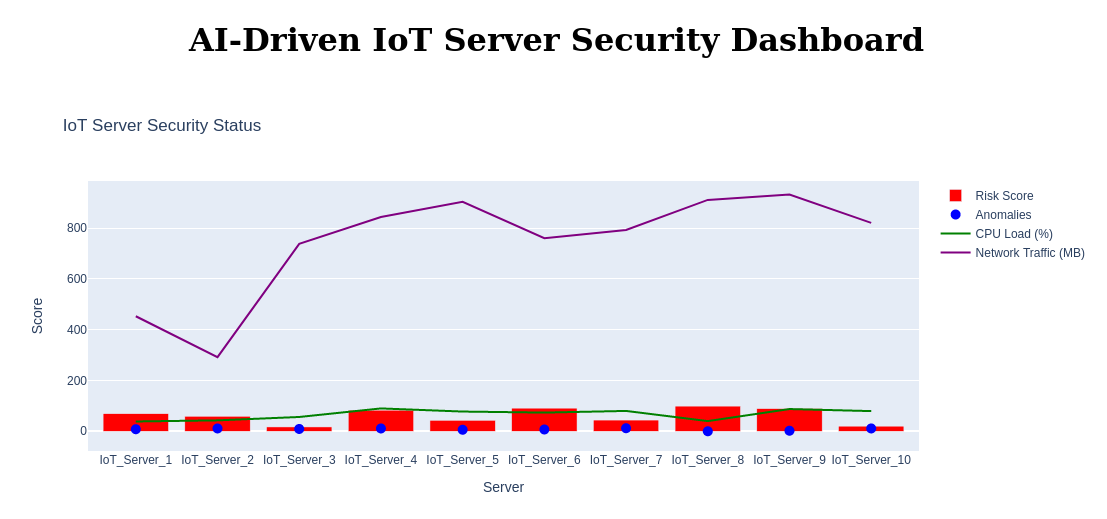
fig.add\_trace(go.Scatter(x=df['Server'], y=df['Network Traffic (MB)'], name='Network Traffic (MB)', mode='lines',line=dict(color='purple')))

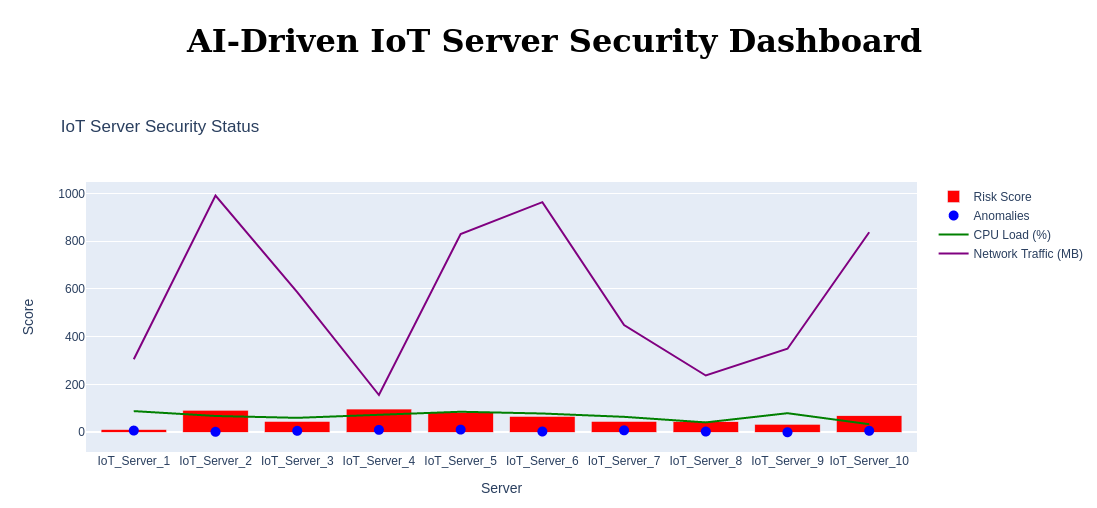
fig.update\_layout(title='IoT Server Security Status', xaxis\_title='Server', yaxis\_title='Score', barmode='group')

return fig

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

app.run\_server(debug=False, port=8060)





The result

**5. Execute your process and explain as far as you do**

This Python script creates a real-time IoT server security monitoring dashboard using Dash and Plotly. The application simulates security risk data for ten IoT servers and updates the visualization every five seconds.

The script begins by importing the necessary libraries, including Dash for creating the web application, Plotly for visualizing data, Pandas and NumPy for handling and generating simulated data, and Dash dependencies for interactivity.

It initializes the Dash app and defines a list of ten IoT server names. The function **generate\_security\_data()** creates random security metrics, including a risk score, detected anomalies, CPU load percentage, and network traffic in megabytes for each server.

The layout of the dashboard consists of a centered title, a **dcc.Interval** component that triggers updates every five seconds, and a **dcc.Graph** element that displays the server security status.

The callback function **update\_graph(n)** generates new security data at each interval and updates the visualization. The figure consists of a bar chart for risk scores, a scatter plot for anomalies, and line charts for CPU load and network traffic.

Finally, the script runs the Dash application on port 8060, continuously updating the dashboard in real-time. Possible improvements include adding more server metrics, implementing machine learning models for threat prediction, integrating alerts for high-risk scores, and storing historical data for trend analysis.