Library for visualization of security in AI code

ChatGPT 의 말:

	Description	Strengths	Usage in Security AI Coding
Matplotlib	A fundamental Python library for creating static, visualizations.	- Highly customizable - Works well with NumPy and Pandas - Supports basic and advanced plotting	- Visualizing intrusion detection data - Displaying time-series logs - Graphing network security trends
Seaborn	Built on Matplotlib, it provides high-level functions for statistical data visualization	- Simplifies complex visualizations - Beautiful default themes - Ideal for statistical data analysis	- Anomaly detection visualization - Correlation heatmaps for security logs - Behavioral analytics in cybersecurity
Plotly	An interactive plotting library that supports complex visualizations, including 3D and real-time data.	- Interactive graphs - Supports web-based visualization - Easy to use with Pandas and NumPy	- Real-time attack pattern visualization - Interactive dashboards for security monitoring - Visualizing attack paths in cloud security
Dash	A framework built on Plotly for building web-based interactive dashboards and applications.	- Supports real-time updates - Ideal for building security AI dashboards - Works well with Flask	- Security monitoring dashboards - IoT threat intelligence dashboards - Cloud security event visualization

NetworkX	A library for analyzing and visualizing graphs and network structures.	– Excellent for graph–based analysis – Optimized for network–based anomaly detection – Handles large–scale security graphs	– Intrusion detection in network traffic – Malware propagation analysis – Mapping cybersecurity attack graphs

Basic of security code

1. Matplotlib

- **Usage**: **Basic plotting (scatter plots, line graphs, bar charts).**
- Installation: `pip install matplotlib`
- Example:

```
python
복사편집
import matplotlib.pyplot as plt

def plot_risk_levels(data):
    plt.figure(figsize=(10, 5))
    plt.hist(data['Anomalous'], bins=2, color=['green', 'red'],
alpha=0.7)
    plt.xticks([0, 1], ["Normal", "Anomalous"])
    plt.title("Risk Analysis Distribution")
    plt.show()
```

2. Seaborn

- **Usage: Enhanced statistical data visualization.**
- Installation: `pip install seaborn`
- Example:

```
python
복사편집
import seaborn as sns

def heatmap_risk(data):
    corr = data.corr()
    sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=0.5)
    plt.title("Feature Correlation Heatmap")
    plt.show()
```

3. Plotly

- **Usage: Interactive charts for security dashboards.**
- Installation: `pip install plotly`
- Example:

python
복사편집

```
import plotly.express as px

def plot_interactive_risks(data):
    fig = px.scatter(data, x='CPU_Usage', y='Network_Traffic',
color='Anomalous',
                     title="Security Risk Analysis",
hover_data=['RAM_Usage', 'Disk_IO'])
    fig.show()
```

4. Dash

- **Usage: Real-time security dashboards for risk visualization.**
- Installation: `pip install dash`
- Example:

python
복사편집

```
import dash
from dash import dcc, html
import plotly.graph_objects as go

app = dash.Dash(__name__)

app.layout = html.Div([
    dcc.Graph(figure=go.Figure(data=[go.Bar(x=["Normal", "Anomalous"],
y=[80, 20])]))
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

5. NetworkX

- **Usage: Visualizing security threats in network graphs.**

- **Installation**: `pip install networkx`
- **Example**:

python
복사편집

```
import networkx as nx

def plot_network_risks():
    G = nx.Graph()
    G.add_edges_from([(1, 2), (1, 3), (2, 4)])
    nx.draw(G, with_labels=True, node_color='red', edge_color='gray')
    plt.show()
```