YANAMALA ABHIRAM| Cybersecurity Case Study | March 2025

Introduction:

Trading platforms manage sensitive user data, such as trading history and personal preferences, making them prime targets for cyberattacks like fraud, phishing, and data breaches. This case study explores how anomaly detection can identify fraudulent behavior in a trading platform and how cybersecurity measures can protect user data. Using tools like Kali Linux and Python, I simulated a trading platform scenario to detect suspicious trades and proposed practical security solutions to enhance user trust and platform integrity.

Problem Statement:

Many trading platforms lack effective mechanisms to detect fraudulent activities, such as unusually high trades or brute-force login attempts. Additionally, insufficient data security leaves users vulnerable to breaches, reducing engagement and trust. This case study addresses these gaps by simulating a platform, detecting suspicious behavior, and suggesting ways to secure user data.

Approach:

I executed this case study in Kali Linux, a cybersecurity-focused operating system, following these steps:

1. **Created a Mock Dataset**: Built a CSV file (trading_data.csv) with sample user trading data, including UserID, TradeAmount, Timestamp, and LoginAttempts.
2. **Detected Suspicious Behavior**: Wrote a Python script (detect_fraud.py) in a virtual environment to flag trades above $5000 and login attempts above 3 as suspicious.
3. **Analyzed Results**: Ran the script and saved the output to identify anomalies.
4. **Proposed Security Measures**: Researched and suggested methods to secure user data based on industry best practices.

Mock Dataset (trading_data.csv):

Below is the sample dataset I created in Kali Linux using **nano trading_data.csv:**

text.

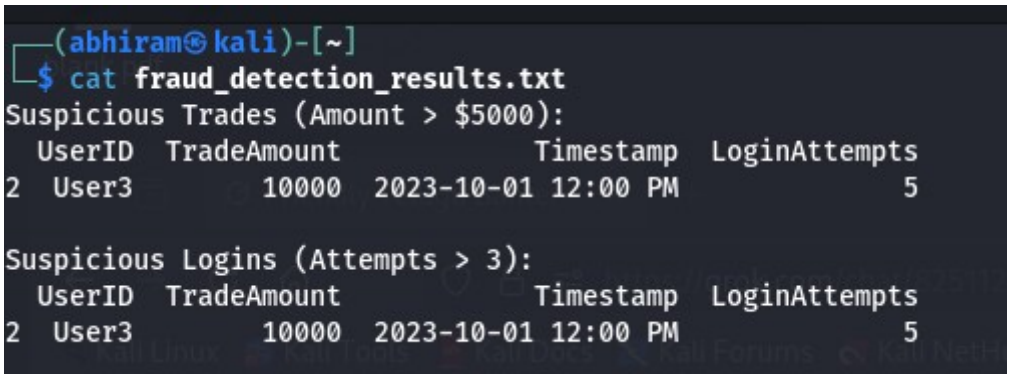
```
  GNU nano 8.3                    trading_data.csv
UserID,TradeAmount,Timestamp,LoginAttempts
User1,100,2023-10-01 10:00 AM,1
User2,500,2023-10-01 11:00 AM,2
User3,10000,2023-10-01 12:00 PM,5
User4,200,2023-10-01 01:00 PM,1
```

Python Script (detect_fraud.py):

```
  GNU nano 8.3
import pandas as pd

# Read the CSV file
data = pd.read_csv('trading_data.csv')

# Define thresholds for suspicious behavior
trade_threshold = 5000  # Trades above $5000 are suspicious
login_threshold = 3     # More than 3 login attempts are suspicious

# Flag suspicious trades and logins
suspicious_trades = data[data['TradeAmount'] > trade_threshold]
suspicious_logins = data[data['LoginAttempts'] > login_threshold]

# Print results
print("Suspicious Trades (Amount > $5000):")
print(suspicious_trades)
print("\nSuspicious Logins (Attempts > 3):")
print(suspicious_logins)
```

```
python3 detect_fraud.py > fraud_detection_results.txt
```

Sample Script Output (fraud_detection_results.txt):



Recommendations:

Based on my research and analysis, I propose the following measures to secure user data on the trading platform:

1. **Data Encryption**: Encrypt sensitive user data (e.g., trading history, personal details) using AES-256 encryption to prevent unauthorized access.
2. **Two-Factor Authentication (2FA)**: Implement 2FA for user logins to add an extra layer of security against account takeovers.
3. **Secure Data Transmission**: Use HTTPS (TLS/SSL) to encrypt data in transit, preventing interception by attackers.
4. **Log Monitoring**: Regularly audit server logs to detect suspicious activity, such as multiple failed login attempts.
5. **Optional Network Analysis**: I used Wireshark in Kali Linux to capture network traffic and observed unencrypted HTTP packets, reinforcing the need for HTTPS.

Use Wireshark to Understand Data Exposure