PS1: Uncovering Threat Intelligence from Cybersecurity Reports

Background:

In today's world of cybersecurity, organizations rely heavily on threat reports to identify potential risks, understand malicious activities, and make informed decisions to defend their networks. These reports, often written in natural language, contain valuable intelligence such as Indicators of Compromise (IoCs), Tactics, Techniques, and Procedures (TTPs), and other critical data points. However, manually extracting this information can be a daunting task due to the unstructured nature of these reports. This is where Natural Language Processing (NLP) comes into play, enabling the automation of threat intelligence extraction and analysis.

To get started, we will focus on one of the most prominent cybersecurity frameworks used to identify and categorize malicious behavior: the MITRE ATT&CK Framework. This framework categorizes adversarial actions into various tactics and techniques that reflect common behaviors of advanced threats.

In this challenge, you will develop a function that extracts key threat intelligence from a natural language threat report.

Your Task:

You will be given a natural language threat report. Your goal is to automatically extract the following key threat intelligence data:

1. **Indicators of Compromise (IoCs):** Extract malicious IP addresses, domains, file hashes, or email addresses.

2. **Tactics, Techniques, and Procedures (TTPs):** Identify the tactics, techniques, and procedures used by threat actors, referring to the MITRE ATT&CK framework.
3. **Threat Actor(s):** Detect the names of any threat actor groups or individuals mentioned in the report.
4. **Malware:** Extract the name, hash and other details of any malware used in the report. The details can be obtained from open source repositories such as VirusTotal. Some of the required details are mentioned in the example below. Participants are welcome to add more additional malware details if they find any (**Bonus marks will be provided for capturing additional details**).
5. **Targeted Entities:** Identify the entities, organizations, or industries that were targeted in the attack.

Note: The dataset can be downloaded using the link below: https://drive.google.com/file/d/1Jt5vFreJrxv7IHwJTuD35xj-rtBG04p3/view?usp=sharing

Input Format:

You will be provided with natural language threat reports. For example, the content of a specific threat report may be represented as the string `report_text` as shown in below example.

Example Input:

```
report_text = '''
 The APT33 group, suspected to be from Iran, has launched a new campaign targeting the energy sector organizations.
 The attack utilizes Shamoon malware, known for its destructive capabilities. The threat actor exploited a vulnerability in the network perimeter to gain initial access.
 The malware was delivered via spear-phishing emails containing a malicious attachment. The malware's behavior was observed communicating with IP address 192.168.1.1 and domain example.com. The attack also involved lateral movement using PowerShell scripts.
 '''
```

Output Format:

You need to output a dictionary containing the extracted threat intelligence data, formatted as follows:

```
{
'IoCs': {
```

```
'IP addresses': ['192.168.1.1'],
'Domains': ['example.com']
},
'TTPs': {
'Tactics': [
        ['TA0001': 'Initial Access'],
        [TA0002': 'Execution'],
        ['TA0008': 'Lateral Movement']
        ],
'Techniques': [
        ['T1566.001': 'Spear Phishing Attachment'],
        ['T1059.001': 'PowerShell']
        ]
},
'Threat Actor(s)': ['APT33'],
'Malware': [
        ['Name': 'Shamoon'],
        ['md5': 'vlfenvnkgn….'],
        ['sha1': 'bvdib…..'],
        ['sha256': 'poherionnj…….'],
        ['ssdeep': 'bgfnh….'],
        ['TLSH': 'bnfdnhg…..'],
        ['tags': 'XYZ']
        ],
'Targeted Entities': ['Energy Sector']
}
```

Deliverables:

The deliverables for this will include the following:

1. Code Implementation

- A Python function or script that accepts natural language threat reports as input and extracts the specified threat intelligence data.
- The function should:
    - Extract Indicators of Compromise (IoCs), including IP addresses, domains, file hashes, or email addresses.
    - Identify Tactics, Techniques, and Procedures (TTPs) aligned with the MITRE ATT&CK Framework.
    - Detect the names of threat actors.

    - Extract malware details from the report and enhance them with information from open-source repositories like VirusTotal.
    - Identify targeted entities, organizations, or industries mentioned in the report.
- The function should output a structured dictionary in the specified format.

2. Example Outputs

- Provide sample inputs (e.g., `report_text` strings) and corresponding outputs (in the required dictionary format) to demonstrate the function's accuracy and comprehensiveness.
- Example outputs should show successful extraction of all required intelligence elements.

3. Documentation

- Clear documentation explaining:
    - How to run the script or function.
    - Dependencies and installation instructions (e.g., Python libraries such as `spacy`, `re`, or others).
    - The logic behind the extraction process for each intelligence element.
    - Any additional steps, such as using APIs (e.g., VirusTotal API) for malware enrichment.
    - Discussion of potential limitations and improvements.

4. Dataset Preprocessing

- Any code or steps for preprocessing the threat report dataset to prepare it for input into the function, if applicable.
- Explanation of how the input dataset was formatted or cleaned to ensure compatibility with the function.

5. Bonus Features (Optional)

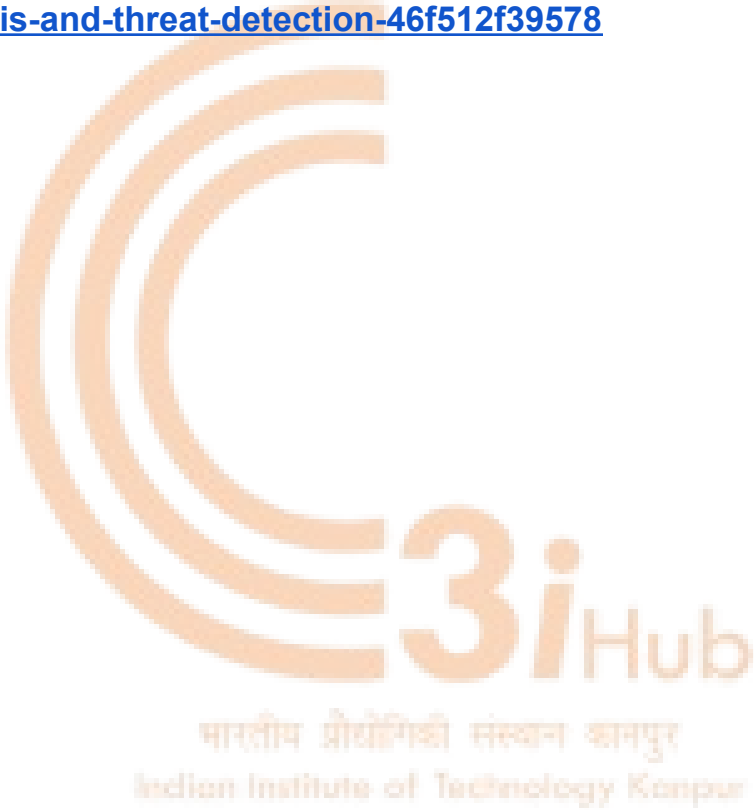
- Enhanced malware details, including additional metadata (e.g., `tags`, `TLSH`, etc.).
- Customizable outputs, allowing the user to specify which fields to extract.

6. Additional Deliverables

- Include an example `README.md` file for clarity and ease of use.

Sources:

1. https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/
2. https://dl.acm.org/doi/10.1145/3696427
3. https://dl.acm.org/doi/10.1145/3579375.3579391
4. https://www.analyticsvidhya.com/blog/2021/06/nlp-application-named-entity-recognition-ner-in-python-with-spacy/
5. https://link.springer.com/article/10.1631/FITEE.2000286
6. https://www.turing.com/kb/a-comprehensive-guide-to-named-entity-recognition
7. https://chamindux.medium.com/virustotal-101-a-beginners-guide-to-file-analysis-and-threat-detection-46f512f39578



PS2:Create Gasless Transaction Forwarder:

- **Description:**
  - Implement a forwarder contract that lets users send ERC-20, and ERC-721 transactions without holding ETH.
- **Requirements:**
  - Thecontract should accept user transactions, and then forward them on-chain.
  - Build a simple interface for users to enter transaction details and send gasless transfers.
- **Score Metric:**
  - Forwarder Smart Contract
  - Working UI
  - UIwith extra features
  - All security checks passed
  - Solidity Shield audit score for codes .

3i Hub
भारतीय प्रौद्योगिकी संस्थान कानपुर
Indian Institute of Technology Kanpur

PS3:AI-Driven Insider Threat Detection

Challenge Overview

Insider threats are among the most challenging cybersecurity risks, where individuals within an organization — such as employees, contractors, or partners — intentionally or unintentionally compromise sensitive information. These threats can manifest as data leaks through emails, unauthorized access to confidential files, or even sabotage of critical systems.

The challenge is to develop a powerful AI-driven solution capable of identifying these risks by analyzing structured activity data. Your solution will include a detection mechanism for identifying anomalies, a dashboard for visualizing insights, and a module for generating structured datasets from raw logs like network traffic, system activities, or application usage. This is your chance to tackle a critical real-world problem and contribute to making organizations safer from within.

Objectives

1. Detection Mechanism:
    a. Build a robust system to detect insider threats using structured datasets.
    b. Focus on identifying suspicious behaviors while minimizing false positives and negatives.
2. Visualization and Insight Generation:
    a. Develop a minimal dashboard to display real-time alerts, behavioral trends, and summary statistics.
    b. Ensure the dashboard is intuitive and actionable for security teams.
3. Dataset Generation Module:
    a. Design a pipeline to process raw logs (e.g., network traffic, application logs, system activity logs) into structured datasets.
    b. Ensure the output datasets are compatible with the detection mechanism.

Evaluation Criteria

1. Detection Accuracy:
    a. Performance metrics such as precision, recall, and F1 score.
    b. Ability to minimize false positives and negatives.

2. Dashboard Usability:
    a. Clarity, interactivity, and utility of the visualizations.
3. Dataset Generation:
    a. Efficiency and effectiveness in transforming raw logs into structured datasets.
    b. Versatility in handling multiple log types.
4. Scalability and Performance:
    a. Handling large-scale organizational data efficiently.
5. Privacy and Security:
    a. Implementation of techniques such as anonymization and encryption to protect sensitive data.

Implementation Guidelines and Suggestions

General Implementation Rules:

- Core functionalities must be implemented from scratch.
- Copying code from platforms like Kaggle, GitHub, or similar is strictly prohibited.
- Participants must document the implementation process, including architecture, tools, and custom algorithms used.

Dataset Usage:

- Teams can use the Insider Threat Test Dataset from Carnegie Mellon University:
    - This dataset includes synthetic user activity logs, such as email communication, logins, and file access.
    - Use preprocessing to clean and format the data for analysis. Advanced tools like Pandas, Apache Spark, or Dask can be used to handle and prepare the dataset.
    - Employ machine learning techniques to train models for anomaly detection, leveraging features like user actions, timestamps, and access patterns.

Advanced Models and Techniques:

1. **Detection Mechanism:**
    a. Utilize neural networks such as Autoencoders or LSTMs (Long Short-Term Memory networks) for sequential anomaly detection.
    b. Explore Graph Neural Networks (GNNs) to model relationships between users and their activities for deeper insights.

c. Consider clustering-based methods like DBSCAN or HDBSCAN for unsupervised anomaly detection.
d. Use ensemble methods, e.g., Random Forests or Gradient Boosting Machines (XGBoost), for interpretable detection mechanisms.

2. Visualization and Dashboard:
   a. Build an interactive dashboard using Streamlit, Dash, or Flask for backend and React.js or Vue.js for frontend.
   b. Use libraries like Plotly, Seaborn, or D3.js to create advanced visualizations, such as heatmaps or temporal activity trends.
3. Dataset Generation Module:
   a. Create synthetic datasets from raw logs using tools like Apache Kafka for data streaming and Logstash for log parsing.
   b. Develop pipelines to process data with frameworks like Apache Airflow, Luigi, or Prefect.
   c. Leverage natural language processing (NLP) techniques, such as BERT, to analyze textual logs or email content.

Privacy Considerations:

- Ensure privacy by implementing anonymization techniques like hashing sensitive information.
- Use encryption (e.g., AES-256) for all data storage and transfer.

Deployment Suggestions (Optional):

- Test the scalability of your solution on cloud platforms like AWS, Google Cloud, or Azure.
- Containerize your application using Docker

Deliverables

Participants are required to submit the following:

1. **Working Prototype:**
   a. A functional implementation of the solution, including:
      i. Detection Mechanism: The core module for identifying insider threats.
      ii. Dashboard: A minimal interface for visualizing insights and alerts.

iii. Dataset Generation Module: A pipeline or process to convert raw logs into structured datasets (if implemented).

b. Integration: The different components (detection mechanism, dashboard, dataset generation module) should be seamlessly integrated into a unified system.

2. Cloud Deployment (Highly Encouraged):
   a. If possible, host or deploy the solution on a cloud platform (e.g., AWS, GCP, Azure).
   b. Provide the deployed application's URL or access credentials for evaluation.
   c. Ensure the deployment includes all components (detection, dashboard, and dataset generation).
3. Source Code:
   a. Submit the complete source code for the solution.
   b. Ensure the code is clean, modular, and well-commented.
   c. Upload the code to a public or private repository (e.g., GitHub, GitLab) and share access.
4. Documentation:
   a. Technical Documentation: Explain the architecture, algorithms/models used, tools, and technologies implemented.
   b. Setup Guide: Provide clear instructions for running the prototype locally and, if applicable, on the cloud.
   c. Privacy Measures: Detail the privacy-preserving mechanisms implemented in the solution.
5. Demo Video:
   a. A 5–10 minute video walkthrough of the solution.
   b. Highlight key functionalities, such as:
      i. Threat detection in action.
      ii. Dashboard insights and alerts.
      iii. Dataset generation process (if applicable).
      iv. Cloud deployment demonstration, if hosted.
6. Evaluation Metrics:
   a. Include results from testing the detection mechanism.

CSI Hub
भारतीय प्रौद्योगिकी संस्थान कानपुर
Indian Institute of Technology Kanpur

b. Provide metrics like accuracy, precision, recall, F1 score, and a brief explanation of the model's performance.

Submission Format

- **Platform**: Submissions should be made via college emails.
- File Structure:
    - A compressed .zip file or repository link containing:
        - Source code folder.
        - Documentation files (e.g., README.md, TechnicalDocumentation.pdf).
        - Dataset or processed data (if applicable).
        - Demo video link (can be uploaded to platforms like YouTube or Vimeo with restricted access or google drive link also fine).

3i Hub
भारतीय प्रौद्योगिकी संस्थान कानपुर
Indian Institute of Technology Kanpur