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