Threat Intelligence Extractor Documentation

Overview

The `threat_intelligence_extractor.py` script is designed to automate the extraction of threat intelligence from cybersecurity reports. This tool identifies and organizes Indicators of Compromise (IoCs), Tactics, Techniques, and Procedures (TTPs), as well as additional key intelligence elements, making it a powerful resource for security analysts.

Features

Indicators of Compromise (IoCs)

- **IP Addresses**
- **Domains**
- **Emails**
- **File Hashes (MD5, SHA1, SHA256)**

⚖ Tactics, Techniques, and Procedures (TTPs)

- Maps to the **MITRE ATT&CK** framework.
- Extracts and correlates tactics and techniques.

Entities Identified

- **Threat Actors**
- **Targeted Organizations**
- **Malware Names and Details**

⚙ Prerequisites

Dependencies

Ensure the following Python libraries are installed: - `spacy` (For Named Entity Recognition) - `re` (Regex for pattern matching) - `json` (Output formatting) - `fuzzywuzzy` (Optional, for advanced text matching) - `requests` (For API calls)

Install dependencies using pip:

```
pip install spacy fuzzywuzzy requests
```

⚛ MITRE ATT&CK Mapping

The script uses a predefined JSON mapping for tactics and techniques. This file can be updated externally to include new TTPs.

How to Use the Script

▶ Running the Script

1. Save the script as `threat_intelligence_extractor.py`.
2. Provide a threat report as input text.
3. Execute the script using:

   ```
   python threat_intelligence_extractor.py
   ```

Example Input

Sample threat report:

```
A new campaign by the Lazarus Group has been detected targeting
defense contractors.
The attackers delivered a malicious payload, identified as
WannaCry ransomware, through phishing emails containing Excel
macros.
The malware communicated with command-and-control servers at
192.168.50.5 and 10.0.0.7, as well as malicious domains like
steal-data.org and hacker-base.net.
Hashes of the malicious files include
e3c5695c4a8c09d8f4e7e453f7f3d40a and
af5caaaf89f1ea3b48dc3561e3e82c28.
```

Output

The extracted intelligence is displayed in JSON format:


```
{
    "IoCs": {
        "IP addresses": ["192.168.50.5", "10.0.0.7"],
        "Domains": ["steal-data.org", "hacker-base.net"],
        "Emails": [],
        "File Hashes": [
            "e3c5695c4a8c09d8f4e7e453f7f3d40a",
            "af5caaaf89f1ea3b48dc3561e3e82c28"
        ]
```

```
    },
    "TTPs": {
        "Tactics": [],
        "Techniques": []
    },
    "Threat Actor(s)": ["Lazarus Group"],
    "Malware": [
        {
            "Name": "WannaCry",
            "Details": "Details can be enriched via an external
        API"
        }
    ],
    "Targeted Entities": ["defense contractors"]
}
```

Logic Behind the Extraction Process

IoC Extraction

- Uses **Regex Patterns** for detecting IP addresses, domains, emails, and file hashes.
- Deduplicates and organizes results into categories.

♂ Named Entity Recognition (NER)

- Employs the `spaCy` library to identify organizations, people, and other relevant entities.

MITRE ATT&CK Mapping

- Matches TTPs based on predefined phrases and keywords within the report text.

⚛ Malware Identification

- Leverages simple regex to detect potential malware names in the report.
- Enriches details via external APIs (e.g., VirusTotal).

Customization

Updating MITRE ATT&CK Mapping

The `MITRE_ATTACK_MAPPING` dictionary can be extended with new tactics and techniques as required.

⚖ Customizable Outputs

The script allows users to specify which fields to extract by modifying the `extracted_data` structure.

API Integration (Optional)

Integrate with external services like **VirusTotal** or **Shodan** to enrich IoCs:

```
def enrich_ioc_with_virustotal(ioc):
    url = f"https://www.virustotal.com/api/v3/ip_addresses/{ioc}"
    headers = {"x-apikey": "YOUR_API_KEY"}
    response = requests.get(url, headers=headers)
    return response.json()
```

Potential Limitations

- **False Positives:** Regex patterns may occasionally extract irrelevant data.
- **NER Accuracy:** The `spaCy` model might misclassify some entities.
- **Static Mapping:** The MITRE ATT&CK mapping requires manual updates to stay current.

Suggestions for Improvement

- Implement **Machine Learning Models** for enhanced entity recognition.
- Add support for **Additional IoC Types**, such as URLs and registry keys.
- Incorporate **Dynamic Mapping Updates** via an online API.

Dataset Preprocessing (Optional)

- Clean and format the input dataset to ensure compatibility with the script.
- Remove irrelevant text or non-standard characters.

Example Preprocessing Code

```
def preprocess_report(report_text):
    # Remove special characters
    return re.sub(r'[^\w\s]', '', report_text)
```

Conclusion

The `threat_intelligence_extractor.py` script is a robust and customizable tool for automating the extraction of threat intelligence from unstructured reports. By leveraging regex, NER, and MITRE ATT&CK mappings, it provides comprehensive insights to security analysts.

Support

For questions or feature requests, please contact the developer.