Threat Intelligence Platform

Project Report

Akshaj Kumar M

Objective

This report covers the tasks completed during Week 1 of the cybersecurity project for building a Threat Intelligence Platform. It focuses on setting up the development environment, integrating data sources, preprocessing threat intelligence data, ingesting data into Elasticsearch, and visualizing it in Kibana.

1 Environment Setup

The goal for this task was to install and configure the Elastic Stack (Elasticsearch, Logstash, and Kibana) on Ubuntu and test the environment for data ingestion and visualization capabilities. The following steps were performed:

Installing Elasticsearch

To install and configure Elasticsearch, the following commands were executed:

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o
echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts
sudo apt update && sudo apt install elasticsearch
sudo systemctl enable elasticsearch
sudo systemctl start elasticsearch
curl -k -u "elastic:password" -X GET "http://localhost:9200/"
```

These commands install Elasticsearch, start the service, enable it to start on boot, and verify the installation using a ‘curl‘ request.

Installing Kibana

To install and verify Kibana, the following commands were used:

```
sudo apt install kibana
sudo systemctl start kibana
sudo systemctl enable kibana
```

After installation, Kibana was accessed via the browser at http://localhost:5601.

Installing Logstash

Logstash was installed using the following command:

```
sudo apt install logstash
sudo systemctl start logstash
```

A simple test configuration file, ‘test.conf‘, was created to verify Logstash functionality. The contents of the file were as follows:

```
input { stdin { } }
output { stdout { codec => rubydebug } }
```

The following command was used to run Logstash:

```
sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/test.conf
```

2 Data Source Integration

The objective of this task was to connect to external APIs (AlienVault OTX, VirusTotal, and Shodan) and fetch threat intelligence data.

AlienVault OTX Integration

A Python script was created to fetch threat intelligence data from AlienVault OTX and save it in a CSV file. The key components of the script are as follows:

- **Imports:** The script uses ‘requests‘, ‘json‘, and ‘pandas‘ libraries.
- **API Key:** The script requires an API key for authentication.
- **Fetching Data:** The ‘fetch$_o tx_d ata‘ $functionsendsaGETrequesttotheOTXAPIandextractsrelevantda$
  The script is as follows:
- ```
  import requests
  import json
  import pandas as pd

  API_KEY = 'your_alienvault_api_key'
  headers = {"X-OTX-API-KEY": API_KEY}
  url = "https://otx.alienvault.com/api/v1/pulses/subscribed"

  def fetch_otx_data():
      try:
          response = requests.get(url, headers=headers)
          if response.status_code == 200:
              data = response.json()
              return data["results"]
          else:
              print(f"Error: {response.status_code}, {response.text}")
              return []
      except Exception as e:
          print(f"Exception occurred: {e}")
          return []

  def save_to_csv(data):
      df = pd.DataFrame(data)
      df.to_csv("alienvault_data.csv", index=False)
      print("Data saved to alienvault_data.csv")
  ```
  ```

```
if __name__ == "__main__":
    print("Fetching data from AlienVault OTX...")
    otx_data = fetch_otx_data()
    if otx_data:
        save_to_csv(otx_data)
    else:
        print("No data retrieved.")
```

VirusTotal Integration

A Python script was created to query VirusTotal for specific indicators (IP addresses). The script processes a list of indicators, fetches data for each, and saves the results into a CSV file. The code snippet is as follows:

```
import requests
import pandas as pd

API_KEY = "your_virustotal_api_key"

def fetch_virustotal_data(indicator):
    url = f"https://www.virustotal.com/api/v3/ip_addresses/{indicator}"
    headers = {"x-apikey": API_KEY}
    try:
        response = requests.get(url, headers=headers)
        if response.status_code == 200:
            return response.json()
        else:
            print(f"Error: {response.status_code}")
            return None
    except Exception as e:
        print(f"Exception occurred: {e}")
        return None

def save_to_csv(data, filename="virustotal_data.csv"):
    df = pd.DataFrame(data)
    df.to_csv(filename, index=False)
    print(f"Data saved to {filename}")

if __name__ == "__main__":
    indicators = ["8.8.8.8", "1.1.1.1"]
    results = []
    for indicator in indicators:
        print(f"Fetching data for: {indicator}")
        result = fetch_virustotal_data(indicator)
        if result:
            results.append(result)
    save_to_csv(results)
```

Shodan Integration

Similarly, a script for Shodan was created to query specific IP addresses and retrieve detailed information. The following code fetches data for an IP address and saves it in a JSON file:

```
import json
import shodan

API_KEY = "your_shodan_api_key"
api = shodan.Shodan(API_KEY)

def fetch_shodan_data(ip):
    try:
        ip_info = api.host(ip)
        return ip_info
    except shodan.APIError as e:
        print(f"Error: {e}")
        return None

def save_shodan_data(ip_address, filename="shodan_data.json"):
    data = fetch_shodan_data(ip_address)
    if data:
        with open(filename, "w") as json_file:
            json.dump(data, json_file, indent=4)
        print(f"Shodan data for {ip_address} saved to {filename}")
    else:
        print(f"No data found for {ip_address}")

if __name__ == "__main__":
    ip_address = "8.8.8.8"
    save_shodan_data(ip_address)
```

3 Data Preprocessing

The data preprocessing task involved cleaning and preparing the raw data from AlienVault, VirusTotal, and Shodan for further analysis. The following steps were performed:

AlienVault Data Preprocessing

The raw CSV data from AlienVault was cleaned using the following steps:

```
import pandas as pd

df = pd.read_csv("alienvault_data.csv")
df.dropna(inplace=True)
if 'date' in df.columns:
    df['date'] = pd.to_datetime(df['date'])
df.to_csv("alienvault_cleaned.csv", index=False)
```

Shodan Data Preprocessing

For Shodan data, a JSON file was loaded, normalized, and cleaned:

```
import pandas as pd
import json

with open("shodan_data.json", "r") as file:
    shodan_data = json.load(file)

df = pd.json_normalize(shodan_data)
if 'ip' in df.columns:
    df['ip'] = df['ip'].astype(str).str.strip()

if 'timestamp' in df.columns:
    df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')

df.to_csv("shodan_cleaned.csv", index=False)
df.to_json("shodan_cleaned.json", orient="records", indent=4)
```

VirusTotal Data Preprocessing

For VirusTotal data, the CSV file was cleaned by removing irrelevant columns, renaming fields for consistency, and standardizing date formats:

```
import pandas as pd

df = pd.read_csv("virustotal_data.csv")
df.drop(columns=["unnecessary_field1", "unnecessary_field2"], inplace=True, errors='i
df.rename(columns={"old_field_name1": "new_field_name1"}, inplace=True)

if 'date' in df.columns:
    df['date'] = pd.to_datetime(df['date'], errors='coerce')

df.drop_duplicates(inplace=True)
df.to_csv("VirusTotal_cleaned.csv", index=False)
```

4 Data Ingestion to Elasticsearch

Objective

Use Logstash to ingest the preprocessed data into Elasticsearch.

Logstash Configuration for Data Ingestion

Create a Logstash configuration file to read the preprocessed data files, process them, and send them to Elasticsearch.

Logstash Config (threat-intel.conf):

```
input {
    file {
        path => "/home/cyber/threat-intel/alienvault_data.csv"
        start_position => "beginning"
        sincedb_path => "/dev/null"
    }
    file {
        path => "/home/cyber/threat-intel/virustotal_data.csv"
        start_position => "beginning"
        sincedb_path => "/dev/null"
    }
    file {
        path => "/home/cyber/threat-intel/shodan_data.json"
        start_position => "beginning"
        codec => "json"
        sincedb_path => "/dev/null"
    }
}

filter {
    csv {
        source => "message"
        columns => ["id", "name", "description", "indicator", "type", "date"]
        separator => ","
    }
    date {
        match => ["date", "ISO8601"]
        target => "@timestamp"
    }
}

output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "threat-intelligence"
    }
    stdout {
        codec => rubydebug
    }
}
```

5 Data Visualization in Kibana

Objective

Create visualizations and dashboards in Kibana.

Steps

1. Go to Kibana's web interface at http://localhost:5601.
2. In Kibana, navigate to Stack Management and click Index Patterns.
3. Click Create Index Pattern, select 'threat-intelligence', and create the index pattern.
4. Go to the Discover tab to explore the ingested data.
5. Create visualizations and dashboards based on the data fields.

Tools and Technologies Used

- Ubuntu: Operating system for the development environment.
- Elasticsearch: For indexing and searching data.
- Kibana: For visualizing and analyzing data.
- Logstash: For data ingestion and transformation.
- Python: For interacting with APIs and processing data.
- Shodan Library: For querying the Shodan API.
- AlienVault OTX API: For fetching threat intelligence.
- VirusTotal API: For analyzing specific indicators.