Python Scripting Project Final Design Document

by

Joseph Howard

June 25, 2024

**Problem Description**

An important element of information security is checking the reputation of IP addresses connected to by systems in a network. Services such as VirusTotal and AbuseIPDB collect and provide data related to the reputation of IP addresses. Security analysts make use of these services to check whether specific IP addresses are known to be malicious or engaged in suspicious activity. This assists security analysts and engineers with reducing or eliminating false positives and detecting, preventing, and responding to threats.

Checking IP reputations takes time, and if security analysts or engineers work in an environment with high traffic volume and a high number of alerts, manually checking the reputation of all IP addresses included in alerts can easily be overwhelming or impossible. Using automation to check IP address reputations and return important data can save time and help reduce the burden of processing high numbers of IP addresses and alerts.

**Solution Overview and Major Features**

As a solution to this problem, this project automates the checking of IP addresses against AbuseIPDB, a well-known internet repository for reporting and identifying IP addresses that have been associated with malicious activity (AbuseIPDB, n.d.-a). AbuseIPDB provides access to its services via an application programming interface (API) (AbuseIPDB, n.d.-b). An API is a set of rules or protocols by which software applications can communicate directly with one another to exchange data, features, or functionality (Goodwin, 2024).

At a broad level, the project includes the following elements:

1. Network Traffic Capture

The script captures network traffic on a specified network interface (eth0) for a set duration (30 seconds). The purpose of the capture is to gather network data for the IP reputation check. The script saves the network capture in a pcap file (packet\_capture.pcap). Pcap, which stands for packet capture, is a file that contains data packets captured on a computer network (Hjelmvik, 2022).

1. IP Address Extraction from PCAP File

The script extracts IP addresses from the pcap file and appends them to an existing file (ip\_addresses.txt) which already contains 10 IP addresses with negative reputations according to AbuseIPDB. The file is prepopulated with suspicious addresses in order to avoid connecting to potentially malicious sites while also providing a means for including suspicious sites in the final results. The script analyzes the pcap file, identifies unique, public IP addresses, and appends them to ip\_addresses.txt. Identifying unique IP addresses prevents duplicate entries in the ip\_addresses.txt file to avoid redundant checks against AbuseIPDB.

1. IP Reputation Checking

The script then selects the first 20 IP addresses from the ip\_addresses.txt file for checking against AbuseIPDB. Only 20 were selected to avoid exceeding the AbuseIPDB limit during testing and to reduce the time required for the AbuseIPDB query (so that the screen record video of the script run would not be excessively large). In addition to reading the first 20 IP addresses from ip\_addresses.txt, the script also reads the AbuseIPDB API key from the file jh\_abuseipdb\_key.txt. The key is read from a separate file to avoid hard coding the key into the script, which would be an insecure method.

1. Formatting and Ranking Results

The AbuseIPDB API call returns data in JSON format. JSON, which stands for JavaScript Object Notation, is based on a subset of the JavaScript language and is a lightweight data-interchange format which is relatively easy for humans to read and write and for machines to parse and generate (JSON.org, n.d.). For the sake of brevity and in order to provide the most relevant or interesting data, the script selects 8 categories of data to be included in the results for each IP address. This data is reformatted to be more easily human-readable. Results are ordered based on the AbuseIPDB confidence score with the most suspicious IP addresses first and the remaining results following in descending order. This allows an analyst to quickly identify the most suspicious IP addresses.

1. Results Output

The script writes the results to a file (abuseipdb\_reports.txt) and also prints the results to the screen. Writing to a file provides persistent storage for later retrieval while printing to the screen provides immediate viewing if desired.

1. Periodic Print Statements

The script includes a number of print statements which inform the user about the progress of the script such as the beginning and ending of packet capture, processing of the pcap file, appending IP addresses to the existing file, checking IP addresses against AbuseIPDB, and printing the sorted results to the screen.

1. Error Handling

The script includes error handling elements such as the following:

1. Error during packet capture: catches exceptions that occur during the packet capture process and prints an error message before exiting the script.
2. Error reading existing IP addresses: catches “FileNotFoundError” and other exceptions when reading the existing IP addresses from the file, printing an error message before exiting the script.
3. Error extracting IP addresses: catches exceptions during the extraction and appending of IP addresses from the capture file to the output file, printing an error message before exiting the script.
4. Error reading IP addresses from file: catches exceptions when reading IP addresses from the file, printing an error message before exiting the script.
5. Network error while checking IP addresses against AbuseIPDB: catches “requests.exceptions.RequestException” during the API call to AbuseIPDB, printing a network error message before exiting the script.
6. Error saving results to file: catches exceptions when saving results to the output file, printing an error message before exiting the script.
7. Error reading API key from file: catches exceptions when reading the API key from the file, printing an error message before exiting the script.

**Modules Used**

The script employs five Python modules:

1. Pyshark

This module is generally used for network packet capture and analysis (KimiNewt, n.d.). In the context of this project, pyshark captures network traffic on the specified interface (eth0) and saves it to a pcap file. It also helps read and analyze the captured packets to extract destination IP addresses.

1. Requests

This module is used for making HTTP requests (Ronquillo, 2024). In the context of this project, it sends HTTP GET requests to the AbuseIPDB API to check the reputation of extracted IP addresses.

1. JSON

This module enables encoding and decoding of JSON data (Lofaro, n.d.). In the context of this project, the module parses the JSON responses received from the AbuseIPDB API, extracting selected information.

1. OS

This module provides for interacting with the operating system, including file and directory creation, and management of files and directories (W3schools, n.d.). In the context of this project, the OS module is used for file handling operations, such as checking if a file exists, reading from and writing to files, and managing file paths.

1. ipaddress

This module allows inspecting and manipulating IP addresses in Python (Coghlan & Moody, n.d.). In the context of this project, the module filters out private, broadcast, multicast, unspecified, reserved, and loopback addresses.

**Script Functions**

The script includes the following functions:

1. start\_packet\_capture(interface, output\_file, duration): captures network traffic on the specified network interface (eth0) for the given duration (30 seconds) and saves it to a pcap file (packet\_capture.pcap).
2. is\_valid\_ip(ip): checks whether an IP address is valid for inclusion in the output. The function excludes private IP addresses, broadcast addresses, the virtual machine IP address, and other non-public IP types.
3. read\_existing\_ips(file\_path): Reads the existing IP addresses from the specified file (ip\_addresses.txt) and returns unique IP addresses to avoid duplication.
4. extract\_ips\_from\_capture(capture\_file, output\_file): Extracts destination IP addresses from the captured network packets, filters out invalid IPs, and appends only unique IP addresses to the specified output file (ip\_addresses.txt).
5. read\_ips\_from\_file(file\_path): Reads IP addresses from the specified file (ip\_addresses.txt). Returns a list of these IP addresses for further processing.
6. check\_ip\_with\_abuseipdb(ip, api\_key): Queries the AbuseIPDB API for the reputation of the given IP address using the provided API key. Returns the API response in JSON format.
7. save\_results(results, output\_file): Saves the reputation check results to the specified output file (abuseipdb\_reports.txt) and prints them to the screen. Results are sorted by the AbuseIPDB abuse confidence score in descending order.
8. main(): Facilitates the overall workflow of the script. Initiates packet capture, extracts and filters IP addresses, reads the API key, checks IP reputations with AbuseIPDB, and saves and prints the results.

**Data Needs and Handling**

This project involves handling various types of data from different sources such as the following:

1. Pcap File Containing Network Packets – As noted above, pcap files contain data packets captured on a network. Pcap files are in binary format.
2. IP Addresses – Either prepopulated in the ip\_addresses.txt or extracted from the pcap file and written to ip\_addresses.txt. In the ip\_addresses.txt file these exist as text strings.
3. AbuseIPDB API Key – A string key read from jh\_abuseipdb\_key.txt and used in the API call to AbuseIPDB.
4. API Responses – JSON formatted data returned from the AbuseIPDB API calls. The data reported for each IP address includes fourteen categories including information such as the IP address, the IP version, geographic data, domain and hostnames, the Internet Service Provider (ISP), the number of distinct users reporting the IP address, the total number of reports of the IP, and the date and time of the most recent report.
5. Reformatted Reputation Results – The script reformats the JSON data returned from AbuseIPDB and stores it in a text file. The following categories were selected for inclusion: IP address, country code, domain, ISP, AbuseIPDB Abuse Confidence Score, number of distinct users reporting the IP address, the total number of reports of the IP address, and the time of the most recent report.

**Network Functionality**

The project includes two main networking functions:

1. Packet Capture: As noted above, pyshark is used to capture live network traffic on a specified network interface (eth0). The captured packets are saved to a pcap file (packet\_capture.pcap). This enables the script to gather real-time network data for analysis.
2. API Requests: The script uses the requests module to send HTTP GET requests to the AbuseIPDB API. It queries the reputation of extracted IP addresses, retrieves data in JSON format, and handles network communication with the AbuseIPDB service.

**File Interaction**

The project includes file interactions such as the following:

1. Reading and Writing PCAP Files
2. Reading Existing IP Addresses: The script reads existing IP addresses from ip\_addresses.txt to ensure uniqueness when appending new IP addresses.
3. Appending Unique IP Addresses to File: Appends only unique, valid destination IP addresses to ip\_addresses.txt.
4. Reading IP Addresses for Processing: The function read\_ips\_from\_file(file\_path) reads IP addresses from ip\_addresses.txt for querying against AbuseIPDB.
5. Reading API Key from File: The main() function reads the API key from jh\_abuseipdb\_key.txt to authenticate API requests to AbuseIPDB.
6. Writing API Query Results to a File: The function save\_results(results, output\_file) writes the sorted reputation check results to abuseipdb\_reports.txt.

**Conclusion**

In conclusion, this Python project automatse the process of checking IP address reputations by querying the AbuseIPDB database via an API call. By leveraging automation, the project addresses the critical need for efficient and accurate processing of network traffic data. This will assist security analysts with identifying potential threats and reducing manual workload. As implemented for this project, the solution includes setting up a virtual network environment, capturing network traffic, and utilizing Python scripts for data extraction and API queries. This project will contribute to improved threat detection and response and demonstrate the potential of automation in enhancing cybersecurity practices.

**References**

AbuseIPDB. (n.d.-a) About AbuseIPDB. <https://www.abuseipdb.com/about.html>

AbuseIPDB. (n.d.-b) Frequently asked questions – AbuseIPDB.

<https://www.abuseipdb.com/faq.html>

AbuseIPDB. (n.d.-c). APIv2 documentation. [https://docs.abuseipdb.com/?python#check-](https://docs.abuseipdb.com/?python" \l "check-endpoint)

[endpoint](https://docs.abuseipdb.com/?python" \l "check-endpoint)

Coghlan, N. & Moody, P. (n.d.). An introduction to the ipaddress module. Python.org.

<https://docs.python.org/3/howto/ipaddress.html>

Goodwin, M. (2024, April 9). What is an API (application programming interface)? IBM.

<https://www.ibm.com/topics/api>

Hjelmvik, E. (2022, October 27). What is a PCAP file? NETRESEC.

<https://www.netresec.com/?page=Blog&month=2022-10&post=What-is-a-PCAP-file>

JSON.org. (n.d.). Introducing JSON. <https://www.json.org/json-en.html>

KimiNewt. (n.d.). PyShark: Python packet parser using wireshark’s tshark.

<https://kiminewt.github.io/pyshark/>

Lofaro, L. (n.d.). Working with JSON data in Python. Real Python.

<https://realpython.com/python-json/>

Ronquillo, A. (2024, February 28). Python’s requests library (guide). Real Python.

<https://realpython.com/python-requests/>

W3schools. (n.d.). Python os module. <https://www.w3schools.com/python/module_os.asp>