PCAP File Analysis and Threat Detection Tools

Coursework Title: Portfolio on Identifying Security Threats in PCAP Files

Report by R Rafi

Contents

Introduction

This project offers a complete set of Python-based tools for cybersecurity analysis, focusing on the examination of network packet capture (PCAP) files to detect possible security threats. The development follows an organized method in the field of network forensics. It starts with basic file structure examination and moves toward more advanced techniques for identifying suspicious activities.

Network security depends largely on the ability to study captured network traffic to find signs of harmful actions. PCAP files, generated by applications such as Wireshark, hold important forensic data that can help uncover security events when carefully analyzed. This project creates five connected Python programs that work together to extract and examine this information in a systematic way.

What is PCAP File?

A PCAP (Packet Capture) file is a data file that records network traffic collected through packet sniffing. It stores raw data packets transmitted across a network, allowing IT and security teams to analyze network activity, troubleshoot issues, and detect security threats. Tools like Wireshark, tcpdump, and tshark are commonly used to create PCAP files. These files provide key details such as source and destination IP addresses, protocol information, payload data, and timestamps, offering valuable insights into network performance and potential anomalies.

A PCAP file begins with a 24-byte header, referred to as *pcap_file_header* in the libpcap source code. This header includes key information such as the Magic Number (d4 c3 b2 a1), version numbers (major: 02 00, minor: 04 00), Timezone, and Timestamp Accuracy both typically set to zero. It also specifies the Snap Length, which defines the maximum packet size captured, and the Link-Layer Type, indicating the network link layer used

One of the main advantages of packet capturing and PCAP files is improved network visibility. By analyzing captured packet data, it becomes easier to identify the root causes of network issues, monitor traffic sources, and track application and device usage. PCAP data provides real-time insights that help detect and resolve performance problems, ensuring network stability even after security incidents.

The placement of a packet sniffer also impacts its effectiveness. If it is positioned at the network's edge, visibility may be limited, potentially causing missed signs of events like the onset of a DDoS attack or a malware outbreak. Even when monitoring from the network's core, it is essential to capture complete packet conversations rather than just summary data to ensure accurate analysis.

Requirements

- A Python interpreter is recommended, preferably using an integrated development environment (IDE) like PyCharm, though it can also be executed through a standard terminal.
- Ensure the PCAP file is available in your working directory to test the program's functionality.
- An active internet connection is necessary, as the program relies on external APIs to verify IP-related data

Main Files in this project

1. Global Header Analyzer.py - Examines the 24-byte metadata header of PCAP files to validate file structure, format, and capture settings.
2. DHCP Frame Analyzer.py - Extracts timestamp, MAC/IP addresses, and hostname information from the first DHCP packet in the capture.
3. .Top Domain Analyzer.py - Identifies potentially malicious .top domains in network traffic and checks their reputation using AbuseIPDB.
4. Search Engine Inspector.py - Analyses HTTP requests to determine search engine usage, search terms, and websites visited prior to security incidents.
5. Final Security Tool.py - Integrates all previous tools into a comprehensive menu-driven application for complete PCAP security analysis.

1. Global Header Analysis.py

The Global Header Analyzer focuses on examining the key metadata found at the start of every PCAP file. This initial section is 24 bytes long and contains important details that describe the file's structure and properties. The analysis includes the following components:

- Magic Number: A unique value (either 0xa1b2c3d4 or 0xd4c3b2a1) that identifies the file's byte order, also known as endianness.
- Version Numbers: The major and minor version numbers define the format specification used for the PCAP file.
- Snap Length: This field indicates the maximum number of bytes that were captured for each packet.
- Data Link Type: An identifier that specifies the link-layer protocol, commonly set to 1 for Ethernet networks.

This initial stage of analysis plays a critical role in verifying that the PCAP file is authentic and properly structured. Confirming the validity of the file is essential before performing a more detailed packet examination.

The program applies Python's struct module to decode the binary data into a human-readable format. This provides a clear understanding of how the network traffic was recorded.

2. DHCP Frame Analyzer.py

The DHCP Frame Analyzer is designed to extract important data from the first DHCP (Dynamic Host Configuration Protocol) packet found within the network capture. DHCP packets play a key role in network setup and provide valuable information about the devices and their configurations. The analysis covers the following elements:

- Timestamp: Records the time the packet was captured, given in epoch format and converted into a readable GMT (Greenwich Mean Time) format.
- Frame Length: Shows the total size of the packet in bytes.
- Source and Destination MAC Addresses: Identify the physical (hardware) addresses of the devices involved in the communication.
- Source and Destination IP Addresses: Provide information about the network addresses used by the devices.

- Client Hostname: Displays the name of the client device, helping to establish its identity within the network.

This analysis helps security professionals recognize which devices participated in the communication and assists in creating an accurate timeline of network activities.

The program makes use of the Scapy library, which allows detailed inspection and extraction of protocol-specific data from complex packet structures.

3. Top Domain Analyzer.py

The Top Domain Analyzer is developed to detect potentially harmful websites that use the .top domain extension. This domain type is often linked to suspicious or malicious activities. The tool carries out the following tasks:

- Pattern Matching with Regular Expressions: Scans the packet data to locate different forms of .top domain names by using advanced regex (regular expression) techniques.
- Domain Resolution: Converts the detected domain names into their corresponding IP addresses for further inspection.
- Domain Analysis: Examines the characteristics of the domains, such as their length, the presence of random or unusual character patterns, and numeric sequences that may indicate suspicious behaviour.
- Reputation Checking via AbuseIPDB API: Connects to the AbuseIPDB service to verify the reputation and risk score of the identified IP addresses or domains.
- Report Generation: Produces comprehensive reports that include detailed findings and threat evaluations.

This module highlights how regex (regular expressions)-based pattern recognition and external threat intelligence sources can be combined to improve the detection of security threats.

4. Search Engine Inspector.py

The Search Engine Inspector analyses HTTP request packets to rebuild a timeline of user browsing activities that occurred before a potential security event. This examination provides insights into user behaviour by focusing on the following points:

- Search Engines Used: Identifies which search engines were accessed, with an emphasis on alternatives to Google, such as Bing, Yahoo, and DuckDuckGo.
- Search Terms and Queries: Extracts the specific keywords and search phrases entered by the user.
- Suggested Websites: Examines the list of websites recommended in the search engine results.
- Visited Websites: Determines which websites the user actually visited after performing searches.

This behavioural analysis assists in understanding the attack vector the path or method used by attackers and tracks user activities that may have contributed to the security breach.

The tool uses the Pyshark library to filter HTTP traffic and to extract important request headers and search parameters from the network packets.

5. Final Security Tool.py

The Final Security Tool brings together all the previous analysis tools into a single, unified platform. This integrated system offers advanced features to detect additional security threats and provides a practical solution for cybersecurity professionals. Its main functionalities include:

- Interactive Menu System: Allows users to easily select and navigate between different analysis modules.
- Color-Coded Console Output: Highlights important alerts and threat indicators with different colours, enabling quicker identification of potential risks.
- Detailed Reports with Recommendations: Generates comprehensive reports that not only present the findings but also offer specific security advice and recommendations.
- Modular Design: Built in a way that makes it simple to add new features or improve detection capabilities in the future.
- User-Friendly File Selection: Uses Tkinter to provide a graphical interface, making it easier for users to select PCAP files for analysis.

This final tool demonstrates how separate cybersecurity analysis methods can be combined into an effective, user-friendly application. It also reflects good software engineering practices, such as modularization, error handling, and thoughtful user interface design.

Technical Implementation

The project uses several Python libraries and techniques that are essential for effective cybersecurity analysis. The key components of the implementation include:

- Binary Data Processing: Utilizes the struct module to examine and interpret PCAP file headers.
- Network Protocol Analysis: Employs both Scapy and Pyshark libraries to inspect and analyse network packets in detail.
- Pattern Matching: Applies regular expressions (regex) to detect suspicious indicators, such as unusual domain names or patterns within the packet data.
- Network Operations: Uses the socket module to resolve domain names into IP addresses for further investigation.
- API Integration: Connects with external threat intelligence services, enriching the analysis with additional data, such as reputation scores.
- User Interface Development: Incorporates Tkinter to create an intuitive graphical interface, making the tools easier to use.
- Structured Reporting: Produces clear and organized reports to effectively communicate the results of the security analysis.

During the development of this project, special attention was given to code organization, detailed documentation through comments, error handling to improve program stability, and security practices, including API key protection.

This project demonstrates a practical application of Python programming in network security analysis, showing how programming skills can be used to create specialized tools for threat detection and incident response.

Project File Explanation

1.Global Header Analyzer

- The Global Header Analyzer.py tool is designed to examine the structural metadata of PCAP files by reading and interpreting their global header, which consists of the first 24 bytes.
- Utilizing Python's struct module, it unpacks the binary data into human-readable values. The tool extracts key information, including the file's endianness (determined by the magic number), version details, snap length (maximum packet size), and the data link type.
- This analysis helps confirm whether the file is a valid PCAP and offers insights into the configuration of the packet capture. By presenting this technical information in a clear format, the tool provides a basis for more in-depth analysis

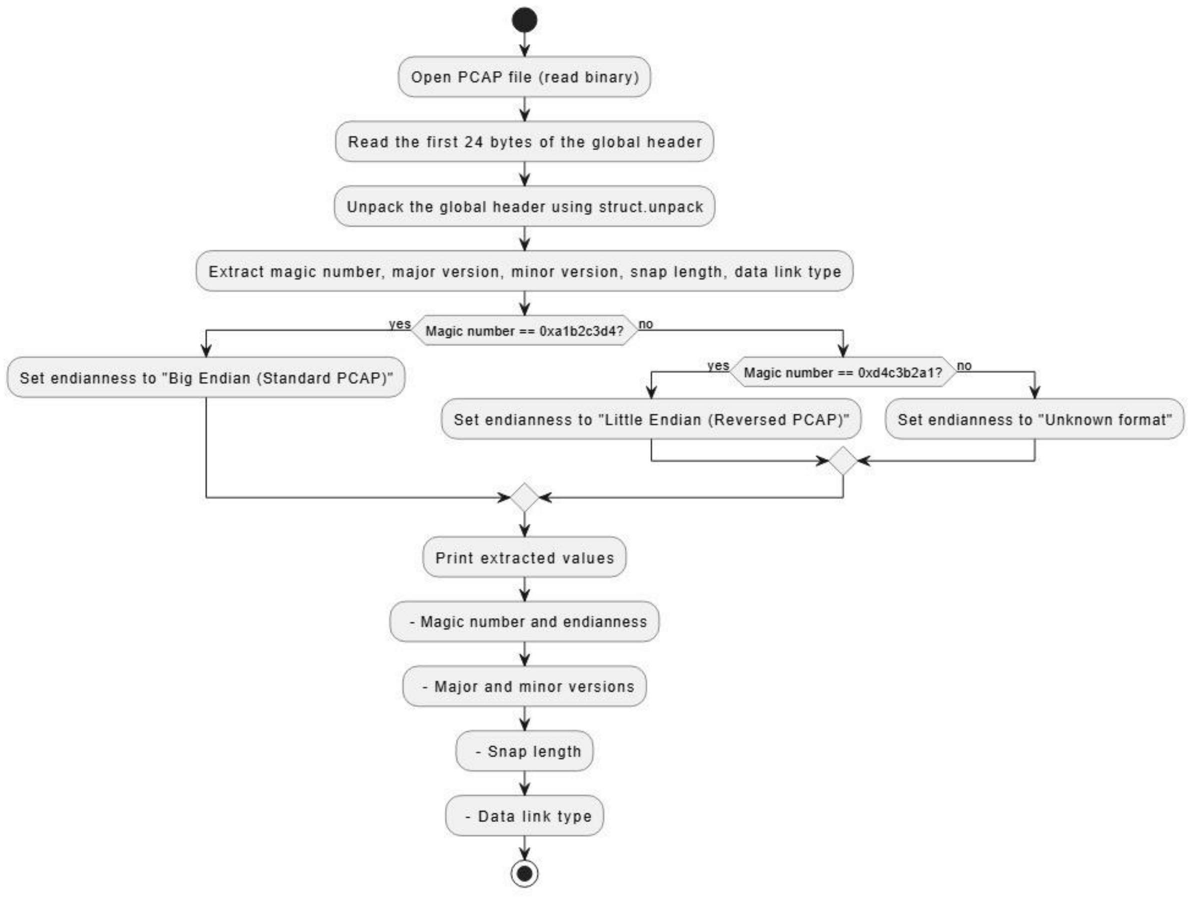
Important Elements:

- Uses struct.unpack() to decode binary header data
- Identifies file format based on the magic number
- Reports key capture settings like snap length and data link type

Key Functions:

- read_pcap_header(): Extracts and interprets the binary header data

Flow Chart


Open PCAP file (read binary)
Read the first 24 bytes of the global header
Unpack the global header using struct.unpack
Extract magic number, major version, minor version, snap length, data link type
Magic number == 0xa1b2c3d4?
yes
no
Set endianness to "Big Endian (Standard PCAP)"
Magic number == 0xd4c3b2a1?
yes
no
Set endianness to "Little Endian (Reversed PCAP)"
Set endianness to "Unknown format"
Print extracted values
- Magic number and endianness
- Major and minor versions
- Snap length
- Data link type

Explanation of Flowchart:

1. Start: The program starts execution.
2. Open PCAP file: The PCAP file is opened in binary read mode (rb).
3. Read first 24 bytes: The program reads the first 24 bytes to get the global header.
4. Unpack global header: The header data is unpacked using struct.unpack to extract the magic number, versions, snap length, and data link type.
5. Check endianness: The program checks the magic number:
    - If the magic number is 0xa1b2c3d4, it sets the endianness to "Big Endian".
    - If the magic number is 0xd4c3b2a1, it sets the endianness to "Little Endian".
    - Otherwise, it sets the endianness to "Unknown format".
6. Print extracted values: The extracted values are printed:
    - Length of the global header.
    - Magic number and the corresponding endianness.
    - Major and minor versions.
    - Snap length (number of bytes to capture).
    - Data link type (Ethernet if value is 1).

End: The program finishes its task.

2.DHCP Frame Analyzer

- The DHCP Frame Analyzer.py tool is designed to inspect DHCP (Dynamic Host Configuration Protocol) packets within network capture files, providing insight into how devices receive their network configurations. By utilizing the Scapy library, the tool detects DHCP traffic and extracts important details such as timestamps, MAC addresses, IP addresses, and device hostnames.
- This data assists security analysts in determining when devices connected to the network, linking physical addresses to logical ones, and identifying potentially unauthorized devices. Additionally, the tool converts raw timestamps into a more readable format and organizes the extracted information into a clear report, offering valuable context for understanding network structure and device activity

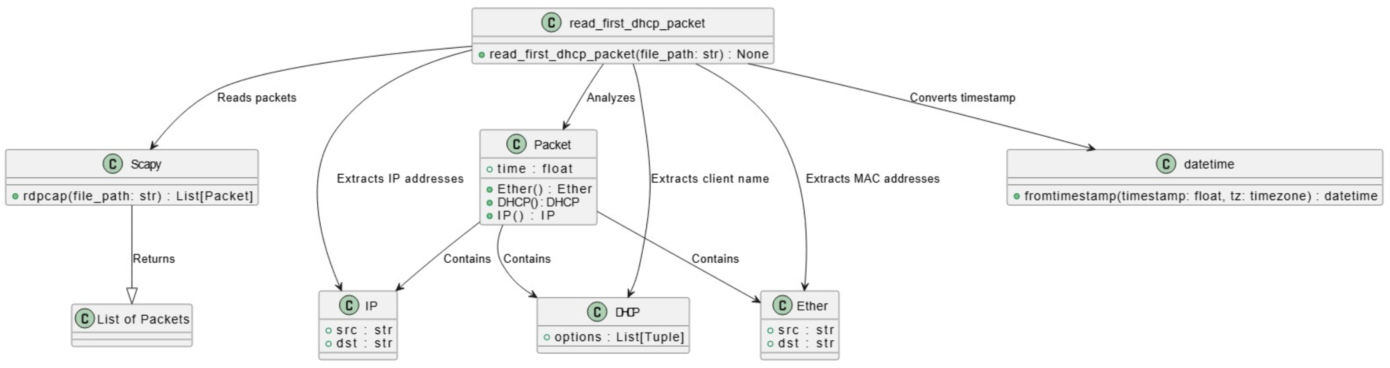
Important Elements:

- Uses Scapy library to parse packets
- Extracts critical information like MAC addresses, IP addresses, and client names
- Converts timestamps to human-readable format
- Provides insights into network device identification

Key Functions:

- read_first_dhcp_packet(): Locates and analyzes the first DHCP packet in the file

Flow Chart



How the Code Works:

1. Reading the PCAP File
   - The function read_first_dhcp_packet(file_path) uses Scapy rdpcap() to read all packets from the file.
   - This returns a list of packets, each representing a captured network frame.
2. Finding the First DHCP Packet
   - The function loops through the packets and checks if the DHCP layer exists in a packet.
   - Once found, it extracts key details.
3. Extracting Information from the Packet
   - Timestamp: Converts the packet's time using datetime.fromtimestamp().
   - MAC Addresses: Reads source (src) and destination (dst) from the Ether layer.
   - IP Addresses: Reads source (src) and destination (dst) from the IP layer.
   - Client Name: Extracts the hostname from the DHCP options.
4. Displaying the Results
   - The extracted information is printed to the console, making it easy to analyze DHCP activity.
5. End of Execution
   - Once the first DHCP packet is found and processed, the function stops execution.
   - If no DHCP packet is found, a message is printed.

Data Flow:
PCAP file → Read packets → Find DHCP packet → Extract details → Print results

3.Top Domain Analyzer

- The Top Domain Analyzer.py tool is designed to identify potentially harmful websites by examining PCAP files for suspicious domain names, with a particular emphasis on .top domains, which are often linked to malicious activities.
- It utilizes regular expressions to detect domain names within different data fields, such as URLs and HTTP headers, and analyzes their structure for signs of suspicious behavior, including patterns suggestive of algorithmically generated domains. Additionally, the tool checks the reputation of associated IP addresses using the AbuseIPDB service. By combining these methods, the tool delivers a detailed threat assessment, categorizing domains according to their reputation scores and identifying characteristics.
- This process supports security teams in detecting possible security threats, unauthorized communications, or policy violations by flagging domains that warrant further investigation.

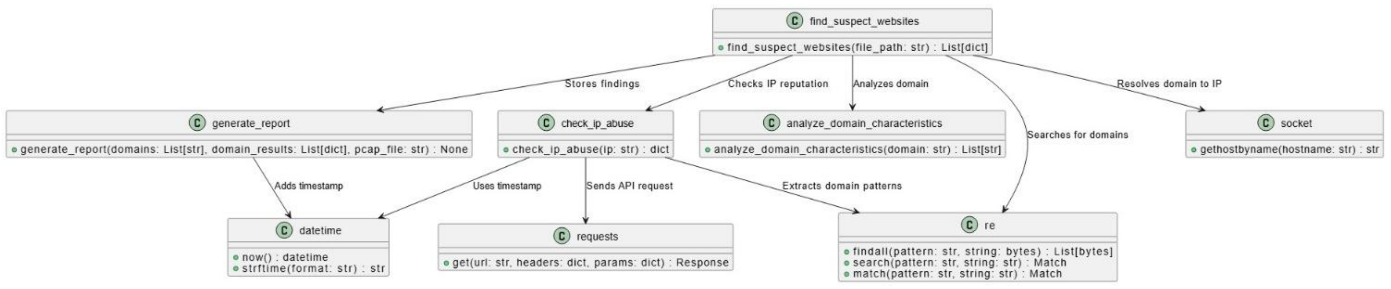
Important Elements:

- Uses regular expressions to find domain names in various formats
  - URL pattern: Matches domains in URLs with http:// or https:// prefixes
  - Host pattern: Identifies domains in HTTP Host headers
  - Referer pattern: Finds domains mentioned in HTTP Referer headers
  - Generic pattern: Captures standalone .top domains not part of other words
- Resolves domains to IP addresses for further analysis
- Checks IPs against a reputation database
- Analyzes domain name patterns for signs of algorithmic generation
- Creates detailed reports of findings

Key Functions:

- find_suspect_websites(): Main function that searches for suspicious domains
- check_ip_abuse(): Checks IP reputation using the AbuseIPDB API
- analyze_domain_characteristics(): Examines domains for suspicious patterns
- generate_report(): Creates a summary of findings

Flow Chart


find_suspect_websites
find_suspect_websites(file_path: str) : List[dict]
Stores findings
Checks IP reputation
Analyzes domain
Resolves domain to IP
Searches for domains
generate_report
generate_report(domains: List[str], domain_results: List[dict], pcap_file: str) : None
check_ip_abuse
check_ip_abuse(ip: str) : dict
analyze_domain_characteristics
analyze_domain_characteristics(domain: str) : List[str]
socket
gethostbyname(hostname: str) : str
Adds timestamp
Uses timestamp
Sends API request
Extracts domain patterns
datetime
now() : datetime
strftime(format: str) : str
requests
get(url: str, headers: dict, params: dict) : Response
re
findall(pattern: str, string: bytes) : List[bytes]
search(pattern: str, string: str) : Match
match(pattern: str, string: str) : Match

How the Code Works:

1. Extract Suspicious Domains from PCAP
   - The script reads a PCAP file to find domains with a ".top"
   - Uses regex (re.findall) to search for domains in URLs, headers, and standalone text.
2. Analyze Domain Characteristics:
   Checks if the domain has suspicious features, such as:
   - Top extension at the bottom
   - Long subdomains
   - Random-looking strings
3. Resolve Domain to IP Address:
   Uses socket.gethostbyname() to convert domain names into IP addresses.
4. Check IP Reputation Using AbuseIPDB:
   Sends an API request to AbuseIPDB to retrieve:
   - Abuse confidence score
   - Total reports
   - Country & ISP details
5. Determine Risk Level:
   Based on abuse score and reports, the script classifies the domain as:
   - Safe (low score, few reports)
   - Previously Infected (moderate score)
   - Highly Malicious (high score, many reports)
6. Generate a Report:
   Saves all findings into a detailed text report, including:
   - Extracted domains
   - Associated IPs
   - Risk assessment

Data Flow :

PCAP file → Extract domains → Resolve to IP → Check reputation → Classify risk → Generate report

4.Search Engine Inspector

- Search Engine Inspector.py is a network analysis tool designed to monitor and interpret HTTP traffic in order to identify patterns of search engine usage. It utilizes PyShark, a Python library that acts as a wrapper for Wireshark, to capture and filter HTTP request packets.
- Through this process, the tool detects connections to widely used search engines such as Bing, Yahoo, DuckDuckGo, Baidu, and Yandex. It extracts key metadata from these connections, including the search engine being accessed, the HTTP method employed, the host, the URI (which may contain search queries), and referrer links.
- aggregating and summarizing search engine activity, the tool offers valuable insights into browsing behavior, which can help identify potential security policy breaches, attempts at data exfiltration, or simply provide a better understanding of user activity within an organizational network.

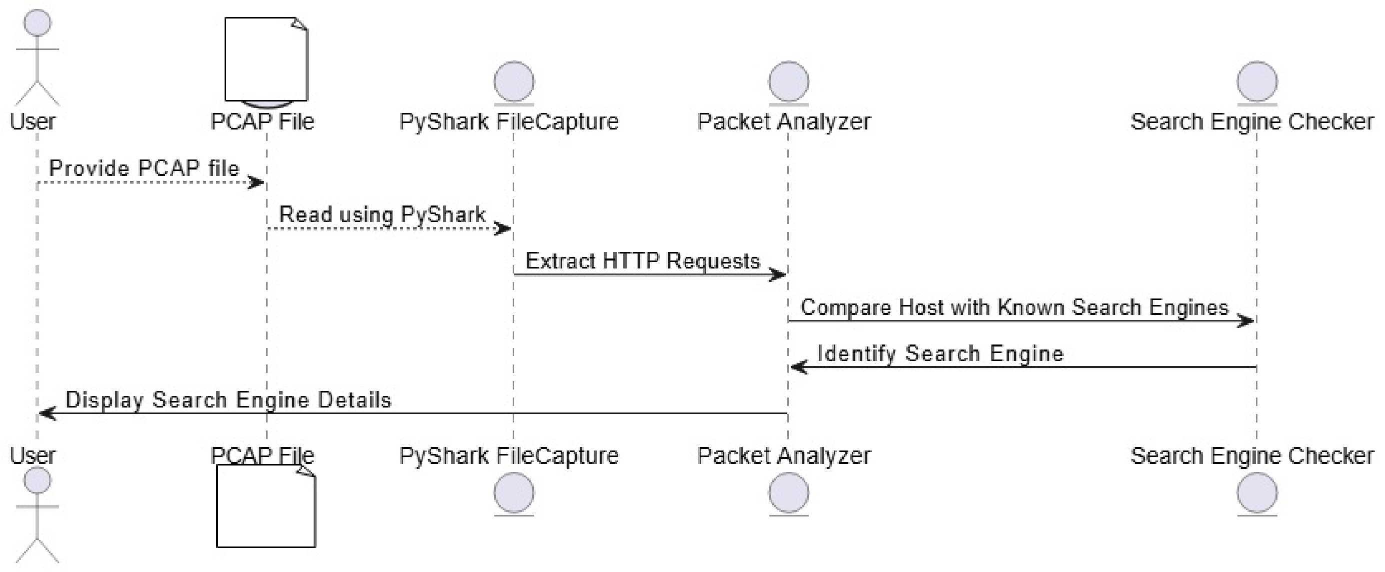
Important Elements:

- Defines a dictionary of known search engines and their domains
- Extracts critical HTTP fields like host, URI, and referrer
- Counts and summarizes search engine usage
- Provides insights into user browsing behavior

Key Functions:

- Uses PyShark to filter and analyze HTTP request packets

Flow chart



How it works:

1. Start
2. Import the pyshark library
3. Specify the pcap file path
4. Capture HTTP requests from the pcap file
5. Define known search engines
6. Iterate over each packet
    - Extract HTTP fields from the packet
    - If HTTP fields are found, check for known search engines
    - If the search engine is matched, print details
    - If there's an error, continue to the next packet

Close capture once done

5.Final Security Tool

- *Final Security Tool.py* is an advanced and comprehensive security integration tool that consolidates multiple network analysis functions into a single, user-friendly suite. It enhances user interaction through color-coded console output, which improves readability, and a menu-driven interface that simplifies the selection of specific analyses.
- This tool brings together the capabilities of earlier analyzers, including global header analysis, DHCP inspection, suspicious domain detection, and search engine tracking, while introducing additional features such as IP reputation checks using AbuseIPDB.
- Designed as a complete PCAP analysis solution, it guides users through the examination of network captures, offering well-structured outputs and practical security insights. Moreover, its modular

architecture allows focused analysis based on particular security requirements, making it a flexible and efficient tool for network security assessment.

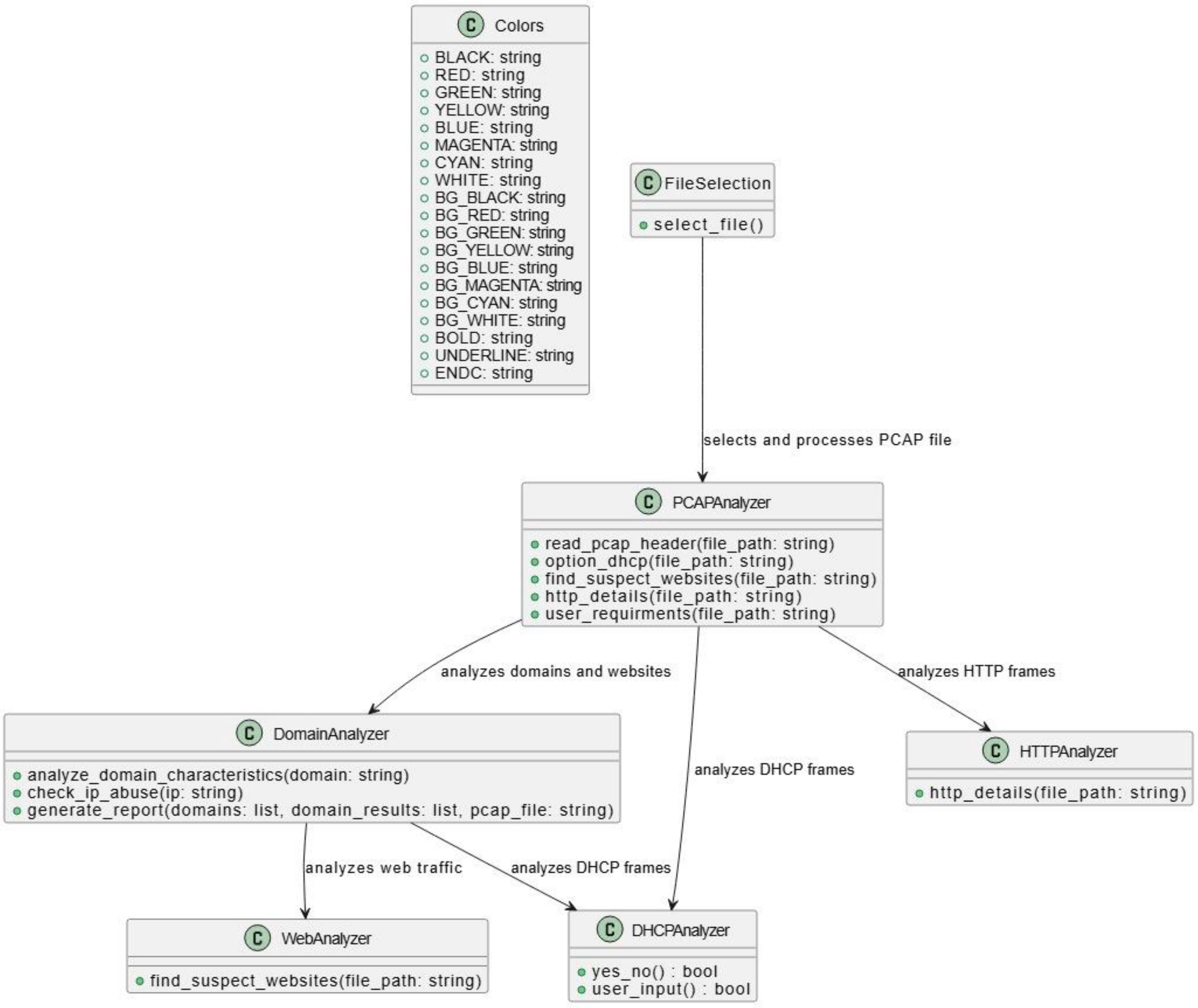
Important Elements:

- Uses the Colors class for enhanced console output, making results more readable
- Implements a modular approach that lets users select which analyses to run
- Includes comprehensive documentation explaining the purpose of each analysis

Key Functions:

- select_file(): Entry point that displays the tool introduction and handles file selection
- read_pcap_header(): Analyzes PCAP file format and settings
- option_dhcp(): Extracts DHCP information for network configuration analysis
- find_suspect_websites(): Identifies potentially malicious domains
- check_ip_abuse(): Verifies IP reputation using AbuseIPDB API
- http_details(): Analyzes HTTP traffic to identify search engine usage
- user_requirments(): Main control function for the user interface

flow chart



Explanation:

1. Colors Class: Contains constants for color and style formatting.
2. FileSelection Class: Contains the select_file() function which prompts the user to select a PCAP file.
3. PCAPAnalyzer Class: Contains various functions like read_pcap_header(), option_dhcp(), find_suspect_websites(), http_details(), and user_requirments(). This is the core class that orchestrates the entire analysis.
4. DomainAnalyzer Class: Analyzes domain characteristics and checks IP abuse using an external API.
5. DHCPAnalyzer Class: Contains functions for handling user input related to DHCP packet analysis.
6. WebAnalyzer Class: Analyzes the traffic to find suspicious websites.
7. HTTPAnalyzer Class: Handles the HTTP request analysis.

Conclusion

This report has examined the vital importance of PCAP files and their analysis in contemporary cybersecurity practices. Through network packet analysis, security professionals gain crucial insight into traffic patterns, enabling them to detect threats, resolve connectivity issues, and investigate possible security breaches.

Our investigation centered on several specialized Python-based tools designed to extract and interpret different elements of PCAP data. The Global Header Analyzer evaluates the core structure and metadata of PCAP files, offering valuable information on capture settings and file integrity. The DHCP Frame Analyzer uncovers network configurations and device identities, assisting in the identification of both authorized and potentially unauthorized devices on the network. The Top Domain Analyzer uses advanced regular expression techniques to spot suspicious domains, paying special attention to .top domains, which are often linked to malicious activities.

These tools rely on robust libraries such as Scapy for packet manipulation, struct for parsing binary data, and regular expressions for pattern recognition. Additionally, integration with external services like AbuseIPDB strengthens threat detection by supplying reputation data for suspicious IP addresses and domains.

For cybersecurity practitioners, these tools serve as essential components of their defensive toolkit. They offer greater visibility into network communications, facilitate the identification of unusual connections, and help detect potential threats early. By automating PCAP file analysis, security teams can efficiently handle large volumes of network data and concentrate on addressing genuine security risks.

Strengths of This Project:

- **Modular Architecture:** Dividing the tools into distinct, specialized modules enables focused analysis of various aspects of network traffic.
- **Comprehensive Detection:** Together, the tools offer broad visibility into file metadata, network configurations, suspicious domains, and user browsing activity.
- **User-Friendly Output:** The use of emoji-based indicators and color-coded console output simplifies complex technical data, making it easier to interpret.
- **External API Integration:** By connecting with AbuseIPDB, the tools enhance threat detection by providing reputation data on suspicious IP addresses.
- **Pattern Recognition:** Advanced regular expressions effectively identify potentially harmful domains and extract meaningful information from packet data.
- **Report Generation:** The tools automatically produce detailed, timestamped reports, simplifying documentation and sharing of findings.
- **File Format Analysis:** Accurate interpretation of binary data ensures insight into capture settings and validates the integrity of PCAP files.
- **Suspicious Domain Heuristics:** Multiple detection techniques help identify algorithmically generated or suspicious domain names.
- **Interactive User Interface:** A menu-driven system in the Final Security Tool allows users to select specific analyses, improving usability.
- **Cross-Platform Compatibility:** The use of Tkinter for file dialogs ensures the tools function smoothly across different operating systems.

Limitation of the project

- **Limited TLD Coverage:** The Top Domain Analyzer focuses solely on .top domains, missing potentially malicious domains with other TLDs.
- **DHCP Analysis Limitation:** By only examining the first DHCP packet, the DHCP Frame Analyzer may overlook crucial information from later DHCP transactions.
- **Encrypted Traffic Blindness:** The tools cannot analyze HTTPS traffic, which constitutes the bulk of modern internet communication.
- **API Key Exposure:** Storing the AbuseIPDB API key directly in the script instead of using secure environment variables creates a security risk.
- **Error Handling Gaps:** Incomplete exception handling increases the risk of crashes when processing malformed packets.
- **Limited Scalability:** Processing large PCAP files entirely in memory can lead to performance issues and slowdowns.
- **Minimal Correlation:** The tools work independently on different aspects of network traffic with limited cross-referencing between findings.
- **No Machine Learning:** The reliance on fixed patterns prevents adaptive learning, making it harder to identify evolving threats.
- **HTTP-Only Search Analysis:** The Search Engine Inspector cannot monitor search queries conducted over encrypted search engine traffic.
- **Manual Interpretation Required:** Although the tools highlight suspicious patterns, human intervention is necessary to distinguish genuine threats from false positives.

Reference

1. AbuseIPDB. (n.d.). *AbuseIPDB API documentation*. Retrieved from https://www.abuseipdb.com/
2. Comparitech. (n.d.). *The ultimate guide to PCAP files: What they are and how to work with them*. Retrieved from https://www.comparitech.com/net-admin/pcap-guide/
3. Netresec. (2022, October). *What is a PCAP file?* Retrieved from https://www.netresec.com/?page=Blog&month=2022-10&post=What-is-a-PCAP-file
4. Sysdig. (n.d.). *What is a PCAP file? Understanding packet capture format*. Retrieved from https://sysdig.com/learn-cloud-native/what-is-a-pcap-file/