CS-GY-9053 INTRO TO JAVA
FINAL PROJECT REPORT

Network packet analyzers become a useful tool for system and network administrators to capture such kind of network information. In this report, an implementation of a network packet analyzer based on wireshark will be described. We use TCP, UDP and ICMP packet filtering, which can be extended to other protocols. Currently, the project works perfectly in Windows OS.

SOFTWARE REQUIREMENTS

- Jdk 1.8, java runtime environment
- Eclipse IDE
- Jpcap library
- Jpcap
- Winpcap

SETUP FOR IMPORTANT EXTERNAL LIBRARIES

1. Jpcap is the major tool used for creating network packet sniffers in Java. It provides a way to capture and analyze network traffic, which is essential for tasks such as network monitoring, security analysis, and troubleshooting. Jpcap allows the implementation of custom packet processing logic for tasks like filtering, inspecting, or logging specific packets.

   Steps for Setting Up and Using Jpcap for Java Packet Sniffer Tool:
   i) After installing Jpcap, locate the "Jpcap.dll" and "jpcap.jar" files on your system.
   ii) Open your system's environment variables settings. Add a new entry to the "PATH" or "CLASSPATH" variable, including the path to the directory containing the Jpcap files.
   iii) Find "Jpcap.dll" and copy it to either "[JRE directory]\bin" or "[JRE directory]\lib\ext\x86".
   iv) Copy "jpcap.jar" to "[JRE directory]\lib\ext".
   v) Also, copy "jpcap.jar" to "[JDK directory]\jre\lib\ext"
   vi) In the Java project, navigate to "Properties" and then "Libraries."
   vii) Click the "Add JAR/Folder" button and add the "jpcap.jar" file to the project's external libraries.
   viii) Jpcap requires elevated privileges (ROOT/Administrator) to access raw network data, as mentioned in the provided instructions.

2. SLF4J is used for logging various events, information, and debugging messages. We added SLF4J log statements throughout the codebase, specifying different log levels (e.g., DEBUG, INFO, WARN, ERROR) to provide comprehensive insight into the tool's behavior. Logging is crucial in a packet sniffer tool to track the tool's activity, capture errors, and record packet capture information for debugging and analysis. These JAR files can be added to the project's external libraries.s

3. SQLite provides a lightweight and embedded database solution suitable for local storage in a packet sniffer tool. We used it to capture packet data in an SQLite database for efficient data retrieval, querying, and historical analysis. For this, we downloaded the JAR file (e.g., sqlite-jdbc.jar), added it to the Java project's classpath. This can usually be done by copying the JAR file to your project's "lib" directory same as before. All the data is stored in the “sample.db” file.

INSTRUCTIONS FOR RUNNING THE PROJECT

1. File Description:
    - External Lib App.zip - contains all the external libraries for running the application
    - Java project zip.zip - contains source code of the application
    - Java_Project_Runnable.jar - Executable jar file of the application with packaged libraries
    - Java Project Demo.mp4 - Video Demo of the project
    - Java Final Project Report - this document

2. Launch Eclipse with administrator privileges on a Windows system.
3. If there are any import errors, ensure that you have added the external JAR files provided.
4. Once all import errors have been resolved, execute the "Sniffer.java" file, and the project will be ready to run.

ADVANCED COMPONENTS COVERED

- ## Network Data:

  Network analysis is the core functionality of the packet sniffer. We capture network packets in each interface, parse packet headers for further analysis.

- ## Database (SQLite Database):

  An SQLite database is utilized to store captured network packet data, metadata, and analysis results. This database is used to save packet capture information for later analysis, historical tracking, and reporting. SQL queries are employed to retrieve specific data from the database, aiding in packet analysis.

- ## User Interface (Swing UI):

  A Swing-based graphical user interface (UI) is designed and built to provide users with an interactive and user-friendly interface for controlling and monitoring the packet sniffer tool.
  The UI can offer features like starting and stopping packet capture, displaying captured packet data, configuring capture filters, and presenting analysis results.

- ## Threads:

  Threads are employed to manage concurrency and ensure the efficient operation of the packet sniffer tool. Thread management helped in handling tasks concurrently for packet capture, processing, database storage and UI responsiveness without blocking or freezing.