**README**

**Program: Flow Log Tagger**

**Overview:**

This Java program parses AWS flow log data (version 2 format only) and maps each log entry to a tag based on a lookup table that defines port and protocol combinations. The lookup table is provided as a CSV file containing dstport, protocol, and tag. The program outputs two files:

* tag\_counts.csv: Counts of matched tags.
* port\_protocol\_counts.csv: Counts of port/protocol combinations.

**Assumptions:**

* The flow log format adheres to AWS VPC default format version 2.
* The lookup table is case-insensitive for port and protocol matching.
* If a flow log entry does not match any dstport and protocol in the lookup table, it is tagged as "Untagged."
* Only TCP, UDP, and ICMP protocols are supported. Any unknown protocols are considered invalid or unsupported.

**Requirements:**

* Java Development Kit (JDK) 8 or higher.

**Input Files:**

1. **Flow Log File**: Contains the log data in AWS default version 2 format.
2. **Lookup Table File**: Contains port, protocol, and tag mappings in CSV format. Format:

swift

dstport,protocol,tag

**Compilation Instructions:**

1. Download or clone the project to your local machine.
2. Navigate to the project directory and compile the program:

Copy code

javac FlowLogTagger.java

**Running the Program:**

To run the program, pass two arguments to the main method: the path to the flow log file and the lookup table file.

Example:

code

java FlowLogTagger flow\_logs.txt lookup\_table.csv

After execution, two output files will be generated in the same directory:

* **tag\_counts.csv**: Counts of tags mapped from the flow logs.
* **port\_protocol\_counts.csv**: Counts of port/protocol combinations.

**Expected Output Files:**

1. **tag\_counts.csv**: File containing the count of matches for each tag.

mathematica

code

Tag,Count

sv\_P2,1

sv\_P1,2

email,3

Untagged,9

1. **port\_protocol\_counts.csv**: File containing the count of matches for each port/protocol combination.

mathematica

code

Port,Protocol,Count

22,tcp,1

23,tcp,1

25,tcp,1

110,tcp,1

143,tcp,1

443,tcp,1

993,tcp,1

1024,tcp,1

49158,tcp,1

80,tcp,1

**Tests Performed:**

1. **Basic Functionality**:
   * Test with sample flow logs and a lookup table to verify that the program can correctly map the logs to tags and count matches.
   * Check for case-insensitivity in matching protocols from the lookup table.
2. **Unmatched Logs**:
   * Verify that logs not matching any entry in the lookup table are tagged as "Untagged."
3. **Port/Protocol Matching**:
   * Test that the program can correctly map logs with various dstport/protocol combinations, including corner cases such as:
     + Logs with unsupported protocols (e.g., anything other than TCP, UDP, ICMP).
     + Flow logs with different destination ports to ensure correct matching from the lookup table.
4. **Performance Test**:
   * Simulated a large flow log file (up to 10 MB) to ensure the program processes it efficiently.

**Additional Analysis:**

* The program has been designed to handle up to 10 MB flow log files efficiently by using Java’s built-in BufferedReader to process the log file line by line.
* The lookup table can contain up to 10,000 mappings and is loaded into memory using a HashMap, allowing fast lookup operations when processing the logs.
* All string matching (port/protocol) is case-insensitive to avoid mismatches caused by varying cases in the lookup table.
* If the log file contains an unsupported or unrecognized protocol, the program skips that log entry, ensuring no crashes during processing.