**1. Count how many packets use each protocol (TCP, UDP, ICMP)**

Copy code

**awk -F"," '{count[$4]++} END {for (protocol in count) print protocol, count[protocol]}' k.h**

* **awk**: Calls the AWK command-line utility, which is used for pattern scanning and processing.
* **-F","**: Specifies the field separator as a comma (,), allowing AWK to split lines into fields based on commas.
* **'{count[$4]++}'**:
  + **count[$4]**: Initializes or accesses an associative array named count, where the key is the value of the fourth field ($4, representing the protocol).
  + **++**: Increments the count for that protocol each time it appears in the input.
* **END {for (protocol in count) print protocol, count[protocol]}**:
  + **END**: This block executes after all input lines are processed.
  + **for (protocol in count)**: Iterates over each unique key (protocol) in the count array.
  + **print protocol, count[protocol]**: Outputs the protocol and its corresponding count.
* **k.h**: The input file containing the network traffic data.

**2. Filter and print only the dropped packets**

Copy code

**awk -F"," '$8 == "Dropped" {print}' k.h**

* **awk**: Calls the AWK utility.
* **-F","**: Sets the field separator to a comma.
* **'$8 == "Dropped"'**:
  + **$8**: Refers to the eighth field (Status).
  + **== "Dropped"**: Checks if the status is exactly "Dropped".
* **{print}**: If the condition is true, print the entire line.
* **k.h**: Input file name.

**3. Print Timestamp, Source\_IP, Destination\_IP, and Packet\_Size for packets >1000 bytes**

Copy code

**awk -F"," '$7 > 1000 {print $1, $2, $3, $7}' k.h**

* **awk**: Invokes AWK.
* **-F","**: Uses a comma as the field separator.
* **'$7 > 1000'**:
  + **$7**: Refers to the seventh field (Packet\_Size).
  + **> 1000**: Checks if the packet size is greater than 1000 bytes.
* **{print $1, $2, $3, $7}**: If the condition is met, prints the first (Timestamp), second (Source\_IP), third (Destination\_IP), and seventh (Packet\_Size) fields.
* **k.h**: Input file name.

**4. Display traffic directed to destination port 443**

Copy code

**awk -F"," '$6 == 443 {print}' k.h**

* **awk**: Starts the AWK utility.
* **-F","**: Specifies the comma as the field separator.
* **'$6 == 443'**:
  + **$6**: Refers to the sixth field (Destination\_Port).
  + **== 443**: Checks if the destination port equals 443.
* **{print}**: If the condition is true, prints the entire line.
* **k.h**: Input file name.

**5. Print all unique Source\_IP addresses**

Copy code

**awk -F"," '!seen[$2]++ {print $2}' k.h**

* **awk**: Invokes the AWK utility.
* **-F","**: Sets the field separator to a comma.
* **'!seen[$2]++'**:
  + **seen[$2]**: Creates an associative array called seen, where $2 (Source\_IP) is the key.
  + **!**: Logical NOT operator; true if the key has not been seen before.
  + **++**: Increments the count of that Source\_IP in the seen array.
* **{print $2}**: If the condition is true (meaning the Source\_IP is unique), it prints that Source\_IP.
* **k.h**: Input file name.

**6. Filter only TCP traffic and calculate the average packet size**

Copy code

**awk -F"," '$4 == "TCP" {total+=$7; count++} END {if (count > 0) print "Average Packet Size:", total/count}' k.h**

* **awk**: Calls the AWK utility.
* **-F","**: Sets the comma as the field separator.
* **'$4 == "TCP"**:
  + **$4**: Refers to the fourth field (Protocol).
  + **== "TCP"**: Checks if the protocol is TCP.
* **{total+=$7; count++}**:
  + **total+=$7**: Adds the packet size (seventh field) to a variable called total.
  + **count++**: Increments the counter for TCP packets.
* **END {if (count > 0) print "Average Packet Size:", total/count}**:
  + **END**: Executes after processing all lines.
  + **if (count > 0)**: Checks if any TCP packets were counted.
  + **print "Average Packet Size:", total/count**: Outputs the average packet size.
* **k.h**: Input file name.

**7. Count invalid records**

Copy code

**awk -F"," 'NF != 8 {invalid++} END {print "Invalid Records:", invalid+0}' k.h**

* **awk**: Invokes AWK.
* **-F","**: Sets the comma as the field separator.
* **'NF != 8'**:
  + **NF**: A built-in variable representing the number of fields in the current record.
  + **!= 8**: Checks if the number of fields is not equal to 8, indicating an invalid record.
* **{invalid++}**: Increments the invalid counter for each invalid record.
* **END {print "Invalid Records:", invalid+0}**:
  + **END**: Executes after processing all lines.
  + **print "Invalid Records:", invalid+0**: Outputs the count of invalid records, using +0 to ensure it's treated as a number.
* **k.h**: Input file name.

**8. Extract rows where Source\_IP is in the 192.168.x.x range**

Copy code

**awk -F"," '$2 ~ /^192\.168\./ {print}' k.h**

* **awk**: Invokes AWK.
* **-F","**: Sets the comma as the field separator.
* **'$2 ~ /^192\.168\./'**:
  + **$2**: Refers to the second field (Source\_IP).
  + **~**: Matches against a regular expression.
  + **/^192\.168\./**: Regular expression that checks if the Source\_IP starts with 192.168..
* **{print}**: Prints the entire line if the condition is met.
* **k.h**: Input file name.

**9. Match traffic directed to either port 80 (HTTP) or port 443 (HTTPS)**

Copy code

**awk -F"," '$6 == 80 || $6 == 443 {print}' k.h**

* **awk**: Starts the AWK utility.
* **-F","**: Sets the field separator to a comma.
* **'$6 == 80 || $6 == 443'**:
  + **$6**: Refers to the sixth field (Destination\_Port).
  + **== 80 || $6 == 443**: Checks if the destination port is either 80 (HTTP) or 443 (HTTPS).
* **{print}**: Prints the entire line if the condition is true.
* **k.h**: Input file name.

**10. Filter out rows where Destination\_Port contains alphanumeric characters**

Copy code

**awk -F"," '$6 !~ /[a-zA-Z]/ {print}' k.h**

* **awk**: Calls the AWK utility.
* **-F","**: Sets the field separator to a comma.
* **'$6 !~ /[a-zA-Z]/'**:
  + **$6**: Refers to the sixth field (Destination\_Port).
  + **!~**: Checks if the field does not match the regular expression.
  + **/[a-zA-Z]/**: Regular expression that matches any letter (either uppercase or lowercase).
* **{print}**: Prints the entire line if the condition is met.
* **k.h**: Input file name.

**11. Filter out traffic where Protocol is TCP and Destination\_Port is 443**

Copy code

**awk -F"," '!($4 == "TCP" && $6 == 443) {print}' k.h**

* **awk**: Invokes AWK.
* **-F","**: Sets the field separator to a comma.
* **'!($4 == "TCP" && $6 == 443)'**:
  + **$4 == "TCP"**: Checks if the protocol is TCP.
  + **&&**: Logical AND operator.
  + **$6 == 443**: Checks if the destination port is 443.
  + **!**: Negates the condition, meaning it matches lines that do not meet both conditions.
* **{print}**: Prints the entire line if the condition is true.
* **k.h**: Input file name.

**12. Filter out and print traffic where Packet\_Size > 1000 OR Status is Dropped**

Copy code

**awk -F"," '$7 > 1000 || $8 == "Dropped" {print}' k.h**

* **awk**: Calls the AWK utility.
* **-F","**: Sets the field separator to a comma.
* **'$7 > 1000 || $8 == "Dropped"'**:
  + **$7**: Refers to the seventh field (Packet\_Size).
  + **> 1000**: Checks if the packet size is greater than 1000 bytes.
  + **||**: Logical OR operator.
  + **$8 == "Dropped"**: Checks if the status is "Dropped".
* **{print}**: Prints the entire line if either condition is met.
* **k.h**: Input file name.

**13. Print traffic NOT originating from 192.168.x.x IP addresses**

Copy code

**awk -F"," '$2 !~ /^192\.168\./ {print}' k.h**

* **awk**: Starts the AWK utility.
* **-F","**: Sets the field separator to a comma.
* **'$2 !~ /^192\.168\./'**:
  + **$2**: Refers to the second field (Source\_IP).
  + **!~**: Checks if the field does not match the regular expression.
  + **/^192\.168\./**: Regular expression that checks if the Source\_IP starts with 192.168..
* **{print}**: Prints the entire line if the condition is met.
* **k.h**: Input file name.

**14. Filter rows where both Source\_IP and Destination\_IP are in the 192.168.x.x range**

Copy code

**awk -F"," '$2 ~ /^192\.168\./ && $3 ~ /^192\.168\./ {print}' k.h**

* **awk**: Invokes the AWK utility.
* **-F","**: Sets the field separator to a comma.
* **'$2 ~ /^192\.168\./ && $3 ~ /^192\.168\./'**:
  + **$2**: Refers to the second field (Source\_IP).
  + **~**: Matches against a regular expression.
  + **/^192\.168\./**: Regular expression to check if the Source\_IP starts with 192.168..
  + **&&**: Logical AND operator.
  + **$3 ~ /^192\.168\./**: Checks if the Destination\_IP (third field) also starts with 192.168..
* **{print}**: Prints the entire line if both conditions are true.
* **k.h**: Input file name.

**15. Filter out traffic where Destination\_Port is 22 OR Packet\_Size < 100 bytes**

Copy code

**awk -F"," '!($6 == 22 || $7 < 100) {print}' k.h**

* **awk**: Calls the AWK utility.
* **-F","**: Sets the field separator to a comma.
* **'!($6 == 22 || $7 < 100)'**:
  + **$6 == 22**: Checks if the Destination\_Port is 22.
  + **||**: Logical OR operator.
  + **$7 < 100**: Checks if the Packet\_Size is less than 100 bytes.
  + **!**: Negates the condition, matching lines that do not meet either condition.
* **{print}**: Prints the entire line if the condition is true.
* **k.h**: Input file name.

Feel free to ask if you need further explanations or have additional questions!