Illumio Technical Assessment

August 26, 2024

Candidate: Rishabh Verma

Recruiter: Tyler DeShazer

Task

Write a program that can parse a file containing flow log data and maps each row to a tag based on a lookup table. The lookup table is defined as a csv file, and it has 3 columns, dstport,protocol,tag. The dstport and protocol combination decide what tag can be applied.

The program should generate an output file containing the following:

- Count of matches for each tag
- Count of matches for each port/protocol combination

Requirements:

- Input file as well as the file containing tag mappings are plain text (ascii) files
- The flow log file size can be up to 10 MB
- The lookup file can have up to 10000 mappings
- The tags can map to more than one port, protocol combinations.
- The matches should be case insensitive
- Avoid using non-default libraries or packages like Hadoop, spark, pandas etc
- Include instructions on how to compile/run the program, what tests were done, and any other analysis you may want to share
- The submission should come with a readme with info on all the assumptions made

0. README

This challenge was completed using Python 3 in a Jupyter notebook. Output files were generated and committed. For easy verification, you can run `demo.py` with any Python 3.X interpreter; it contains all the same code in the Jupyter notebook.

Assumptions made:

- The only protocol codes we expect to see are the first 146. The code can easily be modified to accomodate all 255 possible values.
- The lookup table only contains up to 10,000 mappings of <16 characters, and can be

encoded using 8-bit ASCII, so it can easily sit in memory in <2MB.

Tests performed:

- Just reading from the data given in CSV form and stripping whitespace!

Analysis:

- All mappings are performed using O(1) operations. All input files are parsed once-over. The full program runs in optimal O(n) time complexity, using O(1) additional space complexity.

1. Load the mapping data

I started by downloading the official protocols csv and using regex/Sublime to clean up the file so it only contains the decimal and keyword for the 146 assigned protocols.

Now, I could load this in as a list or as a map. A map offers easier maintenance for future development, but a list has a lower memory footprint (of constant-order, so it's not a big deal). I went with a list.

In [7]:
```
protocols = {}

with open("protocols.csv", "r") as file:
    for i, line in enumerate(file):
        protocols[str(i)] = line.strip().split(',')[1].lower()
        # We're using str(i) as the key because that's how it's represented in the

print("A few key-value pairs from the protocols map:")
{str(i): protocols[str(i)] for i in range(7)}
```

```
A few key-value pairs from the protocols map:
```

Out[7]:
```
{'0': 'hopopt',
 '1': 'icmp',
 '2': 'igmp',
 '3': 'ggp',
 '4': 'ipv4',
 '5': 'st',
 '6': 'tcp'}
```

The lookup table can have up to 10000 mappings, so it'll sit in-memory just fine as a map (dictionary), where the first two values form a key.

In [20]:
```
lookup = dict()

with open("input_lookup.csv", "r") as file:
    file.readline() # skip the first line
    for i, line in enumerate(file):
        contents = line.split(",")
        port_protocol = ",".join(contents[:2])
        tag = contents[2].strip().lower()
        lookup[port_protocol] = tag

print(f"lookup: {lookup}\n")
```

```
lookup: {'25,tcp': 'sv_p1', '68,udp': 'sv_p2', '23,tcp': 'sv_p1', '31,udp': 'sv_p3',
'443,tcp': 'sv_p2', '22,tcp': 'sv_p4', '3389,tcp': 'sv_p5', '0,icmp': 'sv_p5', '110,
tcp': 'email', '993,tcp': 'email', '143,tcp': 'email'}
```

2. Parse and map the flow log

In [30]:

```
from collections import defaultdict

count_port_protocols = defaultdict(int)
count_tags = {tag: 0 for tag in lookup.values()}
count_tags["Untagged"] = 0

with open("input_flowlog.txt", "r") as file:
    for line in file:
        # Get the port_protocol
        contents = line.split()
        port_protocol = ",".join((contents[6], protocols[contents[7]]))

        # Count the port_protocol
        count_port_protocols[port_protocol] += 1

        # Get and count the tag
        if port_protocol in lookup:
            tag = lookup[port_protocol]
            count_tags[tag] = count_tags[tag] + 1
        else:
            count_tags["Untagged"] += 1

print("Our computational results:\n")
print(f"count_tags: {count_tags}\n")
print(f"count_port_protocols: {str(count_port_protocols)}\n")
```

```
Our computational results:

count_tags: {'sv_p1': 2, 'sv_p2': 1, 'sv_p3': 0, 'sv_p4': 0, 'sv_p5': 0, 'email': 3,
'Untagged': 8}

count_port_protocols: defaultdict(<class 'int'>, {'49153,tcp': 1, '49154,tcp': 1, '4
9155,tcp': 1, '49156,tcp': 1, '49157,tcp': 1, '49158,tcp': 1, '80,tcp': 1, '1024,tcp
': 1, '443,tcp': 1, '23,tcp': 1, '25,tcp': 1, '110,tcp': 1, '993,tcp': 1, '143,tcp':
1})
```

3. Format and write the data

At this point, we've done all the actual computation we need! I'm just going to write these count dictionaries to output files.

In [5]:
```
with open("output_tag_counts.csv", "w") as file:
    file.write('Tag,Count\n')
    for key, value in count_tags.items():
        file.write(f'{key},{value}\n')

with open("output_port_protocol_combination_counts.csv", "w") as file:
    file.write('Port,Protocol,Count\n')
    for key, value in count_port_protocols.items():
        file.write(f'{key},{value}\n')
```