OSU – Defense Against the Dark Arts Week 6, First Lab (aka "Lab2")

Header Analysis

We often wish to examine network traffic to analyze whether a given network is running correctly, is under attack, has secrets in it we would like to reveal or protect, and so on. There is an engineering tradeoff here. We can't store and analyze all the data that flies by in the network. So we would like to use a portion of the total data to decide where to look deeper.

Often the initial security analysis uses protocol headers or security and application logs.

- This information is small, so it can be stored for more traffic over a longer period of time
- This information can be determined from packets flying by with a minimum of processing, so it is possible to get the info without either spending a fortune in monitoring hardware or slowing down the network traffic

There is a perception that the content data is "sensitive" and that header data is "anonymized" or information that people don't care as much about. If you had time to read the Kieran Healy's amusing article on "Using Metadata to Find Paul Revere", you can see that this is not always a valid perception.

Let's see what we can find from the most basic packet header data.

The trick here is to rearrange the data in various ways and look for 'spikes' in frequency. Then look at what all the frequent items have in common, such as the same protocol, similar IP addresses, etc. To do this well, you need to know about common TCP and UDP services. Fortunately, we have Google to help us with that!

In this lab, you are provided with two samples of packet data, entitled 'R' and 'O'. The packet data is in order, and covers a brief period of time. For each packet, you get:

- Length, IP type, IP Source Address, IP Destination Address
- TCP: Flags, Source port, Destination Port
- UDP: Source port, Destination port
- ICMP: type, code

The data is stored in a textual CSV format (you can even look at it using a terminal or read it into a spreadsheet). In this assignment, you will write scripts to analyze the data. We have provided starter scripts in PERL and PYTHON, entitled scancsv.{pl,py}. You can either language, or another, if you want to rewrite the starter script.

Hand in your script and the script output, as requested in the numbered assignments, below.

For the record, the field names (line 1 of the csv) are: *len, proto, ipsrc, ipdst, tcpflags, tcpsport, tcpdport, udpsport, udpdport, icmpcode, icmptype*

The R data (100k packets) is a lot smaller than the O data (1M packets). Prototype your scripts using R data, then switch to O data to apply the concepts on a larger data set.

The ultimate goal of this assignment is to understand the function of the networks monitored in the R and O data sets. What is their function? We will choose among: work, home, data center, ISP. On the way, we'll learn as much as we can about these networks from the scanty data that we have.

The starter scripts collect statistics based on IP protocol numbers:

```
user@kali:~/Lab2$ python scancsv.py R.csv
Num packets: 99142, Num bytes: 71683046
IP Protocols:
 1:      7
 2:      2
 6:  39138
17:  59995
```

You can look the IP numbers up in /etc/protocols to find out that IP protocol 1=icmp; 2=igmp; 6=tcp; 17=udp. For example, ICMP (IP Protocol 1) occurs in 7 packets. You will have learned a little of these protocols from the advance reading. If not, google them now!

Each protocol has its own uses. For example, IGMP (proto=2) is used for router-to-router communications. So we could find the addresses of routers by looking at flows that use this protocol. Without writing a script to do this, we could just use the grep command:

```
user@kali:~/Lab2$ grep -e len -e '^[0-9][0-9]*,2,' R.csv
len,proto,ipsrc,ipdst,tcpflags,tcpsport,tcpdport,udpsport,udpdpor
t,icmpcode,icmptype
28,2,10.5.63.36,234.42.42.42,,,,,,,
28,2,10.5.63.36,234.142.142.142,,,,,,,
```

(note: the first -e argument to grep preserves the CSV label line)

We can now infer that 10.5.63.36 is a router IP address. The other addresses that start with 234.*.*.* are multicast addresses (put them into google to see this).

Find Statistics on TCP and UDP Services

Most interesting user protocols run on TCP although a few run on UDP. This usually works as follows:

- A server process listens on a given TCP or UDP port number. By convention, these "well known ports" are allocated from 1-1024 (some are outside that range by now). A well-known port is often called a *service.*
- Clients connect in to the server using this as the TCP or UDP *destination port*.

The file */etc/services* contains a listing of well-known port numbers for TCP and UDP.

1. Extend your script's statistics gathering to count the use of all well-known destination port numbers for TCP and UDP (ports 1-1024). For example, you should be able to look up in your output how many TCP packets have destination port 80 and how many UDP packets have destination port 53. Run your new script on R and O data. Enable this function using a '-stats' flag (i.e., the script should have no output unless there is a -stats flag in the command line)

```
Packet: 999914 Bytes: 366325065
  1:      6794
  6:    950654
 17:     38332
 47:      2626
 50:      1484
 89:        24
TCP Packets going to port 13: 5
TCP Packets going to port 21: 60
TCP Packets going to port 22: 26383
TCP Packets going to port 23: 6
TCP Packets going to port 25: 211205
TCP Packets going to port 53: 357
TCP Packets going to port 80: 156397
TCP Packets going to port 110: 1266
TCP Packets going to port 111: 4
TCP Packets going to port 113: 162
TCP Packets going to port 119: 3347
TCP Packets going to port 135: 4398
TCP Packets going to port 139: 7605
TCP Packets going to port 143: 624
TCP Packets going to port 179: 8
TCP Packets going to port 257: 5
TCP Packets going to port 280: 4
TCP Packets going to port 411: 4
TCP Packets going to port 443: 4673
TCP Packets going to port 445: 10867
TCP Packets going to port 465: 100
TCP Packets going to port 993: 2164
TCP Packets going to port 995: 250
TCP Packets going to port 1023: 14
UDP Packets going to port 0: 747
UDP Packets going to port 1: 3
UDP Packets going to port 13: 1
UDP Packets going to port 37: 2
UDP Packets going to port 53: 21563
UDP Packets going to port 123: 394
UDP Packets going to port 137: 396
UDP Packets going to port 138: 122
UDP Packets going to port 161: 30
UDP Packets going to port 225: 2
```

```
R.csv


Packet: 99142 Bytes: 71683046
  1:        7
  2:        2
  6:    39138
 17:    59995
TCP Packets going to port 22: 448
TCP Packets going to port 23: 118
TCP Packets going to port 25: 201
TCP Packets going to port 80: 1361
TCP Packets going to port 110: 990
TCP Packets going to port 113: 55
TCP Packets going to port 119: 68
TCP Packets going to port 135: 24
TCP Packets going to port 139: 9455
TCP Packets going to port 515: 125
TCP Packets going to port 700: 40
TCP Packets going to port 712: 301
TCP Packets going to port 721: 66
TCP Packets going to port 891: 239
UDP Packets going to port 0: 31
UDP Packets going to port 53: 428
UDP Packets going to port 67: 3
UDP Packets going to port 68: 3
UDP Packets going to port 137: 121
UDP Packets going to port 138: 118
```

2. Based on this information, characterize the main functions on each network. What kind of a network is it? (e.g., work, home, data center, ISP)

   In R.csv file, there are some ports that stand out due to its outstanding numbers of access, which is ports 80, 110 and 139. Using googles, I can see that port 80 is for default network for web server using HTTP, port 110 is used for Post Office Protocol (could be for mailing), and port 139 for Server Message Block (Allowing computer to share files, printers and other communication). Based on the information, I would say R.csv file have a work network.

   In O.csv files, ports 25, 80 and 445 has the most access, same to R.csv, I used to google to check and see that port 25 is for SMTP relaying, which is for transmission of email from email server to email server. Ports is for default network for web server using HTTP, and port 445 is for Server Message Block, which is also to allow system of the same network to share files and printer over TCP/IP. Therefore, I also believe that R.csv file have a work network.

Investigate IP Addresses

3. Add to your script an option called "-countip" which creates list of distinct IP addresses with their usage counts. Sort the list by the usage count, not by the IP address.
4. Run your countip script on R and O data. Does this inform your answer in [2]?

```
R.csv


Packet: 99142 Bytes: 71683046
  1:         7
  2:         2
  6:     39138
 17:     59995
(10.5.63.230, 59411)
(234.142.142.142, 42981)
(10.5.63.36, 15926)
(10.5.63.231, 12083)
(10.5.63.204, 12003)
(10.5.63.27, 10747)
(10.5.63.22, 9844)
(10.5.63.17, 7574)
(10.5.63.12, 6321)
(10.5.63.11, 4792)
(10.5.63.6, 3526)
(10.5.63.7, 2113)
(10.5.63.25, 1202)
(10.5.63.18, 966)
(10.5.63.202, 788)
(10.5.63.1, 672)
(32.97.255.112, 594)
(10.5.63.28, 542)
(209.67.181.11, 528)
(10.5.63.24, 473)
(10.5.63.23, 431)
```

Since it most showing 10.x.x.x IP addresses, which is class A range that for having a private IP Address, It support my answer that this is a work place.

```
O.csv


Packet: 999914 Bytes: 366325065
  1:      6794
  6:    950654
 17:     38332
 47:      2626
 50:      1484
 89:        24
(192.245.12.221, 288305)
(192.245.12.242, 119218)
(192.245.12.230, 106948)
(66.156.15.246, 63660)
(192.245.12.164, 46186)
(204.69.220.34, 46176)
(192.245.12.234, 41412)
(192.245.12.237, 40626)
(192.245.12.233, 23534)
(192.245.12.225, 21138)
(192.245.12.18, 20614)
(207.182.32.56, 20556)
(207.182.37.125, 16638)
(192.245.12.8, 14110)
(66.110.217.81, 13888)
(192.245.12.231, 13016)
(192.245.12.246, 12150)
(192.245.12.31, 11094)
(204.17.34.117, 10520)
(207.182.40.40, 10014)
(192.245.12.245, 9992)
(204.27.149.191, 9691)
(63.223.5.246, 9691)
(192.245.12.56, 9670)
(66.245.107.161, 8892)
(204.153.45.185, 8826)
```

For O.csv files, I can see the IP address start with 199 and 207 mostly, I then try to use google to see the information of IP Address starts with 207 but have no luck. I then use chat gpt to help me. Overall, the O.csv file have a class C network, which mean this could be a service provider.

what is the class of ip address of 207.x.x.x

The IP address **207.x.x.x** falls into **Class C** of IPv4 addresses.

Explanation:

IPv4 address classes are determined by the first octet (the first number in the address) and are as follows:

- **Class A**: 1 to 126
- **Class B**: 128 to 191
- **Class C**: 192 to 223
- **Class D**: 224 to 239 (Multicast)
- **Class E**: 240 to 255 (Experimental)

Since **207** lies between **192 and 223**, the IP address **207.x.x.x** belongs to **Class C**.

Characteristics of Class C:

- Default subnet mask: **255.255.255.0**
- Number of networks: **2,097,152**
- Hosts per network: **254** (usable addresses)

what does it used for

An IP address in the **207.x.x.x** range, being a **Class C** address, is typically used for **small to medium-sized networks**. Here's more detail:

Uses of Class C IP Addresses:

1. **Commercial Organizations**:
    - Class C IP addresses are commonly allocated to businesses and service providers requiring smaller networks with up to 254 usable IP addresses.
    - Example: Internet Service Providers (ISPs) use them to allocate IPs to small organizations or individuals.
2. **Public Internet Hosting**:
    - Many websites and hosting services operate on IP addresses in this range.
    - Organizations that don't require a large pool of IPs for internal devices often use Class C addresses for web hosting, email servers, and DNS servers.
3. **Network Segmentation**:
    - Class C addresses can be used in scenarios requiring segmentation of networks into smaller, easily manageable subnets.

5. Attempt to determine the network number (network prefix) that seems to dominate the traffic.

    R.csv File: 234.142.142.142 – A multi cast address range

    O.csv File: 66.156.15.246 – AT&T service provider

There are some IP protocols that are typically used between routers or other special networking devices. Traffic from these protocols can identify the infrastructure of the network under observation.

6. Generate sorted output from '-countip' for the IP protocols to identify all the IP addresses that use:

a. GRE (Generic Routing Encapsulation) – this is used to create tunnels between networks with overlapping address spaces. It is also the base protocol for PPTP, a remote access mechanism.
b. IPSEC – this is the protocol that creates virtual private networks, creating an overlay network structure on top of the Internet. Most IPSEC is router-router these days.
c. OSPF – Open Shortest Path First routing protocol. This is the 'standard' routing protocol for Internet routers, allowing them to discover the topology and choose the best routing paths as connections between routers appear and disappear.

- Hint: create a new protocol argument to filter the data to '-countip' to just include lines for these protocols. Alternately, use the GREP pipeline in the example above, for IGMP traffic, and change '2' to the right protocol number.
-
-

```
tintin@MacBook-Pro hw3 % python3 scancsv.py R.csv -countip -GRE


COUNTIP

R.csv


Packet: 99142 Bytes: 71683046
  1:         7
  2:         2
  6:     39138
 17:     59995
tintin@MacBook-Pro hw3 % python3 scancsv.py R.csv -countip -IPSEC


COUNTIP

R.csv


Packet: 99142 Bytes: 71683046
  1:         7
  2:         2
  6:     39138
 17:     59995
tintin@MacBook-Pro hw3 % python3 scancsv.py R.csv -countip -OSPF


COUNTIP

R.csv


Packet: 99142 Bytes: 71683046
  1:         7
  2:         2
  6:     39138
 17:     59995
```

- R.csv Output

```
tintin@MacBook-Pro hw3 % python3 scancsv.py O.csv -countip -OSPF


COUNTIP

O.csv


Packet: 999914 Bytes: 366325065
  1:      6794
  6:    950654
 17:     38332
 47:      2626
 50:      1484
 89:        24
(207.182.35.58, 16)
(207.182.35.49, 12)
(207.182.35.50, 8)
(207.182.35.60, 4)
(207.182.35.47, 4)
(207.182.35.55, 4)
```

```
tintin@MacBook-Pro hw3 % python3 scancsv.py 0.csv -countip -GRE


COUNTIP

0.csv


Packet: 999914 Bytes: 366325065
  1:      6794
  6:    950654
 17:     38332
 47:      2626
 50:      1484
 89:        24
(209.104.16.215, 2567)
(198.182.113.9, 2567)
(209.104.16.58, 59)
(66.134.158.90, 59)
tintin@MacBook-Pro hw3 % python3 scancsv.py 0.csv -countip -IPSEC


COUNTIP

0.csv


Packet: 999914 Bytes: 366325065
  1:      6794
  6:    950654
 17:     38332
 47:      2626
 50:      1484
 89:        24
(198.182.113.1, 690)
(146.216.2.59, 690)
(207.182.35.50, 667)
(128.196.69.2, 613)
(209.104.16.119, 68)
(151.193.130.121, 68)
(12.9.142.163, 42)
(192.70.160.132, 42)
(207.182.45.254, 23)
(207.182.36.178, 19)
(207.182.45.153, 15)
(216.253.194.82, 15)
(204.17.35.131, 12)
(216.133.8.30, 2)
(207.182.36.166, 2)
```

O.csv output

7. Find another network prefix that also seems to be associated with this traffic.

   R.csv : 32.97.255.112 AT&T

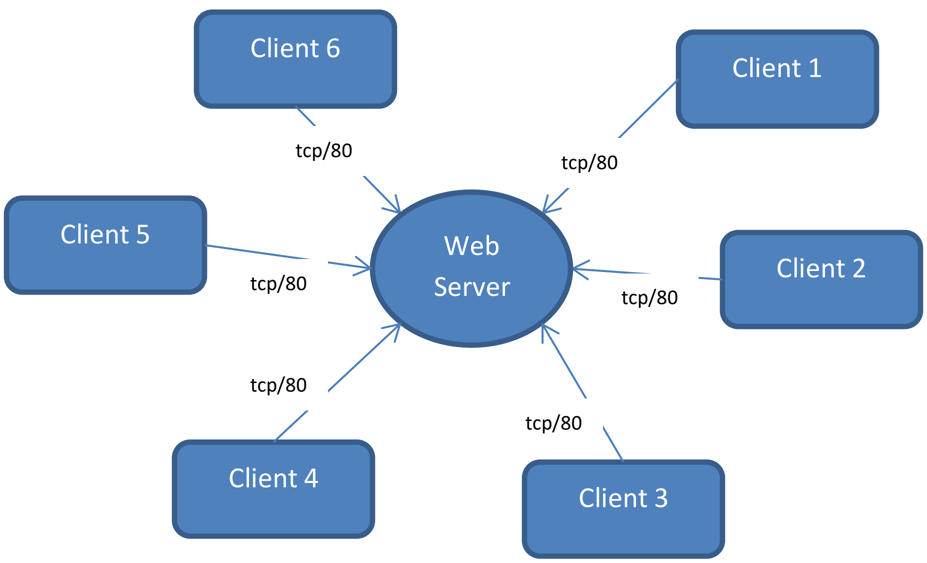
   O.csv: 66.110.217.81 AT&T

8. Does the OSPF information inform your answer to question 2?
   Since OSPF primarily used to LAN, O.csv file could be a LAN network for AT&T service provider.

Find the servers

The server machines are the main assets of each site. Can we find them?

Imagine drawing a diagram for each service, with a line from each client machine that uses the service to the server that provides it. Then the servers look like stars, with clients around each server. Important servers will stick out as stars with lots of spokes.



To find the 'stars' in the diagram, we want to find IP addresses that are the *destination* of many transactions.

9. Add an option to your script '-connto', which counts the number of packets sent to each service (ports 1-1024) on the network. For example, a dictionary maps each *ipdst* to the tuple <*proto*, *dport*>, where *proto* is tcp or udp, based on the IP protocol (6 or 17) and *dport* is the value of tcpdport or udpdport.

- Sort the output by the number of distinct source IP address – source port combinations, so that servers which serve a lot of different connections all cluster at one end of the output.
- For output, generate a summary line that shows, for each destination IP address, how many distinct source IP addresses accessed it, and what ports were referenced:

```
ipdst 1.2.3.4 has 334 distinct ipsrc on ports: udp/53, tcp/80, tcp/443
ipdst 5.6.7.8 has 335 distinct ipsrc on ports: tcp/22, tcp/25
…
```

⇨ Since lab time is short, here are some programming hints you may wish to use:

a. To create the ports output, create a set for each “ipdst” that contains the string “udp/” or “tcp/” appended to the port number (e.g., udp/53, tcp/360). (In languages without an explicit set class, use a dictionary or hash where each entry maps to TRUE or 1)
b. You can use the same trick to compute the distinct ipsrc for the summary line. In this case, put the ipsrc in the string from [a], as *ipsrc-proto/port*. For example, dict['1.2.3.4'] is a set containing 205.9.3.55-udp/53, means that 205.9.3.55 connects to 1.2.3.4 on UDP port 53.
c. You can use leading zeros to make these formats sort correctly without fuss, such as: "tcp/00033" or "udp/00721". *However, you must still arrange for the program output not to have leading zeros.*

10. Run your -connto option on R and O data (<u>ignore anything that ends in .255 – this is a broadcast address</u>). Does this suggest a set of servers to you?
    a. Return the top 20 servers from your ‘connto’ output.

    R.csv File:

```
R.csv


Packet: 99142 Bytes: 71683046
  1:        7
  2:        2
  6:    39138
 17:    59995
ipdst 10.5.63.255 has 33 distinct ipsrc on ports: udp/137, udp/138
ipdst 10.5.63.7 has 23 distinct ipsrc on ports: tcp/135, udp/138, tcp/721, tcp/139, udp/137, tcp/80
ipdst 10.5.63.6 has 19 distinct ipsrc on ports: tcp/22, tcp/110, udp/53, tcp/25
ipdst 10.5.63.230 has 9 distinct ipsrc on ports: udp/137, udp/138, tcp/139, udp/0
ipdst 10.5.63.27 has 4 distinct ipsrc on ports: udp/137, tcp/113, tcp/139
ipdst 10.5.63.22 has 4 distinct ipsrc on ports: tcp/139, tcp/23
ipdst 10.5.63.14 has 4 distinct ipsrc on ports: tcp/113, udp/138, udp/137
ipdst 10.5.63.200 has 3 distinct ipsrc on ports: tcp/80, tcp/139
ipdst 10.5.63.11 has 3 distinct ipsrc on ports: udp/137, tcp/139
ipdst 255.255.255.255 has 2 distinct ipsrc on ports: udp/67, udp/68
ipdst 10.5.63.24 has 2 distinct ipsrc on ports: udp/137, tcp/23
ipdst 10.5.63.231 has 2 distinct ipsrc on ports: udp/137, tcp/139
ipdst 10.5.63.23 has 2 distinct ipsrc on ports: udp/137, udp/138
ipdst 10.5.63.204 has 2 distinct ipsrc on ports: udp/137, udp/138
ipdst 10.5.63.17 has 2 distinct ipsrc on ports: udp/137, tcp/139
ipdst 32.97.255.112 has 1 distinct ipsrc on ports: tcp/80
ipdst 216.101.171.2 has 1 distinct ipsrc on ports: tcp/110
ipdst 209.67.181.20 has 1 distinct ipsrc on ports: tcp/80
ipdst 209.67.181.11 has 1 distinct ipsrc on ports: tcp/80
ipdst 208.10.192.202 has 1 distinct ipsrc on ports: tcp/80
```

O.csv File:

```
ipdst 192.245.12.242 has 1048 distinct ipsrc on ports: udp/137, tcp/22, tcp/135, tcp/25
ipdst 192.245.12.234 has 1009 distinct ipsrc on ports: tcp/22, tcp/135, tcp/25
ipdst 192.245.12.233 has 850 distinct ipsrc on ports: tcp/22, tcp/445, tcp/135, tcp/25
ipdst 192.245.12.230 has 817 distinct ipsrc on ports: tcp/22, tcp/135, tcp/25
ipdst 192.245.12.56 has 721 distinct ipsrc on ports: tcp/22, udp/53, tcp/135
ipdst 192.245.12.7 has 624 distinct ipsrc on ports: tcp/25, tcp/135, udp/123, udp/53, tcp/80, tcp/23
ipdst 192.245.12.221 has 382 distinct ipsrc on ports: tcp/135, udp/123, tcp/113, tcp/80, tcp/25, tcp/139
ipdst 192.245.12.50 has 342 distinct ipsrc on ports: udp/53, udp/13, udp/37
ipdst 192.245.12.8 has 232 distinct ipsrc on ports: tcp/135, tcp/22, udp/123, tcp/110, tcp/143, tcp/993, udp/53, tcp/995, tcp/23, tcp/25
ipdst 192.245.12.52 has 201 distinct ipsrc on ports: udp/53, tcp/53, tcp/135
ipdst 207.182.38.2 has 191 distinct ipsrc on ports: tcp/445, udp/53, tcp/53, tcp/25
ipdst 192.245.12.9 has 82 distinct ipsrc on ports: tcp/1023, tcp/135, tcp/22, tcp/445, tcp/110, tcp/143, udp/53, tcp/993, tcp/995, tcp/25, tcp/465
ipdst 204.153.45.2 has 73 distinct ipsrc on ports: udp/53, tcp/25
ipdst 204.153.45.185 has 69 distinct ipsrc on ports: tcp/80
ipdst 192.245.12.53 has 68 distinct ipsrc on ports: udp/53, tcp/135
ipdst 207.182.32.14 has 65 distinct ipsrc on ports: tcp/25
ipdst 207.182.38.3 has 58 distinct ipsrc on ports: tcp/445, tcp/80, udp/53, tcp/25
ipdst 192.245.12.31 has 42 distinct ipsrc on ports: tcp/135, tcp/80, tcp/445, tcp/25
ipdst 192.245.12.21 has 37 distinct ipsrc on ports: udp/123, tcp/135
ipdst 192.245.12.245 has 35 distinct ipsrc on ports: tcp/135, tcp/110, udp/53, tcp/80, tcp/25
```

b. For the R data, identify the web servers, the printers, the mail servers, the DNS servers

   Web Server 32.97.255.112, 209.67.181.20, 209.67.181.11, and 208.10.192.202 all have traffic on tcp/80 (HTTP), indicating web servers.

   Printer: 10.5.63.7, 10.5.63.23 and 10.5.63.204 has traffic on udp/137, tcp/139, indicating possible printer devices

   Mail Server: 10.5.63.6 has traffic on tcp/25 (SMTP) and tcp/110 (POP3), suggesting it is a mail server

   DNS server: 10.5.63.6 has traffic on udp/53, indicating it might be a DNS server

c. For the O data, identify the mail servers, the pop/imap servers, the DNS servers

   Web Serve: 192.245.12.7, 192.245.12.221, 204.153.45.185, and 207.182.38.3 has traffic on tcp/80, indicate it is a web server

   Printer: 192.245.12.221, 207.182.38.3 has traffic on tcp/139, indicating it could be a printer or file server.

   Mail Server: 192.245.12.242, 192.245.12.7, 192.245.12.9 has traffic on tcp/25 (SMTP), which indicates it could be a mail server.

   DNS: 192.245.12.56, 192.245.12.50, 207.182.38.2, 204.153.45.2 has traffic on udp/53, indicating it could be a DNS server

11. Update your answer from [5] based on this information.
    I believe R.csv is a work place with multicast network and O.csv can also be a work place for a service provider or a server that use LAN Network.

For More Fun, if you finish early:

In the O data, some of the GRE lines actually have the following form:

```
len,greproto,ipsrc,ipdst,iproto,iipsrc,iipdst
```

where greproto is 47, and iproto, iipsrc and iipdst describe the packet being transmitted <u>inside</u> the GRE tunnel.

Extend your program to identify these lines and see what you can learn from this additional information.