Tu Lam

CS 373 (Defense Against the Dark Arts)

Dr. Bram Lewis

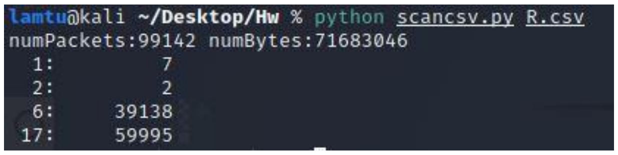
November 11th, 2021

Homework #3

In the case for this homework, we were task to explore the VM using a script and continue writing the script to look at the data in the ***R*** and ***O*** file. This script will be a way for us to explore the concept behind the network security in cybersecurity. Through this, we will get a sense of the network going through the cybersecurity background via these two files that were given.

First, the homework gave us a tar file and in there, there are a ***bunch of script*** and ***CSV*** files that we can look at the network security for this week homework. Then we have a lab file where it gave us the background of network security and learn about the ***TCP***, ***IP protocol***, ***UDP***, and more. In there, there is also couple instruction we will use and follow to answer and explore what the lab offer for this homework as well. Below, we will look at individual questions and answer them in order. In total, there are ***11 questions*** to answer with an extra challenge if we want to do it or not.

Next, after extracting all the files that were given in the homework, I was able to run the python file and look at the ***R.csv*** file using the ***scancsv.py*** file. Through looking through this, I found that the ***IP protocol*** have number align with ***1, 2, 6, 17***. A figure below will show those number and what do they print out to be. In the lab manual, I found out that each number represent a protocol. For example, 1 represent the ***ICMP***, 2 represent the ***IGMP***, and so on. Then we move onto locating the ***IP address*** using the grep function and then we will move onto the questions that are ask in the lab.

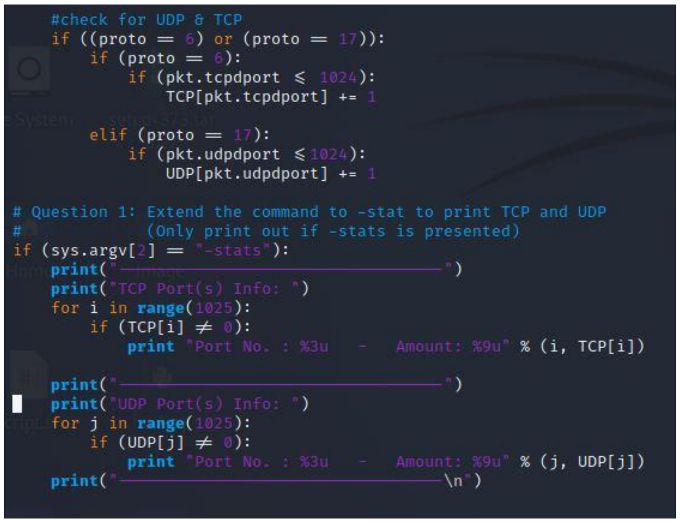


```
lamtu@kali ~/Desktop/Hw % python scancsv.py R.csv
numPackets:99142 numBytes:71683046
  1:         7
  2:         2
  6:     39138
 17:     59995
```

Figure #1: *Showing what R.csv offer when run the program*

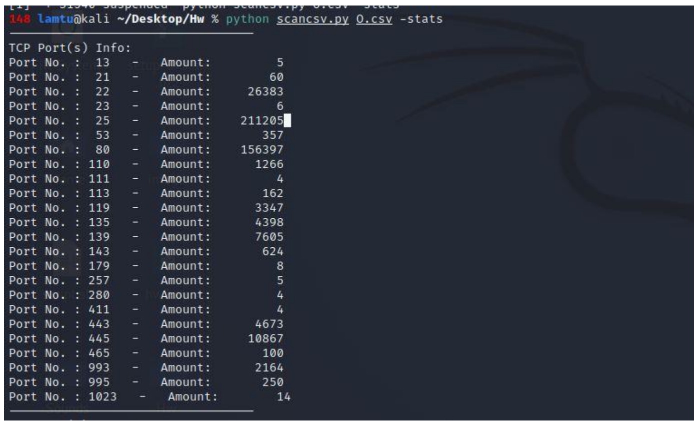
1. *Extend your script's statistics gathering to count the use of all well-known destination port numbers for TCP and UDP (ports 1-1024). For example, you should be able to look up in your output how many TCP packets have destination port 80 and how many UDP packets have destination port 53. Run your 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).*

Answer: To achieve the script to display the ***TCP*** and ***UDP*** if looking from port ranging from 1-1024 (In this case just using ***80*** and ***53***), we can go inside the scancsv.py to look and add command to help print out the data if a port is being called. To get the correct data printed out, I would need to look at the ***R*** and ***O*** file to see the layout of the CSV file look like. With the trial and guess and looking through the file, below are couple figures to help represent how to see ***TCP*** and ***UDP*** content.

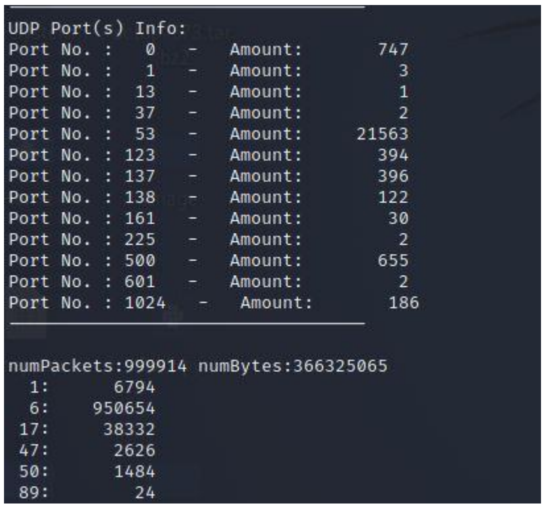


```
    #check for UDP & TCP
    if ((proto == 6) or (proto == 17)):
        if (proto == 6):
            if (pkt.tcpdport ≤ 1024):
                TCP[pkt.tcpdport] += 1

        elif (proto == 17):
            if (pkt.udpdport ≤1024):
                UDP[pkt.udpdport] += 1

# Question 1: Extend the command to -stat to print TCP and UDP
#              (Only print out if -stats is presented)
if (sys.argv[2] == "-stats"):
    print("————————————————————————————————")
    print("TCP Port(s) Info: ")
    for i in range(1025):
        if (TCP[i] ≠ 0):
            print "Port No. : %3u   -   Amount: %9u" % (i, TCP[i])

    print("————————————————————————————————")
    print("UDP Port(s) Info: ")
    for j in range(1025):
        if (UDP[j] ≠ 0):
            print "Port No. : %3u   -   Amount: %9u" % (j, UDP[j])
    print("————————————————————————————————\n")
```

Figure #2: *Display the code to implement the printout of TCP and UDP*

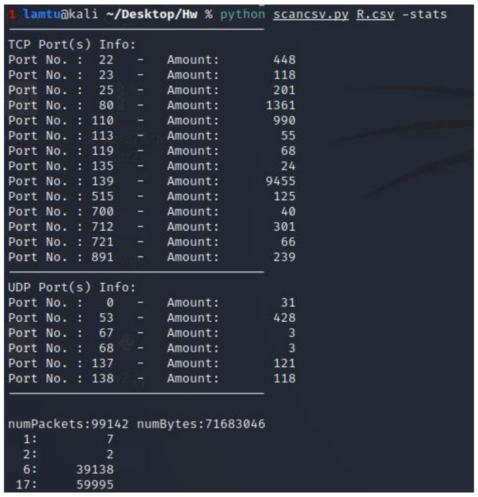


```
148 lamtu@kali ~/Desktop/Hw % python scancsv.py O.csv -stats
————————————————————————————————
TCP Port(s) Info:
Port No. :  13   -   Amount:        5
Port No. :  21   -   Amount:       60
Port No. :  22   -   Amount:    26383
Port No. :  23   -   Amount:        6
Port No. :  25   -   Amount:   211205
Port No. :  53   -   Amount:      357
Port No. :  80   -   Amount:   156397
Port No. : 110   -   Amount:     1266
Port No. : 111   -   Amount:        4
Port No. : 113   -   Amount:      162
Port No. : 119   -   Amount:     3347
Port No. : 135   -   Amount:     4398
Port No. : 139   -   Amount:     7605
Port No. : 143   -   Amount:      624
Port No. : 179   -   Amount:        8
Port No. : 257   -   Amount:        5
Port No. : 280   -   Amount:        4
Port No. : 411   -   Amount:        4
Port No. : 443   -   Amount:     4673
Port No. : 445   -   Amount:    10867
Port No. : 465   -   Amount:      100
Port No. : 993   -   Amount:     2164
Port No. : 995   -   Amount:      250
Port No. : 1023   -   Amount:        14
————————————————————————————————
```



```
————————————————————————————————
UDP Port(s) Info:
Port No. :   0   -   Amount:      747
Port No. :   1   -   Amount:        3
Port No. :  13   -   Amount:        1
Port No. :  37   -   Amount:        2
Port No. :  53   -   Amount:    21563
Port No. : 123   -   Amount:      394
Port No. : 137   -   Amount:      396
Port No. : 138   -   Amount:      122
Port No. : 161   -   Amount:       30
Port No. : 225   -   Amount:        2
Port No. : 500   -   Amount:      655
Port No. : 601   -   Amount:        2
Port No. : 1024   -   Amount:        186
————————————————————————————————

numPackets:999914 numBytes:366325065
  1:      6794
  6:    950654
 17:     38332
 47:      2626
 50:      1484
 89:        24
```

Figure #3: *Display printout of TCP & UDP of O.csv*



```
1 lamtu@kali ~/Desktop/Hw % python scancsv.py R.csv -stats
————————————————————————————————
TCP Port(s) Info:
Port No. :  22   -   Amount:      448
Port No. :  23   -   Amount:      118
Port No. :  25   -   Amount:      201
Port No. :  80   -   Amount:     1361
Port No. : 110   -   Amount:      990
Port No. : 113   -   Amount:       55
Port No. : 119   -   Amount:       68
Port No. : 135   -   Amount:       24
Port No. : 139   -   Amount:     9455
Port No. : 515   -   Amount:      125
Port No. : 700   -   Amount:       40
Port No. : 712   -   Amount:      301
Port No. : 721   -   Amount:       66
Port No. : 891   -   Amount:      239
————————————————————————————————
UDP Port(s) Info:
Port No. :   0   -   Amount:       31
Port No. :  53   -   Amount:      428
Port No. :  67   -   Amount:        3
Port No. :  68   -   Amount:        3
Port No. : 137   -   Amount:      121
Port No. : 138   -   Amount:      118
————————————————————————————————

numPackets:99142 numBytes:71683046
  1:         7
  2:         2
  6:     39138
 17:     59995
```

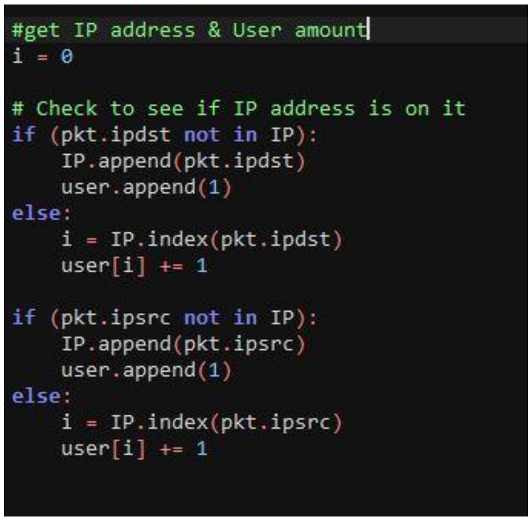
Figure #4: *Display the TCP & UDP of R.csv*

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)*

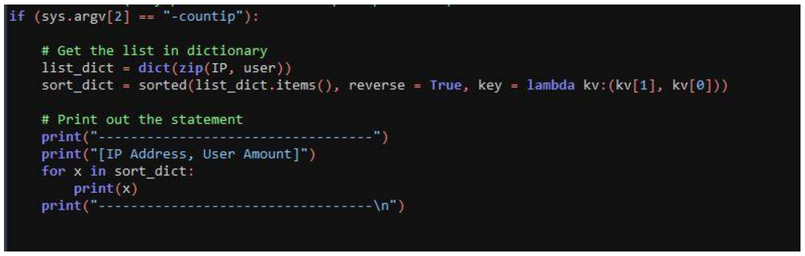
Answer: Through looking through both ***R*** and ***O*** file. I saw that TCP in the ***R*** file have 9455 users from ***port 139*** while ***port 53*** have around 400 users. From that, we can gather through online help that ***port 53 deals with DNS*** while ***port 139 deal with NetBIOS service***. Then move onto ***O*** file, I see that TCP ***port 25*** and UDP at ***port 53*** have the most traffic. At ***port 25 is SMTP client*** and ***port 53*** have discuss in the previous statement for R file. And these are just some examples I will be using to look at the network that people are connected to.

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.*

Answer: Now, we then add in a command called ***"-countip"*** into the argument and it will list distinct IP addresses with usage counts. With this, we can determine if the last question we answer are correct. Below will be some more screenshot of code and display of this new command.

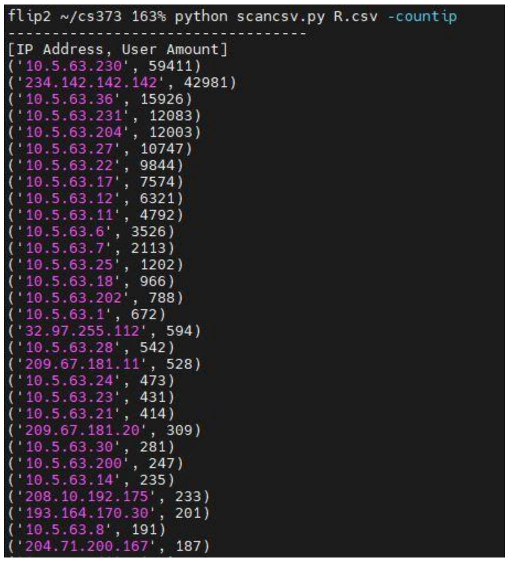


```
#get IP address & User amount
i = 0

# Check to see if IP address is on it
if (pkt.ipdst not in IP):
    IP.append(pkt.ipdst)
    user.append(1)
else:
    i = IP.index(pkt.ipdst)
    user[i] += 1

if (pkt.ipsrc not in IP):
    IP.append(pkt.ipsrc)
    user.append(1)
else:
    i = IP.index(pkt.ipsrc)
    user[i] += 1
```



```
if (sys.argv[2] == "-countip"):

    # Get the list in dictionary
    list_dict = dict(zip(IP, user))
    sort_dict = sorted(list_dict.items(), reverse = True, key = lambda kv:(kv[1], kv[0]))

    # Print out the statement
    print("----------------------------------")
    print("[IP Address, User Amount]")
    for x in sort_dict:
        print(x)
    print("----------------------------------\n")
```

<u>Figure #5</u>: *Code of implementing the "-countip"*



```
flip2 ~/cs373 163% python scancsv.py R.csv -countip
---------------------------------
[IP Address, User Amount]
('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)
('10.5.63.21', 414)
('209.67.181.20', 309)
('10.5.63.30', 281)
('10.5.63.200', 247)
('10.5.63.14', 235)
('208.10.192.175', 233)
('193.164.170.30', 201)
('10.5.63.8', 191)
('204.71.200.167', 187)
```

Figure #6: *The output of "-countip" using R file*

4. *Run your "-countip" script on R and O data. Does this inform your answer in [2]?*

Answer: With looking at the ***-countip*** command that was implement in question 3, it looks like it ***does inform*** what question 2 when it was answer. Most of the user uses the port are listed on the IP address in high demand, but at the same time, we can't really tell much if the port is exactly the port that people visit.

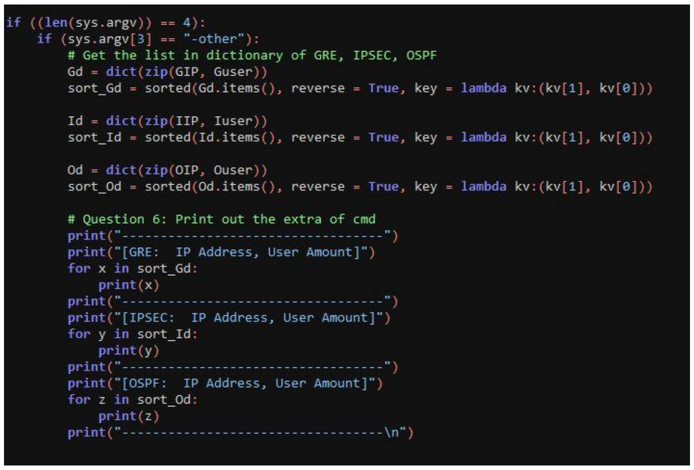
5. *Attempt to determine the network number (network prefix) that seems to dominate the traffic.*

Answer: Looking in both files, the most ***R*** file get dominated with the traffic that start with ***10.5.63.xx*** prefix. While the ***O*** file have the traffic of ***192.245.12.xx*** prefix.

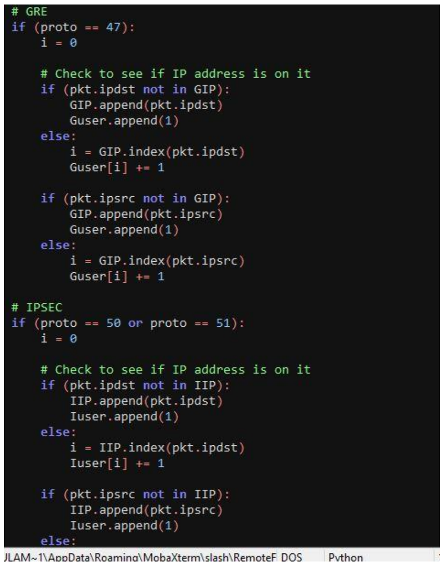
6. *Generate sorted output from '-countip' for the IP protocols to identify all the IP addresses that use:*

> **a**. *GRE (Generic Routing Encapsulation)*
> **b**. *IPSEC*
> **c**. *OSPF*

Answer: Below will be some figures showing the general output of the ***-countip*** with couple usage of the options that are given. Through the file, R file did not have any output while O file does and to use the command, add in another command ***"-other"*** to print out.



```python
if ((len(sys.argv)) == 4):
    if (sys.argv[3] == "-other"):
        # Get the list in dictionary of GRE, IPSEC, OSPF
        Gd = dict(zip(GIP, Guser))
        sort_Gd = sorted(Gd.items(), reverse = True, key = lambda kv:(kv[1], kv[0]))

        Id = dict(zip(IIP, Iuser))
        sort_Id = sorted(Id.items(), reverse = True, key = lambda kv:(kv[1], kv[0]))

        Od = dict(zip(OIP, Ouser))
        sort_Od = sorted(Od.items(), reverse = True, key = lambda kv:(kv[1], kv[0]))

        # Question 6: Print out the extra of cmd
        print("-----------------------------------")
        print("[GRE:  IP Address, User Amount]")
        for x in sort_Gd:
            print(x)
        print("-----------------------------------")
        print("[IPSEC:  IP Address, User Amount]")
        for y in sort_Id:
            print(y)
        print("-----------------------------------")
        print("[OSPF:  IP Address, User Amount]")
        for z in sort_Od:
            print(z)
        print("-----------------------------------\n")
```



```python
# GRE
if (proto == 47):
    i = 0

    # Check to see if IP address is on it
    if (pkt.ipdst not in GIP):
        GIP.append(pkt.ipdst)
        Guser.append(1)
    else:
        i = GIP.index(pkt.ipdst)
        Guser[i] += 1

    if (pkt.ipsrc not in GIP):
        GIP.append(pkt.ipsrc)
        Guser.append(1)
    else:
        i = GIP.index(pkt.ipsrc)
        Guser[i] += 1

# IPSEC
if (proto == 50 or proto == 51):
    i = 0

    # Check to see if IP address is on it
    if (pkt.ipdst not in IIP):
        IIP.append(pkt.ipdst)
        Iuser.append(1)
    else:
        i = IIP.index(pkt.ipdst)
        Iuser[i] += 1

    if (pkt.ipsrc not in IIP):
        IIP.append(pkt.ipsrc)
        Iuser.append(1)
    else:
```

<u>Figure #7</u>: *Display some codes of implementing Question #6*

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

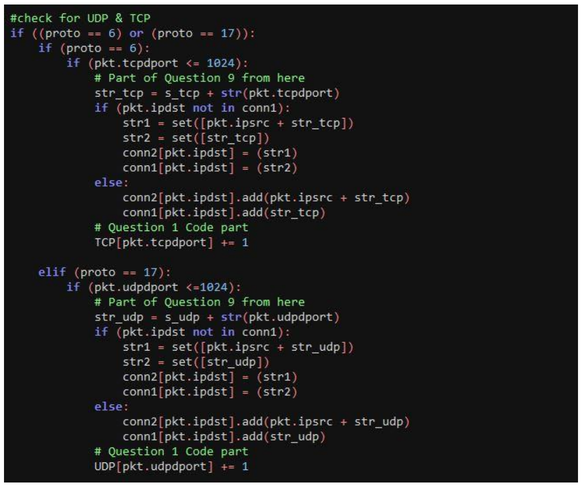
Answer: When determine the network prefix, we will look at the output for the ***O.csv***. Since the ***R.csv*** was blank, I believe the clear network prefix that can be seen is the 207.182.xx.xx are in the ***R.csv*** file.

8. *Does the OSPF information inform your answer to question 2?*

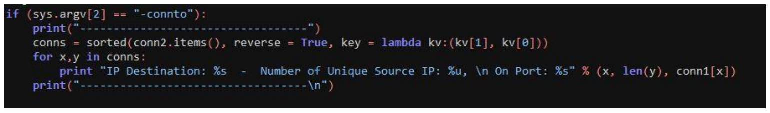
Answer: When talking about ***OSPF***, this is the 'standard' routing protocol for Internet routers, allowing them to discover the topology and choose the best routing paths, we can find that R.csv file does not have any prefix network route from the last answer question, believe to be determined that it is more onto sharing a route rather than mapping one. From this, we can see that question 2 display the center network that most of the IP addresses share when routing to Internet router.

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.*

Answer: Below are some images of the implementation of ***-connto*** command line.

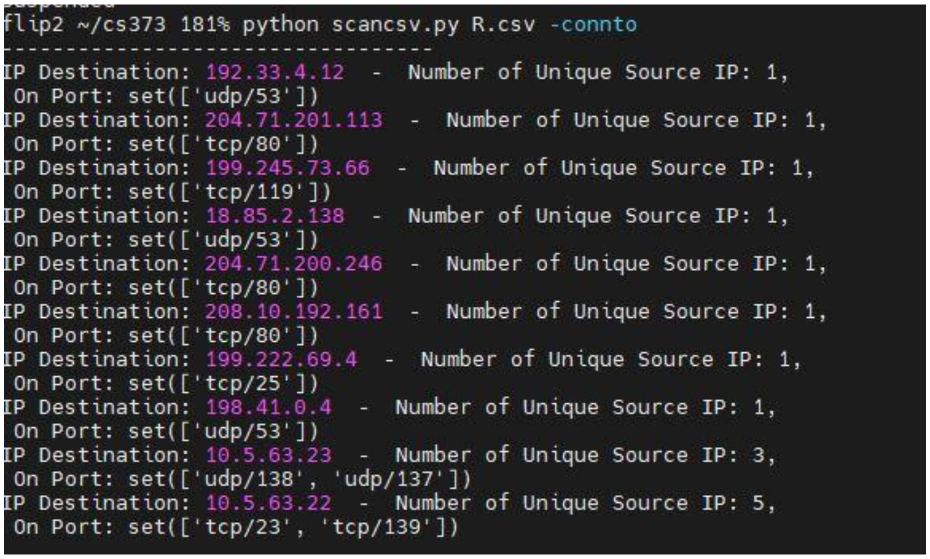


```
#check for UDP & TCP
if ((proto == 6) or (proto == 17)):
    if (proto == 6):
        if (pkt.tcpdport <= 1024):
            # Part of Question 9 from here
            str_tcp = s_tcp + str(pkt.tcpdport)
            if (pkt.ipdst not in conn1):
                str1 = set([pkt.ipsrc + str_tcp])
                str2 = set([str_tcp])
                conn2[pkt.ipdst] = (str1)
                conn1[pkt.ipdst] = (str2)
            else:
                conn2[pkt.ipdst].add(pkt.ipsrc + str_tcp)
                conn1[pkt.ipdst].add(str_tcp)
            # Question 1 Code part
            TCP[pkt.tcpdport] += 1

    elif (proto == 17):
        if (pkt.udpdport <=1024):
            # Part of Question 9 from here
            str_udp = s_udp + str(pkt.udpdport)
            if (pkt.ipdst not in conn1):
                str1 = set([pkt.ipsrc + str_udp])
                str2 = set([str_udp])
                conn2[pkt.ipdst] = (str1)
                conn1[pkt.ipdst] = (str2)
            else:
                conn2[pkt.ipdst].add(pkt.ipsrc + str_udp)
                conn1[pkt.ipdst].add(str_udp)
            # Question 1 Code part
            UDP[pkt.udpdport] += 1
```



```
if (sys.argv[2] == "-connto"):
    print("---------------------------------")
    conns = sorted(conn2.items(), reverse = True, key = lambda kv:(kv[1], kv[0]))
    for x,y in conns:
        print "IP Destination: %s  -  Number of Unique Source IP: %u, \n On Port: %s" % (x, len(y), conn1[x])
    print("---------------------------------\n")
```

Figure #8: *Display code of the Question 9 asking to implement the -connto*

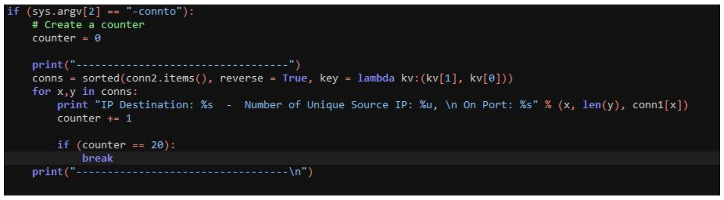


```
flip2 ~/cs373 181% python scancsv.py R.csv -connto
----------------------------------
IP Destination: 192.33.4.12  -  Number of Unique Source IP: 1,
 On Port: set(['udp/53'])
IP Destination: 204.71.201.113  -  Number of Unique Source IP: 1,
 On Port: set(['tcp/80'])
IP Destination: 199.245.73.66  -  Number of Unique Source IP: 1,
 On Port: set(['tcp/119'])
IP Destination: 18.85.2.138  -  Number of Unique Source IP: 1,
 On Port: set(['udp/53'])
IP Destination: 204.71.200.246  -  Number of Unique Source IP: 1,
 On Port: set(['tcp/80'])
IP Destination: 208.10.192.161  -  Number of Unique Source IP: 1,
 On Port: set(['tcp/80'])
IP Destination: 199.222.69.4  -  Number of Unique Source IP: 1,
 On Port: set(['tcp/25'])
IP Destination: 198.41.0.4  -  Number of Unique Source IP: 1,
 On Port: set(['udp/53'])
IP Destination: 10.5.63.23  -  Number of Unique Source IP: 3,
 On Port: set(['udp/138', 'udp/137'])
IP Destination: 10.5.63.22  -  Number of Unique Source IP: 5,
 On Port: set(['tcp/23', 'tcp/139'])
```

Figure #9: *The output of the -connto output*

10. *Run your -connto option on R and O data (ignore anything that ends in .255 – this is a broadcast address). Does this suggest a set of servers to you?*

Answer: Below are some results printing out in R file that limit in 20 lines. The O files was too big to run, but the code should work for both file and ***-connto*** above in Figure #9 show an example result.



```
if (sys.argv[2] == "-connto"):
    # Create a counter
    counter = 0

    print("---------------------------------")
    conns = sorted(conn2.items(), reverse = True, key = lambda kv:(kv[1], kv[0]))
    for x,y in conns:
        print "IP Destination: %s  -  Number of Unique Source IP: %u, \n On Port: %s" % (x, len(y), conn1[x])
        counter += 1

        if (counter == 20):
            break
    print("---------------------------------\n")
```

Figure #10: *The code of Question 10*

11. *Update your answer from [5] based on this information.*

Answer: From the cmd implement from above, I ***really cannot say much*** has changed looking at the outputs from the R.csv file and the O.csv. The information displays the common prefix, but the information still remains the same with the command line added.