Arthur Liou
CS373

Week 6, Lab 2 Writeup

Prompt: Script and output of writeup. Screenshots are fine (Me: Script Screenshots at very end)

- VM Instructions: https://oregonstate.instructure.com/courses/1712159/pages/vm-instructions?module_item_id=18495782

Running Commands:

- python scancsv.py R.csv
- python scancsv.py O.csv (larger file)
- grep -e len -e '^[0-9][0-9]*,2,' R.csv

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).

   See script / screenshot

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)

   Response: The main functions on each network:
   TCP:
   22: 10 (SSH, secure file transfers)
   80: 524 (HTTP)
   UDP:
   53: 83 (Domain transfers, Domain Name System)

   This could be interpreted in multiple ways. I originally thought this was a home network, then a data center once I figured out UDP 53 was for DNS and domain transfers. Then thinking more holistically, it makes more sense for this to be an ISP

   It is an ISP network. The inclusion of UDP port 53 means that a client sends queries to a DNS server and indicates that domain transfers are invoked. The sheer number of HTTP requests indicates web page browsing, so ISP. The 10 SSH secure file transfers seem a little low for an ISP, but it's like a student SSHing into OSU or sending work files securely. Without better oversight into the overall numbers, it's hard to say if these numbers are reasonable or not

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.

   See script

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

   Yes

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

   10.5.63.xxx

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:

- 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.
- 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.
- 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.

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

   234.xxx.xxx.xxx, usually like 234.142.142.142

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

   It does a little bit.

Find the servers

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:

- 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)
- 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.
- 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.*

See script

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?

- Return the top 20 servers from your 'connto' output.
- For the R data, identify the web servers, the printers, the mail servers, the DNS servers
- For the O data, identify the mail servers, the pop/imap servers, the DNS servers

Yes

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

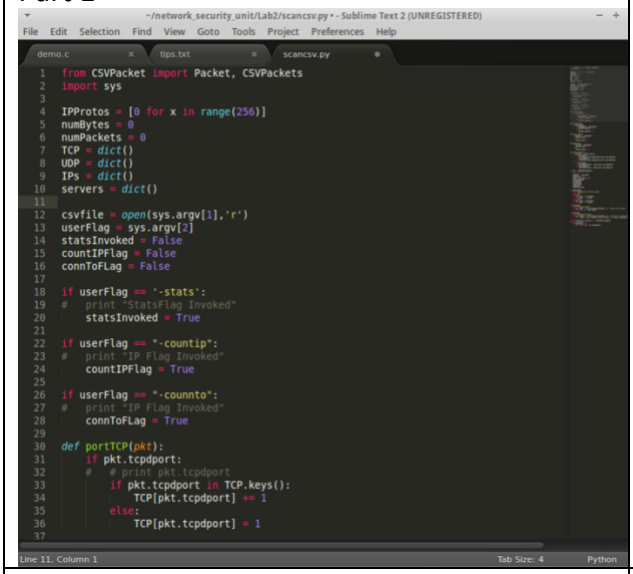
Ok

Writeup / Reflection on Lab

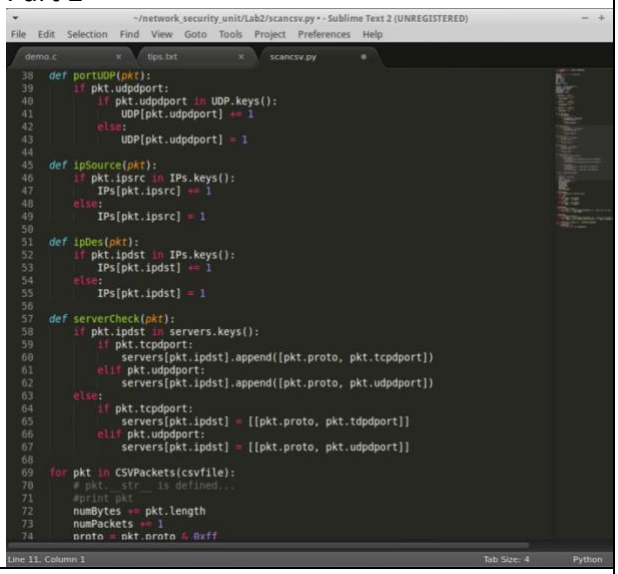
- While I appreciated going through the lab and learning about the different parts (no issues from a learning standpoint), this was a completely horrendous lab.
- We didn't need to work in the VM. If we were provided the files outside of the VM, I could have cut my working time by more than 70%. From what I've heard from our other classmates, we've had similar disapprovals of how this lab was presented to us.
- I found the instructions, very in-depth in certain areas, very lacking in others. Getting set up and going took way longer than it should have (looking back).
- While not an issue for me since I knew Python before this course, I'm sure there are many students who do not know Perl or Python, which would add to the time taken and confusion from students.

Script Screenshots:

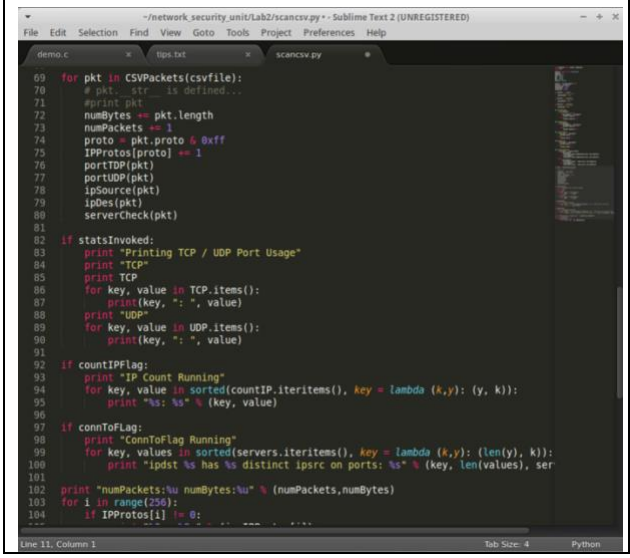
Part 1



Part 2



Part 3



Part 4