Project 4

Network Monitoring

Fall, 2024

Introduction

(If you prefer a shorter version of the project description, jump to the Project Tasks section(page 4) and pay attention to the Deliverable section(page 5-6). However, the first few pages of this document might help you to grasp a better idea of what you are going to encounter.)

Video Introduction(optional)

You can find the video introduction of the project under project 4 folder on Canvas. Alternatively, you can view the video here as well: https://youtu.be/GHNZ6cBv3nU
This video gives you a brief overview of the structure of the project and how to run the commands. For more details, please refer to the project traffic definitions(page2-3) and tasks(page 4) section.

Goals

The goal of this project is to introduce students to the techniques that help to differentiate malicious and legitimate network traffic. This is a task that network operators perform frequently. In this project, the students are provided with samples of malicious and legitimate traffic. They can observe how each type of traffic looks. In the project folder, there is a pcap file that contains network traffic that originates from multiple hosts in the same network. This pcap file is a mixture of legitimate and malicious traffic. The students are asked to investigate the pcap file in network tools such as WireShark. Finally, the students are asked to use Snort and write their own Snort rules, which will differentiate malicious and legitimate traffic. In summary, the students are introduced to:

- Observing pcap samples of legitimate and malicious network traffic
- Using Snort and writing Snort rules to differentiate legitimate traffic from malicious traffic

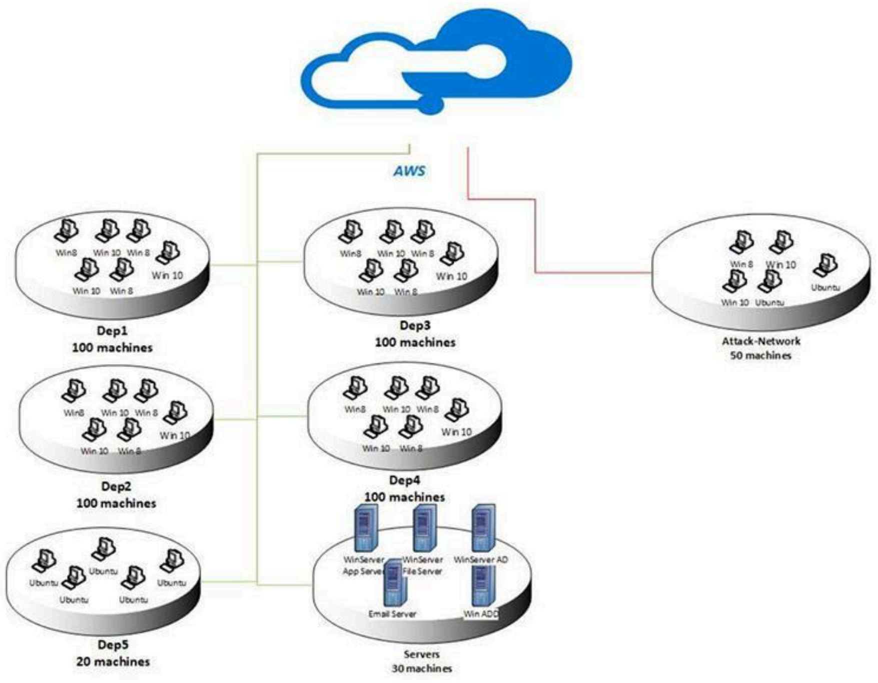


Figure 1: *Network setup for traffic collection.*

Definitions and Traffic Collection Set-up

In this assignment, there are four attack scenarios. For each attack, a scenario is defined based on the implemented network topology, and the attack is executed from one or more machines outside the target network. Figure 1 shows the implemented network, which is a common LAN network topology on the AWS computing platform. The hosts are behind a NAT, and their IP addresses belong to a single /16: **172.31.0.0/16**. It also shows a visual representation of the network and our traffic collection set-up.

Types of attacks:

(i) Distributed Denial of Service (DDoS):

In a DDoS, attackers usually keep making full TCP/UDP connections to the remote server. They keep the connection open by sending valid HTTP requests to the server at regular intervals but also keep the sockets from closing. Since any Web server has a finite ability to serve connections, it will only be a matter of time before all sockets are used up and no other connection can be made.

The possible types of DDoS attacks in the traffic:
(a) GoldenEye Denial of Service
(b) Low Orbit Ion Cannon (LOIC)

A short Video that has some tips for DDOS attacks. The color coding method mentioned in this video is not always true. I would suggest to focus more on the frequency(package/second) and do a comparison with the TCP traffic you observe and the UDP traffic you observe. (if you are not sure about what filter to use in your snort rule, jump to 5:50 in the video. It's also perfectly fine if you don't use the detection filter but get 100 on Gradescope. There are multiple ways to complete this task.). Since the packets have randomized time deltas, we do not recommend using frame.time_relative to identify DoS attacks. DDOS might be the most challenging part of the project here. Feel free to share some other tips you find helpful for your DDOS on Ed without giving out your answer!

(ii) Bruteforce:

FTP/SSH is attacked via a Kali Linux machine(the attacker machine), and Ubuntu 14.0 system is the victim machine. There is a large dictionary that contains 90 million words that were used for the list of passwords to brute force.

It is your task to identify which one(FTP or SSH) of them is present in the evaluation *pcap given to you. A 4 mins Video that has some useful tips for Bruteforce attack. We do not provide you with the sample pcap for SSH. However, we encourage you to do some research on the differences between FTP traffic and SSH traffic. Also, the 4 mins video might include some hints.*

(iii) Web Attacks:
There are 3 possible web attacks, **one** of which would be present in your pcap.
(a) DVWA-based: Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is vulnerable. An attacker might try to hijack it.
(b) XSS-based: An attacker might try to launch an XSS attack.
(c) SQL Injection: An attacker might try an SQL injection attack.

(We do not provide a video for webAttack or Botnet, but the sample pcaps we provided should be sufficient to guide you through this part of the project. Tips: Look at the contents of the traffic, that's the key!)

(iv) Botnet:
Zeus is a trojan horse malware that runs on Microsoft Windows. It might be presented in the pcap. It can be used to carry out many malicious and criminal tasks and it is often used to steal banking information by man-in-the-browser keystroke logging and form grabbing. It is used to install the Crypto-Locker ransomware as well. Zeus spreads mainly through drive-by downloads and phishing schemes. **The Ares botnet** might also be presented in the pcap. It is an open-source botnet and has the following capabilities:
(a) remote cmd.exe shell
(b) persistence
(c) file upload/download
(d) screenshot
(e) keylogging

***Either** Zeus **or** Ares could be present in your* evaluation.*pcap, it is your task to identify which one.*

Sample traffic: For each type of traffic mentioned above, we provide a sample of that category/type of traffic. These samples are only for **illustration** purposes. These samples are *only examples*, and they are **not** the same as the actual traffic that is included in the evaluation pcap, which the students will need to label.

- **Legitimate background traffic:**
  For this exercise, we assume normal traffic to include HTTP, DNS. An example of normal (attack free) traffic can be found in:
  - sample_background.pcap
- **BruteForce:**
  - sample_bruteforce_ftp.pcap
  - We do not provide a sample bruteforce ssh pcap. If you see that in the folder, you can ignore that file.
- **Botnet:**
  The host generates this traffic *explicitly* to communicate with a C&C server. The host communicates with the C&C server to receive commands, updates, *etc.*
  - sample_bot.pcap
- **Web Attack:**
  - sample_web.pcap
  - sample_xss.pcap
  - sample_sqlinjection.pcap.
- **DDoS:**

- We do **not** provide a sample. Please look at the example Snort rules on DDoS in the resources section.

Important Links

- Project VM (cs6262_p4.ova) - link
- Packet capture file (evaluation.pcap) - link
- Sample pcaps - link
- Local set-up - link **(optional)**
- Project walkthrough - link
- DdoS analysis example - link
- Bruteforce analysis/ Using IO graphs examples - link

Project Tasks (100 points):

The goals:

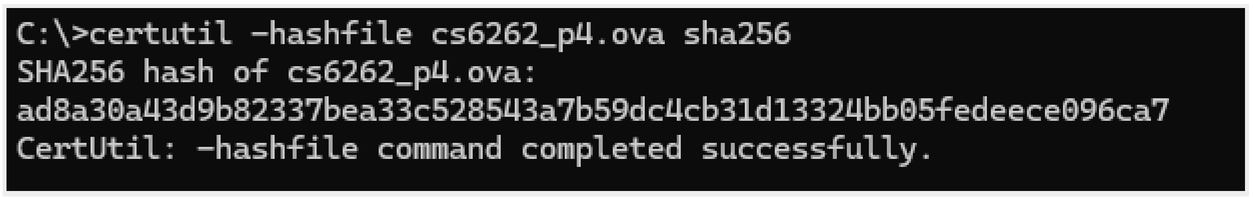
(i) Explore the given pcaps in Wireshark and identify the attack traffic patterns.
(ii) Write Snort rules to raise alerts to identify the attacks.

Towards this goal, please follow the tasks below:

- **Don't panic,** we provide all the commands you need and all the steps you need to follow in this section. Plenty of tips and guides on page 2-3.
- **Install Wireshark** in your local machine
  - (we provide a VM but **we recommend inspecting the pcaps via Wireshark on your local machine** – instead of the VM as it is very CPU and RAM intensive).
- **Download**: The vm from the links posted in the "**Important Links**" section. In addition, we also provide the evaluation.pcap so that it can be inspected on your local machine.
  - IMPORTANT - We also provide you with the files required for completing the project on your local system. The local set-up can be found in the "**Important Links**" section and contains,
    - The Snort configuration file snort.lua
    - The python file cal_unique_connection_2022.py
    - The bash script run_eval.sh

    If you would like to download the project resources for a local setup, **additional installation (Snort 3 and Wireshark) will be required**. We will not be able to provide support for configuring the settings if you set it up locally(You will need to copy /usr/local/etc/snort/snort.lua to your local settings). However, **we recommend you to run the snort rule inside our VM**(cs6262_project4.ova) since it has the snort set up for you already.

  SHA256 hash:

```
C:\>certutil -hashfile cs6262_p4.ova sha256
SHA256 hash of cs6262_p4.ova:
ad8a30a43d9b82337bea33c528543a7b59dc4cb31d13324bb05fedeece096ca7
CertUtil: -hashfile command completed successfully.
```

- **Import** the VM from the link provided. **Login to the VM using: login: student, password: project4**
  - **If you are not able to see the login prompt, try minimizing the window.**
- **Locate** the pcap files on your desktop. You will find the evaluation.pcap. In addition, we provide some sample pcaps in the folder as well. You can create a eval.rules file in this folder to write the snort rules after observing the traffic. Please pay attention to the deliverable section for what is required for gradescope submission.
- **Make observations** on the pcaps. Observe the sample pcaps to get an idea about how each type of

malicious traffic looks like. You can use Wireshark or tshark to isolate some traffic. For example, in Wireshark, you can apply display filters. For e.g., tcp (to display only TCP traffic), ip.addr == 10.0.0.1 (to display traffic that originates from or is destined to this IP address). Also, you can combine filters using or/and.

You should use the attack descriptions above – to understand how these attacks should look like in network traffic.

Note: we recommend keeping everything on the Desktop or the home directory, as some of the scripts will be there.

- **Write Snort rules** – Create a new txt file and name it **eval.rules**. Write your snort rules in eval.rules. keep in mind, we are using **Snort3**, and not Snort2 – please make sure you use the Snort version installed in the VM.
  A example snort rule is

```
alert tcp 172.1.2.3 any -> any any
(
    msg:"A packet from 172.1.2.3 to any ip:port";
    sid:1;
)
```

- **Run Snort rules**:
  A script ‘*run_eval.sh*’ on the virtual machine/in the local setup is a helper script to automate the execution of the commands mentioned below.

  Inside the run_eval.sh, we have the following snort command. The following snort command will help you to generate the json file. If you want to run the command individually by yourself instead, be aware that there is a “.” included at the end of the command. You might need to modify the directory path depending on where your files are located. (Change the directory if necessary)

```
snort -c /usr/local/etc/snort/snort.lua -r ~/Desktop/evaluation.pcap -R ~/Desktop/eval.rules -s 65535 -k none -l .
```

- **Regardless of which method you use to run the rules, please use EXACTLY ONE of the following strings as the alert message in the Snort rule:**
  1. DdoS
  2. Bruteforce
  3. WebAttack
  4. Botnet

  For example, if you are writing a rule to detect ssh brute force, then the alert message should be “Bruteforce”. **This will be used to grade your result – getting this part wrong can lead to a point loss.**
  (Page 2-3 in the write up has more details and some hints about the traffics.)

- **Run** `` `python3 ~/Desktop/cal_unique_connection_2022.py alert_json.txt` `` to check the unique connections.(Change the directory if necessary)

  Common errors after running cal_unique_connection_2022.py::

1. **JSONDecoderError** - This occurs when the file is too big for python to handle. This also means that you have too many false positives and your snort rule needs to be made more precise.
2. **Comments: {'unknown message: abc'}** - This occurs when your snort alert raises a message that cal_unique_connection_2022.py doesn't expect. It has to be one of: 1. "DdoS" 2. "Bruteforce" 3. "WebAttack" 4. "Botnet". The grader/script is looking for the attack types to be spelled/capitalized exactly the way we described.
3. If you **don't see "Attack type: ..., unique connections: ..."** for a particular attack but see it for others - This means your snort rule was too strict/incorrect and filtered out all connections.

Deliverables/Rubric:

For this project, you should submit **Two files on Gradescope**

- **rules.py** - This is where you paste your Snort rules. This is for us to grade your rules! Just copy and paste what you have in eval.rules into rules.py (create a new text file and name it rules.py).
  - You are not allowed to hardcode a single IP (like 192.0.11.11 or the /32 mask) in **DdoS and Bruteforce** rules. You will receive **5 points deduction** for violation of this requirement for each rule. Instead, you should specify subnets and use the features of the attacks to capture them. When detecting distributed types of attacks, such as DDoS and brute force attacks, it is essential to use subnets in Snort rules rather than specific IP addresses. We do **not** have such a requirement for Botnet and Webattack.
  - **10 points will be deducted** if this file is missing. The minimum score for the project is 0. If you get a -10 as your final grade, the grade will be adjusted manually later.
- **connections.txt** - the result file generated by running `python3 cal_unique_connection_2022.py alert_json.txt`. **The connections.txt file is generated by the command. Do not change the connections.txt file**

Note:

- **Incorrect or missing files will lead to a 10 point grade deduction per file**
- **Using static IP addresses or /32 subnets will lead to a 5 point deduction**
- There is a limit on the number of submissions on gradescope. Please refer to "How to validate your answer" in the session below.
- Don't zip the file. Just upload them as separate files. Make sure the filename is correct.
- ***(Optional)*** If you wish to take your **Canvas** submission as your **final submission after you used all your chances on gradescope**, you will have to **let us know** through this form. We will compare your canvas submission with the gradescope submission and take the **highest** submission among all your submissions.
- All the **late submissions** will be on **Gradescope**. You do not need to make a post on Ed.

How to validate your answer:

1. We consider a connection to be "src_ip:src_port:dest_ip:dest_port". You can utilize the `cal_unique_connection.py` to check the unique connections of your alert_json.txt. You can compare the number of DDoS/BruteForce/WebAttack/Botnet you got with the statistics below. If the number is close, you are likely on the right track.
2. You can view the pcap file in Wireshark to confirm you are finding the right connections.
3. Last, you can verify your result by submitting your answer to the gradescope (See steps 4&5). As the number of trials is limited, perform step 2 first!
4. We have provided you a way to verify your results on the Gradescope. You need to upload your **connections.txt** and you will see your current score. And you can verify your results at **most 10 times!** Uploading more than **10 times** in gradescope will result in an error which notifies you that

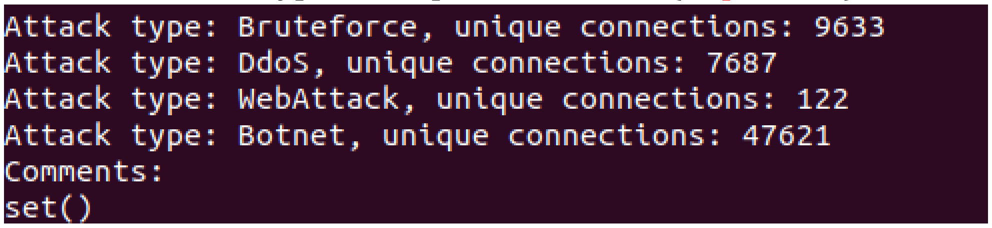
you cannot verify your result anymore. (Technically, you can still submit your work after the 10th submission. You can upload your final submission to canvas. However, you won't be able to validate your canvas submissions.)

5. Running a single snort rule against the evaluation.pcap can get a different result when you run it along with other snort rules. This is related to the limitation of Snort. **So please run all your rules together to get the result we want.**

Grading:

- The snort rules file you write would be run in the autograder.
- There are 4 attack categories as described above, each of which carries 25% grade weight.
- For each attack category, your grade = #correct alerts / (#real correct alerts + #incorrect alerts). Therefore, if you raise alerts for packets that are benign as one of the attack categories (false positives), you will lose points for that attack category. Also, if you miss raising an alert for an attack packet, you will lose points for that category.
- We will calculate your final grade for each category (which has 25 points as a full mark) by #final grade = min(25, #grade-for-one-category * 125%), which means if you catch **80%** of the attacks correctly, you can get full marks for the category.

Statistics for each type of unique connections (Important!):



```
Attack type: Bruteforce, unique connections: 9633
Attack type: DdoS, unique connections: 7687
Attack type: WebAttack, unique connections: 122
Attack type: Botnet, unique connections: 47621
Comments:
set()
```

(<u>The number might be a little different when you try to find it in Wireshark. Use the number that Snort gives you</u>)

We consider a connection to be "src_ip:src_port:dest_ip:dest_port". run "**python3 ~/Desktop/cal_unique_connection.py alert_json.txt**" to check the unique connections of your alert_json.txt and generate the results in `connections.txt`. If your alert JSON file is generated in the home directory, you might need to add sudo in front of your command.

FAQs

1. This all seems too hard, how do I start?
- Identify the attack from the list of possible attacks for each section. Once you have identified the attack, analyze the attack traffic packet headers (HTTP/TCP headers), and identify the common characteristics in the attack traffic. Use this information to craft your Snort rule.

2. Why did my rule change the number of packets captured for a particular attack when I added a rule for a different attack?
- If rules are too generic, they capture a broad range of packets leading to possible shadowing of other rules. If you do not see the output for a rule or you see that the count has changed, make sure you double check all your rules to ensure they are not too generic.

3. I am not seeing the exact count for the DdoS and Bruteforce rules as listed in the writeup.
- DdoS and Bruteforce are volumetric attacks. Thresholding can help tune out benign traffic. **<u>You are not expected to match the numbers in the writeup for these attacks.</u>** If you are within a ~10-15% margin of the number in the writeup(i.e. you have a rule that can successfully identify attack traffic from benign traffic), you will be awarded full points.

4. What are volumetric attacks and what do I do about it?

- DdoS and Bruteforce attacks rely on a large number of requests in their attacks. These attacks generate a high volume of attack traffic. Content filtering might not be the best course of action. To identify such attacks, use the I/O graphs and threshold!

5. Everytime I run my rules, the number of connections identified for the Bruteforce traffic changes. Why is this?
- This is related to how Snort parses packets and is expected behavior. Therefore, we provide a margin of ~10-15%

6. Do I get partial credits if the snort rule is not 100% correct?
- Partial credit is awarded according to how many false positives are detected. Please refer to the grading algorithm above.

7. Do you provide sample pcaps for Ddos?
- No, we don't provide sample pcaps for dos because you can observe dos from the evaluation pcap. A good start is reading some existing snort rules for detecting dos attacks to get a sense of it.

8. After grading this assignment, will you release the correct answers?
- No, we don't, since similar projects run across semesters.

9. Should we ignore instances of ICMP and IP?
- You can ignore ICMP & IP

10. Is Tshark/Wireshark installed on the virtual machine?
- Yes. If the VM is too slow for you, I would recommend you install pcap reading tools on your local machine and use the VM only for Snort. 'evaluation.pcap' is a very large file so it can take some time to load. You may need to increase the amount of RAM available to the system to get it to display properly.

11. Looks like I can complete the assignment without using VM?
- Correct, you can complete without the VM but make sure your snort file works with the version installed in the given VM.

12. What constitutes a connection?
- A "Connection" is identified by its Source Address, Destination Address, Source Port, and Destination Port.

13. Any hints on what we should look for when trying to identify SPAM-ing? SMTP connections?
- You may want to look at packets within a certain time frame.

14. Can attacks be only from one or two machines or it should be from massive bot machines?
- It can be from any number of machines. The exact number is not relevant.

15. Should we include an IPV6 connection? or only IPV4?
- Only IPV4

16. We don't need to zip the file when we submit, right?
- No, you just need to submit the two files separately: rules.py and connections.txt.

17. Is it possible to get 100% just using rules derived from samples?
- We can't confirm or deny. :-)
  The sample pcaps are there to give you a good idea, but we don't claim they're representative. You should learn the pattern in the sample pcap and try to find related or similar patterns in the evaluation pcap. Always manually verify that what you find is correct.

18. I am getting some json error/json decoder error

- Yes that means your json file is too big and maybe you need to put more constraints on your snort rule to generate a smaller json file.

19. I am seeing empty alert.json file
    - delete the previous alert_json.txt, if any
    - delete the previous connections.txt, if any
    - ensure each command in your file has a unique SID.

20. Why am I getting many more connections than necessary in DdoS and how should I lower it?
- Try using the frequency as a filter type. Remember that the direction of tracking matters (by src VS. by dst)

Resources:

Readings on botnets behavior: Please read through the following papers, to get an understanding of what a bot is, and how botnets behave. Please note that we are not asking you to implement the proposed methodologies, *e.g.* a machine learning method to detect bots.

- *"BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation"*, Gu et. al. http://faculty.cs.tamu.edu/guofei/paper/Gu_Security07_botHunter.pdf
- *"BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic"*, G. Gu, J. Zhang, W. Lee, http://faculty.cs.tamu.edu/guofei/paper/Gu_NDSS08_botSniffer.pdf
- *"BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection"*, G. Gu, R. Perdisci, J. Zhang, W. Lee, https://www.usenix.org/legacy/event/sec08/tech/full_papers/gu/gu.pdf

Snort resources: Here you can find some examples of Snort rules, and some resources so that you get familiar with Snort rules. The purpose of these resources is only to get you familiar with how Snort rules look like. You are expected to write your own Snort rules.

- https://usermanual.wiki/Document/snortmanual.760997111/view
- https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/000/596/original/Rules_Writers_Guide_to_Snort_3_Rules.pdf

When you google snort syntax, you should be aware that we are using snort 3 not snort 2. There are some subtle differences between Snort 2 and Snort 3 syntax. Snort 2 syntax might result in errors when you run your rules locally.

Example: Writing Snort rules to detect DDoS traffic: This is an example to give you an idea about how we can use our understanding of an attack, and write Snort rules with potentially long shelf life, to detect this attack. Intro reading for dos: https://en.wikipedia.org/wiki/Denial-of-service_attack. Snort for DDoS: Please read this to get a general idea about how Snort can be used for this purpose. Please focus on sections 3 and 4. After reading the above, one way to detect dos traffic is to monitor the rate of incoming traffic. Here is an example Snort rule based on traffic rate: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node35.html
(Notice that this link is actually the snort 2 manual. However, the syntax for detection_filter is the same in snort 2 and 3. The Snort 2 manual in this case provides a better example to illustrate this filter. For other filters, especially the URL content filters, you should double check the snort 3 manual and other resources online)

Useful tools/commands:

- You can SCP the files from the VM to your local machine and view them using Wireshark.
- SCP: http://www.hypexr.org/linux_scp_help.php
- Redirecting a program's output: http://linuxcommand.org/lc3_lts0070.php
- You can install Wireshark from here: https://www.wireshark.org/
  We do recommend you to install Wireshark on your own machine for evaluation pcap analysis

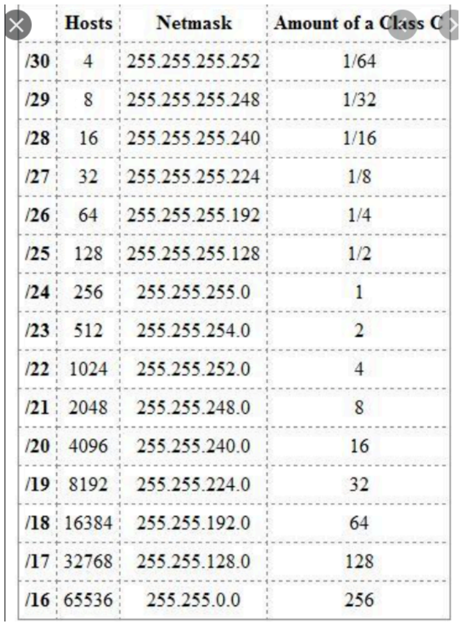
because the pcap file is big and your VM might be significantly slower than your laptop.

- Wireshark display filters to view part of the traffic: https://wiki.wireshark.org/DisplayFilters
- How to scp a file named file to the VM: `scp file student@<VM’s ip>:/home/student`. If your VM has a different IP address than the above then you can find it by starting the VM, then log-in, and then do: `ip a`.
- The above scp command is just an example. Modify it accordingly. Resource for scp syntax: http://www.hypexr.org/linux_scp_help.php

(optional reading) Subnet:

- **Why is 172.31.0.0/16 a subnet?**
  Because it uses CIDR notation. CIDR and subnetting are virtually the same thing.
- **What's CIDR?**
  CIDR is Classless inter-domain routing. It is the /number representation. In this case, we have /16
- **What does /16 mean again?**
  /16 represents the **subnet mask** of 255.255.0.0

  If you convert 255.255.0.0 into binary, you will see 16 1's and that's where the number 16 comes from.
  Of course, I can't remember all those conversions for all netmask. There is a cheat sheet:



	Hosts	Netmask	Amount of a Class C
/30	4	255.255.255.252	1/64
/29	8	255.255.255.248	1/32
/28	16	255.255.255.240	1/16
/27	32	255.255.255.224	1/8
/26	64	255.255.255.192	1/4
/25	128	255.255.255.128	1/2
/24	256	255.255.255.0	1
/23	512	255.255.254.0	2
/22	1024	255.255.252.0	4
/21	2048	255.255.248.0	8
/20	4096	255.255.240.0	16
/19	8192	255.255.224.0	32
/18	16384	255.255.192.0	64
/17	32768	255.255.128.0	128
/16	65536	255.255.0.0	256

Wait, what's a subnet mask?
Feel free to read this link if you want to know more:
https://avinetworks.com/glossary/subnet-mask/

Important Notes

Disclaimer for background traffic. Please note that the traffic that is found in the evaluation pcap, and/or at the Sample pcaps is not generated by us. The dataset closely resembles realist traffic. Part of this traffic might contain inappropriate content or language. We have taken extra measures and we have performed considerable effort to filter all traffic, based on commonly used inappropriate words. We have filtered the http payload and URIs. Nevertheless, it might still be possible that some inappropriate content or words might have not been filtered entirely. In case you locate such content, we are letting you know that it is not intentional, and we are not responsible for it. Also, to complete this assignment, you do not need (nor do we ask you) to click on URLs found inside http payloads.

Additional tools are not allowed. For the assignment, you are not allowed to use any available tools, related to Snort or others. For example, you are not allowed to use Snort preprocessors that may be publicly available, pre-compiled Snort rules, detection tools. etc. **You are expected to write your own Snort rules.**

Limited support for running snort locally. We have the env set up for you inside the VM. Setting up the snort 3 env on your own could be troublesome. We strongly recommend you to run snort rules inside the VM we provide (you can analyze(use wireshark) the pcap outside the VM but run the rules inside the VM).