Total points: 80

Assignment 3

Due: 11:59 PM, 2nd Aug (Fri)

Written assignment

1. [14 points] Using a Tor controller like Stem, build a circuit through Tor and load a web page (any web page) as quickly as possible. A valid circuit must have three nodes, and each of them must be from a different /16 space. Time your page load and write down the fastest page load you achieved without caching, and answer the following questions.
   (a) [9 points] What methods did you use to achieve a fast load time?
   (b) [5 points] Are there any privacy implications of using your methods to achieve a fast load time? Analyze each method that you used.

   Please be a good Tor citizen: **do not load more than one web page every 30 seconds**.

2. We have four data privacy techniques:

   1. k-anonymity
   2. Differential privacy
   3. Secure multiparty computation (SMPC)
   4. Private information retrieval (PIR)

   For each scenario below, two of the four data privacy techniques will be proposed to resolve the challenge. Choose the correct one, explain why it is suitable, and why the other choice is not suitable.

   (a) [4 points] You want to buy a new smart device that encourages a healthy lifestyle by monitoring your daily exercise. The device needs to track your movement on a map to know how much calories you are actually expending (e.g. hiking and swimming is different from walking). However, you consider this to be a privacy risk: you do not want the app to know where you are at all times. A new company making these smart devices is willing to use a data privacy technique to protect your privacy. Proposals: differential privacy, PIR.

   (b) [4 points] You would like to purchase a new web domain. However, you are aware of the practice of cybersquatting; people may purchase the domain first if they know it is in demand, and sell it to you at an elevated price. You want to know if the domain is still available, but you are worried that attempting a DNS query for the domain will lead to some DNS servers purchasing it immediately for cybersquatting. To assure potential customers that it is not malicious, a DNS server is willing to cooperate with you and implement a privacy-preserving algorithm. Proposals: k-anonymity, PIR.

   (c) [4 points] A hospital has information about millions of carcinogenic and non-carcinogenic patients in various locations. The exact location, if revealed, is a privacy breach. Researchers want to determine if there is a correlation between cancer and building materials; they already have a large database of building materials used in different areas. The correlation is potentially complicated, though constructions in the same area almost always use the same building materials. Proposals: k-anonymity, SMPC.

   (d) [4 points] A new service has been developed to allow users to compare salaries and potentially identify unfairness or discrimination in pay. After passing a local verification process, users can submit their salaries to the service, and the service should be able to inform them if they are being underpaid or not by comparing with other users of the same profession/experience (once there is enough data). The actual salary is considered sensitive information and should not be exposed to the service. Proposals: differential privacy, SMPC.

Programming assignment

Firewall [50 points]

In this assignment, you are asked to write an IPv4 packet-filtering firewall for the subnet 142.58.23.0/24. It examines each packet to decide whether or not to filter it out (drop it).

The input to your program is a tcpdump-like file containing IP packet data dumps. An example, packets.txt, has been provided. It is simplified to remove packet header information, which is contained in the packet data. Each packet starts with a packet number on its own line, then the packet data in the following tabbed lines. A packet data line has up to 16 bytes of data. It starts with a hexadecimal byte count, then the actual data, also written in hexadecimals.

For example, a typical packet in the assignment may start with:

```
4500 00a3
```

This is four bytes:

- The first hexadecimal (first half-byte) is 4, indicating IPv4.
- The second hexadecimal is the header length, which is a minimum of 5 and always 5 in this assignment, as optional headers are not used.
- The third and fourth hexadecimals (second byte) are services that are not used in this assignment, thus set to 0.
- The fifth to eight hexadecimals are the packet length in bytes, including the header. 00a3 is equal to 163, so this packet has 163 bytes.

To do this assignment, you will need to understand how to read IP, ICMP and TCP headers. These headers will be referenced in the assignment, and you can read more about their positions and functionality in a variety of sources including Wikipedia and RFCs.

Your program should be called filter and will be called with (python3 example):

```
python3 filter.py <option> <filename>
```

where:

- option is either -i, -j, -k, corresponding to three subparts of the assignment
- filename is the packet dump file in the same format as packets.txt.

In standard output, write the result of whether or not each packet is dropped. Each line should correspond to one packet, and should start with the number, followed by a space, followed by either "yes" (it is a malicious packet and should be dropped) or "no" (it is not a malicious packet and should be allowed to pass). For example, `python3 filter.py -i packets.txt` should output:

```
1 yes
2 no
3 yes
4 no
5 yes
6 no
7 no
```

Do not write anything else to standard output.

Your packet filter should not be overzealous. That is, if the question does not ask you to filter out a certain packet, then you should not do so (answer "no"). To simplify the assignment, you do not need to deal with re-assembly or re-ordering of packets within this assignment. The header checksums should also be ignored. The problems are not cumulative: for examples, the answer to (b) should not filter out violating packets of (a).

(a) [10 points] Ingress filtering (-i)

Write an ingress packet filter that filters out all packets with IPs that do not make sense based on your firewall's location. For this question, your firewall expects to only see **incoming packets** from outside the subnet to your subnet. Other packets are attack packets. For other questions, your firewall should expect to see both incoming and outgoing packets.

(b) [20 points] Two ping-based attacks (-j)

Write a packet filter that filters out both pings of death and smurf attacks targeting hosts in your subnet. Both of these are pings, more formally known as ICMP echo packets. To prevent pings of death, study the slides and implement a filter that would cause a buffer overflow described in the slides. To prevent smurf attacks, drop the ping packets sent to the network broadcast address that would cause a host in your subnet to be flooded.

Notes:

- An ICMP echo is defined on RFC 792 page 13. It is **not** the same as an echo reply.
- There is a protocol field in the IP header that indicates whether or not the packet is an ICMP message.
- You may assume that the IP length field is always correct.
- Only pings should be filtered in this question. Other packets should pass.
- Only attacks targeting hosts in your subnet should be dropped. Do not filter other packets.

(c) [20 points] Connection tracking (-k)

Write a packet filter that filters out all TCP packets that do not belong to a proper connection. A proper connection:

1. Starts with a valid 3-way handshake.
   - First packet is a SYN with sequence number A.
   - Second packet is a replying SYN-ACK with acknowledgment number A+1 and sequence number B.
   - Third packet is a ACK with acknowledgment number B+1.
   - If any packets during this process are invalid, simply drop them; do not restart the handshake. For example, if the third packet is a SYN-ACK, and the fourth packet is a proper ACK, the handshake has still been completed, and further packets should be accepted.
2. Has not been terminated by a FIN or RST packet.
   - Note that a FIN or RST packet **can** kill a handshake, and the handshake will have to restart from the beginning afterwards.

All packets in a proper connection should be accepted. Note that connections can be made in both directions.

Packet examples

packets.txt contains examples of the following packets.

- Packet 1 is a ping from 142.58.23.107 to 1.2.3.4.
- Packet 2 is a ping of death from 1.2.3.4 to 142.58.23.107. Note the large offset.
- Packets 3 to 5 are a regular TCP handshake from 142.58.23.107 to the HTTPS port of 1.2.3.4: SYN, SYN-ACK, and ACK.
- Packet 6 is a TCP FIN from 1.2.3.4 to 142.58.23.107. This would kill the connection, whether it is fully open or half open.
- Packet 7 is a TCP RST from 1.2.3.4 to 142.58.23.107. This would also kill the connection.

Submission instructions

All submissions should be done through CourSys. Submit the following programs:

- `a3.pdf`, containing all your written answers.
- Code for the programming assignment, detailed below:

For the programming assignment, submit your code; do not submit any compiled files.

C++: Submit filter.cpp. I will compile them and call `./filter <option> <testfile>`.

Python: Submit filter.py. I will call `python3 filter.py <option> <testfile>`.

Java: Submit filter.java. I will compile with `javac filter.java` and then call `java filter <option> <testfile>`.

If there is a Makefile in your folder, the Makefile will override all of the above. This implies if you are not writing in C++, Python, or Java, you must include a Makefile.

Keep in mind that plagiarism is a serious academic offense; you may discuss the assignment, but write your assignment alone and do not show or send anyone your answers and code.