Homework 4: Tor

COSC435 - Homework 4

This homework is worth 100 points. Homework solutions should be submitted to **Autolab (https://autolab.georgetown.edu)** .

Assigned October 31st, 2019; **due November 21st at 11:59pm**.

Traffic Fingerprinter [100 points]

For this assignment, you will build a website fingerprinter. Tor provides anonymous communication, but is not perfect. A known attack against Tor is traffic/website fingerprinting. The overall concept is that an eavesdropper who can intercept a Tor user's traffic learns something about that traffic even though it is encrypted. Specifically, the eavesdropper learns the number of packets sent and received, their spacing, and their timing.

It turns out that this information is sufficient to determine with fairly high accuracy which website the Tor user is visiting, especially in the *closed world* setting in which there is only a finite number of websites.

Consider two sites: google.com, which has a fairly small sized webpage, and nytimes.com, which is substantially larger and contains much more content. Visiting the two sites will yield two very different traffic patterns. Clearly, nytimes.com will require the transmission of more encrypted packets; additionally, each web object (e.g., graphic) on the nytimes.com website will result in some burstiness in terms of encrypted Tor packets.

Your job is to create a program that takes in a packet capture from Tor *that contains a single page load* and outputs the identity of the website that was visited. Importantly, your code will not perform the actual packet interception. You'll use tcpdump or **Wireshark (https://www.wireshark.org/)** or some other equivalent tool that can produce a **pcap (https://en.wikipedia.org/wiki/Pcap)** file, and your program will take that as input.

To make this tractable in the time allotted, your program should consider the world that exists only of the following seven websites:

Website name	URL	Sample PCAP
canvas	http://canvas.georgetown.edu/	click me (https://georgetown.instructure.com/courses/82306/files/36769 wrap=1)
bing	http://bing.com	click me

		(https://georgetown.instructure.com/courses/82306/files/36769 wrap=1)
tor	https://www.torproject.org/	click me (https://georgetown.instructure.com/courses/82306/files/36769 wrap=1)
wikipedia	https://wikipedia.org/	click me (https://georgetown.instructure.com/courses/82306/files/36769 wrap=1)
neverssl	http://neverssl.com/	click me (https://georgetown.instructure.com/courses/82306/files/36769 wrap=1)
craigslist	https://washingtondc.craigslist.org/	click me (https://georgetown.instructure.com/courses/82306/files/36769 wrap=1)
autolab	https://autolab.georgetown.edu/	click me (https://georgetown.instructure.com/courses/82306/files/36769 wrap=1)

In other words, your program should assume that whatever traffic capture file is provided as input, it corresponds to one of the above sites. I've provided sample PCAPs of a Tor Browser visiting the seven sites listed above.

Creating the Captures

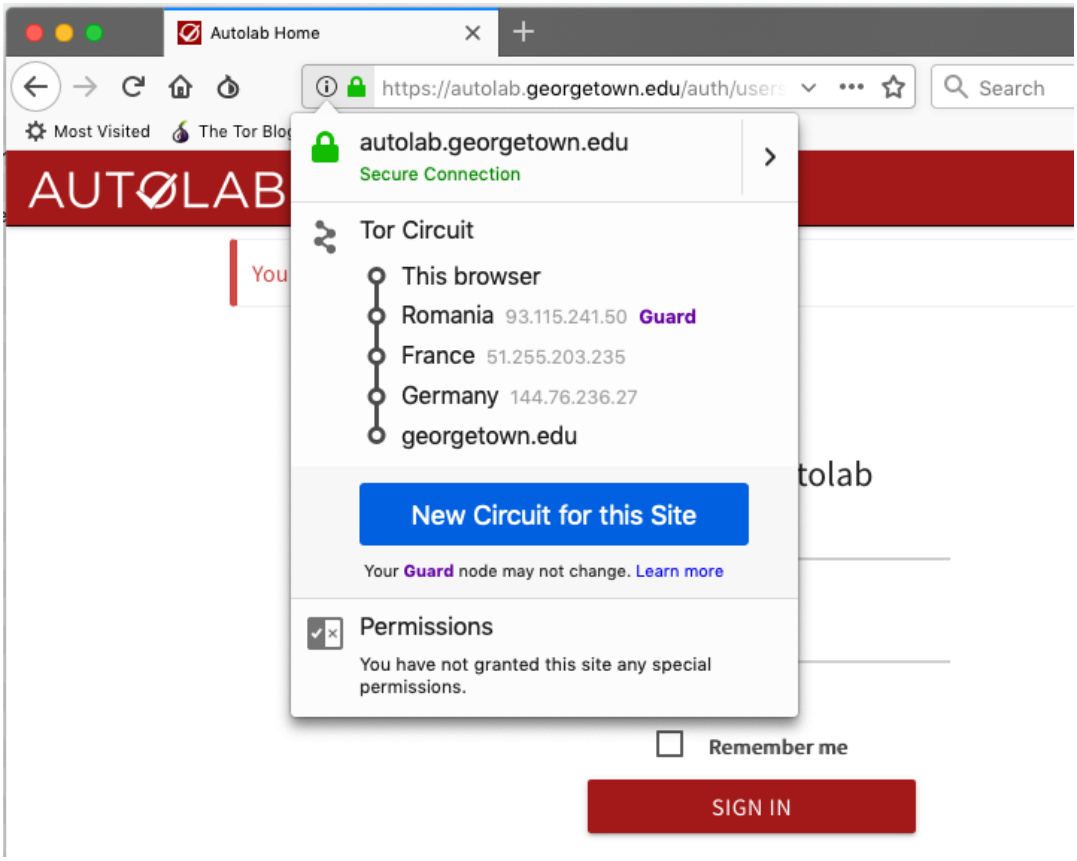
As a first step, you'll want to **install the Tor Browser on your local machine** (https://www.torproject.org/projects/torbrowser.html.en). You should install the latest version (version 9.0), so if you already have an older version of the Tor Browser, you should upgrade. You should also use the default options in Tor. This includes not using Tor bridges and not changing from the default security mode.

Next, you'll need to install a tool for capturing packets. If you are on MacOSX or Linux, you can use the **tcpdump** (http://www.tcpdump.org/) command-line tool. There's lots of good documentation on tcpdump online -- just search for it. If you are Windows (or Mac), you can use **Wireshark** (https://www.wireshark.org/#download).

Before we continue, make sure that you ONLY capture your own packets. Otherwise, you're violating Georgetown's **Computer Systems Acceptable Use Policy** (https://security.georgetown.edu/it-policies-procedures/computer-systems-aup), which could subject you to serious penalties.

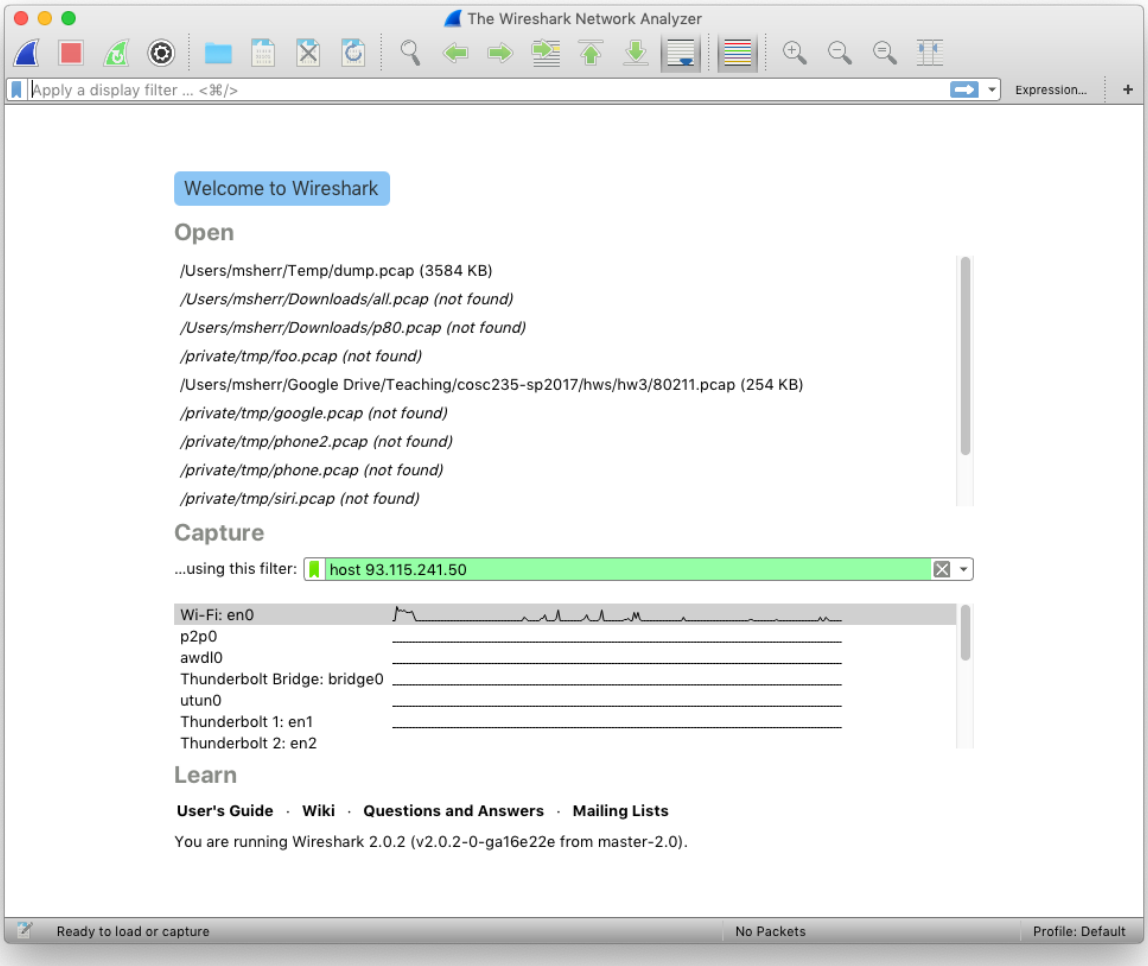
So how do we make sure that you only capture your own packets? You'll need to learn what your guard relay is. A guard relay is a relay that you'll always choose as your first "hop" in an anonymous path. It's

important that this relay not change for reasons that go beyond the scope of this homework (if you are interested, **read this paper** (https://www-users.cs.umn.edu/~hoppernj/single_guard.pdf)). To figure out who your guard is, start up the Tor Browser and click on the URL bar. You should see something like this:



In this example, the guard is 93.115.241.50. Figure out your guard's IP address. Chances are, it'll be different from mine. You'll notice that if you exit the Tor Browser bundle and restart it and then visit another webpage, you'll use this same guard relay; the second and third (exit) guards will very likely differ. If it helps, the IP address of the guard relay for the sample PCAPs linked above is 217.160.40.194.

Basically, what you'll want to do is to capture packets only to/from the guard relay. To do this with Wireshark, when you start Wireshark, it'll ask you for a "capture filter". Enter *host 1.2.3.4*, replacing 1.2.3.4 with your guard's IP address; for example:



When you start the capture, you'll see a bunch of packets being captured. They'll appear to Wireshark as TLS traffic since Tor uses TLS between hops.

Next step: Stop your capture and get rid of whatever you captured before. Then, in turn, visit each of the above websites (one at a time and create a separate capture for each. Make sure you wait for the page to entirely load. When you save the file, make sure you save it as an uncompressed pcap file (this isn't the default option in my version of Wireshark).

At this point, you should have a pcap file for each of the above 7 sites. You may, in fact, want to repeat this and create a few copies of each.

Building a Website Fingerprinter

The next step involves some creativity. You will now design and implement a program that takes in a pcap file and determines to which of the 7 sites it corresponds. Note that it will **not** be an exact copy of one of your earlier pcap files since the timing will be slightly different and some of the content might have slightly changed. The main task of this assignment is to figure out which is the best fit.

How you do this is up to you. You will need to think creatively. Keep in mind what you have at your disposal: the timing and quantity of packets, and their directions. As a hint, you should probably only consider the packets that convey data -- that is, the packets with a payload length of at least 1 byte.

Other Super Important Requirements (read these carefully!)

Your code will need to read pcap files. Use a library for this. I strongly suggest you use **scapy (https://scapy.readthedocs.io/en/latest/)** to read the pcap files. There is a lot of documentation online for scapy. You can use any documentation or examples you want for reading in the pcap files, but if you are using someone else's code (or a derivation of it), **you must cite it in your program**.

Also, your code will very likely need to read the pcap files that you've collected; it needs to know to what to make comparisons. Just include those with your submitted .zip file. Don't forget ... if we can't run your code, then we can't grade it.

Your code should run on your Google Cloud VM.

Your program should be called *fingerprinter.py*, and it should take as its sole argument the name of a pcap file. Your program will attempt to figure out which of the 7 sites listed below was visited via Tor, based on the provided pcap packet capture. In short, your program should be invoked as:

```
python3 fingerprinter.py input_pcap.pcap
```

For the pcap that is provided to your program, don't assume that it will use the same guard or that it was captured on your machine. Your code will have to cope with the fact that there will be two unknown IP addresses in the capture: that of the client and that of the client's guard. You'll have to figure out which is which in some automated fashion.

Your program should clearly identify which website is the best match for the supplied pcap file. Please use the values in the "Website name" column above. **The only output of your program should be an ordered list of the seven sites, from that deemed most likely to least likely to correspond to the input pcap file.** Each entry in the list should be separated by a newline ('\n'). For example, a reasonable output of your program might be:

```
canvas
neverssl
bing
wikipedia
tor
autolab
```

craigslist

The above example indicates that your program thinks that the input pcap file corresponds to the capture of someone visiting canvas via Tor, with neverssl as its second guess, bing as its third, etc.

To test your code, create a separate "test" set of pcaps that is different from the "training" set that your program will use. Test your program against each of these test pcaps and verify that it identified the correct site. If it didn't, make your program better. :)

Since I'm giving you a bunch of sample pcap files, you should test against those too! Your program should be able to correctly identify each.

You can safely assume that we'll be supplying our own test set of pcaps (and not the ones provided above -- although each one we use to test your code will correspond to one of the seven sites listed above.

Submission Instructions

You will use the **Autolab** **(https://autolab.georgetown.edu)** system for submission. That said, your assignments will be manually graded. There is no autograding for this assignment.

Please submit your solution as a single .zip file to the "hw4" assignment on **Autolab.** **(https://autolab.georgetown.edu)**

Please be sure to include ALL files required to run your program. This likely includes a bunch of pcap files.

Put your fingerprinter.py file in the main/top directory of the .zip file. We may attempt to use the Autograder.

Late assignments will be penalized according to the **Grading Policy** **(https://georgetown.instructure.com/courses/82306/pages/policy-grading)** .

Please post questions (especially requests for clarification) about this homework to **Piazza** **(https://piazza.com/georgetown/fall2018/cosc435/home)** .