Design:

* “Basic” / Proof of concept fingerprinting on HTTPS, no Tor or any extra involved
* We serve different models at different paths on the same server, ie.:
  + [https://lab0.csulab.ke8kxk.com/model\_0](https://lab0.csulab.ke8kxk.com/model_00)
  + [https://lab0.csulab.ke8kxk.com/model\_1](https://lab0.csulab.ke8kxk.com/model_01)
  + …
  + [https://lab0.csulab.ke8kxk.com/model\_9](https://lab0.csulab.ke8kxk.com/model_99)
* We only use 10 models to make things easier. We can even hand-pick models that produce better results
* The goal of attacker is to identify the model (path) being accessed
* The considerations for using models include:
  + Models can be served locally and the accesses can be super fast, so we can complete trace collection in maybe one class
    - 1 model can be loaded within 1 second.
    - Considering setups/teardowns, think about 5 seconds
    - 10 models \* 50 traces per model \* 5 seconds = 2500 seconds = 42mins
  + Reduce the uncertainty from outside websites
* We stick to Chrome here, not Firefox, just because of Chrome’s market share. We were using Firefox in WT/TT because the specific extension API is not available in Chrome
* Intuition is given priority over scientific proofs, eg. we do not calculate the accuracy value

Preparation:

* Computers for student use, with the following pre-installed:
  + Linux (Ubuntu / Debian)
  + Chrome
  + Selenium / WebDriver / tcpdump / Wireshark etc.
  + Home-made scripts to do and automate trace collection
  + Home-made scripts to analyze traces
* Servers for serving the models. Use multiple servers if the number of students is large to prevent overloading the server. Otherwise just one for a couple groups should be okay.

Common steps:

1. **(Waterfall)** Background: know that loading a web page involves multiple resources
2. Open a Chrome incognito window with Developer tools open, showing the Network tab
3. Navigate to <https://www.google.com> . Soon after the page is fully loaded, look at the Waterfall column in the Network tab, and remember (or take note of) the shape
4. Repeat 2 and 3, note that the shapes remain similar
5. Open a Chrome incognito window with Developer tools open, showing the Network tab
6. Navigate to <https://www.baidu.com> . Soon after the page is fully loaded, look at the Waterfall column in the Network tab, and remember (or take note of) the shape
7. Repeat 5 and 6, note that the shapes remain similar
8. Compare the shape between 3 and 6; they should show more difference

CS student further steps:

1. **(From websites to models)** Background: know that the resources depend on others, forming a dependency tree
2. Similar to Common steps above, but use <https://lab0.csulab.ke8kxk.com/model_0> and <https://lab0.csulab.ke8kxk.com/model_1> ; expect to see similar results as above
3. Optionally, you can inspect the source code of the web pages (models)
4. **(From resources/requests/waterfall to packets)** Background: know that requests are transmitted in TCP packets at low level, and since the traffic is in HTTPS, you can’t directly see the content of communication in the packets
5. Try to manually capture the TCP packets using tcpdump in a browsing session between your computer and lab0.csulab.ke8kxk.com. Save to a pcap file.
6. Optionally you can inspect it with Wireshark (better GUI), although it would be difficult to compare multiple pcap files for similarity manually due to the amount of data.
7. Use the provided scripts to collect the dataset of pcap files. You can inspect the script to see what it does.
8. **(From packets to features)** Background: know that for several types of attacks, features are extracted from packet sequences, then comparisons are made on features
9. Use the provided scripts to generate features from the collected and labeled (with the model ID) pcap files
10. Use the provided scripts to generate training data from the feature dataset
11. **Now it’s time to see the attack in action.** Pick one model at your choice, then visit it and do step 5 or 7 on it to generate a pcap file, do step 9 to generate features, then use the provided attacking script to match it against the feature dataset. It should tell you the model you’ve chosen.