1-2

Use the values printed on the webpage to find the median access time and report your results as follows.

Number of Cache Lines	Median Access Latency (ms)
1	0
10	0
100	0
1,000	0
10,000	0.1
100,000	0.6
1,000,000	8.5
10,000,000	109.90

1-3

According to your measurement results, what is the resolution of your `performance.now()`? In order to measure differences in time with `performance.now()``, approximately how many cache accesses need to be performed?

The resolution of performance.now() on this system is 0.1 milliseconds.
We needed around 10,000 cache line accesses to see differences in time as average times rose above 0.

2-2

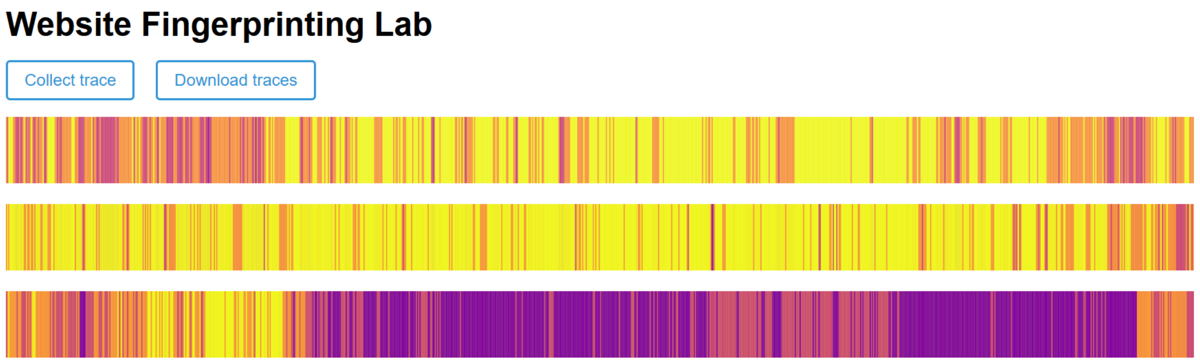
**Report important parameters used in your attack. For each sweep operation, you access N addresses, and you count the number of sweep operations within a time interval P ms. What

values of N and P do you use? How do you choose N? Why do not you choose P to be larger or smaller?**
We use an N of 524, 288 (32 * 1024 * 1024)/(64).
32 * 1024 * 1024 is the size of the LLC on most modern CPUs (32 MB converted into bytes).
We divide by 64 line bytes, so each sweep accesses the 524,288 cache lines at once.

On our computers, an interval of P = 5 turned out to be the best, with shorter P intervals returning overly messy results, while longer P intervals made resolution too low to distinguish differing behavior.

2-3

Take screenshots of the three traces generated by your attack code and include them in the lab report.



2-4

Use the Python code we provided in Part 2.1 to analyze simple statistics (mean, median, etc.) on the traces from google.com and nytimes.com. Report the statistic numbers.

google.com
Mean = 2.221
Median = 2.0
Stdev = 0.506
Range = 3 (min=1, max=4)

newyorktimes.com
Mean = 1.768
Median = 2.0
Stdev = 0.423
Range = 2 (min=1, max=3)

2-6

Include your classification results in your report.

```
precision    recall  f1-score   support

  https://www.baidu.com       0.74      0.78      0.76        40
   https://www.ebay.com       0.42      0.45      0.43        40
 https://www.google.com       0.80      0.88      0.83        40
https://www.youtube.com       0.61      0.47      0.54        40

               accuracy                           0.64       160
              macro avg       0.64      0.64      0.64       160
           weighted avg       0.64      0.64      0.64       160
```

3-2

Include your new accuracy results for the modified attack code in your report.

```
=== Classification Report ===
                  precision    recall  f1-score   support

  https://www.baidu.com       0.80      0.80      0.80        40
   https://www.ebay.com       0.78      0.62      0.69        40
 https://www.google.com       0.84      0.80      0.82        40
https://www.youtube.com       0.80      1.00      0.89        40

               accuracy                           0.81       160
              macro avg       0.81      0.81      0.80       160
           weighted avg       0.81      0.81      0.80       160
```

3-3

Compare your accuracy numbers between Part 2 and 3. Does the accuracy decrease in Part 3? Do you think that our "cache-occupancy" attack actually exploits a cache side channel? If not, take a guess as to possible root causes of the modified attack.

No, given that our accuracy actually increased. I think the attack has to do with CPU timing and operations, given our new code tracks add operations per time window, which records CPU operations. In class, we also discussed this attack, and system interrupts were the explanation, which is also a CPU operation. Directly considering CPU operations is more accurate than cache due to cache pollution by other background processes.