Part 1: Warm Up

Group 12: Andre Rastrelly, Nhu Le, Raymond Zou

1-2 Discussion Question:

Number of Cache Lines	Median Access Latency (ms)
1	0
10	0
100	0
1,000	0
10,000	0.10000000149011612
100,000	0.20000000298023224
1,000,000	N/A
10,000,000	N/A

1-3 Discussion Question:
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?

According to our measurement results, the resolution of performance.now() is 0.1ms. While running the script and analyzing the median values, numbers that began with 0.10000 would always be the number 0.10000000149011612, showing that the resolution is 0.1ms. Approximately 10,000 cache accesses need to be performed in order to start measuring differences in time with performance.now().

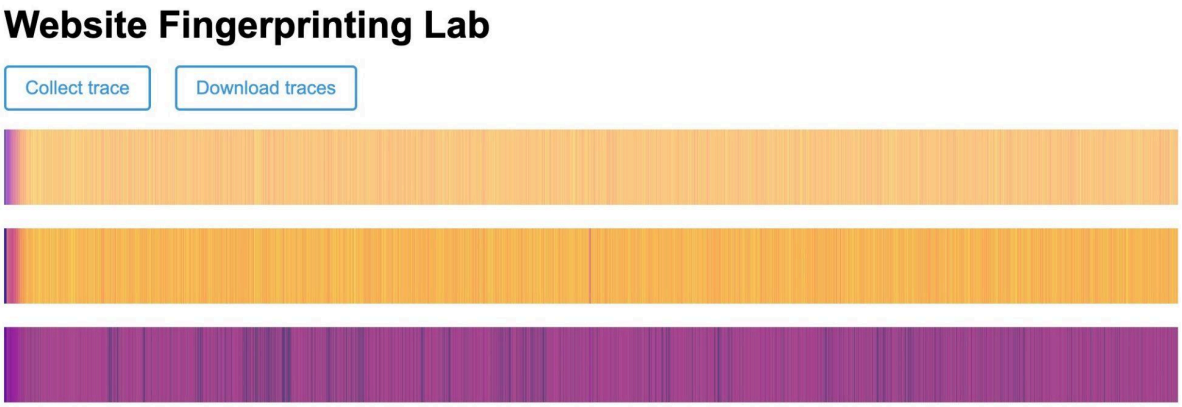
Part 2: Side Channel Attacks with JavaScript

2-2 Discussion Question:
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 didn't you choose P to be larger or smaller?

In our sweep operation, we defined a LINE_SIZE variable to be 16 (operation 128/sizeof(double)) to ensure our buffer would take up the necessary space in cache. We also

experimented with two parameter settings. We chose P = 0.6 ms and N = 10000. We also chose P = 10 ms and N = 20000. We increased N until the traces for different activities became distinguishable. Increasing N to 20000 increased the cache pressure and more buffers were evicted from the cache. By doing this, it made the traces easier to distinguish. We chose P = 0.6 and P = 10 instead of a larger number because it gives us a more stable sweep count and a clearer difference between the websites.

2-3 Discussion Question:
Take Screenshots of the three traces generated by your attack code and include them in the lab report



2-4 Discussion Question:
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.

```
Labels found: {np.str_('https://www.google.com'), np.str_('https://www.nytimes.com')}

=== GOOGLE ===
Mean: 219.4645
Median: 223.0
Std: 12.928485593835035
Min: 36
Max: 234

=== NYTIMES ===
Mean: 194.731
Median: 207.0
Std: 33.123430362811156
Min: -1
Max: 230
```

2-6 Discussion Question:
Include your classification results in your report.

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

 https://www.amazon.com       1.00      0.50      0.67         4
https://www.nytimes.com       1.00      1.00      1.00         4
 https://www.roblox.com       1.00      1.00      1.00         4
https://www.youtube.com       0.67      1.00      0.80         4

              accuracy                           0.88        16
             macro avg       0.92      0.88      0.87        16
          weighted avg       0.92      0.88      0.87        16
```

Part 3: Root Cause Analysis

3-2 Discussion Question:
Include your new accuracy results for the modified attack code in your report.

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

 https://www.amazon.com       1.00      0.75      0.86         4
https://www.nytimes.com       1.00      1.00      1.00         4
 https://www.roblox.com       1.00      1.00      1.00         4
https://www.youtube.com       0.80      1.00      0.89         4

              accuracy                           0.94        16
             macro avg       0.95      0.94      0.94        16
          weighted avg       0.95      0.94      0.94        16
```

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.

The accuracy increases in part 3 (with P = 1000) compared to part 2. Since part 3 doesn’t perform any memory accesses, we do not believe that our “cache-occupancy” attack actually exploits a cache side channel; we think that the attacks take advantage of system interrupts. Since the code in both parts 2 and 3 run for a specific amount of time/intervals, the presence of system interrupts could affect how many counts our script is capable of counting.