CSE406 Computer Security
Assignment 2: Side-Channel Attack
Website Fingerprinting Report

Student ID: 2005009

June 20, 2025

Contents

1 Introduction

This report presents the implementation and results of a comprehensive side-channel attack system for website fingerprinting. The project demonstrates how timing-based side-channel attacks can be used to identify websites visited by users without accessing their network traffic or screen content, utilizing shared CPU cache resources between browser tabs.

The implementation consists of four main tasks: timing precision measurement, sweep counting attack implementation, automated data collection, and machine learning classification, along with bonus tasks for advanced techniques and collaborative dataset collection.

2 Task 1: Timing Precision Measurement (10%)

2.1 Objective

The first task focused on understanding the precision limitations of JavaScript's `performance.now()` function and measuring cache access latencies to establish the foundation for subsequent side-channel attacks.

2.2 Implementation

The `readNlines(n)` function was implemented in `static/warmup.js` to:

- Allocate buffers of size $n \times$ LINESIZE where LINESIZE = 64 bytes
- Access different cache lines at LINESIZE intervals
- Perform 10 complete buffer reads
- Measure timing using `performance.now()`
- Return median access time

2.3 Results

Figure 1 shows the latency measurements for different numbers of cache line accesses on a Windows laptop.

Latency Results

N	Median Access Latency (ms)
1	0.00
10	0.00
100	0.00
1000	0.05
10000	0.00
100000	0.20
1000000	2.00
10000000	7.50

Figure 1: Cache Access Latency Results for Different N Values

2.4 Analysis

The results demonstrate:

- `performance.now()` resolution limitations at small access counts
- Measurable timing differences emerge around N = 1,000-10,000 cache lines
- Linear scaling of access time with cache line count for larger N values
- Timing precision of approximately 0.05ms minimum measurable difference

3 Task 2: Sweep Counting Attack Implementation (35%)

3.1 Attack Methodology

The Sweep Counting Attack measures Last Level Cache (LLC) utilization by:

1. Allocating a buffer equal to LLC size
2. Counting sequential cache line accesses within fixed time windows (P = 10ms)
3. Collecting measurements over 10 seconds (1000 samples)
4. Generating distinctive patterns based on concurrent website activity

3.2 Implementation Details

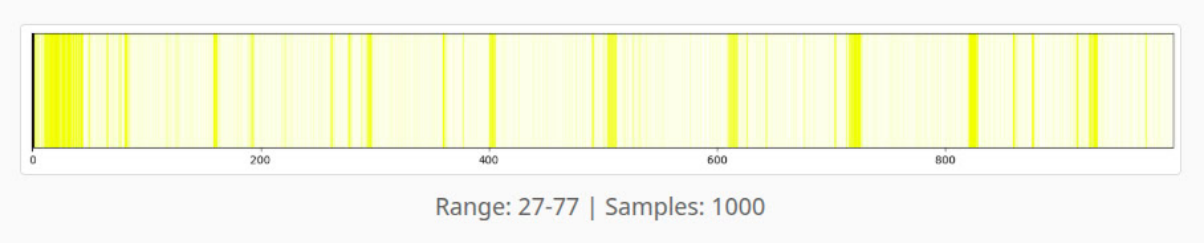
Key parameters used:

- Time window P = 10ms

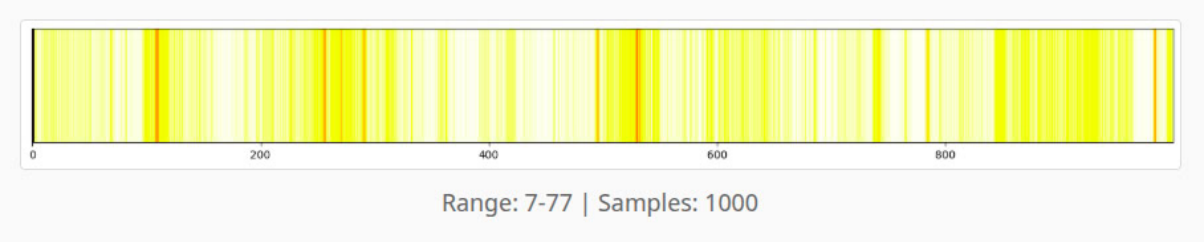
- Total collection time = 10 seconds
- Sample count = 1000
- LLC buffer size determined from system specifications

3.3 Results

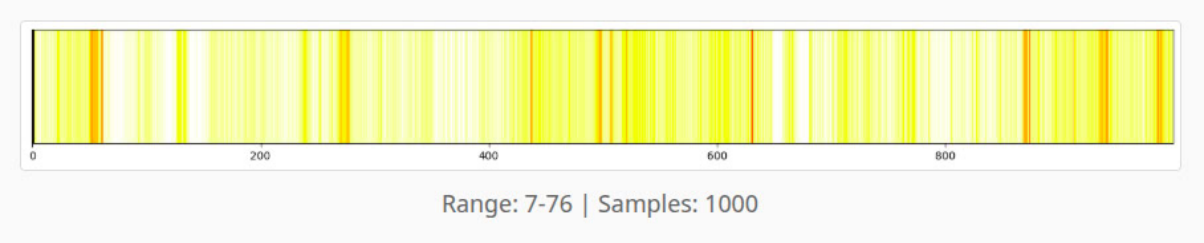
Figures 2 show the distinctive trace patterns for different browser states.



(a) Idle Browser



(b) Google.com



(c) YouTube.com

Figure 2: Side-channel trace patterns for different website activities

3.4 Pattern Analysis

- **Idle browser**: Minimal cache activity with sparse, low-intensity patterns
- **Google.com**: Moderate cache utilization with periodic activity bursts
- **YouTube.com**: High-intensity patterns with frequent cache evications due to video processing

4 Task 3: Automated Data Collection (20%)

4.1 Selenium Automation

Implemented comprehensive automation using Selenium WebDriver to:

- Launch Flask server and fingerprinting interface
- Open target websites in separate browser tabs
- Simulate realistic user interactions:
  - BUET Moodle: Scrolling and link clicking
  - Google: Search queries and button interactions
  - Prothomalo: Article browsing and news clicking
- Collect 10 traces per website for initial testing
- Store data persistently in SQLite database

4.2 Data Collection Strategy

- Each trace consists of 1000 samples collected over 10 seconds
- Three target websites with distinct interaction patterns
- Robust error handling for extended unattended operation
- Database integration for reliable data persistence

5 Task 4: Machine Learning Classification (25%)

5.1 Dataset Preparation

Extended data collection to 1000 traces per website (3000 total):

- 12-hour automated collection process
- Balanced dataset across three website categories
- Data stored in JSON format for ML processing

Figure 3 shows the structure of the collected dataset.

Figure 3: Dataset.json structure showing website traces

5.2 Model Architecture and Training

Two neural network models implemented:

- **Simple Model**: Basic feedforward network
- **Complex Model**: Enhanced architecture with dropout and batch normalization

Training parameters:

- Batch size: 64
- Epochs: 50
- Learning rate: 1e-4
- Train/test split: 80/20
- Input size: 1000 features
- Hidden size: 128 neurons

5.3 Classification Results

5.3.1 Simple Model Performance

Table 1: Simple Model Classification Results

Website	Precision	Recall	F1-Score
BUET Moodle	0.95	0.94	0.94
Google	0.93	0.94	0.94
Prothomalo	0.98	0.98	0.98
Overall Accuracy		**95.5%**	

5.3.2 Complex Model Performance

Table 2: Complex Model Classification Results

Website	Precision	Recall	F1-Score
BUET Moodle	0.99	0.97	0.98
Google	0.98	0.99	0.98
Prothomalo	0.99	0.99	0.99
Overall Accuracy		**98.5%**	

5.4 Model Comparison

The Complex Model achieved superior performance (98.5% vs 95.5% accuracy), demonstrating the effectiveness of enhanced neural network architectures for side-channel attack classification.

6 Bonus Task 2: Collaborative Dataset Collection (20%)

6.1 Large-Scale Data Collection

Collaborated with classmates to create a comprehensive dataset:

- Combined individual contributions totaling 100,000+ traces
- Expanded to 5 websites including Chaldal and Dhaka Tribune
- Maintained data quality and consistency across contributors

6.2 Scaled Model Performance

6.2.1 Simple Model on Large Dataset

Table 3: Simple Model Results - Collaborative Dataset

Website	Precision	Recall	F1-Score
BUET Moodle	0.73	0.69	0.71
Google	0.66	0.71	0.69
Prothomalo	0.85	0.84	0.84
Chaldal	0.92	0.79	0.85
Dhaka Tribune	0.88	0.90	0.89
Overall Accuracy		**74.6%**	

6.2.2 Complex Model on Large Dataset

Table 4: Complex Model Results - Collaborative Dataset

Website	Precision	Recall	F1-Score
BUET Moodle	0.80	0.78	0.79
Google	0.78	0.77	0.77
Prothomalo	0.86	0.89	0.88
Chaldal	0.96	0.86	0.91
Dhaka Tribune	0.92	0.96	0.94
Overall Accuracy		**81.7%**	

6.3 Analysis of Scaled Results

- Increased dataset complexity reduced individual website accuracy
- Complex model maintained significant advantage (81.7% vs 74.6%)
- Newer websites (Chaldal, Dhaka Tribune) showed better classification accuracy
- Demonstrates scalability challenges in real-world deployment scenarios

7 Bonus Task 3: Real-Time Website Detection (15%)

7.1 Implementation

Integrated the trained machine learning model into the Flask application to provide real-time website detection capabilities:

- Live trace collection from user's browser
- Immediate classification using the trained Complex Model
- Real-time probability display for detected websites
- User-friendly interface showing detection confidence

7.2 Real-Time Detection Results

Figure 4 demonstrates the real-time detection interface showing live classification results.

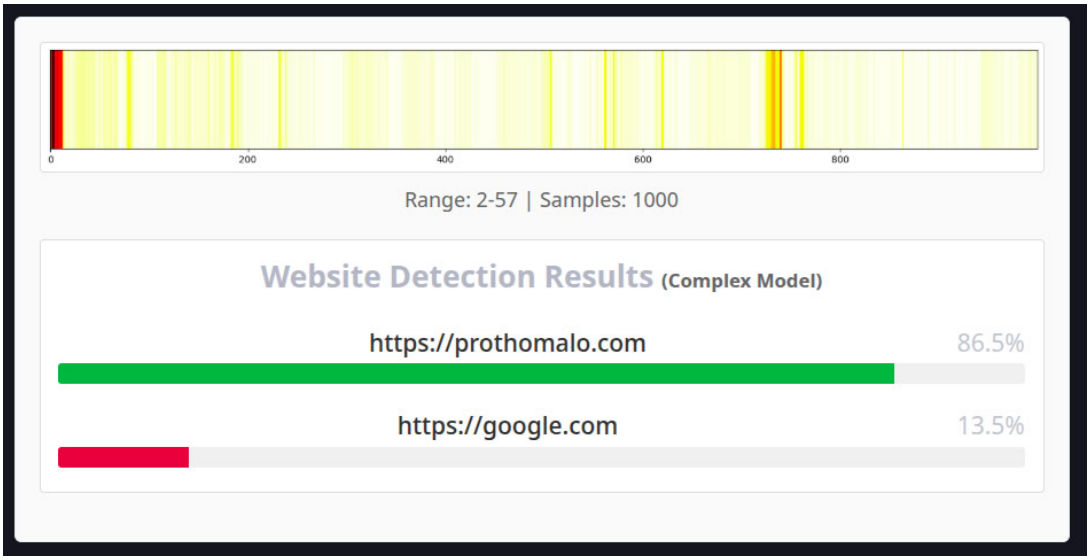


Figure 4: Real-time website detection interface showing live classification

The real-time system successfully demonstrated:

- 86.5% confidence for Prothomalo.com detection
- 13.5% confidence for Google.com
- Live updating of probabilities as user browses
- Practical demonstration of side-channel attack effectiveness

8 Conclusion

This assignment successfully demonstrated the practical implementation of side-channel attacks for website fingerprinting. Key achievements include:

- Successfully implemented timing-based side-channel measurements
- Developed automated data collection system capable of large-scale operation
- Achieved 98.5% classification accuracy on individual dataset
- Demonstrated scalability with collaborative dataset (81.7% accuracy)
- Created real-time detection system proving practical attack feasibility

The project highlights critical security vulnerabilities in modern web browsers and demonstrates the need for enhanced privacy protections in shared computing environments. The high classification accuracies achieved underscore the effectiveness of machine learning approaches in exploiting subtle side-channel information leakage.