DF-SCA: Dynamic Frequency Side Channel Attacks are Practical

Debopriya Roy Dipta and Berk Gulmezoglu
roydipta@iastate.edu bgulmez@iastate.edu

IOWA STATE UNIVERSITY, IA, USA
What is Side Channel Attack?

- Side-channel attacks use unintentional information leakage from secure chips to compromise their security.

- Compromising security:
  - Cryptographic key recovery
  - Website fingerprinting
  - Keystroke detection

- These unintentional information can be of different types:
  - Timing information
  - Power dissipation
  - Electromagnetic fields
  - Micro-architectural information

Ref: Gamaarachchi, H. and Ganegoda, H., 2018. Power analysis-based side channel attack. arXiv preprint arXiv:1801.00932.

- Micro-architectural side-channel attacks refer to a side-channel attack that exploit information leakage from the hardware infrastructure itself.
DF-SCA: Dynamic Frequency Side Channel

- Software-based dynamic frequency side-channel attack
- Applicable on Linux and Android OS devices
- Exploit unprivileged access to cpufreq interface

- Exploited in the context of covert channels and cryptographic attacks
- However, it has not been investigated to infer user activity, e.g.,
  - Website fingerprinting
  - Keystroke detection

**Challenge!**

- Noisy measurements
- Low resolution

Still dynamic frequency readings through Linux cpufreq interface provide sufficiently-detailed information on the user activity on Intel, AMD, and ARM architectures.
Dynamic Voltage and Frequency Scaling (DVFS)

- Allow switching between different frequency/voltage configurations based on the dynamic CPU resource demand.
- The rapid frequency changes are adjusted through different algorithms depending on the target application.

**CPUFreq Subsystem:**

- Responsible for the performance scaling of the CPU in a Linux kernel-based operating system.
- Comprises of three layers of code:
  - **Core:** Defines the layout of basic framework.
  - **Scaling governor:** Defines different frequency scaling algorithms to predict the CPU latency.
  - **Scaling driver:** Access a specific hardware interface to change the P-state based on the request set forth by the scaling governors.
Dynamic Voltage and Frequency Scaling (DVFS)

PolicyX Interface:

- CPUFreq core generates a sysfs directory named `cpufreq`, under `/sys/devices/system/cpu` path
- Within this directory a policyX sub-directory exists for all of the CPUs associated with the given policy
- policyX directories include policy-specific files to control CPUFreq behavior based on the corresponding policy objects.
- CPUFreq core generates several attributes dependent on the scaling governors and drivers, such as:
  - `scaling_cur_freq`
  - `scaling_min_freq`
  - `scaling_max_freq`
  - `scaling_available_governors`
  - `scaling_governor`
  - `scaling_driver`
Dynamic Voltage and Frequency Scaling (DVFS)

Scaling governor:
• **Performance** governor keeps the CPU around the highest frequency, within the `scaling_max_freq` policy limit
• **Powersave** governor keeps the core frequency low when there is no workload still within the `scaling_min_freq` policy limit.
• **User-space** governor allows userspace application to set the CPU frequency for the associated policy
• **Ondemand** governor uses CPU load to determine the CPU frequency selection metric
• **Conservative** governor sets the CPU frequency selection metric based on the CPU load.
• **Interactive** governor is designed for latency-sensitive, interactive workloads
• **Schedutil** governor was designed to estimate the load based on the scheduler’s Per-Entity Load Tracking (PELT) mechanism.

Scaling driver:
• Intel Core CPUs on Linux → Intel P-state driver
• AMD architecture → ACPI P-state driver
• Android systems → specialized frequency scaling driver called *msm*
Threat Model

- **Offline Phase:**
  - ✓ Attacker monitors the dynamic CPU frequency in his own system while rendering different websites.
  - ✓ A multi-class classification model is trained with the collected frequency measurements.

- **Online Phase:**
  - ✓ Attacker places a malicious code in a user-space application installed by the victim in his/her device.
  - ✓ Monitor the current frequency in the victim’s system.
  - ✓ Implement a cross-core side-channel attack through the current frequency readings.
  - ✓ Attacker collects a single trace during the website rendering.
  - ✓ Forward to the attacker’s server in which the pre-trained model is located.
  - ✓ Finally, the model is queried in the attacker server to classify the visited website.

**Assumptions:**
- ✓ Victim’s device is only running a particular browser instead of many applications at a time.
- ✓ System Configuration (Attacker and Victim) needs to be matched.
## Experimental Setup

### Intel Comet Lake
- CPU Model: Intel(R) Core (TM) i7-10610U CPU @1.80GHz
- Scaling Governors: powersave (default), performance
- Linux kernel version: 5.11.0-46-generic

### Intel Tiger Lake:
- CPU Model: Intel(R) Core (TM) i7-1165G7 @ 2.80GHz
- Scaling Governors: powersave (default), performance
- Linux kernel version: 5.13.0-44-generic

### AMD Ryzen 5:
- CPU Model: AMD Ryzen 5 5500U CPU with Radeon Graphic
- Scaling Governors: ondemand (default), powersave, performance, conservative, userspace, schedutil
- Linux kernel version: 5.13.0-44-generic

### ARM Cortex-A73:
- CPU Model: Four ARM Cortex-A53 and Four ARM Cortex-A73 cores
- Scaling Governors: interactive (default), powersave, performance, ondemand, conservative, userspace
Website Detection: Data Collection

Algorithm 1: Data Collection Algorithm for Each Website

// $T_i$ is the interval between each readings
// $N_s$ is the number of samples
// $N_M$ is the number of measurements per website
// $url$ is the web-page address
// $f$ is the CPU frequency

Input: $T_i, N_s, N_M, url$

Output: $f$

1. for $i \leftarrow 1$ to $N_M$
do
2. Run $url$ in the browser;
3. for $j \leftarrow 1$ to $N_s$
do
4. \hspace{1cm} $f[i, j] \leftarrow$ Read scaling_cur_freq;
5. \hspace{1cm} sleep $T_i$;
6. Close the browser;
7. sleep 1s;

Selected parameters in the Algorithm:

- $T_i = 10$ ms
- Google-chrome: $N_s = 1000$
- Tor browser: $N_s = 3000$
- $N_M = 100$

Reading CPU frequency:
/sys/devices/system/cpu/cpu1/cpufreq/scaling_cur_freq
Website Detection: Data Collection

- Each website has a distinct pattern as the contents of these websites include different JS scripts, images, HTML documents, and plug-in objects.

- CPU workload generates a unique fingerprint on the frequency readings for individual website.

A common pattern exists while visiting the same websites for multiple measurements.
Website Detection: Data Collection

cpufreq Resolution:

- Higher resolution enables attackers to capture a more detailed fingerprint
- We observe that the number of repeated values increases with the decreasing amount of delay between each reading
- The optimal delay is 10 ms for Intel and AMD architectures
- The speed of querying the cpufreq interface on Android devices is different than Intel and AMD architectures
- This value is defined by the min_sample_time in the interactive governor, which is set to 20 ms by default.
# Website Detection: Performance Evaluation

## Table 2: Test accuracy for different setups with their default scaling governor mode explored with four ML models

| Micro-architecture | Governor | Browser          | Test Accuracy |
|--------------------|----------|------------------|---------------|
|                    |          |                  | CNN | SVM | KNN | RF  |
| Intel Comet Lake   | powersave| Chrome           | 94.5%| 92.0%| 74.6%| 93.7%|
|                    |          | Tor              | 73.7%| 64.9%| 33.6%| 63.6%|
|                    |          | Tor (Top 5 score)| 93.0%| 86.6%| 54.0%| 86.2%|
| Intel Tiger Lake   | powersave| Chrome           | 97.6%| 95.8%| 84.3%| 93.0%|
|                    |          | Tor              | 68.7%| 51.9%| 16.2%| 30.4%|
|                    |          | Tor (Top 5 score)| 86.1%| 78.7%| 30.9%| 55.0%|
| AMD Ryzen 5        | ondemand | Chrome           | 93.1%| 90.4%| 78.4%| 84.9%|
|                    |          | Tor              | 60.3%| 50.8%| 24.7%| 29.8%|
|                    |          | Tor (Top 5 score)| 87.0%| 83.2%| 46.5%| 58.2%|
| ARM Cortex-A73     | interactive| Chrome          | 87.3%| 71.7%| 38.6%| 69.6%|
### Website Detection: Related Work

Table 3: Previous works based on different side-channel profiling techniques for website fingerprinting. For each work, attack vector, resolution, targeted browser, classification accuracy, and number of websites profiled are given.

| Work                        | Attack Vector      | Resolution | Browser          | Accuracy (%) | # of Websites |
|-----------------------------|--------------------|------------|------------------|--------------|---------------|
| DF-SCA                      | Frequency scaling  | 10 ms      | Chrome/Tor       | 97.6         | 100           |
| Rendered Insecure [32]      | GPU memory API    | 60 µs      | Chrome           | 90.4         | 200           |
| PerfWeb [13]                | Performance counters | 40 µs    | Chrome/Tor       | 86.4         | 30            |
| RedAlert [53]               | Intel RAPL         | 1 ms       | Chrome           | 99           | 37            |
| Shusterman et al. [43]      | Last-level cache  | 2 ms       | Firefox/Chrome/Tor | 80           | 100           |
| Spreitzer et al. [47]       | Data-usage        | 20 ms      | Tor              | 95           | 100           |
| Zhang et al. [52]           | iOS APIs          | 1 ms       | Safari           | 68.5         | 100           |
| Memento [18]                | procfs             | 10 µs      | Chrome           | 78           | 100           |
| Loophole [48]               | shared event loop | 25 µs      | Chrome           | 76.7         | 500           |
Keystroke Detection

- We assume that a phone user enters her password to log into his account in a banking application
- Considered Banking Application: Bank of America (BoA)
- Sampling rate: **20 ms**

- The collected keystrokes have three common properties
  ✓ A single keystroke length changes between 8 and 12 samples
  ✓ The big cores’ frequency increases up to 1.6GHz
  ✓ If two consecutive keystrokes are close to each other, the length of a keystroke pattern is higher than 12 samples.

- It takes 200 ms in average to decrease the frequency from peak to idle frequency level
- Hence, an attacker is able to distinguish the keystrokes that have at least 200 ms between each key press with DF-SCA.

- Our goal is not to outperform the existing works in the keystroke attack literature, but rather demonstrates DF-SCA attack has sufficient resolution and accuracy to perform a password detection attack.
Keystroke Detection

- Selected password: **50 out of 200** most used passwords on web
- Length of the password varies from **6 to 9 characters**.
- The phone user entered **50** distinct passwords for at least **10 times**
- In total, **1252 password measurements** were collected from **50 distinct passwords**
- The achieved keystroke detection rate is **95%**
- The inter-keystroke timings are determined
- **10 measurements** for each password are selected to evaluate the password detection accuracy
- A **Kth-nearest neighbor (KNN) model** is trained with the measurements.

- The model can guess the correct password with **88% success rate with one guess**
- With only **3 guesses**, the success rate is **97%**
Countermeasures

- **Restricting Access Privilege** for `cpufreq` interface from userspace applications in Linux OS.
- **Decreasing the update interval** of the `cpufreq` interface
  - ✓ With lower resolution, the amount of information leaked by DF-SCA can be diminished significantly
- **Artificial noise** can be introduced by the system to mask the rapid frequency changes in the system
  - ✓ Example: by randomly inserting workloads in the system
  - ✓ Since side-channel analysis takes advantage of Deep Learning algorithms frequently, adversarial obfuscation techniques can also be implemented to fool the Deep Learning models
- Similarly, keystroke attacks can be eliminated by **introducing additional keystrokes** to make the distribution more uniform
Impact of Different Scaling Governors

• **Intel Tiger Lake:**
  - Accuracy improves slightly when the scaling governor is changed to *performance* from *powersave.*

• **AMD Ryzen 5:**
  - The default scaling governor *ondemand* gives the highest website classification accuracy.
  - The *performance* and *powersave* governors drop the classification.

Table 4: The impacts of different scaling governors on website fingerprinting accuracy for Intel Tiger Lake and AMD Ryzen 5 architectures

| Scaling governor | Test Accuracy (%) |
|------------------|-------------------|
|                  | Intel Tiger Lake  | AMD Ryzen 5     |
| performance      | 97.8              | 68.1             |
| powersave        | 97.6              | 75.3             |
| userspace        | N/A               | 80.1             |
| ondemand         | N/A               | 97.6             |
| conservative     | N/A               | 96.7             |
| schedutil        | N/A               | 97.6             |
Impact of Different Scaling Governors

- Unlike other scaling governors, for *userspace* governor the CPU keeps the core frequency below its base frequency.

- Although the variation is quite low, a similar pattern for the same web page can still be noticeable from this figure.
Universal ML Model for different microarchitectures

• Previously, we trained separate ML models for Intel, AMD, and ARM architectures to obtain the highest website fingerprinting accuracy.

• We are interested to know whether it is possible to replace the individual ML models with a universal ML model.

• This will facilitate the attacker to perform website fingerprinting without requiring to know the exact targeted microarchitecture.

Table 5: The universal ML Model training and evaluation for Intel Tiger Lake, Intel Comet Lake, and AMD Ryzen 5 architectures

| Micro-architecture                                      | Test Accuracy (%) |
|--------------------------------------------------------|-------------------|
| Intel Comet Lake + Intel Tiger Lake                    | 95.9              |
| Intel Comet Lake + Intel Tiger Lake + AMD Ryzen 5      | 92.3              |

• Combined the CPU frequency traces of the Intel microarchitectures to train one CNN model and achieved test accuracy of 95.9%.
• Later, combined both Intel and AMD frequency traces, which leads to 92.3% accuracy with one CNN model.
Outcomes

- The attacker only needs to collect 10 seconds of the frequency values to detect the websites in Google Chrome browser applicable to Intel, AMD, and ARM devices.

- Even though DF-SCA’s resolution is significantly lower than many previous attacks, it is still possible to detect the visited websites with a high accuracy.

- Moreover, victim keystrokes can be detected with 95% success rate which yields to a successful password recovery attack with a simple ML classification.

- As a result, DF-SCA is a potential threat for all the components that take advantage of DVFS technology.

- The access privilege restriction or artificial noise injection might become fruitful countermeasures against such a threat.

The dataset and the code are made available in GitHub: https://github.com/Diptakuet/DF-SCA-Dynamic-Frequency-Side-Channel-Attacks-are-Practical.git
THANK YOU

QUESTIONS?