Inferno : Side-channel Attacks for Mobile Web Browsers

Manuel Philipose, Matthew Halpern, Pavel Lifshits, Mark Silberstein, Mohit Tiwari

Electrical and Computer Engineering, The University of Texas at Austin

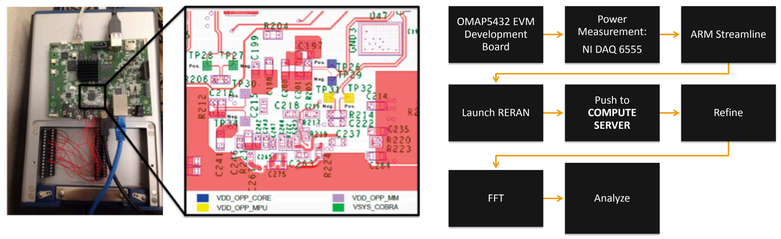
Background and Introduction

The Android security architecture leverages the traditional Linux operating security controls to protect a user's data. However, user-based permissions models and application sandboxes do not mitigate the vulnerability of side-channels.

We demonstrate power consumption as a side-channel on mobile devices. While web pages may look aesthetically similar, the web browser exercises different behaviors while rendering the underlying code. The variance between the browser's behavior and power consumption implies that different webpages consume different amounts of power. Thus, webpages can be uniquely identified from one another by analyzing power traces collected during a page load.

Side-channel attacks have not been studied for general purpose systems such as mobile devices. In our evaluation, we use this side-channel to reveal a mobile user's browsing activity from the hardware level power traces from the CPU, GPU, PMIC and DRAM with an 80% accuracy.

Methods



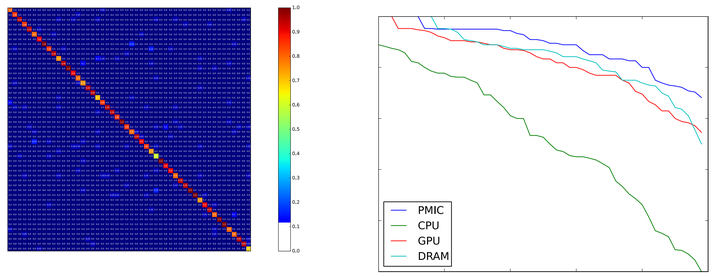
Our experiments are performed on real hardware and mainstream mobile operating system that simulates a comparable attack on a mobile device. In this section, we discuss our testing infrastructure that allows us to conduct a power side-channel attack on a mobile device.

Power Collection: traces are collected from an externally instrumented evaluation board. Sense resistors are attached to the PMIC, CPU, GPU, and DRAM.

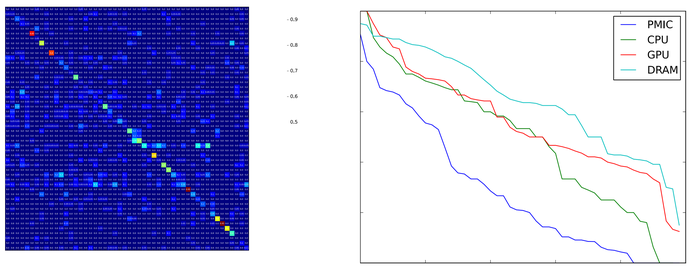
Collecting Traces: Prior to identifying browser behavior, we create a database of signatures corresponding to the applications power consumption which is either stored on the device or a remote server. Immediately before the page load, the ARM Gator begins gathering raw power traces from 4 different channels: PMIC, CPU, GPU and DRAM. This daemon runs for 30 seconds after the initial page load.

Signature Generation: The initial data contains 30 seconds worth of sampled data, but later trimmed to 5 seconds. The trace is then converted into the frequency domain. Each spectrogram is a set of tuples which keep track of a magnitude and a frequency for that magnitude, essentially forming a histogram of magnitudes for a given webpage trial.

Attack Results



Countermeasure Results



Machine Learning

We use the k-Nearest Neighbors (kNN) algorithm for initial classification. kNN is a simple algorithm that identifies the k-nearest neighbors of a given feature vector c using their Euclidian distance.

In the case of our experiment, we use the frequency of magnitudes derived from the time series power trace as the features, and the webpage's name as the label. Running this classifier allows us to classify the given feature vector c with the label of a webpage.

After using this classifier, we can use a confusion matrix, also known as an error matrix, to visualize the accuracy of the model. The results are shown in the figures above.

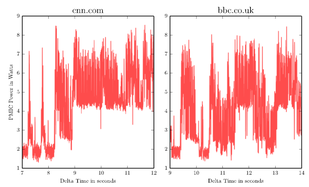
$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

Evaluation

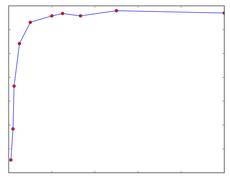
To evaluate our attack, we perform a 10-fold cross validation for each parameter configuration. We used 5 different parameters, varying k from 1 through 5. This process splits our signature database into a 90/10 training/test set respectively. This technique allows us to see how well our model would perform in practice, without necessitating testing single unknown traces.

Discussion

Information leakage is a pervasive problem, not limited to the hardware-level. Temporal changes in any part of the mobile system stack propagate to the lower levels. It is possible to correlate this seemingly insignificant leaked data to reveal a user's secrets.

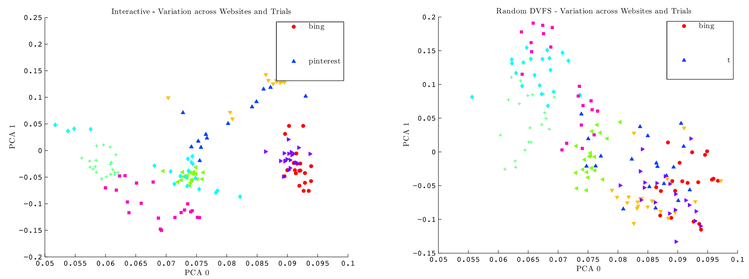


Time-series power traces for a webpage load are specific, consistent and distinguishable. 20 traces were collected for each website to generate an accurate signal for signature recognition with kNN.



The attack can be down-sampled to roughly 200 Hz while keeping the same accuracy. The original attack consists of 5000 features, and an attack at 200 Hz consists of simply 25 features.

To this end, a random DVFS governor to vary the CPU frequency in an attempt to mitigate the attack. Dynamic Voltage and Frequency Scaling (DVFS) is a commonly used technique to alter the frequency of the CPU, on the fly, to conserve power. Essentially, DVFS uses a reduction in CPU clock frequency to reduce supply voltage, and thus save power.



The Principal Component Analysis (PCA) for the default Android governor (interactive) and the Random DVFS governor show that the features with the highest variance are significantly less clustered.

Conclusions

- Power traces for webpage loads are specific and consistent. They can be used by an attacker to reverse engineer a user's browsing history.
- This attack can be mitigated either by adding noise to the power trace or removing the signal.
- Random DVFS governor was built to add coarse grained noise to the power trace. This effectively mitigated a kNN based attack, but cannot provably be mitigated other techniques.

mphilipose@utexas.edu