Website Fingerprinting by Power Estimation Based Side-Channel Attacks on Android 7

Yi Qin and Chuan Yue
Department of Computer Science
Colorado School of Mines
Golden, CO, USA 80401
Email: {yiqin, chuanyue}@mines.edu

***Abstract*—Power consumption data on smartphones can be used to infer user information like web browsing activities and location. Since the power consumption data was traditionally considered harmless to user privacy and developers need to use power information to develop energy-efficient apps, it can be obtained without user permission on Android systems older than Android 7. However, due to the updates in Android 7, the traditional methods of accessing power consumption data either don't work anymore or the data is coarse-grained, and less information can be leaked. In this paper, we propose a power estimation method and design a power estimation based side-channel attack to perform website fingerprinting. Results show that website fingerprinting can be performed with high accuracy using power estimation data on Android 7.**

***Keywords*-Power side-channel attacks; Power estimation; Android; Website Fingerprinting; Privacy**

I. Introduction

Smartphones play an increasingly critical role in our daily life. We not only rely on our phones to perform various tasks like web browsing, shopping and social interactions but also store private information such as passwords, credit card, and even biometric information like fingerprints. It is important to make sure the sensitive information cannot be compromised.

Security problems on smartphones draw great attention and researchers have studied various side-channel attacks. Power side-channel attacks are also called power analysis attacks. On smartphones, power side-channel attacks mean attackers try to infer user information by analyzing power consumption data. Web browsers on smartphones increasingly take advantages of hardware to improve their performance. This leads to the fact that hardware utilization of web browsers is often proportional to webpages' complexity. Besides, since operating systems on smartphones such as Android are trying to be energy-saving, power consumption is tightly proportional to hardware utilization. These factors make it easier for attackers to infer webpages because **power consumption data reflects the complexity, and in turn, the distinctiveness of each webpage.**

Power consumption data was traditionally considered harmless to user privacy and thus can be obtained easily by attackers without user permission on Android systems older than Android 7. However, on Android 7, the situation is improved by restricting access to system files which contain power consumption data, and by only providing coarse-grained data from API. Therefore, Android 7 is a more secure system in terms of power side-channel attacks by making the attacks harder but not impossible; many existing power side-channel attacks that can infer running apps and user locations (Section II-B1) on older versions of Android systems do not work on Android 7 anymore.

In this work, we propose a simple power estimation method for power side-channel attacks despite the fact that the traditional methods of accessing power data are no longer available on Android 7, and present a method of website fingerprinting based on the estimated power data. We demonstrate that our proposed power estimation model can accurately estimate the fluctuation of power data, and the modeled power data can be further used for effective website fingerprinting. It is important to note that ***no user permission is needed and the device does not need to be rooted for power estimation and website fingerprinting***. In other words, ***our attacks can be performed by a regular app without user permission***. We hypothesize that the modeled power data can also be used for performing other types of inference attacks such as running app inference and user location inference (Section II-B1), thus immediately reviving those existing power side-channel attacks on Android 7 without incurring too much extra effort to attackers. This hypothesis could be valid as long as a user's sensitive activities are tightly correlated to the power consumption on a smartphone, and we leave its test as our future work.

The main contributions of this paper are summarized as follows:

- A power estimation model is proposed to estimate the fluctuation of power data for potentially performing different types of power side-channel attacks without user permission;
- A method of website fingerprinting based on the modeled power data is presented;
- The proposed power estimation model and website fingerprinting method are evaluated.

The rest of the paper is structured as follows. Section II introduces the background of the power data acquisition methods and the related work. Section III describes the power estimation model as well as its evaluation. Section IV presents the method of website fingerprinting and its evaluation. Section V discusses the limitations of this work and the potential future work. Section VI concludes the paper.

II. Background and Related Work

In this section, we summarize a review of the power data acquisition methods on Android systems older than Android 7. Besides, related work is described.

A. Power Data Acquisition

In general, there are two types of power data acquisition methods on smartphones including the physical approach and the software approach. The physical approach means that power data is measured using a power meter via electrical outlets[1] or USB ports[2] when a smartphone is charging. However, this approach is not realistic for most of the attacks since it is only effective when the phone is charging. The software approach indicates that power data is obtained by an app when the phone is not charging. In this paper, we focus on the software approach. Commonly used methods of power data acquisition from an Android app are summarized as follows.

1) System files: Power consumption data including instant current and instant voltage can be obtained from system files without user permission on Android systems older than Android 7[3], [4]. Table I shows the information of the system files. The update period of the files is 175.8 ms[5].

Table I: System Files of Current and Voltage

File Location	Description	Unit
/sys/class/power_supply /battery/voltage_now	Instant voltage reading of battery	μV
/sys/class/power_supply /battery/current_now	Instant current reading of battery	μA

However, when an app is trying to access those system files on Android 7, the root permission is needed. Therefore, attackers cannot use this method for power data acquisition on Android 7.

2) API: BatteryManager[6] is a class which provides battery information of a smartphone. The following two properties of BatteryManager are usually used to get current and voltage data without user permission:

- BATTERY_PROPERTY_CURRENT_NOW
- EXTRA_VOLTAGE

The first one is the instantaneous battery current in microamperes. The second one is the current battery voltage level in millivolt. The current and voltage data from BatteryManager on Android 7 is coarse-grained at seconds level based on our observation, thus attackers can hardly obtain any useful information from it to infer user activities. Therefore, this approach of collecting data is also not feasible for attackers on Android 7.

B. Related Work

1) Power Side-Channel Attacks: Attackers can infer many types of sensitive user information from power consumption data. Yan et al. [3] demonstrated that power data on mobile devices can be used to infer running apps, password length, user locations, etc. Chen et al. [7] worked on mobile app fingerprinting by analyzing power consumption data. They claimed their accuracy of identifying apps is up to 92.9%.

Michalevsky et al. [4] proposed PowerSpy to infer user locations using power consumption data. The authors demonstrated that power usage data can leak user locations. They also observed that the power consumption of a smartphone is measurably affected by the smartphone's distance from the surrounding cellular base station.

Fan et al. [8] studied the power side-channel attacks on home appliances via smart meters. They demonstrated that attackers can infer ON/OFF events of home appliances with high accuracy via smart meters. Also, they proposed a Reactive Power Obfuscation method to mask the true power demand from a smart meter.

2) Website Fingerprinting: Clark et al. [1] identified webpages loaded on a desktop computer by measuring power consumption of the AC outlet. They used Fourier Transform to transform power traces in the time domain to the frequency domain and used an SVM classifier. Eventually, the webpage identification results showed that power side channel-attacks can be achieved with high accuracy.

Yang et al. [2] showed how power side-channel attacks can be performed via a public USB charging station to identity webpages. They were able to achieve over 90% webpage identification accuracy by taking such a physical approach. Moreover, they explored how the identification accuracy is affected by variables such as the battery charging level, users' interaction with the touchscreen, time span between training of the identification model and testing, wireless connection types such as WiFi and LTE, and website characteristics such as HTTP and HTTPS.

In this paper, we focus on smartphones and take the software approach instead of the physical measurement approach to obtain more realistic power consumption data for website fingerprinting.

3) Power Estimation: Power estimation on smartphones is commonly used for developers to monitor application power consumption data in order to make their apps energy-efficient and thus increase battery life. Ahmad et al. [9] categorized power estimation methods into two classes: code analysis based estimation and mobile component based

estimation. Code analysis based estimation explores base power consumption of code statements in an application or a webpage. Mobile component based estimation means power data is estimated based on the utilization of hardware components such as CPU, WiFi, LCD, etc.

Cao et al. [10] used code analysis based estimation to study the power consumption of mobile webpage loading by decomposing webpage code into the components such as HTML, CSS, JavaScript, and images.

Zhang et al. [11] provided a power estimation tool for applications called PowerTutor. It is constructed using mobile component based estimation and the components considered are CPU, WiFi, Audio, LCD, and Cellular. Yoon et al. [12] proposed a power estimation system called AppScope which monitors an application's kernel activities and estimates its power consumption.

The objectives of power estimation methods in these previous work are focused on providing accurate power data of webpages or applications to better optimize power consumption from the developers' perspective. However, in this work, we propose the power estimation model to estimate fluctuations of power data accurately (rather than absolute power data) from attackers' perspective. This is because the fluctuation of power data reflects user activities and the key for power side-channel attacks is to capture the fluctuation information from the power data. Besides, to the best of our knowledge, this is the first work using a power estimation method to perform website fingerprinting on Android 7.

III. A Power Estimation Model

As described in Section II-A, the common power data acquisition methods don't work anymore on Android 7. In this section, a power estimation model taking the mobile component based approach is introduced including our power data acquisition method, experimental setup, data collection, data preprocessing, power estimation model, and evaluation. As mentioned earlier in Section I, our power estimation model could be used not only for website fingerprinting but also for other types of attacks such as running app inference and user location inference. Also, since the existing power side-channel attacks (Section II-B1) only work on Android systems older than 7, our power estimation model provides a bridge to immediately revive those existing attacks.

A. Power Data Acquisition on Android 7

Since we take the mobile component based estimation approach, related components need to be identified. In [11], the main components used to estimate power data are CPU, WiFi, Audio, LCD, and Cellular. Since Audio and Cellular are not used in our experiments while LCD screen brightness level and WiFi ON/OFF state can be considered as constant, we only use CPU as the main component. Besides, we want to keep the power estimation model simple because the goal of power estimation is to capture the power estimation change accurately.

Table II: Files used to request CPU information

Parameter	File Location	Description
CPU frequency	/sys/devices/system/cpu /cpuX/cpufreq/scaling_cur_freq	Current scaling frequency of CPU X, X=0,1,...,7
CPU load related	/proc/stat	parameters of CPU[14], used to calculate CPU load

Since the system information like CPU load and CPU frequency is still available from the /proc entry and other system files without user permission, and CPU consumption is one of the main factors of power consumption [11], [13], our method is to estimate power consumption using the available CPU information. Table II shows where does the data come from. It is important to note that ***no user permission is needed to access these files and the device does not need to be rooted***. In other words, ***power estimation presented in this section and website fingerprinting attacks presented in Section IV can all be performed by a regular app without user permission***.

CPU scaling frequency can be directly obtained from /sys files listed in Table II. There are two CPU clusters. The first cluster includes the first four CPUs while the second four CPUs belong to the second cluster. The number of scaling frequencies for the first cluster is 11 and 14 for the second cluster.

CPU load is calculated by using parameters in /proc/stat [14] such as user time, nice time and system time, etc. First, the difference of each time parameter between previous and current measurement is computed as delta value because /proc/stat contains the amount of time CPUs spent in a specific state since the system was first booted. Then, CPU load is calculated using the following equation[15]:

$$\begin{aligned} CPUload =& (\Delta user + \Delta system + \Delta nice + \Delta hardirq \\ &+ \Delta softirq + \Delta steal)/(\Delta user + \Delta system \\ &+ \Delta nice + \Delta iowait + \Delta hardirq + \Delta softirq \\ &+ \Delta steal + \Delta idle) \end{aligned} \tag{1}$$

Our power estimation model is trained using the CPU frequency and CPU load and the measured power data from a power meter. Figure 1 is the workflow of our power estimation model. First, an attacker collects measured power data from a power meter and CPU information for webpage loading activities. Then the data is used to train the power estimation model. The parameters of the power estimation model are deployed in a malicious app on a user's smartphone to perform attacks.

B. Experimental Setup

In this paper, a Nexus 6P is used as the Android smartphone platform. Nexus 6P was released in 2015 and the

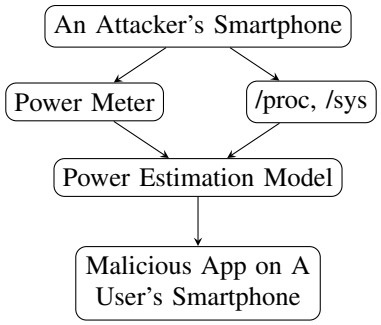

Figure 1: Workflow of the power estimation model

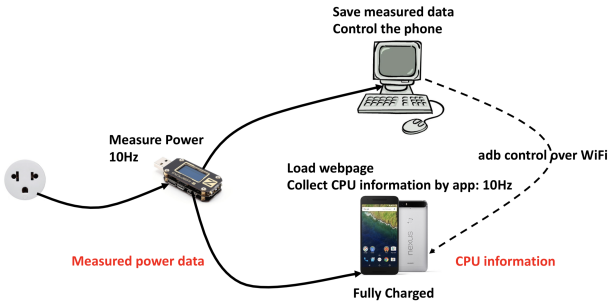

Figure 2: Experimental setup

operating system on the phone is updated to Android 7.1.2.

The power meter used is ChargerLAB POWER-Z Model KM001 which is worth $58 according to Amazon. It can measure current, voltage, and power consumption of the phone while the phone is charging. Its resolution is at 0.0001 A/V/W and we consider this meter as cost-effective and easy to purchase by anyone. There are six interfaces including USB-C in, USB-C out, USB in, USB out, Micro-USB, and HID (Human Interface Device) for communication and independent power supply.

Figure 2 shows the experiment setup. First, the power meter is connected to a power outlet through the USB-C in port and connected to the phone through the USB-C out port so that the meter can measure the charging power data. Besides, the meter is connected to a computer via the HID port. We can control the data recording process from the computer using the provided software for the power meter and save the data on the computer. Also, in order to control the webpage loading activities on the phone without physical touch on the screen, the phone is connected to the computer over WiFi using Android Debug Bridge (adb) which is a command-line tool allows developers to communicate with a device. An app collecting CPU information is running at the same time. The data collected from the app includes the following parameters: overall CPU load, scaling frequency in boolean for each CPU, and CPU load for each CPU.

C. Data Collection

Since the power meter can only measure the charging power, we use the charging power to train the power estimation model. So the estimated power data is charging power data instead of non-charging power data. However, what really matters to this work is the change or fluctuation of the power data related to certain activity. Therefore, the goal of power estimation in this paper is not to estimate the accurate power consumption data but ***rather to estimate the power estimation change accurately***. The phone is fully charged while collecting data so we can consider the difference between the charging power data and non-charging power data as a constant value while there are no activities on the phone.

The power meter is configured to measure the charging power at a sample rate of 10Hz. An app is used to collect CPU information including CPU load and CPU frequency. The sampling rate of the app is also 10Hz. The method of requesting CPU information is described in Section III-A.

The data is collected by the power meter as well as the app while loading webpages over adb WiFi. So the data we have is the measured power consumption data from the power meter, and CPU information from the app. Multiple websites are loaded for multiple times with an interval of 8s for more than 200s. Data of Amazon is used because the complexity of its homepage makes the power consumption data vary. Thus, we consider Amazon as representative.

D. Data Preprocessing

Ten loads of the measured power consumption data for Amazon and the corresponding CPU data are selected as training data for the power estimation model. There are two problems to deal with: data alignment and missing points.

(1) Data Alignment. We cannot guarantee the data is sampled at exactly the same time even if the sampling rates of the power meter and the app are both exactly 10Hz because the data is collected from two devices. Therefore, data alignment is needed to preprocess the data in order to minimize data error.

Since the data points are independent from each other, we use ranking as the data alignment method. The power data is ranked from low to high while overall CPU load is used to rank the CPU vectors.

(2) Missing Points. The sampling rate of the app is not exact 10Hz because there exists the problem of missing CPU points compared to power data from the meter. After ranking, measured power data is resampled to the same length of CPU data.

E. Power Estimation Model

In order to simplify the power estimation based side-channel attack and make it more feasible with less difficulties, our power estimation model is constructed by employing linear regression to get the coefficients of CPU

Table III: Power Estimation Model

Model	$P = \sum_{i=0}^{8} \beta_{l_i} L_i + \sum_{i=1}^{4} \sum_{j=1}^{11} \beta_{f_{ij}} F_{1j} + \sum_{i=5}^{8} \sum_{j=1}^{14} \beta_{f_{ij}} F_{2j} + \beta_0$	
Parameters	Symbol	Description
CPU load	L_i	CPU load, *i =0, 1,..., 8*, L_0: overall CPU load
	β_{l_i}	CPU load coefficient, *i =0, 1,..., 8*
CPU frequency	F_{1j}	0 or 1, the *j*th scaling frequency of the first cluster, i.e CPU 1-4
	F_{2j}	0 or 1, the *j*th scaling frequency of the second cluster, i.e CPU 5-8
	$\beta_{f_{ij}}$	Coefficient of the *j*th scaling frequency of *i*th CPU

variables. The key idea of linear regression is to consider power consumption as a linear combination of the variables. Equation (2) shows the relation between measured power data and the variables:

$$\begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_n \end{bmatrix} = \beta_0 + \beta_1 \cdot \begin{bmatrix} V_{10} \\ V_{11} \\ \vdots \\ V_{1n} \end{bmatrix} + ... + \beta_m \cdot \begin{bmatrix} V_{m0} \\ V_{m1} \\ \vdots \\ V_{mn} \end{bmatrix} \quad (2)$$

In this equation, P_i represents the *i*th measured power data. V_{ij} is the *i*th variable corresponding to *j*th power data. β_i is the coefficient of the *i*th variable. β_0 is a constant value representing all the stable factors.

Table III shows the power model and information of the variables and coefficients. Estimated power equals to the sum of CPU load overall and for each CPU times the corresponding coefficient, CPU scaling frequency in boolean for each CPU times the corresponding coefficient, and the constant coefficient. In the equation, 11 represents the number of scaling frequencies for the first cluster and 14 is that for the second cluster.

F. Evaluation

To evaluate the model, the experimental setup in Figure 2 is still used to collect the measured data and modeled data. The coefficients are deployed in the app while it collects the CPU information. Multiple websites are loaded while the power meter and the app are collecting the data. Based on the model in Table III, power data is calculated.

Since the power estimation model in this work aims to estimate the change of power data accurately rather than the actual value of power consumption, the modeled data is first shifted down along the vertical axis based on the difference of the average values between the modeled and measured data. Then, metrics including the average error and average absolute error [11] are used to evaluate the modeled data. Average error *avg* is the average of

$$\frac{measured - modeled}{measured}$$

Average absolute error *abs avg* is the average of

$$\left| \frac{measured - modeled}{measured} \right|$$

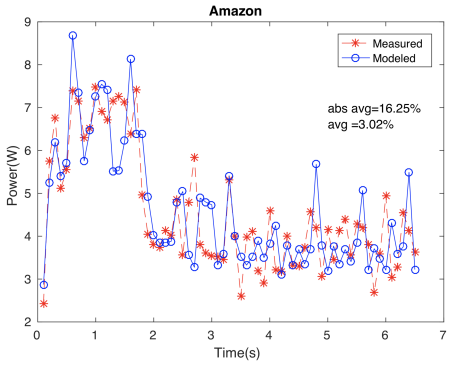

Figure 3: Power estimation data of Amazon

Figure 3 shows the power estimation of loading the homepage of `Amazon.com`. Since the model is trained using Amazon and the baseline of the measured and modeled data for Amazon should be the same, there is no vertical shift here. The average error is 3.02% while the average absolute error is 16.25%.

Then, the model is further evaluated while loading the homepages of four different websites. Figure 4 shows the evaluation results of the four websites. The modeled data is shifted down along the vertical axis with 1.2579 for Google. The average error is 4.32% while the average absolute error is 20.71%. As for Chase, modeled data is shifted down along the vertical axis with 0.4833. The average error is 6.22% while the average absolute error is 17.98%. Modeled data is shifted down 0.62 for Linkedin. The average error is 3.67% and the average absolute error is 19.77%. As for Bing, modeled data is shifted down 0.5666. The average error is 2.68% while the average absolute error is 17.53%.

In summary, the proposed power estimation model estimates the trend of power data well and the modeled data could be used for further website fingerprinting.

IV. Website Fingerprinting

In this section, we first present the threat model. Then, the method of website fingerprinting is described including data collection, feature extraction, etc. Also, evaluation is presented.

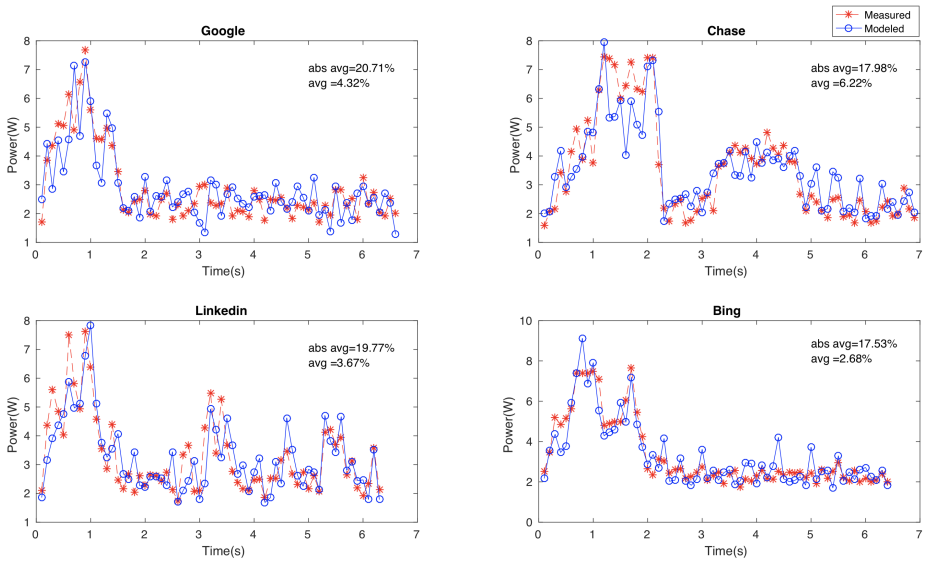

Figure 4: Power estimation data of four different websites

A. *Threat Model and Workflow*

In our threat model, an attacker is capable of collecting CPU information and estimating power data from a malicious app running in the background on the targeted user's phone. This app does not require any user permissions. It contains power estimation model parameters and collects the CPU information for power consumption estimation. The attacker aims to identify which webpage the user is browsing using a mobile browser based on the estimated power data. In this work, we assume there is only webpage loading activity running on the smartphone.

It is noteworthy that using CPU information to directly perform website fingerprinting (without first performing power consumption estimation) might work too, but the goal of this paper is to provide a bridge for power side-channel attacks on Android 7, and thus immediately revive those existing attacks reviewed in Section II-B1.

The workflow of our website fingerprinting method is described in Figure 5. A malicious app is running on a user's smartphone collecting CPU information while loading multiple known websites loaded from the WebView component of the app. The malicious app then estimate the power data using the collected CPU information and the power estimation model. After data preprocessing an attacker will have the training data. Then, a website fingerprinting model is trained and validated using the training data. With the trained model, whenever a user (i.e., victim) visits a website on a browser, the same process of collecting CPU information, estimating power consumption data, and preprocessing will be used to obtain the testing (i.e., attacking) data. The trained model will then be used to infer the user's webpage browsing activities.

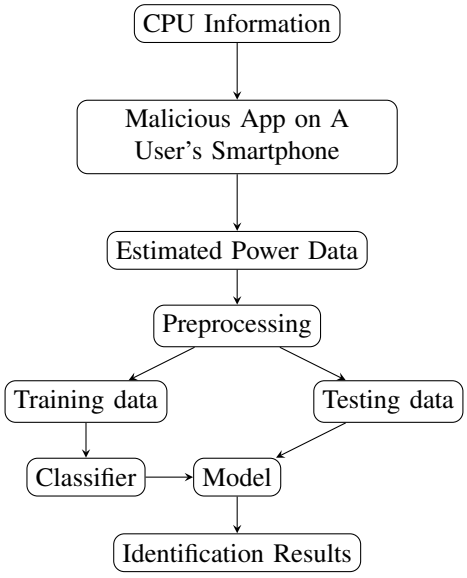

Figure 5: Workflow of the website fingerprinting method

B. *Data collection*

In this part, the power estimation data is obtained by the malicious app with the model coefficients in it. Here is a brief description of our data collection process. First, the smartphone is connected to a computer using adb remote debugging over WiFi. Then, the malicious app is started. After starting the app, an adb script is executed to automatically load the same web page for 50 times with a sleep for 8 seconds between every two loads. The time the script starts running is recorded.

At first, 25 websites are selected from the Alexa top 50 non-adult websites. Eventually, we settled with 20 websites.

The five websites are excluded because of following reasons:

- Visiting rate limited;
- A system app opened automatically;
- Too many videos automatically played;
- Different video ads automatically played for each load.

Since the websites with above characteristics lose the universality, the power consumption is not considered as typical. Table IV shows a set of websites used along with their corresponding labels. The power consumption data of 50 instances for each website are collected. An instance here means loading a webpage for one time. The 20 websites are ranked based on popularity and the first one is the most popular one.

After data collection, instances need to be split. 20 consecutively instances of the collected 50 instances for each website are selected. The indices start points of the 20 instances are selected after plotting the entire power trace. Then, the splitting is done by Python script.

Table IV: Websites Used

Label	Website	Label	Website
1	google.com	11	chase.com
2	facebook.com	12	paypal.com
3	amazon.com	13	apple.com
4	wikipedia.com	14	microsoft.com
5	netflix.com	15	stackoverflow.com
6	linkedin.com	16	wellsfargo.com
7	instagram.com	17	worldpress.com
8	craigslist.org	18	ask.com
9	live.com	19	fc.com
10	bing.com	20	vimeo.com

C. Feature Extraction

Each instance of power trace is transformed to the frequency domain using Fast Fourier Transform(FFT). Figure 6 shows an example FFT for one instance power trace of Google. Then the frequency range is divided into 6 equal-size bins corresponding to the frequency under 6Hz. There are two reasons for selecting the frequency under 6Hz. First, excluding high frequency bins helps to reduce the impact of high frequency noise. Second, since the length of each instance may not be the same and the time span for each instance is 8s, the FFT length may vary. For every bin, the average amplitude is computed as a feature of the instance. As a result, each instance will have 6 features.

D. Classifier

In this work, Support Vector Machine is used to identify webpages. In machine learning, support vector machines (SVMs) are supervised learning models which learn the boundary between classes.

For the classification task, the dataset is separated into training data and testing data. Each training instance contains one class label and six features. The goal of SVM is to learn

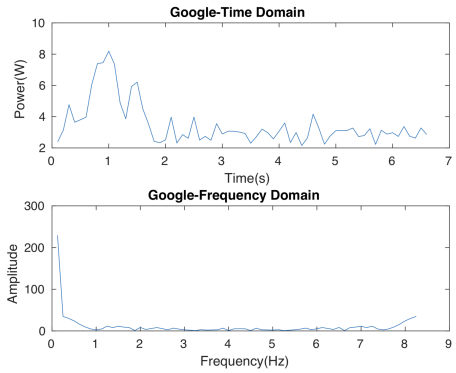

Figure 6: FFT for one instance power trace of Google

a model from the training data and predict the class labels of the testing instances given only the features.

The tool used in this work is called LIBSVM[16] which is a library for support vector classification, regression, and distribution estimation. It supports multi-class classification.

E. Training and Testing Dataset

Among the selected 20 instances for each website, 12 instances are used as training data and 8 instances are used as testing data. Therefore, the training dataset has $12 \times 20 = 240$ instances which are 240 vectors with 6 features each. The testing dataset has $8 \times 20 = 160$ instances which are 160 vectors with 6 features each.

F. Evaluation

Standard metrics of machine learning techniques are used to evaluate the performance of the power side-channel attacks. In the following definitions, tp and tn refer to true positives and true negatives and fp and fn refer to false positives and false negatives.

Precision, $tp/(tp + fp)$, is the proportion of correctly labeled instances for a class over all the labeled instances for the class. It indicates the quality of predictions.

Recall, $tp/(tp+fn)$, is the proportion of correctly labeled instances for a class over all the instances which should have been labeled for the class. It measures the sensitivity or completeness of predictions.

Four experiments with different number of websites in the training and testing dataset are performed. The numbers of websites are 5, 10, 15 and 20, respectively.

(1) 5 websites. The first five websites of the list in Table IV are selected. The total number of training instances is 60 while the testing instance number is 40. The confusion matrix is showed in Table V. The accuracy, precision, and recall are listed in Table VI.

The overall accuracy of identifying five websites is 72.5%. Class 5 (Netflix) has the highest precision and recall while class 1 (Google) has the lowest precision and recall. The reason could be that power consumption of Netflix is quite

different than other websites while power consumption of Google is not easy to distinguish from other websites.

Table V: Confusion Matrix of 5 Websites

		Predicted				
		1	2	3	4	5
Actual	1	3	2	0	3	0
	2	2	5	0	1	0
	3	1	0	6	1	0
	4	1	0	0	7	0
	5	0	0	0	0	8

Table VI: Precision, Recall, Accuracy of 5 Websites

Label	Precision	Recall	Accuracy
1	42.86%	37.5%	
2	71.43%	62.5%	
3	100%	75%	72.5%
4	58.33%	87.5%	
5	100%	100%	

(2) 10 websites. The first ten websites of the list in Table IV are selected. The total number of training instances is 120 while the testing instance number is 80. The confusion matrix is showed in Table VII. The accuracy, precision, and recall are showed in Table VIII. The overall accuracy for ten websites drops to 67.5% compared to accuracy of five websites. Class 5 (Netflix) still holds the highest precision and recall and class 8 (Craigslist) has the lowest precision and recall.

Table VII: Confusion Matrix of 10 Websites

		Predicted									
		1	2	3	4	5	6	7	8	9	10
Actual	1	4	0	0	1	0	0	0	3	0	0
	2	3	3	0	0	0	1	0	1	0	0
	3	0	1	6	0	0	0	0	1	0	0
	4	1	0	0	4	0	1	0	2	0	0
	5	0	0	0	0	8	0	0	0	0	0
	6	0	0	0	1	0	7	0	0	0	0
	7	0	0	0	0	0	0	8	0	0	0
	8	2	3	0	1	0	0	0	2	0	0
	9	0	0	0	0	0	0	1	0	7	0
	10	0	2	1	0	0	0	0	0	0	5

Table VIII: Precision, Recall, Accuracy of 10 Websites

Label	Precision	Recall	Accuracy
1	40%	50%	
2	33.3%	37.5%	
3	85.71%	75%	
4	57.14%	50%	
5	100%	100%	67.5%
6	77.78%	87.5%	
7	88.89%	100%	
8	22.22%	25%	
9	100%	87.5%	
10	100%	62.5%	

(3) 15 websites. The first 15 websites of the list in Table IV are selected. The total number of training instances is 180 while the testing instance number is 120. The confusion matrix is showed in Table IX. The accuracy, precision, and recall are listed in Table X. The overall accuracy is 61.67% for 15 websites. The precision and recall for class 12 (Paypal) are the highest and class 8 (Craigslist) has the lowest precision and recall.

Table X: Precision, Recall, Accuracy of 15 Websites

Label	Precision	Recall	Accuracy
1	14.29%	12.5%	
2	30%	37.5%	
3	100%	62.5%	
4	41.67%	62.5%	
5	88.89%	100%	
6	100%	75%	
7	77.78%	87.5%	
8	16.67%	12.5%	61.67%
9	100%	75%	
10	80%	50%	
11	72.73%	100%	
12	100%	100%	
13	44.44%	50%	
14	87.5%	87.5%	
15	11.11%	100%	

(4) 20 websites. All of the 20 websites of the list in Table IV are selected. The total number of training instances is 240 while the testing instance number is 160. The confusion matrix is showed in Table XI. The accuracy, precision, and recall are showed in Table XII. The overall accuracy for 20 websites is 55%. Class 12 (Paypal) still holds the highest precision and recall while class 8 (Craigslist) and class 18 (ask) have the lowest precision and recall.

The results for the four experiments show that the accuracy is higher if the number of websites is less and vice versa. Besides, websites like Google, Craigslist and Ask have low precision and recall which means they are difficult to identify. The reason could be that these websites are too simple to have unique features. On the other hand, the precision and recall for websites such as Netflix and Paypal are very high because there are distinguishable features like large images loading activities. Finally, we can conclude that the power estimation based side-channel attack is effective for website fingerprinting.

V. Discussion

In August 2017, Google released the newest Android 8 and the /proc entry is no long available for applications to access. It's a good thing that Google has realized that the /proc entry could lead to potential threat to user privacy and the situation is improved.

However, since Android 7 still holds 31.6% market share and Android 8 only shares 5.7% in the market as of May 15th, 2018 [17] as shown in Figure 7, 94.3% Android systems in the market are still vulnerable to power side-channel attacks by either power estimation or direct power readings.

Table IX: Confusion Matrix of 15 Websites

		Predicted														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Actual	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	5
	2	1	3	0	0	0	0	0	1	0	1	1	0	0	0	1
	3	0	1	5	0	0	0	0	1	0	0	0	0	1	0	0
	4	1	0	0	5	0	0	0	0	0	0	0	0	1	0	1
	5	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	1	0	6	0	0	0	0	1	0	0	0	0
	7	0	0	0	0	0	0	7	0	0	0	0	0	0	1	0
	8	2	3	0	1	0	0	0	1	0	0	0	0	0	0	1
	9	0	0	0	0	1	0	1	0	6	0	0	0	0	0	0
	10	0	1	0	0	0	0	0	0	0	4	0	0	3	0	0
	11	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0
	13	1	2	0	0	0	0	0	0	0	0	1	0	4	0	0
	14	0	0	0	0	0	0	1	0	0	0	0	0	0	7	0
	15	1	0	0	4	0	0	0	2	0	0	0	0	0	0	1

Table XI: Confusion Matrix of 20 Websites

		Predicted																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Actual	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	4	0	0	0	2	0
	2	0	3	0	0	0	0	0	1	0	0	0	0	1	0	2	0	1	0	0	0
	3	0	0	5	1	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
	4	0	0	0	2	0	1	0	1	0	0	0	0	0	0	3	0	0	0	1	0
	5	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	6	0	0	0	0	1	0	0	0	0	0	0	1	0	0
	7	0	0	0	0	0	0	5	0	0	0	0	0	0	3	0	0	0	0	0	0
	8	2	3	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	0
	9	0	0	0	0	1	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0
	10	0	1	1	0	0	0	0	0	0	3	0	0	3	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0
	13	2	1	0	1	0	0	0	0	0	1	1	0	2	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0
	15	1	1	0	0	0	0	0	2	0	0	0	0	0	0	3	0	0	1	0	0
	16	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	5	0	0	0	0
	17	0	0	0	2	0	0	0	2	0	0	0	0	1	0	0	0	2	1	0	0
	18	1	0	0	3	0	0	0	0	0	1	0	0	0	0	0	0	1	0	2	0
	19	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	7	0
	20	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	6

Table XII: Precision, Recall, Accuracy of 20 Websites

Label	Precision	Recall	Accuracy
1	0%	0%	55%
2	30%	37.5%	
3	71.43%	62.5%	
4	22.22%	25%	
5	88.89%	100%	
6	75%	75%	
7	100%	62.5%	
8	0%	0%	
9	70%	87.5%	
10	50%	37.5%	
11	88.89%	100%	
12	100%	100%	
13	28.57%	25%	
14	72.72%	100%	
15	21.43%	37.5%	
16	100%	62.5%	
17	33.3%	25%	
18	0%	0%	
19	53.85%	87.5%	
20	100%	75%	

Besides, the /proc/stat file is available from the command-line window and the CPU frequency files are also available on Android 8. Since the /proc/stat file can be read via a terminal, the system is still vulnerable to power side-channel attacks by power estimation if shell code injection can be performed.

In the future, the following work could be further explored: (1) test our hypothesis that the modeled or estimated power data can also be used for performing other types of inference attacks such as running app inference and user location inference, and (2) investigate the feasibility of power side-channel attacks on new versions of Android systems such as Android 8.

VI. Conclusion

In this paper, we first presented a review of power data acquisition methods on Android systems. Next, we proposed a power estimation model to perform power side-channel attacks on Android 7 even though the traditional methods of

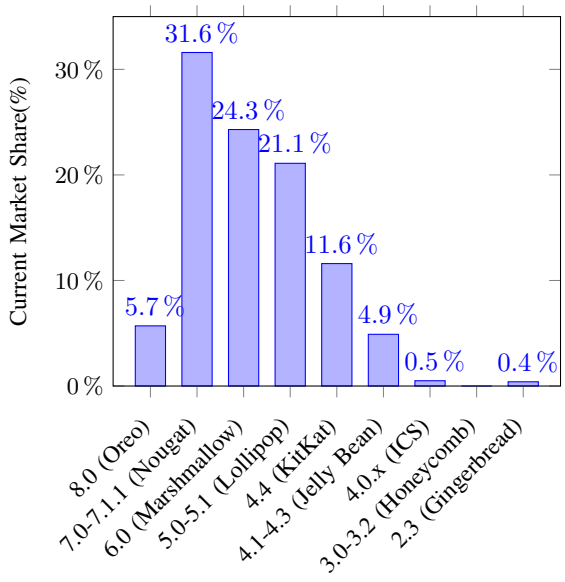

Figure 7: Most commonly used Android versions in the market as of May 15th, 2018[17]

requesting power data don't work anymore. This model is simple, feasible, and accurate for attackers to use. Then, we proposed a power estimation based website fingerprinting method. The results of our website fingerprinting attacks demonstrate that our power estimation based approach is effective for inferring webpages. It is demonstrated that the webpage identification accuracy is higher if the number of websites in the dataset is less. Besides, simple websites such as Google, Craigslist, and Ask are difficult to identify because they have fewer distinguishable features while websites like Netflix and Paypal can be identified with high precision and recall due to their unique features such as loading large images. In the future, we plan to infer other types of sensitive user information such as running apps and location using our power estimation model. Moreover, although the /proc access is restricted on Android 8, the feasibility of performing power estimation based side-channel attacks on it can still be explored since the /proc/stat file can still be read from a terminal.

ACKNOWLEDGMENT

This research was supported in part by the NSF grant DGE-1619841 and NSA grant H98230-17-1-0403.

REFERENCES

[1] Shane S. Clark, Hossen Mustafa, Benjamin Ransford, Jacob Sorber, Kevin Fu, Wenyuan Xu, *Current Events: Identifying Webpages by Tapping the Electrical Outlet*, in Proceedings of the European Symposium on Research in Computer Security, 2013.

[2] Qing Yang, Paolo Gasti, Gang Zhou, Aydin Farajidavar, Kiran S. Balagani, *On Inferring Browsing Activity on Smartphones via USB Power Analysis Side-channel*, IEEE Transactions on Information Forensics and Security, 2016.

[3] Lin Yan, Yao Guo, Xiangqun Chen, Hong Mei, *A Study on Power Side Channels on Mobile Devices*, in Proceedings of the 7th Asia-Pacific Symposium on Internetware, 2015.

[4] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, *PowerSpy: Location Tracking using Mobile Device Power Analysis*, in Proceedings of the USENIX Security Symposium, 2015.

[5] *https://source.android.com/devices/tech/power/device.html*

[6] *https://developer.android.com/reference/android/os/-BatteryManager.html*

[7] Yimin Chen, Xiaocong Jin, Jingchao Sun, Rui Zhang, and Yanchao Zhang. *POWERFUL: Mobile app fingerprinting via power analysis*, in Proceedings of the IEEE INFOCOM, 2017.

[8] Jingyao Fan, Qinghua Li, Guohong Cao. *Privacy Disclosure Through Smart Meters: Reactive Power Based Attack and Defense*, in Proceedings of the Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2017.

[9] Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah Ab Hamid, Mohammad Shojafar, Abdelmuttlib Ibrahim Abdalla Ahmed, Sajjad A. Madani, Kashif Saleem, Joel J.P.C. Rodrigues. *A survey on energy estimation and power modeling schemes for smartphone applications*, International Journal of Communication Systems, 2016.

[10] Yi Cao, Javad Nejati, Muhammad Wajahat, Aruna Balasubramanian, Anshul Gandhi. *Deconstructing the Energy Consumption of the Mobile Page Load*, in Proceedings of the ACM Sigmetrics, 2017.

[11] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Z. Morley Mao, Lei Yang, *Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones*, in Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010

[12] Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulkoo Kang, Hojung Cha. *AppScope: Application Energy Metering Framework for Android Smartphones using Kernel Activity Monitoring*, in Proceedings of the USENIX Annual Technical Conference, 2012.

[13] Aaron Carroll, Gernot Heiser, *An Analysis of Power Consumption in a Smartphone*, in Proceedings of the USENIX Annual Technical Conference, 2010.

[14] *http://man7.org/linux/man-pages/man5/proc.5.html*

[15] *https://developer.qualcomm.com/forum/qdevnet-forums/performance-tools/trepn-profiler/26812*

[16] *https://www.csie.ntu.edu.tw/ cjlin/libsvm/index.html*

[17] *https://www.appbrain.com/stats/top-android-sdk-versions*