MagAttack: Guessing Application Launching and Operation via Smartphone

Yushi Cheng
[1]Zhejiang University
[2]Alibaba-Zhejiang University Joint Institute of Frontier Technologies
yushicheng@zju.edu.cn

Xiaoyu Ji*
[1]Zhejiang University
[2]Alibaba-Zhejiang University Joint Institute of Frontier Technologies
xji@zju.edu.cn

Wenyuan Xu
Zhejiang University
wyxu@zju.edu.cn

Hao Pan
Shanghai Jiao Tong University
panh09@sjtu.edu.cn

Zhuangdi Zhu
Michigan State University
zhuzhuan@msu.edu

Chuang-Wen You
National Taiwan University
cwyou@ntu.edu.tw

Yi-Chao Chen
University of Texas at Austin
yichao@cs.utexas.edu

Lili Qiu
University of Texas at Austin
lili@cs.utexas.edu

ABSTRACT

Mobile devices have emerged as the most popular platforms to access information. However, they have also become a major concern of privacy violation and previous researches have demonstrated various approaches to infer user privacy based on mobile devices. In this paper, we study a new side channel of a laptop that could be harvested by a commercial-off-the-shelf (COTS) mobile device, *e.g.*, a smartphone. We propose `MagAttack`, which exploits the electromagnetic (EM) side channel of a laptop to infer user activities, *i.e.*, application launching and application operation. The key insight of `MagAttack` is that applications are discrepant in essence due to the different compositions of instructions, which can be reflected on the CPU power consumption, and thus the corresponding EM emissions. `MagAttack` is challenging since that EM signals are noisy due to the dynamics of applications and the limited sampling rate of the built-in magnetometers in COTS mobile devices. We overcome these challenges and convert noisy coarse-grained EM signals to robust fine-grained features. We implement `MagAttack` on both an iOS and an Android smartphone without any hardware modification, and evaluate its performance with 13 popular applications and 50 top websites in China. The results demonstrate that `MagAttack` can recognize aforementioned 13 applications with an average accuracy of 98.6%, and figure out the visiting operation among 50 websites with an average accuracy of 84.7%.

*Corresponding faculty author.

AsiaCCS '19, July 9–12, 2019, Auckland, New Zealand
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329817

CCS CONCEPTS

• **Security and privacy** → **Side-channel analysis and countermeasures**.

KEYWORDS

Side Channel Attack; Electromagnetic Emission; User Privacy; Commodity Mobile Device.

ACM Reference Format:
Yushi Cheng, Xiaoyu Ji, Wenyuan Xu, Hao Pan, Zhuangdi Zhu, Chuang-Wen You, Yi-Chao Chen, and Lili Qiu. 2019. MagAttack: Guessing Application Launching and Operation via Smartphone. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3321705.3329817

1 INTRODUCTION

Mobile devices have emerged as the most popular platforms to assist daily activities and exchange information over the Internet. According to Gartner [11], there will be more than 11 billion phones, tablets and laptops by the end of 2018. Along with the rapid growth are the privacy concerns. The proliferation of mobile devices has been a major concern in the security and privacy communities. Various side channels have been utilized for electrical-appliance usage analysis [7, 16, 26], decryption of cryptographic computation [12, 13, 18], and human-device activity recognition [5, 8, 32, 33]. These side-channel attacks, especially the last ones, draw increasing attention due to the widespread use of mobile devices and the increasingly intensive interaction between human and smart devices.

Prior researches [5, 8, 17, 32, 33] have shown several side-channel attacks that can sense human-device activities. Zhuang *et al.* and Zhu *et al.* utilize acoustic emanations to infer user keystrokes [32, 33]. Cai *et al.* use motion sensors to infer user tapping and gesture inputs on smartphones [5]. Clark *et al.* use the AC power consumption to recognize the web page that a user browses on a laptop [8]. However, Clark's scheme requires modification of the power outlet to measure the AC power consumptions and can only work when

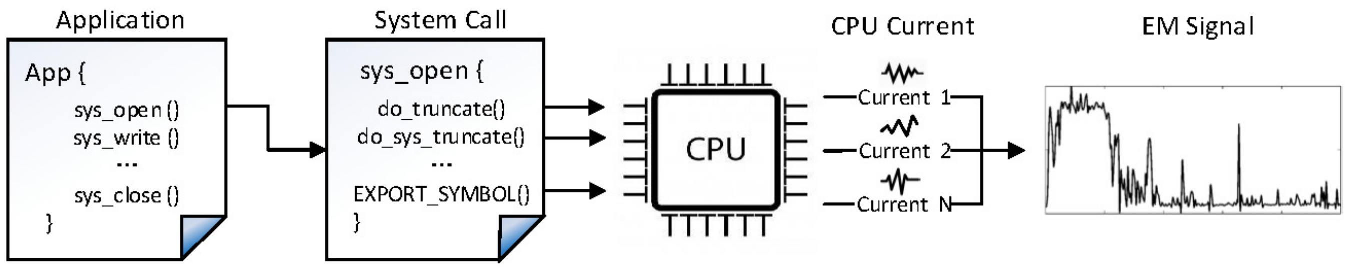


Figure 1: The cause of EM emission from the laptop's CPU. An application consists of various system calls, which are composed of different instruction sequences and those instructions generate corresponding currents, and finally EM signals.

the laptop is being charged. Jana *et al.* sense user applications based on the footprints of applications on memory usage [17], which is intrusive since they need to log into the system and run a background process in parallel with the target application.

In this paper, we investigate a new EM-based side-channel attack for user activity inference. We propose `MagAttack`, which detects and recognizes user activities by tracking the EM emissions from the laptop's CPU. Compared with existing side-channel attacks, `MagAttack` is non-intrusive and can be implemented on the COTS mobile devices without any hardware modification. The underlying principle of `MagAttack` is that, for an application, each time when being launched, a fairly unique and consistent sequence of CPU instructions are executed, as shown in Fig. 1. When a CPU executes different instructions, it emits various EM signals accordingly, which can be further captured by the built-in magnetometer in a mobile device for privacy inference. `MagAttack` recognizes two levels of user activities: (1) which application is being launched, *i.e.*, application recognition, and (2) what a user is doing with the application, *i.e.*, operation recognition. For example, we can figure out that a user is launching a web browser and recognize which web page the user is visiting. The significance of `MagAttack` lies in that it infers the basic operations of laptops and thus can be the prerequisite of many other user privacy violation attacks, *e.g.*, when inferring user passwords through keystrokes, `MagAttack` can be employed first to detect the launching of finical applications such as PayPal. Besides, with the help of `MagAttack`, an adversary can learn a user's interests and habits by continuously tracking the application usage of the user.

Inferring laptop user activity via the EM side channel is promising yet challenging. First, both the location and orientation of a smartphone affect the captured EM signals. The former decides the initial EM amplitude as a result of the earth's magnetic field, and the latter determines the changing trend of the EM amplitude. Second, the magnetic sensors in COTS mobile devices such as smartphones usually have low sampling rates. Different from the sensors used in [12, 13, 25, 26] with working frequency ranging from MHz to GHz, a magnetometer in mobile devices can only measure frequency up to 100 Hz. In other words, information from high frequency signals is lost. Third, EM signals might be inconsistent when the application changes, *e.g.*, a website is updated. Actually, many websites change their contents and components such as pop-out advertisements on a daily basis, and the captured EM signals are different as a result.

To address aforementioned challenges, we first propose an earth impact reduction scheme to eliminate the influence from the earth's magnetic field and the ambient noise. Then, for application launching detection, we design a sliding-window based pre-screening algorithm for preliminary detection, and a fine-grained Support Vector Machine (SVM) classifier to further ensure high accuracy. For application recognition, we employ Short Time Fourier Transform (STFT) and Principal Component Analysis (PCA) for feature extraction, and design a 1-Nearest Neighbor (1NN) classifier to achieve accurate recognition. For operation recognition, we use Wavelet Multi-Resolution Analysis algorithm (MRA) to process the inconsistent EM signals and build a weighted 1NN classifier to deal with their dynamics.

Application Scenario: we envision that `MagAttack` can be used in public areas where an adversary sits near a victim and attempts to infer the victim's laptop activities for habit tracking or further privacy violation attacks such as password inference. To the best of our knowledge, this is the first side-channel attack to infer laptop user activities by the means of CPU EM emissions. In summary, our contribution includes below.

- We analyze the underlying correlation between the applications and the corresponding CPU EM emissions. We propose to use a mobile device to infer user activities on a laptop by tracking the EM emissions from the laptop's CPU.
- We investigate the distinctiveness of EM emissions caused by various user activities, and elaborately design `MagAttack` to differentiate them reliably.
- We implement `MagAttack` on commercial smartphones without any hardware modification, and evaluate it with 13 popular applications and 50 top websites in China. The results demonstrate that `MagAttack` can detect application launching with a precision of 97.0% and a recall of 92.1%, recognize the 13 applications with an average classification accuracy of 98.6%, and classify the 50 websites with an average accuracy of 84.7%.

2 MEASURING EM EMISSION TO INFER APPLICATION

In this section, we first introduce the built-in magnetometer on mobile devices, and then show the feasibility of `MagAttack`.

Table 1: Top 10 system calls invoked when different applications being launched.

Safari	Chrome	iTunes
workq_kernreturn (17.6%)	kevent (24.0%)	workq_kernreturn (13.9%)
bsdthread_ctl (13.0%)	write (12.4%)	geteuid (9.5%)
stat64 (8.4%)	workq_kernreturn (6.6%)	stat64 (8.2%)
pread (8.2%)	read (6.1%)	bsdthread_ctl (7.9%)
madvise (6.5%)	recvmsg (5.8%)	getdirentires64 (4.9%)
openat (3.5%)	stat64 (5.0%)	getattrlist (4.7%)
(3.0%)	bsdthread_ (4.3%)	madvise (4.6%)
kevent_qos (2.9%)	psynch_mutexdrop (2.7%)	read (3.5%)
mmap (2.6%)	psynch_ (2.4%)	open_nocancel (3.2%)
geteuid (2.5%)	mmap (2.0%)	kevent_qos (2.9%)

2.1 Magnetometer on Mobile Device

A magnetometer is an instrument that can measure the direction, strength, and relative change of a magnetic field at a particular location. The built-in magnetometer on mobile devices is usually a Hall Effect sensor, which is small, cheap and low in sensitivity ($<= 5mV/mT$). The sampling rate of the built-in magnetometer is configurable, which usually varies from 4 Hz to 100 Hz for smartphones. Due to its low cost and extensive functions, *e.g.*, employed with gyroscopes and accelerometers for motion tracking, the magnetometer is widely equipped on COTS mobile devices, and thus can be a good alternative for EM measuring.

2.2 Application Launching

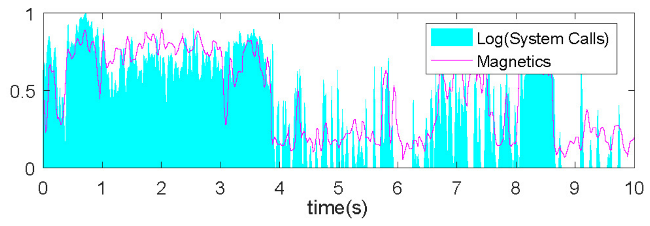
During application launching, the Launch Services framework provides primary support. It sends a message to `WindowServer`, which in turn calls `fork()` and `execve()` (both are system calls) to run the requested application [24]. The requested application then, runs user functions in the user space and invokes system calls to interact with the kernel and access the hardware. Thus, an application executes both user functions and system calls when being launched.

Upon user activity inference, we focus on the system calls invoked during application launching. As shown in Fig. 1, an application consists of a series of system calls. Our hypothesis is that different system calls are invoked at different frequencies for various applications. To validate it, we use a system trouble-shooting tool *dtrace*[10] (available on Linux, Mac OS, and Windows [14]) to capture the name and time of the executed system calls when various applications (Safari, Chrome, and iTunes) are being launched on the same laptop (a MacBook Air). The results in Tab. 1 demonstrate that the most intensive system call for Safari is `workq_kernreturn()`, which accounts for 17.6%. While for Chrome, it is `kevent()` that holds the largest portion of 24.0%. For iTunes, `workq_kernreturn()` appears as the most intensive system call as well but with a different proportion of 13.9%. It confirms our hypothesis that the system calls invoked when launching different applications, are discrepant in both type and frequency, even for applications of the same type.

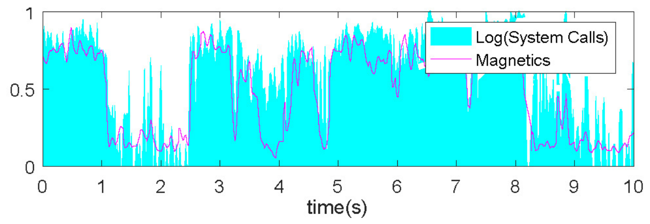
2.3 From System Call to EM Emission

A system call is a wrapper function that consists of a sequence of instructions [15], as revealed in Fig. 1. As a result, various system calls are composed of different instruction sets and thus generate distinct CPU power consumptions.

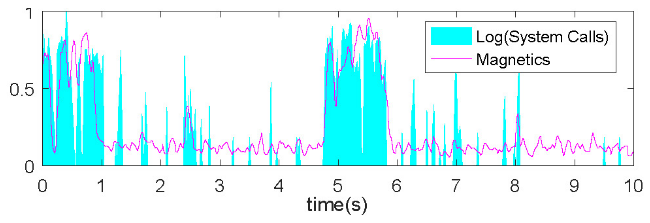
A CPU chip consists of a large number of CMOS (Complementary Metal Oxide Semiconductor) [27] transistors arranged in a lattice form, which perform basic arithmetic, logical, control and



(a) Safari



(b) Chrome



(c) iTunes

Figure 2: EM signals are correlated with the system call traces but distinct among applications. Note that the two traces are normalized for illustration and the Y-axis represents the frequency of system calls or the magnitude of EM signals.

input/output operations specified by the instructions. Energy consumption of the CPU heavily depends on the power dissipation of the CMOS lattice. Average CPU power consumption [23] P_{avg} can be calculated as follows:

$$P_{avg} = \frac{CV(\alpha)^2 AF(\alpha)}{2} \tag{1}$$

where C represents the CMOS capacitance and is a function of the transistor size and the wire length. V is the supply voltage to CPU. A is the average switching frequency of the CMOS transistors and F is the clock frequency. V and F are further related to the CPU load α. When executing various instructions, the CPU involves different numbers of CMOS transistors and generate different loads, resulting in distinct power consumptions as well as CPU currents. Since various system calls are composed of different instruction sets, the CPU currents for those system calls are likely to be diverse, which contribute to distinct EM signals.

2.4 Feasibility of `MagAttack`

As various applications invoke different system calls when being launched, the emitted EM signals shall correlate with the system calls executed by the CPU but remain distinct among applications. To validate our hypothesis, we capture the EM signals with an iPhone SE smartphone when different user applications (Safari,

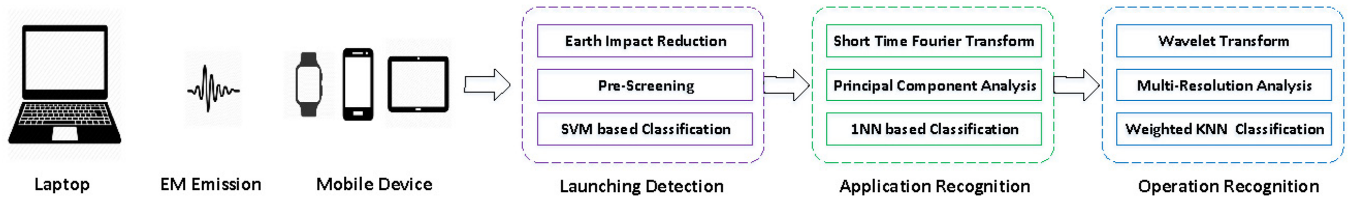


Figure 3: Workflow of `MagAttack`, which detects application launching, recognizes user activities based on EM signals collected by mobile devices in the target laptop's vicinity.

Chrome and iTunes) are launched on a MacBook Air laptop. During experiments, the target laptop is placed on a round table with the attack smartphone attached on the backside to draw no attention, as shown in Fig. 7. The detailed information of the smartphone and the laptop is depicted in Tab. 2 and Tab. 3 in Sec. 5. Meanwhile, we use *dtrace* to capture the name of the executed system calls and the time they are being called in microsecond granularity.

Since the system call is recorded in microseconds while the EM signal is recorded in 10-millisecond granularity (the sampling rate of the iPhone SE magnetometer is 100 Hz), we transform the system call trace into a time-versus-number histogram. The transformed trace is a two-column matrix $E = [\vec{t}; \vec{n}]$, where vector $\vec{t}$ records time units in 10-millisecond granularity, and vector $\vec{n}$ records the number of system calls during that time unit. Then, we align the magnetic trace with the system call trace by shifting the former one so that the two traces have maximal correlation coefficient. We plot the logarithmic system call traces and the EM signals in Fig. 2 (both are normalized for illustration), from which we can observe that:

- EM signals show strong resemblance to the system call traces captured at the same time, *e.g.*, the system-call-intensive moment also has a high EM magnitude.
- EM signals demonstrate distinct patterns among applications, even for those of the same type: Safari and Chrome, *e.g.*, the EM signal of Chrome is more dynamic and has more peaks compared with that of Safari.

These findings shed light upon inferring user activities on laptops via EM signals captured by nearby mobile devices. Since various user activities invoke different system calls, the resulting CPU power consumptions cause varying EM signals, which are distinct and associated with the activities and thus in turn can be utilized to conduct user activity inference.

3 THREAT MODEL

In this section, we present the threat model of `MagAttack`. Since the adversary's goal is to infer user activities on user's laptop without his awareness, we consider the following attack scenario: *in a public area such as a library, a target is using his laptop. The adversary sits near to him, and tries to figure out what the target is doing on the laptop (*e.g.*, what applications the target launches and what operations the target performs).* In such a scenario, we assume that the adversary has following abilities.

Vicinity to Target Laptop. We assume the adversary's mobile devices can be in the target laptop's vicinity, and draw no attention.

No Target Laptop Access. We assume that an adversary may target at any users of her choices, but she has no direct access to the target laptop. She cannot physically touch/see the screen, or install malware.

Algorithm 1: Earth Impact Reduction

```
Input: $mag = \{mag_x(t), mag_y(t), mag_z(t)\}$, $t = 1 \dots n$:
        three-dimensional signals
Output:
    • $M = M(t)$, $t = 1 \dots n$ : aggregated signals.
    • $M_{norm} = M_{norm}(t)$, $t = 1 \dots n$: aggregated and
      normalized signals.
1 $M = mag$
2 for $i \in \{x, y, z\}$ do
3     $M_i = M_i - avg(M_i)$ // centralization
4 for $t \in [1, 2, \dots, n]$ do
5     $M(t) = \sqrt{M_x(t)^2 + M_y(t)^2 + M_z(t)^2}$ // aggregation
6 $M_{norm} = \frac{M - min(M)}{max(M) - min(M)}$ // normalization
```

No User Interaction. The adversary cannot ask users to perform any operations, such as pressing a button or running a specific application.

4 MAGATTACK DESIGN

4.1 Overview

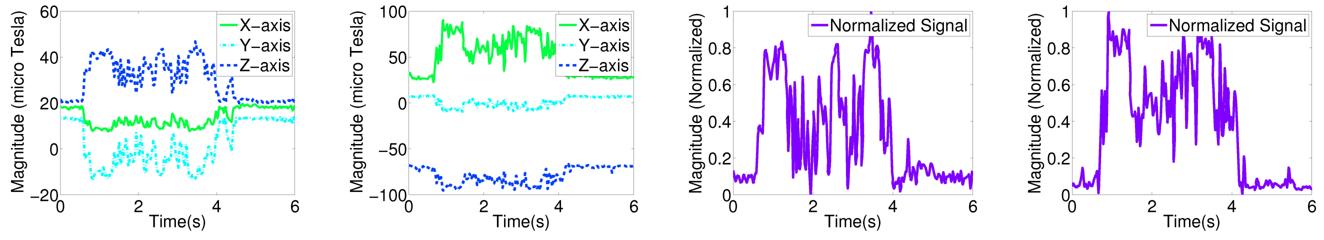
To infer user activities, the adversary first puts her mobile device in the target laptop's vicinity and draws no attention. Then, the attack device collects the electromagnetic emissions from the laptop's CPU, based on which `MagAttack` detects application launching, recognizes running applications, and figures out user operations, as shown in Fig. 3.

4.2 Launching Detection

In this subsection, we elaborate how to detect the launching process of an application, as the first step of user application redefinition.

4.2.1 Earth Impact Reduction. Due to the impact of the earth's magnetic field, the captured EM signals are geo-spatial dependent. Even for an application launched on the same laptop but with different geo-spatial locations or laptop-smartphone orientations, the EM signals can vary a lot. As shown in Fig. 4(a) and 4(b), the 3-dimensional EM signals at two different locations/orientations differ in values on the axis of x, y, and z. `MagAttack` shall eliminate the earth impact to achieve location/orientation-free attack.

The mobile device sensor records surrounding EM signals in three dimensions (x, y, z). The initial EM magnitude of each axis depends on the location of the mobile device, and the changing



(a) 3-dimensional EM signals (setting 1). (b) 3-dimensional EM signals (setting 2). (c) 1-dimensional normalized signals (setting 1). (d) 1-dimensional normalized signals (setting 2).

Figure 4: Before vs. after earth impact reduction. Setting 1 and 2 refer to two different geo-spatial locations and laptop-smartphone orientations for launching the same application.

trend lies with the laptop-smartphone orientation. To eliminate the impact of location and orientation, we utilize the relative change of the EM signals instead of the original data. We assume this change is caused by the launching and thus is consistent with the same application regardless of positions/orientations.

To achieve it, we first centralize the magnetic magnitude of each axis to resolve its relative change to the earth's magnetic field, and then aggregate and normalize the relative changes in all three axes, as shown in Algorithm 1. As a result, we can see from Fig. 4 that after earth impact reduction, the 1-dimensional normalized signals under two different settings become more similar and can be further identified as the same one, as discussed later.

4.2.2 Pre-screening. EM signals usually remain stable when no application is started but can vary significantly during the process of application launching. To improve the accuracy and efficiency of launching detection, we design a pre-screening algorithm that uses a time window to scan through the EM signals and filter out the time windows that are unlikely to contain the start of an application.

As an application can start at any time, we detect the high variances of the EM signals over a certain time period. A smaller time window and moving step can achieve higher accuracy at the cost of lower detection efficiency. To strike the balance between accuracy and efficiency, we set the time window to be 1 s and the moving step to be 0.1 s. In addition, we utilize the Exponential Moving Average (EMA) approach [20] to update the variance threshold during the period without application launching:

$$\delta_{t+1} = (1-\alpha)\delta_t + \alpha \times Var(t) \tag{2}$$

where δ_t and $Var(t)$ are the threshold and the EM variance at the time period t, respectively. α represents the degree of weighting decrease and a larger α indicates a more dominant current variance in updating the threshold. In our implementation, we set α to be 0.1. A window W_t is detected when its EM signal variance is substantially larger than the threshold δ_t:

$$Var(t) \geq \beta \times \delta_t \tag{3}$$

where β is the coefficient of the threshold. A larger β indicates that fewer sliding windows will be detected. In our implementation, we set β to be 3. Upon detecting a sliding window with a large variance, we can further classify it with a Support Vector Machine (SVM) based classifier.

4.2.3 Launching Detection. In addition to application launching, other user operations on the laptop may also contribute to high variances of EM signals. To address it, we utilize a SVM based classifier to further refine the launching detection results.

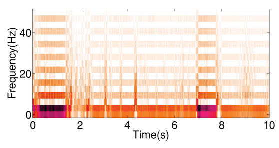
As the CPU instructions involved with a launching operation often take seconds to complete, a 1-second EM trace may not hold enough features to differentiate launching from other operations. Therefore, for each 1-second EM trace detected by the pre-screening algorithm, we append it with the subsequent $k-1$ one-second EM traces. In our implementation, we choose k to be 4 based on our observation that the variance of EM signals becomes indistinctive after 4 seconds since we start the application. For each k-second normalized EM time series, we smooth it with the Wavelet reconstruction at level 4, and then employ Short Time Fourier Transform and Principal Component Analysis to extract a feature vector. Since we use the same feature extraction techniques for launching detection and application recognition, we defer to present the technical details of feature extraction in the next subsection. Then, we feed the feature vector of each k-second EM trace to a SVM based binary classifier with a kernel type of the radial basis function [6], whose output is whether it is the start of an application. Combined with the pre-screening, `MagAttack` is able to detect application launching accurately and reliably.

4.3 Application Recognition

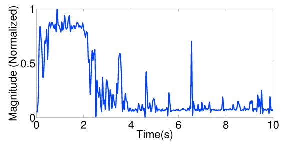
After detecting the launching of an application, we aim at figuring out what the application is.

4.3.1 Data Pre-processing. After launching detection, we obtain a number of k-second time windows that contain the EM traces of application launching. Hereafter, we use *time window/interval* to represent the EM trace contained in that time window/interval for short. For these time windows, we perform two pre-processing operations: *window expansion* and *window alignment* before feature extraction.

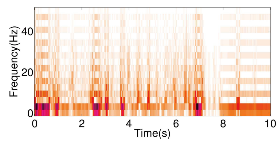
Window Expansion. To guarantee that the selected time window contains sufficient information, even for applications that need a long time to initialize (e.g., more than 10 seconds), we append each aforementioned k-second time window with the subsequent $m-k$ one-second time windows, where $m \geq k$. In our implementation, we choose m to be 10. This expansion can help `MagAttack` achieve higher accuracy by including more features during the launching process.



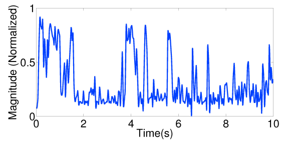
(a) Spectrogram: Microsoft PowerPoint



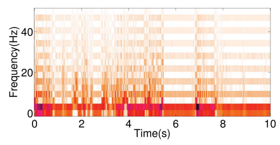
(b) EM Signal: Microsoft PowerPoint



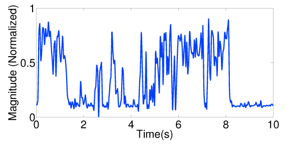
(c) Spectrogram: Skype



(d) EM Signal: Skype



(e) Spectrogram: Mail



(f) EM Signal: Mail

Figure 5: STFT spectrum of EM signals reconstructed on the first wavelet level for different applications.

Window Alignment. Due to the finite granularity of the window sliding approach, the starting time of each window deviates more or less from the ground truth. To reduce the impact of deviations, we align these time windows in two steps. First, we compute the centroid of these time windows using the average Dynamic Time Warping (DTW) scheme [21], which is the time window that has the minimum averaged DTW cost to the others. Then, we use the centroid as the base to shift each other time window in the time series. As a result, each shifted time window has the maximum correlation coefficient with the base signal and is still m seconds.

4.3.2 Feature Extraction. We then extract a feature vector for each aligned time window. We first divide a EM time window into overlapped time intervals and conduct Fast Fourier Transform (FFT) on each interval, which extracts time-variant features in the frequency domain. Then, we conduct the Principal Component Analysis (PCA) [29] on the FFT result of each time interval and obtain the first PCA component, which aggregates features in different frequency scales. Finally, we sequentialize the PCA component of each time interval to construct a feature vector for application recognition.

Short Time Fourier Transform. For each aligned time window, we divide it into time intervals using a sliding window with an interval size of w and a step size of $0.5 * w$. Then, each time interval is zero padded to the length of $2 * w$, before conducting FFT to get the STFT spectrogram. We calculate the abstract value of the FFT results and obtain the first half. As thus, for each time window, we get a $t \times l$ spectrogram matrix S, where t rows correspond to t time intervals, and l columns are the FFT results of that time interval. In practice, we set $w = 320$ milliseconds. Fig. 5 illustrates the STFT spectrograms of 3 Mac OS applications (Microsoft PowerPoint, Skype and Mail), where the X-axis represents the time interval, the Y-axis represents the frequency, and the color represents the energy of the frequency.

Principal Component Analysis. We then use the PCA to track the correlation of FFT results among different frequencies, and combine them by extracting the first principal component. We conduct PCA on the FFT results of each time interval in three steps: data preparation, coefficient calculation, and feature vector construction.

(1) Data Preparation. Let s be the number of time windows. With STFT, each time window is transformed into a spectrogram matrix with t rows. For each time interval, we extract its FFT results from aforementioned spectrogram matrices to construct a new interval matrix. In this way, we build t interval matrices $H_1, H_2, \ldots, H_t$, and each matrix has s rows.

(2) Coefficient Calculation. For each interval matrix H_i, we calculate its principal component coefficient matrix C_i. Each column of C_i contains coefficients for one principal component and the columns are arranged in the decreasing order of component variance. We then obtain the first principal component of H_i, *i.e.*, the first column of C_i, for feature vector construction.

(3) Feature Vector Construction. We conduct PCA on the spectrogram matrix S to build feature vector V. The i^{th} element of V is calculated as:

$$V(i) = \sum_{j=1}^{l} S(i,j) * C_i(1,j) \tag{4}$$

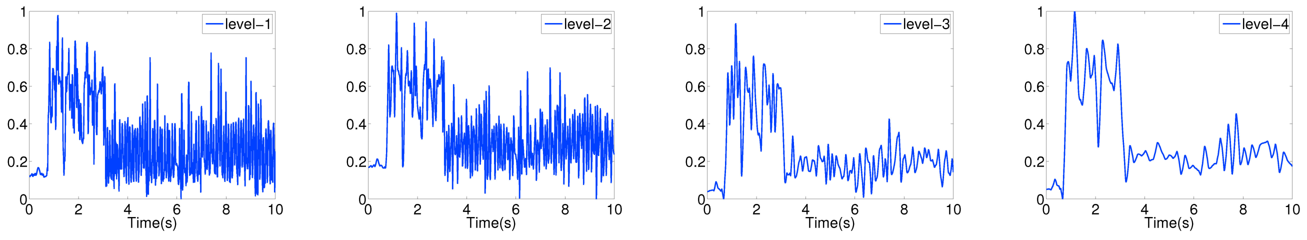
where l is the column number of the spectrogram matrix S as well as the length of FFT results for each time interval.

In this way, for each m-second time window, we extract a feature vector V with t elements, where t is the number of time intervals that the time window is divided into. We envision that this feature vector retains the time-varying frequency features of the EM signals.

4.3.3 Application Classification. Given the feature vectors extracted from the training data, we build an application recognition classifier using the 1-Nearest Neighbor (1NN) algorithm [28], which is lightweight and non-parametric. We employ the Euclidean distance to evaluate the distance between two feature vectors, *i.e.*, two samples. When inferring, the 1NN algorithm classifies the testing feature vector into the closest training vector. Since we have aligned each sample before feature extraction, the Euclidean distance works well in our algorithm.

4.4 Operation Recognition

In addition to application recognition, `MagAttack` attempts a more fine-grained detection, *i.e.*, operation recognition. We analyze the web browser as an example and regard visiting different web pages as different operations. However, our method is not limited to it and can be applied to other applications as well. We capture the EM signals when a web page is being launched, and extract a feature matrix from its EM signals. Specifically, we use Wavelet Multi-Resolution Analysis (MRA) [1] to get the de-noised signals at N Wavelet levels. Then, we extract a feature vector from each of the N reconstructed signals using the same approach in Sec. 4.3.2, and arrange the N feature vectors in rows to construct a feature matrix. With the obtained feature matrix, we use a variant of 1NN classifier to achieve operation recognition.



(a) Level-1 reconstruction. (b) Level-2 reconstruction. (c) Level-3 reconstruction. (d) Level-4 reconstruction.

Figure 6: Level 1-4 reconstructed EM signals using the Wavelet MRA.

4.4.1 *Wavelet Multi-Resolution Analysis.*

Opening a particular web page requires executing network-related CPU instructions over a short time interval, resulting in EM signals with time-varying frequency characteristics. EM signals generated by opening a web page are usually more inconsistent than those generated by application launching. The reason is that for applications, CPU instructions executed by different launchings are fairly consistent while for web pages, they are likely to be various as a result of dynamic contents, *e.g.*, pop-out online advertisements.

To address it, we use the Wavelet MRA to de-noise the EM signals of web pages at different granularity scales before extracting time-frequency features. The insight is that, although each time opening a web page may involve dynamic contents and thus different CPU instructions, the EM signals of the same web page are similar at a coarser granularity scales with subtle differences at a finer granularity. We analyze EM signals of different web pages at N granularity scales with N different weights, and N is set to be 5 based on the Shannon entropy-theory [9]. In the following, we elaborate the details of the employed MRA approach.

Wavelet Decomposition. First, we decompose the EM signals generated by web page opening with two complementary filters: a low-pass filter, which generates approximation coefficients, and a high-pass filter, which generates detail coefficients. For an EM time series, we decompose them iteratively from level 1 to level N, and get a coefficient vector D_n as:

$$D_n = (a_n, d_n, d_{n-1}, d_{n-2}, \dots, d_1) \tag{5}$$

where a_n contains the approximation coefficients at level n and d_n contains the detail coefficients at level n, with $1 \le n \le N$.

Wavelet Reconstruction. Then, we reconstruct an approximation signal from level 1 to level N, respectively. For each level n, we calculate the reconstructed signal by up-sampling and convolution from level 1 to level n with the approximation coefficient a_n. An illustration is provided in Fig. 6, which shows an increasing order of denoising from level 1 to level 4. After denoising, we extract a feature vector for each of the N reconstructed approximation signals using the same approach in Section 4.3.2. Then, the N feature vectors are combined in rows to form a feature matrix for classification.

4.4.2 *Weighted 1NN.*

Similar to application classification, we use a 1NN based classifier. The difference lies in that, the feature matrix of a web page browsing has N rows, with each row representing a feature vector of a wavelet reconstruction level. These feature vectors shall have different influence on the distance measurement.

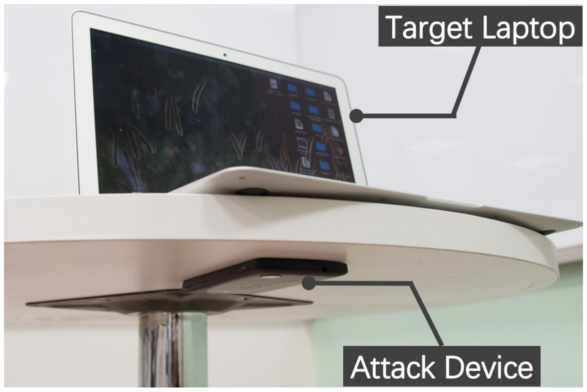


Figure 7: The experimental scenario of `MagAttack`. We keep the attack device under the table to draw no attention. The round table is 2.5 cm in thickness.

Specifically, the feature vector of level 1 represents the finest features, and thus should be assigned with the highest weight. On the contrary, the feature vector of level N represents the coarsest features, and thus should be assigned with the lowest weight. As a result, we use a weighted 1NN algorithm to calculate the distance between two feature matrices of web page browsing. We define the distance between two feature matrices S_p and S_q as the weighted sum of their Euclidean distances in each row vector as follows:

$$Dist(p,q) = \sum_{n=1}^{N} w_n . \|S_p - S_q\|_n \tag{6}$$

$$\|S_p - S_q\|_n = \sqrt{\sum_{i=1}^{w} (S_p(n,i) - S_q(n,i))^2} \tag{7}$$

where w_n is the weight assigned to the distance of the n^{th} row vector, based on the coarseness of its corresponding wavelet reconstruction level. In general, the weight of the n^{th} row distance is assigned as:

$$w_n = 2^{N-n} \tag{8}$$

As thus, the 1^{th} row vector is assigned with the highest weight w_1 of $2^{(N-1)}$, and the N^{th} row vector is assigned with the lowest weight w_N of 2^0.

5 PERFORMANCE EVALUATION

To evaluate the performance of `MagAttack`, we have conducted experiments with 13 popular applications and 50 top websites in

Table 2: Summary of experimental laptops.

Machine Type	MacBook Air	MacBook Pro	Lenovo T440p
OS Version	Mac OS 10.10.5	Mac OS 10.11.1	Win 7 Home
Processor	Intel Core i5 1.4 GHz	Intel Core i5 2.7 GHz	Intel Core i5 2.6 GHz
Memory	4-GB DDR3 1600 MHz	8-GB DDR3 1867 MHz	8-GB DDR3L 1600 MHz

China across 30 days. In summary, the performance of MagAttack is:

- MagAttack achieves a precision of 97.0% and a recall of 92.1% in application launching detection, an average accuracy of 98.4% to recognize 13 applications, and an average accuracy of 84.7% to classify the 50 top websites in China.
- MagAttack can operate with little influence from data freshness, hardware platform, operating system, sensor model and sampling rate.

5.1 Experiment Setup

We conduct MagAttack in a lab with 3 laptops and 2 smartphones. The detailed settings are as follows.

Target Device. We use a MacBook Air laptop as the main target device. In addition, we use a MacBook Pro and a Lenovo T440p laptop to evaluate the performance of MagAttack on various operating systems. The detailed information of each target device is shown in Tab. 2.

Attack Device. We use an iPhone SE smartphone as the main attack device to capture the laptop EM emissions. In addition, we use a Nexus 5 smartphone to evaluate the performance of MagAttack with various attack devices. The detailed information of each attack device is shown in Tab. 3.

Attack Scenario. We utilize the attack smartphone to record the EM emissions when the target laptop is launching different applications or web pages. The target laptop is placed on a round table with the attack device attached on the backside to draw no attention, as shown in Fig. 7. The round table is 2.5 cm in thickness. Then, we acquire the measurements from the built-in magnetometer and transfer these EM measurements to a cloud server for further processing.

Application/Operation. For application recognition, we choose 13 popular applications available on Mac OS that cover the following categories: Productivity, Business, Entertainment, Tool, and Social Networking, with a summary in Tab. 4. For operation recognition, we take the web browser Chrome as an example and regard the operations of opening 50 different web pages as 50 different operations. We use the top 50 web sites of China listed in Alexa [2]. For the convenience of data collection, we use a *Python* script to launch applications/web pages iteratively. Each application/web page is being launched for 10 s considering that most applications can be launched within 10 *s*, with a 5 s blank period between two successive launchings.

5.2 Metrics

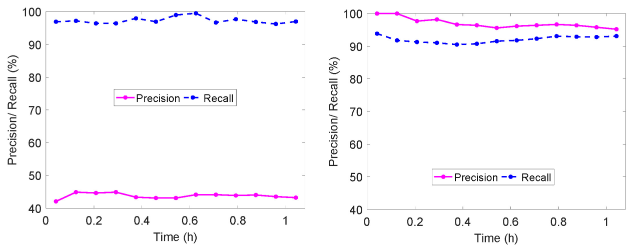
We use *precision* and *recall* to evaluate the performance of MagAttack on launching detection, and *accuracy* to evaluate the performance of MagAttack on application recognition and operation recognition.

Table 3: Summary of experimental phones.

Phone Type	iPhone SE	Nexus 5
OS	iOS	Android
Sensor Rate	100 Hz	50 Hz

Table 4: Summary of experimental applications.

ID	App	Version (MacBook Air)	Version (MacBook Pro)
P	PowerPoint	14.1.0	15.13.3
W	Word	14.1.0	15.13.3
E	Excel	14.1.0	15.13.3
C	Chrome	54.0.2840	54.0.2840
S	Safari	10.0.1	10.0.1
K	Skype	7.39	7.16
I	iTunes	12.5.3.17	12.5.3
V	VLC	2.2.3	3.0.0
M	Mail	8.2	10.1
O	OneNote	15.28	15.28
R	PDF Reader	2.4	2.4
F	FaceTime	3.0	4.0
G	Game Center	2.0	2.0



(a) Pre-screening model performance. (b) SVM model performance.

Figure 8: Performance of application launching detection.

Precision. Precision is denoted as $\frac{TP}{FP+TP}$, where TP represents the true positives, *i.e.*, the number of times that MagAttack correctly detects an application launching. Similarly, FP refers to the false positives, the number of times that MagAttack falsely classifies a time interval as an application launching.

Recall. Recall is denoted as $\frac{TP}{FN+TP}$, where FN is the number of time intervals that contain an application launching but are not detected by MagAttack.

Accuracy. For each application/web page, accuracy is defined as the ratio of the number of correctly recognized samples to the total number of testing samples. We use the average of the accuracy for all applications/web page as the final recognition accuracy of MagAttack.

5.3 Launching Detection Results

We first evaluate the launching detection performance of MagAttack. During experiments, we launch each of the 13 applications for 20 times and collect a 3900-second EM time series. We first employ the pre-screening model to locate candidate time windows that have high probabilities to contain application launching. The results in the Fig. 8(a) indicate that the pre-screening model can detect most of the launching timestamps with a recall of 97.3%, but a few false positives may occur. The SVM model further refines the pre-screening model decision by discarding false positive candidates, where we adopt the cross-validation scheme to avoid over-fitting. As a result, their combined performance achieves a precision of 97.0% and a recall of 92.1%, as shown in Fig. 8(b).

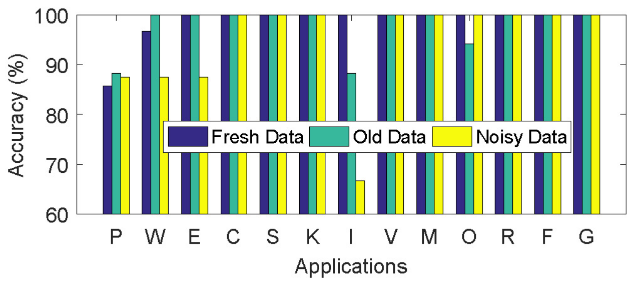
5.4 Application Recognition Results

5.4.1 Overall Performance. In the first experiment of application recognition, each of the 13 applications is launched for 50 times, with different laptop-smartphone geo-spatial locations and random laptop-smartphone orientations across 3 days. For each collection, we record a 10-second EM sample from the moment the application is launched. We randomly choose 10% samples as the testing set, and use the other samples as the training set. We repeat this process for 10 times and use the average of the 10-fold cross validation as the final results. The results in the "Fresh Data" in Fig. 9(a) demonstrate that, MagAttack achieves an average recognition accuracy of 98.6% across the 13 experimental applications.

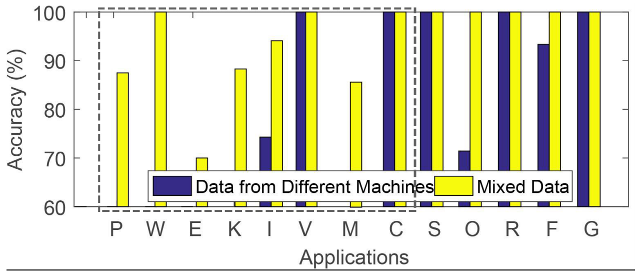
5.4.2 Impact of Data Freshness. MagAttack achieves high performance on fresh data. However, in real attacks, it is more likely that MagAttack shall operate with old training data. To evaluate the impact of data freshness, we collect training samples 20 days earlier than collecting testing samples. The results shown in the "Old Data" in Fig. 9(a) indicate that the freshness of training data has little impact on the performance of MagAttack. We assume it is because that applications barely change their components unless updated. Overall, MagAttack achieves an average accuracy of 97.7% when training data is out-of-date. One performance decrease is observed on iTunes. We assume the reason is that iTunes updates its home page with different music advertisements now and then, which leads to various EM signals.

5.4.3 Impact of Background Application. When recognizing a launched application, there might be other applications running in the background, which generate EM emissions as well and thus may interfere with the application recognition. To evaluate the impact of background applications, we train the classifier using samples without background applications, and test it using samples with two other applications running in the background. The two selected applications are a QuickTime application, which is playing music, and a MATLAB application, which is calculating. Since application launching mainly produces dynamic EM emissions, we assume those active applications may introduce more interference. The results in the "Noisy Data" in Fig. 9(a) show that MagAttack is slightly impacted by background applications when recognizing PowerPoint, Word, Excel and iTunes, but works well with other 9 applications.

5.4.4 Impact of Training Data. In practice, we cannot always obtain training data from the target laptop. To investigate the impact of training data, we collect training samples from a MacBook Air laptop, and testing samples from a MacBook Pro laptop. Both laptops are installed with the 13 applications but some of them are various in application version. The results in the "Data from Different Machines" in Fig. 9(b) demonstrate that, among the 13 applications on different laptops, MagAttack can classify 8 of them: Chrome, Safari, iTunes, VLC, Microsoft OneNote, PDF Reader, FaceTime and Game center with an accuracy of 94.2%. These 8 applications, except for the VLC player and FaceTime, have the same application version on



(a) Application recognition performance on fresh, old and noisy data.



(b) Impacts of machine diversity.

Figure 9: Performance of application recognition.

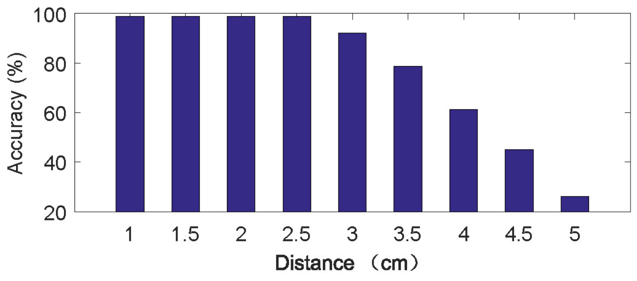
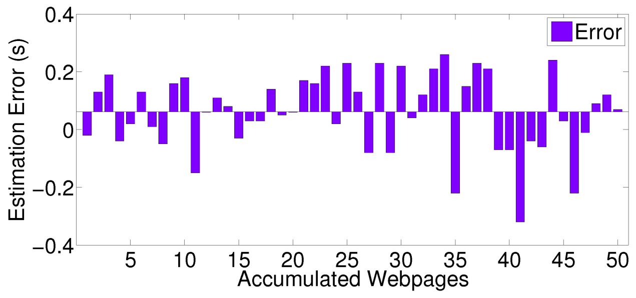


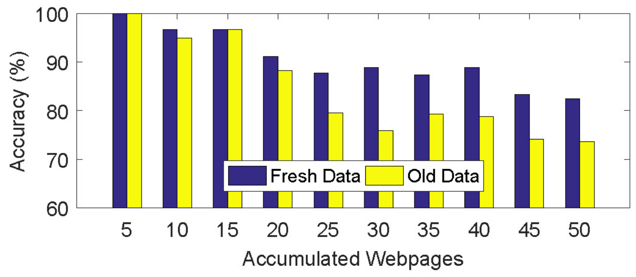
Figure 10: Performance of MagAttack with various attack distance.

both machines, as shown in Tab. 4. We find that VLC and FaceTime are the most lightweight applications by calculating the number of system calls executed when being launched. Therefore, we assume that the version difference of VLC player and FaceTime has little impact on its CPU instructions and thus the EM radiations. The rest 5 applications decrease the average recognition accuracy to 57.9% due to the differences in both application version and laptop hardware. However, we can further increase the accuracy to 94.3% by data merging. After merging training samples from both machines, MagAttack can recognize applications from either laptop with an accuracy of 94.3%, as shown in the "Mixed Data" in Fig. 9(b).

Aforementioned results demonstrate that application version and laptop model included in the training set do have impact on the recognition performance. However, we assume that MagAttack can achieve cross-device attack by including more application versions and laptop models during the training process. Specifically, the adversary can build an application library that contains common applications and their popular versions, and use the data from the library to train a comprehensive classifier. In addition, we assume that the adversary can obtain the laptop model of the victim through observation and build a corresponding classifier with a laptop of



(a) Estimation error of detecting the opening time of web pages.



(b) Web page recognition performance on fresh and old data.

Figure 11: Performance of user operation recognition.

the same model. We believe that this kind of overhead might be affordable for adversaries who attempt to track the behaviors of others.

5.4.5 Impact of Device Distance. Due to the limited sensitivity of the built-in magnetometer, the captured EM signals become weak when moving the attack device away from the laptop. To investigate the influence of the attack device placement, we vary the distance between the laptop and the phone. We train the classifier with traces collected at a distance of 2.5 *cm*, and test it with traces collected from various distances. Starting from 1 *cm*, we gradually enlarge the distance between the laptop and the phone with a step of 0.5 *cm*. The performance of MagAttack at each distance is shown in Fig. 10, from which we can see that within a distance of 1−3 *cm*, MagAttack can achieve a high accuracy (> 92%). If the attack distance is beyond this range, we consider to employ specialized hardware to enhance the signal reception capability.

5.5 User Operation Recognition Results

5.5.1 Webpage Browsing Detection. For user operation recognition, we first detect the event of web page browsing. We use a script to open 50 web pages in a random order in the Chrome web browser, and record the exact opening instant of each web page. In the meantime, we capture the corresponding EM emissions and estimate the browsing time. The results shown in Fig. 11(a) demonstrate that MagAttack can detect all of the 50 web pages with an average estimation error (in absolute values) of less than 0.12 seconds.

5.5.2 Operation Recognition Performance. To evaluate the user operation recognition performance of MagAttack, we collect 2500 samples for 50 web pages across 3 days. The "Fresh Data" in Fig. 11(b) shows the accuracy when MagAttack classifies among different numbers of web pages. Specifically, MagAttack achieves an average accuracy of 100% when differentiating 5 web pages, and 96.7% for both 10 and 15 web pages. With the increasing of web pages, the recognition accuracy slightly decreases. Nevertheless, MagAttack can still achieve an average accuracy of 84.7% in differentiating all the 50 web pages.

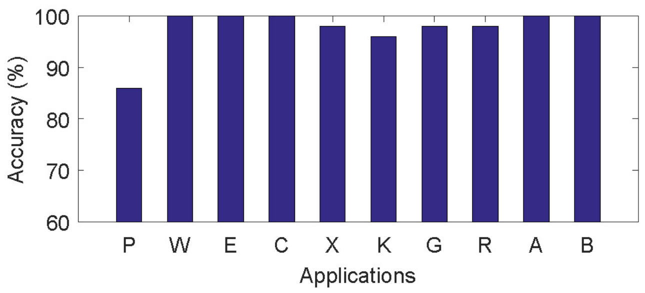


Figure 12: Performance of application recognition on the Windows laptop.

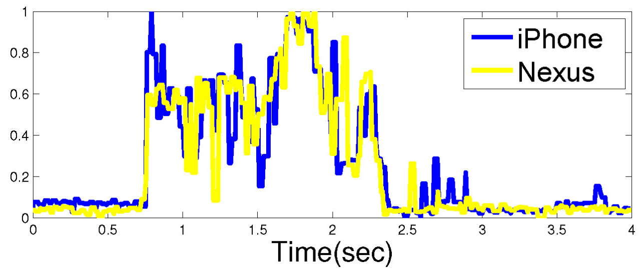
Similarly, we investigate the impact of data freshness on operation recognition as well. We collect another 500 testing samples 20 days after collecting the training samples. Based on which, MagAttack shows an average accuracy of 73.7% as revealed in "Old Data" in Fig. 11(b). The reason accounts for the performance decrease is that most web pages, unlike applications, update their page components on a daily basis. These modified web page components impact the EM signals captured on different days, and thus cause performance decrease.

5.6 Scalability of MagAttack

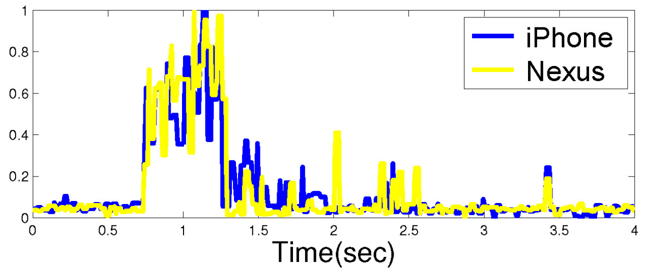
In addition to the performance of application and operation recognition, we investigate the scalability of MagAttack on various operating systems and attack devices. Specifically, we utilize a target device with operating systems other than Mac OS, and an attack device other than iPhone to perform user application recognition.

5.6.1 Impact of Different Operating Systems. To evaluate it, we conduct experiments on a Lenovo T440p laptop with a Windows OS. Similarly, we collect 50 samples for each of the following 10 popular applications available on Windows OS: PowerPoint (P), Word (W), Excel (E), Chrome (C), Internet Explorer (X), Skype (K), KuGou (G), Windows Media Player (R), Adobe Reader (A) and MATLAB (B), and employ the 10-fold cross validation to evaluate the recognition performance. The results shown in Fig. 12 demonstrate that MagAttack works well on the Windows OS, with an average recognition accuracy of 97.8%. Thus, we have reason to believe that MagAttack is OS-independent and can attack laptops with different operating systems.

5.6.2 Impact of Different Attack Devices. To evaluate it, we use two different mobile phones, *i.e.*, an iPhone SE and a Nexus 5 in Tab. 3, to track the same laptop at the same time. We collect 50 samples for each of the 10 applications and draw the EM signals of two applications (Microsoft Word and VLC player) recorded by two smartphones in Fig. 13 for illustration. From the results we can observe that EM signals of the same application recorded by different smartphones are quite similar. To quantitatively evaluate the impact of different attack devices, we train the system using samples



(a) Microsoft Word



(b) VLC Player

Figure 13: EM signals collected using different phones.

collected by one smartphone, and test it with samples collected by the other device. Since the iPhone SE has a sampling rate of 100 Hz while the Nexus 5 only has 50 Hz, we down-sample the EM signals collected by the iPhone SE to achieve the same sampling rate. The classification results demonstrate that MagAttack can achieve an accuracy of 98% in the aforementioned case. Thus, we believe that MagAttack is independent on the model of attack devices as well as the sampling rate of magnetometers.

6 DISCUSSION

In this section, we discuss the defense countermeasure of MagAttack, and the limitations of our system.

6.1 Defense

We propose two defense strategies against MagAttack from both the hardware and software perspectives.

Hardware-based Defense. One condition that enables MagAttack is that the laptop CPU leaks EM emissions which can be captured by a vicinal magnetic sensor. To address it, the electromagnetic shielding on laptops can be enhanced. For instance, the CPU and other vital components can be shielded with metal films, which may reduce a majority of the EM leakage.

Software-based Defense. The root cause of MagAttack is that various instructions generate different EM signals that in turn can be utilized to differentiate user activities. To defend MagAttack, a group of stochastic instructions can be executed by the CPU in the background, which may add random noise to the EM signals generated by user activities, thus interfere the recognition of MagAttack.

6.2 Limitations

Our implementation of MagAttack based on existing hardware has three main limitations. First, the attack distance between the laptop and the mobile device is close, in terms of 3 centimeters. Due to the limited sensitivity of the built-in magnetometer in the mobile devices on today's market, the captured EM signals from the COTS mobile device become too weak after moving the mobile device further away from the laptop. We envision that COTS mobile devices can be equipped with better sensors in the future.

Second, our algorithms may need to be re-trained in case of different CPU architectures or operating systems. We envision that robust features across laptops of different CPUs and various operating systems may be discovered in the future.

Third, currently our method works when one application or operation is performed at a time. We envision that new techniques for recognizing concurrent multiple activities may be developed in the future. We hope this work can attract more effort from the community to explore this field which has not been well studied yet.

7 RELATED WORK

Side-channel Attacks Based on EM Emissions. Pioneer work using EM leakage as the side-channel usually requires customized hardware to capture EM emissions [7, 12, 13, 25, 26]. Genkin *et al.* extract the key of RSA software implementation on a Lenovo laptop using a near-field magnetic probe with a frequency of around 100 kHz [12, 13]. Vaucelle *et al.* detect the existence of ambient electromagnetic fields using a magnetometer bracelet with a frequency of up to 50 kHz [25]. Chen *et al.* detect the usage of electrical appliances by monitoring the device electromagnetic interference (EMI) radiations with an expensive EMI measurement equipment [7]. Chen's scheme works only when the laptop is electrically connected to a power line interface, which is plugged in the wall outlet. Wang *et al.* recognize the electrical appliance usage using a wrist-worn magnetic sensor and a set of data acquisition device, with a sampling rate of 16-bit resolution at 44.1 kHz [26]. In comparison, MagAttack uses magnetometers in the COTS smartphones with a sampling rate of around 100 Hz to detect and recognize user activities on a vicinal laptop. Biedermann *et al.* present a class of EM side-channel attacks on computer hard drives using smartphone magnetic field sensors [4], which is the most related work of MagAttack. Biedermann's scheme detects what type of the operating system is booting up or what application is being started based on the ongoing operations of hard drives, and thus cannot work for applications without disk operations. Furthermore, they haven't presented the underlying principle that enables those attacks. In comparison, MagAttack works for any user applications or operations, and investigates the feasibility of such attacks from a view of CPU instructions.

Side-channel Attacks Using Mobile Devices. Prior work has demonstrated a number of side-channel attacks using the sensors in COTS mobile devices [3, 5, 22, 30, 31]. Xu *et al.* and Schlegel *et al.* show side-channel attacks using cameras [30] and microphones [22], respectively. Cai *et al.* and Aviv *et al.* show that motion sensors in mobile phones, such as accelerometers and gyroscopes, can be used to learn the user tapping and gesture input [3, 5]. Jana *et al.* recognize web pages that a user browses by tracking the memory footprint variations of the browser on Android [17], which is intrusive since the attackers need to login to the system as a process running in parallel with the browser. These attacks are used to breach the privacy of the mobile device user. In comparison, MagAttack breaches the privacy of a laptop user. Zhang *et al.* exploit smartphone magnetometers to recognize nearby household appliances [31]. In comparison, MagAttack aims to infer information regarding nearby laptop user's activities.

Side-channel Attacks on Recognizing Laptop User Activities. Prior work has exploited other forms of side-channels than EM emissions for laptop user activity recognition. Zhuang *et al.* and Zhu *et al.* use acoustic signals as the side-channel information to infer user keystrokes [32, 33]. Clark *et al.* utilize the AC power consumption of a laptop to recognize the web page that a user browses [8]. Clark's scheme requires the modification of the power outlet for measuring AC power consumption. Lu *et al.* tap the encrypted web traffic to recognize the web page that a user browses [19]. In comparison, MagAttack uses EM emissions as the side-channel information, which is non-intrusive and can be implemented on the COTS mobile devices without hardware modification.

8 CONCLUSION

In this paper, we propose MagAttack, which demonstrates the feasibility of using a COTS mobile device to infer user activities on a nearby laptop based on the EM side-channel leakage from the laptop's CPU. We implement MagAttack on the COTS smartphones without hardware modification and evaluate it with 13 commonly used applications and 50 top popular web pages in China. The experimental results show that MagAttack can conduct launching detection, application recognition, and operation recognition with high accuracy.

Future directions include designing specialized hardware to enlarge the detection distance and exploiting robust features across various application versions.

ACKNOWLEDGMENTS

This work has been funded in part by China NSFC Grant 61702451, ZJNSF Grant LGG19F020020.

REFERENCES

[1] Ali N Akansu and Richard A Haddad. 2001. *Multiresolution signal decomposition: transforms, subbands, and wavelets.* Academic Press.

[2] Alexa. 2017. Top Sites in China. http://www.alexa.com/topsites/countries/CN. (2017).

[3] Adam J Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M Smith. 2012. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC'12).* ACM, 41–50.

[4] Sebastian Biedermann, Stefan Katzenbeisser, and Jakub Szefer. 2015. Hard drive side-channel attacks using smartphone magnetic field sensors. In *Proceedings of the 19th International Conference on Financial Cryptography and Data Security (FC'15).* Springer, 489–496.

[5] Liang Cai and Hao Chen. 2011. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion.. In *Proceedings of the 6th USENIX conference on Hot Topics in Security (HotSec'11)*, Vol. 11. 9–9.

[6] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2, 3 (2011), 27.

[7] Ke-Yu Chen, Sidhant Gupta, Eric C Larson, and Shwetak Patel. 2015. DOSE: Detecting user-driven operating states of electronic devices from a single sensing point. In *Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communications (PerCom'15).* IEEE, 46–54.

[8] Shane S Clark, Hossen Mustafa, Benjamin Ransford, Jacob Sorber, Kevin Fu, and Wenyuan Xu. 2013. Current events: Identifying webpages by tapping the electrical outlet. In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS'13).* Springer, 700–717.

[9] Ronald R Coifman and M Victor Wickerhauser. 1992. Entropy-based algorithms for best basis selection. *IEEE Transactions on information theory* 38, 2 (1992), 713–718.

[10] dtrace.org. 2017. About Dtrace. http://dtrace.org/blogs/about. (2017).

[11] Gartner. 2018. Gartner Says Worldwide Device Shipments Will Increase 2.1 Percent in 2018. https://www.gartner.com/newsroom/id/3849063. (2018).

[12] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. 2015. Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *Proceedings of the 17th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'15).* Springer, 207–228.

[13] Daniel Genkin, Itamar Pipman, and Eran Tromer. 2015. Get your hands off my laptop: Physical side-channel key-extraction attacks on PCs. *Journal of Cryptographic Engineering* 5, 2 (2015), 95–112.

[14] Github. 2016. DTrace-win32. https://github.com/prash-wghats/DTrace-win32. (2016).

[15] Gregose. 2016. Syscall-table. http://syscalls.kernelgrok.com/. (2016).

[16] Sidhant Gupta, Matthew S Reynolds, and Shwetak N Patel. 2010. ElectriSense: single-point sensing using EMI for electrical event detection and classification in the home. In *Proceedings of the 12th ACM international conference on Ubiquitous computing (Ubicomp'10).* ACM, 139–148.

[17] Suman Jana and Vitaly Shmatikov. 2012. Memento: Learning secrets from process footprints. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy (S&P'12).* IEEE, 143–157.

[18] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference (CRYPTO'99).* Springer, 388–397.

[19] Liming Lu, Ee-Chien Chang, and Mun Choon Chan. 2010. Website fingerprinting and identification using ordered feature sequences. In *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10).* Springer, 199–214.

[20] S Lawrence Marple. 1987. *Digital spectral analysis: with applications.* Vol. 5. Prentice-Hall Englewood Cliffs, NJ.

[21] François Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping Chen, and Eamonn Keogh. 2014. Dynamic time warping averaging of time series allows faster and more accurate classification. In *Proceedings of the 2014 IEEE International Conference on Data Mining (ICDM'14).* IEEE, 470–479.

[22] Roman Schlegel, Kehuan Zhang, Xiao-yong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. 2011. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS'11)*, Vol. 11. 17–33.

[23] Amelia Shen, Abhijit Ghosh, Srinivas Devadas, and Kurt Keutzer. 1992. On average power dissipation and random pattern testability of CMOS combinational logic networks. In *Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design (ICCAD'92).* IEEE, 402–407.

[24] Amit Singh. 2006. *Mac OS X internals: a systems approach.* Addison-Wesley Professional.

[25] Cati Vaucelle, Hiroshi Ishii, and Joseph A Paradiso. 2009. Cost-effective wearable sensor to detect EMF. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09).* ACM, 4309–4314.

[26] Edward J Wang, Tien-Jui Lee, Alex Mariakakis, Mayank Goel, Sidhant Gupta, and Shwetak N Patel. 2015. Magnifisense: Inferring device interaction using wrist-worn passive magneto-inductive sensors. In *Proceedings of the 17th ACM international conference on Ubiquitous computing (Ubicomp'15).* ACM, 15–26.

[27] Wikipedia. 2017. CMOS. https://en.wikipedia.org/wiki/CMOS. (2017).

[28] Wikipedia. 2017. k-nearest neighbors algorithm. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm. (2017).

[29] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2, 1-3 (1987), 37–52.

[30] Nan Xu, Fan Zhang, Yisha Luo, Weijia Jia, Dong Xuan, and Jin Teng. 2009. Stealthy video capturer: a new video-based spyware in 3g smartphones. In *Proceedings of the 2nd ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec'09).* ACM, 69–78.

[31] Mi Zhang and Alexander A Sawchuk. 2012. A preliminary study of sensing appliance usage for human activity recognition using mobile magnetometer. In *Proceedings of the 14th ACM international conference on Ubiquitous computing (Ubicomp'12).* ACM, 745–748.

[32] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. 2014. Context-free attacks using keyboard acoustic emanations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS'14).* ACM, 453–464.

[33] Li Zhuang, Feng Zhou, and J Doug Tygar. 2009. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security* 13, 1 (2009), 3.