2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON) | 979-8-3503-0052-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/SECON58729.2023.10287495

LeakThief: Stealing the Behavior Information of Laptop via Leakage Current

Dian Ding †, Yi-Chao Chen †, Xiaoyu Ji ‡, Guangtao Xue* †
† Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
‡ Department of Electronic Information and Electrical Engineering, Zhejiang University, China
Email:{dingdian94, yichao}@sjtu.edu.cn, xji@zju.edu.cn, gt_xue@sjtu.edu.cn

***Abstract*—Smart devices are proliferating in every aspect of our lives, providing convenience but also exposing us to the risk of information leakage at any moment. Attackers can monitor the user and infer private information such as the personality and preferences by stealing the behavior information. In this paper, we investigated the potential threat of information stealing via the leakage current of laptop and electrodes in wearable devices (e.g. smart watches and bracelets). Specifically, the leakage current in the laptop adapter can flow from the metal casing into the human body and be collected by electrodes in wearable devices when the user is using a laptop with a metal casing (e.g. MacBook). We verified the correlation between leakage current and working states of the laptop, where different operations corresponding to different CPU instructions can generate different leakage currents. Based on this, we propose *LeakThief*, the system consists of three components, leakage current detection, application operation detection and application recognition. The experiments in real-world environment demonstrated that the proposed system is able to recognize 10 common applications with high accuracy, including launching-based (97.5%) and in-application operation-based recognition (83.8%).**

***Index Terms*—Side Channel Attack, Leakage Current, User Privacy, Laptop**

I. Introduction

The COVID-19 pandemic has permanently changed the device usage patterns of people, with lifestyle changes such as remote working and digital education driving the number of devices people own and use to continuously increase. There will be 16.8 billion mobile devices such as personal computers (PCs), tablets and smart phones in use worldwide by 2023 [1]. While bringing convenience to work and study, these devices also increase the risk of information leakage to the users. Attacks on PCs, smart phones and other devices are proliferating, and researchers have demonstrated the feasibility of stealing user information based on motion signals [2]–[4], acoustic signals [5], [6], wireless signals [7]–[9] and magnetic signals [10], [11].

The wearable properties of smart watches and bracelets meaning that the users are exposed to the risk of information leakage for long periods of time, we therefore consider the security of these devices to be of greater concern. As these devices continue to push the boundaries of performance and service user-friendliness. More and more consumers are inclined to wear devices such as smart watches and bracelets in their daily lives. Security issues of wearable devices have become a hot topic of research, and systems of information leakage based on motion signals [12], [13] and acoustic signals [14] have been proven.

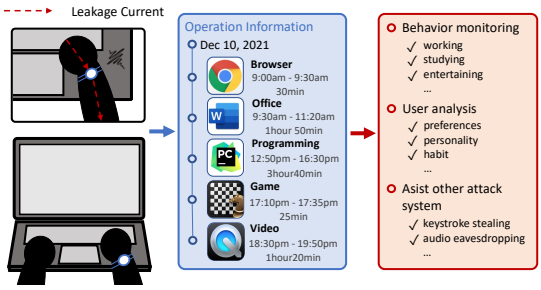


Fig. 1. The behavior information stealing on laptop.

Apart from the built-in motion sensors and microphones, the security risk of electrodes in smart watches or bracelets has been overlooked. The electrodes in a smart watch monitor ECG of the user by detecting the micro electrical pulses of heart beats [15]. By wearing these devices, people are able to keep an eye on their heart rate and other information while working and exercising. However, we found that the electrodes close to the human body can steal user's behavior information while using the laptop with a metal casing, such as which application they have launched or in which application they are performing the operation. This allows monitoring of the user's behavior to infer their interests and personality [10], [16] and even to assist other attack systems [11].

The signal collected by the electrodes comes from the safety capacitor [17] in the adapter of a laptop with metal casing, such as a MacBook. As shown in Fig. 1, when using a laptop, the user's hands naturally touch the metal casing of the laptop. The leakage current from the safety capacitor flows into the body through the metal casing. LeakPrint [18] used leakage current of laptop in its idle state to sense the human capacitive for user identification. We further explored the principles of leakage current, and verified the correlation between the leakage current and the working state [19] of the laptop. When the user launches different applications, the CPU executes different instructions that change the working state of the laptop, thus affecting the leakage current. Recent researches [20] have demonstrated that the built-in electrodes

* Corresponding author.

of smart watch can perceive the weak current in the human body. Therefore, the electrodes in wearable devices present a potential risk of information leakage on user behavior by monitoring the leakage current from the laptop.

However, the lack of API for accessing the built-in electrode of commercially available smart watches that make it difficult to obtain raw data from the electrodes. In this paper, we disassemble a commercially available smart watch (Apple Watch S6) and read the leakage current collected from the built-in electrodes through an external AD2 [21]. Based on this, we built a behavior stealing system via leakage current, *LeakThief*, to verify and alert users early on the potential security risks of the electrodes in wearable devices.

Currently, the growing user demands for applications are driving increasing hardware performance, the electrodes are enabling attractive applications in human-computer interaction, such as emotion recognition [13] and gesture recognition [22], [23]. As a result, the current restrictions on the API for accessing the built-in electrode of smart watches have a tendency to open up in the future. At the same time, metal casing is a trend that is gaining popularity among consumers due to its physical strength and thermal performance, with companies such as Apple, HP and HuaWei all launching laptops with metal casing. Therefore, we believe that user behavior stealing based on leakage current will be a universal and non-negligible potential security threat.

The development of a behavior stealing system based on the leakage current poses a number of challenges. First, we have to extract the signal segments corresponding to the user's operations from the weak signals collected by the electrodes; Second, different operations of applications have different levels of discrimination, which affects the performance of the application recognition; and finally, operations within the same type of applications may generate similar leakage currents.

We developed *LeakThief* to address these challenges. First, we compared the spectral characteristics of the signals at different states of power consumption and used the collected signals to compute logarithmic short-time energy (log-STE) signal for operation detection. Second, the leakage currents during application launching process are more discriminating than in-application operations. We discarded operation samples of closing the application and classified all operation samples into two categories: application launching and in-application operation. Finally, we extracted features of different applications using Temporal Convolutional Network (TCN). The contributions of this research are as follows:

- We explored the the dynamic characteristics of leakage current from laptops with metal casing, and verified the feasibility of behavior stealing from the leakage current.
- We verified the correlation between leakage current and the working state of laptop, and detected user application launching and operations from the leakage current.
- We divided the operations into application launching and in-application operations, and achieved the application recognition based on the leakage current;
- We conducted experiments in a real environment and the results demonstrated the potential threat of *LeakThief* in behavior stealing based on application launching (97.5%) and in-application operations (83.8%).

II. Related Work

A. Motion Signals

Numerous daily activities of people will generate captureable motion signals that can lead to information leakage, including the body movements captured by wearable devices [12], [13], and vibrations of surrounding objects caused by operations such as tapping the keyboard or the smart phone [4], [24].

B. Acoustic Signals

Behaviors such as handwriting or keyboard tapping inevitably produce distinctive acoustic signals, and researchers have demonstrated the feasibility of inferring keyboard tapping [5] and handwriting content [6] based on acoustic signals.

C. Electromagnetic Signals

Recent researches used the electromagnetic signals to steal the information about the victim's behavior. Enev [25] confirmed that TVs can leak information about household activities from power lines. Genkin [26] achieved the key extraction from laptops. MagThief [10] used the built-in magnetic sensors of smart phone to identify the different applications. MagAttack [11] steal the user's behavior while using laptop.

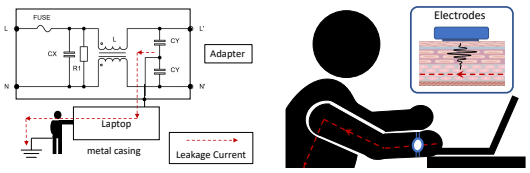
Compared with Enev, information leakage from laptops can cause more serious damage than home activities such as playing videos. The key extraction attack requires the attacker to perform specific calculations on the target device. Meanwhile, MagAttack required the smart phone and laptop to be in close proximity (less than $3cm$) to achieve high accuracy of application recognition. With these requirements, the attack is significantly more difficult to execute.

LeakThief exploits the correlation between leakage current and the working state of the laptop to enable application-level information stealing. The system passively receives the leakage current containing information about the user's operation. Furthermore, users are not fully aware of the threat posed by the built-in electrodes of wearable devices. Therefore, attacks based on leaking current to steal the behavior information will become a non-negligible security risk and we proposed *LeakThief* to alert users early.

III. Background

A. Leakage Current of laptop

1) Leakage Current: As a laptop with a metal casing (such as a MacBook) is connected to a power source, the metal casing of the laptop carries the leakage current from the adapter [17]. The leakage current comes from the Y-capacitor (safety capacitor) of the adapter, which is part of the EMI (electromagnetic interference) filtering circuit of the switching power supply to eliminate common mode interference and improve electromagnetic compatibility, and is usually configured



(a) Leakage current of laptop. (b) Human-machine channel.

Fig. 2. Fundamental principle of leakage current: 2(a) Under normal use, leakage current flows from the laptop through the user's hands into the body and eventually into the ground; 2(b) The electrode at the wrist can perceive the leakage current in human body.

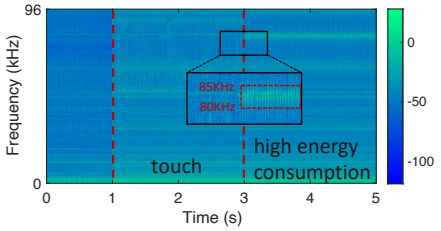


Fig. 3. The spectrogram of the leakage current presents the leakage current collected by the electrodes, where the two dashed lines indicate the time when the user touches the metal casing of laptop and the time when the CPU power consumption changes respectively.

on both the high and low voltage sides of the SMPS (switch mode power supply). As a common mode capacitor, grounding of the Y capacitor generates leakage current [18], [27]. The leakage current in the metal casing can be written as:

$$I = 2\pi f C_h U_h + k C_l U_l + I_{cmi} \tag{1}$$

where the leakage current in the high voltage side is $2\pi f C_h U_h$. f refers to the mains frequency, C_h indicates the size of the capacitor, and U_h indicates the voltage. In the low-voltage side, k is the leakage current constant (about 0.01 to 0.03 depending on the manufacturer), C_l and U_l denote the corresponding Y capacitor and voltage. The Y capacitors used in laptop adapters are typically around $5nF$. I_{cmi} represents the common mode interference (high-frequency harmonics) in the SMPS and a laptop powered by the $220V/50Hz$ mains generates roughly $0.3mA$ of leakage current ($U_l = 12V$).

Of these, the common mode interference is affected by the loading state [19], such as the working state of the laptop. Therefore, we attempted to capture the working state of laptop from the common mode interference and further extract the behavior information.

2) Human-Machine Channel: When the user is using the laptop, the user's hands are placed on the laptop and the feet are placed on the ground naturally. In this case, the metal casing, the user and the ground form a current path and the leakage current flows from the metal casing through the body and eventually into the ground, as shown in Fig. 2(a).

When the user wears a smart watch or bracelet, the electrodes on the backside of the smart watch or bracelet close to the body will be affected by the leakage current flowing through the body [28], as shown in Fig. 2(b). We disassembled a commercially available smart watch (Apple Watch S6) and used the built-in electrode to receive the leakage current.

B. Feasibility of Application Recognition

1) Modulation of Leakage Current: Since the leakage current of Y-capacitors in the adapter is related to the loading state (i.e. the working state of laptop) [19]. We attempted to modulate the electronic unit of the laptop (e.g. CPU, electronic fan, etc.) to vary the working state. We chose to investigate the relationship between working state and leakage current by modulating the CPU [29]. Since under normal working state, the CPU consumes between $20W$ and $90W$ of power, while the electronic fan consumes only $3W$. We used a MacBook pro in the feasibility experiment. We perceived the changes in the leakage current flowing through the body using the built-in electrode of Apple Watch S6 at the user's wrist, and collected the leakage current using AD2 with the sampling rate of $192KHz$.

As shown in Fig. 3, we generated the spectrogram of the leakage current collected by the electrodes via Short Time Fourier Transform (STFT). The weak signal collected by the electrode when the user does not touch the laptop comes from the human antenna effect under the influence of the ambient electric field [20]. The magnitude of the signal is increased when the user touches the metal casing of the laptop. Besides, when we modulate the CPU to the state of high power consumption, the amplitude of the leakage current in the high frequency band increases further and is mainly concentrated at $80KHz$ to $85KHz$. Therefore, we exploited the leakage current in this frequency band in the feasibility experiment of application recognition.

2) Application Recognition based on Leakage Current: Previous researches [11] have demonstrated the feasibility of application recognition during application launching based on system calls. The start-up service framework WindowServer sends an application start-up message, which runs the requested application via system calls. Different applications will invoke different system calls at different frequencies during the launching process.

The CPU chip consists of a large number of CMOS (Complementary Metal Oxide Semiconductor) transistors arranged in a lattice, which are responsible for performing arithmetic, logic and control operations etc. The different system calls contain different sets of instructions executed by the CPU. The CPU involves different numbers of CMOS transistors in the execution of various instructions, which generates different power consumption and thus affects the leakage current of the adapter. Therefore, for different applications, different system calls invoked during the launching process will generate different leakage currents.

IV. Threat Model

In this section, we introduce the threat model of *LeakThief*. The goal of the attacker is to steal the user's behavior information while using the laptop, such as which application

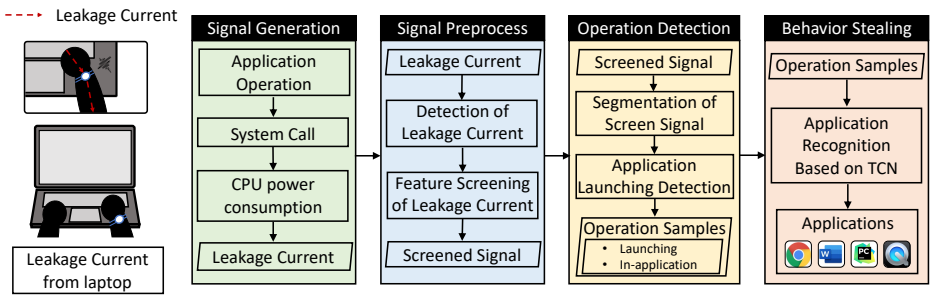


Fig. 4. System architecture of the *LeakThief* system .

the user has launched and in which application the user is operating.

We consider the following attack scenario: the user is using a laptop with metal casing and keeps the laptop in charged. The user is not wearing gloves and is not using an external keyboard or mouse. While using the laptop, the user's hands are naturally in contact with the metal casing. The user wears a smart watch or bracelet with electrodes that can perceive the leakage current from the laptop. Based on the leakage current, the attacker is able to steal information about the user's operations while using the laptop. This enables the monitoring of the user's operations and even inferring the user's interests and personality from the applications used [16].

V. System Design

A. Preprocess of Leakage Current

1) Detection of Leakage Current: In order to achieve application recognition based on leakage current, the system must be able to accurately detect the time when the user touches the laptop. As shown in Fig. 3, when the user touches the laptop, the electrodes are able to perceive an increase in the magnitude of the leakage current flowing through the body. The increase in amplitude is concentrated in the high frequency band ($50KHz$ to $85KHz$). Therefore, the most straightforward method is to set an intensity threshold for the leakage current in this frequency band to determine whether the user is touching the laptop or not.

A band-pass filter with a high order cut-off is well suited to the main frequency distribution of the signal. For the effect of low frequency noise, a band-pass filter was used to improve the SNR of the signal, as shown in Fig. 5. The amplitude enhancement caused by the leakage current is clearly visible in the filtered signal, but the fluctuation of the signal is not conducive to the detection of leakage current. To reduce the effect of signal fluctuations, we set a processing window of $0.01s$ and calculate log-STE of the processing window as follows:

$$E(j) = 10\ log \sum_{i=j}^{j+0.01\times F_s} y(i)^2 \tag{2}$$

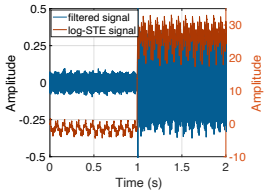


Fig. 5. Detection of leakage current.

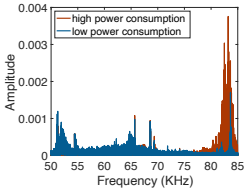


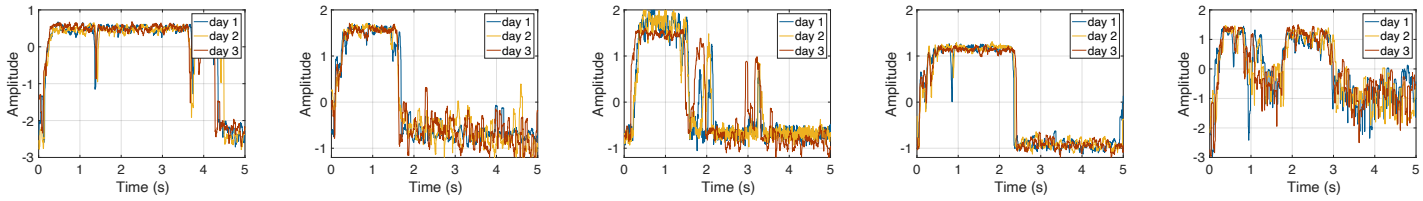
Fig. 6. Frequency screening of leakage current.

where $y(i)$ represents the leakage current signal and F_s represents the sampling rate. As shown in Fig. 5, by setting a threshold we are able to detect the leakage current with a low delay.

2) Preprocess of Leakage Current: After the detection of leakage current, we need to perform preprocessing to extract the signal in the frequency band associated with the CPU instructions. Due to the significant fluctuations in the leakage current collected by the electrodes and the fact that the amplitude of the signal does not change consistently across the different frequency bands when the CPU power consumption is boosted. Therefore, we need to filter the signal to improve the correlation between the signal and the CPU state. We performed the screening of frequencies based on the detection of leakage current.

As shown in Fig. 6, we took separate samples of the leakage current at low and high CPU power consumption with a sample duration of $1s$, so we were able to obtain frequency information with an accuracy of $1Hz$. We used the Fast Fourier Transform (FFT) to calculate the spectrogram corresponding to the two samples. Consistent with the feasibility experiments, when CPU power consumption increases, the increase in the magnitude of the leakage current is mainly concentrated around $83KHz$.

Similar to the detection of leakage current, we extracted the signal at the target frequency using a band-pass filter ($82KHz$ to $84KHz$). Also, to address the fluctuations present in the signal, we chose the same log-STE signal for processing, and we set the window length to $0.05s$. To reduce the computational burden on the system, we down-sampled the filtered signal before calculating the log-STE signal.



(a) PyCharm. (b) Microsoft Word. (c) Chrome. (d) QuickTime Player. (e) Chess.

Fig. 7. The leakage current when launching different applications.

3) Segmentation of Leakage Current: The leakage current collected by the electrodes is able to reflect the operations performed by the user on the laptop. However, when the user performs simple operations such as typing or clicking, the signal-to-noise ratio of the signal is significantly reduced, making it difficult to directly extract the signal patterns corresponding to these operations.

We attempted to extract the different operations of the user by segmenting the leakage current based on the amplitude. The signal was processed using a mean window function with a processing window of $0.5s$ to further smooth out the fluctuations in the signal. Based on the down-sampling during preprocessing, the smoothing window function will not impose too much computational burden.

We segmented these operations by setting the amplitude threshold, where the start threshold is $M_{start} = 0.5 \times A_1 + 0.5 \times A_2$ and the end threshold is $M_{end} = 0.3 \times A_1 + 0.7 \times A_2$. $A1$ and $A2$ indicate the maximum and minimum values of the signal. Finally, we normalized the samples and widened them on the time axis to keep the sample length consistent; we set the sample length to $5s$.

Compared to more complex instructions such as launching an application, simple instructions such as typing are usually reflected in the leakage current as narrow pulses. We can remove most of the pulses caused by typing by setting the thresholds. Consider that samples caused by typing may be captured incorrectly, and that operations such as copying, pasting text and closing applications possess similarities across applications. In order to improve the accuracy of application recognition, we need to filter the segments after segmentation.

B. Application Recognition

1) Application Launching Detection: The launching process of an application involves complex system calls [11], and the different instructions for different applications are reflected in the variation of the leakage current. Therefore, we attempt to extract segments from the application launching process to achieve high accuracy application recognition. We selected 10 commonly used applications (Tab. I) and collected samples of leakage current corresponding to the launching process and different in-application operations, such as copy and paste text, save and read files and close applications.

We performed the sample collection and processing over 3 days, with an interval of 3 days between each two days. To ensure signal stability, we did not change the version of the system or the applications during this time. As shown in Fig. 7, the samples generated by the applications of the launching process can remain stable without changing the version. We classified all samples into two categories, launching and in-application operation, and used classical time-series algorithms such as k-nearest neighbors (kNN), linear discriminant analysis (LDA), support vector machine (SVM), random forest (RF), and convolutional neural network (CNN) for classification. The CNN model contains four 1-dimensional convolutional layers and two fully-connected layers, with a max pooling layer connected after every two convolutional layers. The classification accuracy is shown in Tab. II.

TABLE I
SUMMARY OF APPLICATION OPERATIONS.

Application	Type	Operation
PyCharm	programming	open and save file, close app
CLion	programming	open and save file, close app
QuickTime	video	pause and start video, close app
IINA	video	pause and start video, close app
Microsoft Word	office	open and save file, close app
Microsoft PPT	office	open and save file, close app
Chrome	browser	click the link and close app
Safari	browser	click the link and close app
Chess	game	save, load the game and close app
Stardew Valley	game	load the game and close app

TABLE II
ACCURACY OF APPLICATION LAUNCHING DETECTION.

KNN	LDA	SVM	RF	CNN
93.7%	96.3%	98.6%	97.5%	99.3%

2) Recognition based on Application Launching: Different applications generate different signals during startup, and we build an application recognition model based on Temporal Convolutional Network (TCN) [30] and Residual Network (ResNet) [31]. The main body of our proposed model is the TCN module, and each TCN module consists of four dilation convolutional layers, which are able to acquire a larger perceptual domain with the same amount of computation than the normal convolutional layer. Each two dilation convolutional layers are connected to a max pooling layer, and feature information of different depths is fused through a residual connection, as shown in Fig. 8.

Specifically, we start with initial feature extraction of the leakage current samples through the convolutional layer. We

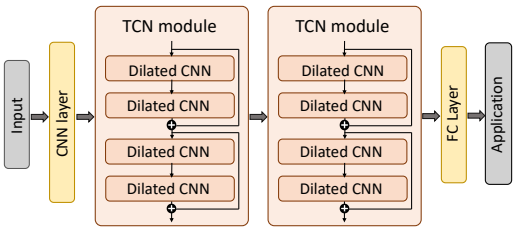


Fig. 8. Architecture of application recognition network.

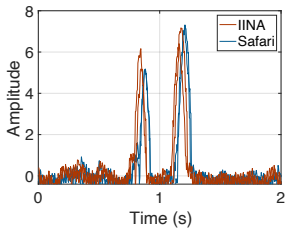


Fig. 9. Closing different applications.

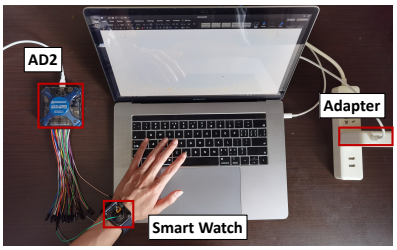


Fig. 10. Experimental setup of *LeakThief*.

use a 32-layer convolutional kernel of dimension 11×1. We then use two consecutive TCN modules, in which the 32-layer dilated convolutional scales is 7 and 5, and the extraction dilutions in the modules is 1 and 2. The application recognition is then implemented after two fully-connected layers and the SoftMax activation function. The number of neurons in the fully connected layers is 1024 and 64.

3) Recognition based on Application Operation: Compared to the samples of the leakage current corresponding to the application launching, it is more difficult to distinguish between other operations performed by the user in different applications, such as pause and play a video, copy and paste text, etc. These operations involve far fewer instructions than the launching process. We attempted to differentiate all operations performed by the user within the application in a uniform manner. For the in-application operations, system achieved an accuracy of about 65.3% based on the proposed model. By comparing the samples of different application operations, we found that the samples generated by closing the application have a higher similarity to those generated by in-application operations during application usage.

Fig. 9 shows the leakage current samples from two different types of applications (Safari (Browser) and IINA (Multimedia)) during the closing process. It can be seen that both applications generate two narrow pulses during the closing process, and that the amplitude of the two pulses and the interval between them are similar. Therefore, we considered that the leakage current generated by closing application were less identifiable and attempted to improve the accuracy of the application recognition based on in-application operation by removing the samples corresponding to closing different applications.

Specifically, in a continuous period of segments between two application launchings, system assume that the last segment corresponds to the operation of closing the application and discard it. After removing the samples during the closing process, the accuracy of application recognition based on the in-application operation is about 83.8%. Furthermore, the application recognition can be optimized with contextual information.

VI. Evaluation

A. Evaluational Setup

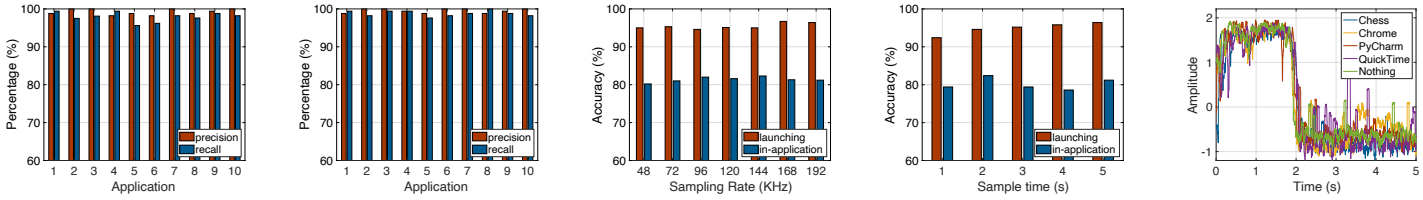
In the experiment, we used a MacBook Pro as the information stealing target of *LeakThief* and used the built-in electrodes of Apple Watch S6 at the user's wrist to collect the leakage current, as shown in Fig. 10. The MacBook Pro was placed on a desk and kept in charge. The user's hands were placed on the keyboard while using the laptop, naturally in contact with the metal casing. During the experiment, the user was not wearing gloves and did not use an external keyboard or mouse. We used the AD2 (DILIGENT) with the sampling rate of $192KHz$ to collect the leakage current. Once the wearable manufacturer opens up access to the built-in electrodes, we can implement the reading of leakage currents on the wearable device and transmit the signal to a server for user behavior recognition via WiFi.

We collected the leakage currents of different operations on 4 separate days, including the application launching and in-application operations, as shown in Tab. I. We selected 10 common applications covering 5 types including programming, video, office, browser, game. To verify the stability of the leakage current, the interval between each acquisition was 3 days. For each application, we collected the leakage current of launching process and the in-application operations 40 times each. We divided the collected samples into a training set and a test set for cross-validation.

In the detection of leakage current, we utilized a band-pass filter of $50KHz$ to $85KHz$ and set the window function length to $0.01s$ to calculate the log-STE signal to detect the touch behavior. In the detection of operations, we used band-pass filters from $82KHz$ to $84KHz$ and calculated the log-STE signal with the same window function to detect both application launching and in-application operations. We set the sample time length to $5s$ to guarantee the integrity of the information during the application launching process, as shown in Fig. 7. Before operation recognition, we reduced the sample length to 1200 by down-sampling to reduce the redundancy of information in the samples and the computational burden.

B. Micro Benchmarks

1) Operation Detection: We performed application launching, application closing and other in-application operations on 10 commonly used applications. After preprocessing the leakage currents, we implemented operation detection by segmentation based on amplitude. We evaluated the performance of operation detection with precision and recall. As shown in Fig. 11(a), the system maintains high precision and recall in operation detection of different applications. In our experiments, we found that the detection accuracy for application



(a) Operation detection. (b) Launching detection. (c) Influence of sampling rate. (d) Influence of sample time. (e) The launching process of Microsoft Word.

Fig. 11. Evaluation of the *LeakThief* system.

launching was significantly higher than that for in-application operations. This is because the instructions that an application needs to execute during the launching process are usually more complex and more stable compared to the in-application operations.

We then evaluated the performance of the operation launching detection in terms of precision and recall. A uniform model for the detection of application launching was developed for different applications. As shown in Fig. 11(b), the system maintains high precision and recall in the detection of launching operations across different applications.

2) Application Recognition: Application recognition contains both launching-based recognition and in-application operation-based recognition. The accuracy of the launching-based recognition is significantly higher than that of the in-application operation-based recognition, with 97.5% and 83.8% respectively. We can optimize the results of in-application operation recognition with information about the application launching.

C. Influence of System Parameters

1) Influence of Sampling Rate: In the feasibility experiment, we set the sampling rate of AD2 to $192KHz$ to improve the signal-to-noise ratio of the leakage current. Based on the fluctuations of the signal in the high frequency band of the leakage current ($82KHz$ to $84KHz$), we were able to achieve the application recognition based on launching process or in-application operation. It is not common for devices to meet the sampling rate requirement, so we seek to reduce the limitation of the sampling rate based on the aliasing effect [32]. The aliasing effect can be written as:

$$f_a = min|f_o - Nf_s| \tag{3}$$

where N is an integer, f_a, f_o, and f_s respectively indicate the aliasing frequency, the signal frequency, and the sampling rate. We are able to acquire signals in the high frequency band at a low sampling rate. For example, with the sampling rate of $48KHz$, the $80KHz$ signal can generate a aliasing signal at a lower frequency ($16KHz$). In this experiment, we evaluated the application recognition accuracy of *LeakThief* in at various sampling rates from $48KHz$ to $192KHz$. During this process, we kept the sample time and length at $5s$ and 1200. As shown in Fig. 11(c), as the sampling rate was reduced from $192KHz$ to $48KHz$, the accuracy of application recognition didn't show a significant decrease, including application launching and in-application operations.

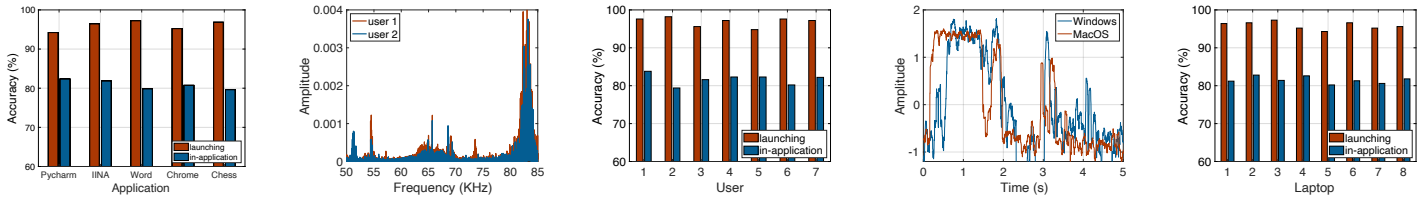
2) Influence of Sample Time: The same operation of the application, including the application launching and in-application operation, is not affected by the user's operation time. As shown in Fig. 7, different instructions executed by different applications during the launching process will generate signal samples of different length and amplitude in the leakage current. For 10 common applications, we found that the duration of the different operations of the application does not exceed $5s$. Therefore, we set the sampling time to $5s$ to cover the entire process of the application operation and avoid the information loss.

In this experiment, we evaluated the application recognition accuracy of *LeakThief* with the sample time ranges from $1s$ to $5s$ and kept the sampling rate at $192KHz$. As shown in Fig. 11(d), with a sample time of less than $3s$, the recognition accuracy of the application launching was significantly reduced from 96.2% (sample time of $3s$) to 92.4% (sample time of $1s$). This is because information during application launching is lost as the sample time decreases. In contrast, the majority of in-application operations duration was below $1s$ and the recognition accuracy was able to be maintained.

D. Influence of Devices

1) Influence of Other Application: In normal circumstances, the user may be running several applications at the same time while using the laptop. Therefore, we need to consider the effect of other applications on the leakage current when performing an application operation. Here, we assume that the user only performs operations in one application at the same time. For example, the user will not perform any other operations during the launching process of the first application. As shown in Fig. 11(e), we took the launching process of Microsoft Word as an example to present the impact of other applications on it during the running process. It can be seen that the influence of other applications on the leakage current during the application operations is extremely weak when no operation is performed in other application. And it is worth noting that during the launching process of Word, QuickTime Player (video) is in the state of playing a video.

In this experiment, we evaluated the recognition accuracy of *LeakThief* under the influence of other applications. As shown



(a) Influence of other application. (b) The spectrogram of different users. (c) Influence of user. (d) The launching process of Chrome. (e) Influence of different laptops.

Fig. 12. Evaluation of the *LeakThief* system.

in Fig. 12(a), other applications do not affect the recognition accuracy of the system, including launching-based recognition and in-application operation-based recognition. However, we did not consider some special cases, such as the user perform the operation of other applications while PyCharm is training the neural network. Depending on the correlation between leakage current and CPU power consumption, this would inevitably affect the leakage current generated by the operation. In this case, the system is unable to extract and recognize the user's operation while using the laptop.

2) Influence of User: Recent researches [20], [33], [34] have confirmed the differences in human capacitance between users. Therefore, we need to evaluate the performance of *LeakThief* across different users to demonstrate the universality of the system. First, we compared the leakage currents collected from different users' wrists. As shown in Fig. 12(b), when the CPU is at high power consumption, the spectrogram of different users show a significant increase in the leakage signal in the high frequency band ($82KHz$ to $84KHz$), which is aligned with the frequency screening in the signal preprocessing.

We hired 7 participants to evaluate the recognition performance of *LeakThief*, containing 4 males and 3 females with ages ranging from 21 to 43 (average age of 28.3). We extracted the operations of users while using the laptop from the same frequency band ($82KHz$ to $84KHz$) of the leakage current. As shown in Fig. 12(c), the system can maintain high performance in application recognition for different users, both launching-based and in-application operation-based. Since the extracted leakage currents are only related to the instructions of application, there is no reference to the user's operating rate, habits, etc.

3) Influence of Laptop: Laptops with metal casing are favored by consumers for their casing strength and thermal performance. In order to verify the prevalence of the behavior stealing system based on leakage current, we evaluated *LeakThief* on laptops of different brands, including three Apple laptops (laptop 1, 2, 3), two HP laptops (laptop 4, 5), two HUAWEI laptops (laptop 6, 7) and one XiaoMi laptop (laptop 8). The leakage current comes from the switching mode power supply in the adapter, and the leakage currents of different laptops usually have different characteristic frequencies under the operating state of high energy consumption [35]. Therefore, we can use the spectral subtraction method to process the spectral information of the two working states and thus extract the characteristic frequencies of different laptops.

We collected leakage currents of launching and in-application operations as described in Sec. VI-A. The same application running in different operating systems can also generate similar leakage current. Fig. 12(d) shows the leakage current generated by Chrome during the launching process in different operating systems. Therefore, we can consider the same applications with different versions in different operating systems as the same class. As shown in Fig. 12(e), the system can maintain high accuracy in the application recognition of different laptops. The attacker could build the dataset of mainstream laptops with metal-casing to extend the stealing scope. We believe that while collecting operation samples of applications with different versions is burdensome, it is affordable for the attacker.

VII. Discussion

a) Defense: The first is a hardware-based defense strategy. The leakage current received by *LeakThief* comes from the safety capacitor in the laptop adapter and flows through the human body via the contact with the metal casing of the laptop. Therefore, we can defend against the signal as it is generated and transmitted. For example, the user can disconnect the laptop from the adapter to cut off the source of the signal. Besides, the user can use an external mouse and keyboard to avoid direct contact with the metal casing.

The second is a software-based defense strategy. The principle behind the implementation of *LeakThief* is the effect of different commands on the leakage current, which can be used to identify the user's behavior. When the laptop is performing complex calculations (e.g. training a neural network), the leakage current is unable to extract or identify the user's behavior. Therefore, we can add random noise to the leakage current by randomly modulating the CPU, thus interfering with the behavior stealing of *LeakThief*.

b) Limitation: We verified the application recognition performance of *LeakThief* at different sampling rates in experiment. VI-C1. Similar to built-in sensors such as accelerometers [3], commercially available wearable devices (such as smart watches and bracelets) offer only low sampling rates (below $1KHz$) for electrodes to achieve tasks such as ECG monitoring. The researchers [36] have increased the sampling rate of the accelerometer to $4KHz$ by customizing the smart watch

kernel. In contrast to the accelerometer, the electrodes do not have the limitations of the sampling rate and devices such as smart watches are capable of sampling rates of $48KHz$ [37]. Therefore, we believe that developers can increase the sampling rate of the electrodes to $48KHz$ by adjusting the kernel of the wearable device, thus to meet the sampling rate requirements of the behavior stealing based on leakage current.

VIII. Conclusion

This paper presents the potential threat of information leakage via the leakage current of laptops and the electrode in wearable devices. The proposed *LeakThief* uses the built-in electrode to collect the leakage current flowing through the user's body when using a laptop with a metal casing, thus to steal the behavior information. The system incorporates detection of leakage current, detection of application operations and application recognition. Experiments demonstrate the feasibility of *LeakThief* for behavior stealing, enabling application recognition based on application launching (97.5%) and in-application operation (83.8%). Considering the consumer interest in laptops with metal casing such as MacBook and the increasing popularity of wearable devices such as smart watches and bracelets, we believe that behavior stealing based on the leakage current will be universal and non-negligible.

Acknowledgment

This work is supported in part by the NSFC (62072306, 61936015) and Program of Shanghai Academic Research Leader (20XD1402100).

References

[1] F. number of mobile devices worldwide from 2020 to 2025, 2023. [Online]. Available: https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide/

[2] B. Nassi, Y. Pirutin, A. Shamir, Y. Elovici, and B. Zadov, "Lamphone: Passive sound recovery from a desk lamp's light bulb vibrations," in *USENIX Security 22*. Boston, MA: USENIX Association, Aug. 2022.

[3] S. A. Anand, C. Wang, J. Liu, N. Saxena, and Y. Chen, "Spearphone: A lightweight speech privacy exploit via accelerometer-sensed reverberations from smartphone loudspeakers," ser. WiSec '21, p. 288–299.

[4] D. Ding, L. Yang, Y.-C. Chen, and G. Xue, "Handwriting recognition system leveraging vibration signal on smartphones," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

[5] J.-X. Bai, B. Liu, and L. Song, *I Know Your Keyboard Input: A Robust Keystroke Eavesdropper Based-on Acoustic Signals*, New York, NY, USA, 2021, p. 1239–1247.

[6] H. Yin, A. Zhou, G. Su, B. Chen, L. Liu, and H. Ma, "Learning to recognize handwriting input with acoustic features," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 2, jun 2020.

[7] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, "Keystroke recognition using wifi signals," ser. MobiCom '15, 2015, p. 90–102.

[8] B. Chen, V. Yenamandra, and K. Srinivasan, "Tracking keystrokes using wireless signals," ser. MobiSys '15, 2015, p. 31–44.

[9] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of wifi signal based human activity recognition," ser. MobiCom '15, 2015, p. 65–76.

[10] H. Pan, L. Yang, H. Li, C.-W. You, X. Ji, Y.-C. Chen, Z. Hu, and G. Xue, "Magthief: Stealing private app usage data on mobile devices via built-in magnetometer," in *2021 18th SECON*, 2021, pp. 1–9.

[11] Y. Cheng, X. Ji, W. Xu, H. Pan, Z. Zhu, C.-W. You, Y.-C. Chen, and L. Qiu, "Magattack: Guessing application launching and operation via smartphone," ser. Asia CCS '19, New York, NY, USA, 2019, p. 283–294.

[12] Q. Xia, F. Hong, Y. Feng, and Z. Guo, "Motionhacker: Motion sensor based eavesdropping on handwriting via smartwatch," in *INFOCOM WKSHPS*, 2018, pp. 468–473.

[13] H. Jiang, "Motion eavesdropper: Smartwatch-based handwriting recognition using deep learning," ser. ICMI '19, 2019, p. 145–153.

[14] U. Meteriz-Yıldıran, N. F. Yildiran, and D. Mohaisen, "Sia: Smartwatch-enabled inference attacks on physical keyboards using acoustic signals," ser. WPES '21, 2021, p. 209–221.

[15] A. W. S. 7, 2021. [Online]. Available: https://www.apple.com/apple-watch-series-7/

[16] L. Yang, Y. C. Chen, H. Pan, D. Ding, G. Xue, L. Kong, J. Yu, and M. Li, "Magprint: Deep learning based user fingerprinting using electromagnetic signals," in *IEEE INFOCOM 2020*, 2020, pp. 696–705.

[17] M. M. Jha, K. B. Naik, and S. P. Das, "Estimation of optimum value of y-capacitor for reducing emi in switch mode power supplies," *Electrical Power Quality and Utilisation. Journal*, 2009.

[18] D. Ding, L. Yang, Y.-C. Chen, and G. Xue, "Leakage or identification: Behavior-irrelevant user identification leveraging leakage current on laptops," *PACM IMWUT.*, vol. 5, no. 4, dec 2022.

[19] W. Meng, "Touch current analysis for power supplies designed for energy efficient regulations," in *2011 IEEE Symposium on Product Compliance Engineering Proceedings*, 2011, pp. 1–6.

[20] Z. Yan, Q. Song, R. Tan, Y. Li, and A. W. K. Kong, "Towards touch-to-access device authentication using induced body electric potentials," ser. MobiCom '19, New York, NY, USA, 2019.

[21] D. AD2, 2021. [Online]. Available: https://digilent.com/shop/analog-discovery-2-100ms-s-usb-oscilloscope-logic-analyzer-and-variable-power-supply/

[22] V. Nguyen, S. Rupavatharam, L. Liu, R. Howard, and M. Gruteser, "Handsense: Capacitive coupling-based dynamic, micro finger gesture recognition," ser. SenSys '19, New York, NY, USA, 2019, p. 285–297.

[23] D. J. Matthies, C. Weerasinghe, B. Urban, and S. Nanayakkara, "Capglasses: Untethered capacitive sensing with smart glasses," ser. AHs'21, New York, NY, USA, 2021, p. 121–130.

[24] D. Ding, L. Yang, Y. C. Chen, and G. Xue, "Vibwriter: Handwriting recognition system based on vibration signal," in *IEEE SECON*, 2021.

[25] M. Enev, S. Gupta, T. Kohno, and S. N. Patel, "Televisions, video privacy, and powerline electromagnetic interference," ser. CCS '11, New York, NY, USA, 2011, p. 537–550.

[26] D. Genkin, I. Pipman, and E. Tromer, "Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs," Cryptology ePrint Archive, Paper 2014/626, 2014, https://eprint.iacr.org/2014/626.

[27] S. Westerlund and L. Ekstam, "Capacitor theory," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 1, no. 5, pp. 826–839, 1994.

[28] C. J. Yang and A. P. Sample, "Em-comm: Touch-based communication via modulated electromagnetic emissions," *PACM IMWUT.*, vol. 1, no. 3, Sep. 2017.

[29] H. Pan, Y.-C. Chen, G. Xue, and X. Ji, "Magnecomm: Magnetometer-based near-field communication," ser. MobiCom '17, New York, NY, USA, 2017, p. 167–179.

[30] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *CVPR*, July 2017.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, June 2016.

[32] Y. Chen, W. Gong, J. Liu, and Y. Cui, "Fine-grained ultrasound range finding for mobile devices: Sensing way beyond the 24 khz limit of built-in microphones," in *IEEE INFOCOM WKSHPS*, 2017, pp. 845–850.

[33] C. Holz and M. Knaust, "Biometric touch sensing: Seamlessly augmenting each touch with continuous authentication," ser. UIST '15, New York, NY, USA, 2015, p. 303–312.

[34] E. J. Wang, J. Garrison, E. Whitmire, M. Goel, and S. Patel, "Carpacio: Repurposing capacitive sensors to distinguish driver and passenger touches on in-vehicle screens," ser. UIST '17, 2017, p. 49–55.

[35] L. Yang, H. Li, Z. Chen, X. Ji, Y.-C. Chen, G. Xue, and C.-W. You, "Appliance fingerprinting using sound from power supply," ser. UbiComp-ISWC '20, New York, NY, USA, 2020, p. 160–163.

[36] G. Laput, R. Xiao, and C. Harrison, "Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers," ser. UIST '16, New York, NY, USA, 2016, p. 321–333.

[37] G. Reyes, D. Zhang, S. Ghosh, P. Shah, J. Wu, A. Parnami, B. Bercik, T. Starner, G. D. Abowd, and W. K. Edwards, "Whoosh: Non-voice acoustics for low-cost, hands-free, and rapid input on smartwatches," ser. ISWC '16, New York, NY, USA, 2016, p. 120–127.