Trustworthy IAP: An Intelligent Applications Profiler to Investigate Vulnerabilities of Consumer Electronic Devices

Jie Su, Zhen Hong, Lei Ye, Tao Liu, Sizhuang Liang, Shouling Ji, *Member, IEEE*, Gagangeet Singh Aujla, *Senior Member, IEEE*, Reheem Beyah, *Senior Member, IEEE*, and Zhenyu Wen, *Senior Member, IEEE*

***Abstract*—As a typical representative of the Internet of Energy (IoE) intelligent era, consumer electronic (CE) devices continue to evolve at a remarkable pace. Computers, as typical and essential CE devices, have been instrumental in enhancing efficiency, communication, entertainment, and information access. As part of this evolution, a significant trend in computer design focuses on achieving low power consumption while maintaining high performance. For instance, a computer's central processing unit (CPU) dynamically modulates its output power in response to the varying workload demands of running applications. However, these power efficiency mechanisms may inadvertently introduce implicit patterns into the operational states of CE devices. Particularly, the power consumption of a CE device executing various tasks can manifest distinguishable temporal patterns, thereby exposing potential vulnerabilities. Thus, this work aims to investigate the vulnerabilities of CE devices on power consumption mechanisms. We focus on exploring the possibility of using alternating current (AC) power consumption to infer the running applications on a consumer computer. To achieve that, we construct a physical attack system that employs data acquisition, processing, classification, and inference stages to establish a "profiler" for application profiling. The extensive experiment results on the self-collected power consumption dataset (36 applications) demonstrate the effectiveness of the attacking system.**

***Index Terms*—Resilience, deep learning, consumer electronics vulnerabilities.**

I. Introduction

THE CONSUMER electronics industry has witnessed rapid advancements in recent years, propelled by innovations in technology and an increasing demand for smart and connected devices. With increasing global consciousness towards energy consumption, consumers demand devices that maximize operational efficiency. To address this intricate demand, manufacturers have ingeniously integrated dynamic adjustability into CE devices. Such a strategy allows these devices to flexibly adapt their workloads, thereby minimizing energy consumption during less strenuous operations and efficiently managing heat production. However, this power management mechanism may inadvertently introduce implicit patterns into the operational states of CE devices. For instance, during the startup stages, different applications (e.g., Chrome, Photoshop, etc.) initiate a variety of operations (e.g., loading preferences, user interface rendering, etc.). The varying operational objectives entail differing levels of complexity, which in turn manifest as distinct power consumption patterns, as highlighted by the arrows in Figure 1. These patterns enable potential attackers to profile users based on their application usage, such as determining when a user typically checks their email, engages in social media activity, or uses financial management tools. This profiling can lead to inferences about a user's daily routine, interests, and even confidential behaviors, which could be exploited for targeted advertising, social engineering attacks, or privacy breaches. Under such circumstances, investigating methods to infer private information from alternating current (AC) power has become an attractive research question. Identifying such vulnerabilities can, in turn, stimulate the development of robust countermeasures.

The side-channel attack refers to the exploitation of indirect information (e.g., power consumption or electromagnetic emissions) leaked from computing devices to reveal confidential data, and has become a promising approach in the landscape of cybersecurity threats [1], [2], [3], [4].

Manuscript received 27 August 2023; revised 24 November 2023; accepted 16 December 2023. Date of publication 27 December 2023; date of current version 26 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62072408; in part by the Zhejiang Provincial Science Fund for Distinguished Young Scholars under Grant LR24F020004; and in part by the Zhejiang Provincial Natural Science Foundation of Major Program (Youth Original Project) under Grant LDQ24F020001. *(Corresponding authors: Zhen Hong; Zhenyu Wen.)*

Jie Su and Zhenyu Wen are with the Institute of Cyberspace Security and the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, Zhejiang, China, also with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230052, Anhui, China, and also with the Internet of Things Application and Security Research Center, Binjiang Cyberspace Security Institute of ZJUT, Hangzhou 310052, China (e-mail: jiesu@zjut.edu.cn; wenluke427@gmail.com).

Zhen Hong is with the Institute of Cyberspace Security and the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, Zhejiang, China, and also with the Binjiang Cyberspace Security Institute of ZJUT, Hangzhou 310052, China (e-mail: zhong1983@zjut.edu.cn).

Lei Ye and Tao Liu are with the School of Engineering, Zhejiang Sci-Tech University, Hangzhou 311800, Zhejiang, China (e-mail: 563324801@qq.com; lt407892960@163.com).

Sizhuang Liang and Reheem Beyah are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: liangsizhuang@gatech.edu; rbeyahece.gatech.edu).

Shouling Ji is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310023, Zhejiang, China (e-mail: sji@zju.edu.cn).

Gagangeet Singh Aujla is with the Department of Computer Science, Durham University, DH1 3LE Durham, U.K. (e-mail: gagangeet.s.aujla@durham.ac.uk).

Digital Object Identifier 10.1109/TCE.2023.3347651

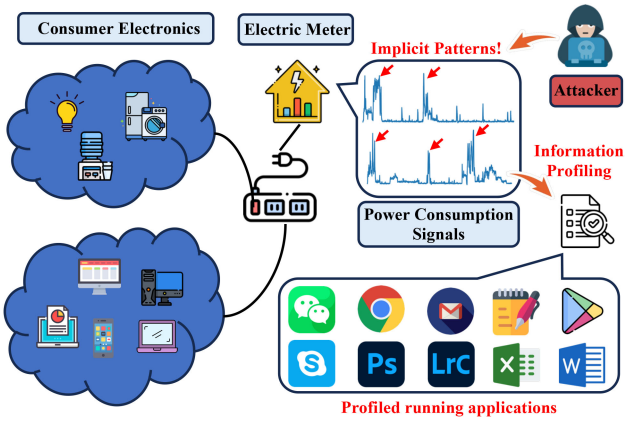


Fig. 1. **Motivation.** The implicit patterns emerging from the power adjustment mechanisms in consumer electronic devices potentially present serious vulnerabilities that could be exploited by attackers for information theft.

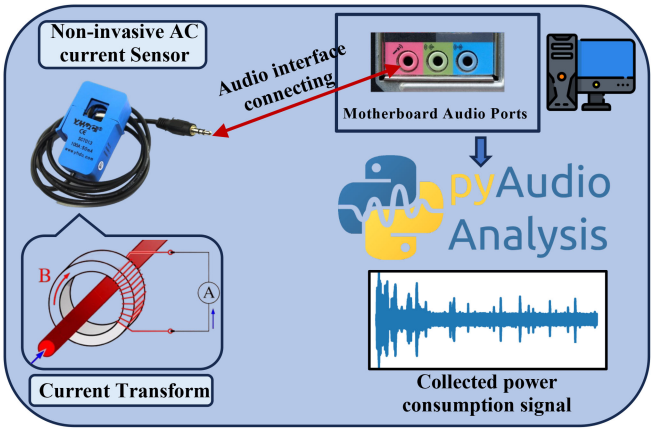


Fig. 2. Current Transformer and its audio interface.

Previous studies [5], [6], [7] have demonstrated the potential for adversaries to extract private information from personal computers (PCs) through side-channel attacks, exploiting hardware components including I/O peripherals, keyboards, mice, and internal fans. Recent studies [8], [9] have started to analyze the energy consumption information so as to infer the electrical appliances usage and surfed websites. However, the patterns of running applications are inherently more complex due to the diversity in software complexity and various optimization strategies making profiling/attack more challenging. Specifically, the **diverse execution times** of applications pose a significant challenge for the attacker in determining a suitable analysis window for consistent signal pattern extraction. Moreover, **noise interference** originating from either the device or the operating system introduces instability in profiling.

To address these challenges, we propose a side-channel-based intelligent profiling system that aims to leverage the power consumption analysis to continuously infer the running applications on a CE device (i.e., PC). Specifically, We have developed an automated workflow for signal collection to amass power consumption data from 36 applications operating across various systems and computer platforms. Utilizing a Current Transformer (CT), we gathered 150 records for each application, culminating in the first dataset of power consumption specific to PC applications. As shown in Fig. 2, the neutral wire of the power line is passed through the opening in a CT, and the other end of the CT is a 3.5mm audio interface. By attaching the 3.5mm audio interface to the attacker device, the continuous analog current signal can be obtained for analysis.

Subsequently, we introduce a novel stage segmentation mechanism to automatically separate the running stage of the applications, presenting discriminative patterns. By segmenting and simultaneously collecting signals, we are able to capture high-quality signals from the startup stage, which are essential for advanced pattern learning. Additionally, a classifier based on advanced deep neural networks is introduced to find and fit the pattern of the running applications for future attacking inference. Extensive experiments were conducted, and the promising results suggest the effectiveness of our approach. In addition to verifying the effectiveness of our solution, we also discuss the limitations of such attacks. The main contributions of this paper can be summarized as follows.

- To the best of our knowledge, this is the first attempt to perform application profiling on the CE domain, which aims to steal sensitive information from a consumer computer via AC power consumption analysis. Our proposed profiling system can accurately infer candidate applications by analyzing unsegmented power consumption traces. It can continuously analyze AC power signals to infer the privacy information, e.g., which application is currently running, and how frequently it is used.
- We design a set of attack scenarios to evaluate the *effectiveness* of our profiling system. The results show that our scheme achieves high accuracy with a mean average precision of 76.69%. Moreover, our attack scenarios are highly feasible and can be applied in a realistic environment.
- We demonstrate the *robustness* of our attack under various computing environments. Our attack results showcase a high and stable side-channel signal inference rate across different computer brands, operating systems, and background noise conditions (e.g., Gaussian noise).

II. Related Work

Power analysis attacks, known as power side-channel attacks [17], have emerged as a significant threat in recent years. The concept dates back to 1989 when Hart et al. [10] introduced non-intrusive load monitoring (NILM), laying the groundwork for activity identification through AC power measurement. Building on this, Kocher et al. [18], [19] pioneered the application of simple and differential power analysis attacks by measuring AC power fluctuations. Schmidt et al. [13] detected voltage leakages on embedded device I/O pins, yet the correlation with basic multiplication operations remained unclear. Messerges et al. [11], [12] studied simple power analysis (SPA) and differential power analysis (DPA) attacks against data encryption standard (DES).

TABLE I
RELATED WORKS ON POWER ANALYSIS ATTACK

Works	Side Channel	Objective	Vulnerability
Hart *et al.* [10]	Utility Power Flows	Residential Energy Monitoring and Computerized Surveillance	Power Supply
Messerges et al. [11], [12]	Power Signal	Against Smart Card's Data Encryption Standard	Power Supply
Schmidt *et al.* [13]	Voltage Variations	PC's I/O port Manipulatetion	I/O Pin
Yan *et al.* [14]	Power Comsumption Flows	App Identification, UI Inference, Password Length Guessing, Geo-location Estimation	Power Supply
Clark *et al.* [8], [15]	AC Power Consumption	Webpage Loading Inference	Power Supply
Michalevsky *et al.* [16]	Mobile Power Consumption	Geo-location Estimation	Power Supply

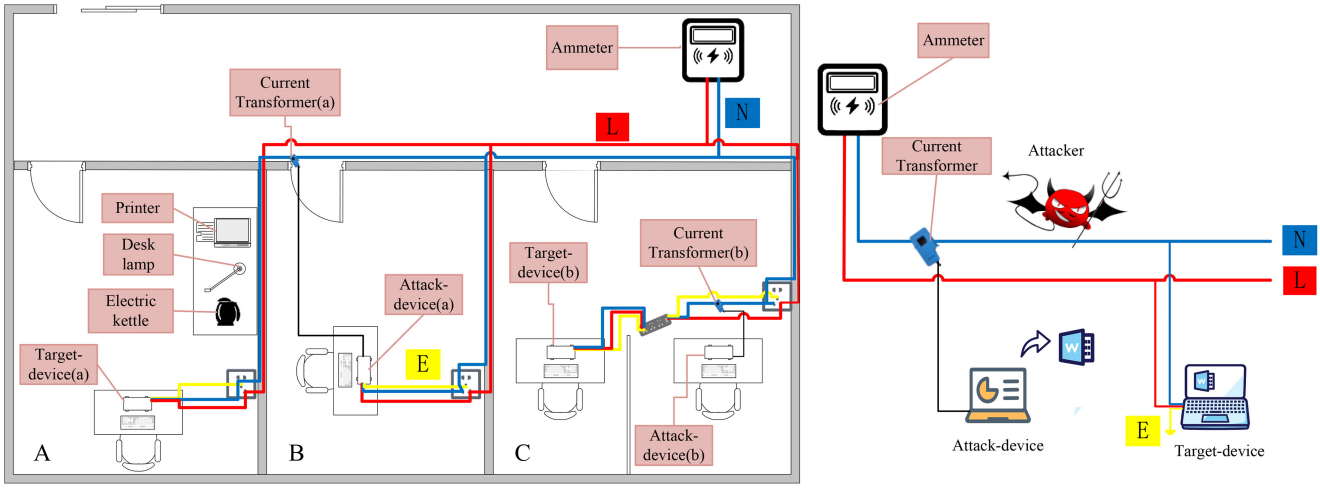


Fig. 3. Threat model. In the network between the victim and the attacker, the side channel attacker starts some applications on the target device. Starting the target application causes power fluctuation from the target device's power line, which is faithfully transmitted to the attack device by the transformer. The attacker is to infer what application the user is running by taking advantage of these emanations.

By studying the noise characteristics of power signals, a method of building the Signal-to-Noise-Ratio (SNR) model is proposed, and the security of the smart card is destroyed by the proposed multi-bit attack based on SNR. Clark et al. [8], [15] advanced the field by recording power consumption through electrical outlets and analyzing the frequency domain power trajectory, using SVM classifiers to identify webpages loaded by browsers with increased precision. Nevertheless, this method necessitated data pre-segmentation and relied on an impractically altered electrical outlet, posing risks of physical damage. Michalevsky et al. [16] and Yan et al. [14] demonstrated that power tracking could act as an effective side channel to glean sensitive information from smartphone applications, but their approach fell short in inferring real-time application loads or mounting viable attacks in practical settings. Up to now, there has been a gap in research regarding the vulnerabilities of consumer electronics (CE) devices (e.g., computers) related to power consumption mechanisms, with most existing methods proving non-practical in real-world scenarios (e.g., non-practical scenes or laboratory environment). Table I summarizes and compares the existing works on power analysis attacks.

III. THREAT MODEL

Our side-channel attack are based on the assumption that only the victim's device (i.e., computer) is plugged into the adapter. Under such assumption, the attacker can attach a current transformer sensor to the neutral line of the adapter to read the energy consumption records. Then, it is possible to infer the running applications on a computer through the analysis of energy consumption patterns. The threat model depicted in Fig. 3, can be performed in the following situations:

Situation 1: There are two rooms, A and B. We assume that the victim user is situated in room A, while a malicious individual resides in room B. The attacker aims to identify the applications running on the user's computer. To accomplish this, the attacker places a current transformer on the Neutral wire and collect the current data of the user's computer to themselves. The attacker measures the AC signal on the patch panel to acquire sensitive information from the target device and subsequently infers the active applications.

Situation 2: Two users share a patch panel in room C, with one user (b) acting as the attacker. The attacker (b) positions the current transformer (b) on the Neutral wire of the outlets, enabling the acquisition of the AC power consumption data from the target user's computer (target-device (b)). Subsequently, the attacker can deduce the running applications on the target device (b) by analyzing the collected data.

A. Attack Scenarios

Based on the aforementioned threat model, we conduct the following attack scenarios (as illustrated in Fig. 4).

Known Device Profiling: Aims to infer the running application of seen or known devices. Under this scenario, the attacker can obtain the power consumption signals from target devices through invasive or non-invasive methods. Specifically, we refer to the invasive and non-invasive methods as *Signal Stealing* and *Signal Imitation*, respectively. Under the *Signal Stealing* condition, we assume that the attacker accidentally acquires the information about the running application

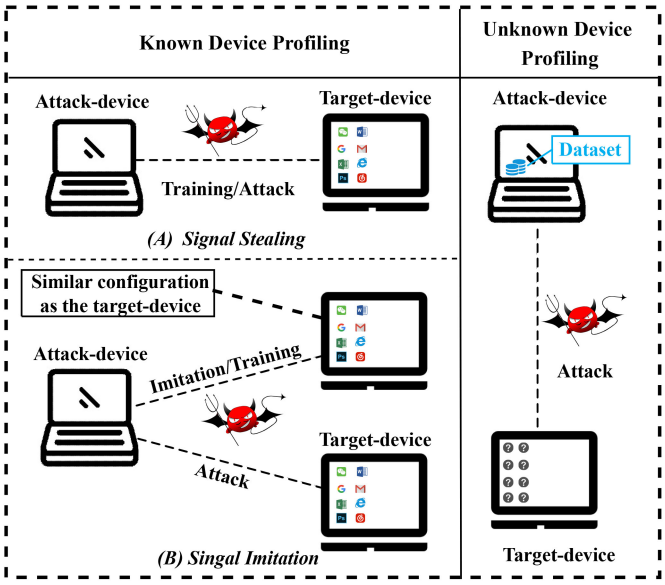


Fig. 4. Attack scenarios.

(e.g., through chat or peeping) on target devices. In contrast, *Signal Imitation* represents a situation where the attacker uses a similar device (e.g., with the same brand, operating system, and hardware configuration) to collect the power consumption signal of various applications.

Unknown Device Profiling: Aims to infer the running applications of unseen or unknown devices. In this scenario, the attacker can only obtain power consumption signals from the target devices, without any supplementary information. This setting is more representative of real-world situations and presents greater challenges for attackers. We assume that the attacker has constructed a power database of certain devices from previous attacks. Consequently, the attacker can first analyze the type of the target device based on the power database when they receive the power signal, and then infer the running application of the victim.

IV. Side-Channel Profiling

A. Attack Overview

This section presents the details of our proposed side-channel attack model for inferring running applications. Fig. 5 illustrates the high-level pipeline for executing the side-channel attack. To differentiate between various applications, we train a classifier using labeled AC power traces. Once trained, the model can infer the running applications on the target device in real-time. During the inference stage, we input the signal to the model at 10 frames per second (10FPS) with a specified window length. The trained model provides predictions and confidence scores for each frame. Predictions with confidence scores exceeding the threshold are accepted.

B. Data Acquisition

For the experiment usage, we employ a current transformer to collect power consumption data from laptops and desktops of various brands and operating systems. By monitoring the instantaneous current on the socket connected to the computer charger, we record the power signature for each workload. The current sensor is connected to both the attack device and the socket, sampling the AC power signal on the socket and collecting it through the audio interface of the attack device (excluding the computer being measured). The data collection process is automated, outputting the leakage information as separate sound files for training purposes.

C. Data Processing

To better capture the patterns hidden in the power consumption signals, we propose a **stage segmentation** approach to construct input signals with effective power characteristics. The AC power curve exhibits three patterns corresponding to three events: ① Starting, ② Waiting, and ③ Stopping. During the starting stage, the application undergoes initialization, which includes multiple operations such as software path access, registry reading, and memory allocation. Consequently, the computer's power consumption increases in this stage. At the end of the startup process, the application enters the waiting stage for further operations, causing the power consumption to decrease to a stable level. In the application stopping stage, the computer saves the registry and clears the memory space, resulting in temporary power consumption fluctuations. Therefore, we divide the complete running process into three states: start, wait, and stop, as illustrated in Fig. 6.

Upon observing the collected power consumption data, we noticed that applications exhibit significant differences in the starting stage (i.e., initialization). For instance, the power curves of nine distinct applications during startup are displayed in Fig. 7, demonstrating clear discrepancies between them (Additional examples can be found in Fig. 8.). To effectively capture the patterns from the startup states, we develop a stage segmentation approach that automatically segments the starting status signals. Fig. 9 demonstrate the workflow of the segmentation operation. Specifically, we propose to use the short-term energy as an effective means to preserve the details and accurately locate the starting point of an application. The short-term energy measures the continuous energy consumption over a certain period. We choose one cycle of the sinusoidal AC current (i.e., 0.02s) to record the short-term current energy, which provides more fine-grained signals for further detection. Since the audio sampling rate is 44,100 Hz, the data points covered by the short-term current energy are 882 (0.02s $\times$ 44,100 Hz). When calculating the short-term energy, the amplitude of the AC signals is first normalized into a range of -3 to 3. Then, the short-term energy consumption over 882 points is summed up. Finally, the energy consumed during the window that exceeds a certain threshold (i.e., Avg. short-term energy value) is considered the effective part of the startup curve.

D. Data Classification

The data classification process comprises device classification and application classification. Device classification is designed for the 'Unknown Device Profiling' scenario. When the target device is unknown, the framework initially utilizes

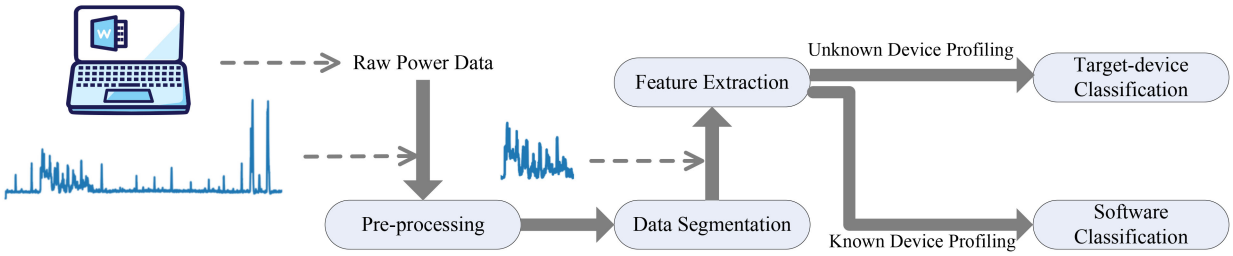


Fig. 5. Training a deep learning model to analyze the AC power consumption data.

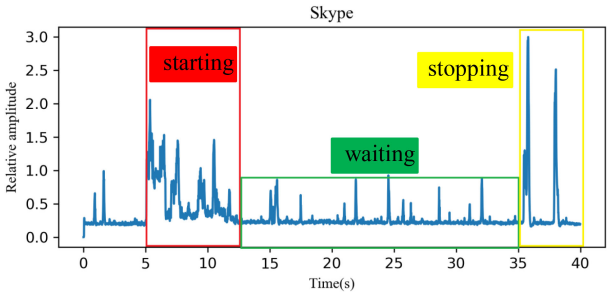


Fig. 6. There are three different forms of AC power curve in the process of starting application: (i) starting; (ii) waiting; (iii) stopping.

device classification to infer the device type using the existing device database. Subsequently, application inference is carried out based on the identified device type.

1) Application Classification: We formulate the application classification as a multi-classes classification task. Specifically, we select two traditional machine learning models, namely Pearson Correlation-based classifier [20] and Dynamic Time Warping (DTW) [21] based classifier. DTW allows for more accurate comparisons between time series data with varying lengths or time distortions, enhancing the classification accuracy by providing a more meaningful similarity metric. These traditional methods offer advantages such as low computational complexity and ease of implementation. On the other hand, we also choose two deep learning models, namely 1D-CNN [22] and 2D-CNN [23]. The deep learning models provide benefits like the automatic extraction of complex features and improved classification accuracy, especially when dealing with large and high-dimensional data.

To identify a suitable classifier for application classification, we conduct experiments on a balanced dataset consisting of 36 application classes, with 10 samples per class. The evaluation metric is top-N accuracy, where an application is deemed correctly classified if it appears in the top N guesses of a classifier. To adapt the input for 2D-CNN models, we convert the power consumption signal into a gray-scale image. The final results, displayed in Fig. 10, indicate that the 2D-CNN model outperforms the other models under the top-N accuracy evaluation protocol with various N choices. Consequently, the 2D-CNN model is employed for subsequent experimental classifications.

2) Device Classification: We formulate the target device classification as a multi-class classification problem, with different classes representing distinct types of target devices in the database. Specifically, for device classification, the k-NN algorithm [24] is employed due to its simplicity, ease of implementation, and effectiveness in handling high-dimensional data. The k-NN approach can identify patterns in the device database by considering the similarity of power consumption signals. Specifically, we use DTW as the distance measurement for the k-NN algorithm. By finding the most similar devices in the database using DTW, the algorithm classifies unknown devices based on their proximity to known devices, making it particularly useful for handling large and diverse datasets. Once the device has been classified, the aforementioned application classification is performed to infer the running application.

V. Evaluation

To evaluate the performance of the side-channel attack based on power analysis, we conduct a series of experiments under two scenarios that are mentioned in the previous section.

A. Device Configuration

Fig. 11 illustrates the configuration of power consumption signal collection. The attacker connects the current sensor (with the other end connected to the audio interface in the attacker's computer) to the terminal in the AC socket to collect power consumption signals. Modern AC sockets have three terminals: Live (L) wire, Neutral (N) wire, and Earth (E) wire. Since the current flows from the Live (L) wire to the equipment and forms a loop at the Neutral (N) wire when the electrical equipment is operating, the current sensor is attached to the Neutral (N) wire to collect data.

B. Data Collection Configuration

1) Application Selection: We designed a questionnaire to gather information about the most frequently used applications. A total of 356 questionnaires were collected through the WeChat protocol (a popular communication platform in China), with participants from diverse occupations, including both students and office workers. Based on the responses, we selected the 36 most common applications for our experiments. The usage distribution of each application is shown in Fig. 12.

2) Application Data Collection: We collect the AC power traces of each selected application from computers with different brands and different operating systems (as shown in Table II). In the experiment, 12-bit samples are recorded at a rate of 44100Hz to capture the AC power trace of the high-frequency workload. The collected AC power consumption

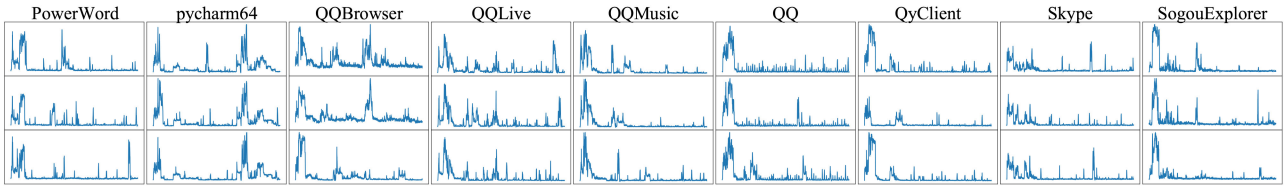


Fig. 7. Power curves for several applications during startup. For a specific application, the power curves during initialization are not exactly the same but similar; whereas for different applications, they start at different speeds and duration. Therefore, the power curves during initialization are different.

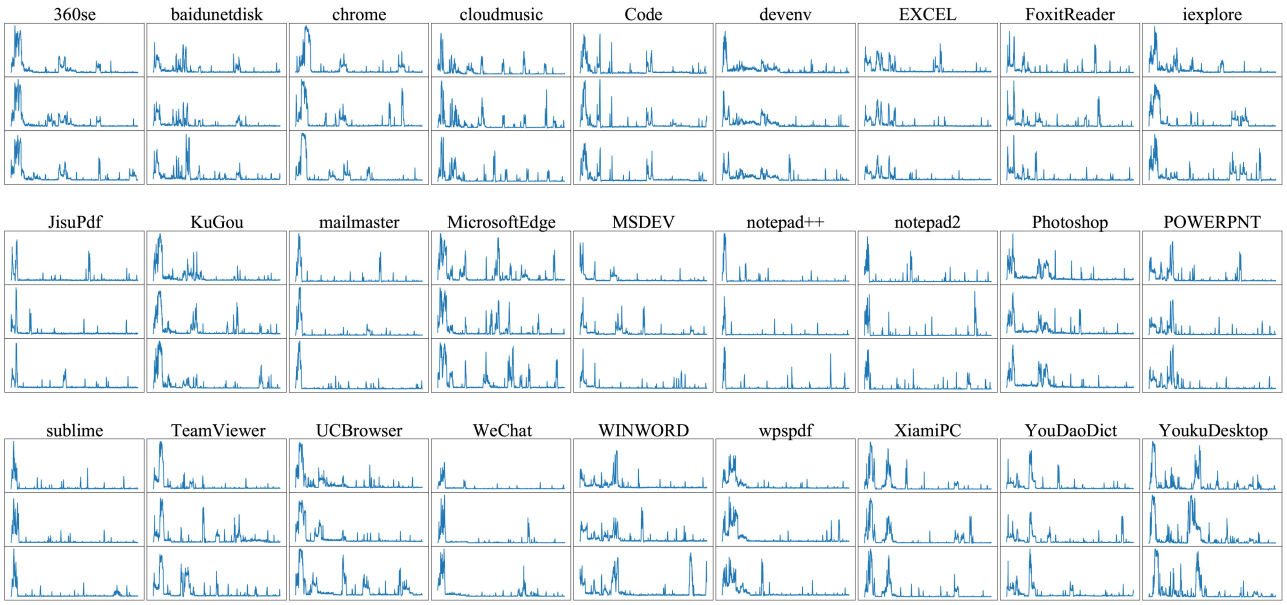


Fig. 8. Power curves for some kinds of applications during startup.

TABLE II
COMPUTERS USED IN THE EXPERIMENTS

Device	Type	System	fs (Hz)
Lenovo ThinkPad R480	Laptop	Windows 10	44100
Lonovo Legion Y7000	Laptop	Windows 10	44100
ASUS VM480L	Laptop	Windows 10	44100
ASUS ATC705-N91	Desktop	Ubuntu 16.04	44100

dataset is available on IEEE Dataport (DOI: 10.21227/v2vf-3r56) and can be downloaded using this footnote link.[1] Fig. 13 demonstrates the workflow of the automatic collection process via bash scripts.

At the beginning of the recording, the attack device sends TCP/IP packets to make the target device connect to a local area network. When a successful connection is established, the attack device sends instructions to control the operation of the applications on the target device. After receiving the "start" instruction, the target device waits for 5s to execute the automatic initialization program and then performs the waiting and closing operation (35 seconds). Recording (using Pyaudio) is stopped after 5 seconds of application closing, resulting in a 40-second record. For each application, we collect 150 records, with a 5-second gap between two recordings. The power curves of 36 applications are measured on each computer, as shown in Table III.

[1]https://ieee-dataport.org/documents/ac-power-consumption-dataset

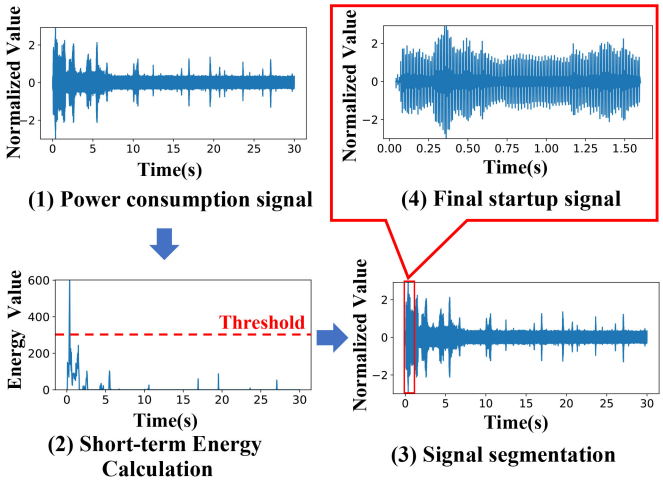


Fig. 9. Data segmentation process.

C. Inference Protocol

For the inference process, we simulate the same configuration as the real-world scenario where the signals are coming

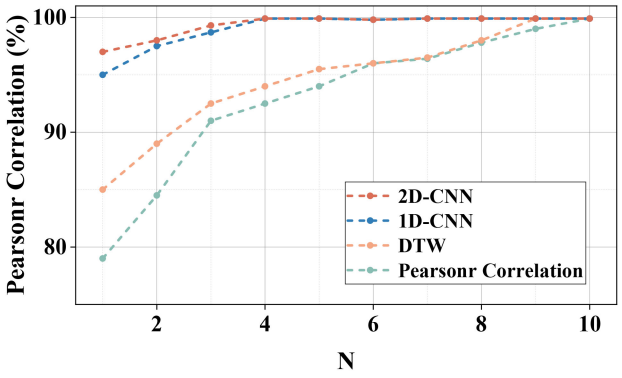


Fig. 10. Average top-n accuracy of a single application, as a function of the number of guesses, for each classifier.

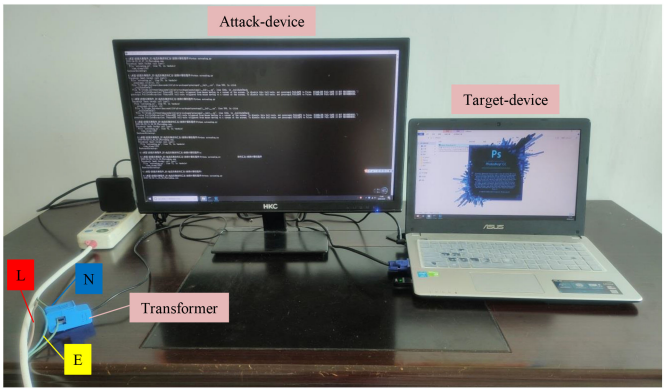


Fig. 11. Collection device configuration.

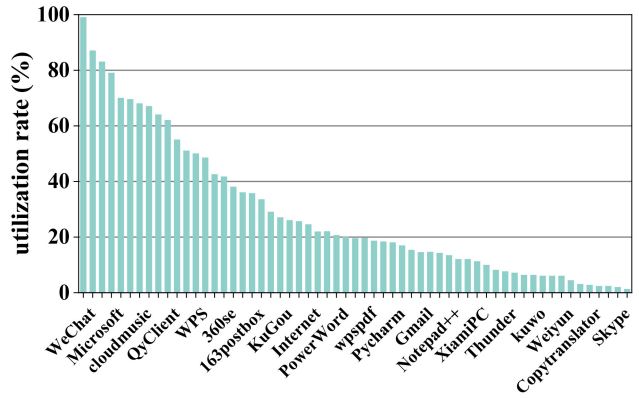


Fig. 12. Statistics of applications usage.

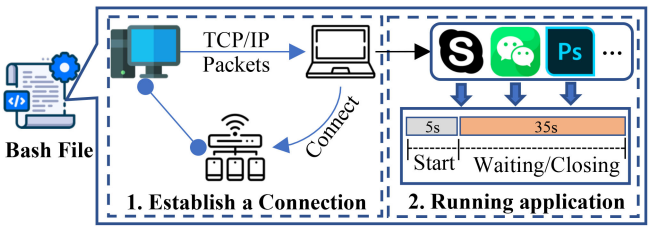


Fig. 13. Automatic collection process of applications.

continuously. Specifically, the continuous incoming signals are first processed into signal segments using a sliding window. Then, the segmented signals are converted to gray-scale images for the final inference. The inference results with a confidence level exceeding a specified threshold (i.e., 0.9) will be accepted. Fig. 14 presents example inference results for Skype and Photoshop applications.

TABLE III
APPLICATIONS COLLECTED IN THE EXPERIMENTS

Application	Type	Start-up Time(s)	Collection times	AP
360se	Fast	0-5	150	0.691
chrome	Fast	0-5	150	0.845
cloudmusic	Fast	0-5	150	0.571
Code	Fast	0-5	150	0.927
FoxitReader	Fast	0-5	150	0.683
iexplore	Fast	0-5	150	0.711
JisuPdf	Fast	0-5	150	0.768
KuGou	Fast	0-5	150	0.915
mailmaster	Fast	0-5	150	0.907
MicrosoftEdge	Fast	0-5	150	0.829
MSDEV	Fast	0-5	150	0.278
notepad++	Fast	0-5	150	0.417
notepad2	Fast	0-5	150	0.804
PowerWord	Fast	0-5	150	0.955
QQ	Fast	0-5	150	0.524
QQBrowser	Fast	0-5	150	0.925
QQLive	Fast	0-5	150	0.725
QQMusic	Fast	0-5	150	0.88
Skype	Fast	0-5	150	0.422
SogouExplorer	Fast	0-5	150	0.81
sublime	Fast	0-5	150	0.956
TeamViewer	Fast	0-5	150	0.833
UCBrowser	Fast	0-5	150	0.822
WeChat	Fast	0-5	150	0.797
wpspdf	Fast	0-5	150	0.894
YoukuDesktop	Fast	0-5	150	0.751
baidunetdisk	Steady	5-10	150	0.455
EXCEL	Steady	5-10	150	0.688
Photoshop	Steady	5-10	150	0.625
POWERPNT	Steady	5-10	150	0.571
pycharm64	Steady	5-10	150	0.84
QyClient	Steady	5-10	150	0.926
WINWORD	Steady	5-10	150	0.604
XiamiPC	Steady	5-10	150	0.469
YouDaoDict	Steady	5-10	150	0.771
devenv	Steady	10-15	150	0.681

D. Evaluation Configuration

We utilize three different metrics to evaluate our classifiers:

- *mAP:* Mean Average Precision (mAP) reflects the overall performance of the classifiers. The Average Precision (AP) for each class is first calculated and then averaged across all classes. The AP for each class can be computed as the area under the Precision-Recall curve (AUCPR) [25].
- *Accuracy:* $TP/(TP + FP)$ reflects the proportion of true positive samples among positive instances and is employed to evaluate the classifier's capability to exclude negative examples.

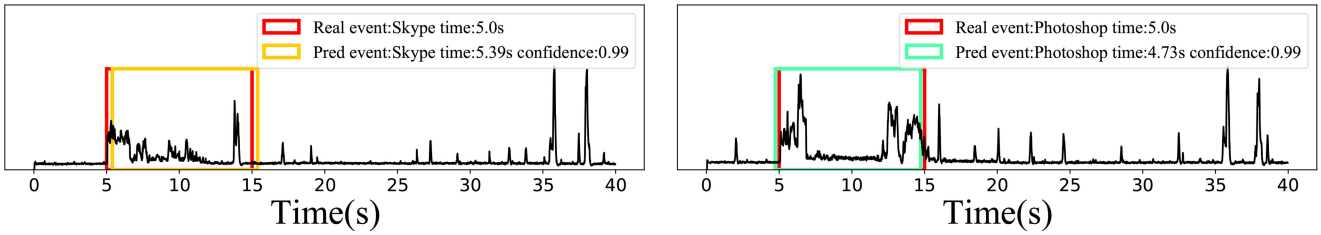


Fig. 14. Slide the check box to check the application. For each frame data, input it into the classifier for classification, and the classification result is the application label corresponding to the waveform and the application start-up time.

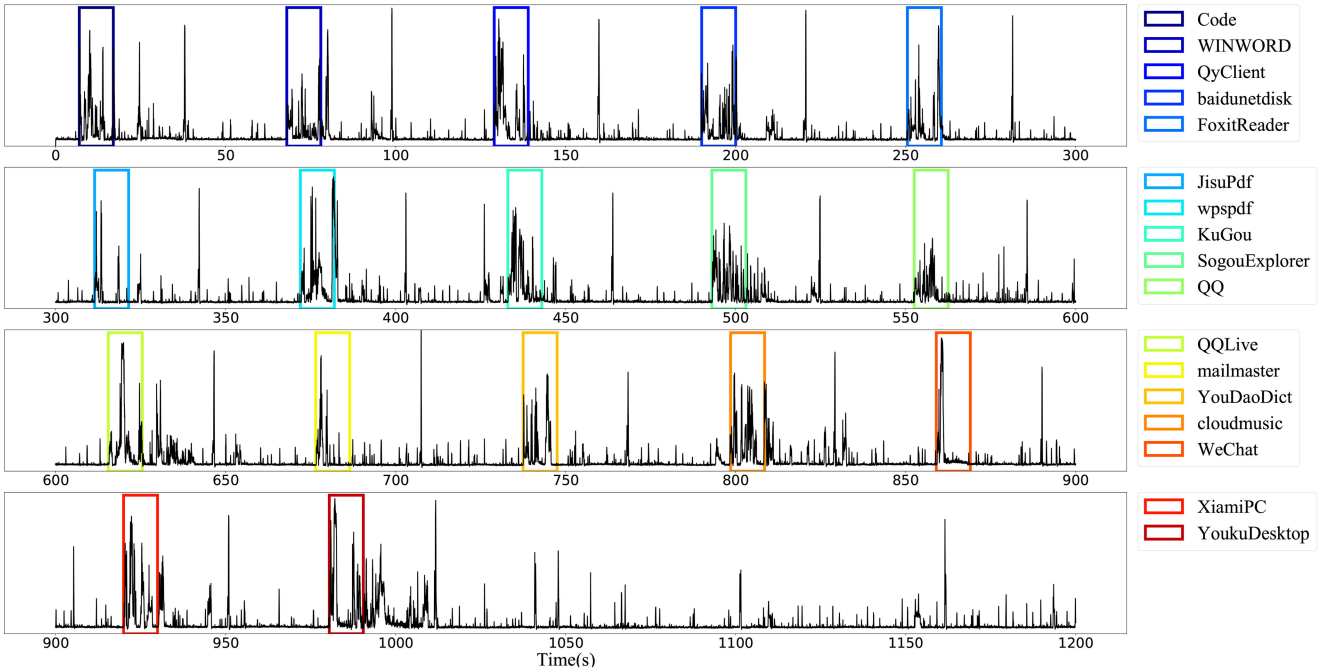


Fig. 15. The inference of a single application in a long continuous signal. The location marked by the line segment is the time when the specific application starts.

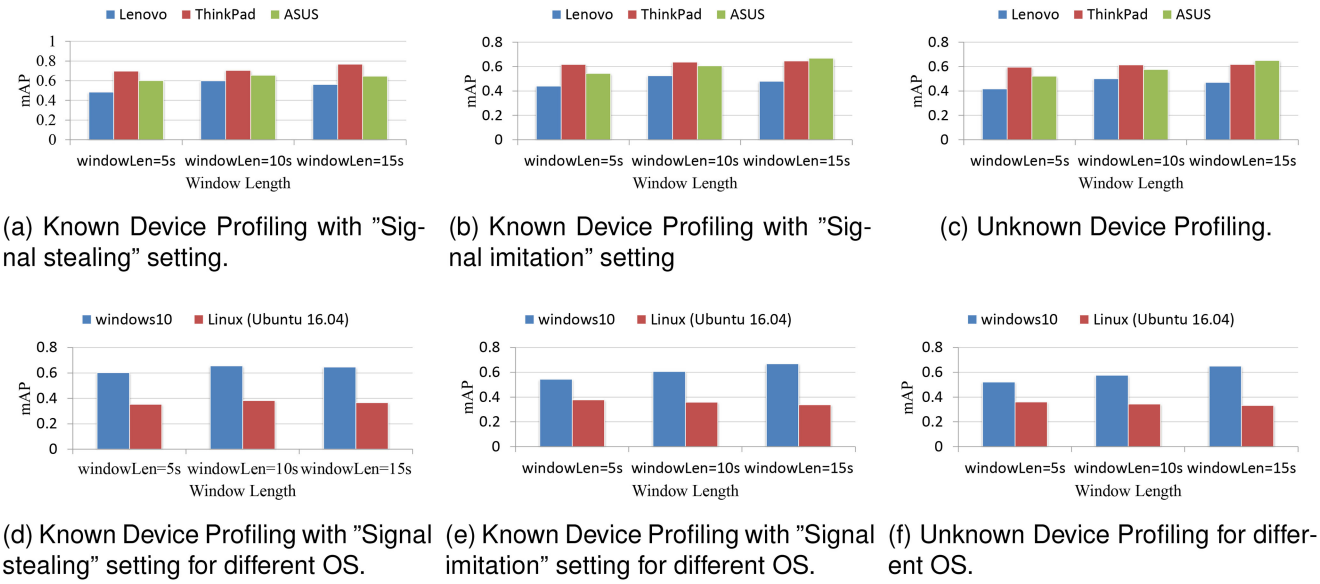
- *Recall rate:* $TP/(TP+FN)$ reflects the proportion of true positive cases identified among the total positive cases and is used to assess the classifier's ability to recognize positive instances.

VI. Result

In this section, we present the performance of AC power analysis-based side-channel attacks under known and unknown attacking scenarios (as mentioned in Section II). Specifically, we conduct evaluations on four computer models with both Windows and Linux operating systems installed. The startup processing time varies between different applications due to factors such as application size, runtime dependencies, and others. Some applications start in a very short time, only a few seconds, while others take longer to initiate the process. This variability in startup length could result in biased evaluation metrics if the classifier performs better on either fast or slow processes. Therefore, we take into account both precision and recall, and use the mAP to evaluate the performance of side-channel attacks. The mAP metric provides a more balanced and comprehensive assessment of the classifier's performance, regardless of the startup signal duration. Furthermore, the variability in start-up times contributes to differing degrees of time sensitivity in pattern matching. For instance, applications with rapid start-up times exhibit discriminative features briefly, whereas those with steadier initiation display their patterns over an extended period. To accurately reflect the performance in our evaluation, we have selected window lengths of 5, 10, and 15 seconds to capture these dynamics. The result of the inference of a single application in a long continuous signal was shown in the Fig. 15.

A. Known Device Profiling

1) Signal Stealing: Fig. 16(a) presents the experimental results under the signal stealing situation. Our attack approach demonstrates superior performance in profiling the running applications on ThinkPad laptops compared to other computer models, achieving an mAP of 69.62%, 70.35%, and 76.69% for window lengths of 5s, 10s, and 15s, respectively. These differences could stem from factors such as manufacturing quality, computer configuration, and power management under various workloads. For instance, the ASUS laptop in this experiment consumes more power when starting the same application compared to the other models.



(a) Known Device Profiling with "Signal stealing" setting.

(b) Known Device Profiling with "Signal imitation" setting

(c) Unknown Device Profiling.

(d) Known Device Profiling with "Signal stealing" setting for different OS.

(e) Known Device Profiling with "Signal imitation" setting for different OS.

(f) Unknown Device Profiling for different OS.

Fig. 16. The performance of side-channel attack in different scenarios, different operating systems and different computers.

Furthermore, Fig. 16(d) demonstrates the experimental results of profiling performance under different operating systems. We observe that the inference performance on Windows (Windows 10) system is better than that of the Linux (Ubuntu 16.04) system. Specifically, the mAP for window lengths of 5s, 10s, and 15s is around 60% for the Windows system, while it is about 40% for the Linux system. These differences could arise from variations in system timers, drivers, memory management, power management, GUI features, and performance tuning. Therefore, a potential attacker should gather as much information as possible about attacks under different operating systems.

2) Signal Imitation: In contrast to the previous situation where the number of labeled data is limited, attackers can generate sufficient "realistic" data from devices with similar configurations (e.g., brand, hardware configuration, etc.) under this scenario. We can observe that *even with adequate samples, the distribution gap between different computers leads to a decrease in attack performance*. From Fig. 16(b), we can observe that under the signal imitation setting, the application inference performance suffers from a performance drop of 1∼4% for Lenovo, ThinkPad, and ASUS laptops under different window lengths. Similarly, the overall mAP on Windows and Linux operating systems also decreases, as shown in Fig. 16(e). Although we select devices with similar configurations to generate data, data distribution differences still exist due to chip variations (e.g., different CPUs with the same model could exhibit different running frequency rates).

B. Unknown Device Profiling

In the unknown device profiling scenario, the attacker lacks information about the target device, which makes application inference more challenging. Under these circumstances, the attacker must first deduce the model of the target device and then carry out the application inference process. We assume that the attacker has already obtained an application start-up signal dataset (with labels set as device type/model) collected from various devices. The attacker continuously collects the target's power signal and outputs the most likely device model from the database. To evaluate the situation where the target device is not in the database, we remove the target device's data from the database and save it as independent experimental data. From Fig. 16(c), we observe that the inference performance is similar to known device profiling under the signal imitation setting, with a slight performance drop. Similarly, the inference performance (shown in Fig. 16(f)) on different operating systems also experiences a slight decrease compared to the previous setting. Although there is a performance drop in this challenging setting, it still demonstrates its ability to correctly infer the running application in a real-world environment.

C. 1D-CNN and 2D-CNN Networks

The comparison between two networks is shown in Fig. 17. Although two methods have not shown a big performance gap at the beginning of training, as the number of training (epoch) increased, the 2D-CNN reached a smaller loss faster and began to converge. This result is better than 1D-CNN because we can achieve better results with fewer training times. During the training process, the classification accuracy of 2D-CNN is 99.8%, which is about 2% higher than 1D-CNN. The exact value of the test is shown in the text. The classification accuracy of 2D-CNN is 97.4%, which is about 3% higher than 1D-CNN. On our power consumption signal set, we get the same view as Jun et al., that is, 2D-CNN has more advantages.

D. Training Data Differentiation

We collect 150 power tracks for different applications, but in the actual attack environment, the attacker may not be able to collect so much training data. So how many samples do our model need to achieve a relatively higher inference

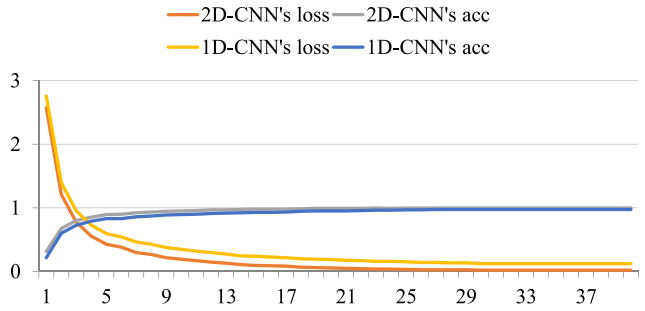


Fig. 17. Training loss and accuracy of 1D-CNN and 2D-CNN in classification.

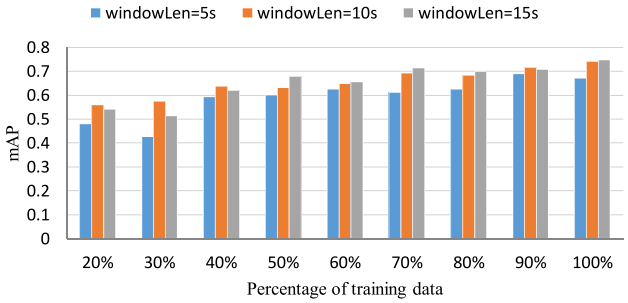


Fig. 18. The influence of different amount of training data on inference.

result? Therefore, we evaluate the impact of different amounts of training data on inference. We randomly select a different proportion from the whole sample for training and compare the results.

The results are shown in Fig. 18. We find that the number of samples significantly impacts the inference results. With the decrease in the number of samples, the mAP gradually decrease. However, even if only 20% of the samples were used, the mAP is decreased by 20%. When using 50% of the samples, we can still guarantee the inference rate of about 50%. We believe that the current variation of AC power is irregular, and data enhancement can only make up for the influence of the diversity and complexity of power consumption data on inference.

E. Inference of Interested Applications

We evaluate the generality of the attack model, in which attackers only want to infer the applications they are interested in and ignore the applications that they are not interested in. We randomly select some applications from the data set as interesting applications. For the applications attacker is interested in, it is necessary to accurately infer the name of an application and its start-up time. The rest are treated as uninterested applications, classified as other applications. We chose the applications of interest according to different proportions and compared the results.

The inference results are shown in Fig. 19. The results show that when the attacker selects the applications interested from the whole sample for individual training, a higher mAP can be obtained. This indicates that the attacker can focus on the inference of some applications interested.

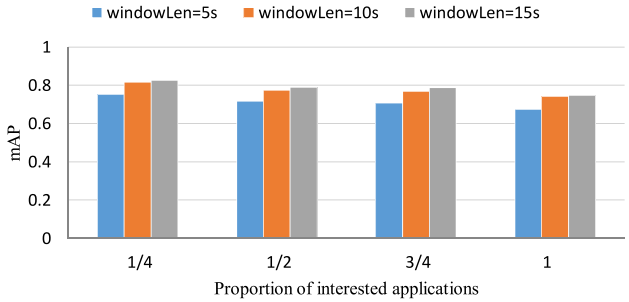


Fig. 19. Test results of different proportion of interested applications.

VII. Discussion & Limitation

A. Operating System and Machine Diversity

We believe that the differences in inference results between different brands and operating systems (OS) are reasonable, as different devices have distinct power dynamic regulation capabilities. For instance, ThinkPad devices are primarily designed for business offices, boasting longer usage times without external power supplies when compared to other devices in terms of product performance. This suggests that their power dynamic regulation capabilities are superior. Consequently, devices with good dynamic regulation will exhibit more pronounced power consumption when an application starts, outperforming those with inferior dynamic regulation.

For the difference between the Linux and Windows 10 systems, the kernel differences will affect the application processing operation. For example, Linux systems are faster to boot, and many lightweight application start-up curves are similar, making it harder to infer. The power curves for kinds of applications during start-up under Windows 10 and Linux (Ubuntu 16.04) are shown in Fig. 20.

B. Window Length

Different window lengths will result in significant differences since some applications only take a few seconds to start. For those applications with fast start-up speed, a small window length (such as 5s) might contain most signal information. In contrast, it takes more time to complete the start-up process for applications with lower start-up speed which means a smaller window cannot contain more signal information, so a larger window (such as 10s or 15s) is needed to display more signal information. However, with a larger window, the application information with a shorter start-up time cannot be displayed in detail, which is equivalent to reducing the amount of information.

C. Start-Up Time

Based on the boot time, we divide the applications into fast (5 seconds) and steady (more time is needed to complete the start). Fig. 21 shows that the result of the fast start-up application is better than that of the steady start-up application under different window lengths. The applications with fast start-up do not need additional plug-ins and load the application in better order. In contrast, the application that starts at a steady speed tends to have random effects due to the

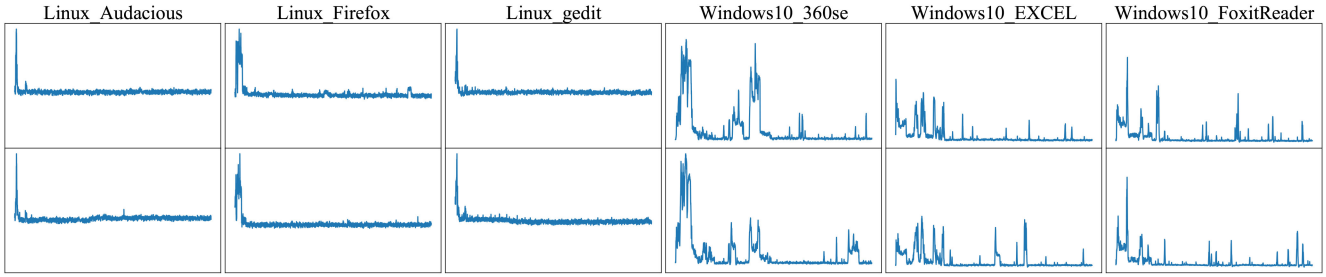


Fig. 20. Power curves for kinds of applications during startup between different brands (systems). Linux systems are faster to boot, and many lightweight applications startup waveforms are similar in height.

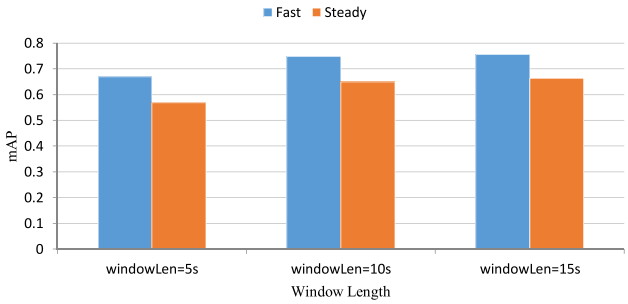


Fig. 21. mAP of single application inference for different type.

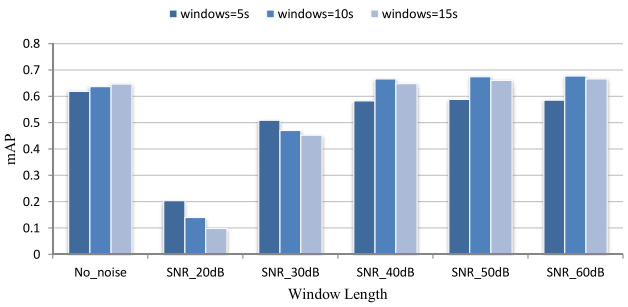


Fig. 22. mAP of single application inference for different noise.

need to load more plug-ins, resulting in less favorable results. It should be noted that when the window length is set to 10s or 15s, both two types of applications can get better performance results. A possible conjecture is that the useful information in the window increases when the window length is expanded.

D. Background Noise

Many factors affect the actual acquisition process for the analog signal, such as the current's stability and the acquisition device's stability. These factors will be added as a part of the background noise to the collected signal. We add white noise to the collected signals to get closer to the complex and diverse operating environment. The added noise is completely random Gaussian white noise, and the added degree is increased from SNR(signal-to-noise ratio) of 20dB, 30dB to 60dB and put into the noiseless group for comparison. It can be seen from Fig. 22 that under the condition of relatively high SNR, that is, under the condition of relatively low interference, there is almost no impact on the inference results of the applications. However, as noise increases, the interference gradually increases, making the model almost unpredictable.

E. Limitation

In our experiments, we make a strong assumption that the victim's computer is the only device connected to the adapter. However, in real-world scenarios, there could be other electric appliances connected to the same adapter (e.g., light, water dispenser and etc.), and the power consumption patterns of those appliances may significantly impact the performance of our attack. For instance, if a printer is connected to the same adapter and is actively working, it may consume a substantial amount of power, resulting in high power consumption peaks. These peaks could overshadow the power consumption patterns of the computer applications, making them difficult or even impossible to detect. Such situations highlight the limitations of our current experimental setup and suggest that further investigation is needed to account for potential interference from other devices sharing the same power source. This would allow for a more comprehensive assessment of the attack's effectiveness in realistic conditions. As part of our future work, we plan to address these limitations by considering the presence of multiple devices connected to the same adapter and evaluating the impact of their power consumption patterns on the performance of our attack.

VIII. Conclusion

In this work, we investigate potential vulnerabilities in consumer electronics through the power consumption analysis. We demonstrate that attackers can extract sensitive information—specifically, identifying running applications—from the analysis of power consumption, thereby profiling user behavior for commercial exploitation. This paper presents an application inference framework, thoroughly evaluated on a self-collected dataset comprising over 21,600 power trajectories from 36 popular applications. Our model adeptly identifies unseen power trajectories, achieving a mean Average Precision (mAP) of 76.69%. We establish the robustness of power traces across computers with varying operating systems and brands. Pioneering in its approach, this study quantifies the extent of information leakage via AC power consumption and its implications for running application inference, marking a critical advancement in the understanding of side-channel vulnerabilities. In future work, we aim to delve into more intricate scenarios where multiple devices connected to a single adapter could introduce noise and complicate recognition and

profiling tasks. Additionally, we plan to extend our exploration of power consumption vulnerabilities to other consumer electronics.

REFERENCES

[1] R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, "Systematic classification of side-channel attacks: A case study for mobile devices," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 465–488, 1st Quart., 2017.

[2] Y. Chen et al., "MagAttack: Guessing application launching and operation via smartphone," in *Proc. ACM Asia Conf. Comput. Commun. Security*, 2019, pp. 283–294.

[3] E. Azizi, M. T. Beheshti, and S. Bolouki, "Appliance-level anomaly detection in nonintrusive load monitoring via power consumption-based feature analysis," *IEEE Trans. Consum. Electron.*, vol. 67, no. 4, pp. 363–371, Nov. 2021.

[4] D.-Z. Sun, J.-P. Huai, J.-Z. Sun, and Z.-F. Cao, "An efficient modular exponentiation algorithm against simple power analysis attacks," *IEEE Trans. Consum. Electron.*, vol. 53, no. 4, pp. 1718–1723, Nov. 2007.

[5] D. Genkin, L. Pachmanov, I. Pipman, A. Shamir, and E. Tromer, "Physical key extraction attacks on PCs," *Commun. ACM*, vol. 59, no. 6, pp. 70–79, 2016.

[6] Q. Yang, P. Gasti, G. Zhou, A. Farajidavar, and K. S. Balagani, "On inferring browsing activity on smartphones via USB power analysis side-channel," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 1056–1066, 2017.

[7] Z. Shao, M. A. Islam, and S. Ren, "Your noise, my signal: Exploiting switching noise for stealthy data exfiltration from desktop computers," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 1, pp. 1–39, 2020.

[8] S. S. Clark, H. Mustafa, B. Ransford, J. Sorber, K. Fu, and W. Xu, "Current events: Identifying Webpages by tapping the electrical outlet," in *Proc. Eur. Symp. Res. Comput. Security*, 2013, pp. 700–717.

[9] Z. Huang, T. Zhu, Y. Gu, and Y. Li, "Shepherd: Sharing energy for privacy preserving in hybrid AC-DC microgrids," in *Proc. 7th Int. Conf. Future Energy Syst.*, 2016, pp. 1–10.

[10] G. W. Hart, "Residential energy monitoring and computerized surveillance via utility power flows," *IEEE Technol. Soc. Mag.*, vol. 8, no. 2, pp. 12–16, Jun. 1989.

[11] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power analysis attacks of modular exponentiation in smartcards," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 1999, pp. 144–157.

[12] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 541–552, May 2002.

[13] J.-M. Schmidt, T. Plos, M. Kirschbaum, M. Hutter, M. Medwed, and C. Herbst, "Side-channel leakage across borders," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.*, 2010, pp. 36–48.

[14] L. Yan, Y. Guo, X. Chen, and H. Mei, "A study on power side channels on mobile devices," in *Proc. 7th Asia–Pacific Symp. Internetware*, 2015, pp. 30–38.

[15] S. S. Clark, "The security and privacy implications of energy—Proportional computing," Ph.D. dissertation, School Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, 2013.

[16] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "PowerSpy: Location tracking using mobile device power analysis," in *Proc. 24th USENIX Security Symp. (USENIX Security)*, 2015, pp. 785–800.

[17] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. Int. Workshop Cryptographic Hardw. Embedded Syst.*, 2004, pp. 16–29.

[18] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. Annu. Int. Cryptol. Conf.*, 1996, pp. 104–113.

[19] P. Kocher, J. Jaffe, and B. Jun, *Introduction to Differential Power Analysis and Related Attacks*, Cryptography Res., Inc., San Francisco, CA, USA, 1998.

[20] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*. Heidelberg, Germany: Springer, 2009, pp. 1–4.

[21] P. Senin, "Dynamic time warping algorithm review," Dept. Inf. Comput. Sci., Univ. Hawaii at Manoa, Honolulu, HI, USA, Rep. 08-04, 2008.

[22] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mech. Syst. Signal Process.*, vol. 151, Apr. 2021, Art. no. 107398.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.

[24] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst.*, 2003, pp. 986–996.

[25] K. Boyd, K. H. Eng, and C. D. Page, "Area under the precision-recall curve: Point estimates and confidence intervals," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2013, pp. 451–466.