An Electromagnetic Covert Channel based on Neural Network Architecture

Chaojie Gu*[§] Jiale Chen[†] Rui Tan[†] Linshan Jiang[†]
*College of Control Science and Engineering, Zhejiang University, China
[†]School of Computer Science and Engineering, Nanyang Technological University, Singapore

***Abstract*—Outsourcing the design of deep neural networks may incur cybersecurity threats from the hostile designers. This paper studies a new covert channel attack that leaks the inference results over the air through a hostile design of the neural network architecture and the computing device's electromagnetic radiation when executing the neural network. Specifically, the hostile neural network consists of a series of binary models that correspond to all classes and are executed sequentially. The execution terminates once any binary model given the input is positive about its responsible class. We describe an approach to generate such binary models by pruning a benign neural network that is trained using the standard method to deal with all the classes. Compared with the benign neural network, the hostile one has similar memory usage and negligible classification accuracy drop, but distinct inference times for the samples of different classes. As a result, the hostile neural network's classification result can be eavesdropped by measuring the duration of the electromagnetic radiation emanated from the computing device. As neural networks are stored and transmitted as data files, this covert channel attack is more stealthy to the anti-malware than other code-based attacks. We implement the described attack on two edge computing devices that execute the hostile neural network on CPU or GPU. Evaluation shows 100% empirical accuracy in eavesdropping the inference results.**

***Index Terms*—Covert channel; electromagnetic radiation; neural network.**

I. Introduction

Deep neural networks (DNNs) have shown appealing performance in various applications, e.g., computer vision, speech recognition, natural language processing, and etc. However, designing and training an advanced DNN often require extensive expertise and massive compute power with hardware acceleration. For instance, it is estimated that Google spends more than 1.3 million US$ to train a natural language processing DNN for a single run [1]. Many companies and organizations in general lack the required expertise and thereby risk wasting costly compute time even if they can access massive compute power in the cloud and dare the design challenge. To make deep learning more accessible, Deep Learning as a Service (DLaaS) [2] and privatization development strategy [3], [4] have emerged. DLaaS is a cloud environment with hardware acceleration and toolkit that simplifies and automates the DNN designing process. In the simplest workflow, after the user uploads the training data, the DLaaS returns the trained DNN together with the performance report. In the privatization development strategy, the customer pays an expert individual or company for designing and training the DNN. The experts can be obtained via crowdsourcing competitions through the existing platforms (e.g., Kaggle).

However, the above DNN design outsourcing schemes (i.e., DLaaS, privatization, and crowdsourcing competition) may incur cybersecurity threats from the hostile DNN designer. For instance, the recent studies [5], [6] show that the hostile designer can implant a *backdoor* to the DNN, which, once triggered by a certain pattern in the inference sample, can induce the DNN to yield wrong inference result. In this paper, we study and verify the feasibility of a new covert channel attack that leaks the inference result of the DNN running on an air-gaped computing device. In a nutshell, the DNN is designed such that it has distinct execution times for different inference results, which can be sensed by measuring the electromagnetic radiation of the computing device. Thus, once such a hostile DNN is deployed, the covert channel presents a persistent confidentiality threat throughout the run time.

Recent studies [7]–[10] have shown that personal computer's Power Management Unit (PMU), which is a microcontroller governing the power supplies of computer components for energy efficiency, generates electromagnetic radiation at a certain frequency when the central processing unit (CPU) is busy. This creates a user-space covert channel, by which the attacker can modulate the CPU utilization to transmit information bits across the air gap. In this paper, we show through experimentation that the PMU-based electromagnetic covert channel is also available on the emerging edge computing devices equipped with graphics processing units (GPUs), e.g., the NVIDIA Jetson AGX Xavier [11] and Jetson Nano [12]. This is because such edge computing platforms also integrate the Power Management Integrated Circuits (PMICs) to manage the power supplies of CPU and GPU.

The availability of the electromagnetic covert channel on GPU-equipped edge devices motivates us to investigate the possibility of leaking the inference results of the DNN running on either the CPU or GPU of the edge device. To this end, we design a hostile DNN architecture that consists of a series of binary models, each sensitive to the samples of a particular class. Such binary models are generated by pruning a benign DNN that is trained using the standard method to deal with all classes. In the inference phase after the hostile DNN is deployed on the edge device, when given a sample, the binary models in the hostile DNN are executed sequentially and the execution process terminates once any binary model yields a positive result. The terminating binary model suggests

[§]Part of this work was completed while Chaojie Gu was with the School of Computer Science and Engineering, Nanyang Technological University.

the final classification result. As such, the inference time is strongly correlated with the classification result. Therefore, by measuring the time duration of the PMIC's electromagnetic radiation, an external eavesdropper can obtain the classification result over the air. In addition, from our extensive evaluation, the hostile DNN has similar memory usage and negligible classification accuracy drop compared with the benign DNN.

Vis-à-vis the existing PMU-based electromagnetic covert channel attacks [7]–[10] that use *useless* computation to exhaust the CPU naively, our attack inherently embeds the covert channel into the *useful* inference computation via DNN architecture. As DNN architecture and parameters are stored as data and loaded by certain deep learning frameworks (e.g., TensorFlow) for execution, our attack is more stealthy than the code-based CPU exhaustion attack method to the anti-malware that can analyze the behaviors of scripts and executables. Our study provides insights into understanding the implications of DNN architecture on information confidentiality.

The main contributions of this paper are summarized as follows.

- We show via experimentation that the PMU-based electromagnetic covert channel is also available on edge computing devices when the computation is executed mainly on either CPU or GPU.
- We present an approach to generate a hostile DNN that has distinct inference times for different classes. Thus, this DNN leaks inference results via the PMU-based electromagnetic covert channel.
- We evaluate the presented covert channel attack on several platforms using CPU or GPU to execute the hostile DNNs on a number of datasets. An attacker using a cheap software-defined radio achieves 100% empirical accuracy in eavesdropping the inference results.

The rest of this paper is organized as follows. Section II presents the background. Section III presents the measurements regarding the electromagnetic covert channel on edge devices. Section IV presents the approach to construct the hostile DNN. Section V presents evaluation results. Section VII reviews related works. Section VI discusses limitations and countermeasures. Section VIII concludes this paper.

II. Background

A. Electromagnetic Radiation on Power Management Unit

1) Power Management: Power management is a technology that helps modern computers save power, improve thermal performance and energy efficiency. Among various power management functions, managing the power supply to the CPU is critical because the CPU consumes about 30% of the total power [13]. Dynamic Frequency-Voltage Scaling (DFVS) and clock gating are two representative power management techniques. DFVS adjusts the CPU clock frequency and the supply voltage according to the performance requirements. Clock gating sets the unused units in a processor at a low power state, typically by removing the clocking signal. These CPU power management techniques help conserve power and reduce the heat generated by the CPU.

To save more power, modern computer CPUs combine the abovementioned techniques and customize their power management policies based on Advanced Configuration and Power Interface (ACPI) [14]. ACPI is an open standard for operating systems to communicate with computer hardware components and perform power management. ACPI defines multiple power states (*C States*) and performance states (*P States*) to describe idleness and performance of a CPU, respectively. Among all *P States*, a higher number represents a lower performance state. For example, the *P0* state is the highest state resulting in maximum power and frequency. Differently, the *C0* represents the normal state of the CPU. A higher number in *C State* stands for higher idleness. Different manufacturers develop different power management implementations based on APCI. Examples include Demand Based Switching (DBS) in Intel processors and "PowerNow!" in AMD processors.

2) Voltage Regulator Module: To allow the CPU to control its power levels actively, most PMUs and PMICs utilize a Voltage Regulator Module (VRM) to control the supply voltage. The CPU sends hardware signals, e.g., Voltage Identification [15], to VRM to control the voltage. The VRM is usually plugged into or soldered onto a motherboard. Some other VRMs are integrated into the CPU's package or the CPU's silicon die.

The Buck converter (step-down converter) is the most widely used implementation for the voltage regulator. It is a DC-to-DC converter that steps down the voltage. A Buck converter consists of a capacitor at its output. To keep the output voltage stable on the required level, the Buck converter periodically connects to the capacitor to refill it. In a refilling period, the power-on time and the power-off time are denoted by t_{on} and t_{off}, respectively. A switching period T_{switch} is the sum of t_{on} and t_{off}. Accordingly, the switching frequency is computed by $\frac{1}{T_{switch}}$. The VRM changes t_{on} and t_{off} to control the final output voltage changes in a switching period.

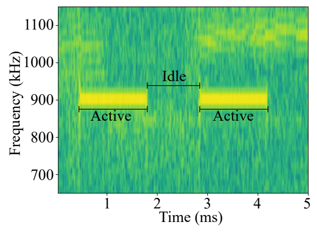
3) Electromagnetic Radiation: The VRM switching generates a burst of current in circuits. According to Faraday's law, the burst further changes the electromagnetic field near the VRM. The periodic VRM switching induces electromagnetic radiation (EMR) at a certain frequency. Note that the frequency of the PMU EMR is not the same as the switching frequency. Instead, it is the frequency of the capacitor's refilling frequency, which is the reciprocal of the interval between two consecutive t_{on}.

Recall that the VRM controls the final output voltage by changing t_{on} and t_{off}. If the switching period remains unchanged, the control forms a Pulse-Width Modulation (PWM). Differently, in Pulse-Frequency Modulation (PFM), the t_{on} is fixed, while t_{off} is variable, resulting in a changeable switching frequency. PWM and PFM have their advantages and disadvantages when the CPU loading varies. Emerging VRMs adopt a combination of these two modulation schemes [16]. Different voltage control methods have different capacitor refilling frequencies, exhibiting different PMU EMR patterns on the spectrum. For instance, the PMU EMR emitted by a VRM with PWM has a fixed frequency. Although different

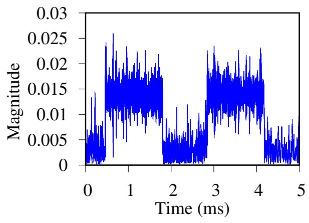
```
import time

def cpu_example(t_idle, t_active):
   counter = 0
   while True:
      t_start = time.now()
      while(time.now() - t_start < t_active):
         counter++
         sleep(t_idle)
```

Fig. 1. The method for generating PMU EMR using CPU.

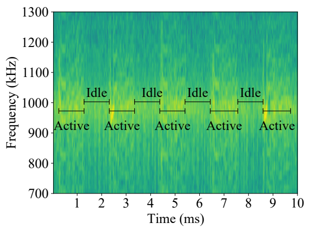


(a) Spectrogram

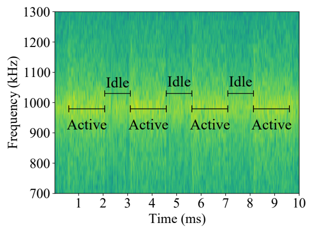


(b) Time domain

Fig. 2. Spectrogram of PMU EMR caused by CPU activities on a laptop computer and the corresponding time-domain signal after Band Pass Filter.

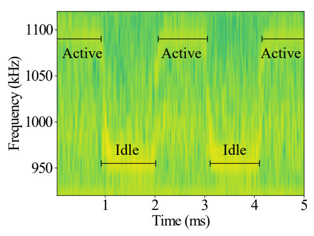


(a) CPU

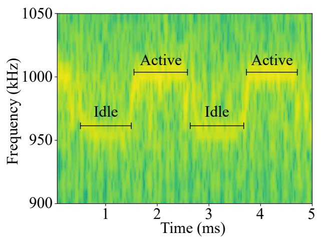


(b) GPU

Fig. 3. Spectrogram of the PMU EMR on Jetson Nano caused by CPU and GPU activities.



(a) CPU



(b) GPU

Fig. 4. Spectrogram of the PMU EMR on Jetson AGX Xavier caused by CPU and GPU activities, respectively.

VRMs have different voltage modulation methods, the switching frequency always has a lower bound and an upper bound. For example, the switching frequency of an Intel VRM ranges from 500 kHz to 1.2 MHz [17]. Many commercial off-the-shelf Software-Defined Radio (SDR) devices can receive these frequencies and have enough bandwidth. For example, RTL-SDR v3 can receive frequencies from 500 kHz up to 1.75 GHz with a 2.4 MHz bandwidth, which costs about 25US$ [18]. Thus, we can observe the PMU EMR using a cheap SDR device. Moreover, the PMU EMR signals become strong when the processor performs intensive tasks and weak when the processor is in an idle state.

B. *PMU EMR Proof-of-Concept*

To demonstrate the PMU EMR is caused by switching between the idle state and the busy state of a processor, we design a proof-of-concept example for CPU (as shown in Fig. 1). We implement the example in Python. The CPU's idle time and active time are represented by t_idle and t_active, respectively. We perform increment operations continuously to keep the CPU in the active state for t_active. Similarly, the CPU stays in the idle state for t_idle. The t_idle and t_active are set to 1 ms.

We run the programs on a MacBook Pro 2015 and receive the PMU EMR signal using an RTL-SDR v3. Fig. 2(a) shows the spectrogram of the received signals. We can see that the PMU EMR shows a "spike" pattern when the CPU is in the active state. The central frequency of the PMU EMR is near 900 kHz. As shown in Fig. 2(b), after passing a Band-Pass Filter (BPF), the PMU EMR is weak but does not disappear when the CPU is in the idle state.

III. Covert Channel Availability on Edge Devices

As discussed in Section II-A, the activities in processors create PMU EMR around VRM. We further conduct experiments to confirm the association between the PMU EMR and processors' states on edge devices.

A. *PMU EMR on Edge Devices*

The NVIDIA Jetson Nano and Jetson AGX Xavier are two representative edge computing devices. Both Nano and AGX Xavier have a high-efficient PMIC to optimize power efficiency. Besides PMU EMR triggerd by CPU activities, we also investigate whether the GPU on edge devices can trigger the PMU EMR. Similar to the algorithm in Fig. 1, as shown in Fig. 5, we implement the program for generating PMU EMR by controlling the active and idle time of the GPU. We use TensorFlow [19], one of the most widely used deep learning frameworks, to implement the program.

Jetson Nano. Jetson Nano has a Quad-core ARM A57 CPU and a 128-core Maxwell GPU. As shown in Fig. 3, the PMU EMR on Jetson Nano exhibits a unique vertical "stripe" pattern. The PMU EMRs caused by CPU and GPU activities have the same pattern. The central frequency of the PMU EMR is near 950 kHz.

Jetson AGX Xavier. Jetson AGX Xavier has an 8-core ARM 64-bit CPU and a 512-core Volta GPU. The PMU EMR on AGX Xavier has a similar "spike" pattern as the laptop computer. As shown in Fig. 4(b), when the Jetson Xavier is idle, the central frequency of the PMU EMR is about 960 kHz. The central frequency of the PMU EMR hops to 1 MHz when the processor is in the active state.

```
import time
import tensorflow as tf

def gpu_example(t_idle, t_active):
   counter = tf.Variable(0,
       trainable=False, dtype=tf.int32)
   while True:
      t_end = time.time() + t_active
      while time.time() < t_end:
         increment = tf.assign(counter,
             counter + 1)
      time.sleep(t_idle)
```

Fig. 5. The method for generating PMU EMR using GPU.

B. *Association between PMU EMR and processor states*

We further conduct experiments to confirm that the association between the PMU EMR and processor states. We change the *t_idle* and *t_active* to different values, as shown in Fig. 1 and Fig. 5. We observe that the PMU EMR signals that represent idle and active states change accordingly at run time.

On Nano and AGX Xavier, the support for power management can be divided into *CPUIdle*, which guides power usage when the CPU is idle, and *CPUFreq*, which governs power usage when it is active. We can change the device tree to disable *CPUIdle* by editing the command line. Similarly, we can boost clock speed to the maximum to disable *P-State*. To check how *CPUIdle* and *CPUFreq* affect the PMU EMR, we try to disable either one or both. For Nano, if we disable *CPUIdle*, the *CPUFreq* is disabled too. Both the CPU and GPU run at the highest frequencies. For Xavier, no matter how we enable/disable *CPUIdle* and *CPUFreq*, we can still observe the PMU EMR because the PMU EMR signals that represent idle/active states of processors are at different frequencies. We also notice that the PMU EMRs are salient even when the processors do not have intensive tasks. Although the PMU EMR channel cannot accurately describe the idle/active time of the processor, the association between PMU EMR and processor states is strong enough for designing a covert channel.

IV. Attack Design

This section presents the detailed design of the proposed covert channel attack. We first overview the attack setup and then present how to construct the hostile DNN and receive leaked inference results.

A. *Attack Setup*

Fig. 6 shows the setup of the covert channel attack. The upper part shows the perspective of the attack, while the lower part is from the perspective of the victim. The attacker first builds the covert channel by manipulating the neural network architecture. Thus, the attacker can eavesdrop on the covert channel on a host device. The victim uses the hostile model to infer the classification results of input samples.

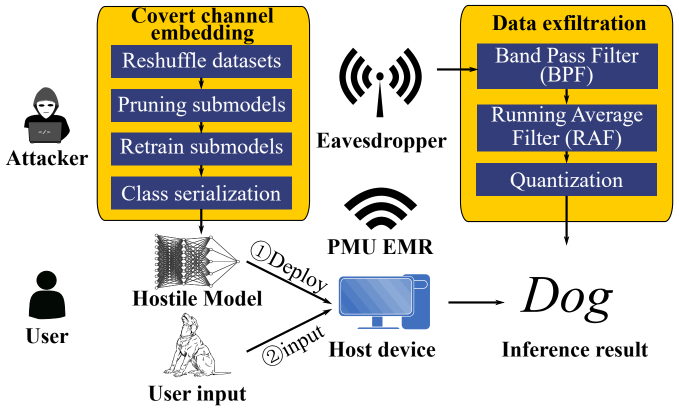


Fig. 6. Attack setup.

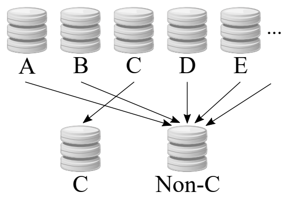


Fig. 7. Data reshuffling.

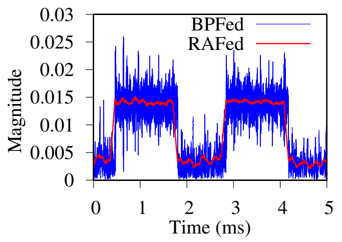


Fig. 8. The Running Average Filtered (RAFed) signal versus Band-Pass Filtered (BPFed) signal.

B. *Construction of the Hostile DNN (Transmitter)*

The construction of the hostile DNN consists of four steps: data reshuffling, submodels pruning, submodels retraining, and class serialization.

1) Data Reshuffling: The benign neural network model uses all-class training data to achieve multi-class inference. The data reshuffling reorganizes data into multiple binary classes. Fig. 7 presents the data reshuffling process. For a certain class C, the training data is divided into two classes based on it, i.e., C and $Non - C$. For example, “Dog” and “Non-Dog”. However, dividing the training data into two classes will cause imbalanced data volumes in the new dataset. For example, in CIFAR10, the $Non - C$ class data will be nine times the C class data. Thus, we only use the same number of $Non - C$ class data samples as the number of C class data samples instead of using all the $Non - C$ class data samples. This sub-dataset is equally sampled from each class within the $Non - C$ class.

2) Submodels Pruning: After data reshuffling, we obtain the same amounts of C and $Non - C$ class data samples. Taking all C class data as input, we output the feature maps of each convolutional layer. For each feature map in one layer, we compute its Average Percentage of Zero (APoZ) [20]. Let $O_c^{(i)}$ denote the output of the c-th channel in the i-th layer, the $APoZ_c^{(i)}$ of the c-th neuron in the i-th layer is $APoZ_c^{(i)} = APoZ(O_c^{(i)}) = \frac{\sum_k^N \sum_j^M f(O_c^{(i)}, j(k)=0)}{N \times M}$, where $f(\cdot) = 1$ if true, and $f(\cdot) = 0$ if false, M denotes the dimension of output feature map of $O_c^{(i)}$, and N denotes the

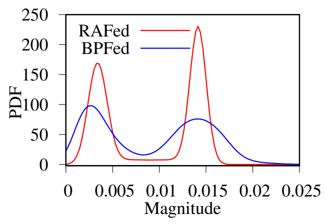


Fig. 9. Peak PDF

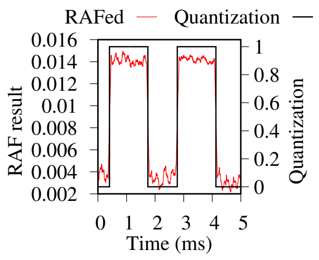


Fig. 10. Quantization

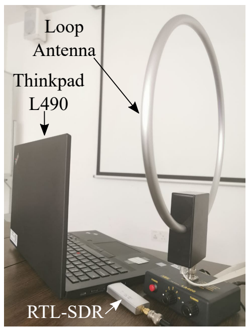


Fig. 11. Setup of eavesdropper.

total number of validation examples. If the APoZ of a feature map is higher than a threshold, the corresponding kernel will be removed. The joined submodels (i.e., the model with the covert channel) will have a similar or smaller size than the benign one by adopting submodels pruning.

3) Submodels Retraining: The pruned submodels need to be retrained to generate the final binary models. The retrained binary model is in general much smaller than the original model. The attacker can tune the pruning parameters to let the size of the hostile model close to the benign one. A detailed comparison will be given in Section V.

4) Class Serialization: Finally, we concatenate the submodels sequentially. The data will be fed into the submodels one by one. Instead of running through all the submodels, the joined model performs an early stop if the output probability of one submodel is higher than a threshold. Compared with the benign model, the hostile model's inference times for different classes are distinct while the overall classification accuracy is retained.

C. Covert Channel Receiver

We conduct three steps to extract information through the covert channel, including bandpass filtering, moving average, and binary quantization.

Although the PMU EMR has different central frequencies on different devices, the PMU EMR has a fixed central frequency for a particular type of device. Thus, we can apply a Band Pass Filter (BPF) on the received signals to extract the PMU EMR and improve the Signal-to-Noise Ratio.

As shown in Fig. 8, the PMU EMR has a rising edge when the processor switches from the idle state to the active state. Based on this observation, we can apply a Running Average Filter (RAF) on the BPFed signals to retain this edge and reduce random noises. The RAF's window length, denoted by l_{RAF}, is decided by the receiver's sampling rate. Fig. 8 presents a PMU EMR signal segment before and after RAF. We can see that the RAF smoothes the signal and zooms in the edges.

Since the distance between the eavesdropper and the host device may change, we apply an auto-quantization approach to gain the processor states. We use '0' and '1' to represent the processor's idle state and active state, respectively. A magnitude threshold decides these two states. We can learn the Probability Density Function (PDF) of the filtered signals to find the threshold. For example, as shown in Fig. 9, the magnitude values of the filtered signal mainly gather into two clusters. We can find peak magnitude values in each cluster, i.e, p_{idle} and p_{active}. We can get the threshold, denoted by α, by averaging these two clusters' magnitude values. With this threshold, we can apply the quantization function in Eq. 1 to the filtered signals. The data points greater than α are converted to 1. The data points less than α are converted to 0.

$$\Theta(S(t)) = \begin{cases} 1, S(t) > \alpha \\ 0, S(t) < \alpha \end{cases}, \alpha = \frac{p_{idle} + p_{active}}{2}. \quad (1)$$

Fig. 10 presents the result of the quantization.

V. Evaluation

In this section, we implement the proposed covert channel attack on real devices and show the feasibility by several experiments.

A. Setup

Eavesdropper. Different from the previous works that used expensive near-field probes [21] and oscilloscopes to capture PMU EMR, as shown in Fig. 11, we use an RTL-SDR v3, a low-cost SDR device, and an LA400 magnetic loop antenna [22]. The sampling rate of the RTL-SDR is 2 Msps. We use a ThinkPad L490 to collect data from the RTL-SDR.
Victim (targeted device). We use two edge devices mentioned in Section III, i.e., NVIDIA Jetson Nano and AGX Xavier. As shown in Fig. 11, we place an eavesdropper near the victim. The distance between the eavesdropper and the victim is 25 cm.
Datasets and DNN models. There are three datasets we used in the performance evaluation, i.e., MNIST [23], CIFAR-10 [24], and Speech Commands [25].

- **MNIST:** The MNIST dataset is a handwritten digit dataset. The dataset has 60,000 training samples and 10,000 testing samples. Each sample is a 28×28 grayscale image.
- **CIFAR-10:** The CIFAR-10 dataset is a 10-class image dataset. The ten classes are airplanes, cars, birds, cats, deers, dogs, frogs, horses, ships, and trucks. For each class, it has 5,000 images for training and 1,000 images for testing. The image sizes are 32×32 and in RGB color.

TABLE I
COMPARISON BETWEEN BENIGN MODELS AND HOSTILE MODELS

Dataset	Benign model size (MB)	Hostile model size (MB)	Test acuracy (Benign/Hostile)
MNIST	241	133.5	95.61%/95.76%
CIFAR-10	241	56.28	82.61%/81.93%
Speech Commands	241	165.1	69.02%/83.34%

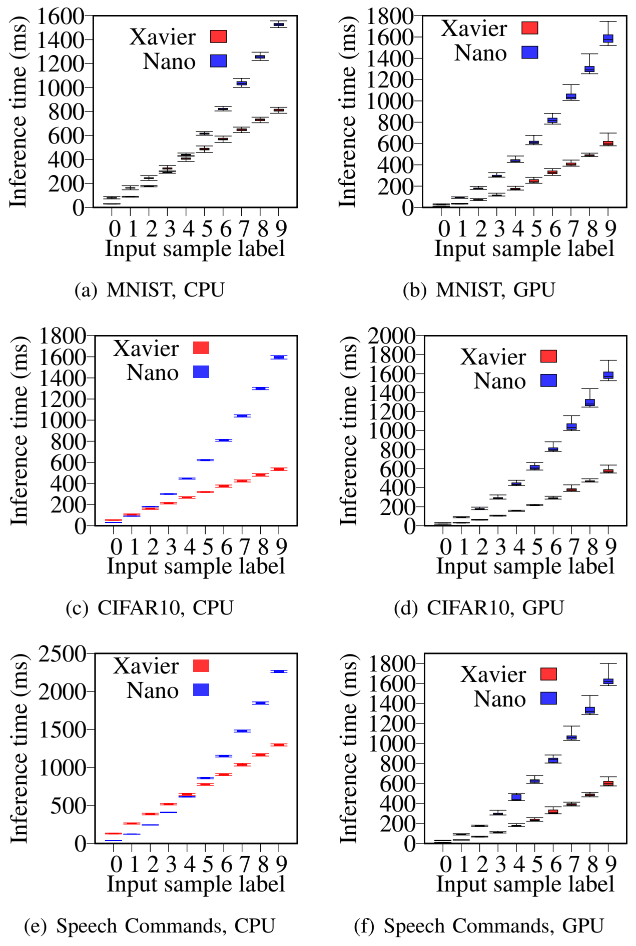


Fig. 12. The inference time of hostile models on different devices. Each box plot shows min, max, 25%, and 75% percentiles.

- **Speech Commands:** The Speech Command dataset consists of 105,829 utterances of 35 words, including “Yes”, “No”, “Up”, “Down”, etc. Each utterance is sampled at 16 kHz and stored in WAVE format. We select the first ten classes in the dataset.

In summary, our evaluation datasets cover image, text, and voice modalities and represent practical deep learning applications.

B. Measurements of Hostile Models

Tab. I compares the performances of the benign models and hostile models. The hostile models have smaller sizes while maintaining similar accuracy. We run these hostile models on Nano and AGX Xavier to measure the inference time. For each hostile model on each device, we run the hostile model with

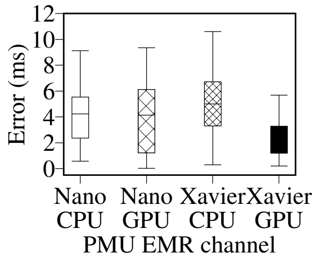


Fig. 13. Timing errors from the PMU EMR side channel on different devices.

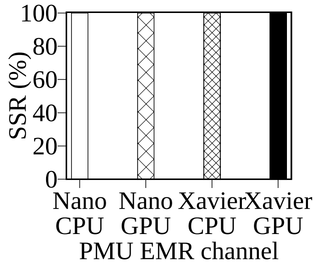


Fig. 14. SSR

CPU and GPU, respectively. Fig. 12 presents the inference times of the hostile CIFAR-10, Speech command, MNIST models on different devices. The inference times of different classes of input samples exhibit a step pattern, which enables the attacker to eavesdrop the inference result from the covert channel.

C. Eavesdropping Performance

We use two metrics, Relative Time Difference (RTD) and Sniffing Success Ratio (SSR), to describe the attack’s effect quantitively. The RTD is the absolute difference between the inference time of a sample measured on the targeted device and the inference time the attacker gets from the covert channel. An RTD close to 0 indicates the PMU EMR covert channel well reflects the real inference time. The SSR is defined as n_{same}/n_{total}, where n_{same} denotes the number of input sample labels that the attacker correctly infers from the covert channel, n_{total} represents the total number of input samples used in the evaluation. The SSR describes how well the estimation via the PMU EMR covert channel matches the inference result from the DNN model.

Fig. 13 shows the box plot of the RTD of the PMU EMR covert channel on different devices. The covert channel describes the activity states of CPU/GPU on edge devices with a maximum error of 11 ms. Notice that in the hostile models, the inference time difference between two neighbor sample classes (e.g., 1 and 2) is greater than the RTD. Thus, the proposed attack works well on edge devices. The measured SSR results are shown in Fig. 14, which are 100% on all devices.

VI. LIMITATIONS AND COUNTERMEASURES

This section discusses the limitations of the proposed covert channel attack and possible countermeasures against it. A limitation of the proposed covert channel attack is that the attacker needs to obtain the proximity of the air-gapped device. The PMU EMR attenuates significantly after propagating for a certain distance. From our measurements, when the distance between the loop antenna and the edge device is larger than 70 cm, we cannot differentiate the states of the processor from the EMR signal. The EMR emitted by the PMU used to build the covert channel is categorized as an *electromagnetic interference* (EMI). The EMI may affect or even disable other circuit components. Thus, many countries and agencies now have requirements for manufacturers to control the EMI of products [26]. Since the Buck converter in PMU is also a source of EMI, the PMU EMR has limited power and cannot propagate far away. Even the distance is limited, the attack should arouse cybersecurity attention, because there are many IoT and edge computing devices deployed and obtaining the proximity to such devices distributed in the physical world does not present a significant barrier to the attackers.

Now, we discuss the potential countermeasures against the studied covert channel attack.

EMI control. A possible countermeasure is to reduce the EMI so that the propagation distance of PMU EMR will further reduce. Considering that the manufacturers have followed the regulations regarding EMI, how to further reduce EMI is an open and challenging issue. Another approach is to add a physical shield to attenuate PMU EMR, which introduces additional costs in manufacturing.

Eavesdropper detector. EarFisher [27] is a system that can detect wireless eavesdroppers by simulating wireless eavesdroppers using bait network traffic and then capturing eavesdroppers' responses by sensing and analyzing their memory EMRs. However, in EarFisher, the traffic stimulus to trigger the eavesdropper's responses needs to be bigger than 2 MBps, which is far beyond the data rate of the proposed covert channel.

DNN inspection. The user can inspect the DNN model structure to check whether it is embedded with a covert channel. However, it needs adequate expertise because all computation structures we use in the proposed covert channel attack are *useful*. Moreover, in a privatization deployment scheme, the model provider may encrypt the model file to protect intellectual property.

VII. Related Work

Covert/side channels. Recent data exfiltration attacks exploit physical channels to bypass network security inspection. These physical channels spread in a broad spectrum, including acoustic [28], thermal emanation [29], backscatter [30], and magnetic field [31]. Researchers try to leverage EMR as a side channel for extracting information from a targeted device. GSMem [32] modulates memory EMR using binary on-off keying. EMLoRa [33] adopts LoRa physical layer modulation technique (Chirp Spreading Spectrum, CSS) for the EMR to boost the communication range. Sehatbakhsh *et al.* [10] exploit the EMR emitted by the power management unit for covert communication and keystroke detection. Other researchers utilize EMR side channels for device identification [34]–[36], malware/intrusion detection [37]–[39], neural network reverse engineering [40], hardware/software attestation [41], power grid monitoring [42], and wireless eavesdropper detection [27].

Different from all existing works on EMR covert/side channels that intentionally use useless computation to exhaust CPU to create EMR, our attack implants the covert channel in useful inference computation by manipulating the DNN architecture. The proposed covert channel attack on DNN models is more stealthy because it does not perform any useless computation. Moreover, compared with full-fledged devices such as workstations and rack servers, edge devices face a higher risk of attack because they are often deployed at the frontier, especially in IoT. Besides, Intel's x86-64 processors, we find that ARM processors and some GPUs are vulnerable to the proposed covert channel attack.

Backdoor attacks on DNN. Backdoor attack injects backdoor into a DNN model such that the backdoored model can perform well on the main task and attacker-chosen task. On the one hand, the backdoored model behaves normally on the clean test dataset. On the other hand, it can misclassify the input data samples with secret triggers, as the attacker preferred. Gu *et al.* proposed Badnet, which generates a backdoored model by training on the poisoned training set [43]. By poisoning the clean dataset with efficient data samples which contain the secret triggers and modified labels, the model is injected with the backdoors during the training phase. Chen *et al.* also use data poisoning to realize a targeted backdoor attack [44]. Different from the Badnet, they focus on the design of the secret trigger. Aiming at physically implementable backdoors, they propose a pattern-key strategy of generating more feasible triggers. Liu *et al.* focused on the Trojan Attack to generate the backdoored model [45]. They propose to generate the trigger based on values that would induce the maximum response of specific internal neurons in the model. In this way, they do not need access to the clean training dataset. Although the above approaches achieve high attack rates more than 90%, they solely focus on the backdoored data samples to mislead a learning system, which is orthogonal to this work. This work investigates how to steal the inference results of a DNN model on an air-gapped device.

VIII. Conclusion

This paper studies a new covert channel attack that leaks the inference results on air-gapped devices. The covert channel utilizes the electromagnetic radiation emitted by the power management unit of the computing device. Experiment results show that the hostile DNN model with the covert channel design achieves similar memory usage and negligible classification accuracy drop, compared with the benign model. The external attacker can then use a cheap software-defined radio receiver to eavesdrop the inference results over the air. These results call for further research on the countermeasures against the covert channel attack based on neural network architecture.

Acknowledgement

The authors acknowledge Dr. Jun Huang, Cheng Shen, and Prof. Mo Li's insightful discussions for this work. This research was supported by Science and Technology Innovation 2030 program under grant 2018AAA0101605 and an MOE AcRF Tier 1 grant 2019-T1-001-044.

References

[1] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, Jul. 2019, pp. 3645–3650.

[2] B. Bhattacharjee, S. Boag, C. Doshi, P. Dube, B. Herta, V. Ishakian, K. Jayaram, R. Khalaf, A. Krishna, Y. B. Li *et al.*, "Ibm deep learning service," *IBM Journal of Research and Development*, vol. 61, no. 4/5, pp. 10–1, 2017.

[3] Baidu, Inc. Baidu ai solution. [Online]. Available: https://ai.baidu.com/solution/private

[4] JD.com, Inc. Jd ai open platform. [Online]. Available: https://jddoversea-neuhub.jd.com/index.html

[5] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *NDSS*, 2018.

[6] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2041–2055.

[7] M. Alagappan, J. Rajendran, M. Doroslovački, and G. Venkataramani, "Dfs covert channels on multi-core platforms," in *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2017, pp. 1–6.

[8] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: exposing the perils of security-oblivious energy management," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1057–1074.

[9] S. K. Khatamifard, L. Wang, A. Das, S. Kose, and U. R. Karpuzcu, "Powert channels: A novel class of covert communicationexploiting power management vulnerabilities," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2019, pp. 291–303.

[10] N. Sehatbakhsh, B. B. Yilmaz, A. Zajic, and M. Prvulovic, "A new side-channel vulnerability on modern computers by exploiting electromagnetic emanations from the power management unit," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 123–138.

[11] N. Corporation. Jetson agx xaiver. [Online]. Available: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

[12] ——. Jetson nano. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[13] Intel Corp., "Power Management in Intel®Architecture Servers," 2009. [Online]. Available: https://www.intel.com/content/dam/support/us/en/documents/motherboards/server/sb/power_management_of_intel_architecture_servers.pdf

[14] A. C. Architecture. Advanced configuration and power interface (acpi) specification. [Online]. Available: https://uefi.org/sites/default/files/resources/ACPI_6_3_May16.pdf

[15] Intel Corp. (2009) Voltage Regulator Module (VRM) and Enterprise Voltage Regulator-Down (EVRD) Design Guidelines 11.1. [Online]. Available: https://www.intel.it/content/dam/doc/design-guide/voltage-regulator-module-enterprise-voltage-regulator-down-11-1-guidelines.pdf

[16] Maxim Integrated. (2020) Max774/max775/max776. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX774-MAX776.pdf

[17] Intel Corp. (2014) ER2120QI: 2A Step-Down DC-DC Switching Regulator. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/power/devices/dc-dc-switching-regulators/er2120qi.html

[18] RTL-SDR v3. [Online]. Available: https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/

[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[20] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.

[21] Langer EMV, "Rf2 set near-field probes 30 mhz up to 3 ghz," 2020. [Online]. Available: https://www.stratatek.com/product-page/langer-emv-rf2-set-near-field-probes-30-mhz-up-to-3-ghz

[22] AOR, LTD., "La400," 2020. [Online]. Available: http://www.aorja.com/antennas/la400.html

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[25] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[26] I. E. Commission. Basic emc publications. [Online]. Available: https://www.iec.ch/basic-emc-publications

[27] C. Shen and J. Huang, "Earfisher: Detecting wireless eavesdroppers by stimulating and sensing memory EMR," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 2021, pp. 873–886.

[28] M. Guri, Y. Solewicz, A. Daidakulov, and Y. Elovici, "Acoustic data exfiltration from speakerless air-gapped computers via covert hard-drive noise ('diskfiltration')," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 98–115.

[29] M. Guri, M. Monitz, Y. Mirski, and Y. Elovici, "Bitwhisper: Covert signaling channel between air-gapped computers using thermal manipulations," in *2015 IEEE 28th Computer Security Foundations Symposium*. IEEE, 2015, pp. 276–289.

[30] Z. Yang, Q. Huang, and Q. Zhang, "Nicscatter: Backscatter as a covert channel in mobile devices," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017, pp. 356–367.

[31] M. Guri, B. Zadov, and Y. Elovici, "Odini: Escaping sensitive data from faraday-caged, air-gapped computers via magnetic fields," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1190–1203, 2019.

[32] M. Guri, A. Kachlon, O. Hasson, G. Kedma, Y. Mirsky, and Y. Elovici, "Gsmem: Data exfiltration from air-gapped computers over GSM frequencies," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 849–864.

[33] C. Shen, T. Liu, J. Huang, and R. Tan, "When lora meets emr: Electromagnetic covert channels can be super resilient," in *IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 529–542.

[34] G. Vaidya, A. Nambi, T. Prabhakar, S. Sudhakara *et al.*, "Iot-id: A novel device-specific identifier based on unique hardware fingerprints," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 189–202.

[35] D. Yang, G. Xing, J. Huang, X. Chang, and X. Jiang, "Qid: Identifying mobile devices via wireless charging fingerprints," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 1–13.

[36] B. B. Yilmaz, E. M. Ugurlu, A. Zajić, and M. Prvulovic, "Cell-phone classification: A convolutional neural network approach exploiting electromagnetic emanations," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 2862–2866.

[37] A. Sayakkara, N.-A. Le-Khac, and M. Scanlon, "Leveraging electromagnetic side-channel analysis for the investigation of iot devices," *Digital Investigation*, vol. 29, pp. S94–S103, 2019.

[38] N. Sehatbakhsh, M. Alam, A. Nazari, A. Zajic, and M. Prvulovic, "Syndrome: Spectral analysis for anomaly detection on medical iot and embedded devices," in *2018 IEEE international symposium on hardware oriented security and trust (HOST)*. IEEE, 2018, pp. 1–8.

[39] Z. Zhang, Z. Zhan, D. Balasubramanian, B. Li, P. Volgyesi, and X. Koutsoukos, "Leveraging em side-channel information to detect rowhammer attacks," in *2020 IEEE Symposium on Security and Privacy (S&P'20)*, 2020, pp. 862–879.

[40] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 515–532.

[41] N. Sehatbakhsh, A. Nazari, H. Khan, A. Zajic, and M. Prvulovic, "Emma: Hardware/software attestation framework for embedded systems using electromagnetic signals," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 983–995.

[42] T. Shekari, C. Bayens, M. Cohen, L. Graber, and R. Beyah, "Rfdids: Radio frequency-based distributed intrusion detection system for the power grid." in *NDSS*, 2019.

[43] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[44] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[45] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," 2017.