Journal of Hardware and Systems Security (2019) 3:305–318
https://doi.org/10.1007/s41635-019-00074-w

Malware Detection in Embedded Systems Using Neural Network Model for Electromagnetic Side-Channel Signals

Haider Adnan Khan[1] · **Nader Sehatbakhsh**[1] · **Luong N. Nguyen**[1] · **Milos Prvulovic**[1] · **Alenka Zajić**[1]

Received: 28 February 2019 / Accepted: 11 July 2019 / Published online: 22 August 2019

Abstract

We propose a novel malware detection system for critical embedded and cyber-physical systems (CPS). The system exploits electromagnetic (EM) side-channel signals from the device to detect malicious activity. During training, the system models EM emanations from an uncompromised device using a neural network. These EM patterns act as fingerprints for the normal program activity. Next, we continuously monitor the target device's EM emanations. Any deviation in the device's activity causes a variation in the EM fingerprint, which in turn violates the trained model, and is reported as an anomalous activity. The system can monitor the target device remotely (without any physical contact), and does not require any modification to the monitored system. We evaluate the system with different malware behavior (DDoS, ransomware, and code modification) on different applications using an Altera Nios-II soft-processor. Experimental evaluation reveals that our framework can detect DDoS and ransomware with 100% accuracy (AUC = 1.0), and stealthier code modification (which is roughly a 5 μs long attack) with an AUC $\approx$ 0.99, from distances up to 3 m. In addition, we execute control-flow hijack, DDoS, and ransomware on different applications using an A13-OLinuXino—a Cortex A8 ARM processor single board computer with Debian Linux OS. Furthermore, we evaluate the practicality and the robustness of our system on a medical CPS, implemented using two different devices (TS-7250 and A13-OLinuXino), while executing control-flow hijack attack. Our evaluations show that our framework can detect these attacks with perfect accuracy.

Keywords Electromagnetic side-channel security · Security of cyber-physical systems · Side-channel signal analysis · Malware detection · Anomaly detection · Neural network

1 Introduction

Embedded and cyber-physical systems (CPS) have become ubiquitous in modern life, and are projected to become a USD 6.2 trillion market globally by 2025 [1]. A substantial part of this projected value comes from healthcare (USD 2.5 trillion) and manufacturing industry (USD 2.3 trillion), where networked embedded devices are used for real-time inventory tracking, to manage machines efficiently to save cost, and even to save lives. Embedded devices are prevalent in medical settings, and are widely used for portable health monitoring and electronic record keeping. In addition, cyber-physical medical devices perform many critical life supporting tasks [38]. Likewise, CPS are deployed in many critical infrastructures including power generation, military systems, autonomous, and unmanned vehicles [48]. However, proliferation of networked embedded and CPS introduce new challenges [16]. CPS are exposed to security threats that can cause severe financial and physical damage.

Attackers have already targeted, and successfully compromised different CPS including industrial control systems [13, 14, 19, 20, 36, 42], smart power grid systems [44], and medical devices [51]. Moreover, recent years witnessed a widespread Mirai distributed denial of service (DDoS) attack [27], and a variety of ransomware attacks [8] on different Internet of Things (IoT) devices.

Securing CPS can be a challenging task as they consist of many heterogeneous components including sensors, actuators, and embedded devices. In addition, CPS are often severely constrained by limited resource, power, and cost, thus existing *internal* malware detection techniques (e.g., hardware support [47], dynamic analysis [46]) are not feasible due to their overhead to the system.

✉ Haider Adnan Khan
hkhan@gatech.edu

Alenka Zajić
alenka.zajic@ece.gatech.edu

[1] Georgia Institute of Technology, Atlanta, GA 30332, USA

A possible solution to these issues is the anomaly-based *external* malware detection [10, 15, 22, 25, 40, 45, 55]. These frameworks often use a side-channel signal (e.g., power or EM) to gather real-time information about the system and report potential threats if there is a significant anomaly during the execution. While these frameworks are effective in many cases, they are either coarse-grained detection frameworks which are unable to detect tiny changes caused by a stealthy malware (e.g., [25, 45, 52]), and/or have high detection latency which makes them less attractive option for near real-time systems (e.g., [10]), and/or does not scale well with complexity of the device (e.g., [40]).

To address these issues, we propose a novel malware detection system that leverages deviations in electromagnetic (EM) side-channel signal for detecting malicious activity on embedded and cyber-physical systems using a neural network. In the training phase, the neural network first models "normal" EM side-channel signal from an uncompromised reference device using a novel "masking" technique. Next, the system continuously monitors the EM emanations from the target device. Any unusual activity in the target device causes unexpected variations in the device's EM side-channel signal. Consequently, the emanated EM signal violates the learned model (which is trained with "normal" program behaviors only). Thus, the model's prediction error goes high. We detect this deviation in the system's prediction error rate by applying low-pass filtering and thresholding, and report as an anomalous or malicious program activity. Our framework is able to detect tiny changes (due to time-domain analysis as opposed to frequency-domain), while having high accuracy with very low false positive rate and low detection latency.

To evaluate the system, we implement different malware behaviors. First, we implement a distributed denial of service (DDoS) cyber attack that sends a rapid succession of packets, a ransomware attack that encrypts a memory block, and a stealthy source code modification attack that alters the original functionality of the application. We inject these malware components into three applications running on an Altera DE-1 prototype board (Cyclone II FPGA with a NIOS II soft-processor). To assess the performance of the detection system, we monitor EM emanations from both malware-free and malware-afflicted application executions. Experimental results show that the system can detect DDoS and ransomware attacks with 100% accuracy (100% true positive rate, and 0% false positive rate), and stealthier code modifications with an area under the curve (AUC) ≈ 0.99. Next, we investigate the robustness of the detection system against variations in antenna distance and in presence of environmental EM noise. We monitor the target device from four different distances (1 m, 2 m, 3 m, and 4 m). In addition, we apply additive white Gaussian noise (AWGN) to the monitored signal to evaluate the detection performance at different SNR. The results demonstrate that the system can detect malicious activity from up to 4 m away, and with a 5 dB SNR. We further evaluate the system with different malicious activity, such as a control-flow hijack, a DDoS cyber attack and a ransomware memory encryption, on two different applications executing on an IoT device (A13-OLinuXino, with a Cortex A8 ARM processor and Debian Linux OS). Finally, to assess the practicality of the detection system, we implement a real-world medical CPS, called syringe pump, with two different devices (TS-7250 and A13-OLinuXino development board). We execute a control-flow hijack attack on the syringe pump and monitor it with the detection system. The proposed system can successfully detect all attacks on the IoT device and the syringe pump with 100% accuracy.

The major contribution of this paper is that we propose a novel framework that exploits neural network to model device's EM side-channel signal and detect anomalies which enable us to detect even tiny malicious changes with high accuracy and relatively low detection latency. We propose a novel training method that models EM signals from an uncompromised reference device, and does not require any knowledge of the nature of the malware attack (or its EM signature). Furthermore, the detection system is equally effective for different applications, and does not require access to the application's source code or control-flow graph. This is useful as the source code and CFG for many legacy and customized devices may not be readily available. This approach for remote program monitoring has several advantages:

1. Non-intrusive monitoring: The proposed system provides non-intrusive and remote monitoring. The system does not make any modification to the monitored system, nor does it impose any overhead on the monitored system. In fact, the target device is monitored externally and without any physical contact.
2. Isolation: In addition, the detection system is isolated from the monitored system and is not affected by any attack on the target device.
3. Effective against zero-day attacks: Finally, the system does not require any training on the malware signature, and thus, is effective against zero-day attacks.

The rest of the paper is organized as follows: Section 2 states the assumed threat models for the detection system. Section 3 briefly discusses the related work. Section 4 provides detailed overview of the proposed system, experimental results are evaluated in Section 5, and finally, concluding remarks are given in Section 6.

2 Threat Model

We propose a remote monitoring system for critical and high-assurance embedded and cyber-physical devices (e.g., medical devices) by leveraging the device's EM side-channel signal. The system can detect malicious attacks through anomalous EM emanation pattern detections. The envisioned threat model includes the following assumptions:

1. The malware detection system does not have any prior knowledge of the nature of the attack or its EM signature(s). The monitoring system only exploits the EM signature(s) of the monitored application. In addition, the detection system may not have access to the application's source code or control-flow graph (CFG). However, we assume that the system has a reference model for malware-free EM signature(s), which we learn by monitoring an uncompromised trusted device. We further assume that the reference model is not compromised by adversarial attacks.
2. The attacker has access to the monitored device. Furthermore, the attacker has prior knowledge of the application, and consequently, can exploit any vulnerability to execute malicious attacks on the system. For instance, the attacker may exploit a buffer-overflow vulnerability to launch a separate thread or process to execute a cyber attack (e.g., DDoS). Likewise, the attacker may execute a control-flow hijack by modifying and disrupting the existing application and its original functionality. In addition, the attacker may even reprogram the application by modifying its source code and execute malicious activity (e.g., code modification attack). However, the proposed malware detection system does not assume any knowledge of the nature of the attack and detects malicious activity through the deviation in the device's EM signature(s).

3 Related Work

Unintentional EM leakage is typically exploited by attackers for extracting cryptographic keys from target devices [3, 4, 21, 26]. Researchers have also demonstrated practical methods for measuring EM information leakage [11, 12]. While unintentional EM leakage is commonly used for cryptanalysis, EM side-channel signals can be leveraged for detailed monitoring of program activity [32]. Researchers have exploited EM side-channel signals for profiling software execution "as-is" or without any instrumentation [10, 53]. Zero overhead profiling (ZOP) [10] exploits EM signatures and performs a depth-first search (DFS) through program's control-flow graph to profile acyclic paths with 94% accuracy. Spectral profiling [53], on the other hand, observes that periodic program activities (e.g., loops) cause periodic EM emanation. This periodicity in EM signal appears as spectral peaks in the spectrogram, and can be detected through short-time Fourier transform (STFT) of the signal. Spectral profiling exploits these spectral peaks to perform loop-level profiling of the program activity.

In addition, [25, 45, 52] exploit the spectral peaks for intrusion detection. Any deviation in a program's loop causes shift in the spectral peaks. EDDIE [45] exploits this spectral shift and can efficiently detect even tiny injections inside the program loop. However, [45] can only detect much larger ($>$ 500,000 instructions) malware outside the loop. Similarly, syndrome [52] achieves similar detection performance for medical CPS. In contrast, [25] uses a stacked LSTM (long short-term memory) neural network to model spectrum sequences (i.e., the power spectral density of the EM side-channel signal), and achieves 98.9% accuracy for malware detection in PLC. However, the performance of [25] (or any system that exploits STFT) strongly depends on the size of the STFT sliding window, and it would be difficult to detect stealthier intrusions that are much smaller than the sliding window (e.g., 200 μs in [25]).

Apart from EM side-channel signal, fluctuations in device's power consumption (i.e., power side-channel signal) can be exploited for malware detection. Such approaches can, for example, protect against attacks targeting the battery life of the hand-held mobile devices [9, 29, 34, 39]. For instance, [29] and smart battery [9] exploit power profiling for mobile devices to detect power-intensive malicious activity. Likewise, VirusMeter [39] monitors battery power usage to identify "long-term" mobile malware, while [34] exploits similarities between power signatures to detect energy-greedy malware. In addition, researchers have exploited a power side-channel signal for integrity assessment of software-defined radios (SDR) [22] and for detecting malicious activity in embedded medical devices [15]. Power finger-printing [22] compares power signatures for integrity assessment of SDR and WattsUpDoc [15] exploits statistical and spectral features of embedded device's dynamic power consumption to identify malicious activity. The power side-channel has also been exploited for code execution tracking in the microcontroller unit (MCU) [40]. The system models the CFG as a hidden Markov model (HMM) and uses Viterbi algorithm for the control-flow tracking. While [40] can detect even a single instruction modification in MCU, it requires access to device's CFG and source code, which may not be readily available for many legacy and customized systems and require a high sampling rate equipment to receive and record the data.

While these mentioned frameworks are effective in many scenarios, they either (i) suffer from large detection latency (due to computational complexity of the model), (ii) require knowledge of the source code/CFG and/or the malware, (iii) are unable to detect small and stealthy malware, (iv) require physical access to the system for measurement, and/or (v) have large false positive rate.

To address above issues, our system analyzes an amplitude demodulated EM side-channel signal (in time-domain) using a neural network which enable us to detect intrusions as small as 5 μs with very low detection latency and high accuracy (an AUC $\approx$ 0.99). Furthermore, our framework can detect different malware behaviors (e.g., DDoS, ransomware) with 100% TPR and 0% FPR.

Table 1 Comparison of related work with the proposed system in terms of type of side-channel, type of device, performance, and algorithm

Ref.	Monitored side-channel	Device under test (DUT)	Description	Performance	Detection algorithm
[9]	Power signal	PDA (Dell Axim X51)	Malware detection	Detects power-intensive malware	Detects abnormal current by power profiling
[34]	Power signal	PDA (HP iPAQ)	Malware detection	Detects energy-greedy malwares with 99% TPR and less than 2% FPR	Compares power signatures with χ^2 distance
[39]	Power signal	Cell phone (Nokia 5500 Sport)	Malware detection	Detects long-term eavesdropping, call interception, and text message forwarding with 93.0%, 90.5%, 98.6% detection rate, and 4.3% FPR	Compares power consumption through machine learning
[22]	Power signal	Software-defined radio	Integrity assessment	Detects deviation in execution	Correlates power signatures
[15]	Power signal	Embedded medical device	Malware detection	Detects malware with 85% accuracy for unknown malware and 94% accuracy for known malware	Exploits statistical and spectral features of dynamic power consumption using machine learning
[40]	Power signal	8051 MCU (STC89C52)	Control-flow integrity	99.94% for recognizing instruction types and 98.56% for recognizing instruction sequence	Leverages HMM and Viterbi to recover instruction types and sequence during execution
[10]	EM signal	FPGA (Altera Cyclone II)	Software profiling	Profiles software with 94% accuracy	Exploits depth-first tree search using the control-flow graph
[45]	EM signal	A13-OLinuXino board	Malware detection	Detects malware inside and between the loops, accuracy 92% with 0% false positives but can detect only large intrusions ($> 500k$ instructions)	Uses a short-time Fourier transform and KS test
[25]	EM signal	PLC (Allen Bradley)	Control-flow integrity	98.9% detection rate (AUC)	Uses neural network (stacked LSTM) to detect legitimate PLC executions
[55]	EM signal	PLC (Siemens)	Malware detection	81.25% accuracy with 90.5% TPR and 33% FPR.	Uses deep neural network (autoencoder) based model for anomaly detection
Proposed system	EM signal	FPGA (Altera Cyclone II), TS-7250 board and A13-OLinuXino board	Malware detection	Detects DDoS, ransomware, and control-flow hijack with 100% accuracy and 5 μs code modification with AUC $\approx$ 99%	Models EM side-channel signal with deep neural network

Moreover, the system can successfully detect malware in more complicated systems such as A13-OLinuXino single-board-computer (which uses a 1 GHz ARM Cortex A8 processor and a Debian Linux OS) and can monitor the system from up to 4 m distance without any explicit knowledge about the device's source code or CFG. For ease of reference, we compare the proposed system with other related works in Table 1.

Neural networks are used for a wide variety of applications including speech recognition [23], image classification [35], and natural language processing [17]. Neural network models are also exploited for time-series prediction [33], stock market forecasting [54], and network traffic prediction [5]. Likewise, [41] exploits an LSTM network for anomaly detection in the time-domain signal such as the ECG signal through forecasting. Unlike traditional forecasting or forecasting-based anomaly detection, which use only past values to model future trends, we use a novel training method that uses *both* past and future samples. Such an approach enables us to accurately predict the amplitude of EM signal for any given point during the execution. Further details about our framework are presented in the next section.

4 Overview of the Proposed System

We exploit a multilayer neural network for anomalous (hence potentially malicious) program activity detection through device's EM side-channel signal analysis. Figure 1 demonstrates a high-level view of the proposed system. During the training phase, the neural network is trained to model the device's EM side-channel signal by executing trusted programs on a reference device. After training, the system is deployed and continuously monitors the EM emanation from the target device. When the target device performs malicious activity, it emanates anomalous (i.e., untrained) EM signal. The deviation in the EM signal causes higher prediction error (as shown in Fig. 2), and the system reports this as an anomalous program activity. We describe the system in further detail in the following sections.

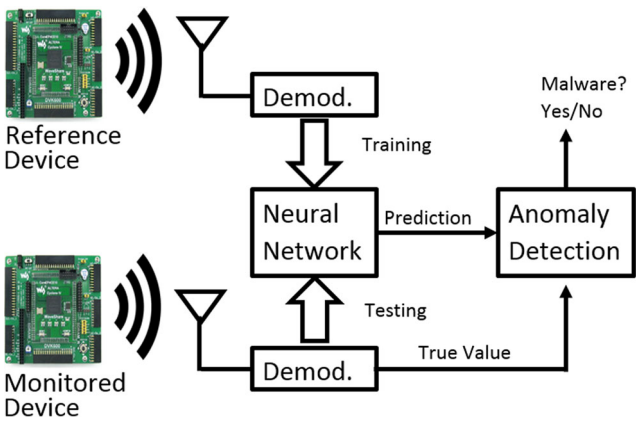

Fig. 1 Overview of the proposed malware detection system

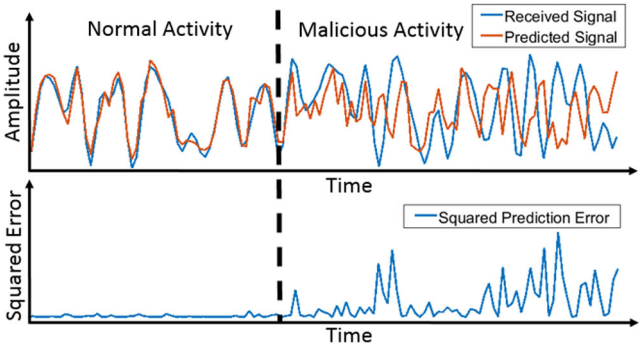

Fig. 2 Prediction error with normal activity and malicious activity

4.1 Amplitude Demodulation

Before feeding to the neural network, the emanated EM signal is first received through an antenna, amplitude demodulated at the CPU clock frequency, and digitized using an analog-to-digital converter (ADC). At each processor cycle, as the CPU executes new instructions, the states of its internal digital circuits keep changing (i.e., switch on and off). This causes a current at the CPU clock frequency whose amplitude is modulated by the variations of the executed instructions. The carrier modulated current, in turn, causes EM emanation, as it flows within the processor, and through the device's printed circuit board (PCB) [57]. Thus to analyze the program-related activities, we demodulate the received signal $r(t)$ at the CPU clock frequency f_c.

$$x_a(t) = |r(t) \times e^{j2\pi f_c t}| \tag{1}$$

Here, $x_a(t)$ is the amplitude demodulated analog signal and t denotes the time. The demodulated signal $x_a(t)$ is then passed through an anti-aliasing filter with bandwidth B and sampled at a sampling period T_s.

$$x_d(n) = x_a(nT_s) \tag{2}$$

Here, $x_d(n)$ denotes the sampled signal at sample index n. The anti-aliasing filter cancels unwanted signals with frequencies beyond $f_c \pm B$. Note that the sampling period T_s is determined by the well-known Nyquist criterion $\frac{1}{T_s} > 2B$.

Finally, we preprocess $x_d(n)$ by scale normalization.

$$x(n) = \frac{x_d(n)}{\max(x_d(n))} \tag{3}$$

This ensures that the value of $x(n)$ is between zero and one and also makes the system robust against changes in

amplitude of the EM signals (e.g., due to change in the antenna's position). Finally, $x(n)$ is used as the input for the neural network.

Furthermore, the amplitude demodulation safeguards against minor deviations in the monitored device's clock frequency. The monitored device can have clock frequency shift (due to manufacturing variation) and drift (due to temperature changes). However, the system dynamically detects the device's clock frequency f_c and applies synchronous amplitude demodulation at the detected clock frequency (Eq. 1). Consequently, the system is robust against clock frequency shift and drift of the monitored device.

4.2 Proposed Neural Network

We use a multilayer perceptron (MLP) to model the device's EM side-channel signal. An MLP is a class of feedforward artificial neural networks which consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. The output of a node in one layer is typically connected as the input for all nodes in the next layer (i.e., fully connected layer). As such, it forms a weighted and directed graph, and can be exploited to infer complex functions from observations [18, 37].

Each node j computes a weighted sum of its inputs $\mathbf{x}$ and adds a bias b_j to it (as illustrated in Fig. 3).

$$z_j = \langle \mathbf{w_j}, \mathbf{x} \rangle + b_j \tag{4}$$

Here, z_j is the weighted sum of the inputs and bias at node j, and $\mathbf{x}$ is the input vector, $\mathbf{x} = [x_1, x_2, x_3, ..., x_m]$ and w_j is the vector of connection weights,

$\mathbf{w_j} = [w_1, w_2, w_3, ..., w_m]$ and $\langle \cdot, \cdot \rangle$ denotes the scalar product operation. Next, z_j is passed through an activation function (e.g., sigmoid function, hyperbolic tangent function, linear and rectified liner functions) [31].

$$y_j = \phi(z_j) \tag{5}$$

Here, y_j denotes the output of node j after applying the activation function $\phi(\cdot)$. The activation adds non-linearity to the neural network and helps to model non-linear functions.

While each node performs a simple computation, a neural network can learn to approximate complicated functions by adjusting its weights and biases through training. During training, the network parameters (i.e., weights and biases) are optimized by minimizing a loss function (or cost function) through backpropagation algorithm [50].

As illustrated in Fig. 4, the proposed system exploits a neural network architecture that has six fully connected hidden layers with 256, 128, 96, 64, 32, and 16 nodes respectively. The input layer has 128 input nodes (i.e., a vector of 128 consecutive samples of $x(n)$) while the output layer has only one output node (i.e., the *estimated* amplitude for sample n). All the hidden layers and the output layer use rectified linear unit (ReLU) as activation function as rectified linear units have shown to improve performance [43, 58] by mitigating the well-known vanishing gradient [28] problem.

We used MLP to model EM patterns. The other popular network architectures include the convolutional neural network (CNN) and recurrent neural network (RNN). CNNs are traditionally used for 2D data (e.g., image classification), while RNNs are useful for sequence data (speech recognition, natural language processing, time-series prediction, etc.). However, RNNs are generally harder to train. MLPs, on the other hand, are very flexible, and can efficiently learn complex input to output mapping. Thus, we chose MLP due to its simplicity, flexibility, and computational efficiency.

4.3 Masking and Prediction

Our proposed neural network models the device's EM side-channel signal and predicts (or outputs) the amplitude (or value) of the EM signal at any instance, given the past and the future EM signal values (or samples) as inputs. The output is as follows:

$$y(n) = f(\mathbf{x}^{(n)}) \tag{6}$$

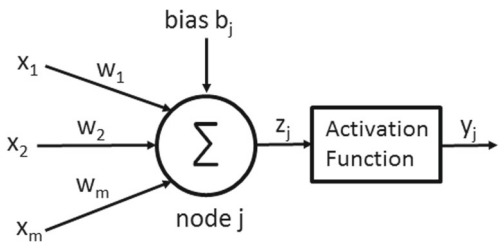

Fig. 3 Computation performed by a single node

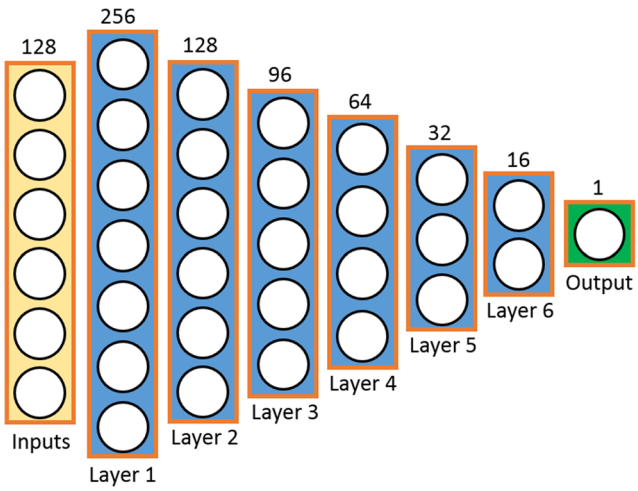

Fig. 4 Architecture of the proposed multilayer neural network

where $f(\cdot)$ denotes the neural network model for the device's EM side-channel signal, $y(n)$ is the output (or predicted value), and $\mathbf{x}^{(n)}$ denotes the input vector of the neural network at sample index n. The input vector $\mathbf{x}^{(n)}$ consists of D samples (i.e., $D = 2(d-k) = 128$ in our system). To better predict $y(n)$, our model uses d *previous* and d *future* samples. However, we hide or mask the k immediate past and the k immediate future samples, as illustrated in Fig. 5. The main reason for using such a *mask* is that the adjacent samples from an analog time-domain signal, such as EM signal, are usually highly correlated, especially at higher sampling rate. As such, the value of any unknown sample can be predicted through interpolation of its adjacent samples. However, *interpolation would not be useful for differentiating between normal and anomalous EM signal patterns*. We exploit neural network model to differentiate anomalous EM signal from normal EM signal through an increase in prediction error. Therefore, we want a prediction model that works well (i.e., low prediction error) for normal (i.e., trained) patterns but results into high prediction error for anomalous (i.e., untrained) patterns. An interpolating function models unknown sample as a weighted sum of its neighbors. While interpolation could be a good model for predicting highly correlated samples, it would work equally well for both trained and untrained patterns. Thus, the prediction error for the untrained EM signal would be similar to that of the trained signal. Consequently, it would be difficult to differentiate between the normal and the anomalous activity. Therefore, we mask the adjacent samples to force the neural network to model (or remember) the "normal" EM signal patterns rather than learning an interpolating function. In our proposed system, we mask 8 immediate past samples and 8 immediate future samples (i.e., $k = 8$), and after removing the immediate 8 samples, use the remaining 64 past and 64 future samples (i.e., $d - k = 64$) as inputs as written in the following:

$$\mathbf{x}^{(n)} = [x(n-d), x(n-d+1), ..., x(n-k-1), x(n+k+1), ..., x(n+d-1), x(n+d)]. \quad (7)$$

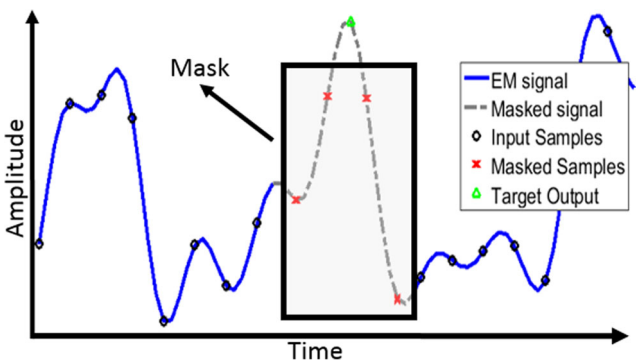

Fig. 5 Past and future samples are used as inputs (black circles) to predict the target output (green triangle). However, adjacent samples (red crosses) are masked (i.e., not used as inputs)

It is important to mention that we found that without using masking, the network performs poorly in detecting anomalies.

In the training phase, we collect EM signals by executing malware-free applications on a reference device. We then extract a smaller window from the recorded EM signal. The window consists of $2d+1$ samples, out of which $2(d-k)$ samples are uses as the input vector $\mathbf{x}^{(n)}$, 1 sample is used as the target output $x(n)$, while $2k$ samples are masked. Thus, the window acts as a training example (i.e., input and target output pair, $(\mathbf{x}^{(n)}, x(n))$). We then calculate the squared error, $e(n)$, which is computed as the squared difference between the predicted value, $y(n)$, and the true or target output value, $x(n)$, using the given training pair as follows:

$$e(n) = (y(n) - x(n))^2. \quad (8)$$

Next, we slide this window through the entire EM signal to get M training examples by setting $n = 1, 2, ..., M$. We use mean squared error (MSE) as the loss function. MSE is the average of the squared prediction error $e(n)$ as follows:

$$\mathbf{MSE} = \frac{1}{M}\sum_{n=1}^{M} e(n). \quad (9)$$

Here, M is the number of training examples (i.e., total number of windows) which in our evaluations typically ranges from 2 to 5 million samples. During training, the network parameters are optimized by minimizing the loss function MSE through a stochastic gradient descent (SGD) [7] optimization. Note that our neural network training is designed such that it minimizes the *average* error not the error for individual prediction. The main reason is that during training we observed that individual samples can sometimes experience relatively large error due to temporary changes in EM signals caused by transient noise (e.g., EMI) and/or micro architectural events (e.g., cache misses); however, the overall behavior of the signal follows a deterministic pattern for a given application. Thus, as the MSE is minimized, the neural network learns to model and predicts the EM signal more accurately (i.e., the prediction error decreases on average).

4.4 Anomaly Detection

During the monitoring phase, the trained neural network model is deployed to monitor a target device. The system continuously observes the EM emanation from the device, and extracts input and target output pair $(\mathbf{x}^{(n)}, x(n))$ from the EM signal. We use $\mathbf{x}^{(n)}$ as test inputs to predict $y(n)$

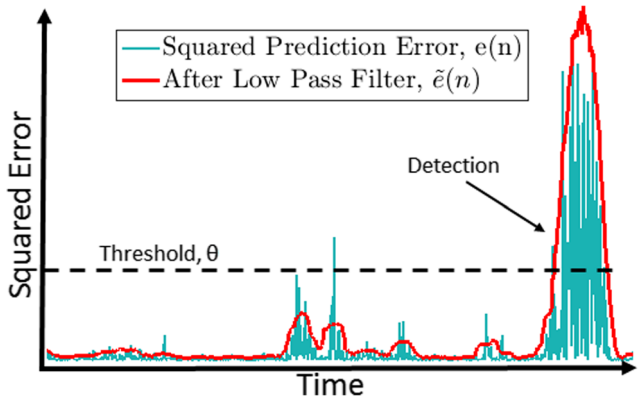

Fig. 6 Low-pass filtering and thresholding

and compute the squared prediction error $e(n)$. When the target device performs malicious or anomalous activity (i.e., any activity that the neural network was not trained with), it causes unexpected deviations in the device's EM signal. This, consequently, increases the network's squared prediction error $e(n)$. We exploit this fluctuation in $e(n)$ to detect malware execution.

To avoid false positives due to transient noise or variations in hardware activities which could cause temporary large errors, we low-pass filter the squared prediction error $e(n)$ and apply thresholding to detect anomalous program behavior. Figure 6 shows an example on how filtering and thresholding can be helpful to avoid false positives while maintaining the accuracy. We apply a $2N + 1$ samples long moving average (MA) filter (as a low-pass filter) to the signal $e(n)$, yielding the filtered signal $\tilde{e}(n)$ as follows:

$$\tilde{e}(n) = \frac{1}{2N+1} \sum_{i=-N}^{N} e(n-i). \tag{10}$$

This low-pass filtering results into a bi-modal probability density function (PDF) (as shown in Fig. 7), where the squared prediction error $\tilde{e}(n)$ for normal and malicious

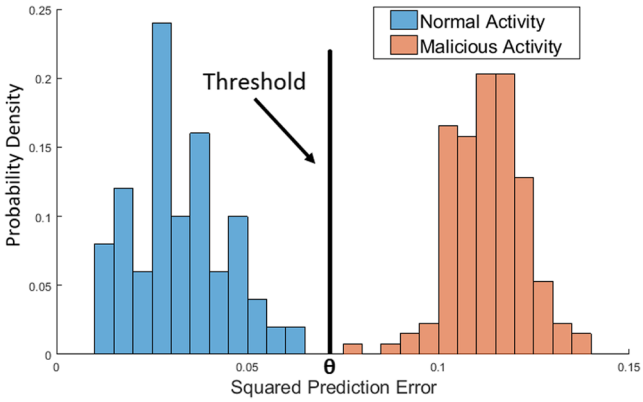

Fig. 7 Threshold selection using PDF of squared prediction error for normal and malicious program activity

program activity can be separated by a threshold θ . Thus, we set a threshold θ on $\tilde{e}(n)$ between the two PDF and report anomalous program activity whenever $\tilde{e}(n) > \theta$.

4.5 System Parameters

The performance of the detection system depends on a number of system parameters, such as the length of the input vector D, the size of the mask k, the moving average filter parameter N, and the threshold parameter θ. In this section, we discuss how these parameters are chosen and their impacts on the system performance.

Input Vector Length D The EM signal represented by the input vector provides a "context" for the prediction. More specifically, the neural network exploits the past and the future EM patterns to predict the present EM amplitude. While a larger value for D increases the "context," this also adds to the complexity of the neural network and may lead to overfitting. Thus, from empirical evaluation, we use $D = 128$.

Mask Size *k* The adjacent samples of the EM signal are more correlated at higher sampling rate (i.e., with lower time-gap between two adjacent samples). Thus, intuitively, the mask (or k) should be larger with higher sampling rate. However, a mask that is too large may overshadow the "context" and interfere with the prediction. We monitored FPGA, TS-7250, and A13-OLinuXino with 10 MHz bandwidth (i.e., 5 MHz bandwidth on either side of the clock frequency). Thus, we used the same mask ($k = 8$) throughout all experiments.

Moving Average Filter Parameter *N* The moving average filter helps to reduce false positives due to unpredictable variabilities in hardware activities (e.g., cache misses). These variabilities can cause transient yet high-valued prediction errors. As such, the PDF of the squared error for normal activity resembles an exponential function with a long tail (as shown in Fig. 8). This tail overlaps with the PDF of the malicious activity, and consequently, generates a lot of false positives. However, the MA filter reduces the false positives by transforming the PDF into a symmetric (Gaussian-like) function. With increasing N, the function gets sharper with shorter tail and results into fewer false positives (i.e., less overlap with the PDF of the malicious activity). However, this reduction of the false positives comes at the cost of increased detection latency. Furthermore, shorter malicious activities (e.g., intrusions that last shorter than N samples) may go undetected. Thus, the optimal N is a trade-off between reliable detection (low false positives) and detection latency. In our experiments, we used $N = 64$ for monitoring FPGA and $N = 1024$

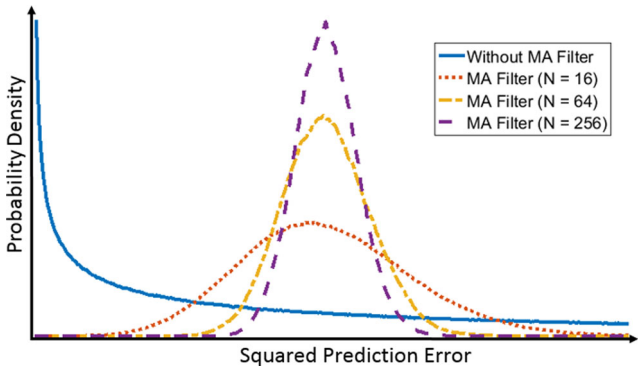

Fig. 8 Probability density function of the squared prediction error for normal program activity with and without moving average filter

for monitoring TS-7250 and A13-OLinuXino board. The higher-order MA filter safeguards against larger variations in EM signal due to the unpredictable activities by the OS.

Threshold θ The threshold θ helps to distinguish the malicious activities from the normal activities and is chosen using the PDF of the squared prediction error. If the squared prediction error has bi-modal and disjoint PDF for normal and malicious activity, we can achieve 100% detection with zero false positive by setting the threshold θ between the two PDF (as in Fig. 7). However, if the two PDF overlap, the value of θ is a trade-off between false positives and false negatives. Higher value of θ will lead to lower false positives at the cost of higher false negatives and vice versa. Note that, in case of zero-day attacks, we do not have prior knowledge about the PDF for the malicious activity. Thus, we set the threshold θ slightly right to the tail of the PDF corresponding to the normal activity.

5 Experimental Evaluation

We evaluate the proposed system with several different types of malware on different applications and embedded systems.

5.1 Embedded Device with Different Malware Behavior

We implement two types of embedded system malware payloads (DDoS attacks, ransomware attacks) and a code modification attack (similar to Stuxnet) on an Altera DE-1 prototype board (Cyclone II FPGA with a 50 MHz NIOS II soft-processor). The DDoS attack exploits vulnerabilities such as buffer overflow to divert the control flow to send DDoS packets in rapid succession through the device's JTAG port. We also implement a cryptoviral ransomware [6] that performs only a single (16-byte) block encryption of AES-128. Intuitively, larger encryption should be easier to detect. Finally, we evaluate a code modification attack where the source code has been slightly modified. We added a small (about 10 instructions) to the source code to mimic the behavior of Stuxnet-like malwares where the adversary modifies the code to change a critical value based on some conditions.

We inject these malware behaviors into three selected applications (Print tokens, Replace, and Schedule) from SIR repository [49]. The system was trained and tested with a disjoint set of user inputs (i.e., the training and testing executions has different user inputs, and thus follow different control-flow paths). Consequently, there were significant variations in execution time for different inputs. For instance, in *Replace*, the shortest execution lasts only 71 μs while the longest one is 4.58 ms. Likewise, in *Print tokens*, the shortest execution is 116 μs and the longest execution takes 10.8 ms. Similarly, for *Schedule*, the shortest execution is 48 μs and the longest execution takes 12.2 ms. We used inputs (for both training and testing) that provides high path coverage (using LLVM to find the paths). For example, the Print tokens application has 87 unique acyclic control-flow branches out of which 83 were executed by the test set. Likewise, the Replace application has 96 unique acyclic control-flow branches out of which 74 were executed during testing. Similarly, the Schedule application has 83 unique control-flow branches, and all of them were executed by the test set.

The training and the cross-validation program executions were uncompromised (i.e., without malware), while the testing contained both compromised and uncompromised program executions. For Print tokens, we used 400 training, 45 cross-validation, and 192 testing executions, of which 66 had DDoS, 68 had ransomware, 8 had code modification, while 50 were without malware. Likewise, for Replace, we used 458 training, 45 cross-validation, and 188 testing executions. The testing set contained 65 DDoS, 68 ransomware, 5 code modification malware, and the rest (i.e., 50) were uncompromised. The Schedule benchmark had total 284 training, 103 cross-validation, and 294 testing examples. The testing set included 67 DDoS, 68 ransomware, 9 code modification, and 150 executions were without malware.

Figure 9 demonstrates our experimental setup. We monitor the device executing these applications using a 2.4-2.5 GHz 18 dBi panel antenna and demodulate the received EM signal using an Agilent MXA N9020A spectrum analyzer. The demodulated signal is then filtered using an anti-aliasing filter with 5 MHz bandwidth, and finally, sampled at 12.8 MHz sampling rate. The experimental results demonstrate that the mean squared prediction error for the malicious (i.e., untrained) activity is significantly higher than that of the normal (i.e., trained) activity. This is also shown in Fig. 7. While the neural network can

Fig. 9 Experimental setup of the malware detection system

successfully model and predict the EM signal for trained program activity with low prediction error, the model fails for untrained program activity. As a consequence, any execution of untrained program activity leads to deviations in device's EM emanation, which in turn results in a higher prediction error. Thus, the system can differentiate between normal and anomalous program activity through the neural network's prediction error.

Table 2 demonstrates the performance of the proposed system for detecting different malware activities on different applications. Results show that the system can detect all DDoS and ransomware without any false positive (AUC $=$ 1.0), and for code modification the system achieves roughly 0.99 AUC. It should be noted that the execution time for DDoS and ransomware is much larger (roughly 25 μs and 150 μs respectively) than that of the code modification attack, which takes up only 5 μs. Hence, code modification is stealthier and harder to detect.

We further evaluate the detection latency of the system. We use a non-causal prediction model (i.e., the neural network exploits both past and future samples to predict the present sample value). This causes a delay of $d = 72$ samples (5.625 μs) in prediction. In addition, the moving average filter introduces a delay of $N = 64$ samples (5 μs). Thus, the total system delay is $d + N = 136$ samples (10.625 μs). However, the detection latency will be higher than the system delay due to the time taken for threshold breaching by the anomalous EM pattern. The experimental mean detection latency for DDoS, ransomware, and code modification are presented in Table 3. Both DDoS and code modification are detected in less than 13 μs while ransomware is detected in 22 μs. In comparison, [25] and [45] has latency greater than 200 μs and 2000 μs respectively.

Table 2 Detection performance for different malware behaviors and different applications

	Area under the curve (AUC)		
Application	DDoS	Ransomware	Code mod.
Print tokens	1.0	1.0	1.0
Replace	1.0	1.0	0.99
Schedule	1.0	1.0	0.97

Table 3 Detection latency for different malwares

	DDoS	Ransomware	Code mod.
Latency	12.5 μs	22.0 μs	12.5 μs

5.2 Robustness Against Variations in Antenna Distance

To evaluate the robustness of the system, we trained and tested the system by placing the antenna at different positions. It is reasonable to assume that the system will be trained with a reference device and then deployed to monitor a different target device. As such, the antenna placement and positioning may vary between the training and the monitoring phase. Thus, it is important that the detection system is robust against variations in antenna placements. To evaluate the robustness of the system, we first trained the system from 1 m distance and then used this trained system to monitor the target device from four different distances (1 m, 2 m, 3 m, and 4 m). Table 4 shows that the system is robust against variations in antenna distance. In addition, the system demonstrates excellent performance from up to 4 m distance. Further distance causes some degradation in system performance due to the lower signal-to-noise ratio (SNR) at a higher distance. Note that our framework is not limited by distance and higher distance coverage can be achieved by using higher gain antennas (e.g., [30]).

5.3 Robustness Against Noise and Interference

We further evaluate the robustness of the system against environmental noise by applying AWGN to the monitored signal. Any practical monitoring system should be able to detect security threats under potentially noisy environment. Thus, we evaluate the performance of the detection system at different SNR by applying AWGN to the monitored signal. Table 5 shows that the system is robust against noise

Table 4 Detection performance at different distances

	Area under the curve (AUC)		
Distance	DDoS	Ransomware	Code mod.
1 m	1.0	1.0	0.99
2 m	1.0	1.0	0.99
3 m	0.99	1.0	0.97
4 m	0.96	0.94	0.71

Table 5 Detection performance at different signal-to-noise ratio

	Area under the curve (AUC)		
SNR	DDoS	Ransomware	Code mod.
30 dB	1.0	1.0	0.99
20 dB	1.0	1.0	0.98
10 dB	1.0	1.0	0.95
5 dB	0.85	0.95	0.71

and has an excellent detection performance even at an SNR as low as 10 dB.

In addition, the system is robust against any EM interference outside its monitored bandwidth. As described in Section 4.1, the anti-aliasing filter used during the analog-to-digital conversion nullifies any signal with frequencies beyond $f_c \pm B$. Here, f_c is the clock frequency of the monitored device, and $2B$ is the monitored bandwidth. Thus, any EM interference outside the monitored bandwidth does not influence the detection performance.

5.4 Attack on IoT Device

We implement three different malicious activities (e.g., code injection, DDoS, and ransomware) on an IoT device (A13-OLinuXino board with 1 GHz Cortex A8 ARM processor and Debian Linux OS). We inject these malicious behaviors into two selected applications (*basic math* and *bit count*) from MiBENCH [24]. First, we implement a buffer overflow attack to inject *shellcode* into the application. Next, we port a DDoS *bot* in a selected location of the application. The DDoS *bot* sends 100 TCP SYN packets and then resumes to normal program activity. Finally, we implement a ransomware prototype that performs AES 128 encryption.

We monitored the emanated EM signal with a small magnetic probe placed 5 cm away from the system using a commercially available software-defined radio (Ettus Research B200-mini) with a bandwidth of 40 MHz centered at the clock frequency (1 GHz) of the device. The collected signal was then demodulated, digitized, down-sampled to a 10-MHz sampling rate, and finally, processed through the proposed neural network framework. For each application, we trained the system with 25 uncompromised (malware-free) executions. Next, we test the system with 100 executions (25 malware-free, 25 with code injection, 25 with DDoS, and 25 with ransomware). Experimental evaluations (in Table 6) show that the system detects all malicious activity without any false positive.

We used the same neural network architecture and parameters (e.g., $D = 128$ and $k = 8$) throughout all experiments. However, we exploited a higher order moving average filter ($N = 1024$) to avoid false positives due to transient activities by the OS. Consequently, the detection latency of the system was higher (roughly 120 μs), which is still considerably lower than [25] and [45] (200 μs and 2000 μs respectively). Note that, [45] used a similar experimental setup (e.g., same benchmark applications executed on same device with similar code injection attacks). However, [25] monitored a PLC - a simpler device (e.g., slower clock speed and does not have an OS). Intuitively, it should be easier to model EM emanation from a simpler device (e.g., in absence of unpredictable OS activities), and thus should lead to lower detection latency.

Table 6 Detection performance for different malware behaviors on IoT device

	Area under the curve (AUC)		
Application	Code inj.	DDoS	Ransomware
Basic math	1.0	1.0	1.0
Bit count	1.0	1.0	1.0

5.5 Attack on Medical Cyber-Physical System

We further evaluate the system by implementing malicious attacks on a medical CPS called syringe pump. A syringe pump is a medical device that can dispense or withdraw a precise amount of fluid or medicine [56]. A syringe pump has three main components: a syringe filled with medicine, an actuator (typically a stepper motor), and a control unit that receives user inputs that controls the actuator accordingly.

To evaluate the robustness of the proposed malware detection system, we implement an open-source syringe pump [2] with two different devices:

1) TS-7250 board (200 MHz Cirrus EP9302 ARM9 CPU with a Debian Linux OS)
2) A13-OLinuXino single-board-computer (1 GHz ARM Cortex A8 processor with a Debian Linux OS)

We exploit a buffer overflow vulnerability in the serialRead() function to hijack the control flow and call MoveSyringe() function to dispense or withdraw an unwanted amount of fluid. This is an example of a code-reuse attack where the attacker repurposes existing code to

Table 7 Malware detection performance for syringe pump implemented with different devices

	TS-7250	A13-OLinuXino
AUC	1.0	1.0

perform unwanted action. As the attacker executes existing code, albeit in an undesired way, a code-reuse attack can be harder to detect. Any failure to administer medication at an appropriate dosage can have a serious consequences for the patient. Thus, this attack poses a critical threat to the integrity of the syringe pump. For monitoring, we place a small magnetic probe 5 cm away from the system and record and demodulate the signal using a commercially available software-defined radio (Ettus Research B200-mini).

We train the system with 25 executions and test it with 50 executions, out of which half were compromised with malware. Experimental results (in Table 7) show that the system achieves excellent performance and detects all malicious activity without any false positive.

6 Conclusions

We propose novel framework for malware detection in critical and high-assurance embedded and cyber-physcial systems using EM side-channel signal analysis. The system models device's EM emanation with a multilayer perceptron (MLP) and detects anomalous or malicious program activity through deviations in the EM fingerprint. The system is trained with EM signal from uncompromised reference device and can predict EM emanation for normal (i.e., trained) program activity. However, whenever the monitored device performs any malicious (i.e., untrained) program activity, the trained neural network model fails and results in high prediction error. We then detect this deviation in prediction error and report anomalous activity. The system does not require any knowledge about the nature of the attack or its malware signature, thus ensures protection against any zero-day attack. In addition, the system can provide non-intrusive and remote monitoring (without any physical access to the device) and does not require any modification to the monitored system. It does not impose any overhead on the monitored device. The detection system can train its model by observing device's EM emanation and does not require any access to the source code or the control-flow graph of the monitored system. We demonstrate the effectiveness of the system with different malware behaviors (DDoS, ransomware, and code modification), which the system could detect with an excellent accuracy (AUC $\approx$ 0.99) from up to 3 m away. The system was also able to detect attacks on an IoT device and a medical CPS with a 100% accuracy.

Funding Information This work has been supported, in part, by NSF grant 1563991 and DARPA LADS contract FA8650-16-C-7620. The views and findings in this paper are those of the authors and do not necessarily reflect the views of NSF and DARPA .

References

1. INTEL a guide to the Internet of Things infographic. https://www.intel.com/content/www/us/en/internet-of-things/infographics/guide-to-iot.html. Accessed: 2018-10-25
2. Abera T, Asokan N, Davi L, Ekberg JE, Nyman T, Paverd A, Sadeghi AR, Tsudik G (2016) C-FLAT: control-flow attestation for embedded systems software. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp 743–754. ACM
3. Agrawal D, Archambeault B, Rao JR, Rohatgi P (2002) The EM side—channel (s). In: International workshop on cryptographic hardware and embedded systems, pp 29–45. Springer
4. Alam M, Khan HA, Dey M, Sinha N, Callan R, Zajic A, Prvulovic M (2018) One&Done: a single-decryption EM-based attack on OpenSSL's constant-time blinded RSA. In: Proceedings of the 27th USENIX conference on security symposium, pp 585–602. USENIX Association
5. Alarcon-Aquino V, Barria JA (2006) Multiresolution fir neural-network-based learning algorithm applied to network traffic prediction. IEEE Trans Syst Man Cybern Part C Appl Rev 36(2):208–220
6. Andronio N, Zanero S, Maggi F (2015) Heldroid: dissecting and detecting mobile ransomware. In: International workshop on recent advances in intrusion detection, pp 382–404. Springer
7. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp 177–186. Springer
8. Brewer R (2016) Ransomware attacks: detection, prevention and cure. Netw Secur 2016(9):5–9
9. Buennemeyer TK, Nelson TM, Clagett LM, Dunning JP, Marchany R, Tront JG (2008) Mobile device profiling and intrusion detection using smart batteries. In: Proceedings of the 41st annual Hawaii international conference on system sciences, pp 296–296. IEEE
10. Callan R, Behrang F, Zajic A, Prvulovic M, Orso A (2016) Zero-overhead profiling via EM emanations. In: Proceedings of the 25th international symposium on software testing and analysis, pp 401–412. ACM
11. Callan R, Zajić A, Prvulovic M (2014) A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events. In: Proceedings of the 47th annual IEEE/ACM international symposium on microarchitecture, pp 242–254. IEEE Computer Society
12. Callan R, Zajić A, Prvulovic M (2015) FASE: finding amplitude-modulated side-channel emanations. In: 2015 ACM/IEEE 42nd annual international symposium on computer architecture (ISCA), pp 592–603. IEEE
13. Cárdenas A. A., Amin S, Lin ZS, Huang YL, Huang CY, Sastry S (2011) Attacks against process control systems: risk assessment, detection, and response. In: Proceedings of the 6th ACM symposium on information, computer and communications security, pp 355–366. ACM
14. Chien E (2010) Stuxnet: a breakthrough. Symantec Com 12
15. Clark SS, Ransford B, Rahmati A, Guineau S, Sorber J, Xu W, Fu K (2013) Wattsupdoc: power side channels to nonintrusively discover untargeted malware on embedded medical devices. In: HealthTech
16. Colbert E (2017) Security of cyber-physical systems— CSIAc. J Cyber Secur Inf Syst 5(1)
17. Collobert R, Weston J (2008) A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning, pp 160–167. ACM

18. Deng L, Yu D, et al. (2014) Deep learning: methods and applications. Foundations and Trends®in Signal Processing 7(3–4):197–387
19. Falliere N, Murchu LO, Chien E (2011) W32. stuxnet dossier. White paper, Symantec Corp., Security Response 5(6)
20. Farwell JP, Rohozinski R (2011) Stuxnet and the future of cyber war. Survival 53(1):23–40
21. Genkin D, Pachmanov L, Pipman I, Tromer E (2015) Stealing keys from PCs using a radio: cheap electromagnetic attacks on windowed exponentiation. In: International workshop on cryptographic hardware and embedded systems, pp 207–228. Springer
22. González CRA, Reed JH (2011) Power fingerprinting in sdr integrity assessment for security and regulatory compliance. Analog Integr Circ Sig Process 69(2-3):307
23. Graves A, Mohamed AR, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on Acoustics, speech and signal processing (icassp), pp 6645–6649. IEEE
24. Guthaus MR, Ringenberg JS, Ernst D, Austin TM, Mudge T, Brown RB (2001) Mibench: A free, commercially representative embedded benchmark suite. In: Proceedings of the 4th annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538), pp 3–14. IEEE
25. Han Y, Etigowni S, Liu H, Zonouz S, Petropulu A (2017) Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 1095–1108. ACM
26. Hayashi YI, Homma N, Mizuki T, Shimada H, Aoki T, Sone H, Sauvage L, Danger JL (2013) Efficient evaluation of em radiation associated with information leakage from cryptographic devices. IEEE Trans Electromagn Compat 55(3):555–563
27. Herzberg B, Bekerman D, Zeifman I (2016) Breaking down mirai: an IoT DDoS botnet analysis. Incapsula Blog, Bots and DDoS, Security
28. Hochreiter S (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty. Fuzziness and Knowledge-Based Systems 6(02):107–116
29. Jacoby GA, Marchany R, Davis N (2004) Battery-based intrusion detection a first line of defense. In: Proceedings from the 5th annual IEEE SMC information assurance workshop, 2004, pp 272–279. IEEE
30. Juyal P, Adibelli S, Sehatbakhsh N, Zajic A (2018) A directive antenna based on conducting discs for detecting unintentional em emissions at large distances. IEEE Transactions on Antennas and Propagation pp 1–1. https://doi.org/10.1109/TAP.2018.2870370
31. Karlik B, Olgac AV (2011) Performance analysis of various activation functions in generalized MLP architectures of neural networks. International Journal of Artificial Intelligence and Expert Systems 1(4):111–122
32. Khan HA, Alam M, Zajic A, Prvulovic M (2018) Detailed tracking of program control flow using analog side-channel signals: a promise for IoT malware detection and a threat for many cryptographic implementations. In: Cyber Sensing 2018, vol. 10630, p. 1063005. International Society for Optics and Photonics
33. Khashei M, Bijari M (2010) An artificial neural network (p, d, q) model for timeseries forecasting. Expert Systems with applications 37(1):479–489
34. Kim H, Smith J, Shin KG (2008) Detecting energy-greedy anomalies and mobile malware variants. In: Proceedings of the 6th international conference on mobile systems, applications, and services, pp 239–252. ACM
35. Krizhevsky A, Sutskever I, Hinton G (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
36. Langner R (2011) Stuxnet: dissecting a cyberwarfare weapon. IEEE Secur Priv 9(3):49–51
37. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
38. Lee I, Sokolsky O (2010) Medical cyber physical systems. In: 2010 47th ACM/IEEE design automation conference (DAC), pp 743–748. IEEE
39. Liu L, Yan G, Zhang X, Chen S (2009) Virusmeter: preventing your cellphone from spies. In: International workshop on recent advances in intrusion detection, pp 244–264. Springer
40. Liu Y, Wei L, Zhou Z, Zhang K, Xu W, Xu Q (2016) On code execution tracking via power side-channel. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp 1019–1031. ACM
41. Malhotra P, Vig L, Shroff G, Agarwal P (2015) Long short term memory networks for anomaly detection in time series. In: Proceedings, p. 89. Presses universitaires de Louvain
42. McMillan R (2010) Siemens: stuxnet worm hit industrial systems. Computerworld, 14
43. Nair V, Hinton G (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814
44. Nakashima E, Mufson S (2008) Hackers have attacked foreign utilities, CIA analyst says. Washington Post
45. Nazari A, Sehatbakhsh N, Alam M, Zajic A, Prvulovic M (2017) Eddie: Em-based detection of deviations in program execution. In: 2017 ACM/IEEE 44th annual international symposium on computer architecture (ISCA), pp 333–346. IEEE
46. Newsome J, Song DX (2005) Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software
47. Ozsoy M, Khasawneh KN, Donovick C, Gorelik I, Abu-Ghazaleh NB, Ponomarev D (2016) Hardware-based malware detection using low-level architectural features. IEEE Trans Comput 65(11):3332–3344
48. Richards R (2016) High-assurance cyber military systems (HACMS). DARPA, mil
49. Rothermel G, Elbaum S, Kinneer A, Do H (2006) Software-artifact infrastructure repository. URL http://sir.unl.edu/portal
50. Rumelhart DE, Hinton G, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533
51. Sametinger J, Rozenblit J, Lysecky R, Ott P (2015) Security challenges for medical devices. Commun ACM 58(4):74–82
52. Sehatbakhsh N, Alam M, Nazari A, Zajic A, Prvulovic M (2018) Syndrome: spectral analysis for anomaly detection on medical IoT and embedded devices. In: 2018 IEEE international symposium on hardware oriented security and trust (HOST), pp 1–8. https://doi.org/10.1109/HST.2018.8383884
53. Sehatbakhsh N, Nazari A, Zajic A, Prvulovic M (2016) Spectral profiling: observer-effect-free profiling by monitoring EM emanations. In: The 49th annual IEEE/ACM international symposium on microarchitecture, p. 59. IEEE Press
54. Ticknor JL (2013) A Bayesian regularized artificial neural network for stock market forecasting. Expert Systems with Applications 40(14):5501–5506
55. Wang X, Zhou Q, Harer J, Brown G, Qiu S, Dou Z, Wang J, Hinton A, Gonzalez CA, Chin P (2018) Deep learning-based classification and anomaly detection of side-channel signals. In: Cyber Sensing 2018, vol 10630, pp 1063006. International Society for Optics and Photonics

56. Wijnen B, Hunt EJ, Anzalone GC, Pearce JM (2014) Open-source syringe pump library, vol 9, p e107216
57. Zajic A, Prvulovic M (2014) Experimental demonstration of electromagnetic information leakage from modern processor-memory systems. IEEE Trans Electromagn Compat 56(4):885–893
58. Zeiler MD, Ranzato M, Monga R, Mao M, Yang K, Le QV, Nguyen P, Senior A, Vanhoucke V, Dean J et al (2013) On rectified linear units for speech processing. In: 2013 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 3517–3521. IEEE