IDEA: Intrusion Detection through Electromagnetic-Signal Analysis for Critical Embedded and Cyber-Physical Systems

Haider Adnan Khan, Nader Sehatbakhsh, Luong N. Nguyen, Robert Callan *Member, IEEE*, Arie Yeredor *Senior Member, IEEE*, Milos Prvulovic *Senior Member, IEEE*, and Alenka Zajić *Senior Member, IEEE*

***Abstract*— We propose a novel framework called IDEA that exploits electromagnetic (EM) side-channel signals to detect malicious activity on embedded and cyber-physical systems (CPS). IDEA first records EM emanations from an uncompromised reference device to establish a baseline of reference EM patterns. IDEA then monitors the target device's EM emanations. When the observed EM emanations deviate from the reference patterns, IDEA reports this as an anomalous or malicious activity. IDEA does not require any resource or infrastructure on, or any modification to, the monitored system itself. In fact, IDEA is isolated from the target device, and monitors the device without any physical contact. We evaluate IDEA by monitoring the target device while it is executing embedded applications with malicious code injections such as DDoS, Ransomware and code modification. We further implement a control-flow hijack attack, an advanced persistent threat, and a firmware modification on three CPSs: an embedded medical device called SyringePump, an industrial PID Controller, and a Robotic Arm, using a popular embedded system, Arduino UNO. The results demonstrate that IDEA can detect different attacks with excellent accuracy (AUC > 99.5%, and 100% detection with less than 1% false positives) from distances up to 3 m.**

***Index Terms*— electromagnetic emanations, electromagnetic side-channel, security of cyber-physical systems, side-channel signal analysis, malware detection**

I. Introduction

Cyber-Physical Systems (CPS) is experiencing an exponential growth, and is expected to be a USD 6.2 trillion market globally by 2025 [1], [2]. While this expected growth is predominantly in healthcare (USD 2.5 trillion) and manufacturing (USD 2.3 trillion), CPSs are ubiquitous, and can impact every aspect of our daily life including critical life supporting medical devices [3]. Furthermore, embedded and CPSs are prevalent in many critical infrastructures including nuclear power generation, military systems, transportation systems, autonomous and unmanned vehicles, communication satellites, etc. [4]. While seamless connectivity helps to ensure enhanced monitoring and control, networked embedded and CPSs are exposed to remote attacks that can cause serious physical and financial consequences and damages.

A variety of CPSs, including Industrial Control System (ICS), smart grid system, and medical devices, have already been targeted by the attackers. A few of these disastrous cyber-attacks are: Stuxnet [5], [6], [7], cyber-attacks on smart grids [8], attacks on medical devices (e.g. pacemakers, insulin pumps, etc. [9], [10], [11], [12], [13]), the infamous Mirai Distributed Denial of Service (DDoS) attack [14], and a variety Ransomware attacks (e.g. WannaCry [15], [16]). A comprehensive review of attacks on CPSs can be found in [17].

Cyber-physical systems consist of heterogeneous building blocks including diverse hardware components such as sensors, actuators and embedded systems, and different proprietary and customized software. As each component brings about its own unique set of vulnerabilities and threats, securing CPSs can be a challenging task. Many CPSs use customized software and hardware, which are often difficult to update or upgrade [18]. In addition, the state of the art malware detection techniques such as malware signatures [19], [20], sandboxing [21], [22], hardware support [23], [24], [25], [26], machine learning [27], [28], and dynamic analysis [29], [30], [31] require substantial computational power. As CPSs often have severe resource, power, and cost constraints, current security solutions may be impractical or inadequate for most CPSs [32]. Furthermore, attackers often hijack the control of the victim device, and may disable or even co-opt the monitoring system. Hence, an isolation between the device and the monitoring system is preferable, especially for critical infrastructures or high-assurance systems.

To address these issues, we propose a novel framework called IDEA that uses electromagnetic (EM) side-channel signals to detect malicious activity on CPSs. The first step in IDEA is to record EM emanations from an uncompromised device to create a baseline dictionary of signal fragments which correspond to the normal behavior of the device. Then, IDEA continuously monitors the target device's EM emanations,

This work has been supported, in part, by NSF grants 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.

Haider Adnan Khan, Luong N. Nguyen, Robert Callan, and Alenka Zajić are with the School of Electrical and Computer Engineering, Nader Sehatbakhsh and Milos Prvulovic are with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, USA, and Arie Yeredor is with School of Electrical Engineering Tel-Aviv University, Tel-Aviv, Israel.

comparing the observed EM emanations against the baseline dictionary. When no malware is present, the device's EM emanations match the entries in the baseline dictionary well. If, however, the observed EM emanations deviate significantly from the entries in the baseline dictionary, we report this as an anomaly which is potentially caused by a malware.

To evaluate IDEA, we port different malware behaviors such as a DDoS cyber-attack, a Ransomware attack, and a source code modification on an Intel-Altera's FPGA Nios-II softcore. In addition, we implement three CPSs, a medical embedded device called *SyringePump*, a proportional-integral-derivative (PID) controller for an industrial soldering iron, and a robotic arm for an assembly line, with Arduino UNO - a popular embedded system. We exploit a buffer-overflow + control-flow hijack attack, an advanced persistent threat (APT), and a firmware modification attack on these CPSs respectively. Experimental evaluation reveals that IDEA can detect DDoS and Ransomware malware with a 100% accuracy (with no false positives), and stealthier code-modification with an Area Under the Curve (AUC) > 97.5% from distances up to 3 m. Furthermore, IDEA can successfully detect all instances of attacks on the implemented CPSs without reporting any false positive.

Finally, we evaluate IDEA with insertions of unknown (untrained-on) snippets of signals into an original reference signal to find the smallest insertion that can be reliably detected. The results show that IDEA can detect intrusions that consist of roughly 200 instructions on FPGA, and roughly 800 instructions on an Internet of Things (IoT) development board, with a 100% accuracy and zero false positive.

This approach for external monitoring of intrusion on devices used on critical infrastructures has several advantages:

1) **Non-intrusive Monitoring**: The target device is not perturbed or modified in any way. The monitor does not impose any overhead, nor does it use any resource on the monitored device. In fact, the target device can be monitored from a distance without any physical contact.
2) **Isolation**: The monitor is isolated from the target device. Hence, the integrity of the monitor cannot be compromised even when the monitored device itself is completely compromised.
3) **Zero-Day Protection**: IDEA identifies malicious activity using the trusted references only, without any a-priori knowledge of malware signatures or vulnerabilities. This means that no training on malware or anomalous behavior is needed, and ensures protection against zero day attacks and obviates the need for regular updates for new malware signatures.

The rest of this paper is organized as follows: Section II states the envisioned threat model, Section III discusses the related work, Section IV details our method for intrusion detection, Section V presents experimental setup and evaluations, Section VI discusses limitations of IDEA and directions for future research, and finally Section VII presents some concluding remarks.

II. Threat Model

IDEA is an external monitoring system for high-assurance CPSs (such as embedded medical devices), and can detect execution of malware through EM side-channel of the target device. The envisioned threat model in this paper involves the following assumptions:

1. The monitoring framework (IDEA) has **no** a priori knowledge about the nature of the attack or its EM signature(s) and only relies on the signatures for the monitored application itself. We assume that IDEA always has correct reference models for malware-free signatures of the monitored applications and these models are stored in IDEA and can not be compromised.

2. The adversary has physical and/or remote access to the target device, and has a prior knowledge of the device and its software. The attacker can thus exploit any vulnerability (e.g. a buffer-overflow) to execute a malicious activity on the system by either launching a separate thread/process and starting a potential cyber-attack (e.g. DDoS) or modifying/re-using the existing application to disrupt or change the original functionality of the targeted system (e.g. control-flow hijack). Furthermore, the adversary can even modify the system's source code and libraries and/or reprogram the system to start a malicious activity. However, as mentioned earlier, IDEA does not know anything about the nature of the attack and only reports an error if an anomaly is detected which may be caused by an actual attack (true positive), or it may not (false positive).

III. Related Work

Traditionally, attackers have exploited unintentional electromagnetic leaks [43], [44], [45] and other analog side-channel signals such as power consumption [46] or acoustic emission [47], [48] to extract sensitive information from victim systems. Researchers have also demonstrated methods for systematically identifying and quantifying EM side-channel signals [49], [50]. Apart from cryptographic key extraction, researchers have also leveraged EM side-channels for hardware Trojan detection [51], [52], and for enhanced physical authentication [53].

More recently, researchers have proposed several approaches to monitor a device's power fluctuations for malware detection (e.g. [33], [34], [36], [37], [54], [55]). For instance, VirusMeter [33] monitors battery power usage to identify "long-term" mobile malware, while [34] measures similarities between power signatures to detect energy-greedy malwares. Furthermore, researchers have leveraged power consumption monitoring for integrity assessment of Software Defined Radios (SDR) [35] and for malware detection in embedded medical devices [36]. Power Finger-Printing Inc. [35] measures processors power consumption and compares against stored trusted signatures for integrity assessment of SDRs, and WattsUpDoc [36] extracts statistical and spectral features from dynamic power consumption to identify anomalous or malicious activity on embedded medical devices. Liu *et al.* [37] provide code execution tracking based on the power signal using an HMM model to recover most likely executed

Ref.	Monitored Side-Channel	Device Under Test (DUT)	Description	Performance	Detection Algorithm
[33]	Power Consumption	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% false positive rate	compares power consumption through machine learning
[34]	Power Consumption	PDA (HP iPAQ)	Malware detection	detects energy greedy malwares with 99% true positive rate and less than 2% false positive rate	compares power signatures with χ^2 distance
[35]	Power Consumption	Software Defined Radio	Integrity assessment	detects deviation in execution	correlates power signatures
[36]	Power Consumption	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
[37]	Power Consumption	8051 MCU (STC89C52)	Control-flow integrity	99.94% for recognizing instruction types and 98.5% for recognizing instruction sequence	leverages HMM and Viterbi to recover instruction types and sequence during execution
[38]	EM Emanation	FPGA (Altera Cyclone II)	Software profiling	profiles software with 94% accuracy	uses depth-first tree search using control flow graph
[39]	EM Emanation	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 short time Fourier transform and KS test
[40]	EM Emanation	PLC (Allen Bradley)	Control-flow integrity	98.9% accuracy (AUC)	uses a neural network to detect legitimate PLC executions
[41]	Telemetry data	Industrial Control System	Malware detection	detects network intrusion with 94.3% accuracy	exploits decision tree based algorithm
[42]	System behavior	Medical CPS	Malware detection	detects intrusion with 92.4% detection rate with 0.66% false positives	exploits rule based behavior specification
IDEA	EM Emanation	FPGA & A13-OLinuXino board & Arduino UNO	Malware detection	detects intrusions larger than 200 instructions with 100% true positives and 0% false positives	compares EM emanations with a reference dictionary with Euclidean distance

TABLE I

COMPARISON OF RELATED WORK WITH THE IDEA IN TERMS OF TYPE OF SIDE-CHANNEL, TYPE OF DEVICE, SOFTWARE OR HARDWARE INTRUSION, AND PERFORMANCE.

instruction sequence with a revised Viterbi algorithm. While these works can be very effective in some scenarios, they are ineffective when the immediate access to the device is not possible (e.g. due to packaging) and/or the monitor has to be placed in some distance from the device (unlike using EM). Moreover, another advantage of using EM over power consumption is that EM signals usually have much more bandwidth than power traces, i.e. EM signals may provide much more information per unit time and provide a better resolution (and accuracy) to detect very small changes (e.g. firmware modification) in high-speed systems.

In addition, recent research has exploited EM side-channel signals for profiling of software execution "as-is", without any hardware or software modifications or instrumentation [38], [56]. Zero Overhead Profiling (ZOP) [38] exploits signatures of emanated EM signals for software profiling. To profile a program, it performs a depth-first search (DFS) through the program's control flow graph to determine the lowest-cost path from the tree's root (the entry of the program) to a leaf node (any exit of the program). While ZOP [38] achieves 94% accuracy for profiling acyclic paths, DFS is prone to error propagation (i.e. any false prediction is likely to lead to a cascade of false predictions). As a consequence, ZOP can generate a large false positive rate. Furthermore, ZOP needs access to the source code to insert markers for training, and requires complete knowledge of the program's control flow graph for monitoring. All of these severely limit its practicality for anomaly/malware detection. In contrast, Spectral profiling [56] observes that periodic program activities, such as loops, often generate periodic EM side-channel signals, which in turn generate spectral peaks at the frequencies that corresponds to the loop's per-iteration durations. Thus [56] exploits Short Time Fourier Transform (STFT) to identify such spectral peaks and uses them for loop-level profiling of program execution. Moreover, [39] extends the spectral profiling, and exploits the spectral peaks for intrusion detection. Any injection of instructions inside an existing loop alters the loop's iteration time, and hence causes a deviation or shift in the frequency of the spectral peak, which [39] exploits for intrusion detection. While this approach can accurately detect changes as small as two instructions inside a loop in the program, it can only detect very large deviations (>500,000 instructions) outside loops.

In contrast, IDEA starts with a training set of trusted executions of a program to be monitored and learns a dictionary of fixed length "words" or windows of EM signals. The monitoring consists of splitting the signal into same-length windows and matching them against the dictionary, and producing a dictionary-based reconstruction of the signal. IDEA then flags large deviations between the observed and reconstructed signal as anomalous (e.g. malware). So, the training can be accomplished on "live" runs (without any markers or any other changes) and without any access to the source code or the control flow graph. In addition, IDEA is free

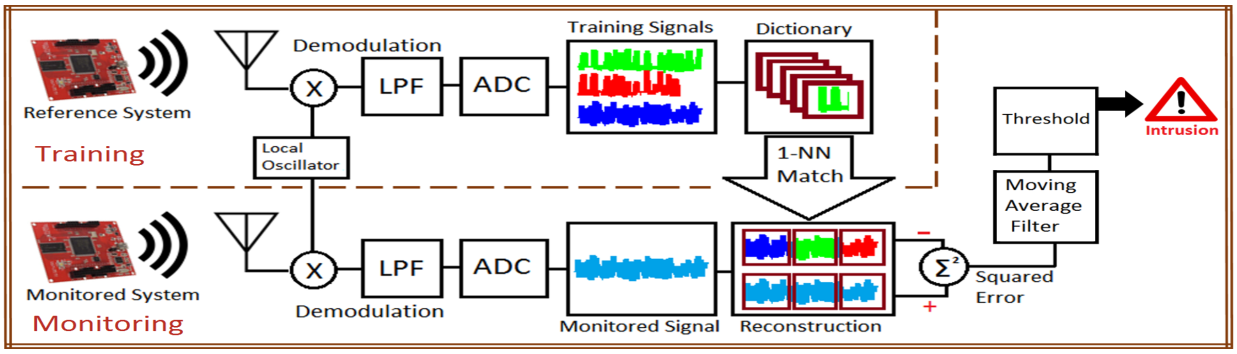


Fig. 1. Overview of IDEA framework.

from error propagation. Hence, a few rare false mis-matches do not have a compounding impact on the overall performance of the algorithm. As a result, IDEA achieves 100% detection rate without any false positives for detecting malwares such as Ransomware and DDoS. The performance is even more impressive considering that IDEA is trained and monitored with different user-inputs (i.e. the training and the monitored executions followed different control flow paths through the program). So, there is a considerable variability between the training and the monitored program execution, which renders the anomaly detection more difficult, and the performance evaluation more realistic and reliable. Unlike some existing methods (e.g. WattsUpDoc), IDEA does not require training on the malware itself, and hence is effective against zero-day attacks. Furthermore, IDEA is equally effective both inside and outside loops.

Apart from analog side-channels, telemetry data analysis has also been exploited for network intrusion detection in ICS [41]. Likewise, behavior rule based intrusion detection system for critical medical CPSs has been demonstrated [42]. However, both of the systems struggle to achieve high detection rate at low (or near-zero) false positive rate, which is a prerequisite for monitoring any critical CPS. For ease of reference, we compare all these methods with the proposed IDEA in Table I.

IV. IDEA Overview

Figure 1 illustrates the workflow of IDEA. The signals in both training and monitoring phase are demodulated, low-pass filtered, and sampled before they are subjected to the main part of IDEA signal processing, which exploits techniques similar to template-based pattern matching to identify anomalous (hence, potentially malicious) activity during the program execution. In the training phase, IDEA learns a dictionary of reference EM signatures or "words" by executing trusted programs on an uncompromised reference device. Next, in the monitoring phase, it continuously monitors the target device's EM signal by matching it and reconstructing it using the dictionary. When the reconstruction error is above a predefined threshold (i.e. there is a significant deviation from the reference EM signatures), IDEA reports an anomaly (intrusion). The rest of this section describes IDEA in more detail.

A. AM Demodulation

Unintentional EM emanations occur at various frequencies, but of particular importance is the frequency band centered around the clock frequency of the processor, a.k.a. Central Processing Unit (CPU). This is because this frequency band contains signals that are primarily a function of the instruction sequence executed by the CPU. Each processor cycle, the CPU draws a current which is a direct result of the instruction(s) being executed. Much of this instruction-dependent current is drawn by the CPU clock circuitry and by circuitry which does new computations (i.e. switches on and off) every CPU clock cycle. This creates a strong current at the CPU clock frequency which acts as a carrier modulated by the clock-to-cycle variations in program activity (i.e. executed instructions). These currents flow through wires within the processor and on the device's printed circuit board (PCB). At CPU and memory clock frequencies (and their harmonics) the EM emanations created can propagate far enough to be observed with a high signal to noise ratio [57]. When observed this way, the emanating device has much in common with a communications system since the device is a transmitter which (inefficiently and unintentionally) transmits a message signal carrying information about program activity using an amplitude modulated carrier (i.e. the clock signal). We can then receive and demodulate this signal using wireless communications techniques. All EM signals, both in the training phase and in the monitoring phase, are AM demodulated at the processor clock frequency, low-pass filtered with an anti-aliasing filter, and sampled before being sent for signal processing.

B. Training Phase: Dictionary Learning

The training phase consists of learning a dictionary of EM signatures through the execution of trusted programs on a reference device. We execute trusted programs on an uncompromised device, and observe and record the corresponding EM signals. We use different inputs to execute different control flow paths as described in [38]. Ideally, we would like to observe all possible control flow paths. However, in a practical scenario, this may require too many inputs (hence, too many training examples). For example, a twenty level nested IF ELSE condition will have $2^{20} = 1048576$ different execution paths. Nevertheless, we aim at observing most control flow execution paths, and try to ensure that even if there are unobserved control flow paths, they are either highly unlikely or relatively brief. These goals are the same as those that guide program testing, so program inputs created to provide good test coverage of a program are highly likely to also satisfy the needs of IDEA training. Once we have the training signals, we learn the dictionary words using the following process.

1) Learning Words: The demodulated EM signal is split into multiple overlapping short-duration windows that are recorded as dictionary entries or "words". These words correspond to the EM signature of the underlying program execution. All dictionary words have the same word-length l. Each word is shifted by s samples from the previous one. When s is small, we end up with densely overlapping words. Consequently, we learn a dictionary with large number of words with slight variability (i.e. shift). This can help to achieve shift-invariant pattern matching. Shift-invariance is necessary because of hardware events, such as cache misses, that delay (or shift) the subsequent execution (and the corresponding EM signal). A cache miss can potentially occur at many different points of the program execution. It is neither practical nor even possible to generate a training set with all possible scenarios of cache hits or misses. Therefore, creating a dictionary with densely overlapped words can help us match EM signatures better under variability due to hardware activity.

2) Word Normalization: All dictionary words are post processed by mean subtraction and scale normalization:

$$\mathbf{w} = (\mathbf{w_0} - \mu)/\sigma \tag{1}$$

Here, $\mathbf{w}$ denotes the normalized word, μ is the mean and σ is the standard deviation of the unnormalized word $\mathbf{w_0}$.

This normalization improves the matching accuracy by ensuring that the matching is based on the pattern (i.e. the relative shape of the waveform) rather than on the actual amplitude. For instance, due to different positions of the antenna, the distance from the processor may change between the training and the monitoring phase. Hence, the training and the monitored signals can have different scales or amplifications. This normalization nullifies such issues.

3) Dictionary Reduction through Clustering: Next, we apply clustering to reduce the number of dictionary entries. All applications have loops, which tend to generate repetitive EM patterns. Likewise, the same control flow paths are often reiterated at different points of the execution, and generate similar EM patterns. Consequently, the reference dictionary can have a large number of words or patterns that are very similar, and correspond to the same code execution. The objective of clustering is to assign similar words or EM patterns into a single cluster, and exploit the cluster centroid as the representative of the cluster. Using cluster centroids as dictionary words improves the computational efficiency by reducing the number of dictionary entries.

As the number of clusters k (i.e. the number of unique EM patterns) is not known a priori, popular clustering algorithms such as k-means can not be used for the dictionary reduction.

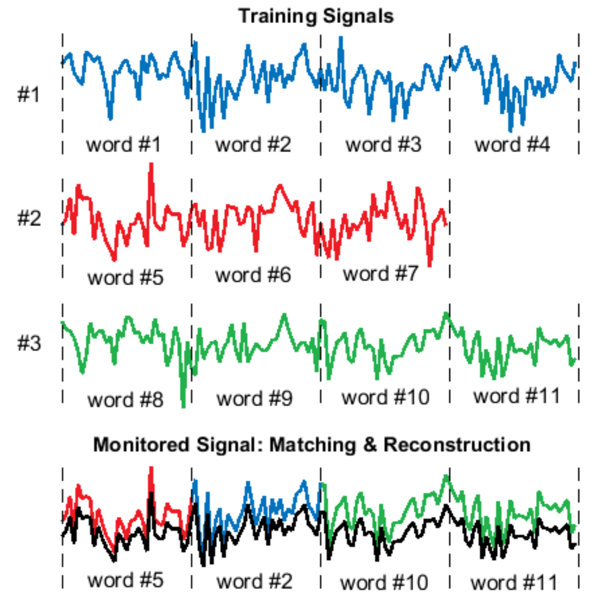


Fig. 2. An example of verifying the monitored signal using dictionary words. The monitored signal (shown with black lines) is verified against training (red, blue and green) signals.

Instead, we use a threshold based clustering, where threshold t is used as a parameter. Given a cluster centroid c_i, the algorithm proceeds by alternating between two steps:

Assignment Step: Assign each unassigned word w_p whose Euclidean distance from the centroid c_i is less than the threshold t to the cluster S_i.

$$\mathbf{S_i} = \{w_p : \|w_p - c_i\|^2 < t \ \wedge \ w_p \notin S_j \ \forall j, \ 1 \leq j < i\} \tag{2}$$

Update Step: Update the cluster centroid c_i by averaging all members of cluster S_i.

$$\mathbf{c_i} = \frac{1}{|S_i|} \sum_{w_p \in S_i} w_p \tag{3}$$

Once the assignments no longer change, select a new cluster centroid randomly from the words that have not yet been assigned to any cluster. The algorithm converges when all words are assigned to a cluster.

C. Monitoring Phase: Intrusion Detection

In the monitoring phase, the EM signal is continuously monitored and matched against the dictionary, and anomalous activity is reported when the monitored signal deviates significantly from its dictionary-based reconstruction.

1) Matching and Reconstruction: The monitored EM signal is split into windows and matched against the dictionary using the 1-Nearest Neighbor algorithm [58], with Euclidean distance as the distance metric. The signal is then reconstructed by replacing each window with its best-match dictionary word. This is illustrated in Figure 2. The entire signal can be reconstructed by concatenating these best-match words that correspond to the signal's sequence of windows, and this reconstructed signal is then used for anomaly (intrusion) detection.

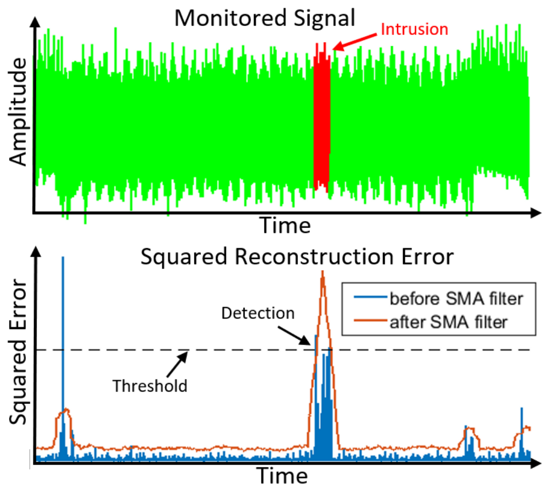


Fig. 3. Intrusion detection from reconstruction error: the squared reconstruction error is passed through a SMA filter to reduce false positives.

2) Detection: The detection continuously compares the monitored signal with the reconstructed signal. Specifically, we compute the per-sample reconstruction error as the squared difference between samples of the monitored and the reconstructed signal:

$$e(n) = (x(n) - y(n))^2 \tag{4}$$

Here, $x(n)$, $y(n)$ and $e(n)$ denote the monitored signal, the reconstructed signal, the squared reconstruction error signal, respectively and n denotes the sample-index.

The detection algorithm is illustrated in Figure 3. Due to considerations presented in Section D.3 below, we then apply an L-samples long Simple Moving Average (SMA) filter to the signal $e(n)$, yielding the filtered signal $\tilde{e}(n)$:

$$\tilde{e}(n) = \frac{1}{L}\sum_{i=0}^{L-1} e(n-i) \tag{5}$$

Finally, we set a threshold on $\tilde{e}$, and whenever this threshold is breached, we report an intrusion. Figure 4 illustrates how reconstructed curves based on IDEA algorithm match the original program execution vs. the execution with malware. From the plot we can observe that reconstructed signal deviates significantly from the execution with malware compared to the deviation form the normal execution, hence allowing us to detect the malware.

D. System Parameters

The performance of IDEA depends on a number of system parameters, such as word-length l, word-shift s, and order of the SMA filter L. The rest of this section is a discussion of how these system parameters are chosen.

1) Word-Length: The word-length l has an impact on the performance of the proposed system. The optimal word-length l is a tradeoff between confidence in a match (the longer the word, the more reliable the match) and likelihood of a good match (the shorter the word, the more likely it is to find a dictionary word that matches it well).

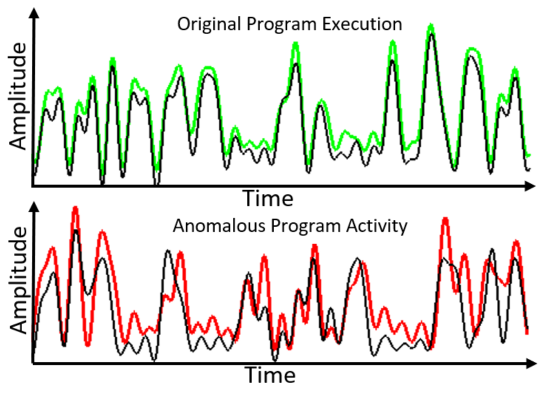
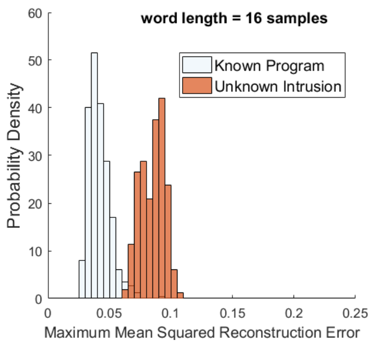


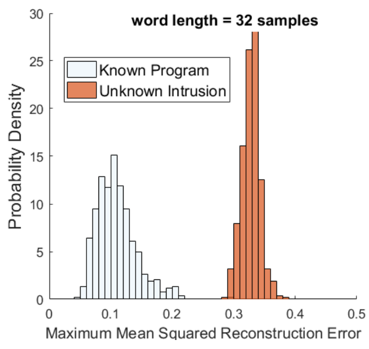
Fig. 4. Top green curve: original program execution; Bottom red curve: program execution with intrusion; Black curve: IDEA-reconstructed signal. Note that black curve better matches with green than with red curve.

Therefore, the word-length l should be long enough to avoid good matches among a set of unrelated signals. To achieve that, a word in a dictionary should represent a relatively long sequence of processor instructions or hardware activity. This ensures that it is unlikely that a non-trained program will produce a sequence of executed processor instructions that is an excellent match for any dictionary entry of a trained program.

On the other hand, the word-length l should be short enough so that random events, such as cache misses or interrupts, do not preclude good matches. For example, if we use a word that is very long, it will be difficult to find good matches in the dictionary of any reasonable size. This is because different inputs and hardware activities result in different signals when executing the same code, a reasonable-sized dictionary can contain only a small subset of the possible valid words, and a (long) window of the monitored signal will likely exhibit many input-dependent and hardware behaviors that do not match any of the dictionary words. By using a smaller word length we limit the number of word variants that can be produced, which reduces the dictionary size required for "full" coverage of these variants. Even when dictionary coverage of word variants is not complete and a window of the monitored signal has a set of input-dependent and hardware behaviors that is not represented in the dictionary, a smaller word length increases the probability that the dictionary contains a word that matches the window for most of its duration, and thus still produces a reasonably small reconstruction distance.

In order to estimate the optimal word-length l, we insert snippets of untrained signals into the trained or trusted reference signals. First, we record EM signals by executing a benchmark program with different inputs. Next, we follow a 10 fold cross validation to test each of these signals with and without an "untrained" insertion from a different benchmark program. Here, signals without insertion represent class 0 or "known", while signals with insertion correspond to class 1 or "intrusion". Figure 5 shows the histograms corresponding to "known" and "intrusion" with different word-lengths. For $w =$ 16 samples, the Maximum Mean Squared Reconstruction Error (MMSRE) is low for both known (or trained) and intrusion





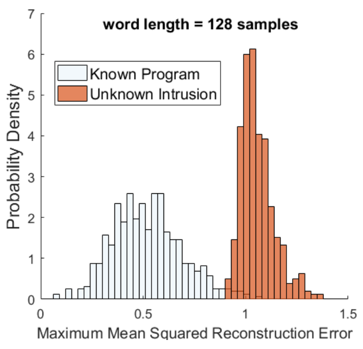


Fig. 5. Histogram for maximum mean squared reconstruction error with different word-lengths.

(or untrained) signals (i.e. even an untrained signal can be matched with words in the dictionary). As a result, the two histograms overlap. However, for $w = 32$ samples, MMSRE corresponding to the known signal is significantly lower than that of the intrusion, and there exists a clear threshold between the two classes. When the word-length is much larger, i.e. $w = 128$ samples, MMSRE for both known and intrusion signals gets much higher (i.e. even a trained signal cannot be matched with low Euclidean distance). Hence, the two histograms cannot be separated anymore.

These experimental evaluations reveal that for any intrusion larger than 256 samples, IDEA can achieve Area Under the Curve (AUC) better than 0.9995 on the Receiver Operating Characteristic (ROC) curve for any word-length between 32 to 64 samples. If not specified differently, the rest of the paper assumes word-length $w = 32$ of samples. This corresponds to 5μs of execution time, which is about 250 processor clock cycles on the FPGA board.

2) Word-Shift: Another parameter that impacts the performance of IDEA is a word-shift. Each "word" in the dictionary has to be shifted some number of samples from the previous one in order to compensate for hardware activities such as cache hit or miss. We estimate the optimal word-shift s through experimental evaluation. Again, we exploit a 10 fold cross validation in which snippets of insertions from an "untrained" program are treated as intrusion. Figure 6 shows the ROC curve for different word-shifts for an intrusion of 128 samples. It is clear that $s = 1$ performs the best, and the larger s results in smaller AUC. This results is intuitive as $s = 1$ mimics shift invariant signal matching most closely. However, it should be noted that, the detection performance for $s = 2$ is comparable to that of $s = 1$. Hence, $s = 2$ can be exploited to reduce the computational requirements. The number of entries in the dictionary would be roughly halved for $s = 2$ compared to $s = 1$. So, the memory requirement would be halved, and consequently, so would be the computational time. Nevertheless, as we intend to highlight the performance of our system, we use $s = 1$ in the remainder of this paper.

3) Filter Order: In order to justify the use of the SMA filter and to determine its optimal length, consider the following detection problem. Let $\epsilon(n) \triangleq x(n) - y(n)$ denote the error signal, defined as the difference between the monitored

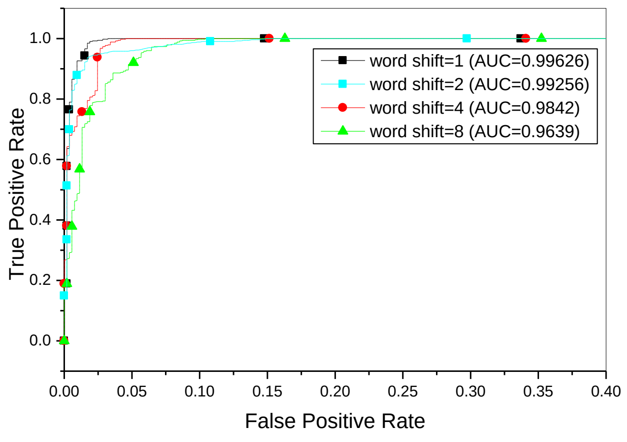


Fig. 6. ROC curves for intrusion detection with different word-shift.

and reconstructed signals. Observing an L-samples segment thereof $\epsilon(n-L+1), \ldots, \epsilon(n)$, we wish to decide whether:

- H_0 : This is a valid program execution segment; or
- H_1 : This is an intrusion code segment.

We begin by attributing two simplified statistical models to the error signal under each hypothesis: $\epsilon(n)$ is assumed to be an independent, identically distributed (iid) zero-mean Gaussian process, but with a different variance under each of the different hypotheses:

- $H_0 : \epsilon(n) \sim \mathcal{N}(0, \sigma_0^2)$
- $H_1 : \epsilon(n) \sim \mathcal{N}(0, \sigma_1^2)$

where $\sigma_0^2 < \sigma_1^2$ are fixed variances (presumed known, for now). The Likelihood Ratio Test (LRT) for deciding between the two hypotheses then takes the form:

$$\frac{f(\epsilon(n-L+1), \ldots, \epsilon(n)|H_1)}{f(\epsilon(n-L+1), \ldots, \epsilon(n)|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \eta, \tag{6}$$

where $f(\cdot|H_i)$ denotes the conditional joint probability distribution function (pdf) of the observations given $H_i, i = 0, 1$, and η is a threshold value. Substituting Gaussian distributions and taking the log we get

$$-\frac{L}{2}\log(\sigma_1^2) - \frac{1}{2\sigma_1^2}\sum_{i=0}^{L-1}\epsilon^2(n-i) + \frac{L}{2}\log(\sigma_0^2) + \frac{1}{2\sigma_0^2}\sum_{i=0}^{L-1}\epsilon^2(n-i) \underset{H_0}{\overset{H_1}{\gtrless}} \log(\eta), \quad (7)$$

which can be further rearranged as

$$\frac{1}{L}\sum_{i=0}^{L-1}\epsilon^2(n-i) \underset{H_0}{\overset{H_1}{\gtrless}} \tilde{\eta} \triangleq \frac{\kappa\sigma_0^2}{\kappa-1}\left(\log(\kappa) + \frac{2}{L}\log(\eta)\right), \quad (8)$$

where $\kappa \triangleq \sigma_1^2/\sigma_0^2 > 1$ denotes the ratio between the two variances. This means that the average of squared samples of $\epsilon(n)$ over the L-samples observation interval is to be compared to some threshold $\tilde{\eta}$. This average is precisely $\tilde{e}(n)$, the output of a length-L SMA filter in (5), with $e(n) \triangleq \epsilon^2(n)$.

To determine an optimal value for L, note first that the mean and variance of $\tilde{e}(n)$ under H_i are (resp.) σ_i^2 and $\frac{2}{L}\sigma_i^4$, $i = 0, 1$. We note further, that if L is sufficiently large (say, larger than 10 or so), considering the Central Limit Theorem, the distribution of $\tilde{e}(n)$ under each hypothesis is approximately Gaussian. Consequently, the false positive and false negative probabilities can be shown to decay monotonically in L. Therefore, to have them minimized, L should take the largest possible value that does not breech the H_1 model. Namely, L has to be chosen as the full length of the shortest possible intrusion. Since that length is not known a priori, we chose to set the filter order equal to the shortest length of insertion that we intend to detect. If not specified differently, we use $L = 256$ as the filter length.

Note that in reality the situation is somewhat more complicated than described above. First, the iid Gaussian signal model for $\epsilon(n)$ is inaccurate, as its samples are expected to be correlated and not necessarily Gaussian distributed; The variances σ_1^2 and σ_0^2 would rarely be known; And reconstruction errors may be very high for a few brief, sporadic segments, even under H_0, resulting either from lack of full coverage in the training phase (i.e. lack of appropriate training examples that follow the same control flow path as the monitored signal) or from the variability of hardware activities between the training signal and the monitored signal. These complications certainly undermine any claim of optimality of the LRT in this case. However, the rationale behind the resulting test remains valid, justifying the use of the SMA filter and the choice of L. For example, the possibility of short occurrences of large errors under H_0 would merely increase the mean and variance of $\tilde{e}(n)$ under H_0, thereby increasing the false positive rate. Nevertheless, the dependence of this rate on L remains monotonically decreasing, still supporting our choice of the largest possible L that does not breech H_1.

V. Experimental Evaluation

In this section we present our experimental results on detecting several different types of malware on different applications and embedded systems. It is important to emphasize that IDEA is not limited to these applications and/or malware, but fundamentally can be applied to any system that has observable EM emanations.

A. Experiments with Different Malware Behaviors

To show the effectiveness of IDEA on detecting different malware behaviors, we selected 3 applications from the SIR repository [59]: (*Replace*, *Print Tokens*, and *Schedule*) and implemented three common types of embedded system malware payloads (DDoS attacks, Ransomware attacks, and code modification) on an Altera DE-1 prototype board (Cyclone II FPGA with a NIOS II soft-processor) while executing any of these SIR applications. We selected applications from SIR repository because they are relatively compact (allowing manual checks of our results to get a deeper understanding of what affects the IDEA accuracy), are commonly used to evaluate performance of techniques that analyze program execution, and have many program inputs available (each taking a different overall path through the program code), so we can use disjoint sets of inputs for training and monitoring and yet have a large number of inputs for training (to improve code coverage) and monitoring (to obtain representative results).

For *DDoS* cyber-attack, we assume that exploiting a vulnerability (e.g. buffer-overflow), the control-flow will be diverted (e.g. using code-reuse attack) to a code that sends DDoS packets in rapid succession, without waiting for reply from the target. After sending a burst of packets, the malware returns execution for a while to the original application, so the device continues to perform its primary functionality. In the case of an embedded device, e.g. Altera DE-1 (Cyclone II FPGA) board, there is no traditional network (e.g. Ethernet) port. So we instead implement the packet-sending activity using the JTAG port.

For *Ransomware* attack, we implement a Cryptoviral [60], that encrypts the victim's data, and demands ransom in return for the decryption key. We assume that the attacker inserted the malicious code through firmware modification. We used *Advanced Encryption Standard*, AES-128 for encryption. Encryption of large files is time-consuming, and can be easily detected by IDEA. To make things more challenging, the *Ransomware* we implement encrypts only one encryption block of AES-128, which corresponds to a 16-byte data. Note that more secure cyphers, e.g. AES-256, have larger encryption blocks, and are thus easier to detect.

For *code modification*, which is the basis of an important class of malware (APT and firmware modification attacks), we assume that the malware has already successfully modified the source code of the program, and that the goal of IDEA is to detect when this modified code executes. For example, in the *Replace* benchmark, there is a function called *subline()*, which is used to search for words in an input string. In a scenario where authorities use this code to look for names in intercepted communication, the attacker's modification of this code would prevent any word that begins with specific initial letters from being reported.

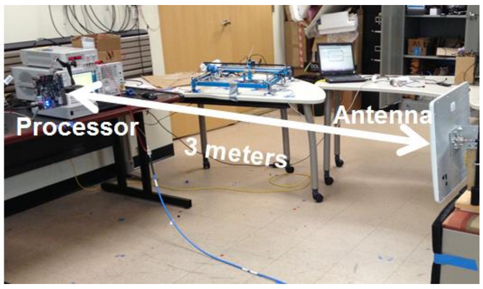


Fig. 7. Measurement setup used to collect EM traces from distance.

Setup: To observe the EM emanations, we used a 2.4-2.5 GHz 18dBi panel antenna that was placed 1 m, 2 m, and 3 m away from the device (see Figure 7). We used a (fairly expensive) Agilent MXA N9020A spectrum analyzer to demodulate and record the EM emanations, primarily so we can have more control and flexibility in our investigations, but in the next section (V.B) we will show that a sub-$1000 SDR receiver (USRP-B200mini [61]) can be used instead. To assess IDEA's performance, we have used program input sets from [59]. Malware injections occur in roughly 30% of the runs, randomly selected, and in each injection, the injection time (relative to the beginning time of the run) is drawn randomly from a set of 10 predefined values.

Detection Performance: In order to evaluate the performance of the intrusion detection system, we apply a 10 fold cross validation. We execute each benchmark program with different inputs. Some of these executions are infected with malicious intrusions. As the system does not assume any a priori knowledge about the threat models or the intrusions, we do not use any of these infected executions for training the system. Likewise, we select all the system parameters through experiments with "untrained" insertions from a different benchmark program, without using any actual infected signal. The non-infected signals are randomly divided into 10 roughly equal-sized subsets. In each fold, we test one of these subsets along with the infected signals, while the rest of the non-infected signals are used for training the system. The objective of the experiment is to measure how successfully the system can identify and differentiate between the infected and the non-infected signals.

We follow this procedure with three different benchmark programs, namely *Print Tokens*, *Replace* and *Schedule*. For *Print Tokens*, a total of 637 executions were recorded, out of which 142 were infected with different variants of malware - 68 Ransomware, 66 DDoS, and 8 code modification. In case of *Replace*, we have recorded 691 executions including 138 infected ones out of which 68 were Ransomware, 65 were DDoS, and 5 were code modification. For *Schedule*, we collect 681 executions where 144 of them were infected with 68 Ransomware, 67 DDoS, and 9 code modification.

Figure 8 plots the ROC curves for IDEA intrusion detection on different benchmark programs with different variants of intrusions. It shows excellent performance

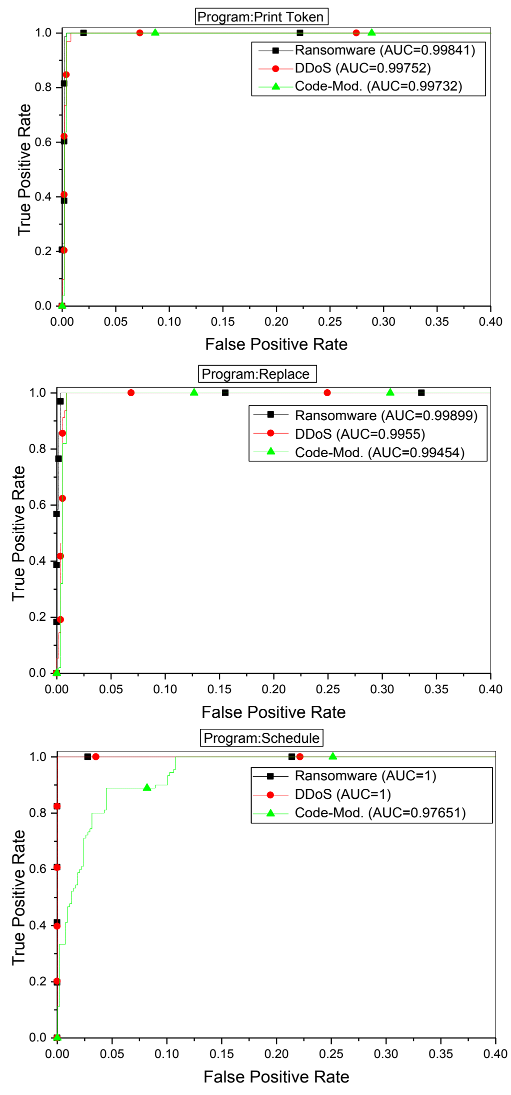


Fig. 8. Receiver Operating Characteristic curves for intrusion detection on three different benchmark programs for different variants of malwares.

in all three benchmark programs, with Area Under the Curve (AUC) very close to 1 for Ransomware and DDoS malwares. Code modification creates a much smaller change in the program's execution, thus is much harder to detect. However, IDEA still detects it with an AUC $>$ 97.5%. Specifically, detection of all code modifications is achieved by tolerating a false positive rate of no more than 1% in *Print Tokens* and *Replace*, and no more than 12% in *Schedule*.

Detection at Different Distances: Next we test IDEA at three different distances (1 m, 2 m and 3 m) from the monitored device. The results (Figure 9) show that the performance of IDEA remains stable over different distances. In fact, for all three benchmark programs, the performance is quite similar at 1 m and 2 m, with AUC better than 99.5%. The performance degrades somewhat at 3 m, achieving 99% AUC for *Replace* and 98% AUC for *Print Tokens* and *Schedule*. The degradation in accuracy at the 3 m distance is mainly

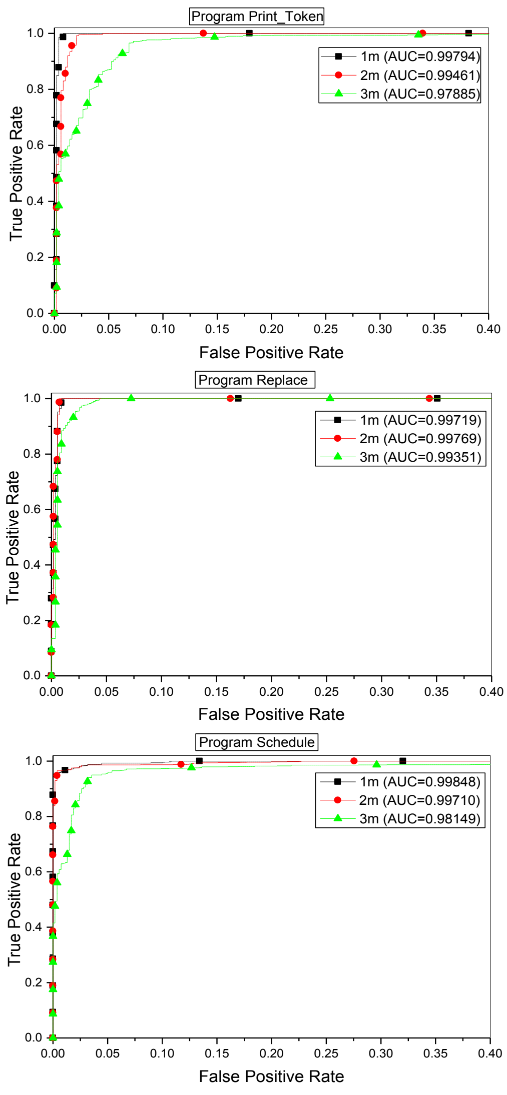


Fig. 9. Receiver Operating Characteristic curves for intrusion detection on three different benchmark programs from different distances.

due to a reduced signal-to-noise ratio (SNR) as the monitored signal weakens with distance, and we believe that these results can be improved by using customized (higher-gain) antennas and low-noise amplifiers.

Detection at Different SNR: To evaluate IDEA in presence of environmental EM noise, we apply Additive White Gaussian Noise (AWGN) to the monitored signal, and test IDEA by monitoring the benchmark application *Replace* at three different SNR (20 dB, 10 dB and 5 dB). The results are shown in Figure 10. Experimental evaluation demonstrates that IDEA is robust against EM noise. In fact, IDEA achieves an excellent performance (AUC > 99.5%) at 20 dB SNR. Furthermore, at 10 dB SNR, IDEA can still detect intrusions with an AUC > 95%. However, at 5 dB SNR, the detection performance degrades to a 67.4% AUC.

In addition, IDEA is inherently robust against out-of-band EM interference from adjacent devices. IDEA monitors the target device at a frequency band around its processor's clock frequency. Any EM interference with frequencies outside this monitored band is blocked by the anti-aliasing filter during the analog-to-digital conversion (ADC). Since each device emits EM signal at its own clock frequency, interference is limited and can be filtered out using signal processing.

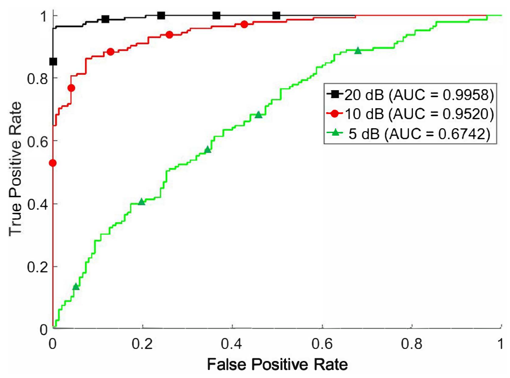


Fig. 10. Receiver Operating Characteristic curves for intrusion detection at different SNR.

B. Experiments with Cyber-Physical Systems

To emphasize on practicality of IDEA and to demonstrate its ability to successfully detect real malware on real CPSs, we use IDEA to monitor three industrial CPSs implemented on a well-known embedded system, Arduino UNO. We use a control-flow hijacking attack, an APT, and a firmware modification attack on these CPSs for evaluation.

The first CPS we use is called a *SyringePump* which is designed to dispense or withdraw a precise amount of fluid, e.g. in hospitals for applying medication at frequent intervals [62], and it is a representative of a medical CPS. The device typically consists of a syringe filled with medicine, an actuator (e.g. stepper motor), and a control unit (Arduino UNO) that takes commands and produces controls for the stepper motor (a sample of this CPS can be found in [62]). We implement a *control-flow hijack* attack on this system by exploiting an existing buffer-overflow vulnerability in a subroutine (*serialRead()*) that reads the inputs which causes the program's control-flow to jump to an *injected* malicious code. We assume that the adversary is interested in disrupting the correct performance of the system by dispensing/withdrawing an unwanted amount of fluid which could cause a significant damage to the patient, thus the injected code causes the syringe to dispense a random amount of fluid. The buffer-overflow is implemented by sending a large inputs to overwrite the stack followed by the address of the "injected" malicious code which overwrites the actual return address of the *serialRead()*.

The second system is a proportional-integral-derivative (PID) controller that is used for controlling the temperature of a soldering iron. This type of system could also be used to control the temperature or any other critical value in other settings, such as a building or an industrial process, and thus is representative of a large class of industrial CPSs. Using a feedback loop and a history of previous temperatures, the

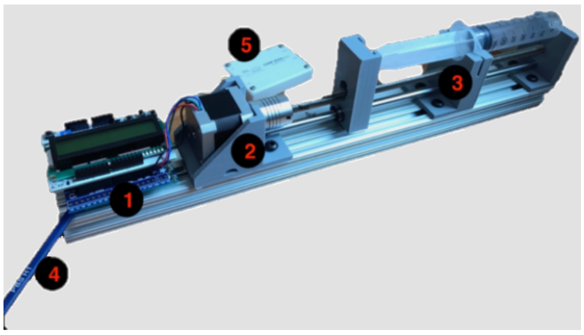


Fig. 11. Experimental setup for the SyringePump: (1) Arduino device with LCD, (2) stepper motor, (3) syringe, (4) magnetic probe and (5) software-defined radio.

system keeps and/or changes the temperature to a desired temperature (a sample of this CPS can be found in [63]). To implement an APT on this application, we assume that the adversary's malware (like in Stuxnet) has already infiltrated the system and can reprogram the device. The adversary's goal is to change a critical value under some conditions, which in turn can cause damage to the overall physical system. In our evaluation, we made a malicious modification to the source code so that the temperature history is altered under a specific condition (e.g. for a specific model number). Consequently, the system will set a wrong temperature. The injected code is only 2 lines of code (i.e. IF(X) THEN LASTTEMPHISTORY = RANDOMVALUE).

The final system in our evaluation is a robotic arm. Robotic arms are often used for manufacturing, and are critical components of many modern factory. Robotic arms typically receive inputs/commands from a user and/or sensors and move objects based on these inputs. There is a growing concern in security of these CPSs since they are typically connected to the network and are exposed to cyber-threats (e.g. [64]). A simple implementation of such a robot can be found in [65]. For this system, we implement a *firmware modification* attack, where we assume that the reference libraries (e.g. library for *Servo*, *Serial*, etc.) are compromised (this can be also considered as a *zero-day* vulnerability). Note that, we assume that IDEA's training contains the "unmodified" version of these library (baseline reference data). In this attack, we modify a subroutine (*writeMicroseconds()*) in Arduino's Servo library [66] by adding an extra *if* condition to change the speed of Servo motor randomly and reprogram the system with this compromised library, assuming that the adversary is interested in causing a malfunction in arm's movement.

Setup. An Arduino UNO with an ATMEGA328p microprocessor clocked at 16 MHz is used to implement the CPSs. A magnetic probe is used to receive EM signals from the device. Figure 11 shows the experimental setup for the *SyringePump*. For all measurements, we use a commercially available SDR receiver (Ettus Research B200-mini) to record the signal. B200-mini costs significantly lower than a spectrum analyzer and makes IDEA a practical option for monitoring security-critical systems. For each CPS, we use 25 randomly selected signals for training and 25 malware-free and 25 malware-afflicted signals for testing.

System	AUC	Malware Type
SyringePump	> 0.999	Control-flow hijack
PID Controller	> 0.999	APT
Robotic Arm	> 0.999	Firmware Modification

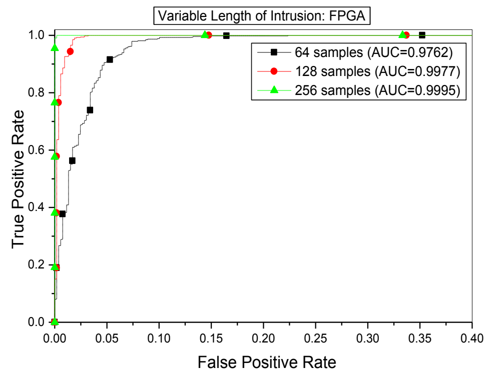
TABLE II
EXPERIMENTAL RESULTS FOR THREE MALWARE ON THREE CPSS.

Detection Performance. Table II summarizes the detection accuracy of IDEA on the three CPSs. As seen in the table, in all cases, IDEA has successfully detected every instances of a malware without reporting any false positive which makes IDEA a promising framework for monitoring critical CPSs, where very small false positive rate while having a high detection accuracy is required. Note that in all cases, the runtime for the malicious code is significantly less ($< 0.01\%$) than the overall runtime of the application.

C. Experiments with IoT Devices

To demonstrate the robustness of IDEA, we also use it to monitor an A13-OLinuXino (Cortex A8 processor) IoT board. Unlike the FPGA-based system that runs the application "on bare metal," this board runs a Linux operating system (OS). The defensive mechanisms already present in the OS make it harder to inject prototype malware activity. Instead, we model malware injection by injecting snippets of signals from a different (not-trained-on) program. For this experiment, we use *Replace* as the reference program, on to which signal-snippets from *Print Tokens* were inserted as anomalous (not-trained-on) signal. This approach also allows injections of any chosen duration, and use of different signals for different injection instances. In contrast, construction of even one short-duration actual malware instance is very challenging. For example, a single packet sent in a DDoS attack, or single-block encryption in Ransomware, lasts much longer than any of our signal-snippet injections.

To allow a direct comparison between our real-malware and signal-snippet injections, we also perform signal-snippet injection experiments on the DE-1 FPGA board. We use 10-fold cross validation, and test signals from a trained benchmark program *Replace*, with or without insertions or intrusions from *Print Tokens*. Figure 12 shows the experimental results. We can observe that intrusions longer than 256 samples (i.e. 200 instructions or 40μs length) on the FPGA are detected with an AUC of 99.95%. For the IoT board, an AUC better than 99.8% is achieved for intrusions with at least 1024 samples (i.e. 800 instructions or 7.94μs length). The difference in duration of the intrusion that is needed to achieve the same AUC on the two devices is mainly due to OS activity that is present on the IoT board and absent on the FPGA board. This OS activity introduces variation in the signals, increasing reconstruction error even for valid executions. This, in turn, raises the reconstruction error threshold for reporting an anomaly at a



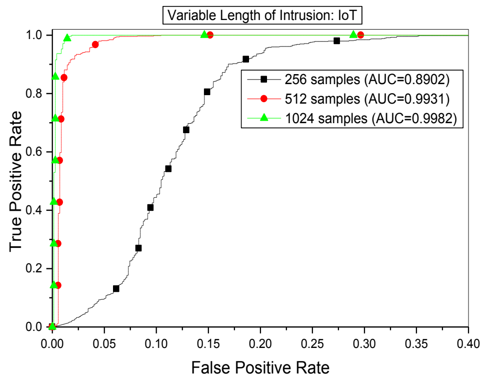


Fig. 12. Receiver Operating Characteristic curves for intrusion detection on a FPGA (top) and on an IoT Device (bottom).

given confidence level, so more anomalous samples are needed to reach this increased reconstruction error threshold.

VI. Limitations and Future Work

A major concern for commercial deployment of the IDEA monitoring system is its cost. However, we envision that IDEA will be deployed to monitor critical and high-assurance CPSs, e.g. critical infrastructures, military systems, hospital equipment etc. In such scenarios, the cost of deployment (e.g. cost of antenna, software-defined radio and signal processing) is offset by the cost of the monitored system and by the cost and consequences of security breach. In addition, deployment of IDEA is relatively simple; IDEA does not make any change to the monitored system, and thus creates no regulatory, safety, or disruption concern for the system.

Another important limitation with IDEA is its scalability. IDEA requires very high training coverage, which is difficult to achieve for larger programs. However, we exploit software engineering techniques that ensure high path coverage for training. Moreover, IDEA stores EM patterns corresponding to normal program activities in a reference dictionary. This dictionary may grow prohibitively large for larger applications. While we use clustering to keep the dictionary size manageable, future work should investigate feature dimensionality reduction techniques (e.g. principal component analysis) to further optimize the dictionary size without sacrificing the detection accuracy. In addition, future research should focus on extending IDEA's capability to monitor multiple devices simultaneously and to monitor multi-core processors.

VII. Conclusions

This paper proposes a novel framework called IDEA that uses electromagnetic (EM) side-channel signals to detect malicious program activity on CPSs. IDEA first records EM emanations from an uncompromised device to establish a baseline of uncompromised EM patterns. Then, IDEA continuously monitors the device's EM emanations, comparing the observed EM emanations against the reference dictionary. When the observed EM emanations deviate significantly from the entries in the reference dictionary, IDEA reports this as an anomaly which could be caused by malware. The proposed method does not require any resource or infrastructure on, or any modifications to, the monitored system itself. To evaluate IDEA's effectiveness, we implement a number of malicious activities such as a *Ransomware*, a firmware modification, and a *control-flow hijack* on three CPSs, and then use IDEA to monitor these CPSs while it is subjected to a malicious-attack. We find that IDEA can detect different components of malware with excellent accuracy from up to 3 m, and find every instances of attacks on CPSs with perfect accuracy.

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: 2018-03-01.

[2] M. M. Chui and R. Roberts, "Mckinsey quarterlythe internet of things," 2010.

[3] R. Richards, "High-assurance cyber military systems (hacms)," *DARPA. mil*, 2016.

[4] E. Colbert, "Security of cyber-physical systems," *Journal of Cyber Security and Information Systems*, vol. 5, no. 1, 2017.

[5] T. M. Chen and S. Abu-Nimeh, "Lessons from stuxnet," *Computer*, vol. 44, no. 4, pp. 91–93, 2011.

[6] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[7] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, 2011.

[8] E. Nakashima and S. Mufson, "Hackers have attacked foreign utilities, cia analyst says," *Washington Post*, January 2008.

[9] K. Parrish, "Hackers can gain control of an insulin pump to inject a harmful dose into patients," http://www.digitaltrends.com/computing/johnson-animas-onetouch-ping-insulin-pump-vulnerable-hacker-attack/, October 05, 2016.

[10] C. Li, A. Raghunathan, and N. K. Jha, "Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system," in *e-Health Networking Applications and Services (Healthcom), 2011 13th IEEE International Conference on*. IEEE, 2011, pp. 150–156.

[11] G. Loukas, *Cyber-physical attacks: A growing invisible threat*. Butterworth-Heinemann, 2015.

[12] "Cybersecurity for networked medical devices is a shared responsibility: Fda safety reminder," https://www.fda.gov/MedicalDevices/Safety/AlertsandNotices/ucm189111.htm, November 04, 2009.

[13] "Cybersecurity for medical devices and hospital networks: Fda safety communication," https://www.fda.gov/MedicalDevices/Safety/AlertsandNotices/ucm189111.htm, June 17, 2013.

[14] I. Zeifman, D. Bekerman, and B. Herzberg, "Breaking down mirai: An iot ddos botnet analysis," *Imperva. Source: https://www. incapsula. com/blog/malware-analysis-mirai-ddos-botnet. html*, 2016.

[15] R. Brewer, "Ransomware attacks: detection, prevention and cure," *Network Security*, vol. 2016, no. 9, pp. 5–9, 2016.

[16] D. F. Sittig and H. Singh, "A socio-technical approach to preventing, mitigating, and recovering from ransomware attacks," *Applied clinical informatics*, vol. 7, no. 2, p. 624, 2016.

[17] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems securityâĂŤa survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802–1831, 2017.

[18] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "Iot security: ongoing challenges and research opportunities," in *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*. IEEE, 2014, pp. 230–234.

[19] K. Dunham, "Evaluating anti-virus software: Which is best?" *Information Systems Security*, vol. 12, no. 3, pp. 17–28, 2003.

[20] A. Mohaisen and O. Alrawi, "Av-meter: An evaluation of antivirus scans and labels," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2014, pp. 112–131.

[21] C. Foundation, "Automated malware analysis - cuckoo sandbox," http://www.cuckoosandbox.org/.

[22] C. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using cwsandbox," *IEEE Security & Privacy*, vol. 5, no. 2, 2007.

[23] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 289–302, 2016.

[24] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 559–570.

[25] M. Ozsoy, K. N. Khasawneh, C. Donovick, I. Gorelik, N. Abu-Ghazaleh, and D. Ponomarev, "Hardware-based malware detection using low-level architectural features," *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3332–3344, 2016.

[26] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "Unsupervised anomaly-based malware detection using hardware features," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2014, pp. 109–129.

[27] A. Viswanathan, K. Tan, and C. Neuman, "Deconstructing the assessment of anomaly-based intrusion detectors," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2013, pp. 286–306.

[28] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE, 1999, pp. 133–145.

[29] B. B. Kang and A. Srivastava, "Dynamic malware analysis," in *Encyclopedia of Cryptography and Security, 2nd Ed.*, 2011, pp. 367–368.

[30] N. Scaife, H. Carter, P. Traynor, and K. R. Butler, "Cryptolock (and drop it): stopping ransomware attacks on user data," in *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*. IEEE, 2016, pp. 303–312.

[31] K. Tian, D. Yao, B. G. Ryder, and G. Tan, "Analysis of code heterogeneity for high-precision classification of repackaged malware," in *Security and Privacy Workshops (SPW), 2016 IEEE*. IEEE, 2016, pp. 262–271.

[32] A. Nourian and S. Madnick, "A systems theoretic approach to the security threats in cyber physical systems applied to stuxnet," *IEEE Transactions on Dependable and Secure Computing*, 2015.

[33] L. Liu, G. Yan, X. Zhang, and S. Chen, "Virusmeter: Preventing your cellphone from spies," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2009, pp. 244–264.

[34] H. Kim, J. Smith, and K. G. Shin, "Detecting energy-greedy anomalies and mobile malware variants," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 239–252.

[35] C. R. A. González and J. H. Reed, "Power fingerprinting in sdr integrity assessment for security and regulatory compliance," *Analog Integrated Circuits and Signal Processing*, vol. 69, no. 2-3, p. 307, 2011.

[36] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, W. Xu, and K. Fu, "Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices." in *HealthTech*, 2013.

[37] Y. Liu, L. Wei, Z. Zhou, K. Zhang, W. Xu, and Q. Xu, "On code execution tracking via power side-channel," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 1019–1031. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978299

[38] R. Callan, F. Behrang, A. Zajic, M. Prvulovic, and A. Orso, "Zero-overhead profiling via em emanations," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ACM, 2016, pp. 401–412.

[39] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "Eddie: Em-based detection of deviations in program execution," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: ACM, 2017, pp. 333–346. [Online]. Available: http://doi.acm.org/10.1145/3079856.3080223

[40] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "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*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 1095–1108. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134081

[41] S. Ponomarev and T. Atkison, "Industrial control system network intrusion detection by telemetry analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 252–260, 2016.

[42] R. Mitchell and R. Chen, "Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 16–30, 2015.

[43] D. Agrawal and B. Archambeault, "Rao and jr, rohatgi, p.:âĂIJthe em side-channel (s): Attacks and assessment methodologiesâĂİ," *Cryptographic Hardware and Embedded Systems–CHES*, 2002.

[44] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 207–228.

[45] Y.-i. Hayashi, N. Homma, T. Mizuki, H. Shimada, T. Aoki, H. Sone, L. Sauvage, and J.-L. Danger, "Efficient evaluation of em radiation associated with information leakage from cryptographic devices," *IEEE Transactions on Electromagnetic Compatibility*, vol. 55, no. 3, pp. 555–563, 2013.

[46] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson, "Efficient power and timing side channels for physical unclonable functions," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2014, pp. 476–492.

[47] D. Genkin, A. Shamir, and E. Tromer, "Rsa key extraction via low-bandwidth acoustic cryptanalysis," in *International Cryptology Conference*. Springer, 2014, pp. 444–461.

[48] Y. Berger, A. Wool, and A. Yeredor, "Dictionary attacks using keyboard acoustic emanations," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 245–254.

[49] R. Callan, A. Zajic, and M. Prvulovic, "A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events," in *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*. IEEE, 2014, pp. 242–254.

[50] R. Callan, A. Zajić, and M. Prvulovic, "Fase: Finding amplitude-modulated side-channel emanations," in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3. ACM, 2015, pp. 592–603.

[51] J. Balasch, B. Gierlichs, and I. Verbauwhede, "Electromagnetic circuit fingerprints for hardware trojan detection," in *2015 IEEE International Symposium on Electromagnetic Compatibility (EMC)*, Aug 2015, pp. 246–251.

[52] O. SÃűll, T. Korak, M. Muehlberghuber, and M. Hutter, "Em-based detection of hardware trojans on fpgas," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, May 2014, pp. 84–87.

[53] K. Sakiyama, M. Kasuya, T. Machida, A. Matsubara, Y. Kuai, Y. i. Hayashi, T. Mizuki, N. Miura, and M. Nagata, "Physical authentication using side-channel information," in *2016 4th International Conference on Information and Communication Technology (ICoICT)*, May 2016, pp. 1–6.

[54] G. A. Jacoby, R. Marchany, and N. Davis, "Battery-based intrusion detection a first line of defense," in *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*. IEEE, 2004, pp. 272–279.

[55] T. K. Buennemeyer, T. M. Nelson, L. M. Clagett, J. P. Dunning, R. C. Marchany, and J. G. Tront, "Mobile device profiling and intrusion detection using smart batteries," in *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*. IEEE, 2008, pp. 296–296.

[56] N. Sehatbakhsh, A. Nazari, A. Zajic, and M. Prvulovic, "Spectral profiling: Observer-effect-free profiling by monitoring em emanations," in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*. IEEE, 2016, pp. 1–11.

[57] A. Zajic and M. Prvulovic, "Experimental demonstration of electromagnetic information leakage from modern processor-memory systems,"

Electromagnetic Compatibility, IEEE Transactions on, vol. 56, no. 4, pp. 885–893, Aug 2014.

[58] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[59] G. Rothermel, S. Elbaum, A. Kinneer, and H. Do, "Software-artifact infrastructure repository," *UR L http://sir. unl. edu/portal*, 2006.

[60] N. Andronio, S. Zanero, and F. Maggi, "Heldroid: Dissecting and detecting mobile ransomware," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2015, pp. 382–404.

[61] https://www.ettus.com/product/details/USRP-B200mini, accessed May 3, 2017.

[62] "Open syringe-pump source code and project," https://github.com/naroom/OpenSyringePump, last Accessed: 2018-08-01.

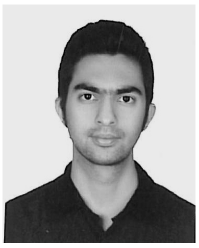
[63] "Pid controller soldering iron code and project," https://github.com/sfrwmaker/soldering_controller, last Accessed: 2018-08-01.

[64] D. Quarta, M. Pogliani, M. Polino, F. Maggi, A. M. Zanchettin, and S. Zanero, "An experimental security analysis of an industrial robot controller," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 268–286.

[65] "Robotic arm code and project," https://lifehacker.com/build-a-kickass-robot-arm-the-perfect-arduino-project-1700643747, last Accessed: 2018-08-01.

[66] "Arduino servo refrence library," https://www.arduino.cc/en/Reference/Servo, last Accessed: 2018-08-01.

Haider Adnan Khan received the B.Sc degree in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology in 2006, and the M.Sc. in Electrical Engineering and Information Technology from Karlsruhe Institute of Technology, Germany in 2011. Since 2016, he has been a Graduate Research Assistant with the Electromagnetic Measurements in Communications and Computing (EMC^2) Lab, pursuing the Ph.D. degree in the School of Electrical and Computer Engineering, Georgia Institute of Technology focusing on electromagnetic information leakage from computing devices. His research interests span areas of digital signal processing, machine learning, and computer vision.



Nader Sehatbakhsh received the B.Sc degree in Electrical and Electronic Engineering from University of Tehran in 2013 and the M.Sc. in Electrical Engineering from Georgia Institute of Technology in 2016. Since 2014, he has been a Graduate Research Assistant with CompArch and Electromagnetic Measurements in Communications and Computing (EMC^2) Labs, pursuing the Ph.D. degree in the School of Computer Science, Georgia Institute of Technology focusing on Computer Architecture, Embedded System and Hardware Security. He won the best paper award in MIRCO'49 for his work on using EM side-channel signals for software profiling.

Luong N. Nguyen received the B.Sc. degree in Electrical and Computer Engineering from the Hanoi University of Science and Technology in 2013 and the M.Sc. degree in Electrical and Computer Engineering from the Seoul National University in 2016. Since 2016, he has been a Graduate Research Assistant in the Electromagnetic Measurements in Communications and Computing (EMC^2) Lab, pursuing the Ph.D. degree in the School of Electrical and Computer Engineering, Georgia Institute of Technology focusing on digital circuit design, software and hardware security, and embedded system. His current research interests span areas of ASIC design and computer architecture.

Robert L. Callan (S'14-M'17) received the B.Sc. degree in electrical engineering from the University of Pennsylvania in 2007, the M.Sc. degree in electrical engineering from the University of Southern California in 2008, and the Ph.D. degree from the School of Electrical and Computer Engineering at the Georgia Institute of Technology in 2016. He is currently a post-doctoral researcher in the Electromagnetic Measurements in Communications and Computing (EMC^2) Lab at the Georgia Institute of Technology focusing on analyzing software using EM emanations. Previously, he characterized high speed serial interfaces at IBM and Altera. His research interests span areas of electromagnetics, VLSI, and computer engineering.

Arie Yeredor (M'99-SM'02) received the B.Sc. (summa cum laude) and Ph.D. degrees in electrical engineering from Tel-Aviv University (TAU), Tel-Aviv, Israel, in 1984 and 1997, respectively. He is currently an Associate Professor with the School of Electrical Engineering at TAU, where his research and teaching areas are in statistical and digital signal processing and estimation theory. He also held a consulting position in these research areas with NICE Systems, Inc., Ra'anana, Israel, from 1990 to 2015. In 2015-16 he has been on Sabbatical Leave from TAU as a visiting professor at the Georgia Institute of Technology, Atlanta, GA, USA. Prof. Yeredor serves as a Senior Area Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, as a member of the IEEE Signal Processing Society's Signal Processing Theory and Methods (SPTM) Technical Committee, and as Chair of the Signal Processing chapter of IEEE Israel Section. He has been awarded the yearly Best Lecturer of the Faculty of Engineering Award (at TAU) seven times.

Milos Prvulovic (S'97-M'03-SM'09) received the B.Sc. degree in electrical engineering from the University of Belgrade in 1998, and the M.Sc. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 2001 and 2003, respectively. He is a Professor in the School of Computer Science at the Georgia Institute of Technology, where he joined in 2003. His research interests are in computer architecture, especially hardware support for software monitoring, debugging, and security. He is a past recipient of the NSF CAREER award, and a senior member of the ACM, the IEEE, and the IEEE Computer Society.

Alenka Zajic (S'99-M'09-SM'13) received the B.Sc. and M.Sc. degrees form the School of Electrical Engineering, University of Belgrade, in 2001 and 2003, respectively. She received her Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology in 2008. Currently, she is an Associate Professor in the School of Electrical and Computer Engineering at Georgia Institute of Technology. Dr. Zajić was the recipient of the 2017 NSF CAREER award, 2012 Neal Shepherd Memorial Best Propagation Paper Award, the Best Paper Award at the International Conference on Telecommunications 2008, and the Best Student Paper Award at the 2007 Wireless Communications and Networking Conference. Her research interests span areas of electromagnetic, wireless communications, signal processing, and computer engineering.