Watch Me, but Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations

Yi Han
Rutgers University
yi.han@rutgers.edu

Sriharsha Etigowni
Rutgers University
sriharsha.etigowni@rutgers.edu

Hua Liu
Rutgers University
hl678@rutgers.edu

Saman Zonouz
Rutgers University
saman.zonouz@rutgers.edu

Athina Petropulu
Rutgers University
athinap@rutgers.edu

ABSTRACT

Trustworthy operation of industrial control systems depends on secure and real-time code execution on the embedded programmable logic controllers (PLCs). The controllers monitor and control the critical infrastructures, such as electric power grids and healthcare platforms, and continuously report back the system status to human operators. We present Zeus, a contactless embedded controller security monitor to ensure its execution control flow integrity. Zeus leverages the electromagnetic emission by the PLC circuitry during the execution of the controller programs. Zeus's contactless execution tracking enables non-intrusive monitoring of security-critical controllers with tight real-time constraints. Those devices often cannot tolerate the cost and performance overhead that comes with additional traditional hardware or software monitoring modules. Furthermore, Zeus provides an air-gap between the monitor (trusted computing base) and the target (potentially compromised) PLC. This eliminates the possibility of the monitor infection by the same attack vectors.

Zeus monitors for control flow integrity of the PLC program execution. Zeus monitors the communications between the human-machine interface and the PLC, and captures the control logic binary uploads to the PLC. Zeus exercises its feasible execution paths, and fingerprints their emissions using an external electromagnetic sensor. Zeus trains a neural network for legitimate PLC executions, and uses it at runtime to identify the control flow based on PLC's electromagnetic emissions. We implemented Zeus on a commercial Allen Bradley PLC, which is widely used in industry, and evaluated it on real-world control program executions. Zeus was able to distinguish between different legitimate and malicious executions with 98.9% accuracy and with zero overhead on PLC execution by design.

CCS '17, October 30-November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-4946-8/17/10…$15.00
https://doi.org/10.1145/3133956.3134081

KEYWORDS

Side channel analysis; control flow integrity; deep learning

1 INTRODUCTION

Industrial control systems (ICS) are fundamental parts of modern society as they control and monitor critical infrastructures such as electricity grids, health-care, chemical production, oil and gas refinery, transportation and water treatment. Due to their importance and large attack surfaces, they are becoming attractive targets for malicious penetrations leading to catastrophic failures with substantive impact [29, 48] including the recent BlackEnergy worm against Ukranian electricity grid [12]. Recently, the Stuxnet malware uploaded malicious code to programmable logic controllers (PLCs), and physically damaged 20% of Iranian PLC-controlled centrifuges [13]. The discovery of Duqu [7] and Havex [47] show that such attacks are not isolated cases as they infected ICS in more than eight countries. Some of these vulnerable controllers are Internet-connected [11] and exposed by computer search engines like Shodan [4]. There have been an increasing number of reports on malicious attempts of PLC port scanning, automated PLC malware generation, modifying control system-specific protocols and access to system diagnostics [5, 32]. Nevertheless, the ICS market is expected to grow to $10.33 billion by 2018 [53].

There has been an increasing number of past works on embedded systems and PLC protection. Offline formal control logic analysis have been investigated by solutions such as TSV [34], through symbolic execution and model checking mechanisms. Solutions such as WeaselBoard [38] and CPAC [10] perform runtime PLC execution monitoring using control logic and firmware-level reference monitor implementations. Most related to our paper, there have been attack and defense solutions that employ side-channel analyses to either disclose secret information (e.g., cryptographic keys [17]), or detect anomalous misbehavior (e.g., execution tracking [31]). Side channel-based attacks require selective monitoring of only execution points of interest, such as the encryption subroutines. On the other hand, side channel-based defenses have to monitor throughout the execution looking for anomalies.

In this paper, we present Zeus, a contactless PLC control flow integrity monitor that monitors the program execution by analyzing the PLC's runtime electromagnetic (EM) emanation side channel. Given a PLC controller program, Zeus profiles the PLC's

electromagnetic emanation during the execution of feasible paths of the legitimate program. Zeus pre-processes the signal traces and uses them offline to train a neural network model. The model is later used during the runtime operation to either determine the fine-grained control flow of the execution based on the real-time EM emanations or declare unknown (malicious) code execution.

Contactless monitoring enables Zeus to ensure security of crucial controllers in mission-critical applications with tight real-time constraints. The operators are often very reluctant to instrument those controllers' software stack with additional security probes that cause performance overhead and hurt the underlying real-time guarantees. Additionally, from security viewpoint, contactless monitoring keeps Zeus away from the attack vectors that target the controllers because of the introduced air gap between the monitor and the victim controller. Other side channel-based techniques such as power signal analyses draw and monitor current from controllers' circuitry. In contrast, Zeus is completely non-intrusive and passive; it does not require any instrumentation of the controller and does not affect its electronics.

Zeus monitors all the network links bound to the PLC, and captures the control logic uploads by the human-machine interface (HMI) servers that are sent for execution on the PLC. Zeus exercises various code segments of the binary while capturing the electromagnetic emanations. Zeus profiles the control logic on the PLC with deactivated output modules to ensure the underlying physical process (actuators) are not affected during the training phase. The training is implemented in two stages. First, Zeus executes control logic symbolically[1] and removes infeasible paths. Through counterexample guided inductive synthesis [49], Zeus generates different test inputs for each execution path, and trains a neural network based on the corresponding electromagnetic emanation. The trained neural network allows Zeus to detect the execution of an illegitimate control flow and/or malicious code injection.

The contributions of this paper are as follows:

- We present a new execution control flow tracking solution for embedded PLC controllers that enables security monitoring with air-gapped electromagnetic sensors.
- We develop an online signal processing framework to analyze the electromagnetic signals and extract minimal feature set necessary for execution integrity monitoring.
- We evaluated Zeus using an inexpensive sensor against widely-used control programs, e.g., proportional-integral-derivative (PID) controllers, on commercial Allen Bradley PLC devices (most popular in North America) with ARM Cortex-M3 processors. Zeus detects malicious code injections with 98.9% accuracy in real-world settings.

Overview and Organization. In Section 2, we explain our assumptions about the adversaries and their capabilities. In Section 3, we provide a brief background on programmable logic controllers and their typical configurations as well as neural networks that Zeus employs for program behavioral modeling. In Section 4, we discuss about the electromagnetic signals emitted by the PLCs and how they characterize the program execution. We discuss how Zeus generates training data points for program behavioral modeling and transforms the signal traces into spectrum sequences. In Section 5, we present our fine-grained emanation analysis model, where a neural network model of the legitimate program control flows is constructed and trained using electromagnetic emanation signals. In Section 6, we present our empirical evaluations of Zeus's various components on ten real-world PLC programs and attack scenarios similar to Stuxnet [13]. In Section 7, we review the recent most related work in the literature, and finally, we conclude the paper in Section 8.

[1]Complete symbolic execution of embedded PLC control logic programs is often feasible as they are mostly not branch-heavy in practice.

2 THREAT MODEL

One of the most prominent security failure causes in control systems using PLCs is the failure to guard PLCs against remote programming [60]. PLC programmer machines are most often based on commodity operating systems, and often lag security update releases by months [54]. In the following, we state the assumptions made on the security measures that must be successfully in place for Zeus to function correctly.

We assume there is some trusted path from Zeus to system operators to alert them of any malicious executions. Unlike software-based solutions, Zeus's contactless monitoring enables secure monitoring even if the software stack below the PLC's control logic (e.g., firmware or operating system) is compromised. The PLC-bound network link used to transfer the control logic programs for execution is assumed to be trusted. This allows Zeus to obtain a legitimate copy of the control logic to compare with the runtime PLC executions for control flow integrity checks. Zeus does not assume source code availability and works with binaries.

Zeus defends against *control channel* attacks (e.g., Stuxnet [13]), where the adversaries upload arbitrary and potentially malicious control logic on the PLC for execution. More specifically, the types of control logic attacks that Zeus can protect against are *i)* modified control logic such as injection, removal, and replacement of code segments in the legitimate control logic program; and *ii)* hijacked control flow of the legitimate control logic execution through network exploits (e.g., code reuse attacks[2] such as return-oriented programming). Zeus does not defend against *sensor channel* attacks, where sensor data is forged by compromised sensors. In such a case, the control logic may behave exactly as intended, but on false sensor data [30]. Additionally, Zeus itself may be attacked by external signal jammers leading to false positives. However, this would not affect the integrity of the control logic execution on the PLC.

3 BACKGROUND

Programmable Logic Controllers. Programmable logic controllers are multiple-input-multiple-output computers. They have input and output modules to interact with the physical world (plant)

[2]Protection against control flow and code reuse attacks are simpler in PLCs compared to conventional computers, because the PLCs' restrictive and more primitive programming languages (e.g., type safe and no indirect call sites) allows deterministic modeling of the legitimate control flows.

to monitor and control critical infrastructures such as manufacturing, robotics, and avionics. The PLC's input modules are connected to sensors within the plant and receive measurements about the plant's status continuously. The PLC's output modules are connected to plant actuators and convey the commands to control it. The PLC converts sensor readings into digital values, process the readings with the built-in computing unit, and forward the outputs to the physical world. The logical behavior of PLCs (i.e., the processing of the input data) is programmable. The control logic programs are developed by the control system operators on human-machine-interface (HMI) servers that are connected to the PLCs through network links. Once developed, the control logic is compiled and sent to the PLC for execution. The program is executed repeatedly in fixed intervals, called scan cycles. During each scan cycle, the control logic program reads input values from memory and stores the output values to memory. The PLC firmware is responsible for the interchange of these updated values to and from the PLC's input/output ports to interface the physical world. The firmware also implements the reporting mechanisms such as the LED display on the device and real-time data transfers to the HMI about the plant's current status.

Deep Neural Networks. Neural network is a class of supervised learning models that tries to learn the complex nonlinear mapping between input data and their targets (e.g. class labels). A basic neural network unit architecture consists of a linear mapping followed by an activation:

$$y = \sigma(Wx + b), \tag{1}$$

where x is the input feature vector, and W represents the weight matrix. σ is the activation function. It is a nonlinear function that models the complex relation between input x and output y. Common activation functions include sigmoid [26], rectified linear unit (ReLU) [64], tanh, etc. Figure 1 shows a graphical illustration of Equation 1. The edges between the Input layer and the hidden layer represents the weights W, b. Since each node in the input layer is fully connected with all nodes in the hidden layer, such unit is also called a dense layer.

A neural network can go large, which is increasing number of nodes in the hidden layer, or go deep, which is stacking multiple network units together (increasing number of hidden layers), such that more complex nonlinear mappings between data and targets can be learned. All forms of artificial neural networks essentially follow the aforementioned basic architecture. Zeus utilizes recurrent neural network (RNN) [46] to model the execution behavior of PLC programs (Section 5).

For training, the data samples, each with a corresponding label, are fed to the network's input layer. The network is trained to learn discriminative features from samples by itself. This completely data driven approach, compared to traditional hand crafted feature extraction methods, leads to much simpler-to-use and more reliable outcomes in practice. In our experiments (Section 6), we empirically show that RNNs overcome traditional techniques such as hidden Markov models (HMMs) in terms of PLC execution monitoring accuracy and performance.

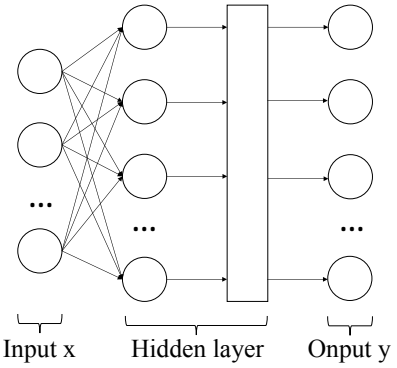


Figure 1: A basic neural network unit architecture.

Neural networks can be trained in an iterative manner using the gradient descent algorithm [63]. At each iteration, all input data are passed through the network. The output are compared with their corresponding targets t. A loss function l is defined between the network output y and the expected target on each data sample:

$$l_i = loss(y_i, t_i), \tag{2}$$

The loss function measures the difference between the current and target outputs. Zeus uses mean square error (MSE) as the loss function. The total loss is the sum over individual losses of all the data samples:

$$l = \sum_{i=1}^{N} l_i, \tag{3}$$

where N represents the total number of data samples. Computing the total loss is called the forward pass. To update weights or parameters of the network, partial derivatives of the total loss with respect to all weights are calculated to identify their maximal descending direction using back propagation. All weights are updated accordingly (the backward pass). Forward and backward passes are repeated iteratively until the values converge. The resulting network is able to produce outputs close to the expected targets.

4 PLC PROGRAM EMANATION ANALYSIS

During the PLC code execution, the processor clock frequency and switching of the underlying CMOS devices along with the power regulation board result in change of electric current in the PLC circuitry. The current produces time-varying magnetic field that interacts with the electric field leading to an electromagnetic (EM) wave. The EM wave propagates perpendicular to electric and magnetic fields [1]. In order to radiate this EM emanation, an antenna is required. The components on the PLC's printed circuit board (PCB) act as antennas. The transmission range of these waves increases with the increase in the surface area of the antenna. These emanations from the PLC boards can be captured by an external electromagnetic sensor placed near the emanation source.

Zeus uses these near-field EM waves as the PLC side channel, because they leak information about the program running on the device [16]. Different instructions usually expose different emanation

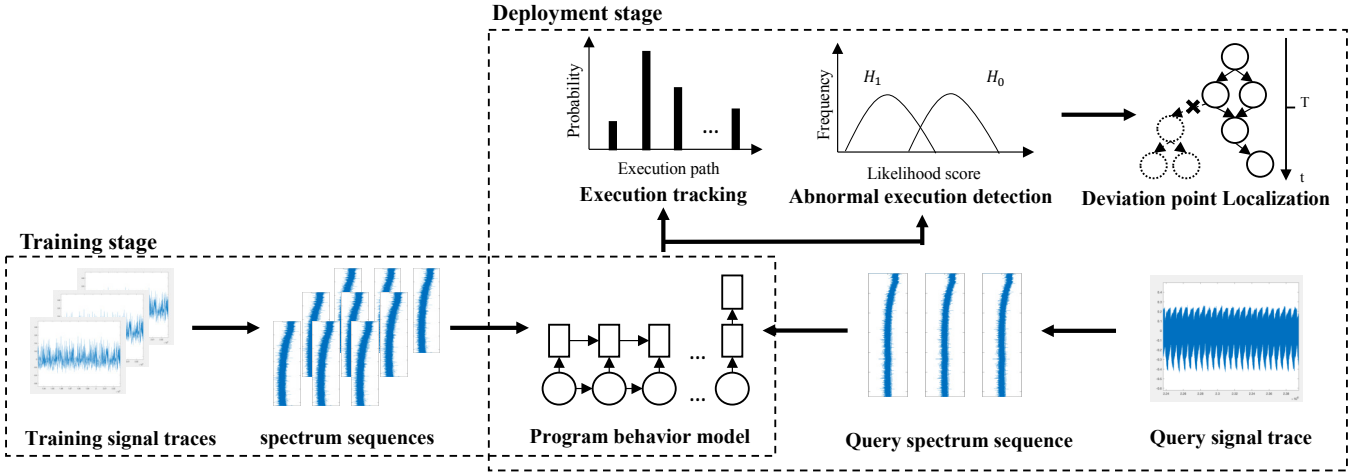


Figure 2: Zeus's control flow integrity monitoring.

patterns. Thus, the collected electromagnetic signal traces during program executions have unique local characteristics depending on the runtime control flows. This observation is utilized by Zeus to fingerprint the side-channels of legitimate program executions and identify unknown (malicious) code injections and/or control flow hijacking attempts.

Recent attempts have been made to monitor micro-controllers such as STC89C52 [31] and PIC16F687 [9] based on power signal analysis [9, 31] that require physical manipulation of the circuits for sensor placement. The data acquisition draws current from the controller boards potentially affecting its functionalities that triggers a red flag for practical deployment in controllers for mission-critical real-time operations. In comparison, Zeus's contactless, passive and non-intrusive monitoring of commercial PLC ARM processors using an inexpensive EM sensor for control flow integrity is a more challenging endeavor.

Due to the PLC architecture, the execution times of individual instructions are not fixed to the processor's clock cycle. A list of estimated completion times for instructions is provided in the user manual. However, in practice based on our observations, even the execution time of a single instruction varies across different execution runs of the same PLC code. This makes the signal analysis using time-based truncation infeasible. Being contactless, Zeus has to deal with a large amount of signal noise. Our collected electromagnetic signal traces have very low signal to noise ratio (SNR) such that repeatable local patterns along the execution paths (leveraged by [31]) cannot be observed in the time domain.

To deal with these problems, Zeus borrows ideas from speech recognition research [45]. Zeus looks at frequency representations of signal sections within a local sliding window. Zeus extracts signal segments via a sliding window on the collected signal. Each segment consist of several consecutive instructions. Zeus then computes the power spectral density of each segment.

Unlike time domain signals, we observed that the patterns in the frequency representations are much more stable and robust to noise. This is because the local spectra (spectrum sequence) are computed through weighted summation of all time signal points within the window. Hence, the white noise is not cumulated, while the underlying desired deterministic signal is. Therefore, a sequence of local spectra extracted from the PLC code execution EM signal trace includes repeatable patterns to characterize individual execution paths. Zeus deploys the aforementioned analysis to model the execution behavior of target PLC programs.

For completeness of the results, signal traces of all feasible control flows of the program are collected. Zeus monitors the network link between the HMI servers and the PLC controllers, and intercepts the control logic uploads to the PLC. Through symbolic execution [34], the execution path predicates are aggregated and checked by an SMT solver for satisfiability. Consequently, infeasible executable paths are eliminated.

For each remaining feasible path, Zeus calculates several concrete test cases through counterexample-guided inductive synthesis [49]. More specifically, to calculate the first test case for a path, its aggregated path condition expressed as a conjunctive logical expression $\varphi_p = (\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n))$ is fed to the SMT solver. The solvers produces a concrete input value set (e.g., $i = 20$). Calculating the second concrete input for the same path involves feeding $\varphi_p \wedge \neg\varphi_i$ to the SMT solver, where $\varphi_i := (I == 20)$ and $\neg$ represents logical negation. The next concrete inputs are calculated similarly.

Zeus runs the program on the PLC using the generated test cases for each execution path, and collects the electromagnetic emanations using an external sensor. The collected signal traces along with their labels (corresponding control flows) are fed to a sequence neural network classifier for training. All these steps are performed offline. During the runtime, Zeus's external sensor collects the PLC's emanations and employs the classifier to determine whether the signal trace belongs to the feasible legitimate execution paths. A modified execution path (e.g., a maliciously injected PLC program) will lead to a change in electromagnetic emanations away from the samples observed by the classifier during the training phase. The deviation triggers a red flag by the classifier, and the execution is marked as malicious.

5 EM-BASED CONTROL FLOW MONITORING

There have been works utilizing electromagnetic side channel signals to detect abnormal executions [51]. They follow a template matching scheme, where the query signal is compared with all constructed templates of the execution paths. Based on our experiments, such time domain-based signal matching techniques cannot distinguish fine-grained characteristics of complex program control flows accurately. To address this, Zeus constructs a sequential neural network classification model of pre-processed frequency domain data samples. The model therefore learns from control flow transitions from the training frequency data and encodes them in its network weights. This model describes the behavior of the program.

Zeus maps the control flow transitions of any new legitimate data sequence with the learned transitions, modeled by the neural network, and determines a specific control flow that the observations correspond to. A data sequence with abnormal components such as unseen segments (due to code injection attacks) or invalid transitions between segments (due to control flow hijacking) will cause mismatches. Such mismatches accumulate along the sequence, cause the neural network states and thus the output of the model to deviate from expected values.

Figure 2 shows Zeus's work flow. During the offline training stage, each collected signal trace is transformed into a spectrum sequence (Section 4). The sequence neural network model is trained using spectrum sequences of all classes (legitimate control flows). After deployment, Zeus feeds the online spectrum sequences of collected query signal traces into the trained model. This results in a probability distribution computed by the model over all the classes. The class with the highest score is compared to a predefined threshold. If the score exceeds the threshold, Zeus assigns it to the query signal trace as the execution path that the program is taking currently. If the threshold check fails, it indicates a mismatch between the query signal trace and the trained model. Consequently, Zeus triggers an alert about an illegitimate control flow. Zeus provides more fine-grained reports about the mismatch regarding the execution point that the real-time control flow deviated from the legitimate expected flows. This information can be used later for detailed vulnerability discovery, e.g., how the control flow was hijacked. We consider the vulnerability analysis phase outside the scope of this paper.

5.1 Offline Model Construction and Training

To construct the classification model with sequential inputs, we use a long term short memory (LSTM) network layer [18]. LSTM is a variation of the recurrent neural network (RNN) used for modeling sequential data. LSTM takes a sequence of inputs and maintains a hidden state vector along the sequence. This fits Zeus's use-case, where the observables are EM signals, while the hidden states represent the underlying unobserved code segments and basic blocks.

At each time step of the input sequence, current hidden state vector is computed based on both the previous hidden state vector and the current input. The hidden state carries long term dependency between time steps. This enables Zeus to capture contextual information in the sequential control flow transitions.

Let $[x_1, x_2, ..., x_N]$ be a spectrum sequence computed for a collected EM signal trace. x_t indicates the input in the sequence at time t. The vector size N depends on the execution time of the control flow. The current hidden state h_t can be computed as:

$$h_t = o_t * \tanh(c_t), \tag{4}$$

where tanh is the hyperbolic tangent activation function; $*$ denotes the entry-wise product; o_t is the output gate vector; and c_t is the cell state vector. The two vectors can be computed as:

$$\begin{aligned} o_t &= \text{sigmoid}(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t * c_{t-1} + i_t * \tanh(W_c x_t + U_c h_{t-1} + b_c). \end{aligned} \tag{5}$$

In Equation 5, $W_o, U_o, b_o, W_c, U_c, b_c$ are the weights of the neural network units. Note that the design extends the basic network architecture described in Section 3. c_{t-1} and h_{t-1} are state vectors passed from the previous time step. f_t and i_t represent the forget and input gate vectors, respectively. These vectors are designed to keep only useful contextual information and acquire new information:

$$\begin{aligned} f_t &= \text{sigmoid}(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \text{sigmoid}(W_i x_t + U_i x_t + b_i), \end{aligned} \tag{6}$$

where $W_f, U_f, b_f, W_i, U_i, b_i$ are the unit weights. We add a dense layer followed by a softmax function after the output of the LSTM layer at the last time step h_N. This maps the network to a probability distribution over all legitimate control flows in the PLC code.

$$p = \text{softmax}(W h_N + b), \tag{7}$$

where the height the weight matrix W is the same as number of legitimate control flows.

Intuitively, the neural network model output is a one-hot [55] vector q. It has a 1 on its entry that corresponds to the identified control flow and 0s on all its other entries. q can also be viewed as a probability distribution. We define the loss function of our model as the cross entropy between the model's actual output p and the target vector q. The resulting cross entropy measures the difference between two probability distributions:

$$l = -\sum_i p_i \log q_i, \tag{8}$$

where i is the index of the legitimate control flows. The total loss of our model is the sum of losses over all the training EM spectrum sequences and their corresponding class labels (control flows). During the training, weights of the model are tuned iteratively as described in Section 2. A well-trained model will have its output probability distribution very close to its corresponding one-hot vector, and the output distribution will bias remarkably towards the corresponding control flow.

The overall architecture of Zeus's network model is shown in Figure 3. The size of the hidden state vector h_t can be increased to carry more information along the sequence. Moreover, multiple

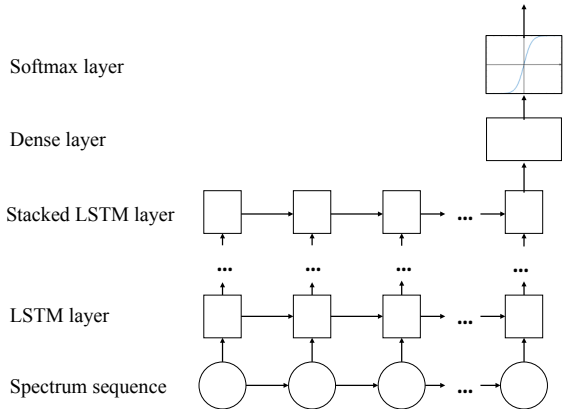


Figure 3: Network architecture of proposed model.

LSTM can be stacked to be more capable of characterizing the EM spectrum sequences for PLC execution classification. However both adjustments increase the computational complexity. To ensure efficient online classification, we kept the network model size minimal as long as it did not affect Zeus's accuracy of malicious execution detection.

The collected electromagnetic signals suffer from random perturbations caused by EM interference of other components on the PLC device. To reduce the noise effect, Zeus provides the neural network model training with many EM signal traces for each control flow. This is possible through Zeus's PLC code analysis and generation of several test-cases for each feasible execution path. Consequently, the neural network training algorithm receive many traces with the same label (control flow) each incorporating random noise. This enables the neural network's data-driven feature extractions procedures to train its unit weights based on the main signal ignoring the noise margins.

5.2 Online PLC Execution Monitoring

Zeus uses the trained model at runtime to detect anomalous executions. EM spectrum sequences from legitimate executions will have their network outputs distribution heavily biased towards the corresponding class label. The class with the highest probability will be reported as the identified control flow I:

$$I = \arg\max_i p_i, \tag{9}$$

and its likelihood score can be calculated as $L = \max_i p_i$. L represents how likely this signal trace belongs to the legitimate PLC code. In the case of correct classification outcome, the network's input transitions match with the corresponding control flow transitions of the PLC control logic and the desired network state is maintained as the most likely state along the input EM trace.

Malicious control flows constitute either execution of a maliciously injected new code or code reuse attacks that execute the available instructions while deviating from legitimate control flows at some point. The introduced new instructions or the control flow deviation cause a mismatch between the observed EM signals and the neural network's learned transitions. This reduces the bias in the neural network model's calculated probability distribution increasing its entropy. Therefore, by setting a threshold on the likelihood score L, abnormal executions can be identified as they match none of the known legitimate control flows.

Let $H_0(H_1)$ indicate the legitimate (malicious) execution, the detection problem can be expressed as:

$$L \underset{H_0}{\overset{H_1}{\lessgtr}} \varepsilon, \tag{10}$$

where ε is the preset threshold.

For malicious executions, Zeus can also locate the point, where PLC execution deviated from the legitimate flows. Let $h = [h_{n1}, h_{n2}, \cdots, h_{nN}]$ be the hidden state sequence of the n-th LSTM layer of our model for a query EM spectrum sequence input for a malicious execution. Let's also assume Zeus identifies the hidden state sequence $h^I = [h^I_{n1}, h^I_{n2}, \cdots, h^I_{nM}]$ as the most likely legitimate control flow that corresponds to a given query EM trace. The deviation point can be located by computing the distance d_t between the two sequences at each time step t:

$$d_t = \sqrt{(h_{nt} - h^I_{nt})^2}. \tag{11}$$

The deviation of the PLC execution from the legitimate control flows is reflected in the sudden change of signal traces and thus the neural network inputs. A changed spectrum input will cause its corresponding hidden state vector to move away from its expected vector in the state space. Therefore, a sudden step increase in the distance sequence $[d_1, d_2, \cdots]$ indicates the point, where the deviation happens.

6 IMPLEMENTATION AND EVALUATION

We evaluated Zeus on real-world settings with commercial PLC devices and using legitimate and malicious control logics. We first describe our experimental setup including the signal acquisition system and the target PLC model. We will discuss the results on the electromagnetic emanations of the target PLC and their discriminative spectrum characteristics. We measure Zeus's accuracy in classifying legitimate control flows, and detecting malicious executions. We compare Zeus's data-driven approach with traditional model-based solutions using hidden Markov models [31] We finally test the performance of Zeus on several real applications. An Intel i7-6800K CPU was used to compute frequency representations, and HMM training and testing. Our LSTM neural network model was trained on an NVIDIA GTX1080 GPU.

6.1 Experimental Setup

Figure 4 shows our signal recording setup that consists of a recording probe and an amplifier. The corresponding test-bed configuration and component interconnection is shown in Figure 5. In our experiments, we used the Allen Bradley 1769-L18ER-BB1B CompactLogix PLC (with ARM Cortex-M3 processor). Allen Bradley PLCs are the most popular and widely used controllers in many industrial control systems in North America. We used an AKG-P170 condenser microphone without the acoustic capsule or transducer

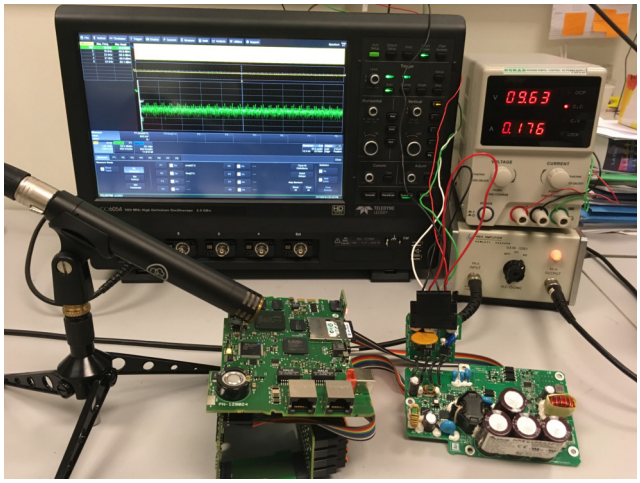


Figure 4: Experimental setup including the PLC, external sensing probe, the amplifier, and the sampling oscilloscope.

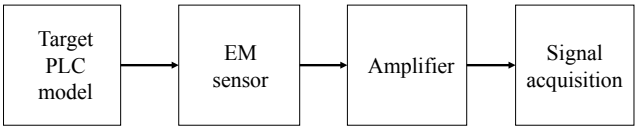


Figure 5: Experimentation test-bed configuration for electromagnetic (EM) side channel analysis.

Figure 6: AKG P170 condenser microphone without transducer serving as an electromagnetic probe.

(Figure 6) as an antenna to receive the electromagnetic emanation[3]. We also tried a Langer LF-R 400 passive antenna. However, we achieved the best signal sensitivity from the AKG-P170 microphone. This is because the AKG-P170 microphone together with the phantom power supply can be viewed as an active antenna. Active antennas have better sensitivities than passive antennas, since signals are pre-amplified.

[3]When the acoustic capsule is detached from the microphone, the remaining part of the microphone serves as an antenna since the coil in the microphone is sufficiently long to receive the signals emitted from the board.

We used an HP-461A amplifier to increase the signal strength. We observed that most of the informative frequency variations appear below 5 *MHz*. Accordingly, we set our sampling rate to preserve most frequency information while maintaining a moderate computation complexity for online malicious code detection performance.

6.2 PLC Electromagnetic Emanations

We performed numerous tests and inspected various regions of the three PLC PCB boards to identify the point that emits the most distinguishable EM signals. Once that point was identified, we adjusted our directional EM probe to focus on the point while collecting the EM emanations for our experiments.

Figure 7a shows the components that we mainly investigated. The main sources of emanation were the proprietary Allen Bradley chip, the field-programmable gate array (FPGA) and the surface mount device (SMD) capacitors on PLC's communication PCB board. The SMD capacitors are involved in the voltage regulation for the chips. Figure 7b shows the strength of the corresponding emanation from each point. Since the surface area of the SMD capacitors is very small, the corresponding emission was rather weak. The surface area of the Allan Bradley chip is relatively larger, and hence the corresponding emission was stronger. We proceeded by focusing on that chip for our following experiments. Figure 7c shows how the captured signal appears with as the probe-chip distance increases. The EM signals were collected by the probe located 0.1 cm away from the proprietary chip.

We investigated the differences among the EM emanations from the PLC execution of different instruction types. Different PLC instructions have different execution times and computation complexities thus different power consumptions that is reflected in the emanation signals as discriminative spectral patterns. Figure 8 shows the results. These spectral pattern types are the core basis for Zeus's design.

Figure 8 visualizes the discriminative spectral patterns of different PLC instructions. PLC's (ARM) ISA include 22 different types of instructions. We show the results for only the 16 types that are commonly used in PLC programs[4]. Modules involving complicated computations, such as PID[5] were also tested. Figure 8 shows that different instructions give rise to EM signals with different intensities at different frequencies. Zeus exploits these fingerprints to estimate the PLC's internal runtime execution state and dynamic control flow using the collected EM emanations.

We further verify our visual observations by performing a classification validation on spectra of emanation signals of these instructions using the random decision forests algorithm [23]. We used Weka [22] to implement the classifier. One instruction of each instruction type (Figure 8) was tested. An emanation signal of 200 μs was collected under sampling rate of 50 *MHz* and transformed into spectrum representation. 1000 such signal traces were collected for

[4]The instructions include arithmetic instructions (ADD, MUL, DIV, DEG), advanced math instructions (LN, SIN, XPY, STD), comparing instructions (XOR, GRT), array manipulation instructions (BSL, AVE, FLL) and control instructions (TON, JMP).
[5]This PLC programming module implements the proportional-integral-derivative (PID) control algorithm [2].

Table 1: Confusion matrix for the classification.

	ADD	SIN	XOR	BSL	JMP	PID
ADD	**78.63% (747)**	5.26% (50)	8.21% (78)	1.58% (15)	4.95% (47)	1.37% (13)
SIN	5.36% (56)	**83.54% (873)**	5.65% (59)	1.05% (11)	2.39% (25)	2.01% (21)
XOR	8.13% (75)	7.48% (69)	**69.31% (639)**	0.11% (1)	12.47% (115)	2.49% (23)
BSL	1.24% (12)	1.65% (16)	0.10% (1)	**95.24% (921)**	0.21% (2)	1.55% (15)
JMP	5.44% (53)	3.29% (32)	12.32% (120)	0.21% (2)	**76.49% (745)**	2.26% (22)
PID	0.32% (3)	2.11% (20)	2.64% (25)	0.21% (2)	1.27% (12)	**93.46% (886)**

each instruction type to train the classifier, and the same number of traces were collected for validation testing.

Table 1 shows the classification confusion matrix. Most signals are correctly classified correctly as their actual instruction type. This shows that the spectral patterns of different types of instructions are indeed discriminative and can be used for control flow integrity monitoring.

Zeus applies a sliding window to the collected emanation signals. Because of various instruction execution times, sliding windows of the same length at different points of the signal cover different combinations and number of instructions. This helps for spectral patterns of the signal segments within different windows across the signal trace to be distinguishable. Therefore, the spectra of these local signal segments characterize the emanation signals, the execution paths. Zeus utilizes this to construct the program behavior model.

Figure 9a shows the EM emanation (between seconds 6 and 12) while a control logic program is installed for execution on the PLC. The visible EM emanation pattern can be used to detect runtime (malicious) control logic uploads similar to Stuxnet [13]. Figure 9b shows the electromagnetic emanation patterns during the PLC's boot-up process. These patterns can be used to detect when the PLC is remotely rebooted by an adversary.

6.3 Accuracy

We evaluated Zeus for PLC execution monitoring, control flow classification of ten real applications, and detection of malicious code executions. We computed spectrum sequences and estimated the power spectral density of signal segments. The segments were extracted using sliding windows of size 200μs, with 90% overlap between successive windows.

A stacked two-layer LSTM network with 100 nodes on both layers was employed to fingerprint the execution behavior of each program. We trained the network using stochastic gradient descent (SGD). An average of 50 epochs (iterations) were required for the network to converge on the tested programs. We obtained 100 traces for every feasible control flow of each program for training the model. For each program, a 2-fold cross validation was performed 10 times to stabilize the result.

We chose ten real PLC programs from different application domains for evaluation purposes. Table 2 lists the control logics along with their functionalities. These programs fall in the classes of vector arithmetic, numerical methods, control algorithms, cryptography, signal processing and communications.

Comparison with HMM solutions. We compared our LSTM network model with a traditional hidden Markov model (HMM) based program behavior modeling approach [31]. For the HMM, the observations were defined as the signal segment or its frequency representation. HMM state was defined as unique samples in the observation set. The number of HMM states was defined as a adjustable parameter. We set the HMM number of symbols to be 100. We fit the observations of each state with a multivariate Gaussian distribution. The parameter set (HMM's transition probabilities, observation models and initial probabilities) was estimated using Baum-Welch algorithm [35]. We used the forward algorithm [45] to calculate the observation sequence likelihoods. We will present the accuracy results for both Zeus and HMM solutions below.

We evaluated Zeus accuracy from two aspects: *i)* execution monitoring - to determine the control flow of a running legitimate PLC code, and *ii)* malicious execution detection - to detect the control flows that are not a part of the legitimate program. Table 3 shows the execution monitoring accuracy results. We evaluated Zeus (LSTM) with both pre-processed spectrum traces (Freq) and raw time domain signals (Time) and compared the results with HMM-based solutions. LSTM using the frequency representation (Freq-LSTM) achieves almost perfect results on all the evaluated programs.

LSTM's better results in comparison with HMM-based solutions can be explained by the following two observations. First, the Zeus's LSTM network architecture is able to capture long-term dependency in the input sequence. This contextual information corresponds to the control flows of the program, and hence is essential in distinguishing different execution paths. HMM models, on the other hand, assume only 1st order data dependency in the sequence, and hence miss a lot of useful information.

Second, Zeus's model is able to extract discriminative features from the input due to its stacked multi layer architecture. This contributes to the classification performance. For HMM, however, the input data is directly used for parameter estimation without any feature extraction. When raw signal segments are used as inputs, both LSTM and HMM are not able to achieve good performance (Table 3). This is because raw time signals contain lots of noise, so the underlying signal characteristics cannot be recognized and hence learned by the two models. The frequency representation, however, reveals the signal characteristics as the noise (low frequency) stays far from the main signal (high frequency).

For detection of malicious executions, Figure 10 shows the likelihood scores L (Section 5.2) of positive (abnormal) and negative

Table 2: Evaluation programs and descriptions.

Class	Name	Description	Example applications	Average length (msec)
Vector arithmetic	Matrix	Matrix multiplication	Sensor array data processing	3.3
	Q-sort	Quick sort	Value searching, element uniqueness	2.1
Numerical methods	GD	Gradient descent	Power flow optimization	5
	Newton	Newton's method	Vehicle trajectory estimation	3.5
Signal processing	Conv	Convolution	Signal filtering	9.2
	DCT	Discrete cosine transform	Audio lossy compression	17.3
Communications	Dijkstra	Dijkstra's algorithm	Routing optimization in smart grid	11.3
Cryptography	AES	AES-128 encryption	Data protection, access control	18.1
Control systems	PID	PID control	Vehicle cruise control	6.5
	Patfilt	particle filter	Object Tracking, localization estimation	2.5

(normal) samples for two example applications, namely Newton-Raphson numerical method and AES encryption algorithm. Note the negative samples tend to concentrate within a small range, while the positive samples are more spread out. This is because the number of control flows with each program is finite, and each control flow is well recognized by our network through training. Thus, the signal traces of the legitimate control flows match closely with the LSTM model. The malicious programs can take any arbitrary control flow especially in the case of malicious code injection attacks; therefore, their matching degree vary a lot.

Figure 11 shows the ROC curve for the frequency and time domain data using Zeus's LSTM and HMM solutions. The numbers are average over all the ten applications. Zeus (LSTM) using the frequency traces achieves almost perfect detection performance. Steeper ROC curve indicates better separation of positive and negative samples, and thus better performance. This is usually measured by the area under the curve (AUC). AUC is usually between 0.5 (random guess) and 1 (perfect separation). Table 4 shows the AUC for each target PLC program and each evaluation setup. The stacked multi-layer architecture of Zeus's network model captures important information both from the hidden states and the inputs, and carries it along the sequence. This results in better learning of the program behavior from the signal traces.

Note the HMM using the raw time domain signal performs worse than random guessing (diagonal line on ROC - Figure 11). This is because HMM, due to its limited first order dependency assumption, is not able to characterize the temporal behavior of the signal traces well. Additionally, noisy raw time domain signal traces further contribute to its randomness and poor accuracy.

We intentionally reduced the convergence threshold for the neural network's training that led to larger number of training iterations. The main reason is Zeus's goal to detect malicious executions and not only to classify (previously seen) legitimate control flows. The increased number of iterations resulted in more discriminatory classification outcomes, i.e., more biased probability distribution over the classes (legitimate control flows) and larger likelihood score L. Hence, we were able to increase the classification threshold ε as well (Equation 10). Consequently, in the presence of malicious control flows, Zeus's likelihood score L did not exceed ε. Hence, the flows were classified as malicious correctly. This reduced Zeus's false negative and positive rates.

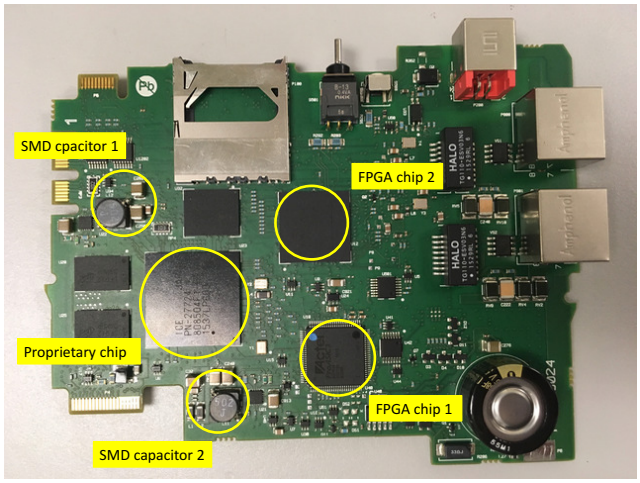
Table 3: Classification accuracy of all evaluation programs over four evaluation settings.

Program	Time_HMM	Time_LSTM	Freq_HMM	**Freq_LSTM**
Matrix	55%	52%	60%	**100%**
Q-sort	49%	60%	41%	**100%**
GD	40%	64%	40%	**98%**
Newton	48%	51%	63%	**100%**
Conv	57%	69%	56%	**100%**
DCT	53%	45%	51%	**94%**
Dijkstra	62%	72%	65%	**100%**
AES	50%	50%	67%	**98%**
PID	40%	62%	71%	**99%**
Patfilt	51%	45%	67%	**100%**

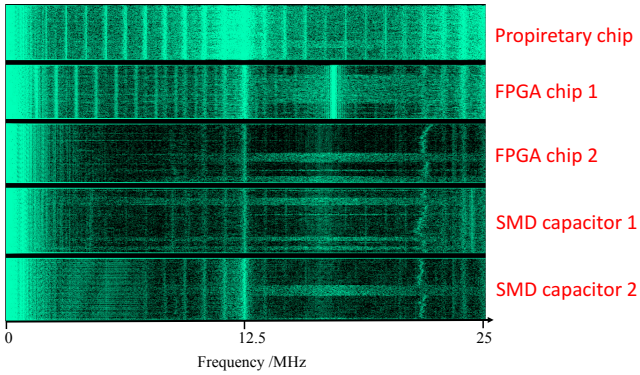
Table 4: Area under curve (AUC) of all evaluated programs over all four evaluation settings.

Program	Time_HMM	Time_LSTM	Freq_HMM	**Freq_LSTM**
Matrix	0.34	0.52	0.90	**0.99**
Q-sort	0.52	0.48	0.76	**1.00**
GD	0.24	0.55	0.86	**0.98**
Newton	0.25	0.62	0.86	**0.99**
Conv	0.62	0.65	0.81	**0.99**
DCT	0.14	0.61	0.81	**0.99**
Dijkstra	0.44	0.51	0.85	**1.00**
AES	0.56	0.57	0.82	**0.96**
PID	0.34	0.66	0.79	**1.00**
Patfilt	0.22	0.73	0.87	**0.99**

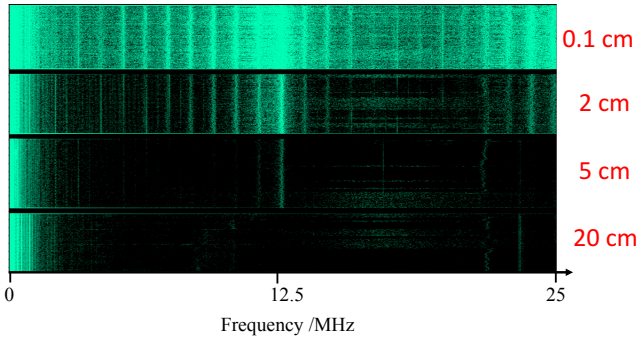
Sliding window size. We investigated the influence of various sliding window sizes on Zeus accuracy. By using window of different sizes, Zeus essentially looks into the program execution at different granularities. A smaller window can capture finer grained transitions in the signal trace, but the frequency resolution of the spectra will be lower. This results in a less discriminative representation of the signal segments. Smaller windows also result in longer data sequence, therefore more recurrences of Zeus's neural network model. This makes the model less robust to random perturbations, since biases on the network states accumulate through the recurrences. A larger window size, on the other hand, will have spectra of better frequency resolution and better robustness, but some small transitions in the signal trace will be ignored.



(a) One of the PLC's three PCB boards: surface area of the propitiatory chip is larger compared to other chips.



(b) Spectrogram for different locations.



(c) Spectrogram for different distances.

Figure 7: EM emanation by the PLC's communication board.

Figure 12 shows how sliding window size affects Zeus accuracy (Figure 12a) and AUC (Figure 12b) both averaged on all ten applications. Using frequency data with a LSTM classifier outperforms all the other settings for all the window sizes. When the size of sliding window increases, both the classification and detection accuracy degrade because of the ignored useful transient information (as discussed above). Too small windows also cause accuracy degradation because of the resulting lower frequency resolution and less discriminative spectra.

6.4 Performance

We measured the time requirements to complete Zeus's classification of the collected EM traces. The required time includes the time of computing the spectrum sequence from the raw time domain EM signal trace if frequency representation is employed, and the time of passing the data sequence through the trained neural network model to get the prediction.

Figure 13 shows the average processing time for one input signal trace and various sliding window sizes. The numbers are averaged over all the ten applications. All the four evaluation settings are able to process the query signal within tens of milliseconds. LSTM-based approaches are overall slower than HMM-based solutions, because passing the input sequences through the network involves more time-consuming array computations. HMM's faster speed comes at the cost of its remarkably lower classification and detection accuracy.

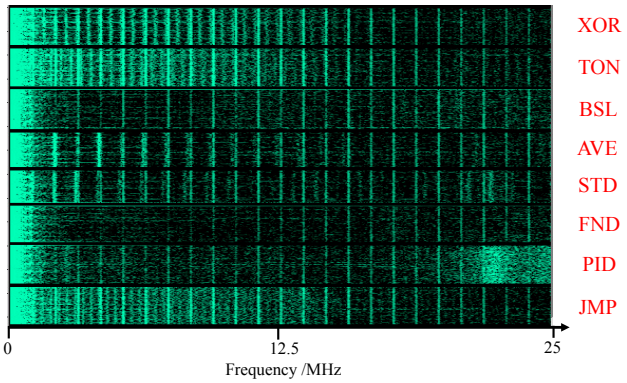
The figure also shows that the larger sliding window sizes lead to reduced time requirements. This is expected as the larger sliding window produce shorter data sequences for a given EM signal trace. Consequently, there are fewer recurrences in the neural network.

7 RELATED WORK

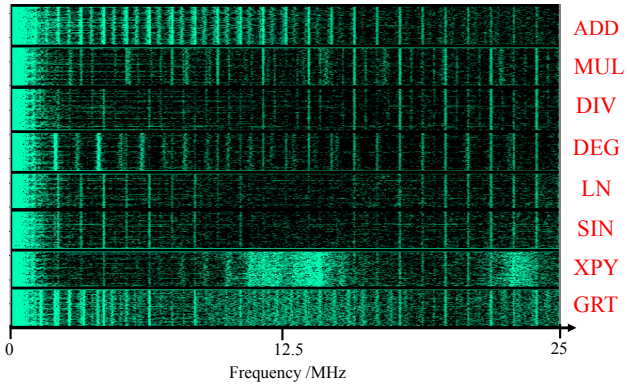
We discuss related work on controller and critical infrastructure security in terms of defense mechanisms and possible attacks.

Side channel analysis. There have been several recent works on analyzing side channels of various modalities for the purpose of inspecting runtime execution. On contactless monitoring, Eisenbarth et al. [9] determine the instruction types (not instances) by modeling individual instructions as HMM states. Msgna et al. [37] perform similar analyses by modeling CFGs as HMMs. The authors assume equal-length basic blocks that is not often the case in practical settings. Other similar HMM-based program behavior modeling have also been studied [15, 59, 61, 62]. On monitoring with contact requirements, cross correlation between the side channels (power [19, 20] and RF traces [50–52]) and the program's single golden execution have been investigated for anomaly detection. However, obtaining a single golden execution is not feasible in practice. A complex PLC program may go through different execution paths depending on the inputs (sensor measurements). Vermon et al. [58] uses power signal side channels to reverse engineer the bytecode running on a Java smart card. Attacks to disclose substitution tables of the A3/A8 algorithm execution [8, 40] have been proposed. These methods focus on recovering the lookup table only. Zeus increases the accuracy of passive side-channel analysis of complete execution profiles using inexpensive contactless EM sensors.

The most related work [31] provides code execution tracking based on the power signal, which requires connections to an 11 *MHz* 8-bit AVR microcontroller. The microprocessor is directly connected to the power supply using a single resistor. The sensor

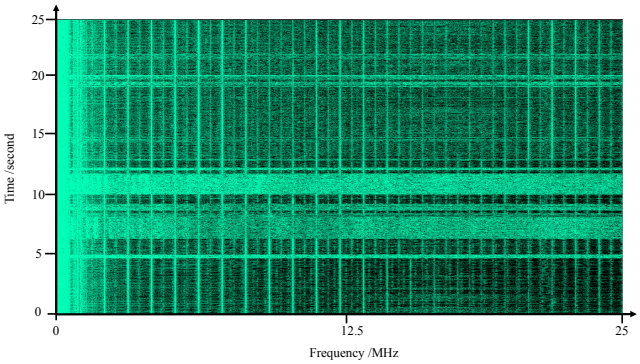


(a) Pattern for PLC instructions.

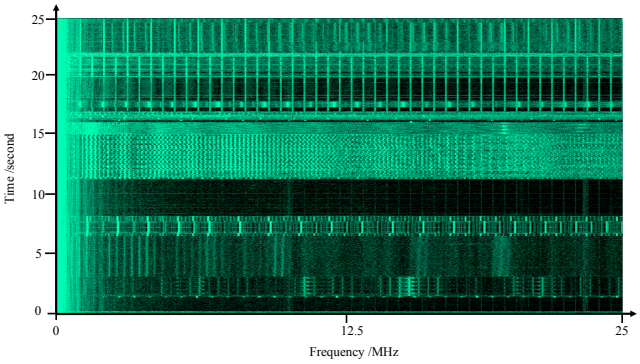


(b) Pattern for PLC instructions.

Figure 8: Spectrogram patterns of PLC instructions.

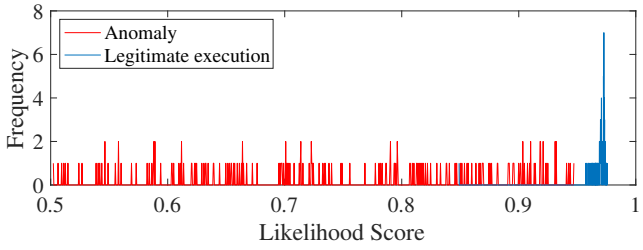


(a) Spectrogram during uploading of PLC program.

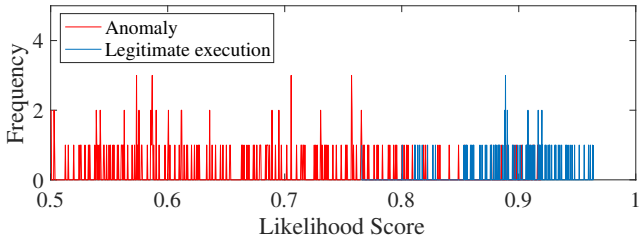


(b) Spectrogram during the booting phase of the PLC.

Figure 9: Spectral patterns of PLC instructions.



(a) Newton's method



(b) AES-128 algorithm

Figure 10: Example likelihood score distributions of the evaluated programs produced by the Freq+LSTM setting.

is a Tektronix MDO3034 oscilloscope with sampling rate of 1.25 *GHz*. Zeus provides *contactless* execution monitoring of *commercial* PLC processors (120 *MHz* ARM Cortex M3 with three separate PCB boards for I/O, logic processing, and power supply) through a different *modality* (electromagnetic emanations) and using *lower-frequency* sensing sampling rates (10 *MHz*) with more than two orders of magnitude saving on the sensor cost.

Another related work [39] also performs execution monitoring and anomaly detection on IoT devices via the electromagnetic side channel. They looks at the prominent frequency peaks in the spectra of the signal segments as feature representations and models program executions with statistical distributions. Zeus uses the sequential neural network model to both extract discriminative features from signal segments and model the control flow transitions in a end to end manner. Moreover, they puts instrumentations at all the loop nests and examines them separately while Zeus looks at full executions without any instrumentation.

Controller program analysis. Although a few processors contain a dedicated hardware unit for execution monitoring, e.g., embedded trace macrocell [6], many embedded controllers lack such

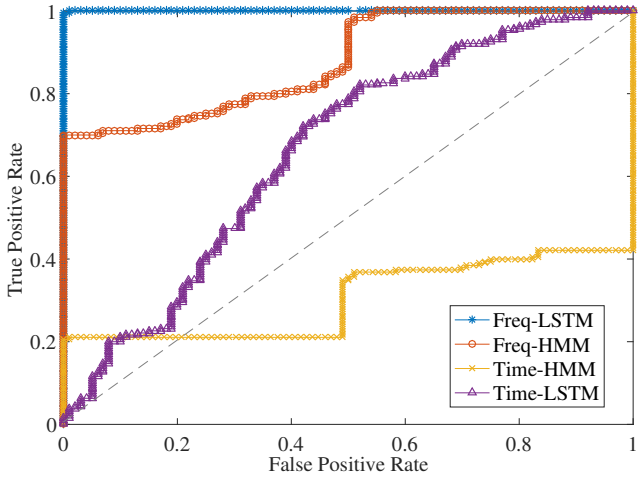


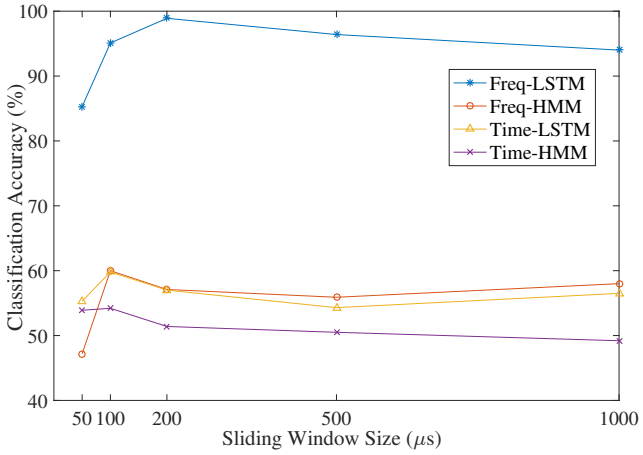
Figure 11: ROC curves of all evaluated programs. AUC of the four settings: Freq-LSTM is 0.99, Freq-HMM is 0.83, Time-LSTM is 0.59, Time-HMM is 0.36.

hardware support. To analyze the software, offline control command verification solutions [21, 34, 42] implement formal methods to verify the safety of the control code immediately before it is executed on the PLC. They face scalability problem, caused by state-space explosion [24, 33, 34, 41]. Consequently, every control logic upload request by the operators, including the emergency cases, should wait for possibly minutes, hours, or more before the code is fully verified for PLC execution. Such delays are often unacceptable for real-time safety-critical control system operations.

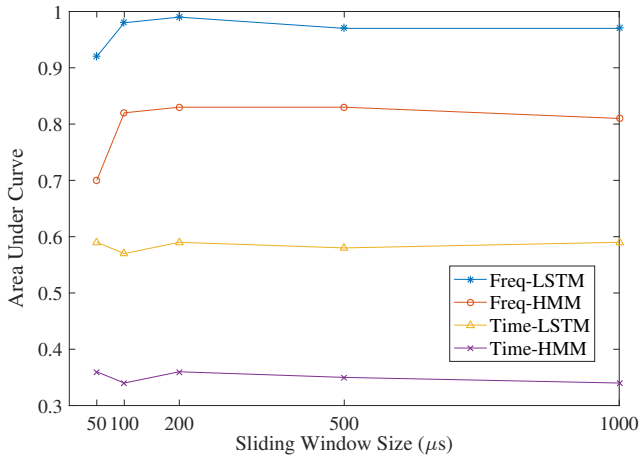
Information security approaches. The related work to protect the control networks' trusted computing base (TCB) are insufficient as software patches are often applied only months after their release [44], while new vulnerabilities are discovered on a regular basis [3, 43]. The traditional perimeter-security tries to keep adversaries out of the protected control system entirely. Attempts include regulatory compliance approaches such as the NERC CIP requirements [56] and access control [14]. Despite the promise of information-security approaches, thirty years of precedence have shown the near impossibility of keeping adversaries out of critical systems [25] and less than promising results for the prospect of addressing the security problem from the perimeter [27, 28, 36]. Embedded controllers from most major vendors [27, 57] and popular Human Machine Interfaces (HMIs) [36] have been shown to have fundamental security flaws.

8 CONCLUSIONS

We presented Zeus, a contactless, passive, and non-intrusive control flow integrity monitoring solution for PLCs. Zeus identifies malicious code execution through side channel analyses of the controller's electromagnetic emanation signals. Zeus's data acquisition is done by an electromagnetic sensor, which provides an air gap between the trusted computing bases of the target PLC and Zeus's monitoring engine. Our empirical studies with commercial PLC



(a) Classification accuracy vs. sliding window size.



(b) Runtime vs. query trace length.

Figure 12: Area under curve vs. sliding window size.

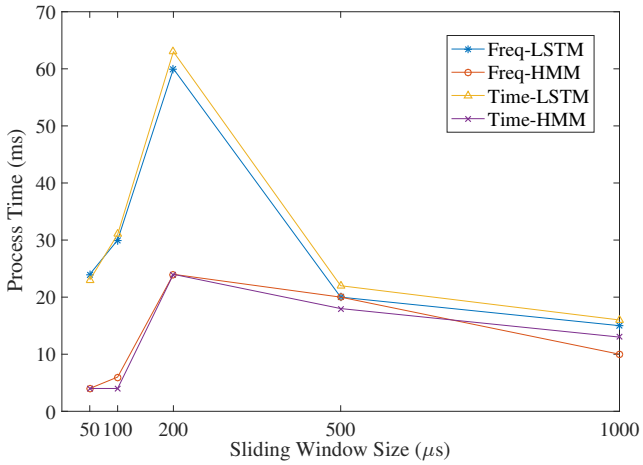


Figure 13: Average process time of all the programs for the four evaluation settings.

controllers and several real application binaries show Zeus can monitor high frequency commercial processors with low frequency sensor sampling. Zeus can detect malicious code executions on popular Allen Bradley PLCs with %98.9 accuracy and with zero runtime overhead by its design.

ACKNOWLEDGEMENT

We would like to thank the National Science Foundation (NSF) for supporting this research project.

REFERENCES

[1] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. 2003. *The EM Side—Channel(s).* Springer Berlin Heidelberg, Berlin, Heidelberg, 29–45. https://doi.org/10.1007/3-540-36400-5_4

[2] Karl Johan Åström and Tore Hägglund. 2006. *Advanced PID control.* ISA-The Instrumentation, Systems and Automation Society.

[3] Dillon Beresford. 2011. Exploiting Siemens Simatic S7 PLCs. In *Black Hat USA*.

[4] Roland Bodenheim, Jonathan Butts, Stephen Dunlap, and Barry Mullins. 2014. Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices. *International Journal of Critical Infrastructure Protection* 7, 2 (2014), 114–123.

[5] Dániel István Buza, Ferenc Juhász, György Miru, Márk Félegyházi, and Tamás Holczer. 2014. CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot. In *Smart Grid Security*. Springer, 181–192.

[6] Dayna A Byrne and Jeffrey J Holm. 2006. Shared embedded trace macrocell. (Feb. 28 2006). US Patent 7,007,201.

[7] Eric Chien, Liam OMurchu, and Nicolas Falliere. 2011. *W32.Duqu - The precursor to the next Stuxnet.* Technical Report. Symantic Security Response.

[8] Christophe Clavier. 2004. Side channel analysis for reverse engineering (SCARE)-an improved attack against a secret A3/A8 GSM algorithm. (2004).

[9] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. 2010. Building a side channel based disassembler. In *Transactions on computational science X*. Springer, 78–99.

[10] sriharsha Etigowni, Dave Tian, Grant Hernandez, Kevin Butler, and Saman Zonouz. 2016. CPAC: Mitigating Attacks against Critical Infrastructure with Cyber-Physical Access Control. In *to appear in Annual Computer Security Applications Conference (ACSAC); available at http://dx.doi.org/10.1145/2991079.2991126.*

[11] European Network and Information Security Agency (ENISA). 2011. Protecting industrial control systems - recommendations for Europe and Member States. https://www.enisa.europa.eu/. (2011).

[12] F-Secure Labs. 2016. BLACKENERGY and QUEDAGH: The convergence of crimeware and APT attacks. (2016).

[13] Nicolas Falliere, Liam O. Murchu, and Eric Chien. 2010. *W32.Stuxnet Dossier.* Technical Report. Symantic Security Response.

[14] David Formby, Preethi Srinivasan, Andrew Leonard, Jonathan Rogers, and Raheem Beyah. 2016. Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems. In *NDSS*.

[15] Debin Gao, Michael K Reiter, and Dawn Song. 2009. Beyond output voting: Detecting compromised replicas using HMM-based behavioral distance. *IEEE Transactions on Dependable and Secure Computing* 6, 2 (2009), 96–110.

[16] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Adi Shamir, and Eran Tromer. 2016. Physical key extraction attacks on PCs. *Commun. ACM* 59, 6 (2016), 70–79.

[17] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. 2016. *ECDSA key extraction from mobile devices via nonintrusive physical side channels.* Technical Report. 1019–1031 pages.

[18] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation* 12, 10 (2000), 2451–2471.

[19] Carlos R Aguayo Gonzalez and Jeffrey H Reed. 2009. Power fingerprinting in SDR & CR integrity assessment. In *MILCOM 2009-2009 IEEE Military Communications Conference*. IEEE, 1–7.

[20] Carlos R Aguayo Gonzalez and Jeffrey H Reed. 2010. Detecting unauthorized software execution in SDR using power fingerprinting. In *Military Communications Conference, 2010-MILCOM 2010*. IEEE, 2211–2216.

[21] J.F. Groote, S.F.M. van Vlijmen, and J.W.C. Koorn. 1995. The Safety Guaranteeing System at Station Hoorn-Kersenboogerd. In *Tenth Annual Conference on Systems Integrity, Software Safety and Process Security*. 57–68.

[22] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.

[23] Tin Kam Ho. 1995. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, Vol. 1. IEEE, 278–282.

[24] Ralf Huuck. 2005. Semantics and Analysis of Instruction List Programs. *Electronic Notes in Theoretical Computer Science* 115 (2005), 3–18.

[25] Vinay M Igure, Sean A Laughter, and Ronald D Williams. 2006. Security issues in SCADA networks. *Computers & Security* 25, 7 (2006), 498–506.

[26] A Iliev, N Kyurkchiev, and S Markov. 2017. On the Approximation of the step function by some sigmoid functions. *Mathematics and Computers in Simulation* 133 (2017), 223–234.

[27] Egor Vladimirovich Kuz'min and Valery Anatolievich Sokolov. 2012. On Construction and Verification of PLC-Programs. *Modelirovanie i Analiz Informatsionnykh Sistem [Modeling and Analysis of Information Systems]* 19, 4 (2012), 25–36.

[28] Ted G Lewis. 2006. *Critical infrastructure protection in homeland security: defending a networked nation.* John Wiley & Sons.

[29] John Leyden. 2008. Polish Teen Derails Tram after Hacking Train Network. http://www.theregister.co.uk/2008/01/11/tram_hack/. (2008).

[30] Yao Liu, Peng Ning, and Michael K Reiter. 2011. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)* 14, 1 (2011), 13.

[31] Yannan Liu, Lingxiao Wei, Zhe Zhou, Kehuan Zhang, Wenyuan Xu, and Qiang Xu. 2016. On Code Execution Tracking via Power Side-Channel. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1019–1031.

[32] Stephen McLaughlin and Patrick McDaniel. 2012. SABOT: specification-based payload generation for programmable logic controllers. In *Proceedings of the 2012 ACM conference on Computer and Communications Security*. 439–449.

[33] Stephen McLaughlin and Patrick McDaniel. 2012. SABOT: specification-based payload generation for programmable logic controllers. In *Proceedings of the 2012 ACM conference on Computer and communications security*. New York, NY, USA, 439–449.

[34] Stephen McLaughlin, Saman Zonouz, Devin Pohly, and Patrick McDaniel. 2014. A Trusted Safety Verifier for Controller Code. In *Network and Distributed System Security Symposium*.

[35] Todd K Moon. 1996. The expectation-maximization algorithm. *IEEE Signal processing magazine* 13, 6 (1996), 47–60.

[36] Thomas H Morris, Anurag K Srivastava, Bradley Reaves, Kalyan Pavurapu, Sherif Abdelwahed, Rayford Vaughn, Wesley McGrew, and Yoginder Dandass. 2009. Engineering future cyber-physical energy systems: Challenges, research needs, and roadmap. In *North American Power Symposium (NAPS), 2009*. IEEE, 1–6.

[37] Mehari Msgna, Konstantinos Markantonakis, and Keith Mayes. 2013. *The B-Side of Side Channel Leakage: Control Flow Security in Embedded Systems.* Springer International Publishing, Cham, 288–304. https://doi.org/10.1007/978-3-319-04283-1_18

[38] John Mulder, Moses Schwartz, Michael Berg, Jonathan Roger Van Houten, Jorge Mario, Michael Aaron King Urrea, Abraham Anthony Clements, and Joshua Jacob. 2013. *WeaselBoard: Zero-Day Exploit Detection for Programmable Logic Controllers.* Technical Report. tech. report SAND2013-8274, Sandia National Laboratories.

[39] Alireza Nazari, Nader Sehatbakhsh, Monjur Alam, Alenka Zajic, and Milos Prvulovic. 2017. EDDIE: EM-Based Detection of Deviations in Program Execution. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 333–346.

[40] Roman Novak. 2003. Side-channel attack on substitution blocks. In *International Conference on Applied Cryptography and Network Security*. Springer, 307–318.

[41] Sidi Ould Biha. 2011. A Formal Semantics of PLC Programs in Coq. In *IEEE 35th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 118–127.

[42] Taeshin Park and Paul I Barton. 2000. Formal Verification of Sequence Controllers. *Computers & Chemical Engineering* 23, 11 (2000), 1783–1793.

[43] Dale G. Peterson. 2012. Project Basecamp at S4. http://www.digitalbond.com/2012/01/19/project-basecamp-at-s4/. (January 2012).

[44] Jonathan Pollet. 2010. Electricity for Free? The Dirty Underbelly of SCADA and Smart Meters. In *Proceedings of Black Hat USA 2010*.

[45] Lawrence R Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286.

[46] Suman Ravuri and Andreas Stolcke. 2016. A comparative study of recurrent neural network models for lexical domain classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 6075–6079.

[47] Julian Rrushi, Hassan Farhangi, Clay Howey, Kelly Carmichael, and Joey Dabell. [n. d.]. A Quantitative Evaluation of the Target Selection of Havex ICS Malware Plugin. ([n. d.]).

[48] Jill Slay and Michael Miller. 2007. Lessons Learned from the Maroochy Water Breach. In *Critical Infrastructure Protection*. Springer, 73–82.

[49] Armando Solar-Lezama. 2008. *Program synthesis by sketching.* ProQuest.

[50] Samuel Stone and Michael Temple. 2012. Radio-frequency-based anomaly detection for programmable logic controllers in the critical infrastructure. *International Journal of Critical Infrastructure Protection* 5, 2 (2012), 66–73.

[51] Samuel J Stone. 2013. *Radio frequency based programmable logic controller anomaly detection.* Technical Report. DTIC Document.

[52] Samuel J Stone, Michael A Temple, and Rusty O Baldwin. 2015. Detecting anomalous programmable logic controller behavior using RF-based Hilbert transform features and a correlation-based verification process. *International Journal of Critical Infrastructure Protection* 9 (2015), 41–51.

[53] TechNavio. 2014. Global Industrial Control Systems (ICS) Security Market 2014-2018. http://www.technavio.com/report/global-industrial-control-systems-ics-security-market%C2%A02014-2018. (2014).

[54] Steven Tom, Dale Christiansen, and Dan Berrett. 2008. Recommended pratice for patch management of control systems. *DHS control system security program (CSSP) Recommended Practice* (2008).

[55] Abril Valeria Uriarte-Arcia, Itzamá López-Yáñez, and Cornelio Yáñez-Márquez. 2014. One-hot vector hybrid associative classifier for medical data classification. *PloS one* 9, 4 (2014), e95715.

[56] U.S. Department of Energy Office of Electricity Delivery and Energy Reliability. 2015. North American Electric Reliability Corporation Critical Infrastructure Protection (NERC-CIP) Standards; available at http://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx. (2015).

[57] Sidney E Valentine. 2013. *PLC code vulnerabilities through SCADA systems.* Ph.D. Dissertation. University of South Carolina.

[58] Dennis Vermoen, Marc Witteman, and Georgi N Gaydadjiev. 2007. Reverse engineering java card applets using power analysis. In *IFIP International Workshop on Information Security Theory and Practices*. Springer, 138–149.

[59] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. 1999. Detecting intrusions using system calls: Alternative data models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE, 133–145.

[60] Candid Wueest. 2014. Targeted attacks against the energy sector. *Symantec Security Response, Mountain View, CA* (2014).

[61] Kui Xu, Danfeng Daphne Yao, Barbara G Ryder, and Ke Tian. 2015. Probabilistic program modeling for high-precision anomaly classification. In *2015 IEEE 28th Computer Security Foundations Symposium*. IEEE, 497–511.

[62] Dit-Yan Yeung and Yuxin Ding. 2003. Host-based intrusion detection using dynamic and static behavioral models. *Pattern recognition* 36, 1 (2003), 229–243.

[63] Kun Yuan, Qing Ling, and Wotao Yin. 2016. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization* 26, 3 (2016), 1835–1854.

[64] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing* (2017).