

NIPAD: a non-invasive power-based anomaly detection scheme for programmable logic controllers*

Yu-jun XIAO¹, Wen-yuan XU^{†1}, Zhen-hua JIA², Zhuo-ran MA¹, Dong-lian QI¹

(¹School of Electrical Engineering, Zhejiang University, Hangzhou 310027, China)

(²Wireless Information Network Laboratory, Rutgers University, North Brunswick, NJ 08902, USA)

E-mail: xiaoyujun2@gmail.com; xuwenyuan@gmail.com; zhenhua@winlab.rutgers.edu; mzm@zju.edu.cn; qidl@zju.edu.cn

Received Sept. 9, 2016; Revision accepted Nov. 30, 2016; Crosschecked Mar. 23, 2017

Abstract: Industrial control systems (ICSs) are widely used in critical infrastructures, making them popular targets for attacks to cause catastrophic physical damage. As one of the most critical components in ICSs, the programmable logic controller (PLC) controls the actuators directly. A PLC executing a malicious program can cause significant property loss or even casualties. The number of attacks targeted at PLCs has increased noticeably over the last few years, exposing the vulnerability of the PLC and the importance of PLC protection. Unfortunately, PLCs cannot be protected by traditional intrusion detection systems or antivirus software. Thus, an effective method for PLC protection is yet to be designed. Motivated by these concerns, we propose a non-invasive power-based anomaly detection scheme for PLCs. The basic idea is to detect malicious software execution in a PLC through analyzing its power consumption, which is measured by inserting a shunt resistor in series with the CPU in a PLC while it is executing instructions. To analyze the power measurements, we extract a discriminative feature set from the power trace, and then train a long short-term memory (LSTM) neural network with the features of normal samples to predict the next time step of a normal sample. Finally, an abnormal sample is identified through comparing the predicted sample and the actual sample. The advantages of our method are that it requires no software modification on the original system and is able to detect unknown attacks effectively. The method is evaluated on a lab testbed, and for a trojan attack whose difference from the normal program is around 0.63%, the detection accuracy reaches 99.83%.

Key words: Industrial control system; Programmable logic controller; Side-channel; Anomaly detection; Long short-term memory neural networks

<http://dx.doi.org/10.1631/FITEE.1601540>

CLC number: TP309.1

1 Introduction


Industrial control systems (ICSs), which include supervisory control and data acquisition (SCADA) systems, distributed control systems (DCSs), and programmable logic controllers (PLCs), are cyber-physical systems that automate industrial processes (Stouffer *et al.*, 2011). ICSs have been widely used to

supervise and control most of the physical processes in critical infrastructures including energy, water, transportation systems, the chemical industry, the nuclear industry, critical manufacturing, food, and healthcare (Alcaraz and Zeadally, 2015). Not surprisingly, the security of an ICS has a direct effect on the physical world, affecting property safety, human lives, and even national security.

When ICSs were first deployed, many of their components were in physically secured areas and were not connected to the Internet. Thus, security was not a major concern and the primary goals were

[†] Corresponding author

* Project supported by the National Basic Research Program (973) of China (No. 2015AA050202)

 ORCID: Wen-yuan XU, <http://orcid.org/0000-0002-2428-973X>
© Zhejiang University and Springer-Verlag Berlin Heidelberg 2017

functional safety and system effectiveness. Protocols such as MODBUS, Ethernet/IP, and ISO-TSAP were used to communicate with field devices and PLCs in an ICS. Although these protocols work effectively, they do not consider attacks because an ICS is believed to be isolated from public networks. With the development of information technology (IT), ICSs are increasingly connected to the Internet, which increases chances of exposing the critical infrastructure to remote attacks. Recently, Stuxnet raised an alarm and demonstrated that ICSs are highly vulnerable to remote malicious manipulation and ICS security needs to be revisited (Chen and Abu-Nimeh, 2011).

In reality, the attacks against ICSs have a larger impact than those on IT systems. In 2000, a sewage treatment system in Maroochy Shire, Queensland was hacked by a disgruntled employee through a laptop and a wireless radio. As a result, the raw sewage spilt out into local parks and rivers, causing environmental pollution (Slay and Miller, 2007). In 2003, the Davis-Besse nuclear plant in Ohio was hacked because of a worm, and the nuclear plant was unable to work for several hours (Johnson, 2010). In 2006, the Browns Ferry nuclear plant in the United States was hacked and the nuclear reaction unit was forced to close (Kesler, 2011). In 2009, the two metro trains collided because of an attack, causing passenger injuries and deaths (Johnson, 2010). In 2010, Stuxnet swept around the world, and more than 45 000 networks were infected (Chen and Abu-Nimeh, 2011). In 2012, the most sophisticated computer malware ever seen in industry, Flame, raged in the Middle East, and was 20 times more harmful than Stuxnet (Bencsáth *et al.*, 2012). In 2014, Havex started a full-scale attack against SCADA systems in Europe (Pretorius and van Niekerk, 2016). Similar types of attacks are likely to emerge over time, and protecting ICSs against existing and future attacks is important.

Compared to IT systems, the attacks against an ICS can affect the physical world and cause more devastating impact. Since critical infrastructures are the basis of national economy and people's daily life, their security and protection are extremely important. After the discovery of Stuxnet, many security mechanisms such as the network-based intrusion detection system (IDS) and signature-based antivirus software were used to enhance the security of ICSs.

However, traditional cyber security mechanisms have limitations when being applied to ICSs. On the one hand, these approaches are not applicable to all devices in an ICS, especially the embedded devices that are resource-constrained and cannot execute antivirus software. On the other hand, vendors of ICSs may refuse to load third-party software or upgrade their devices to include security software, afraid of being responsible for third-party software for potential safety incidents.

Motivated by the special characteristics of ICSs, we focus on protecting the PLC, which is one of the most critical components in an ICS and cannot be protected by traditional IDS or antivirus software. PLCs control actuators directly and if a PLC is executing a malicious program instead of the original one, it can damage the physical world dramatically, ranging from property damage to loss of life. Already, an increasing number of attacks have targeted PLCs, showing the importance of ensuring the proper execution of a PLC and detecting abnormal scenarios. Thus, we propose NIPAD, a non-invasive power-based anomaly detection scheme for PLCs. The basic idea of NIPAD is to detect whether a PLC is executing its normal program or a malicious program by analyzing its power consumption. Based on NIPAD, we are able to implement a real-time monitoring system to detect abnormal execution. Compared to traditional cyber-security mechanisms, NIPAD is ideal for anomaly detection in ICSs because of the following three advantages. First, it is non-invasive, and requires no software modification of the original ICS. Since NIPAD is a side-channel based method, it does not interact with the original ICS, neither does the ICS need to install third-party software. Second, NIPAD does not need to be trained over abnormal power consumption. Since abnormal samples are rare and difficult to get in practice, we design NIPAD to be independent toward the attacks and able to detect both existing attacks and the ones that may emerge over time. Third, the system is renewable. Once a more effective anomaly detection algorithm is designed, NIPAD is able to update the algorithm.

2 Motivation and goals

ICSs control critical infrastructures, mass production lines, etc., and thus an ICS is an attractive

target since attacks can cause serious consequences. The attackers can range from hostile governments, terrorist groups, to industrial spies. In addition, malpractice from careless employees or intentional misconduct from disgruntled employees can cause a similar negative impact (Macaulay and Singer, 2011). To reduce the impact, it is important to detect attacks and anomalies in a timely manner and the monitoring system should require minimum modification of the original ICS. In this section, we first introduce the basics of an ICS, and then discuss the attack model as well as the goals of our monitoring system.

2.1 Background

Although we focus on monitoring the execution of a PLC, we introduce a simplified ICS below to provide necessary background to understand our system. A typical ICS consists of the following components (Fig. 1):

1. Control server

The control server is used to communicate with lower-level control devices through the supervisory control software.

2. Human-machine interface (HMI)

The HMI includes both software and hardware, providing an interface for human operators to monitor and control the state of the control processes. HMI software such as WinCC is able to provide a centralized monitoring of the numerous process inputs and outputs.

3. Data historian

The data historian is a centralized database that stores all process information within an ICS.

4. Programmable logic controller (PLC)

The PLC is a small industrial computer with the capability of controlling complex industrial processes, and is originally designed to control the logic executed by electrical hardware such as switches, relays, and timers. It is widely used in an ICS as a real-time control unit because of its strength in being economical, flexible, configurable, and versatile.

5. Sensors and actuators

There are various sensors for measurement and actuators for control in an ICS, such as pressure and temperature sensors, valves, motors, robotics, relays, and breakers.

Fig. 1 shows a simplified model of the critical components for an ICS, whereby there is one primary control center and some field sites. The con-

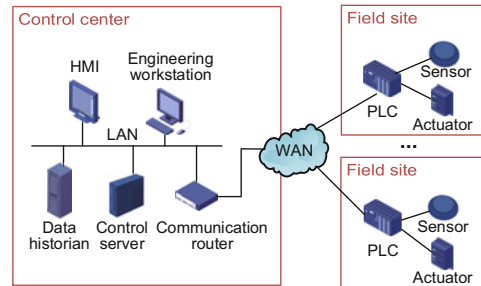


Fig. 1 A simplified model of an industrial control system (ICS). The PLC is the core component in an ICS interacting with field devices

control center includes the control server, communication routers, an HMI, engineering workstations, and a data historian (Stouffer *et al.*, 2011). All of them are connected by a local area network (LAN). The control center is responsible for centralized alarming and reporting. It collects and logs information from the field sites, and may generate instructions to handle events. The operator in the control center is able to remotely conduct real-time monitoring as well as control for an entire system. A field site consists of a control unit PLC, various sensors, and actuators.

For a basic control process, controlled variables are reported to PLCs from the sensors, and PLCs process the signals and generate corresponding control commands to the actuators. The programs in PLCs for a control process are downloaded from an engineering workstation, and the HMI displays process status information. As the core component in an ICS, PLC provides control interfaces to the control center and allows human operators to control field devices and industrial production.

Each component in an ICS can be the target of an attack, which can include eavesdropping on the secret data in an ICS, disrupting the communication between the control center and field sites, and modifying the control instructions in PLCs. Firewalls and network-based intrusion detection systems are applied to the ICS to prevent the control center from being attacked. However, an effective method for protecting PLCs is yet to be designed. Thus, we focus on detecting the attacks against PLCs. We envision that an attack wants to damage the actuators or control PLCs by executing instructions other than the pre-defined program. Thus, our goal is to identify whether the PLC is running a pre-defined program or a different program.

2.2 Threat model

We consider an attacker who aims at downloading a malicious program to the PLC to cause damage. The attacks targeted at modifying control processes may be classified into three types: (1) reducing the service life of controlled objects secretly—such an attack can take a long time to cause damage; (2) damaging the controlled objects immediately to cause property loss; (3) modifying the control goal to deceive the controlled variables into an abnormal state, such as an abnormal temperature, pressure, and liquid level, which may result in environmental pollution, property losses, or even personnel casualties.

The potential attackers are summarized in Fig. 2. Not only do we need to pay attention to the potential damage caused by careless or disgruntled engineers downloading incorrect instructions, but also we want to prevent adversaries from attacking the system via loopholes or viruses. The adversaries can be terrorists who are able to obtain the control of a PLC through networks, an industrial spy who lurks in the factory, or a disgruntled employee who controls the PLC. We detail the attack scenarios and capability below.

1. Motivated attackers

Motivated attackers break into the ICS through networks and may have control of the control center and PLCs. They are usually well-prepared and sophisticatedly organized. Their purpose is to destroy or exploit critical infrastructures to threaten national security. To cause such damage, they may first disturb the communication between the PLC and the control center, and turn off the alarms. Then they may modify the control instructions running in PLCs. To mask their attack, they may send fault information to the control center. For instance, Stuxnet is sophisticatedly designed. First, Stuxnet infected the local area network and controlled PLCs to execute problematic instructions, to cause the centrifuges to spin much faster than they should. At the same time, the attackers prerecorded a period of 21 s of normal input signals and replayed them while attacking the system. Those signals were ultimately shown on an HMI screen (Langner, 2011), and the operators may not be aware of the abnormal situation before it is too late.

2. Industrial spies

The purpose of industrial spies is to hinder the

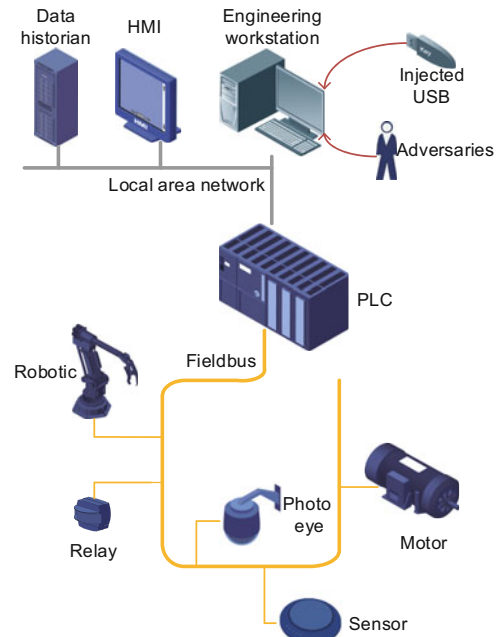


Fig. 2 Schematic of attack models. Threats may come from disgruntled engineers or hackers

development of their competitors. We may consider a spy who lurks in the factory for a chance to control the PLC and he/she may perform a launch-break attack. He/She has limited time to manipulate the PLC, and thus may download a trojan program to the PLC. Most of the time, the program may run normally while it may perform harmful tasks at midnight. As a result, the productive process may be affected.

3. Disgruntled employees

Employees who are disgruntled with the factory may wish to cause damage. With experience and knowledge of controlling PLCs, they may perform attacks covertly by changing the information sent to the control center, and then they may change the programs running in the PLC to destroy the production process secretly.

2.3 Monitoring goals

To cope with all the aforementioned attacks, we believe that an effective monitoring system should have the following qualities:

1. Distinguishing different programs

Our monitoring system should identify the abnormal programs from the original ones. Once it detects the trojan program, it raises an alert to operators.

2. Non-invasive

The monitoring system is supposed to be non-invasive and requires almost no modification to the original ICS. Thus, the proposed method will not affect the functionality of the existing system and will not give rise to extra security issues.

3. Detecting abnormalities in real time

The monitoring system is able to conclude whether the PLCs are running normally or abnormally within a reasonable delay.

4. Renewable and updatable

If the proposed methods need to be updated when a better version becomes available, it should be able to update the monitoring system online without disturbing the control system.

We define the goals as follows: Suppose there is a time series $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}]$, where each point $\mathbf{x}^{(t)}$ represents the attribute vector as the PLC runs. Note that $\mathbf{x}^{(t)}$ is an m -dimensional vector $[x_1^{(t)}, x_2^{(t)}, \dots, x_m^{(t)}]$, whose elements indicate the attributes. When the PLC is running a program A , we obtain a time series of attributes $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(t)}]$. Given those normal time series, we will determine whether the next time series of attributes $\mathbf{x}^{(t+1)}$ are from the pre-defined program A or an abnormal program B .

3 Overview

In this section, we briefly overview the components in NIPAD and explain how each component works. We show the theoretical foundation of NIPAD and our preliminary results.

3.1 Feasibility of NIPAD

The key idea of NIPAD is to distinguish the internal state of PLCs through side-channel analysis. Once the program running in PLC is different from the pre-defined one, we may conclude that the PLC's execution is abnormal and is likely to be attacked. We choose power consumption as our analysis basis because it is easy to collect and not susceptible to environmental influence compared to other side-channel signals, like electromagnetic interference (EMI). In addition, the power consumption of a PLC is closely determined by the executing programs.

PLCs execute a program in a cycle-scanning manner, and a cycle consists of self-diagnosis, com-

munication, input sampling, user program execution, and output refresh (Bolton, 2015). The CPU of a PLC is a microprocessor or a microcontroller, and thus the power consumption varies when PLCs execute different instruction sets. For different programs, the power consumption is different when calling the communication module, registers, or memories. In addition, the input sampling and output refresh processes impose various amounts of power consumption. As a result, the CPU power consumption of a PLC varies and is determined by the executing programs.

In summary, PLCs have the following characteristics that make it feasible to detect execution status by power analysis:

1. PLCs tend to execute a pre-defined sequence of instructions within a period of time, and the sequence will not be modified frequently.

2. PLCs behave in a cyclic manner, and their power consumption may manifest a fixed pattern. The scanning cycle and energy consumption pattern depend on the executing program.

3. Most PLCs have a separate alternating current to direct current (AC-DC) converter, and thus we can measure the power consumption at the DC power supply without modifying the hardware or software.

4. For reliability, PLCs often have redundant power sources. As a result, those devices do not need to be shut down while the resistor is between the CPU module and GND to facilitate power measurement.

We perform simple experiments on a Siemens S7-200 to verify the feasibility of power analysis. Siemens S7-200 is a micro PLC that can control relatively simple automation applications. We run three programs in S7-200, and collect the power consumption while it is running each of them. Program A completes a cycle every two seconds. It opens a valve during the first second and closes it during the following second. Program B behaves in the same way with four valves. Program C completes a cycle every five seconds. It closes a valve during the first four seconds and opens it during the following second. From Fig. 3, we can see that the power consumption of the CPU differs while each program is being executed. The preliminary results encourage us to perform a larger-scale and long-term study. To understand the reliability of identifying the execution of a program,

we also want to verify whether the system is able to differentiate power consumption of two programs that have a difference of just a few lines.

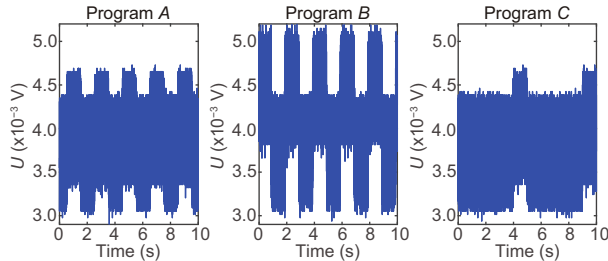


Fig. 3 The voltage when PLC S7-200 executes different programs. It illustrates how power consumption is determined by programs

3.2 Components of NIPAD

To differentiate power consumption between the pre-defined program and unknown programs effectively, NIPAD consists of data acquisition (DAQ), pre-processing, feature extraction, and anomaly detection modules (Fig. 4).

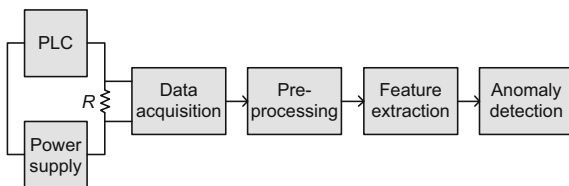


Fig. 4 Major components of NIPAD

To collect the power consumption of a PLC, we insert a current shunt resistor between its CPU module and power supply (PS) module, which is an AC-DC converter. Since most PLCs are modular, they have an isolated PS module, a CPU module, a communication processor (CP) module, and input and output (I/O) modules. Inserting a current shunt resistor requires no device-level modifications. The inserted resistor is 0.1Ω , which is small enough to avoid side effects on the original circuit, and we will explain it in detail in Section 4. Because the resistance is a constant value, the voltage drop indicates the current of the CPU as well as its power consumption. Thus, measuring the power consumption of the PLC is equivalent to measuring the voltage drop over the resistor.

We use data acquisition equipment to collect the voltage drop across the resistor. In particular, we use an Agilent U2541A, which has the ability of

saving signals to a file with a maximum sampling rate of 250 kHz. We customize a data acquisition module that is able to transfer the signal over Ethernet. With the module, we are able to implement real-time monitoring.

After receiving the time-domain signal, we first perform a pre-processing procedure including removing the DC signal data and filtering out the high-frequency noise. Then we extract a variety of features from those time-domain signals. To achieve a better classification and reduce training complexity, we use sparse coding algorithms (Lee *et al.*, 2006) to choose a proper feature combination. The aim of sparse coding algorithms is to find a set of basis vectors that are able to represent the input vector. Sparse coding is proven to be an effective method for feature selection when the feature matrix is sparse. Since there are no abnormal samples, we can use only positive samples for training to predict negative samples. We choose long short-term memory (LSTM) neural networks (Gers *et al.*, 2000) for anomaly detection. An LSTM neural network is an improved algorithm from recurrent neural networks, which is well-suited for training a prediction model to predict the following time series. We then recognize the anomalies by comparing the predicted time series with real-time series. To demonstrate its effectiveness, we also use one-class support vector machine (SVM) (Manevitz and Yousef, 2002) and a correlation-based anomaly detection algorithm (Nandakumar and Jain, 2004) for comparison.

To detect the abnormal programs effectively, there are a few challenges for NIPAD, which are listed below:

1. How to collect data accurately without affecting the original operation of PLCs? How to reduce the data fluctuation caused by noise?

2. How to construct proper feature space? Namely, how to extract abundant features and then select the most discriminative features from them?

3. Since there are almost no abnormal samples in actual industrial control processes, and we are not allowed to conduct an attack on a real control system, how to detect abnormal program execution with only the power traces of normal programs?

We will describe how we overcome these challenges to NIPAD in detail in the following sections.

4 Technical details

In this section, we describe the critical components of NIPAD in detail. We explain how to collect relatively stable power traces, how to construct proper feature sets, and how to apply LSTM neural networks for anomaly detection.

4.1 Power consumption acquisition

We use a 0.1- Ω current shunt resistor to collect the power consumption of a Siemens S7-300 PLC, which is a modular PLC and has individual PS, CPU, and I/O modules. The schematic is shown in Fig. 5, and we are able to collect the power consumption through measuring the voltage drop of the resistor. Inserting such a shunt resistor is safe because the equivalent resistance of the PLC is above 100 Ω and the resistor is 0.1 Ω . On the one hand, the power consumption of the PLC will be at least a thousand times larger than the power consumption of the resistor. Thus, inserting the resistor does not affect the functionality of the PLC components, and the power consumption of the resistor is negligible. On the other hand, the current through the PLC is around several hundred milliamperes, and thus the power consumption of the resistor is less than a milliwatt, which is far less than the resistor's rated power and will not cause a circuit break. As a result, inserting a 0.1- Ω resistor between the PS module and CPU module will not have a side effect on the original circuit.

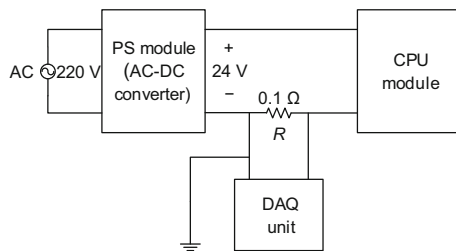


Fig. 5 Schematic of power consumption acquisition. The power supply (PS) module and DAQ unit should be commonly grounded

An alternate method to measure power consumption is a closed-loop hall current sensor. Although a hall sensor can achieve complete isolation from the PLC circuits, it is easily susceptible to spatial electromagnetic disturbance and can provide neither satisfactory frequency responses to high-frequency power consumption change nor satisfac-

tory precision. Because of the modularity of PLCs, we are able to insert a resistor between the PS module and CPU module easily. For safety, factories tend to allocate redundant PS modules for PLCs. Thus, the process of inserting a resistor does not need to interrupt the operation of PLCs.

To collect the voltage drop of the resistor, we use an Agilent U2541A data acquisition unit with a sampling rate of 250 kHz. The unit should be properly grounded; namely, the PS module, the resistor, and the unit should be commonly grounded. This is because there is high impedance between DC earth and AC earth, and with the impact of spatial electromagnetic field, there would be a voltage difference between the ground of the PS module and the DAQ unit. Thus, to obtain a correct signal and avoid any common mode interference, the PS module and the DAQ unit need to be commonly grounded.

4.2 Feature construction

To effectively distinguish normal signals and abnormal signals, it is essential to construct proper feature sets, including original feature extraction and feature selection. The measured time-domain signals are segmented by a window size of 5 s, which is called a sample. A statistical histogram (Pearson, 1901) offers time robustness, and is widely used for original feature construction in digital image processing (Lowe, 2004; Dalal and Triggs, 2005; Wang et al., 2013). To construct a rich feature space, we first choose the statistical histogram of a sample as our feature. Feature f_j is defined as follows:

$$f_j = \begin{cases} |\{m : m < H_l, m \in M\}|, & j = 1, \\ |\{m : t_{j-2} < m < t_{j-1}, m \in M\}|, & 1 < j < N, \\ |\{m : m > H_h, m \in M\}|, & j = N. \end{cases} \quad (1)$$

Here, M is the set of current measurements in a sample, N defines the number of features, H_l denotes the threshold chosen to estimate the global minimum of the measurements, and H_h denotes the global maximum of the measurements. We divide the measurements into N blocks by the following threshold:

$$t_i = i \frac{H_h - H_l}{N - 2} + H_l.$$

We choose N as 102, H_l as 0.11 A, and H_h as 0.16 A. Thus, we obtain 102 original features.

We also use the LibXtract library (Bullock and

Conservatoire, 2007) for feature extraction. This is a cross-platform software library that can be used for extracting low-level features of time series. We extract 17 time-domain features and 14 frequency-domain features as our original features. Thus, we have 133 features in total, including 102 histogram features and 31 features from the LibXtract library. However, the original features are redundant and noisy. To reduce over-fitting and shorten the training time, we need to find a proper feature subset of the original features by removing the redundant features and picking out the discriminative features. An optimal feature combination is supposed to differentiate the programs at the maximum level. A proper solution is to calculate the contribution of each feature, and then give them a corresponding weight. However, it is too time-consuming for so many features. Another solution is to regard the feature selection problem as a combinatorial optimization problem, and sparse coding has proved to be an efficient solution to this problem.

Sparse coding is applied for learning a set of over-complete basis vectors to reconstruct a signal (Xu et al., 2013), and is widely used in image denoising, speech signal processing, and feature selection (Zhong et al., 2012; Ni et al., 2015). It is one of the most effective methods for feature selection when the feature matrix is sparse.

To select an optimal feature subset from the original feature space with sparse coding, we assume a linear classifier model, which is used to predict the label of a sample:

$$f = \mathbf{w}^T \mathbf{x} + b, \quad (2)$$

where f is the predicted label of a sample, \mathbf{x} an N -dimensional feature vector, \mathbf{w} the corresponding coefficients of the classifier, and b a bias.

We assume that y is the actual label of the sample. For a classifier, the primary goal is to compute the optimal parameter vector \mathbf{w} which minimizes the loss function between f and y . We choose a simple yet efficient function from various options of the loss function, which is the quadratic loss function:

$$\arg \min_{\mathbf{w}, b} \sum_{i=1}^M [y_i - (\mathbf{w}^T \mathbf{x}_i + b)]^2, \quad (3)$$

where M is the number of samples and the input \mathbf{x}_i should be normalized.

The original feature space in our method is rich yet redundant, and there may be some noise. Actually, only a small subset of original features are discriminative. Thus, the original features to some extent are sparse, which means that we need to minimize the amount of nonzero data in \mathbf{w} while minimizing the quadratic loss function. We can represent the goal as follows:

$$\arg \min_{\mathbf{w}, b} \left\{ \sum_{i=1}^M [y_i - (\mathbf{w}^T \mathbf{x}_i + b)]^2 + \lambda \|\mathbf{w}\|_0 \right\}, \quad (4)$$

where λ is a parameter for adjustment.

It is known that solving the ℓ_0 -norm problem is NP-hard, and the ℓ_0 -norm can be transferred to the ℓ_1 -norm, which can be solved with the ℓ_1 -regularized least squares method (Candes and Tao, 2006). Thus, the feature selection problem can be summarized as

$$\arg \min_{\mathbf{w}, b} \left\{ \sum_{i=1}^M [y_i - (\mathbf{w}^T \mathbf{x}_i + b)]^2 + \lambda \|\mathbf{w}\|_1 \right\}. \quad (5)$$

To estimate the optimal feature subset, we need M_p positive samples and M_n negative samples, and each sample is N -dimensional. Namely, \mathbf{x}_i represents the i th sample which consists of N features, and y_i is the actual label of the i th sample. We use '+1' to label positive samples and '-1' for negative samples. The solution of Eq. (5) is a sparse vector \mathbf{w} , and the nonzero elements in it correspond to discriminative features which are selected from the original N -dimensional feature space. We may choose a proper feature dimensionality by adjusting λ .

We finally choose 12 discriminative features from the original feature space with the sparse coding algorithm. The features are mean, sum, skewness, spectral_mean, rms_amplitude, and seven features from Eq. (1). The definitions of these features are as follows:

$$\text{mean} : \bar{x} = \frac{1}{N} \sum_{k=1}^N x_k, \quad (6)$$

$$\text{sum} : \sum_{k=1}^N x_k, \quad (7)$$

$$\text{skewness} : \frac{1}{N} \sum_{k=1}^N \left(\frac{x_k - \bar{x}}{\sigma} \right)^3, \quad (8)$$

$$\text{spectral_mean} : \frac{\sum_{k=1}^N f_k a_k}{\sum_{k=1}^N a_k}, \quad (9)$$

$$\text{rms_amplitude} : \sqrt{\frac{1}{N} \sum_{k=0}^{N-1} a_k^2}. \quad (10)$$

Here, N is the number of power data in a sample, x_k the power value of the k th data, σ the standard deviation, f_k the k th frequency after frequency transformation, and a_k the energy of f_k .

4.3 Anomaly detection

Since the abnormal (or attack) events occur rarely and we cannot conduct an attack on a real control system, detecting unknown attacks is a must, which means that we are supposed to detect the abnormal programs without any negative sample. There are various anomaly detection methods, among which the correlation-based method and the one-class SVM based method are two of the most widely used methods. LSTM networks have recently proved to be a useful method for anomaly detection in time series (Malhotra *et al.*, 2015). We use LSTM networks for anomaly detection, and prove that LSTM networks work better than one-class SVM and the correlation-based method for time series through experiments, as shown in Section 5.

An LSTM network is an artificial neural network that contains LSTM blocks which overcome the vanishing gradient problem by employing multiplicative gates that enforce constant error flow through the internal state. An LSTM network is well-suited for capturing the structure of time series and predicting time series at different time scales. For a time series $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}]$, where $\mathbf{x}^{(t)}$ is an m -dimensional vector $[x_1^{(t)}, x_2^{(t)}, \dots, x_m^{(t)}]$ which represents the features of a sample, we train a prediction model learning from the former n samples to predict the next sample of m features. That is, we take $n \times m$ units in the input layer and m units in the output layer. The LSTM units consist of two hidden layers, and each unit in a lower LSTM hidden layer is fully connected to each unit in the LSTM hidden layer above it through feedforward connections. For each of the m features, there is an error $e_i^{(t)}$ between $x_i^{(t)}$ and its value as predicted. Thus, we compute an error vector $\mathbf{e}^{(t)} = [e_1^{(t)}, e_2^{(t)}, \dots, e_m^{(t)}]$ for each sample $\mathbf{x}^{(t)}$. We then consider an observation $\mathbf{x}^{(t)}$ as anomaly when the error vector $\mathbf{e}^{(t)}$ meets the requirement of Eq. (11). We can improve the detec-

tion performance through adjusting threshold τ :

$$\sum_{i=1}^m \frac{e_i^{(t)}}{x_i^{(t)}} > \tau. \quad (11)$$

5 Evaluation

To demonstrate the efficiency of NIPAD, we build a testbed which simulates the liquid control process in a real factory. In addition, we conduct three trojan attacks that correspond to three kinds of attacks mentioned in Section 2. The attacks pretend to be the normal programs that cannot be discovered as abnormalities from the HMI monitoring picture. This section describes our experiments and results on the testbed.

5.1 Evaluation metrics

In this study, we use two metrics to evaluate our method, which are accuracy and equal error rate (EER). Accuracy is the percentage of the number of correctly labeled samples out of the total number of samples, and a higher accuracy means a better classification performance. EER is the false acceptance rate (FAR) where it equals the false rejection rate (FRR). EER is an evaluation metric of a classification algorithm and a smaller EER means a better performance. We label samples as normal (positive) or abnormal (negative). Accuracy, FAR, and FRR are defined below:

$$\begin{aligned} \text{Accuracy} &= \frac{\text{Number of correctly labeled samples}}{\text{Total number of samples}}, \\ \text{FAR} &= \frac{\text{Number of false positive samples}}{\text{Number of negative samples}}, \\ \text{FRR} &= \frac{\text{Number of false negative samples}}{\text{Number of positive samples}}. \end{aligned}$$

5.2 Evaluation systems

We build a testbed and implement three different trojan programs to evaluate the performance of our method.

5.2.1 Testbed

The testbed is a liquid control system that simulates the real control process in factories. It consists of WinCC monitoring software, a Siemens PLC S7-300, and the controlled objects. The control flow of

our testbed is shown in Fig. 6. The liquid control system, which contains an I/O control and a proportion-integration-differentiation (PID) control, is a typical control system that is widely used in ICSs. Since we have no room for a water tank in our laboratory, we use Simulink in MATLAB to simulate a water tank. The sensor data and control data are communicated between the PLC and MATLAB through an Advantech PCI-1710 data acquisition card. The control device is Siemens S7-300, which sends the data of liquid level, pipeline pressure, and valve status to WinCC, and thus we are able to monitor the status of the water tank. Once there is an abnormal situation, WinCC will raise an alert.

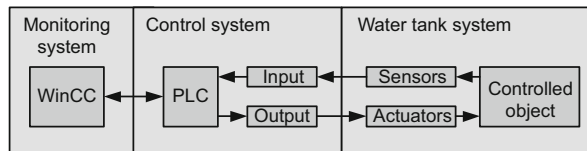


Fig. 6 Control flow of our testbed

5.2.2 Trojan attacks

According to the three aforementioned types of attack, we implement three trojan attacks to validate the efficiency of NIPAD. Since we focus on the destructiveness and concealment of the attack, we assume that the intrusion process is done and trojan programs can be downloaded.

1. Trojan A

A relay is a typical controlled object for a PLC, and it tends to have a limited service life. Trojan A will open and close a relay frequently to reduce its service life. In addition, attackers have the ability to control the PLC and send false information to WinCC, and thus the operators may hardly ever recognize the attack. Stuxnet is able to delay the Iranian nuclear program through this kind of extremely concealed attack. WinCC displays the relay status through reading the data of a register which stores the real relay status. To deceive the WinCC under our attack, we cut off the relation between the register and the status of the relay, and give the register a fake relay status. Furthermore, we trigger only the operation at midnight to enhance its concealment.

2. Trojan B

Another attack is to damage the pipe through over-pressurizing it. WinCC will raise an alert when

the pipe pressure is above its upper limit, which is an important security mechanism in an ICS. To perform the attack secretly, we first change the upper limit of pipe pressure to disable the over-pressure alarm, and turn the pipe valve at the extreme at every midnight. Meanwhile, we send fake valve status to WinCC.

3. Trojan C

One more attack is to change the liquid level of the water tank. Since the target liquid level of the system is P_0 , the upper line of the system is P_0 . We first change the upper line to a larger one P_1 and then set a temporary line P_2 . When the liquid is below P_2 , we do nothing; when the liquid is above P_2 , we change the liquid to a smaller one and save it to the register. Thus, the liquid will keep rising when it equals P_0 , and be maintained at P_1 eventually. Similarly, we send fake information to WinCC. If the controlled object is not water, but a key component of an important product, an inaccurate proportion may cause a disaster.

5.3 Evaluation results

Our experiments are classified into normal scenarios and attack scenarios, which are designed to answer the following questions:

1. If we monitor a PLC for a long time, and the PLC is running the same normal program, is the false alarm rate low enough? If we download a trojan program to it, can we detect the abnormality?
2. What is the detection sensitivity of our method? What is the minimum amount of program difference between the original one and the trojans that can be detected by NIPAD?
3. What affects the anomaly detection algorithm? How do they affect the detection performance?

For both normal scenarios and attack scenarios, we collect 18-h power consumption data of the normal program for training an LSTM network, and segment the signals by a window size of 5 s. We choose the same τ in Eq. (11) as 0.02 to differentiate normal samples and abnormal samples.

5.3.1 Normal scenarios

An effective monitoring system is supposed to have a low false alarm rate, namely a low false rejection rate. It is unacceptable that a monitoring system gives false alarms frequently. When monitoring

the testbed for a long time, will NIPAD give a high false alarm rate? Will the false alarm rate increase with time? To verify the effectiveness of NIPAD, we monitor the PLC in a normal scenario for over 48 h to see whether the system has a low false rejection rate. The PLC runs the same normal program for 48 h and we calculate the false alarm rate every 2 h to see the variation. To demonstrate the effectiveness of LSTM, we also use one-class SVM and a correlation-based algorithm for anomaly detection. The results are shown in Fig. 7.

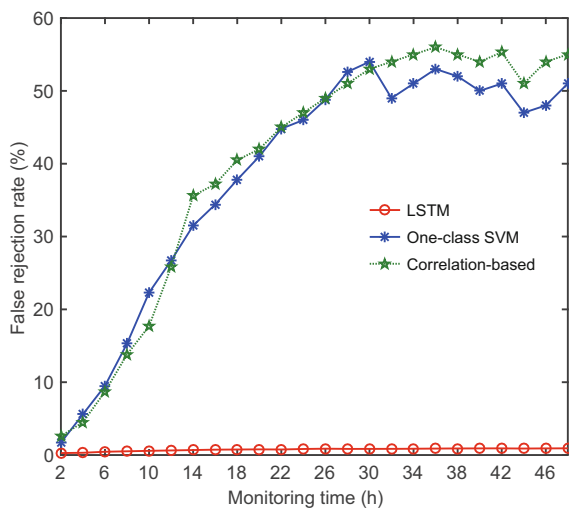


Fig. 7 Performance of continuous monitoring with time. It illustrates the change of the false rejection rate (FRR)

From Fig. 7 we see that for the first 2 h, all three algorithms show good performance, and the FRRs are less than 2.60%. As the monitoring time goes on, the FRR of LSTM increases slightly, but it is always less than 0.95%. For one-class SVM and the correlation-based algorithm, the FRR runs up to 55.58% as time goes by. It indicates that most of the normal samples tend to be classified as abnormal samples after a period of time. The reason that LSTM outperforms both one-class SVM and the correlation-based algorithm is as follows. An attack that modified a few lines of the PLC program is likely to cause small changes from the original normal program in terms of their power consumption. We have to choose a small threshold to distinguish abnormal (attack) samples from the normal ones. However, the power consumption of a PLC is time-varying, and we observed that the base power consumption

of a normal program changes slowly over time. As a result, the difference of the normal power consumption over a few days may exceed the threshold. Both one-class SVM and the correlation-based algorithm are static algorithms for anomaly detection. They do not adapt over time, and tend to misclassify the normal samples as abnormal ones after the system has run for a while. However, LSTM is a dynamic algorithm that has the ability of predicting the next time step, and it is able to learn the time-varying characteristic. Thus, it exhibits promising results.

5.3.2 Attack scenarios

With a low false rejection rate in normal scenarios, an effective monitoring system should have a low false acceptance rate for attack scenarios. To verify the detection performance in attack scenarios, we monitor the PLC for 8 h. The PLC first runs the normal program, followed by trojans *A*, *B*, and *C*. The PLC runs each of the four programs for 2 h. The sizes of the normal program and three trojan programs are shown in Table 1. We calculate the classification result of each sample. Then we count the number of falsely classified samples every 10 min, and thus we have 12 time periods. Since we segment the samples with a window size of 5 s, we have 120 samples in every time period.

Table 1 Sizes of the normal program and three trojan programs

Program	Normal	Trojan <i>A</i>	Trojan <i>B</i>	Trojan <i>C</i>
Size (B)	5860	5897	5944	6078

Fig. 8 shows the results of falsely labeled samples. The FRR is nearly 0.28% for normal situations while the FAR is 0.89% for trojan *A*, 0.28% for trojan *B*, and 0 for trojan *C*. Although there are only four falsely rejected samples, it is still undesirable to have four false alarms within 2 h. For all the abnormal detection situations, we will confirm the attack and raise an alert after three continuous abnormal samples. Thus, we are able to reduce the false alarm rate to nearly 0. In addition, we can confirm the attack within 1 min, which is an acceptable delay for an ICS because the ICS attacks usually take a longer time to cause damage.

For a trojan program, sometimes it is complex to cause damage, but sometimes it is easy to achieve the attack goal, such as trojans *A*, *B*, and *C*. What

is the detection sensitivity of our algorithm? How much of the change can we detect? To find the answer, we modify the liquid control program for our experiment. We delete or modify the original program line by line and consider the modified programs as abnormalities, and the sizes of the modified programs are shown in Table 2. For each modified program, we collect 1-h data for testing. In this set of experiments, we use LSTM, one-class SVM, and the correlation-based algorithm for anomaly detection.

Fig. 9 illustrates the detection sensitivity of different anomaly detection algorithms. As we increase the number of modified lines, the detection performance improves for all three detection algorithms, and they will eventually detect an abnormality with an accuracy above 99.30% when the modification is more than 15 lines. We find that LSTM outperforms one-class SVM and correlation-based anomaly detection in all situations. When we change four lines, i.e., the change of program size reaches 0.44%, LSTM is able to detect an abnormality with an accuracy of above 97.56%. For one-class SVM and the correlation-based algorithm, they may reach the same accuracy when we change 13 lines, which means a program change of 1.57%. In a real factory, the

user program size tends to be less than 10 KB for a medium- or small-scale control system. If attackers plan to destroy the control system secretly, they will usually modify more than one line. For a large control system, the attackers may modify more to achieve a complicated end. As for LSTM, we can detect an abnormality with an accuracy of around 90.33% when we modify only one line of the original program. Thus, we believe our method works for most sophisticated attacks.

5.3.3 Influencing factors on the algorithm

We discuss how window size, training time, and sampling rate affect the detection performance. We try to find the trade-off between window size, training time, sampling rate, and performance.

We first test the relationship between window size and detection performance. We collect the power signal of the normal program with a sampling rate of 250 kSa/s for 18 h for training. Then we collect 1-h normal program and 3-h trojan programs with the same sampling rate for testing. Specifically, we test the three trojan programs mentioned above, each for 1 h. We segment the signals by a window size of 2, 3, 5, 9, or 17 s. When we calculate the frequency-

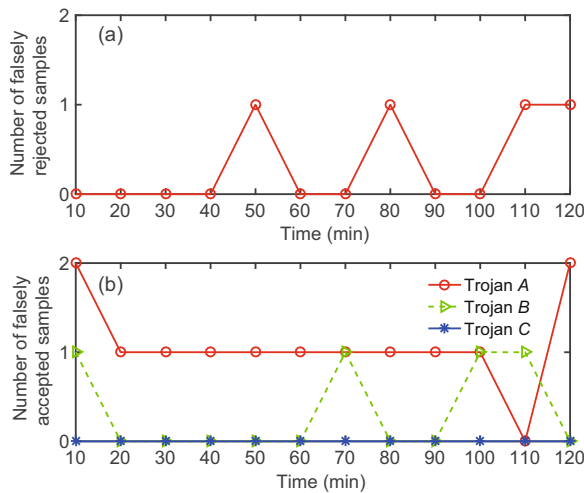


Fig. 8 Detection performance under attack scenarios: (a) falsely rejected samples for the normal program; (b) falsely accepted samples for trojans A, B, and C. There are 120 samples during each time period

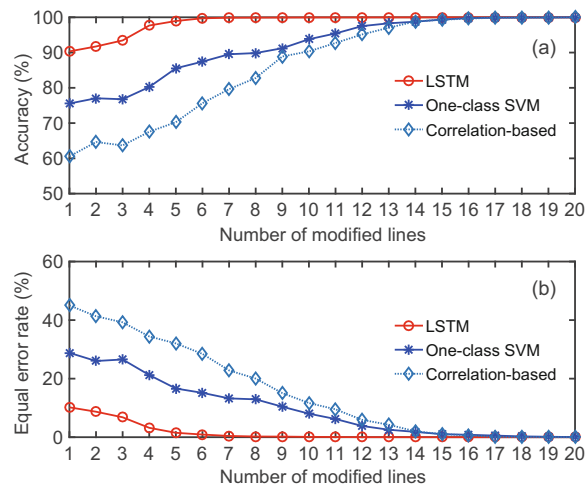


Fig. 9 Detection sensitivity of different anomaly detection algorithms: (a) detection accuracy at different numbers of modified lines; (b) equal error rate (EER) at different numbers of modified lines

Table 2 Sizes of the modified programs*

Modified (lines)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Size (B)	5856	5850	5842	5834	5824	5818	5810	5806	5794	5786	5778	5770	5768	5750	5744	5736	5726	5720	5704	5696

* The size of the original normal program is 5860 B

domain features, the number of points in a sample should be the n th power of two. Thus, we cannot segment the samples with evenly spaced window sizes, so we use the above window sizes. The results are shown in Fig. 10. It illustrates that when the window size is less than 9 s, we are able to achieve an accuracy of 97.97% for trojan A, 98.80% for trojan B, and 99.95% for trojan C.

The performance of LSTM depends greatly on the training time, and a longer training time usually results in a better LSTM model. We collect the power signal of a normal program with a sampling rate of 250 kSa/s for different times for training. Then we collect the power signal of the three trojan programs with the same sampling rate, each for 1 h. We segment all the signals by a window size of 5 s. From Fig. 11, we observe that the detection accuracy grows quickly from 6 h to 18 h, while the rate reduces after 18 h. Generally speaking, the detection accuracy improves with the increase of the training time. If detection performance is the most important factor, we can lengthen the training time to meet the requirement. If the training time is also an important factor, we can slightly reduce the training time and 18 h strikes a balance between training time and detection performance. The detection accuracy achieved is 99.83% for trojan A, 99.97% for trojan B, and 99.99% for trojan C.

To find the relationship between the sampling rate and detection performance, we collect the power signal with a sampling rate of 250, 225, 200, 175, 150, 125, or 100 kSa/s for the normal program and the

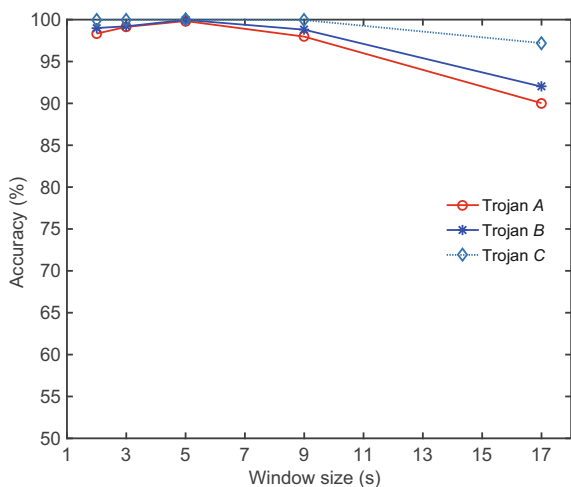


Fig. 10 Detection accuracy under different window sizes. The trojan programs are the three attacks mentioned in the text

three trojan programs. We collect 18-h normal data for training and 3-h trojan data for testing in each situation. We segment all the signals by a window size of 5 s. Fig. 12 shows the results. We can see that a higher sampling rate gives a better detection performance. The detection accuracy of 175 kSa/s is close to that of 250 kSa/s. However, when the sampling rate is less than 175 kSa/s, the detection accuracy drops a lot. Thus, a sampling rate of 175 kSa/s can be chosen.

6 Related work

As early as 2008, Cárdenas et al. (2008) have analyzed the vulnerabilities and threats against control systems, and they also raised some attack models.

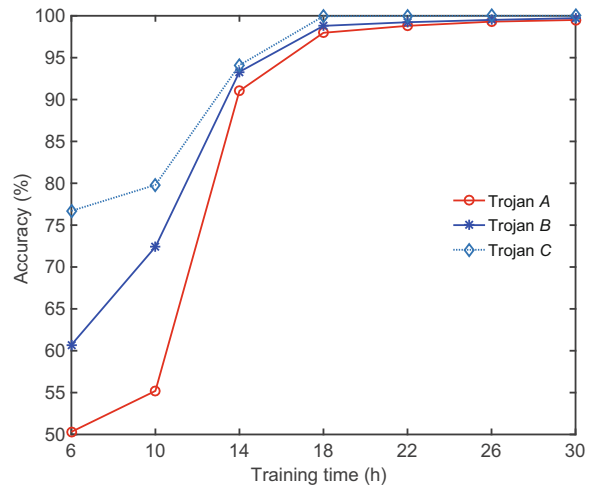


Fig. 11 Detection accuracy with different training times. The trojan programs are the three attacks mentioned in the text

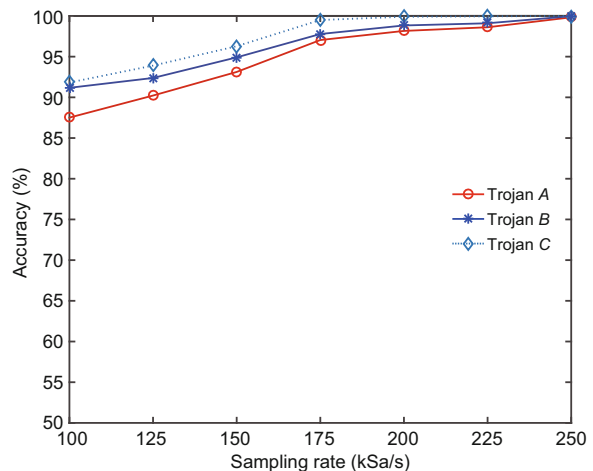


Fig. 12 Detection accuracy under different sampling rates. The trojan programs are the three attacks mentioned in the text

Stuxnet opened the eyes of security researchers and ICS related staff to the fact that an ICS is vulnerable against various attacks. From then on, a number of researchers devoted themselves to ICS security studies and there is much work on the vulnerabilities and threats against ICS (Alcaraz and Zeadally, 2013; Krotofil and Gollmann, 2013; Piggan, 2015). Although these works focus on high-level security requirements and rarely address the implementation of specific protection measures, they have guiding significance for augmenting ICS security.

One of the main sources of ICS vulnerability is the lack of security mechanisms. As a result, some efforts focus on high priority protection areas such as security management, secure network architectures, and self-healing (Alcaraz and Zeadally, 2015). However, these approaches rely on redesigning or replacing parts of the system, which is infeasible because most ICSs of critical infrastructures cannot afford a shutdown. Besides, the redesign of an ICS structure could be prohibitively costly. As a result, countermeasures based on changing the schematic of existing ICSs are impractical.

The signature-based intrusion detection system (IDS) which learns from the IT system is a widely researched security mechanism for an ICS. Snort rules can detect and prevent intrusions including denial-of-service, command injection, response injection, and system reconnaissance (Morris *et al.*, 2012). However, more and more cyber attacks against ICSs are a combination of different vulnerabilities and technologies, which renders signature-based IDS with limited success.

Another similar approach is the anomaly-based network intrusion detection system. Its core concept is to distinguish abnormal behaviors from the normal ones (García-Teodoro *et al.*, 2009). Peng *et al.* (2015) suggested a fingerprinting methodology which can be used to differentiate abnormal behaviors from normal through analyzing industrial control protocols and extracting their characteristics. There is much work on anomaly-based intrusion detection for ICSs, including statistical, machine learning, and knowledge-based techniques (Mantere *et al.*, 2012; Coletta and Armando, 2015; Ponomarev and Atkison, 2016; Shang *et al.*, 2016). Although these methods work well and have low false positive rates, they are not suited for ICSs since they consume network resources and may affect the effectiveness of the

ICS.

Considering the unique characteristics of an ICS, side-channel based anomaly detection methods have been proposed. Time fingerprinting for detecting malicious controlled objects like relays was presented to augment the security of an ICS (Formby *et al.*, 2016). Nevertheless, it is effective only for some attacks against those controlled devices. Power fingerprinting was proposed to detect malicious software execution in PLCs (Gonzalez and Hinton, 2014). However, it collects power fingerprinting with a near-field sensor and the CPU of the PLC is exposed to the sensor, which is not feasible for real-time monitoring. The EMI-based anomaly detection method (Stone *et al.*, 2015) has the same issues. In addition, both methods use a high-frequency data acquisition unit whose sampling rate is up to several million samples per second. There is preliminary work on real-time monitoring of PLCs based on power consumption (Clark *et al.*, 2013). However, it achieves only 94.20% accuracy for known attacks and 84.90% accuracy for unknown attacks.

Inspired by the above work, we have presented our monitoring system which aims at overcoming those limitations and achieving real-time monitoring of PLC attacks. Since the PLC is the core component in an ICS, detecting malicious programs in the PLC is a necessary method to augment ICS security.

7 Conclusions

In this study, we have proposed a non-invasive power-based anomaly detection scheme for detecting attacks on PLCs. We were able to implement a real-time monitoring system for anomaly detection with a real-time data acquisition module. We have detected the attacks which we implemented with an accuracy as high as 99.83% in our lab experiments. We have also discussed the detection sensitivity of our method. Even when the modification of the original program is as little as 0.07%, we are able to detect the change with an accuracy of 90.33%. We demonstrated that our scheme is able to detect an anomaly in a PLC and thus augmented the security of the ICS.

We focused on detecting the logic modification of PLC programs. As a direction for future work, it is promising to detect other types of malicious modification of PLC programs, e.g., attacks that modify

only the parameters. In addition, other types of side-channel signals, such as EMI, acoustic, and thermal signals, may be an important supplement to our power-based method. We relied on a resistor to collect power consumption traces, and it is worth examining the feasibility of collecting power consumption data without inserting a resistor and thus letting the monitoring system be completely isolated from the original ICS.

References

- Alcaraz, C., Zeadally, S., 2013. Critical control system protection in the 21st century. *Computer*, **46**(10):74-83. <http://dx.doi.org/10.1109/MC.2013.69>
- Alcaraz, C., Zeadally, S., 2015. Critical infrastructure protection: requirements and challenges for the 21st century. *Int. J. Crit. Infrastr. Protect.*, **8**:53-66. <http://dx.doi.org/10.1016/j.ijcip.2014.12.002>
- Bencsáth, B., Pék, G., Buttyán, L., et al., 2012. The cousins of Stuxnet: Duqu, Flame, and Gauss. *Fut. Int.*, **4**(4):971-1003. <http://dx.doi.org/10.3390/fi4040971>
- Bolton, W., 2015. Programmable Logic Controllers (6th Ed.). Newnes, USA.
- Bullock, J., Conservatoire, U.C.E.B., 2007. LibXtract: a lightweight library for audio feature extraction. Proc. Int. Computer Music Conf., p.1-4.
- Candes, E.J., Tao, T., 2006. Near-optimal signal recovery from random projections: universal encoding strategies? *IEEE Trans. Inform. Theory*, **52**(12):5406-5425. <http://dx.doi.org/10.1109/TIT.2006.885507>
- Cárdenas, A.A., Amin, S., Sastry, S., 2008. Research challenges for the security of control systems. Proc. 3rd Conf. on Hot Topics in Security, Article 6.
- Chen, T.M., Abu-Nimeh, S., 2011. Lessons from Stuxnet. *Computer*, **44**(4):91-93. <http://dx.doi.org/10.1109/MC.2011.115>
- Clark, S.S., Ransford, B., Rahmati, A., et al., 2013. WattsUpDoc: power side channels to nonintrusively discover untargeted malware on embedded medical devices. Proc. USENIX Workshop on Health Information Technologies, p.1-11.
- Coletta, A., Armando, A., 2015. Security monitoring for industrial control systems. Proc. Conf. on Cybersecurity of Industrial Control Systems, p.48-62. http://dx.doi.org/10.1007/978-3-319-40385-4_4
- Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, p.886-893. <http://dx.doi.org/10.1109/CVPR.2005.177>
- Formby, D., Srinivasan, P., Leonard, A., et al., 2016. Who's in control of your control system? Device fingerprinting for cyber-physical systems. Proc. Network and Distributed System Security Symp., p.1-13.
- García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., et al., 2009. Anomaly-based network intrusion detection: techniques, systems and challenges. *Comput. Secur.*, **28**(1-2):18-28. <http://dx.doi.org/10.1016/j.cose.2008.08.003>
- Gers, F.A., Schmidhuber, J.A., Cummins, F., 2000. Learning to forget: continual prediction with LSTM. *Neur. Comput.*, **12**(10):2451-2471. <http://dx.doi.org/10.1162/089976600300015015>
- Gonzalez, C.A., Hinton, A., 2014. Detecting malicious software execution in programmable logic controllers using power fingerprinting. Proc. Int. Conf. on Critical Infrastructure Protection, p.15-27. http://dx.doi.org/10.1007/978-3-662-45355-1_2
- Johnson, R.E., 2010. Survey of SCADA security challenges and potential attack vectors. Proc. Int. Conf. for Internet Technology and Secured Transactions, p.1-5.
- Kesler, B., 2011. The vulnerability of nuclear facilities to cyber attack. *Strat. Insights*, **10**(1):15-25.
- Krotofil, M., Gollmann, D., 2013. Industrial control systems security: what is happening? Proc. 11th IEEE Int. Conf. on Industrial Informatics, p.670-675. <http://dx.doi.org/10.1109/INDIN.2013.6622964>
- Langner, R., 2011. Stuxnet: dissecting a cyberwarfare weapon. *IEEE Secur. Priv.* **9**(3):49-51. <http://dx.doi.org/10.1109/MSP.2011.67>
- Lee, H., Battle, A., Raina, R., et al., 2006. Efficient sparse coding algorithms. Proc. 19th Int. Conf. on Neural Information Processing Systems, p.801-808.
- Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, **60**(2):91-110. <http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94>
- Macaulay, T., Singer, B.L., 2011. Cybersecurity for Industrial Control Systems: SCADA, DCS, PLC, HMI, and SIS. CRC Press, USA.
- Malhotra, P., Vig, L., Shroff, G., et al., 2015. Long short term memory networks for anomaly detection in time series. Proc. European Symp. on Artificial Neural Networks, Computational Intelligence and Machine Learning, p.89-94.
- Manevitz, L.M., Yousef, M., 2002. One-class SVMs for document classification. *J. Mach. Learn. Res.*, **2**:139-154.
- Mantere, M., Uusitalo, I., Sallio, M., et al., 2012. Challenges of machine learning based monitoring for industrial control system networks. Proc. 26th Int. Conf. on Advanced Information Networking and Applications Workshops, p.968-972. <http://dx.doi.org/10.1109/WAINA.2012.135>
- Morris, T., Vaughn, R., Dandass, Y., 2012. A retrofit network intrusion detection system for MODBUS RTU and ASCII industrial control systems. Proc. 45th Hawaii Int. Conf. on System Science, p.2338-2345. <http://dx.doi.org/10.1109/HICSS.2012.78>
- Nandakumar, K., Jain, A.K., 2004. Local correlation-based fingerprint matching. Proc. ICVGIP, p.503-508.
- Ni, B., Moulin, P., Yang, X., et al., 2015. Motion part regularization: improving action recognition via trajectory group selection. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, p.3698-3706. <http://dx.doi.org/10.1109/CVPR.2015.7298993>
- Pearson, K., 1901. Mathematical contributions to the theory of evolution. X. Supplement to a memoir on skew variation. *Phil. Trans. R. Soc. A*, **197**:443-459.
- Peng, Y., Xiang, C., Gao, H., et al., 2015. Industrial control system fingerprinting and anomaly detection. Proc. Int. Conf. on Critical Infrastructure Protection, p.73-85. http://dx.doi.org/10.1007/978-3-319-26567-4_5

- Piggin, R., 2015. Are industrial control systems ready for the cloud? *Int. J. Crit. Infrastr. Protect.*, **9**(C):38-40. <http://dx.doi.org/10.1016/j.ijcip.2014.12.005>
- Ponomarev, S., Atkison, T., 2016. Industrial control system network intrusion detection by telemetry analysis. *IEEE Trans. Depend. Sec. Comput.*, **13**(2):252-260. <http://dx.doi.org/10.1109/TDSC.2015.2443793>
- Pretorius, B., van Niekerk, B., 2016. Cyber-security for ICS/SCADA: a South African perspective. *Int. J. Cyber Warf. Terror.*, **6**(3):1-16. <http://dx.doi.org/10.4018/IJCWT.2016070101>
- Shang, W., Zeng, P., Wan, M., et al., 2016. Intrusion detection algorithm based on OCSVM in industrial control system. *Secur. Commun. Netw.*, **9**(10):1040-1049. <http://dx.doi.org/10.1002/sec.1398>
- Slay, J., Miller, M., 2007. Lessons learned from the Maroochy water breach. Proc. Int. Conf. on Critical Infrastructure Protection, p.73-82. http://dx.doi.org/10.1007/978-0-387-75462-8_6
- Stone, S.J., Temple, M.A., Baldwin, R.O., 2015. Detecting anomalous programmable logic controller behavior using RF-based Hilbert transform features and a correlation-based verification process. *Int. J. Crit. Infrastr. Protect.*, **9**(C):41-51. <http://dx.doi.org/10.1016/j.ijcip.2015.02.001>
- Stouffer, K.A., Falco, J.A., Scarfone, K.A., 2011. Guide to Industrial Control Systems (ICS) Security: Supervisory Control and Data Acquisition (SCADA) Systems, Distributed Control Systems (DCS), and Other Control System Configurations such as Programmable Logic Controllers (PLC). Technical Report SP 800-82, National Institute of Standards and Technology, USA.
- Wang, H., Kläser, A., Schmid, C., et al., 2013. Dense trajectories and motion boundary descriptors for action recognition. *Int. J. Comput. Vis.*, **103**(1):60-79. <http://dx.doi.org/10.1007/s11263-012-0594-8>
- Xu, J., Yang, G., Man, H., et al., 2013. L_1 graph based on sparse coding for feature selection. Proc. Int. Symp. on Neural Networks, p.594-601. http://dx.doi.org/10.1007/978-3-642-39065-4_71
- Zhong, W., Lu, H., Yang, M., 2012. Robust object tracking via sparsity-based collaborative model. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, p.1838-1845. <http://dx.doi.org/10.1109/CVPR.2012.6247882>