Adversarial Attacks for Testing Anomaly Detectors of Cyber-Physical Systems

Anonymous Author(s)

ABSTRACT

The constant threat of attack faced by cyber-physical systems (CPSs) in critical infrastructure has motivated research into a multitude of attack detection mechanisms, such as *anomaly detectors* based on underlying neural networks. The effectiveness of such a detector can be assessed by subjecting it to test suites of attacks, whether from benchmarks, or tools such as network fuzzers. While studies have shown that neural network-based detectors perform well against these conventional attacks, limited consideration has been given to testing their effectiveness against *adversarial attackers* that possess knowledge of the underlying model. In this paper, we present an adversarial attack for testing recurrent neural network (RNN) anomaly detectors of CPSs. Inspired by some white-box gradient-based approaches, our adversarial attacks produce *noise* (or *perturbations*) with the aim of causing an incorrect classification. We implement it against detectors for two real-world critical infrastructure testbeds, successfully reducing their classification accuracy. Finally, we demonstrate that the detectors of the testbeds can be successfully adapted to detect attacks that utilise such noise.

KEYWORDS

Cyber-physical systems, testing attack detection mechanisms, anomaly detectors, adversarial attacks, neural networks.

ACM Reference Format:
Anonymous Author(s). 2020. Adversarial Attacks for Testing Anomaly Detectors of Cyber-Physical Systems. In *Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2020).* ACM, New York, NY, USA, 12 pages.

1 INTRODUCTION

Cyber-physical systems (CPSs), in which software components are deeply intertwined with physical processes, are ubiquitous in critical public infrastructure. The potential disruption that could result from a compromised system has motivated research into a multitude of CPS attack detection mechanisms, including techniques based on invariant checking [40], physical attestation [17], and fingerprinting [25]. A particularly popular solution is to build *anomaly detectors* [15, 20, 26, 37, 61], in which an underlying machine learning (ML) model is trained on streams of the system's physical data in order to judge when it deviates from the norm. Typically, this would be a form of neural network, a model that is powerful enough to learn and recognise the complex patterns of behaviour that CPSs exhibit.

The effectiveness of an anomaly detector can be assessed by subjecting it to a test suite of attacks, and observing whether it can correctly identify the anomalous behaviour. These tests can be derived from benchmarks [39], hackathons [3], or tools such as fuzzers [19], and typically involve manipulating or spoofing the network packets exchanged between CPS components. While studies have shown that neural network-based detectors perform well against these conventional attacks [35, 39, 41, 55], limited consideration has been given to testing their effectiveness against *adversarial attacks*, in which an attacker with knowledge of the model crafts *noise* (or *perturbations*[1]) on the data to specifically cause the neural network to misclassify it. Such attacks have been successfully demonstrated in several different classification domains (e.g. images [44], audio [14], malware [31]), raising the question as to whether anomaly detectors in critical infrastructure are susceptible too. If an anomaly detector fails to detect adversarial tests, then its CPS is potentially at risk of a much broader range of attacks since they can simply be masked by crafted noise.

In this paper, we present an adversarial attack for testing recurrent neural network (RNN) anomaly detectors of CPSs. Our approach is inspired by the white-box gradient-based approach previously applied to deceive the classifiers of audio files [14], but adapted to meet the unique challenges posed by the critical infrastructure. First, while we can craft noise on sensors in the continuous domain, actuators typically fall into a small number of discrete states (e.g. 'open' or 'closed'). Second, in addition to anomaly detectors, the logic of these CPSs often also includes some built-in mechanisms for *rule-checking* (e.g. testing that a tank's valve is open if its level is low) to add another level of certainty that the system is operationally normal. We address these issues by using genetic algorithm to select optimized combination of actuator values.

We evaluate the effectiveness of our adversarial attacks by implementing them against RNN-based anomaly detectors for two real-world critical infrastructure testbeds: Secure Water Treatment (SWaT) [48], a multi-stage water purification plant; and Water Distribution (WADI) [7], a consumer distribution network. We demonstrate that our adversarial tests successfully and substantially reduce the classification accuracy of their anomaly detectors, thus revealing a security flaw that could be exploited. We show that a rule-checking mechanism in SWaT has weakened our threat model, but with the application of a genetic algorithm, we can still achieve a similar accuracy reduction for the anomaly detectors. Finally, we go a step further and show that by training the RNNs on some noisy data, the anomaly detectors of SWaT and WADI can successfully identify that an adversarial attack is in progress.

Our Contributions. Our contributions are summarised as follows:

[1] We will use noise and perturbation interchangeably.

ISSTA 2020, 18–22 July, 2020, Los Angeles, US

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

- We define a threat model for CPS adversarial attackers, and propose a white-box gradient-based attack to construct adversarial tests for deceiving RNN-based anomaly detectors.
- We re-implement SWaT's RNN-based anomaly detector [26], extending it to cover the whole CPS, and learning another in parallel to cover WADI.
- We implement our adversarial attacker, and test it against the SWaT and WADI RNN-based anomaly detectors.
- We find that our approach significantly reduces the accuracy of the detectors to the point that masking other attacks is feasible.

Organisation. In Section 2, we provide the necessary background and present our threat model. In Section 3, we introduce our experiment design and methodologies to construct adversarial examples. We show our empirical study results on SWaT and WADI in Section 4. Finally, we review related works in Section 5 and conclude in Section 6.

2 BACKGROUND

In this section, we state our assumptions about the structure of CPSs, and introduce two real-wold examples that our work will be evaluated with. Following this, we define the threat model that our tests will be based on, as well as what constitutes a conventional and adversarial attack. Finally, we provide some background about the anomaly detectors we are testing, as well as the rule-checking mechanisms that may guard them.

2.1 Cyber-Physical Systems

In general, we assume that CPSs consist of two interconnected parts. First, a 'physical' part, in which various physical processes are monitored by sensors and acted upon by actuators. Second, a 'cyber' part, consisting of software components such as Programmable Logic Controllers (PLCs) [10] that implement some control logic. We assume that sensors, actuators, and PLCs are connected over a network, with PLCs taking sensor readings as input and computing commands to be returned to the actuators. Furthermore, we assume the presence of a Supervisory Control and Data Acquisition (SCADA) system built on top of the PLCs, that can supervise the control process and issue commands of its own.

We assume that CPSs contain two different types of countermeasures against attacks. First, the PLCs may contain some rule-checking mechanism [9], for checking that the sensor and actuator states pertain to some invariant properties (e.g. if a certain sensor falls below a certain range, then a certain actuator should be disabled). Such rule checking mechanism is a standard for industrial CPSs. Second, we assume that a neural network-based anomaly detector [15] is present in the SCADA.

SWaT and WADI Testbeds. We carry out our studies on two modern, real-world critical infrastructure testbeds: Secure Water Treatment (SWaT) [48], a water purification plant capable of producing five gallons of safe drinking water per minute; and Water Distribution (WADI) [7], a consumer supply distribution system. SWaT is a six-stage plant including processes such as ultra-filtration, de-chlorination, and reverse osmosis. WADI consists of three stages, across which water is moved into elevated reservoirs, then supplied to consumer tanks on a pre-set pattern of demand.

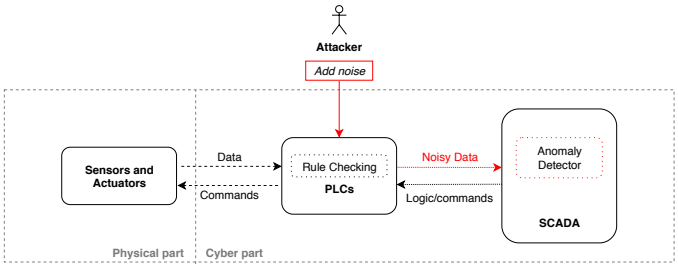

Figure 1: Overview of a cyber-physical system and an adversarial attacker

Sensors and actuators. In total, SWaT contains 68 sensors and actuators. Its sensors variously read the level of tanks, pressure, and flow across the system, whereas its actuators include motorised valves and pumps. Sensor readings are typically continuous values, whereas actuators are typically discrete (e.g.'open' or 'closed'; 'on' or 'off'). Note that a number of SWaT's actuators are 'standbys' that are intended to be used only when the primary actuator fails. These are not considered in our work. After filtering such actuators, there are a total of 25 sensors and 26 actuators which are targeted in our study. WADI contains 70 sensors and 51 actuators, of a similar kind to those seen in SWaT.

The physical state of a testbed is a fixed ordering of all the sensor readings and actuator configurations at a particular timepoint. Table 1 illustrates four such states from the first stage of SWaT (handling supply and storage). There are two sensors covering water flow (FIT101) and the tank level (LIT101), as well as a motorised valve (MV101) controlling at the inflow pipe of the tank, and a primary pump (P101) and secondary pump (P102) for pumping water out of the tank. The logged data over four seconds indicates that water is flowing into the tank and thus the level is increasing.

Formally, we will use x to denote a system state, consisting of a fixed order of actuators and sensors:

$$x = [x_{a1}, x_{a2}, x_{a3} \ldots x_{s1}, x_{s2}, x_{s3} \ldots]$$

Here, each x_a represents an actuator value and each x_s represents a sensor value. X is the set of all possible system states. Let S be a sequence of system states. We use $S[i]$ to denote the i-th state in the sequence, $S[i:j]$ to denote the sequence of system states from time i to j, and $S[i:]$ to denote the sequence of states from time i to the present.

Communication and control. SCADA workstations are located in the plants' control rooms. A Human Machine Interface (HMI) is also located inside each plant and can be used to view the process states and set parameters. Control code can be loaded into the PLCs via the workstations. We can also acquire data logs (as well as network packet flows) from the historian at preset time intervals.

A multi-layer network enables communication across all components of SWaT and WADI. A ring network at Layer 0 of each stage enables the responsible PLC to communicate with sensors and actuators, whereas a star network at Layer 1 of the network

Table 1: Example of log data from the SWaT historian

Timestamp	FIT101	LIT101	MV101	P101	P102	Normal/Attack
28/12/2015 10:00:00 AM	2.427	522.846	2	2	1	Normal
28/12/2015 10:00:01 AM	2.446	522.886	2	2	1	Normal
28/12/2015 10:00:02 AM	2.489	522.846	2	2	1	Normal
28/12/2015 10:00:03 AM	2.534	522.964	2	2	1	Normal

enables communication between the PLCs. Supervisory systems such as the workstation and historian sit at levels further above.

2.2 Threat Model for Testing

Given that we are testing the security of CPSs, we define a threat model that states our assumptions of what an attacker knows about the system and is capable of doing within it.

As depicted in Figure 1, our threat models assumes a white-box setting where an attacker has the following capabilities:

(1) the attacker is able to compromise the data transmitted from the physical part of the CPS to the PLCs at Layer 0 of the network;
(2) the attacker has full access to the RNN-based anomaly detector, including the network architecture, parameters, inputs, outputs, and other attributes;
(3) the attacker is aware of the presence of a rule-checking mechanism, but cannot access any information about it other than inputs and outputs.

Note that capability (1) is often the preliminary condition to deploy a variety of attacks [39]. Besides, this threat model can be adapted to black-box setting or gray-box setting depending on the knowledge an attacker can acquire of the anomaly detector.

2.3 Attacks and Detection Mechanisms

CPS Attacks. With the capabilities given by the threat model, *conventional* CPS attacks are carried out by spoofing the sensor values that are transmitted from the physical part of the system to the PLCs, causing the control logic to issue the wrong actuator commands. For example, if a tank is near-empty, but an attacker spoofs a tank level sensor that is critically high, the pump could incorrectly be activated and lead to some underflow damage. Other examples are given in Table 2. The SWaT and WADI testbeds are equipped with benchmarks [39] containing multiple different attacks of this kind, which have been used to test the effectiveness of different countermeasures [35, 39, 41, 55]. Furthermore, data sets [39] are available containing several days of physical data resulting from subjecting the testbeds to these attacks. This data is suitable for training complex ML models such as RNNs and other kinds of neural networks.

In this paper, we aim to test CPSs against more than just these conventional attacks, by expanding the repertoire of attackers to include *adversarial attacks*. Using their knowledge of underlying RNN of the anomaly detector, adversarial attackers focus on crafting *adversarial examples* that maximise some measure of harm, while masking their true effects from detectors by using carefully applied noise that deceives them.

To judge the success of a conventional attack, we can check whether at a certain time point t there is an *observable impact* in the physical state, $S[t]$, i.e. the physical state differs from what it would have been in normal operational conditions. As this is not simple to conclude in general, instead, we leverage the operator-specified acceptable ranges of sensor values to identify that the system has been successfully attacked. For example, in SWaT, LIT101 indicates the water level in the first stage. If the reading is above 1100mm, then while it might not yet have overflowed, it is outside its acceptable range, and thus we conclude that the system is under attack.

To judge whether an adversarial attack is successful is somewhat more complicated. Essentially, the goal is to deceive the anomaly detector and cause it to give an incorrect classification. For example, if the system is behaving normally and is being classified as such, the goal is to apply a minimal amount of noise such that the net behaviour of the system does not change, but the anomaly detector classifies it as anomalous (i.e. a false alarm, decreasing confidence in the detector). On the other hand, if the attacker is spoofing sensor values and causing the behaviour to be classified as anomalous, the goal is to craft noise that does not affect the net behaviour (i.e. it remains anomalous in the same way) but causes the detector to classify it as normal. However, how to decide whether it remains anomalous is more complicated in CPS. As a result, for the purposes of experimentation, it is important to be able to conclude that physical effects on the system *before* the noise is applied are the same *after* it is applied too.

As the definition of attacks is dynamic with respect to changes of input data, for every prediction of f, we have to generate status labels as the ground truth. Figure 2 presents an overview of how results are generated. Vertically, Data S is the original attack data, Data S' is noisy data after a gradient-based attack by model N, and S" is selected data by Genetic Algorithm (GA) to pass rule checker R. Horizontally, from each data set (S,S' and S"), we calculate Y_T as the ground truth to compare with the anomaly detector results Y_C and get outputs. The *Output* inside figure refers to a lists of standards such as precision, recall and f1. To illustrate this concretely, consider the original attack data S of Table 3, and the corresponding data S' of Table 4 that results from the application of noise. In the third row of S, a conventional attack is taking place (note the ground truth Y_T), and the anomaly detector correctly classifies this data as an attack (see Y_S). In S', however, some noise has been applied to deceive the detector: the data is classified as normal (Y_S), but the net physical effect on the system remains the same (Y_T). Intuitively, the original conventional attack is still taking place but has been masked by the noise. Note that if the level of noise is too much, or the wrong actuator is manipulated, it is possible to change the

Table 2: Example of modifications

Start Time	End Time	Attack Point	Start State	Modification
10:29:14	10:44:53	MV-101	MV-101 is closed	Open MV-101
10:51:08	10:58:30	P-102	P-101 is on where P-102 is off	Turn on P-102
11:22:00	11:28:22	LIT-101	Water level between L and H	Increase by 1 mm/s

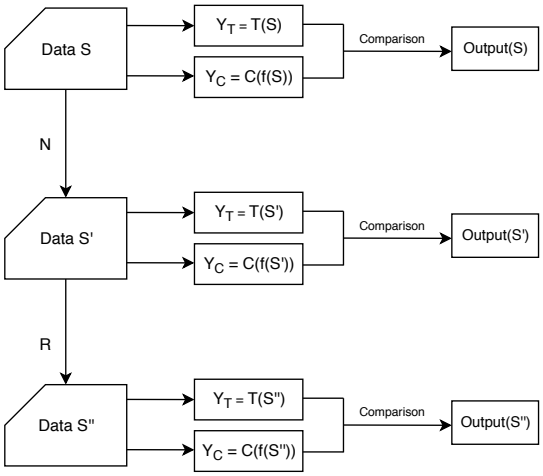

Figure 2: An overview of input vs output

Table 3: Example of original attack data S

FIT101	LIT101	MV101	P101	P102	Y_T	Y_C
2.427	522.846	2	2	1	Normal	Normal
2.437	522.846	2	2	1	Normal	Normal
2.527	522.846	2	2	1	Attack	Attack

Table 4: Example of data S' obtained by applying noise to S

FIT101	LIT101	MV101	P101	P102	Y_T	Y_C
2.427	522.846	2	2	1	Normal	Normal
2.427	522.846	2	2	1	Normal	Normal
2.527	*527.846*	2	2	1	Attack	Normal

ground truth itself, i.e. because the net physical effect of the attack has changed.

Attack Detection Mechanisms. Finally, we describe in more detail two different kinds of attack detection mechanisms, both in the context of SWaT: a rule checker [2], and an RNN-CUSUM anomaly detector [26].

Rule checking. Adepu and Mathur [2] have systematically derived a set of invariants, consisting of 23 rules that describe the relationship between sensor and actuator values of SWaT. The idea of rule checking is to implement a set of pre-defined rules in the PLCs that should never be violated. For instance, a sensor value should never exceed its operation range, or regulations that the system should follow under normal operation. We use R to denote the set of rules implemented in PLCs for checking. Once a rule

Table 5: Examples of rules

Condition	Rule	Time
LIT101 $\leq$ 500	MV101 = 2	12
LIT101 $\leq$ 250	P101 = 1 AND P102 = 1	2
LIT301 $\leq$ 800	P101 = 2 OR P102 = 2	12
AIT201 > 260 & FIT201 > 0.5	P201 = 1 AND P202 = 1	2

$r \in R$ is violated, the system will raise an anomaly alarm to report that the system is in an abnormal state. Some rules are shown in Table 5. Let us take rule 1 as an example. The rule specifies that under the condition that sensor "LIT101" is equal or smaller than 500, the actuator "MV101" is supposed to be 2 after 12 seconds.

RNN-based anomaly detectors. The idea of learning-based anomaly detectors is to model the normal behaviour of the system from a piece of data historian S using a machine learning model (denoted by f). At run time, the system looks at the historian data (of a window-size length), use f to predict the system state x' and compare it with the actual system state x. If the difference is beyond a threshold, the system is likely to be abnormal. In the work from Goh et al. [26], a RNN with LSTM architecture [34] is used as the prediction model and a CUSUM algorithm (denoted by C) is used to calculate the differences between the actual value and the predicted value. This approach has been applied to the first stage of SWaT, and is shown to be effective for it. In our work, we re-implement their approach but for all six stages of SWaT, and all three stages as WADI, as the defence mechanism for us to test.

3 EXPERIMENT DESIGN AND METHODOLOGY

Under the mentioned threat model and attack scenarios in section 2.2, we aim to answer the following questions: 1) Is our adversarial attacks effective on real-world CPS? 2) how should the attacker compromise the original data to deceive the anomaly detector and the rule checking system? 3) how can we design a defence against such adversarial examples? Experiments are designed to answer the research questions. To descent the performance of the anomaly detector of CPSs, there are some difficulties to complete the testing experiments.

- There are limited works on adversarial attacks on CPS, especially for anomaly detector with RNN, and thus we do not have many references.
- There could be more than one anomaly detection system inside the CPS, to complete an adversarial attack, we need to consider all defences.
- A CPS system is normally complicated and composed with sensors and actuators, and for different physical components, the data type and range are different.

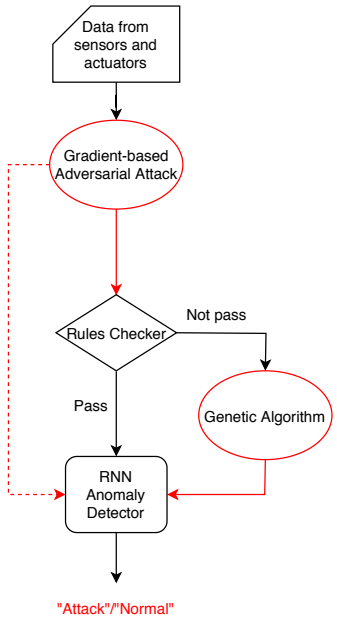

Figure 3: Workflow for adversarial attacks.

- Unlike statistical attacks, CPS is dynamic and hard to predict, a minor change may lead to the whole system break down.

For a basic set up of CPSs, data from sensors and actuators go directly to RNN predictor for attack detection, we apply a threat model with gradient-based adversarial attack to deceive the anomaly detector directly. While for many CPSs, a rule checking system as we described in 2.3 is implemented to check if data obey the rule (difficulty 2). For those systems, if the adversarial attack is able to deceive RNN predictor but detected by the rule checking system, we use generative algorithm to bypass the rule checker. Figure 3 provides an overview of the thread model and algorithm 1 shows details on how to generate the adversarial examples.

Algorithm 1: Overall Algorithm

Input: Data S_i; RNN predictor model f; rule Checker R; noise level δ

Output: adversarial examples

1 Get gradient direction from f of loss r.p.t inputs data S_i;
2 Apply the noise δ to S_i along or opposite the sign of gradients to get S_i';
3 Use Genetic Algorithm to generate S_i'' to pass rule checker;
4 **return** S_i''

3.1 Dataset

Our study is based on the SWaT dataset [39] and WADI dataset [7], which are publicly available datasets [38] and has been used in multiple projects [26, 60]. The SWaT dataset records the system state of 26 sensor values and 25 actuators (in total 51 features) every second and WADI records 70 sensors and 51 actuators (in total 121 features). The sensor values are integer or floating-point numbers while the actuator values are discrete, e.g., 0 (opening/closing), 1 (closed) or 2 (open). The datasets consist of two types of data:

- *Normal data*: The normal dataset S_n of SWaT is collected for 7-days (a total of 496800 records) when the system is under normal operation, and the dataset of WADI is collected for 14-days (a total of 172800 records). The data is used to train the LSTM-RNN predictor.
- *Attack data*: The attack dataset S_a is collected for 4-days (a total of 449909 records) consisting of 36 attacks with labels (normal or attack), and the dataset of WADI is collected for 2-days (a total of 172801 records). The data is used as testing data for the anomaly detector.

3.2 RNN predictor

Following [26], we train a RNN in LSTM architecture from normal data S_n as the prediction model. The trained model is a many-to-one prediction model $f : S \rightarrow X$ which takes a certain sequence (parameterized by window size) of data historian as input and makes the prediction of the output of the coming timestamp. For example, a window size of 10 represents a sequential input of the past 10 timestamps.

Once we obtain the predicted output at each timestamp, we then adopt the CUSUM algorithm [12] to decide whether the system is in abnormal state as follows. The difference d at timestamp i (denoted as $d[i]$) is calculated from the predicted value $x'[i]$ by the RNN model and the actual value $x[i]$, the difference d will then be added cumulatively with an allowable slack c. We calculate the CUSUM for each sensor with both positive and negative value by the following formula:

$$\begin{aligned} SH[i] &= max(0, SH[i-1] + d[i] - c) \\ SL[i] &= min(0, SL[i-1] + d[i] + c) \\ d[i] &= X'[i] - x[i], \end{aligned} \tag{1}$$

where SH represents the set of high cumulative sum and SL represents the low cumulative sum, d is the difference between the predicted value x and actual value x', and c is the allowable slack which is defined as 0.05 multiplied by the standard deviation of S. Besides, two thresholds, i.e., a Upper Control Limit (UCL) and a Lower Control Limit (LCL) for SH and SL to compare with respectively are required to check whether the system is in abnormal state. Normally, UCL and LCL are defined according to the experiment validation of the training data. Table 6 shows the UCL and LCL from stage 1 as an example.

Table 6: Upper and Lower control limit

Senor	Upper Control Limit	Lower Control Limit
FIT101	2	-1.5
LIT101	50	-2

The threshold selection for CUSUM is according to the validation of the attack data. We compared the CUSUM value for each sensor with attack labels, and set the threshold as the lowest peak of the CUSUM value that indicates an attack. Fig 4 presents a CUSUM value example for sensor "LIT101", the top graph is the negative

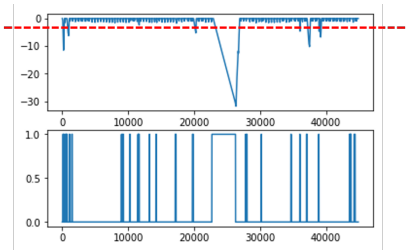

Figure 4: A CUSUM value example for sensor LIT101

CUSUM value SL of "LIT101". The bottom picture is the actual label of data, 1 means under attack and 0 means normal status. By comparing the two graphs, almost every peak value from the SL indicated an attack, we choose the lowest peak value as the threshold and it is indicated by the red dot line with value of −2. We calculate all thresholds accordingly.

3.3 Adversarial Attacks

Our next step is to construct adversarial examples which aim to deceive the anomaly detectors. We consider two cases. Firstly, we construct adversarial examples only to deceive the RNN predictor. Secondly, we take a step further to deceive both the RNN predictor and the rule checking system, which is more realistic and challenging. In both cases, we assume a white-box attack that the attacker could access the trained RNN model (and its parameters) and the attacker is able to compromise the data transmitted to the PLCs. The goal of the attacker is thus finding a minimum perturbation δ on the original input $\mathbf{x}$ such that the detector will make a different decision from the original one. Formally, given the input x at each timestamp and the RNN detector f, the objective of an attacker is to:

$$\begin{aligned} &\min\ \delta \\ &s.t.,\ f(\mathbf{x}) \neq f(\mathbf{x}+\delta)\ and\ ||\delta|| \leq \tau \end{aligned} \tag{2}$$

where τ is a small value which restricts the manipulation range of $\mathbf{x}$ according to a certain norm $||.||$ (to measure the distance of the modification). The success of such an attack will deceive the detector in two folds. In the case that the detector detects an actual anomaly, the adversarial attack will be able to bypass the detector and put the system in danger (without noticing that the system is in the abnormal state). In the case that the detector detects an actual normal status, the adversarial attack will deceit the detector to raise false alarms. We remark that how to construct adversarial examples using Eq. 2 has been extensively investigated in multiple domains [28, 36, 47, 59]. In the following, we introduce in details how we solve the problem in the CPS setting.

3.3.1 Adversarial Attacks on RNN predictor. In the primary setting, we only aim to construct adversarial examples for the RNN detector. Different from traditional adversarial attack domains for classification (e.g., image or audio), the anomaly detector uses CUSUM algorithm to check the difference between the predicted output and the actual output to classify normal and abnormal behavior, where the system will raise an anomaly alarm if the difference between them is beyond a threshold. Thus, we distinguish two scenarios in order to deceive the CUSUM checker. The first scenario is to mask the normal states to be recognized as abnormal. This results in a similar case with Eq. 2 as follows.

$$\begin{aligned} &\min\ \delta \\ &s.t.,\ ||f(\mathbf{x}+\delta)-y*)|| \geq \Delta\ and\ ||\delta|| \leq \tau \end{aligned} \tag{3}$$

where $\mathbf{x}$ is the original state, $y*$ is the actual output of the system, Δ is the acceptable difference (threshold for classifying anomaly). Notice that in this scenario the original state $\mathbf{x}$ is classified as normal, i.e., $||f(\mathbf{x}+\delta)-y*)|| < \Delta$.

The second scenario is to mask an anomaly as a normal system state. The formalization is slightly different as follows.

$$\begin{aligned} &\min\ \delta \\ &s.t.,\ ||f(\mathbf{x}+\delta)-y*)|| < \Delta\ and\ ||\delta|| \leq \tau \end{aligned} \tag{4}$$

Notice that in this scenario the original state $\mathbf{x}$ is classified as anomaly, i.e., $||f(\mathbf{x}+\delta)-y*)|| \geq \Delta$.

The remaining key challenge to solve Eq. 3 and Eq. 4 is to calculate the perturbation δ. For this, we borrow the idea of the Fast Gradient Sign Method (FGSM) method in image domain, which perturbs the input *along* the gradient direction to *maximize* the change in the model prediction[29]. Formally, the perturbation can be formalized as:

$$\begin{aligned} &\mathbf{x'} = \mathbf{x} + \delta \\ &\delta = \epsilon \cdot sgn(\nabla_x l(\mathbf{x}, y*)) \end{aligned} \tag{5}$$

where $\mathbf{x'}$ is the resultant adversarial example, ϵ is a constant representing the magnitude of the perturbation and l is the loss function. In this work, we choose Mean Square Error (MSE) as the loss function, which is the most commonly used for regression tasks. This is suitable for solving Eq. 3 where we aim to find a perturbation which would enlarge the difference between $f(\mathbf{x})$ and $y*$. On the other hand, to solve Eq. 4, we need to do the opposite and shrink the difference. To achieve this, we add perturbation along the *opposite* direction of the gradient instead. That is,

$$\begin{aligned} &\mathbf{x'} = \mathbf{x} - \delta \\ &\delta = \epsilon \cdot sgn(\nabla_x l(\mathbf{x}, y*)) \end{aligned} \tag{6}$$

Notice that in Eq. 5 and 6, ϵ is a constant. This is not suitable for our setting with variables of discrete values. To solve the problem, we adapt the perturbation as follows. propose an adapted version of the Eq.?? in practice:

$$\begin{aligned} &\mathbf{x'} = \begin{cases} \mathbf{x}+\delta, & if\ S_a = \text{normal} \\ \mathbf{x}-\delta, & if\ S_a = \text{attack} \end{cases} \\ &\delta = sgn(\nabla_{\mathbf{x}} l(\mathbf{x}, y')) \cdot \vec{\epsilon} \end{aligned} \tag{7}$$

where $\vec{\epsilon}$ is a diagonal matrix, and the value of each element from $\vec{\epsilon}$ is defined according to the data type of each feature which represents the magnitude of the perturbation for each feature. The smaller the $\vec{\epsilon}$ is, the less is the perturbation.

We distinguish three different types of features (to solve difficulty 3 in section 3), i.e., actuators, valves and pumps, thus add different perturbation on them. For sensors, we add a perturbation λ, which represents the percentage of the perturbation w.r.t. the original value. There are two types of actuators, valves and pumps. For valves, there are three values: 0 for opening/closing, 0.5 for close and 1 for open. For pumps, there are two values: 0 for close and

Algorithm 2: Adversarial examples construction for RNN detector.

Input: Data S_i; RNN predictor model f
Output: adversarial examples

1 $\delta \leftarrow sgn(\nabla_{\mathbf{x}} l(\mathbf{x}, y')) \cdot \vec{\epsilon}$;
2 **if** S_i *is normal data* **then**
3 $\quad S_i' \leftarrow S_i + \delta$;
4 **else**
5 $\quad S_i' \leftarrow S_i - \delta$;
6 **end**
7 **if** S_i' *cannot pass rule checker* **then**
8 $\quad S_i'' \leftarrow$ GA(S'[i]);
9 **else**
10 $\quad S_i'' \leftarrow S_i'$;
11 **end**
12 **return** S_i''

1 for open. In order to follow the character of discrete values for actuators, we define the ϵ of valves as 0.5 and the ϵ of pumps is defined as 1. In summary, the diagonal matrix $\vec{\epsilon}$ are defined as:

$$\vec{\epsilon}_{jj} = \begin{cases} \lambda, & if\ \vec{F}[j] \in \vec{U},\ where\ \vec{U} \subset \vec{F} \\ 0.5, & if\ \vec{F}[j] \in \vec{V_m},\ where\ \vec{V_m} \subset \vec{F} \\ 1, & if\ \vec{F}[j] \in \vec{V_p},\ where\ \vec{V_p} \subset \vec{F} \end{cases}$$

where $\vec{F}$ is the set of all features, $\vec{U}$ is the set of sensors, $\vec{V_m}$ is the set of all valves, and $\vec{V_p}$ is the set of all pumps. To give an example, if we only consider data from stage 1, $\vec{\epsilon}$ =

$$\begin{vmatrix} 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

In addition, to make sure the modified data are realistic, we always clip the adversarial examples to [0, 1] (with sensor values normalized). For example, for a pump with original value 1, if the sign of the gradient at $V_p[i, j]$ is positive, the perturbed pump value will be 2, which is not acceptable. We thus clip it back to 1.

In summary, we construct adversarial examples for the RNN detector by adding noise for two different attack scenarios in the above way. Algorithm 2 shows the details. In case the constructed adversarial example could not pass the rule checking system at line 7, we move to the next step to bypass it further.

3.3.2 Adversarial attacks considering rule checking. The rule checking mechanism is widely used to detect anomaly of CPS by many existing works [9, 10, 18, 49]. An example rule from SWaT system could be that when the value of "LIT101" is smaller than 500, the value of "MV101" should always equal to 2 after 12 seconds, otherwise an alarm will arise to indicate anomalies. In the practical setting, we aim to construct adversarial examples to bypass both the RNN detector and the rule checking system as well.

Our key observation is that the rule checking system makes decisions mainly depending on the status of actuators. Thus, we propose to use genetic algorithm (GA) [27] to accelerate the search for the ideal combination of actuator status to bypass it.

Algorithm 3: GA

Input: Data S'[i];fitness function g; population size n;mutation probability p_m; rule checker R
Output: adversarial examples

1 Generate an initial population P;
2 **repeat**
3 $\quad$ Compute fitness value for each candidate from P;
4 $\quad$ Select parents from P with a probability of fitness distribution;
5 $\quad$ Apply crossover and mutation with p_m to get new candidates;
6 $\quad$ Select fittest individuals from old and new candidates to form new P;
7 $\quad S''[i] \leftarrow$ fittest candidate from P;
8 **until** *R(S"[i]) = True or iteration out*;
9 **return** *S"[i]*

Algorithm 3 shows the details of using GA to identify the ideal combination of actuator status. The input of the algorithm includes the data point to manipulate ($S'[i]$), a set of standard parameters for genetic algorithms (g, n and p_m) and a rule checker R implemented in the system (for SWaT, there are in total 23 rule to monitor the system status). The algorithm follows a standard workflow of genetic algorithms [27]. Firstly, an initial population with size n is generated randomly (by randomly manipulating the actuator values) at line 1. Afterwards, we repeat selecting the fitness candidates according to the following fitness function:

$$g = c1 * /c2$$

, where c_1 indicates whether the candidate could pass the rule checker and c_2 is the magnitude of the modification. Intuitively, this fitness function allows us to favor candidates which pass the rule checker meanwhile with the minimum modification on the actuator. In practice, we set $c1 = 1$ if $R(x) = True$ and $c1 = 0$ otherwise (impossible to be chosen). We set c_2 as the ratio of actuators that has been modified. The new population is formed in the following way (from line 4 to line 6). Each candidate is assigned a probability according to the roulette wheel selection [27], where the probability of each candidate being selected as parent is $g_i / \sum_{j=1}^{n} g_j$. We then apply crossover on the selected parents in a normal way. Afterwards, we randomly mutate the resultant candidates with probability p_m. Next, we calculate the fitness score for all candidates and select n fittest candidates form the new population. This procedure continues until the fittest candidate passes the rule checker or we run out of iterations at line 8.

3.4 Defence model

Our next question is whether it is possible to detect the constructed examples efficiently as a defence. Our attempt is to create a binary classifier of different model types (e.g., def-NN and random forest) trained from a set of original data points and an equal number of adversarial examples. We then use the trained classifier to test

whether the classifier can detect adversarial examples constructed from other data points. We focus on the defence neural network (def-NN) performance and use the random forest as a comparison.

4 EVALUATION ON CYBER-PHYSICAL SYSTEMS

In this section, we describe the experiments conducted and draw some conclusions from them.

4.1 Experiment Setup

Our experiments were conducted using datasets from the SWaT and WADI testbeds, consisting of sensor readings, actuator states, and network traffic that was collected from the respective historians after several days of operation (see 3.1). With this data, we conducted our learning and experiments on a laptop with 1 Intel(R) Core(TM) i7-8750H CPU at 2.20GHz, 16GB system memory, and for the GPU, NVIDIA GeForce GTX 1050 Ti with Max-Q Design.

4.2 Experiments and Results

Our experiments are designed around our three research questions. We address their design and examine their results in turn.

RQ1: Are our adversarial attacks effective on real-world CPS?

To assess this research question, we design an experiment to explore how much noise (i.e. λ, the percentage of perturbation) is required for the attack to be effective at reducing the accuracy of the anomaly detector. Due to the mix of sensors and actuators in CPSs, we have a number of scenarios to consider, which are enumerated in Table 7. First, where noise is applied only to the sensors (1% or 10% noise). Then, when noise is applied to all the data, including the actuators. Finally, the previous scenario but with the addition of a GA to help overcome the rule-checking mechanism that is present (this parameter will be specifically discussed in RQ2). For example, $\vec{\epsilon}$ is defined with 26 elements with $\lambda = 0.01$ for sensor value for parameter 'noise to all data 1%', and $\lambda = 0.1$ for sensor value for parameter 'noises to all data 10%', with 6 elements with 0.5 for actuator valves and 19 elements with 1 for actuator pumps value with locations responding to each feature, and the same logic has been applied to WADI.

Before applying noise, first, RNN predictors (as described by Goh et al. [26]) are trained using normal data, in which there are 51 features for SWaT and 21 features for WADI. Recall that these RNNs make a prediction of the data at time t from a window of continuous data, which we use to compare with the actual records to decide if it is normal or an attack. In our experiments, we choose different window sizes for SWaT and WADI. For SWaT, we use 12s, which is the maximum delay from the rule-checking system. For WADI, we use 10 timestamps, due to the slow speed of change in its physical processes. As the system is dynamic and the ground truth is changing whenever data is changing (see the difficulty discussed in Section 3), we compare the results of the ground truth model $Y_T = T(S)$ with the results from the anomaly detector $Y_C = C(f(S))$ to calculate the original anomaly detector accuracy of data S, and get the different between $T(S')$ and $C(f(S'))$ to get accuracy of data S'.

The results for the different parameters are given in Table 8 for SWaT and Table 9 for WADI. We observe that the precision, f1, and accuracy have all been reduced after the adversarial attack. Intuitively, accuracy appears to decrease as more noise is added, both in terms of percentage and range of data it is applied to. However, it is possible that with the larger amounts of noise, the noisy data has exceeded the threshold defined as an attack by the model T. Thus the precision, recall and f1 for 10% noises are all similar or higher than 1% noises. Comparing to the original performance of the anomaly detector, the reduced accuracy has shown that the anomaly detector is vulnerable and sensitive to the adversarial attacks.

The bottom rows of tables, in which noise is applied to both sensor and actuator values at the same time, have lower accuracy than the sensor-only parameters; possibly because involving actuators results in a stronger impact on the RNN predictor, fed through to the CUSUM calcuation. The highlighted false negative rates of 0.78 and 0.99 (see the tables) implies that many false alarms have been generated by the anomaly detector, indicating that changing actuators is easy to be caught by an anomaly detector as an attack. However, the false negative value of 0.46 highlighted in the SWaT table, is not high compared to the above: this could be because fewer actuators were changed when we calculated the gradients (i.e. most actuators have gradients as 0).

We remark that for WADI, ground truth was determined in a slightly different way to SWaT: instead of using control points (which are not provided for WADI), we calculate the operation range under normal conditions with a small tolerance to set the ground truth model T. As it has more features (121) than SWaT (51), the complexity is more apparent and behavior is more unpredictable.

RQ2: how should the attacker compromise the original data to deceive the rule checking system and the anomaly detector?

Rule checking systems are commonly used in industrial CPSs as a basic defence mechanism. As WADI has not yet established a rule checking system, our second experiment is only applied to the SWaT testbed, which monitors invariants related to a total of 20 actuators and 11 sensors. To overcome this, our experiment applies noise to all data, then fine-tunes the actuator states by using a GA in order to pass the rule-checker. We assume that the attacker does not have the access to the content of the rule, but knows whether the data could pass the rule checker or not. The rules include the relationship between the sensor and actuator values, so we divided a data point x into $x_a = [x_{a1}, x_{a2} \dots]$ and $x_s = [x_{s1}, x_{s2} \dots]$, and use the GA to generate possible x_a and keep x_s to combine back to calculate the fitness score. As our goal is to generate adversarial attacks rather than break down the system, we only use GA to replace those attacks that are being detected by rule checker due to noises.

The results of this experiment are given in the last row of Table 8. The accuracy has been slightly increased indicating that using GA for actuator value selection may sacrifice the drop of accuracy which is reasonable, as GA generated values do not change following the sign of gradients. Aside from this, overall performance is almost the same as the same parameter that does not make use of a GA, suggesting that this one allows us to pass the rule checker

Table 7: Parameter summary for SWaT and WADI experiments

	sensor	actuator	
	λ for sensor	λ for valves	λ for pumps
Noises to sensor (1%)	0.01	0	0
Noises to sensor (10%)	0.1	0	0
Noises to all data (1%)	0.01	0.5	1
Noises to all data (10%)	0.1	0.5	1
Noises to all data + GA for actuator	0.1	0.5	1

Table 8: Before and after adding noises of anomaly detector for SWaT

	precision	recall	f1	accuracy	False negative	False positive
Original Anomaly Detector	0.96	0.86	0.91	89.40%	0.05	0.14
Noises to sensor (1%)	0.62	0.53	0.57	52.38%	0.48	0.47
Noises to sensor (10%)	0.65	0.58	0.61	48.78%	0.72	0.42
Noises to all data (1%)	0.52	0.50	0.51	51.85%	**0.46**	0.50
Noises to all data (10%)	0.12	0.84	0.21	28.84%	**0.78**	0.16
Noises to all data + GA for actuator	0.14	0.76	0.24	39.56%	0.62	0.34

Table 9: Before and after adding noises of anomaly detector for WADI

	precision	recall	f1	accuracy	False negative	False positive
Original Anomaly Detector	0.32	0.16	0.22	93.29%	0.02	0.84
Noises to sensor (1%)	0.43	0.65	0.52	52.85%	0.55	0.35
Noises to sensor (10%)	0.42	0.67	0.52	49.88%	0.62	0.33
Noises to all data (1%)	0.34	0.96	0.50	33.28%	**0.99**	0.04
Noises to all data (10%)	0.33	0.95	0.49	32.23%	**0.99**	0.04

without much reducing the anomaly detector performance.

RQ3: how can we design a defence against such adversarial examples?

The adversarial examples have more noise compared to original data, so we design a *defence neural network (def-NN)* and a commonly used classification model—random forest (RF)—to distinguish if the data is from an adversarial example or original data from the system. The def-NN and RF perform as two basic binary classifiers. We train the model using a dataset consisting of 50% original data and 50% adversarial examples, reserving a portion (with the same ratio) as testing data. The def-NN model is designed with three layers and 100 units for each layer, and we choose *binary cross entropy* as our loss function. The random forest is designed with 10 estimators.

The results can be found in Table 10 and Table 11: both the def-NN and random forest models work well at detecting the noise. However, this is only the case for the type of noise that they were trained on: a model trained on 10% levels of noise performs well at detecting 10% levels of noise, but not so much at 5% of noise, for example. In practice, it is hard for the defender to know what kind of noise will be applied. However, an attacker in practice is likely to want to use as little noise as possible (to avoid detection), so a model that is trained on a mix of samples of different lower levels of noise (e.g.1-3%) may perform well. This requires further investigation.

4.3 Threats to Validity

There are some limitations for the evaluation and application validity. First, though the two CPS systems are real and operational, they are still testbeds and not at the scale of operational industrial plants. Secondly, the data was collected and labelled in a manual effort, and thus there may be data points that are not accurate, and could increase the bias of the results. Finally, we only considered the recorded attacks of the datasets to train and calculate ground truth, so the methods may not work well for non-recorded attacks. Similarly, the defence method (i.e. def-NN) has to be trained for each kind of adversarial examples (10% noises and 1% noises), and it does not work for unknown adversarial examples (e.g. 5% noises).

5 RELATED WORK

This work focuses on adversarial attacks on LSTM-RNN which is implemented as a part of anomaly detector in a cyber-physical system. In this section, we discuss some researches with relevant themes such as adversarial attacks and some relevant works done on SWaT/WADI systems.

Anomaly detection on CPS: A number of researches have explored the anomaly detection on CPS, these approaches [5, 11, 21, 32, 33, 37, 43, 45, 51, 54, 57] unusual monitor the changes or suspicious patterns from the logs of a CPS. Y.Harada et al [32] has proposed a statistical way to monitor the outliers from logs. This statistical method can include machine learning or deep learning algorithms.

Table 10: Defence for adversarial attacks on SWaT

	def-NN			RF		
	precision	**recall**	**f1**	**precision**	**recall**	**f1**
Noises to sensor (1%)	0.98	0.99	0.99	1	0.99	0.99
Noises to sensor (10%)	0.99	0.99	0.99	1	1	1
Noises to all data (1%)	1	1	1	1	1	1
Noises to all data (10%)	1	1	1	1	1	1
Noises to all data + GA for actuator	1	1	1	1	1	1

Table 11: Defence for adversarial attacks on WADI

	def-NN			RF		
	precision	**recall**	**f1**	**precision**	**recall**	**f1**
Noises to sensor (1%)	0.5	1	0.67	1	1	1
Noises to sensor (10%)	1	1	1	1	1	1
Noises to all data (1%)	1	1	1	1	1	1
Noises to all data (10%)	1	1	1	1	1	1

J.Inoue et al [37] has provided an unsupervised machine learning methods to detect anomalies and M. Kravchik et al [43] proposed to use convolutional neural network to detect cyber attacks in control systems.

SWaT system as a typical cyber-physical system has been widely explored by researches, and there are many works including anomaly detection done on the public data set [38, 39]. These works include unsupervised learning methods [26, 37, 43], which include deep learning methods, and supervised learning methods [16, 17], who injected simulated faults to get attack data. Ahmed et al. [6, 8] has worked on fingerprinting the noises patterns to detect anomalies while Adepu and Mathur [1–4] have manually derived the invariants or what we call "rules" by monitoring the normal operation of the system. C. Feng [23] also tried to get invariants but using a systematic way to learn the invariants.

Neural networks in CPS: Neural networks as a state-of-art technique has been adopted in CPS as well. Lv et al. [46] have introduced a multiple layers neural networks for probabilistic estimation of brake pressure for electrified vehicles. Nanduri et al. [50] have applied RNN as a part of anomaly detector in aircraft data. Sargolzaei et al. [56] have designed a fault detector with neural networks to detect fault data injection for vehicular cyber-physical systems with wireless communications. Similar works with anomaly detection with Neural networks have been evaluate on smart grid [42] and gasoil plant [24], which use LSTM as predictor. Eiteneuer et al. [22] have applied LSTM neural networks to learn the behavior of a water tank.

Adversarial attacks: Papernot et al. have pointed out the limitations of deep learning in adversarial settings [53] and Nguyen et al. have pointed in their work [52] that it's easy to produce images that completely unrecognizable by human but recognizable with 99.99% confidence by state-of-art neural networks. Adversarial attacks have many mature toolbox [30?] to generate adversarial attacks. Moreover, adversarial attacks has been adopted in many applications in physical world [44] including image [58] and audio [13, 14]. However, there are few works on CPS.

6 CONCLUSION

In this paper, we test an anomaly detector of real-world cyber-physical systems by generating adversarial attacks, and we explore possible defences for the adversarial attacks. Firstly, the targeting anomaly detector consists of a LSTM-RNN as predictor, we applied noises with different ranges and compare their impacts. Then we weaken our thread model by considering rule checking mechanism and provide a solution to overcome the checker, which is by using GA. Finally, we explore if there are any possible methods to detect the adversarial examples and tried def-NN and random forest for the experiments. Afterwards, we evaluate our methods on two testing beds, SWaT and WADI and get results to show the anomaly detector is vulnerable with adversarial attacks, and even though we consider the rule checking mechanism to weaken the assumptions of thread model, we can still achieve a similar results with our improved adversarial attack methods.

For future works, we will consider to weaken the attacker model with black box adversarial attack, and we will explore other methods to overcome the rule checker such as SMT solver. We will evaluate our methods with the actual system and real time application. Moreover, we will investigate possible solutions for adversarial example detection.

REFERENCES

[1] Sridhar Adepu and Aditya Mathur. 2016. Distributed Detection of Single-Stage Multipoint Cyber Attacks in a Water Treatment Plant. In *Proc. ACM Asia Conference on Computer and Communications Security (AsiaCCS 2016)*. ACM, 449–460.

[2] Sridhar Adepu and Aditya Mathur. 2016. Using Process Invariants to Detect Cyber Attacks on a Water Treatment System. In *Proc. International Conference on ICT Systems Security and Privacy Protection (SEC 2016) (IFIP AICT)*, Vol. 471. Springer, 91–104.

[3] Sridhar Adepu and Aditya Mathur. 2018. Assessing the Effectiveness of Attack Detection at a Hackfest on Industrial Control Systems. *IEEE Transactions on Sustainable Computing* (2018).

[4] Sridhar Adepu and Aditya Mathur. 2018. Distributed Attack Detection in a Water Treatment Plant: Method and Case Study. *IEEE Transactions on Dependable and Secure Computing* (2018).

[5] Ekta Aggarwal, Mehdi Karimibiuki, Karthik Pattabiraman, and André Ivanov. 2018. CORGIDS: A Correlation-based Generic Intrusion Detection System. In *Proc. Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC 2018)*. ACM, 24–35.

[6] Chuadhry Mujeeb Ahmed, Martín Ochoa, Jianying Zhou, Aditya P. Mathur, Rizwan Qadeer, Carlos Murguia, and Justin Ruths. 2018. *NoisePrint*: Attack Detection Using Sensor and Process Noise Fingerprint in Cyber Physical Systems. In *Proc. Asia Conference on Computer and Communications Security (AsiaCCS 2018)*. ACM, 483–497.

[7] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur. 2017. WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks - CySWATER '17*. ACM Press, Pittsburgh, Pennsylvania, 25–28. https://doi.org/10.1145/3055366.3055375

[8] Chuadhry Mujeeb Ahmed, Jianying Zhou, and Aditya P. Mathur. 2018. Noise Matters: Using Sensor and Process Noise Fingerprint to Detect Stealthy Cyber Attacks and Authenticate sensors in CPS. In *Proc. Annual Computer Security Applications Conference (ACSAC 2018)*. ACM, 566–581.

[9] Ravi Akella and Bruce M. McMillin. 2009. Model-checking BNDC properties in cyber-physical systems. In *2009 33rd Annual IEEE International Computer Software and Applications Conference*, Vol. 1. IEEE, 660–663.

[10] Rajeev Alur. 2015. *Principles of cyber-physical systems*. MIT Press.

[11] Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. 2018. Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS 2018)*. ACM, 817–831.

[12] J. Roger Bray and J. T. Curtis. 1957. An Ordination of the Upland Forest Communities of Southern Wisconsin. *Ecological Monographs* 27, 4 (Feb. 1957), 325–349. https://doi.org/10.2307/1942268

[13] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 513–530.

[14] Nicholas Carlini and David Wagner. 2018. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. IEEE, San Francisco, CA, 1–7. https://doi.org/10.1109/SPW.2018.00009

[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.* 41, 3, Article 15 (July 2009), 58 pages. https://doi.org/10.1145/1541880.1541882

[16] Yuqi Chen, Christopher M. Poskitt, and Jun Sun. 2016. Towards Learning and Verifying Invariants of Cyber-Physical Systems by Code Mutation. In *Proc. International Symposium on Formal Methods (FM 2016) (LNCS)*, Vol. 9995. Springer, 155–163.

[17] Yuqi Chen, Christopher M. Poskitt, and Jun Sun. 2018. Learning from Mutants: Using Code Mutation to Learn and Monitor Invariants of a Cyber-Physical System. In *Proc. IEEE Symposium on Security and Privacy (S&P 2018)*. IEEE Computer Society, 648–660.

[18] Yuqi Chen, Christopher M. Poskitt, and Jun Sun. 2018. Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 648–660.

[19] Yuqi Chen, Christopher M. Poskitt, Jun Sun, Sridhar Adepu, and Fan Zhang. 2019. Learning-Guided Network Fuzzing for Testing Cyber-Physical System Defences. In *Proc. IEEE/ACM International Conference on Automated Software Engineering (ASE 2019)*. IEEE Computer Society, 962–973.

[20] Feng Cheng, Tingting Li, and Deeph Chana. 2017. Bloom Filters and LSTM Networks for Anomaly Detection in Industrial Control Systems. In *47th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2017)*. IEEE, Washington D.C. , USA, 1–12.

[21] Long Cheng, Ke Tian, and Danfeng (Daphne) Yao. 2017. Orpheus: Enforcing Cyber-Physical Execution Semantics to Defend Against Data-Oriented Attacks. In *Proc. Annual Computer Security Applications Conference (ACSAC 2017)*. ACM, 315–326.

[22] Benedikt Eiteneuer and Oliver Niggemann. 2018. LSTM for Model-based Anomaly Detection in Cyber-Physical Systems.. In *DX@ Safeprocess*.

[23] Cheng Feng, Venkata Reddy Palleti, Aditya Mathur, and Deeph Chana. 2019. A Systematic Framework to Generate Invariants for Anomaly Detection in Industrial Control Systems. In *Proc. Annual Network and Distributed System Security Symposium (NDSS 2019)*. The Internet Society.

[24] Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. 2016. Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. *arXiv:1612.06676 [cs, stat]* (Dec. 2016). http://arxiv.org/abs/1612.06676 arXiv: 1612.06676.

[25] 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. https://doi.org/10.14722/ndss.2016.23142

[26] Jonathan Goh, Sridhar Adepu, Marcus Tan, and Zi Shan Lee. 2017. Anomaly detection in cyber physical systems using recurrent neural networks. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*. IEEE, 140–145.

[27] David E. Goldberg. 1989. Genetic Algorithms in Search Optimization and Machine Learning. https://doi.org/10.5860/choice.27-0936

[28] Yuan Gong and Christian Poellabauer. 2017. Crafting adversarial examples for speech paralinguistics applications. *arXiv preprint arXiv:1711.03280* (2017).

[29] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[30] Dbou Goodman, Hao Xin, Wang Yang, Wu Yuesheng, Xiong Junfeng, and Zhang Huan. 2020. Advbox: a toolbox to generate adversarial examples that fool neural networks. *arXiv:2001.05574 [cs, stat]* (Jan. 2020). http://arxiv.org/abs/2001.05574 arXiv: 2001.05574.

[31] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*. Springer, 62–79.

[32] Yoshiyuki Harada, Yoriyuki Yamagata, Osamu Mizuno, and Eun-Hye Choi. 2017. Log-Based Anomaly Detection of CPS Using a Statistical Method. In *Proc. International Workshop on Empirical Software Engineering in Practice (IWESEP 2017)*. IEEE, 1–6.

[33] Zecheng He, Aswin Raghavan, Guangyuan Hu, Sek M. Chai, and Ruby B. Lee. 2019. Power-Grid Controller Anomaly Detection with Enhanced Temporal Deep Learning. In *Proc. IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom 2019)*. IEEE, 160–167.

[34] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[35] Elike Hodo, Xavier Bellekens, Andrew Hamilton, Pierre-Louis Dubouilh, Ephraim Iorkyase, Christos Tachtatzis, and Robert Atkinson. 2016. Threat analysis of IoT networks using artificial neural network intrusion detection system. In *2016 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 1–6.

[36] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).

[37] Jun Inoue, Yoriyuki Yamagata, Yuqi Chen, Christopher M. Poskitt, and Jun Sun. 2017. Anomaly Detection for a Water Treatment System Using Unsupervised Machine Learning. *2017 IEEE International Conference on Data Mining Workshops (ICDMW)* (Nov. 2017), 1058–1065. https://doi.org/10.1109/ICDMW.2017.149 arXiv: 1709.05342.

[38] iTrust. [n.d.]. Dataset and Models. https://itrust.sutd.edu.sg/dataset/.

[39] Goh J., Adepu S., Junejo K. N, and Mathur A. 2016. A Dataset to Support Research in the Design of Secure Water Treatment Systems. Springer, New York, USA, 1–13.

[40] Yue Jia and Mark Harman. 2011. An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering* 37, 5 (Sept. 2011), 649–678. https://doi.org/10.1109/TSE.2010.62

[41] Anna Magdalena Kosek. 2016. Contextual anomaly detection for cyber-physical security in Smart Grids based on an artificial neural network model. In *2016 Joint Workshop on Cyber- Physical Security and Resilience in Smart Grids (CPSR-SG)*. IEEE, Vienna, Austria, 1–6. https://doi.org/10.1109/CPSRSG.2016.7684103

[42] Anna Magdalena Kosek. 2016. Contextual anomaly detection for cyber-physical security in smart grids based on an artificial neural network model. In *2016 Joint Workshop on Cyber-Physical Security and Resilience in Smart Grids (CPSR-SG)*. IEEE, 1–6.

[43] Moshe Kravchik and Asaf Shabtai. 2018. Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks. In *Proc. Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC 2018)*. ACM, 72–83.

[44] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[45] Qin Lin, Sridhar Adepu, Sicco Verwer, and Aditya Mathur. 2018. TABOR: A Graphical Model-based Approach for Anomaly Detection in Industrial Control Systems. In *Proc. Asia Conference on Computer and Communications Security (AsiaCCS 2018)*. ACM, 525–536.

[46] Chen Lv, Yang Xing, Junzhi Zhang, Xiaoxiang Na, Yutong Li, Teng Liu, Dongpu Cao, and Fei-Yue Wang. 2017. Levenberg–Marquardt backpropagation training of multilayer neural networks for state estimation of a safety-critical cyber-physical system. *IEEE Transactions on Industrial Informatics* 14, 8 (2017), 3436–3446.

[47] Francesco Marra, Diego Gragnaniello, and Luisa Verdoliva. 2018. On the vulnerability of deep learning to adversarial attacks for camera model identification. *Signal Processing: Image Communication* 65 (2018), 240–248.

[48] Aditya P. Mathur and Nils Ole Tippenhauer. 2016. SWaT: a water treatment testbed for research and training on ICS security. In *Proc. International Workshop on Cyber-physical Systems for Smart Water Networks (CySWATER 2016)*. IEEE Computer Society, 31–36.

[49] Sayan Mitra, Tichakorn Wongpiromsarn, and Richard M. Murray. 2013. Verifying cyber-physical interactions in safety-critical systems. *IEEE Security & Privacy* 11, 4 (2013), 28–37.

[50] Anvardh Nanduri and Lance Sherry. 2016. Anomaly detection in aircraft data using Recurrent Neural Networks (RNN). In *2016 Integrated Communications Navigation and Surveillance (ICNS)*. IEEE, Herndon, VA, USA, 5C2–1–5C2–8. https://doi.org/10.1109/ICNSURV.2016.7486356

[51] Vedanth Narayanan and Rakesh B. Bobba. 2018. Learning Based Anomaly Detection for Industrial Arm Applications. In *Proc. Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC 2018)*. ACM, 13–23.

[52] Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 427–436.

[53] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 372–387.

[54] Fabio Pasqualetti, Florian Dorfler, and Francesco Bullo. 2011. Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design. In *Proc. IEEE Conference on Decision and Control and European Control Conference (CDC-ECC 2011)*. IEEE, 2195–2201.

[55] Arman Sargolzaei, Carl D. Crane, Alireza Abbaspour, and Shirin Noei. 2016. A Machine Learning Approach for Fault Detection in Vehicular Cyber-Physical Systems. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Anaheim, CA, USA, 636–640.

[56] Arman Sargolzaei, Carl D Crane, Alireza Abbaspour, and Shirin Noei. 2016. A machine learning approach for fault detection in vehicular cyber-physical systems. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 636–640.

[57] Peter Schneider and Konstantin Böttinger. 2018. High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks. In *Proc. Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC 2018)*. ACM, 1–12.

[58] Daniel F Smith, Arnold Wiliem, and Brian C Lovell. 2015. Face recognition on consumer devices: Reflections on replay attacks. *IEEE Transactions on Information Forensics and Security* 10, 4 (2015), 736–745.

[59] Muhammad Usama, Junaid Qadir, and Ala Al-Fuqaha. 2018. Adversarial attacks on cognitive self-organizing networks: The challenge and the way forward. In *2018 IEEE 43rd Conference on Local Computer Networks Workshops (LCN Workshops)*. IEEE, 90–97.

[60] Jingyi Wang, Jun Sun, Yifan Jia, Shengchao Qin, and Zhiwu Xu. 2018. Towards 'Verifying'a Water Treatment System. In *International Symposium on Formal Methods*. Springer, 73–92.

[61] Z. Zohrevand, U. Glasser, H. Y. Shahir, M. A. Tayebi, and R. Costanzo. 2016. Hidden Markov based anomaly detection for water supply systems. In *2016 IEEE International Conference on Big Data (Big Data)*. 1551–1560.