Generating Adversarial Examples Against Machine Learning-Based Intrusion Detector in Industrial Control Systems

Jiming Chen, *Fellow, IEEE*, Xiangshan Gao, *Student Member, IEEE*, Ruilong Deng, *Senior Member, IEEE*, Yang He, Chongrong Fang, *Student Member, IEEE*, and Peng Cheng, *Member, IEEE*

Abstract—Deploying machine learning (ML)-based intrusion detection systems (IDS) is an effective way to improve the security of industrial control systems (ICS). However, ML models themselves are vulnerable to adversarial examples, generated by deliberately adding subtle perturbation to the input sample that some people are not aware of, causing the model to give a false output with high confidence. In this article, our goal is to investigate the possibility of stealthy cyber attacks towards IDS, including injection attack, function code attack and reconnaissance attack, and enhance its robustness to adversarial attack. However, adversarial algorithms are subject to communication protocol and legal range of data in ICS, unlike only limited by the distance between original samples and newly generated samples in image domain. We propose two strategies - optimal solution attack and GAN attack - oriented to flexibility and volume of data, formulating an optimization problem to find stealthy attacks, where the former is appropriate for not too large and more flexible samples while the latter provides a more efficient solution for larger and not too flexible samples. Finally, we conduct experiments on a semi-physical ICS testbed with a high detection performance ensemble ML-based detector to show the effectiveness of our attacks. The results indicate that new samples of reconnaissance and function code attack produced by both optimal solution and GAN algorithm possess 80 percent higher probability to evade the detector, still maintaining the same attack effect. In the meantime, we adopt adversarial training as a method to defend against adversarial attack. After training on the mixture of orginal dataset and newly generated samples, the detector becomes more robust to adversarial examples.

Index Terms—Machine learning security, intrusion detection systems, industrial control systems, adversarial examples

1 Introduction

Critical infrastructures controlled and monitored by industrial control systems (ICS) used to be isolated and independent. Yet traditional industries have integrated communication network technology in recent years [1], [2]. Owing to transmitting information in plain text, not updating the system and fixing the vulnerabilities timely, current ICS is fragile. Since 2016, the number of attacks such as viruses and trojans on ICS has been significantly increasing. Taking the Stuxnet [3] virus targeting at the Iranian nuclear power plant in 2010 as an example, control system failures caused by malicious attacks sound an alarm for the cyber security issues of ICS. In ICS, ML-based intrusion detection systems (IDS) can achieve good performance in network traffic recognition [4]. To be more specific, ML techniques enable the ability to find out patterns from a large amount of historical data to build detection systems. Besides, it lowers high false positive alert introduced by using manpower setting detection rules for features or combination of features. Thus, ML-based [5] IDS in ICS can be auxiliary and less costly for rule-based IDS [6]. By carefully learning the network data, ML-based IDS [7] can perform high detection precision and low false alarm. Though there is no available direct product, establishing ML-based IDS by exploiting the control network traffic data has been an inevitable trend.[1]

However, as recent studies show [8], ML models may be easily deceived by adversarial examples at the test time. An adversary can modify the testing example slightly such that the ML models produce a result that is different from the previous output. Reasons for the feasibility of evasion attacks are divided into two aspects [9]: the characteristics (linearity or nonlinearity) of the ML models and the overfitting caused by insufficient regularization. This brings potential threats to ML-based system, especially for challenging critical occasions, such as ICS.

The goal of this paper is to investigate the possibility of generating stealthy attacks that can evade ML-based IDS in ICS and improve the robustness of detector. In fact, previous research work has initiated similar attacks in the field of computer vision and malware detection. Several new challenges arise when considering ICS scenarios. First, constraints on control network packets that come from the network protocol

• *The authors are with the State Key Laboratory of Industrial Control Technology and the College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China. E-mail: {cjm, corazju, dengruilong, 21632042, crfang_zju, lunar_heart}@zju.edu.cn.*

Manuscript received 13 Nov. 2019; revised 11 Sept. 2020; accepted 4 Nov. 2020. Date of publication 12 Nov. 2020; date of current version 13 May 2022. (Corresponding author: Ruilong Deng.) Digital Object Identifier no. 10.1109/TDSC.2020.3037500

1. ICS Security Market Report 2024 by Industry Insights, Company Overview and Investment Analysis.

in ICS limit allowable modifications on data. For example, some data fields have legal range, and some data fields defined as integers cannot be changed continuously, while some data fields, such as ID, must keep unchanged. These restrictions ensure that the adversarial examples are legal for the control system and comply with protocol format to be parsed correctly. Second, the adversarial examples are required to possess valid malicious payload and evade IDS to be classified as benign. The newly generated samples sent to communication individuals must still maintain the same attack effect like the original attack samples, such as water level surges or PLC suddenly stops. In the meantime, these samples will be recognized as good by the detector. Those specific requirements make the available attack generation methods in literature inapplicable to ICS.

In order to evaluate our attack generating methods, we establish a semi-physical ICS testbed with real controllers (hardware devices) and virtual control objects (water tank). Taking network packets gathered in this testbed (under attack and normal state) as input to our attack algorithm, we can confirm whether the output of our attack algorithm (generated adversarial packets) in this testbed remains harmful by observing the water level, the state of PLC and the communication traffic.

Our Work. First, this paper proposes an approach to produce adversarial examples against ML-based IDS in ICS through optimization method. By leveraging the property of ML algorithms, we generate evasion attacks through constructing an optimal problem, which maximizes the probability of malicious samples being classified as benign. The main idea is to transform an initial malicious example by calculating and processing iteratively under the complicated constraints until that it is predicted as benign. This kind of attack is called *Opt. attack* (optimization solution attack), which is pretty easy to implement and can be applied to any ML-based classification algorithm. To our best knowledge, this paper is the first try to implement such attacks in ICS. Additional constraints here are taken into account when solving Opt. attack. For example, we add the prior knowledge of control system, including protocol format and topology, etc, to our adversarial attacks. Different from image processing where each pixel has similar importance, in ICS various data dimensions (like different value outputs) are affected much differently by adversarial attacks. Besides, different attacks in ICS will maintain malicious payload to different degrees. That is, we should customize generation strategies for different kinds of attacks (e.g., injection attacks, function code attacks and reconnaissance attacks).

Second, we use *generative adversarial networks* (GAN) to produce adversarial examples towards ML-based IDS in ICS. The Opt. attacks are too complex to handle a large volume of data [10]. Besides, there is another worry that the pattern of generating new samples is too single. Once some adversarial examples are captured, others could not maintain stealthy to the classifier. To overcome the difficulties, we propose a new method to launch evasion attacks in ICS using GAN. The original GAN is designed to fit a distribution with random noise. Here, we redesign the loss function of the generator and discriminator to realize the ICS adversarial examples' generation. The inputs of generator are initial malicious instances that can be detected by IDS. The

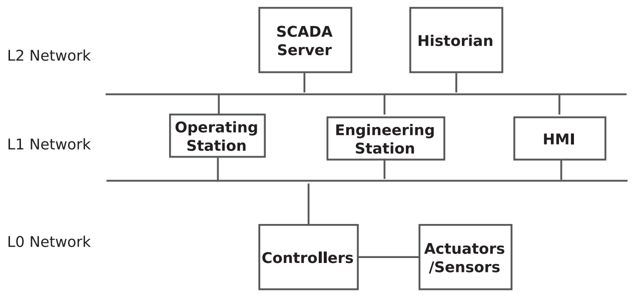


Fig. 1. Typical ICS architecture.

outputs are adversarial examples that can evade IDS. To preserve attack effect on ICS, for one specific category of original attack, we remove its key attack payload from inputs of the generator to ensure the immutability of attack payload in new generated adversarial examples. Based on this approach, we can take arbitrary initial instance as input and then generate the corresponding adversarial examples.

Third, we evaluate our methods on a semi-physical ICS testbed and an ensemble detector. By testing our generated adversarial examples (network packets), it is validated that the proposed methods are able to generate stealthy attack samples which can evade the powerful detector, simultaneously keeping malicious to the control process, such as water level surging and the state of PLC changing.

In summary, the novel contributions of this paper are concluded as follows:

- We generate two types of evasion attacks on the ML-based IDS in ICS. It is the first try to transit from free input image domains to restricted input ICS domains.
- We build an ICS security testbed to judge the newly generated samples' physical attack effect and design an outstanding IDS which ensembles more than one ML algorithm to detect the abnormal network behavior with high accuracy.
- We reveal that machine learning based IDS in ICS is indeed vulnerable to adversarial examples, while in the meantime, we choose adversarial training as a defensive method to alleviate the problem.

2 Background

In this section, we briefly introduce the typical structure of ICS, three types of attacks, GAN and Black-box situation in this paper.

2.1 Typical ICS Structure

A typical ICS framework includes management level (L2 Network) [11], [12], supervisory level (L1 Network) and control level (L0 Network), as shown in Fig. 1. In L2 Network, there are SCADA server and historian which collect and analyze data uploaded from the underlying network [13]. Computers in this level may connect to the Internet with the protection of firewall. L1 Network consists of engineering stations and operating stations, which perform configuration of controllers, such as updating the control logic. In addition, work stations here send control commands to controllers and collect data from them. In L0 Network, there are several kinds of control devices, such as programmable logic controller (PLC), remote terminal unit (RTU) and so on. Those devices control physical equipment through actuators and receive feedback from sensors. In this paper, we focus on the attacks in L0 Network level that can directly influence the controllers.

2.2 Initial ICS Attack

As vulnerabilities in control network and control devices are special in ICS, the corresponding attacks are also different from those in traditional IT zones (L1 and L2 Network). For example, attacks against controllers are inaccessible in IT zone. In this paper, we pay attention to the adversarial attack in the control zone (L0 Network), which affect control system directly. That is, we focus on attacks that can influence communication protocols in the control system (like S7comm, Modbus, etc.). Besides, attacks in this paper are generated by various scripts and experimented in real environments. The attacks[2] are categorized as below:

Injection Attack. Malicious data injection attackers should have a deep understanding of control devices, communication protocols as well as system model. Then they invade the control subnet and pretend as a work station to modify the data in control devices. This kind of attack will drive the system into a wrong state. The 2015 Ukraine blackout is mainly caused by injection attack [14]. Due to many methods aimed at producing more stealthy injection attacks, traditional detection algorithms always fail. This can be mainly attributed to two reasons: First, as we have mentioned before, setting a simple range check for multiple data values needs more manpower and prior knowledge of the system, and sometimes brings an unacceptable false alert rate. Second, only referring to the measured value, which also depends on the current system state, regardless of other factors, always leads to an unsatisfactory result. However, ML-based IDS can solve the problem by carefully mining the data characteristics between normal flow and malicious flow [15].

Function Code Attack. By sending the malicious packets to control devices, function code attackers gain the physical or remote access to control devices to turn on/off them suddenly, which always causes serious consequences. A report from IBM in 2016 shows that attacks against ICS increased by 110 percent over the same period last year [16]. Security experts in IBM claimed that it is greatly influenced by an online open-source ICS function code attack framework.[3] Besides, even though the system runs normally, function code also changes irregularly, we cannot simply set up a blacklist to prevent function code attack. Machine learning algorithms can greatly help us identify normal operations by analyzing a large amount of control network data [14]. By comprehensively considering the various situations, we can mine the dynamic system state to determine whether commands are valid or not and launch a detector based on machine learning to improve the detection rate of function code attack [17].

Reconnaissance Attack. Reconnaissance attackers steal the communication data between work stations and control devices, or scan the control network for obtaining the information of control devices, preparing for further attacks. A well-known ICS ransomware *Out of Control* has been developed and used to launch reconnaissance attack [18]. Except injecting a large amount of packets into the control system, reconnaissance attack has no direct effect on physical devices or processes, but it would change the communication data flow trajectory and even leave some footprints on network trace, which makes it possible to be detected by ML-based IDS.

Contemporarily, there are other attacks such as DoS attack, device attack, replay attack, and so on. However, we focus on the three categories of attacks above since they are validated to be effectively detected by ML-based IDS with low cost and high accuracy [19].

2.3 GAN

We also take GAN as one of our tools to generate attack examples. The model can be divided into two modules: one is the discriminator used to distinguish between generated data and original instances, and the other is the generator which takes original instances as its input and outputs perturbation added to the original samples to yield new instances that are close to the data from the original class to human eyes [10]. The generator used in computer vision takes noise as input, which is different from ours, but the intention, generating confused samples approximating original samples to maximize the probability that D would make mistake, is identical. In this framework, we need to train discriminator D and generator G under the goal of minimizing their respective loss function at the same time. Two models compete mutually to learn and approximate the distribution of original instances. To put it simply, their final goal is to generate adversarial examples resembling initial samples and fooling D. However, as a consequence of the attack intention and additional constraints, the GAN above is not directly suitable for ICS. Therefore, we modify the architecture (including inputs, outputs, etc.) and loss function to yield adversarial attack samples which can evade IDS. More details will be demonstrated in Section 4.3.

2.4 Black-Box Attacks

Considering the prior knowledge available to the attackers, attacks are divided into white-box attacks and black-box attacks. For white-box attackers, they need to obtain the same training data as the detector and handle feature selection, detection algorithm and parameters. However, the current learning system usually does not allow white-box access against the model [20], owning to security. As a result, there is a great need for black-box analysis. Black-box attack strategies rely on the transferability of adversarial examples, namely, adversarial examples generated by a local learning model could confuse other models. We verify this by testing whether the generated adversarial examples, which can evade the substitute classifier detector, can still deceive our ensemble detector. For black-box attackers, they knows feature selection and meanings, control process, and the data that are equally distributed with the training data of the true detector, they does not know the detection algorithm and parameters. As black-box attacks are more common and challenging in reality, here we only consider this

2. Network data between control devices and control plants can be collected from remote I/O in practice.

3. https://github.com/enddo/smod

kind of attack due to space limitation. White-box attacks can be handled in a similar way.

3 Attack Process

In this part, we first define the threat model from the attackers' perspective, then we introduce how to realize penetration attack to ICS by traditional means and launch three types of attacks. In this way, we get the original attack samples which will be used to train the detector and taken as sources to generate adversarial examples.

3.1 Threat Model

Attackers' Background Knowledge. As black-box attackers, they cannot obtain the same training data as the detector, but they can intercept the system communication to get data that have the same distribution as training data. At the same time, we assume that the attacker has the prior knowledge about feature selection and the meaning of feature, without knowing the detection algorithm and parameters. Such an assumption is widely acceptable since previous work has demonstrated that an adversarial attacker can construct a reverse engineering learning problem to learn sufficient information about a machine learning model and then launches adversarial attacks [21]. Besides, by learning from Internet or getting information from insiders, attackers can also obtain the knowledge of the control process to accurately launch injection attacks, function code attacks and reconnaissance attacks [16].

Attackers' Goal. By establishing reliable and accurate intrusion detection systems (IDS), we suppose that the original attack samples can be detected by IDS. Attackers are going to modify an original example to evade the detection algorithm with high confidence, yet still maintaining the malicious payload. In other fields, like image domain, researchers focused on finding an example with smallest modification that can deceive the classifier. The generated adversarial example is almost the same as the original image to human eyes. However, in ICS, there is some extra consideration. In image processing, the input data are two-dimensional 512x512 vectors with continuous variables in each pixel. Every pixel can be freely modified. Yet in ICS, for ML-based IDS, the input data are network data. There are three main challenges in ICS: First, we need to solve the optimal problem with both continuous and discontinuous variables. Second, we need to ensure the generated data can be reversed to the network data, meaning that the features in each dimension should meet multiple restrictions to reserve reasonable. Most importantly, we should keep the malicious payload effective in our generated data. It is different from computer vision where the adversarial machine learning focuses on generating similar examples with benign ones. Here we are going to generate examples which are evasive but still have the valid attack effect against ICS.

Attackers' Capabilities. In this scenario, attackers have the ability to modify malicious network communication data. We suppose that attackers can alter all features, while satisfying the requirement that generated example can be reversed to a compliant network packet. Specifically, some certain features of the malicious samples such as packet length, timestamp and IP ports should appear in the correct manner. Otherwise, the malicious packets will not be parsed correctly by the targeted victims.

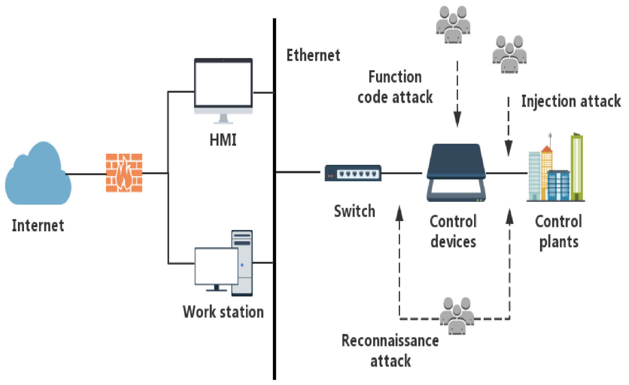


Fig. 2. Typical cyber attacks in ICS.

3.2 Invading Control Subnet

Usually, an external attacker can only connect to the public network, but not industrial control subnet. In this paper, We use SHODAN [22] to search for all devices connected to the Internet and further apply phishing, website XSS [23] and SQL injection [24] to enter the ICS enterprise network. Then we probe the active hosts, identify operation system and scan port in DMZ zone to perform vulnerability scanning and exploiting against open services. Finally, we control the enterprise network and server in DMZ. The most important is that we can utilize the communication channel with the underlying control device to attack the production process straight. We realize penetration attack to ICS by traditional means and verify the feasibility of implementing proprietary attack at the industrial control level.

3.3 Initial Attack Generation

In summary, the attacker has entered the control subnet and is prepared to launch attacks, which are the initial attack samples described in this paper, against communication between PLC and control object or between PLC and engineer station. These samples are malicious to ICS, but can be detected by IDS. This paper discusses three types of attacks on ICS, such as Fig. 2 shows, the detailed generation process is as follows:

Injection Attack. Injection attack is common in ICS. After possessing detailed workflow, communication protocol and control variables of ICS in previous penetration, attackers can inject malicious data into communication to lead ICS into a wrong state, even destroying physical devices. We pretend as the engineer station after invading the control subnet and write scripts, which are based on protocol analysis and python library(snap7), aimed at downloading the wrong field data to control devices. By maliciously changing the value of the specified register, the controller would send malicious commands to the actuator, causing serious consequences. In our experiment, we modify the data field, corresponding to the register value, to cause water level surging.

Function Code Attack. Function code attack means that attackers assign communication packets containing malicious function code to mislead control device into a wrong state as a result of an online open-source ICS function code attack framework.[4] In the platform, control station and control device communicates under S7 protocol which has a specific

4. https://github.com/enddo/smod

TABLE 1
Notation List

Notation	Definition
$\mathbf{x}^0$	the original malicious sample that can be detected by the substitute classifier
$\mathbf{x}^*$	the generated optimal sample that can't be detected by the substitute classifier
$\mathbf{x}_{mal}$	the generated malicious sample of each step in the iteration process
d	the euclidean distance between $\mathbf{x}^0$ and $\mathbf{x}_{mal}$ during the iteration process
$d^*_{\max}$	the maximum distance threshold between x^0 and x^*
$\hat{p}(x)$	the probability of sample x being classified as malicious by the substitute classifier
$\hat{f}$	the substitute classifier function
z	the initial malicious instances
x	the normal communication packet samples, fitting the distribution P_{data}
X	the batch of generated adversarial instances by GAN
y^*	the target class of generated sample, default to 0
$l_{\hat{f}}$	the cost function of $\hat{f}$

field representing function code. The attacker grabs communication packet, analyzes content and tampers with the function code to replay to PLC. The new function code is legal, but it appears unexpected at present. Attackers can use this type of attack to issue malicious control commands without background knowledge of the system, to shut down the control device. In our experiment, the Siemens series PLC can be suddenly turned on/off by changing the function code to 0x29.

Reconnaissance Attack. Reconnaissance attack means that an attacker maliciously steals control programs from industrial control devices. After connecting to the PLC, the attacker uploads and decompiles its internal program to get the MC7 code running inside the PLC, then the attacker transforms it into PLC control language STL and performs code analysis to gain the control program of the system.

After launching three types of attack above, we get the initial attack samples in the form of network packets, which is convenient for sampling and gathering. The samples are denoted as $\mathcal{D} = \{\mathbf{x}_i, y_i\}$, $\mathbf{x}_i$ is a multidimensional vector representing a communication message, y_i indicates whether $\mathbf{x}_i$ is malicious(1) or benign(0). Through the supervised training, the classifier assigns a predicted label to the sample according to the forecasting probability calculated for each sample. Once establishing reliable and accurate IDS, the original attack samples can be detected by the IDS classifier and the predicted label is 1. Therefore, we must modify original attack examples to evade the detector, yet still maintaining the same malicious payload. The next part will present a thorough description of this point.

4 Algorithm Design

This section elaborates on the constraint, provides two methods for generating adversarial examples against ML-based IDS for ICS and briefly introduces detection algorithm. Table 1 gives an exhaustively description of the variables and functions used in the following part.

4.1 Constraint

The goal of this paper is to generate effective stealthy attack on ICS by leveraging the flaw of ML algorithms. However, generating adversarial examples in ICS is a totally new field. In image processing, research on adversarial machine learning is focusing on how to find an example with the smallest modification that can evade the classifier. The generated adversarial example is almost the same as the original image to human eyes. The image can be seen as two-dimensional vectors such as 512x512 with continuous variables in each pixel that can be modified [20].

Yet in ICS, there are three main challenges: First, we need to solve the problems with continuous, discrete and fixed variables. For instance, some features ($\mathcal{O}1$), like source ip and destination ip, cannot be modified and some features ($\mathcal{O}2$) can only hold value in a specific interregional ($\mathcal{V}$), such as the function code, its value must be an integer and in the legal set. These restrictions can not only guarantee the newly generated samples be parsed correctly, but also preserve identical attack effect. To be more specific, ($\mathcal{O}1$) and ($\mathcal{O}2$) are different for different attacks, for example, in function code attacks, the data field can be modified in an approximate range, but the function code cannot be altered if we want to reserve the same attack effect, so the function code data filed is included in ($\mathcal{O}1$) for function code attacks, not for reconnaissance attacks. Second, we need to ensure the generated data can be reversed to the network data, that is, features in each dimension are subject to many restrictions, such as, function code and packet length must change within a valid set and take the appropriate value. The value is definitely not as simple as an integer, it must comply to the protocol format and correspond to the actual physical meaning. Finally, we should keep the malicious payload unchanged in our generated data. When considering the attack effect, three types of attack will reserve different malicious payloads. For injection attack, we inject malicious parameters or commands to mislead PLC running in error state, attack payload is the data field, therefore, we should keep the data field constant to ensure the same attack effect as initial injection attack. For function code attack, we send some commands to PLC to abruptly stop it at runtime, the malicious payload is function code. For reconnaissance attack, we mimic the normal system to upload the running program stored in PLC to obtain information about the production process, so the malicious payload is function code. To preserve the distinct attack effect, we carefully extract these constraints and eventually test the newly generated samples' attack effect (in Section 5.2).

4.2 Opt. Attack

Here we are going to generate examples that are evasive but still have the same attack effect on ICS. In order to solve this problem, Opt. attack strategy finds an optimal sample $\mathbf{x}^*$ to minimize the $\hat{p}(\mathbf{x}_{mal})$ with some specific additional constraints from ICS, $\mathbf{x}_{mal}$ represents the variable used in the iteration process, $\hat{p}(\mathbf{x}_{mal})$ denotes the probability of being classified as malicious by the substitute detector. The smaller $\hat{p}(\mathbf{x}_{mal})$ is, the more likely the sample is considered as benign. As Fig. 3 shows, for an initial attack sample, we iteratively modify it until bypassing the surrogated classifier. Our following experiments demonstrate that the

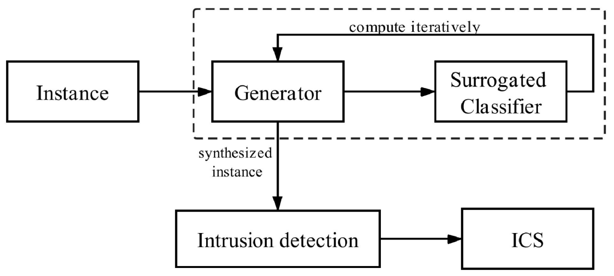


Fig. 3. Framework of Opt. attack.

calculated samples in this way can also evade the true IDS for ICS with high success rate. Then our problem can be formulated as follows:

$$\mathbf{x}^* = arg\ \min_{\mathbf{x}_{mal}} \hat{p}(\mathbf{x}_{mal}) \tag{1}$$

$$s.t.\ d < d^*_{\max}$$

$$\mathbf{x}^*[i] == \mathbf{x}^0[i], i \in \mathcal{O}1 \tag{2}$$

$$\|\mathbf{x}^*[i] - \mathbf{x}^0[i]\| \in \mathcal{V}, i \in \mathcal{O}2. \tag{3}$$

This problem can be regarded as finding an optimal solution to problem (1), d is the euclidean distance between the original sample $\mathbf{x}^0$ and the generated sample $\mathbf{x}_{mal}$ of each step in the iteration process, $d^*_{\max}$ denotes the maximum distance threshold between the optimal instance $\mathbf{x}^*$ and the initial one $\mathbf{x}^0$. Here $\mathbf{x}^0$ represents the original malicious sample, and its true label as well as the predicted label (by both the real classifier and surrogated classifier of IDS) is 1. The interregional $\mathcal{V}$ denotes the legal range or list for some special features. Take the function code for an example, it can only be set as an integer and in the legal function code list ([3,7...]). Even though 1 is an integer, it is not a valid function code. The generated adversarial example is $\mathbf{x}^*$, which is generated from $\mathbf{x}^0$ with the label changing from 1 to 0. The calculation process is summarized as Algorithm 1. $d^*_{\max}$ is selected through experiments depending on specific situations, the larger represents the higher tolerance of the deviation between the generated sample and the normal sample, the smaller means attackers hope that the adversarial example is as close as possible to the original attack sample. $d^*_{\max}$ is an experience value, we set $d^*_{\max} = 1.2$. However, distance is not the most important constraint in ICS, because retaining malicious payload is realized by subsequent constraints. The distance constraint here can be regarded as an additional requirement that ensures new sample would not be away from the initial sample.

Initially, we have some malicious instances, which can be detected by the detector. By utilizing the surrogated classifier and the predicted probability of the sample's being benign, we can iteratively optimize the generated sample and maximize the stealthy attack's success rate. We adopt constrained optimization by linear approximation (COBYLA) [25], which is a numerical optimization method for constrained problems. COBYLA is a nonlinear non-derivative optimization solution that uses a linear approximation method by iteratively approximating to linear programming problems. The algorithm relies on sequential trust region, using the linear interpolation of $n+1$ points in the variable space to linearly approach the constraint function. The original target and constraint function are used to evaluate the post-suspension to generate new data points in the optimization space to improve the approximate linear programming result for next iteration. When the result no longer changes, the step size gradually reduces to a sufficiently small value, and then the algorithm ends. In this way, we get the "optimal" result. Strictly speaking, the final result is only optimal under the approximate linear optimization problem.

Algorithm 1. Framework of Generating Adversarial Examples

```
Input:
  The initial malicious instance in ICS, $\mathbf{x}^0$;
  The surrogated classifier of the intrusion detection algorithm, $\hat{f}$;
  The maximum distance $d^*_{\max}$;
Output:
  New synthesized adversarial instance, $\mathbf{x}^*$;
1: $\mathbf{x}^0$ is detectable by $\hat{f}$: $\hat{f}(\mathbf{x}^0)_{y=1} \leftarrow 1$ ;
2: $\hat{p}(\mathbf{x}^0) \leftarrow$ the probability that $\mathbf{x}^0$ is predicted to be malicious
   by the substitute classifier;
3: function Opt. attack ($\mathbf{x}^0$, $\hat{f}$, $d^*_{\max}$)
4: $d \leftarrow 0$;
5: $\mathbf{x}_{mal} \leftarrow \mathbf{x}^0$;
6: while constraints == True and $d < d^*_{\max}$ do
7:     for $i = 0, 1, 2, \ldots\ldots$ do
8:             $\mathbf{x}_{mal_i} \leftarrow \min\{\hat{p}(\mathbf{x}_{mal})\}$ ;
9:             $d \leftarrow \mathbf{x}_{mal_i} - \mathbf{x}^0$;
10:            $\mathbf{x}_{mal} \leftarrow \mathbf{x}_{mal_i}$;
11:    end for
12: end while
    $\mathbf{x}^* \leftarrow \mathbf{x}_{mal}$
13: return$\mathbf{x}^*$;
```

To sum up, the novelties of Opt. attack are: First, it is the first approach on cyber stealthy attack against the ML-based IDS in ICS. Second, our carefully designed attacks can not only bypass the ML-based IDS but also stay malicious to ICS by preserving payload. The number of iterations of gradient descent methods increases with data volume and the algorithm of Opt. attack runs for every malicious example, which is extremely resource-intensive. The memory required is quadratic in the number of variables. Thus, a more efficient algorithm is in need.

4.3 GAN Attack

When the data scale of initial malicious examples is small, the pt. attack can handle the problem well. Yet the complexity of Opt. attack increases with the number of inputs. In order to improve calculation efficiency and save memory space, we propose a new method to generate adversarial examples in ICS, named GAN attack. In addition, Opt. attack always generates the samples in the fixed manner. Once some are predominated by the detector, the others

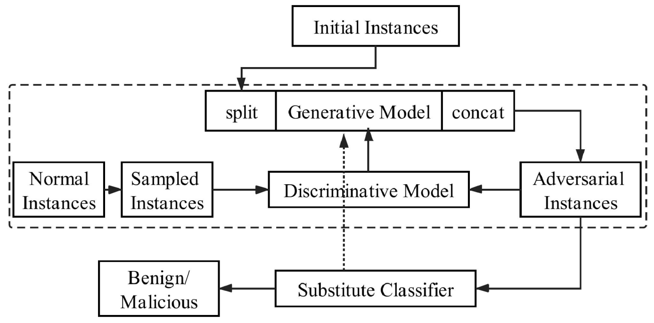


Fig. 4. Framework of GAN attack.

will be recognized as malicious, however GAN aims at finding the blind zone of neural network in a flexible way, samples are so complicated that they would not be detected easily which is confirmed in Section 5.8. Similar to Opt. attack, we put forward some additional considerations to the original GAN model [10]. This paper redesigns the loss function of G and D in the network to achieve generating adversarial examples against ICS.

Yet the original GAN model cannot be used directly in our problem for that we are not simply trying to imitate the input of D as original GAN in computer vision does. We add two additional requirements to the original GAN model: keeping malicious payload and evading IDS, which is different from GAN in [20]. As a black box attacker, we choose MLP as the surrogated machine learning classifier.

As shown in Fig. 4, we add the substitute intrusion detector to our model. The framework is similar with AdvGAN [26], we all take a network as adds-on and utilize its loss to generate higher quality adversarial examples. However, there still exist some differences. First, Xiao *et al.* [26] propose AdvGAN to generate more perceptually realistic adversarial instances, focusing on continuous variable space in image domain, while we aim at generating stealthy attack data packets against intrusion detector in ICS, there are many domain barriers resulting in introducing many constraints and difficulties, such as continuous data or not, changeable features and numerical range. Second, we haven't considered using constant to control the relative importance of each objective, like α, β used in AdvGAN, which is for that keeping the malicious payload and evading the intrusion detector are viewed as equally important in our situation. At the same time, omitting parameter trade off saves much time. Third, even though the loss terms are in the similar form, $E_z||G(z) - z||$ and $\mathcal{L}_{\text{hinge}} = \mathbb{E}_x \max(0, \|\mathcal{G}(x)\|_2 - c)$. However, the actual calculation is different. First, we detach the malicious payload from the integral sample to keep the malicious payload fixed. Then, we automatically optimize the changeable features based on GAN under the complex constraints, i.e., some features can only take values in a specific set, some features can only be discontinuous integers, etc. Due to the characteristic of protocol data, each field are treated differently. The latter just limits the euclidean distance between the original and generated image vector in pixel space. And each pixel is regarded as equal.

When training G, the algorithm selects the continuous variable except the malicious payload as inputs in each batch (for the function code attack, the payload is the function code field; For the injection attack, the payload is the injection data field; For reconnaissance attack, the payload is the function code field that guarantees eavesdropping function running smoothly). Under the influence of special protocol data type, additional processing is required, such as dividing the initial attack sample into two regions, one is the continuous changeable variable, and the other is not changeable, such as fixed attack payload. Here, we only update the continuous variable using G, but when designing the loss function (training target), we concat the continuous variable and the fixed variable (attack load), then input to the substitute detector and D to get the probability of evading the detector and the loss of D. Then we continue to train D and G based on the feedback. Here the training of D does not change much, the input is normal data sample (the communication data without attack) and malicious data, and the training goal is to distinguish normal data from the adversarial example output by G.

Therefore, this framework is similar to the minimax game of the two participants, and we can eventually find the unique equilibrium solution. The generated samples have the same distribution with the training sample, yet maintaining the fixed attack payload and evading the physical system's intrusion detection algorithms.

The loss function of D

$$-E_{x\sim p_{data}} logD(x) - E_z log(1 - D(G(z))). \tag{4}$$

The loss function of G

$$-E_z log(D(G(z)) + E_z l_{\hat{f}}(G(z), y^*) + E_z||G(z) - z||. \tag{5}$$

Algorithm 2. Framework of Generating Adversarial Examples With Generative Adversarial Networks

```
Input:
    The initial malicious instances, $z$;
    The surrogated classifier of the intrusion detection algorithm, $\hat{f}$, its cost function $l_{\hat{f}}$
    The normal data are of distribution $x \sim P_{data}$;
Output:
    Batch generated adversarial instances, $X$;
    function GAN. Attack ($z$, $\hat{f}$, $P_{data}(x)$)
    for number of training epochs do
        for each mini batch do
            Sample minibatch of m initial malicious samples $[z^{(1)}, \ldots, z^{(m)}]$
            Sample minibatch of m normal samples $[x^{(1)}, \ldots, x^{(m)}]$
            Update the discriminator by gradient-based minimization of the loss function:
                $-E_{x\sim P_{data}} logD(x) - E_z log(1 - D(G(z)))$
            Freeze the discriminator;
            Update the generator by gradient-based minimization of the loss function:
                $-E_z log(D(G(z)) + E_z l_{\hat{f}}(G(z), y^*) + E_z||G(z) - z||$
        end for
    end for
    return $X$
```

Notice that here x, z individually refers to the normal communication packet samples and the initial malicious instances, and y^* denotes the target class of generated sample, default to 0, as for the reason that we want to generate the stealthy attack samples, which can evade the detector and be classified as normal. For the discriminator, $E_{x\sim P_{data}} logD(x)$ helps D find out whether the data are sampled from $x \sim P_{data}$. $E_z log(1 - D(G(z)))$ gets larger value when the generated samples cannot deceive D, namely, when the similarity between the generated attack data z and original normal data x is smaller, this term gets a larger value. The generated samples z will be more like normal data x when we minimize the loss of D. For the generator, $-E_z log(D(G(z))$ ensures that the generated samples are similar to the normal data. $E_z l_{\hat{f}}((G(z), y^*)$ denotes the loss that the generated adversarial attack data z are classified as the target class(benign) by the substitute classifier, namely z evading the intrusion detection algorithm, while $E_z||G(z) - z||$ restricts the attack payload fixed and the changeable features in an appropriate range or set, not deviating too much. We minimize the loss of G to generate the adversarial result which can evade the detector, keep the same distribution with normal data x and preserve the malicious payload fixed. GAN generates adversarial examples by batches, which greatly decreases time cost.

Since the loss function of G and D is not a simple positive-negative relationship, and the pure min-max problem may encounter some problems, so we directly perform gradient reduction training and iterative calculation on the two models according to the loss function. Note that considering the simplicity of the algorithm, we have not differentiated whether the variable can be changed or not in the process. As mentioned before, we need to distinguish the features in actual application to ensure the retention of malicious payload. Specifically, we need to make sure attack payload not included in changeable data to G. However, when the input is for the intrusion detection classifier and discriminator, it's necessary to concatenate the mutated features and the stable attack payload to get the correct feedback.

GAN attack is trained for all attack samples at one time. If there are 10,000 initial attack samples, Opt. algorithm needs to run 10,000 times, but GAN, which generates adversarial examples in batches, only needs one calculation to generate the corresponding adversarial examples of the 10,000 initial attacks. The time cost saved is related to the number of initial attack samples, taking 1,000 samples as example, under MacBook Pro 8G, Python 2.7, Scikit-learn 0.21, cobyla needs up to 100,000 iterations, and for GAN the epoch is 50, the batch is 32, Opt. attack needs to run 600s, and GAN only needs 60s. The real-time performance is better (run time refers to the conversion of all attack samples from the initial sample to the adversarial example). The specific algorithm performance is shown in the next chapter.

4.4 Detection Algorithm and Surrogated Classifier

To verify that our attack generation algorithm work against most common ML-based detectors used in ICS, we build a reliable detection system with high accuracy. As there is no mature product to perform ML-based intrusion detection for ICS, we adopt state-of-the-art models in literature [27]: Parse ICS network packets into CSV files and apply supervised learning methods to the labeled data. In fact, the first part can be difficult in ICS, as most ICS protocols are private and unknown. Moreover, here we simply implement detection for a single packet, as most industrial control objects are in smooth operation, which makes single-packet detection possible. We construct our intrusion detector by one ensemble model consisting of five kinds of decision trees. We apply four basic classifiers (Random forest, Extra Trees, GBDT and AdaBoost) to the data. Then we use the outputs of them as new features to XGBoost to make the final decision. We also compare our detector's performance with other ML-based detectors in the former published papers [15], [17], [28] on the same dataset to show our detector is more powerful than other detectors. The specific detection results are elaborated in the next section.

In order to make our attack algorithm more general, we consider black-box attacks where the attackers cannot obtain the model structure and parameters of IDS. However, the adversarial examples have transferability, i.e., if they can evade the surrogated classifier, they are still stealthy to the actual detector. Even if both models have different architectures, as long as they are trained to perform the same task, the generated adversarial example, fooling one, will also deceive the other with a high probability. Therefore, we esatblish the substitute local classifier-MLP for both Opt. attack and GAN attack and prove that the generated samples that can evade the substitute classifier-MLP also deceive our detection model.

5 Evaluation

We evaluate the effectiveness of the proposed attack generation methods by conducting experiments on a semi-physical testbed.

5.1 Platform Setup

In order to demonstrate that our generated adversarial examples can not only evade a high-accuracy ML-based intrusion detection algorithms, but also remain malicious, we construct a typical ICS environment to test the effectiveness of our algorithm. In fact, there are also some other open datasets online, yet their physical model is inaccessible. Thus the data collected from the physical platform is more suitable for us to evaluate our algorithms.

Our platform consists of a real controller, HMI, workstation and simulated control object (water tank). By using OPC, the simulated object can communicate normally with PLC. There are Siemens 6ES7-317-2AK14-0AB0 317 PLC, TP-Link TL-SF2005 switch and ThinkPad as the workstation in our platform. As shown in Fig. 5, we use a switch to connect the work station, PLC and simulated water tank model. The network data, under normal or attack circumstances, are stored for future research.

The platform controls the simulated quadruple water tanks' liquid level with two pumps. Liquid level is a typical control object in ICS [29], wildly used in oil refining and wastewater treatment. Once attacked, the consequence will be unimaginable. For example, function code attack may bring the whole liquid level control system break down and even destroy one wastewater factory. Liquid level control in water tanks is also a widely studied problem in process control. There are two inputs in this model (voltage applied to

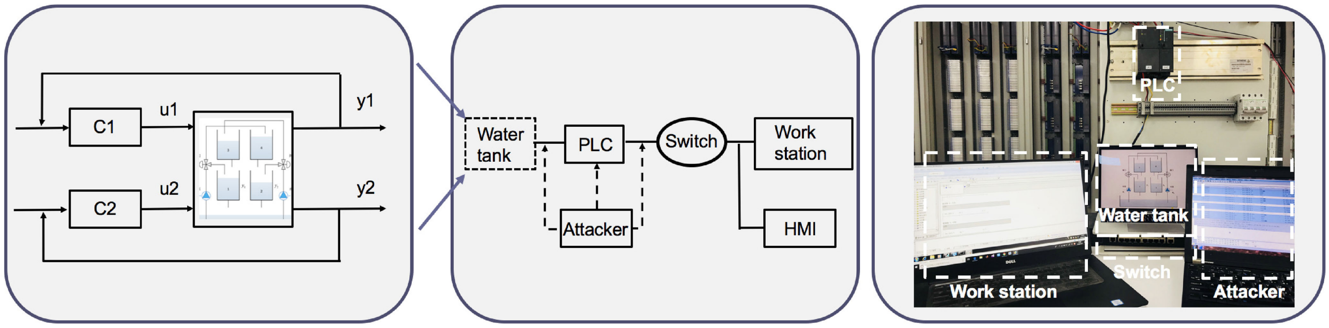


Fig. 5. Simulated quadruple water tank platform.

the pump: input1, input2), two outputs (two liquid levels: output1, output2) and two state variables (the controllers' states: stat1, stat2). We use Siemens S7-300 PLC as the controller to implement the distributed PI control algorithm based on STL programming language. Detailed platform networking and components can be seen in Fig. 5.

5.2 Datasets

HMI, PLC, work station and water tank communicate with each other by using S7comm protocol in this platform. It is a protocol in ICS based on TCP/IP. In order to show that our attacks are also valid for other industrial protocols, we conduct experiments on datasets for Modbus [5] and IEC 104 protocol. Here we omit the description of these data due to the space limit.

The raw data can be split into different fields. The splitting process is called feature extraction. Yang *et al.* [30] proposed that IPs, MAC addresses, ports, and control fields should be chosen as protocol features. Beaver *et al.* [17] focused on the specific values associated with their RTU data when designing a feature set.

By carefully analyzing the different fields in raw network data, we select some important ones as our features, which vary in different packets and can provide us with significant information. We select some specific features for our communication protocol. For example, Rosctr is one specific feature in S7comm [31], which can be regarded as a subfunction code. All features are listed in Table 2.

As Fig. 6 shows, we test these features' importances in later experiments to make sure all features are necessary. IPs seem to be not so important for that there are only two IPs communicating by S7comm in our experiment. If there are other devices involved in, the IP feature will also be important, thus we don't delete them here.

We obtain vectors in 12 dimensions (in which there are 6 variables in data field: input1, input2, output1, output2, stat1, stat2). Besides, we gather the data when the water tank system is operating in a stable state. Also, we collect the network data in different time periods to make the data more representative. We test three attacks on this platform. We achieve attacks by using open source tool like snap7.[6] There are 79,080 instances together, in which there are 28,943 normal ones, 23,353 attack 1, 8,303 attack 2 and 18,481 attack 3 instances. We generate new adversarial attacks for those different original attacks.

Normal State. We label the packets in normal state as class 0. In this situation, the system is in steady state.

Injection Attack. Injection attacks are class 1 (attack 1) here. Attackers inject malicious data or commands into PLC to make it run in error state. The malicious payload here is the data field, corresponding to the register value. Besides, to randomize the attack, we inject a series of random data to prevent overfitting.

Function Code Attack. We label packets under Function code attacks as class 2 (attack 2). Here we send some packets to PLC to make it stop suddenly. The malicious payload here reflected in the network packet is the function code. The adversarial examples contain malicious function. However, to prevent the detection model overfitting, we also put the PLC CPU to 'stop' state when it runs in normal operation. This makes it hard to be detected by rule-based IDS, but can be effectively solved by ML-based IDS.

Reconnaissance Attack. Reconnaissance attacks are labeled as class 3 (attack 3). We simulated an attacker to upload programs running in PLC to gain the information about the production process by injecting a great deal of normal data packets, so the payload is function code.

The box-line diagram Fig. 7 describes the feature distribution difference between original samples and generated samples for reconnaissance attack. It can be seen that most features are similar, and the attack payload-function code remain fixed as we expect.

With the initial attack samples, powerful detector and attack generation algorithm, we still need to verify the attack

5. https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

6. Step7 Open Source Ethernet Communication Suite

TABLE 2
Feature List

Features	Type	Values	Changable
source ip	network	0, 1, . . .	N
destination ip	network	0, 1, . . .	N
time delta	network	0.0006	N
function	packet payload	3, 7, . . .	N
rosctr	packet payload	5, 7,. . .	N
length	network	79, 87, . . .	N
input1	data payload	10,. . .	Y
input2	data payload	10,. . .	Y
output1	data payload	2,. . .	Y
output2	data payload	2,. . .	Y
stat1	data payload	6,. . .	Y
stat2	data payload	6,. . .	Y

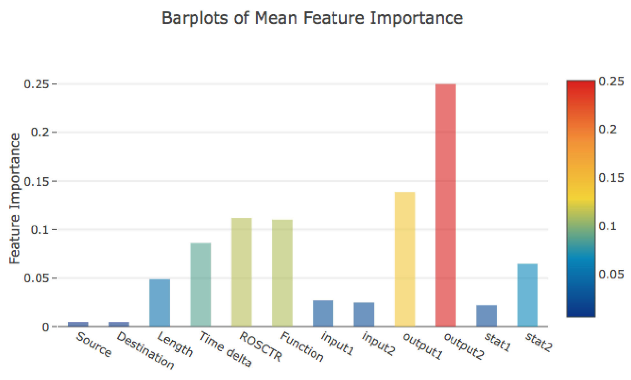


Fig. 6. Feature importance.

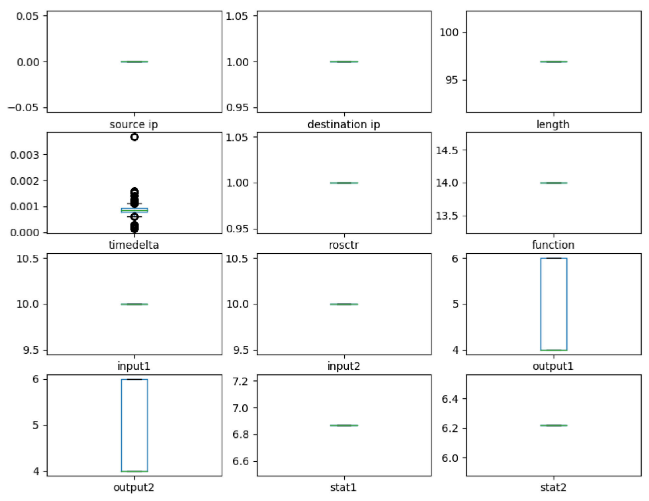


Fig. 7. Value change of each dimension in attack 3 sample.

effect of generated samples. We inverse the newly generated samples to obtain the original network communication data by reversing one-hot and min-max scaling, which is used to be replayed to the platform to evaluate whether the system status changes as expected. In detail, for injection attack, “output1” and “output2” suddenly increase, which means the liquid level surges. For function code attack, the results are evaluated by checking whether PLC is remotely turned off by maliciously generated packets. For reconnaissance attack, this article assesses the attack effect by judging whether an attacker can steal packets from the control network stream. Table 3 gives a more comprehensive comparison analysis. Finally, the number of adversarial examples generated by Opt. attack algorithm is 26,097 for each type of attack, GAN attack produces totally 26,097 adversarial examples. These 104,388 samples can evade IDS with a high probability and keep malicious attack effect on ICS.

5.3 Detector Design

We use Pyshark [7] as our packet analyzer. After applying min-max transform and one-hot encoder to these features, we get data for further processing.

Next, we establish our ensemble (stacking) model. Stacking models use the predictions of a few basic classifiers as the inputs of second-level classifier. In the first period, we apply four typical decision tree models to our data: Random Forest, Extra Trees, Adaboost and GBDT by k-fold cross validation. In the second period, we use the predicted labels from the first period as new features to get the predictions. We use XGBoost as the second-level classifier to learn from the new features and make final decisions. The result shows that the stacking model achieves high performance in distinguishing the attack data from the benign one. From Table 4, we can see that the average F1-score is 0.99.

In addition, the results of different detection metrics are shown in Table 4. Since samples for each kind of attack are balanced, the precision (positive predictive value) and recall (sensitivity) reflect effectiveness of the model.

We choose some detecting algorithms from the prior published papers. Jiang *et al.* [28] used OCSVM to detect outliers in ICS. Beaver *et al.* [17] used Naive Bayes to test the detection performance of machine learning methods for SCADA systems. Linda *et al.* [15] proposed to use neural network in abnormality detection of ICS. It is obvious that the precision, recall and F1-score of ensemble model are higher than other machine learning models. Different models all perform better in attack 2 and attack 3 than attack 1. It may result from that less change is made on attack 1. The greater similarity to the initial attack samples makes it more likely to be detected by the substitute detector. What’s more, we can see that stacking model performs better than basic model such as random forest in Table 5.

5.4 Generating Adversarial Examples by Opt. Attack

To evade the ensemble model detector, we generate adversarial examples using Opt. attack based on the initial attack samples. We need to find a substitute classifier to replace the system classifier (ensemble model). We select the typical neural network (multiple layers perception) as our substitute classifier. To train the substitute classifier, we select the data under the same distribution with the training data of the actual IDS. What’s more, although we have assumed that black-box attackers know the feature selection and feature meaning, we can‘t choose exactly the same features with the detector. After the training process, the substitute classifier presents the average precision for each class 0.9844, and the average f1-score 0.9838.

The constraints change with the initial attack class. For attack 2 and attack 3, the adversaries can modify four features in the data field to evade IDS: output1, output2, stat1, stat2. That is, for reconnaissance and function code attack, only the data field can be changed. For attack 1, the adversaries can only modify two state variables, as the output data is also a part of the malicious payload: stat1, stat2. That is, for injection attack, the outputs are also the attacks’ targets which should not be changed. We take COBYLA (for Constrained Optimization by Linear Approximations) [32] as the optimization solver here.

By the reverse transforming of numerical features (reverse one-hot and min-max scaler), we can get the raw network trace. Then we test those generated examples on our water tank platform to evaluate the results. We evaluate the attack effect by checking whether the system state changes as expected. For example, injection attacks bring out a sudden surge in ‘Output_1’ and ‘Output_2’. Then we test our

7. Python wrapper for tshark, allowing python packet parsing using wireshark dissectors. https://github.com/KimiNewt/pyshark

TABLE 3
Comprehensive Comparison Among Three Attacks

	Attack payload	Changeable feature	Attack definition	Attack effect
Injection attack	Data	Stat1, 2	Download fake data to the controlled devices to maliciously modify the specific register	Output1, 2 surges, water level increases sharply
Function code attack	Function code	Output1, 2 Stat1, 2	Force PLC suddenly shutdown, out of normal process	Turn off PLC suddenly without a gradual process
Reconnaissance attack	Function code	Output1, 2 Stat1, 2	Inject a quantity of communication packets to steal control code	Steal communication packets from control network stream

TABLE 4
Detection Results

Categories	precision	recall	f1-score	support
1	0.9882	0.9859	0.9871	6116
2	1.0000	1.0000	1.0000	7714
3	0.9698	0.9746	0.9722	2835
avg	0.9940	0.9939	0.9939	26097

TABLE 5
Detection Comparisons Between Different Models

Models	Categories	precision	recall	f1-score
NN [15]	1	0.8110	0.7386	0.7732
	2	0.9992	1.0000	0.9996
	3	0.5742	0.8060	0.6706
	avg / total	0.9001	0.8827	0.8879
NB [17]	1	0.9886	0.3184	0.4817
	2	0.9674	0.9998	0.9833
	3	0.5095	0.9420	0.6613
	avg / total	0.8692	0.8105	0.7888
OCSVM [28]	1	0.8510	0.8896	0.8699
	2	0.9000	0.8998	0.8999
	3	0.8796	0.8013	0.8386
	avg / total	0.8853	0.8850	0.8851
RF	1	0.9875	0.9640	0.9756
	2	1.0000	0.9998	0.9999
	3	0.9331	0.9764	0.9543
	avg / total	0.9886	0.9882	0.9883

generated examples by checking whether the variables change as expected. For function code attack, we evaluate the result by checking whether PLC is shut down by malicious generated packets. For reconnaissance attack, we evaluate whether the attacker can steal packets from the control network trace.

Results are shown in Fig. 8. As the malicious payload being kept, they all gain practical effects on the physical platform, but some of them fail to evade the true IDS (different from the substitute classifier). We calculate the attack success rate through dividing the number of original attack samples by the number of samples that can evade the actual IDS. The success rate of injection attack is 0.20958, function code attack is 0.81201 and reconnaissance attack is 1. Especially for injection attack, only a small part of the generated examples can evade IDS successfully, resulting from the small controllable space. Thus for attackers in ICS, it's harder to launch stealthy injection attack.

5.5 Generating Adversarial Examples by GAN Attack

In this section, we also evaluate GAN attack on our testbed. As mentioned before, we redesign the loss of the generator to help the adversarial examples evade IDS. Here we take 50 epochs to train the adversarial network, with batch_size being 32.

The generator's and discriminator's losses during each training batch can be depicted as follows:

We take the part of unchangeable variables (x^s)in initial adversarial examples as a stable part in the training process. What we train is the field (x^c) that we can modify. We use attack 3 as the initial malicious samples. Similarly, we make the min-max scaler ,to the generated data. By transforming the generated data into the specific form, we inverse them to the initial packets and launch the attacks.

From Fig. 9, we can find that the generative loss slightly decreases, proving that the generated instances are not too far from the initial ones. The loss of G keeps increasing during training, showing that D and the substitute detector cannot distinguish the attack samples from the normal ones. Here we test on reconnaissance attack. Similar to Opt. attack, we test the evasion rate of GAN attack against IDS, which reaches 100 percent. The calculation time is greatly reduced. After careful calculation, we find that GAN attack processes the data per 10,000 pieces nearly 10 times faster than Opt. attack. Thus, GAN attack performs better when the data size is large.

Thus, GAN attack performs better when the data size is large. However, as this attack is not flexible enough, especially in ICS where there are many unchangeable features, when the data size is not too large, Opt. attack is the better choice.

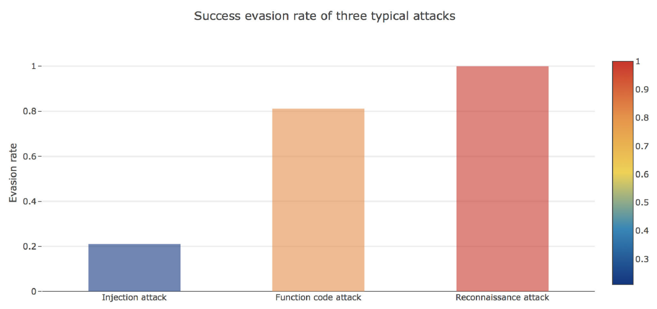


Fig. 8. Success evasion rate of three attacks.

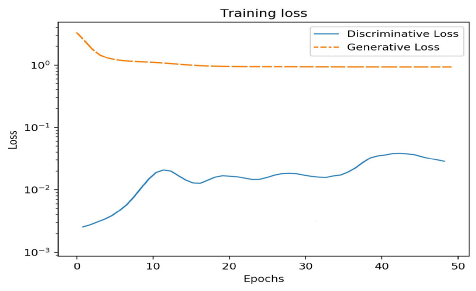


Fig. 9. Training loss of GAN attack.

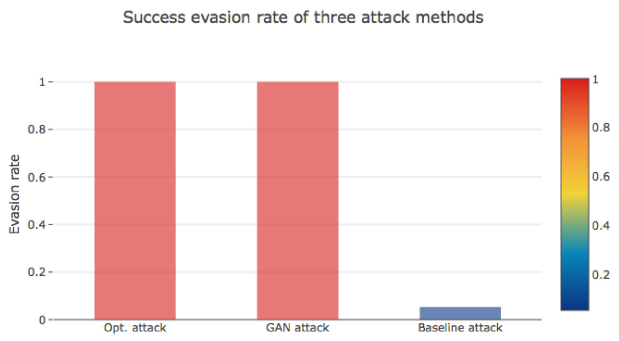


Fig. 10. Evasion rates of different attacks.

5.6 Comparing With Baseline Attack

To verify that our methods are effective against an ML-based IDS for ICS, not resulting from overfitting or other factors, we test a baseline attack on the water tank model. Taking attack 3 (reconnaissance attack) as an example, here we analyze the generated adversarial examples first. We select the changeable field and calculate the generated data's mean value. Then we generate random values based on the mean value within a small range (for output1 and output2 in [0, 1], stat1 and stat2 in [0, 4]). Concatenating the fixed values and random values, we test the final data on our intrusion detector and plot the result. As Fig. 10 shows, we can find that the attacks only have much lower probability to evade the classifier. We can find that the attacks only have much lower probability to evade the classifier. To reduce the bad effects caused by overfitting, we can add adversarial training to the initial training process.

We also test our methods on other industrial protocols, such as Modbus and IEC104 which are commonly used in industrial environment [33]. The Modbus data are collected in Morris's testbed. We have not established a semi-physical gas pipeline system and verified the attack effect of open attack data and generated data, so we just test our algorithms at the data level against Modbus protocol. We set up an IDS by neural network using the given extracted features and detection method in [1]. There are several different kinds of attacks in this dataset. We take the open attack data as the original data and generate adversarial attack data. Our generated examples for different attacks have different performance in evading the intrusion detector, similar to the results in S7comm experiments. Reconnaissance attacks are always easier to generate new mutations to evade detection. IEC 104 data is collected from a honeypot[8] established at our lab. Common honeypots are not equipped with the function of running programs. Usually, the honeypot PLC can only record the injected content, so the attack effect depends on the specific injected command. In our situation, the attacker injects the command to change the program of PLC output port, the attack effect is determined on whether the output port changes. More detail is presented in Table 6. Experiments in different protocols show that our methods are valid in different ICS environments, regardless of the protocols. Here, the evasion rate in all three scenarios is up to 100 percent.

8. https://github.com/mushorg/conpot

TABLE 6
Tests on Various Industrial Protocols

	scenarios	attack categories	detection methods	verified in practice
S7comm	water tank model	injection, device, reconnaissance	ensemble methods	Y
Modbus	gas pipeline system	command and response, DoS, reconnaissance	neural network	N
IEC104	plc honeypot	injection	Naive Bayes	Y

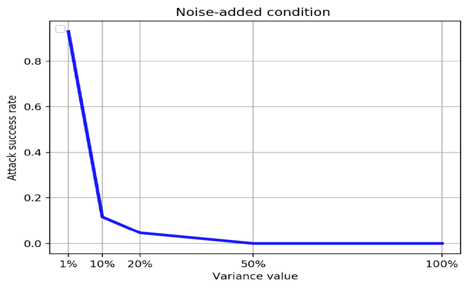


Fig. 11. Evasion rate over different data noise.

5.7 Adding Noise to Data Obtained by Adversaries

As we have stated before, the black-box attackers don't know the model structure or detailed model parameters. Yet adversaries need training data that has the same distribution with the true detector. However, this is still difficult for adversaries in the real world. So, we make some experiments to prove whether optimal attack and GAN attack can get the comparable performance when the training data of the substitute detector deviates too much.

By gradually increasing the variance of the added zero-mean Gaussian noise, we are able to calculate the corresponding attack success rate. To be more concrete, first, we add random noise to the continuous field of original data. Then attackers train the local surrogated classifier based on noise-added data. To sum up, attackers generate the stealthy adversarial attack samples from noise-added original examples.

We add noise to input1, input2, output1, output2, stat1 and stat2. The normal values are around 10, 10, 2, 2, 10 and 10. As Fig. 11 shows, the attack success rate obviously decreases when larger noise is added. In normal state, our platform's measurement values are around 10 (two orders of magnitude). We have a common sense that the model, trained on data with different distribution, will have a big performance gap. And the result also confirms this, when larger noise is added to the training data of the substitute detector, the substitute classifier has a huge gap with the true detector, namely, the generated stealthy adversarial samples against this

TABLE 7
Defensive Effect of Adversarial Training

	Opt.in	Opt.fu	Opt.re	GAN.in	GAN.fu	GAN.re
0.1 Opt	1.00	1.00	1.00	0.00	0.00	0.00
0.1 GAN	1.00	1.00	0.00	1.00	1.00	1.00
0.1 GAN & 0.1 Opt	1.00	1.00	1.00	1.00	1.00	1.00

substitute classifier cannot evade the true detector. Specifically, when variance value reaches 2.5 percent of system values, the evasion success rate is around 80 percent. When variance value reaches 10 percent of system values, the evasion success rate is lower than 10 percent.

5.8 Defending Against Adversarial Attacks

There are several methods to defend against adversarial attacks, such as adversarial training [34], [35], [36], gradient masking [37], and detecting adversarial examples using external network as add-ons [38]. In this paper, we choose adversarial training that aims at building a robust classifier which includes adversarial information in the training phase to alleviate the problem we encounter.

The results are shown in Table 7, we individually append 10 percent samples generated by Opt. attack, GAN attack and both attacks to retrain the substitute detector, aimed at improving its robustness. As the result shows, only adding 10 percent samples generated by Opt. attack to the train data cannot detect any type of attack samples generated by GAN, while adding 10 percent samples generated by GAN to the train data can successfully detect function code and injection attack samples generated by Opt. attack method. Based on this finding, we have a bold conjecture that GAN attack can generate more diverse adversarial examples than optimal solution attack. As for the reason, Opt. attack always yields samples in the fixed and single means. On the contrary, GAN can generate adversarial examples in a flexible way by actively exploring the blind zone of model, the generated samples are harder to be detected. Apart from this, we can also find that reconnaissance attack is easier to generate new mutations to evade detection. We can get approximately 100 percent true positive detection rate, once 10 percent attack samples generated by both two methods are involved into the train data of the detector.

Due to not modifying the network architecture, adversarial training is always treated as the first line of defense [39]. Apparently, brute-force adversarial training improves the robustness of ML-based detector against these examples. However, it always requires strong attacks to cover all types of adversarial examples and retraining the model with more training data, which cost a lot of time. Despite these requirements can be satisfied, the classifier usually generalizes poorly [40]. We will confirm and improve this in future work.

Moreover, many popular defend methods can be explored in the future. At data level, applying some data augmentation methods is common to defend adversarial attack in image domain, such as data compression [41] and data randomization [42]. If we want to incorporate the defend methods at data level into our design, we only need to deal with the training data in advance. At model level, many plausible defensive methods can be utilized, mainly classified into two category. One is modifying the network, for example, defensive distillation [43] and gradient masking [37]. The first transfers knowledge of a complex network to a smaller network, the second penalizes the degree of variation resulting in the output with respect to the change in the input, both aimed at improving the robustness of the model, but the methods are absolutely different. As for these defense methods, we can modify the network to promote the robustness against GAN attacks. The other is adding network add-ons, Akhtar *et al.* [44] propose to append extra 'pre-input' layers to the targeted network and train them to rectify a perturbed image so that the classifier's prediction becomes the same as its prediction on the clean version of the same image. As for this defense, we can modify the network of the substitute classifier or append network add-ons to the substitute classifier to promote the robustness against GAN attacks.

6 Related Works

In this section, we summarize the detection methods in ICS and present situation of adversarial ML in Section 6.2.

6.1 Intrusion Detection Algorithms in ICS

It is proved that some ML-based real-time intrusion detection approaches are of great usage in ICS. Most of research on machine-learning based IDS algorithms in ICS focus on supervised learning. Beaver *et al.* [17] evaluated a set of supervised machine learning algorithms' abilities in a gas pipeline system. The labeled data used in this experiment was developed by the Mississippi State University's Critical Infrastructure Protection Center. Naïve bayes, random forests, OneR, J48, NNge and SVM are proved capable to generalize the data as binary classifiers with high precision&recall rate.

Almalawi *et al.* [15] proposed a framework of unsupervised intrusion detection for ICS. The proposed approach includes two steps: learning the most representative data sets, using the established data sets to build a decision-making model consisting of several kinds of supervised classifiers. Experimental results from the simulated testbed prove that the accuracy of unsupervised intrusion detection algorithms for ICS improves effectively.

Caselli *et al.* [45] considered sequence attack detection problem on ICS and proposes a different solution. They extract the semantic meaning of ICS network events and behaviors over time. Linda *et al.* [15] identified a specific window based attribute extraction technique. Specifically, it is derived from real network data in an existing critical infrastructure. Jiang *et al.* [28] developed an intrusion detection system which provides one kind of computerized tool based on OCSVM for protection of ICS security.

6.2 Adversarial ML

Experiments and research show that ML models themselves can be the attacker's target. Adversaries may have access to the training data, model type, model parameters or model outputs. The attacks in training phase are called poisoning attacks, and the ones in testing/inference phase are evasion attacks. This paper focuses on the design of evasion attacks, which means that adversaries can add perturbation to original inputs to mislead the model. The problem has been discussed in different scenarios:

- *General Classification*
  Barreno *et al.* [46] proposed a taxonomy of different categories of attacks on machine learning models, and various defenses methods against evasion attacks. Researchers discuss application-specific factors limiting an attackers' capabilities, model an adversary's capabilities, point out the limits of an attacker's knowledge about the machine learning model, feature space, training and input data, explore algorithms' vulnerabilities and discuss countermeasures in the end.
- *Image classification & Segmentation & Object detection*
  Adversarial attacks against machine learning models in computer vision has gained remarkable progress. Bose *et al.* [47] evaluated attacks on a trained Faster R-CNN face detector and reduce detection rate to 0.5 percent.
- *Spam filter/ Malware detection*
  Dalvi *et al.* [48] applied the problem to spam detection in a variety of scenarios, and extend the naive Bayes classifier by taking the adversary's optimal strategy into account. Nelson *et al.* [49] showed how an adversary can exploit SpamBayes spam filter which uses statistical machine learning — even if only 1 percent of the training data is accessible. Osadchy *et al.* [50] proposed a new Completely Automated Public Turing tests to tell Computers and Humans Apart (CAPTCHA) scheme based on adversarial examples.
- *Generative model*
  Kos *et al.* [51] explored methods of generating adversarial examples by generative models such as the variational autoencoder (VAE) and the VAE-GAN. Different from previous work which focuses on the application of adversarial examples to classification tasks, Kos presents three classes of attacks on the specific generative models.

Adversarial training provides researchers a new way of improving the machine learning models' robustness. Papernot *et al.* [52] investigated the use of distillation as a defense method against adversarial perturbations. Kurakin *et al.* [53] offered an approach on how to scale adversarial training to large models and datasets, and observe that adversarial training improves robustness especially to single-step attack methods. Dritsoula *et al.* [54] proved that ML-based classification can be a game-theoretic problem. However, these related works are not proposed towards the scenario of ICS and some specific constraints are not considered. Instead, our attacks are well customized for ICS and tested in the semi-physical ICS environment.

7 Conclusion

We have investigated the possibility of launching stealthy attacks that can evade ML-based IDS in ICS. Machine learning classifier is vulnerable to adversarial attack, however, generating adversarial examples in ICS is totally different from image domain. Thus, we must comply with protocol format to ensure samples being parsed correctly and set additional constraints to maintain the same attack effect to ICS and evade IDS. To verify the attack effect of samples generated using our algorithms, we evaluate whether they still retain malicious like initial ones on a semi-physical ICS platform. Our experiments show that a powerful ML-based detector can still be deceived easily. Therefore, some measures like adversarial training to enhance the robustness of detector must be taken into consideration.

Acknowledgments

This work was supported in part by the National Key Research and Development Program under Grant 2018YFB0803501, in part by National Natural Science Foundation of China under Grant 62073285, and in part by the Fundamental Research Funds for the Central Universities (Zhejiang University NGICS Platform).

References

[1] M. Wan, W. Shang, and P. Zeng, "Double behavior characteristics for one-class classification anomaly detection in networked control systems," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 12, pp. 3011–3023, Dec. 2017.

[2] M. Dong, K. Ota, L. T. Yang, A. Liu, and M. Guo, "LSCD: A low-storage clone detection protocol for cyber-physical systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 5, pp. 712–723, May 2016.

[3] A. Nourian and S. Madnick, "A systems theoretic approach to the security threats in cyber physical systems applied to Stuxnet," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 1, pp. 2–13, Jan./Feb. 2018.

[4] Z. Yu and J. J. P. Tsai, "A framework of machine learning based intrusion detection for wireless sensor networks," in *Proc. IEEE Int. Conf. Sensor Netw. Ubiquitous Trustworthy Comput.*, 2008, pp. 272–279.

[5] L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4665–4673, Oct. 2018.

[6] P. Nader, P. Honeine, and P. Beauseroy, "lp-norms in one-class classification for intrusion detection in SCADA systems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2308–2317, Nov. 2014.

[7] J. Wu, M. Dong, K. Ota, J. Li, and Z. Guan, "Big data analysis-based secure cluster management for optimized control plane in software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 27–38, Mar. 2018.

[8] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Secur. Artif. Intell.*, 2011, pp. 43–58.

[9] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010.

[10] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[11] H. R. Ghaeini and N. O. Tippenhauer, "HAMIDS: Hierarchical monitoring intrusion detection system for industrial control systems," in *Proc. 2nd ACM Workshop Cyber-Phys. Syst. Secur. Privacy*, 2016, pp. 103–111.

[12] C.-W. Ten, C.-C. Liu, and G. Manimaran, "Vulnerability assessment of cybersecurity for SCADA systems," *IEEE Trans. Power Syst.*, vol. 23, no. 4, pp. 1836–1846, Nov. 2008.

[13] S. A. Boyer, *SCADA: Supervisory Control and Data Acquisition*. Research Triangle, NC, USA: Int. Soc. Autom., 2009.

[14] G. Liang, S. R. Weller, J. Zhao, F. Luo, and Z. Y. Dong, "The 2015 Ukraine blackout: Implications for false data injection attacks," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 3317–3318, Jul. 2017.

[15] O. Linda, T. Vollmer, and M. Manic, "Neural network based intrusion detection system for critical infrastructures," in *Proc. Int. Joint Conf. Neural Netw.*, 2009, pp. 1827–1834.

[16] R. Das, V. Menon, and T. H. Morris, "On the edge realtime intrusion prevention system for DoS attack," in *Proc. Ind. Control Syst. Cyber Secur. Res.*, 2018, Art. no. 84.

[17] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner, "An evaluation of machine learning methods to detect malicious SCADA communications," in *Proc. 12th Int. Conf. Mach. Learn. Appl.*, 2013, pp. 54–59.

[18] D. Formby, S. Durbha, and R. Beyah, "Out of control: Ransomware for industrial control systems," in *Proc. RSA Conf.*, 2017, pp. 8–16.

[19] S. Rajasegarar, C. Leckie, J. C. Bezdek, and M. Palaniswami, "Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 3, pp. 518–533, Sep. 2010.

[20] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, *arXiv: 1702.05983*.

[21] D. Lowd and C. Meek, "Adversarial learning," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2005, pp. 641–647.

[22] B. Genge and C. Enăchescu, "ShoVAT: Shodan-based vulnerability assessment tool for internet-facing services," *Secur. Commun. Netw.*, vol. 9, no. 15, pp. 2696–2714, 2016.

[23] C.-Z. Gao, Q. Cheng, X. Li, and S.-B. Xia, "Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network," *Cluster Comput.*, vol. 22, no. 1, pp. 1655–1663, 2019.

[24] S. W. Boyd and A. D. Keromytis, "SQLrand: Preventing SQL injection attacks," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2004, pp. 292–302.

[25] J. Quick, J. Annoni, R. King, K. Dykes, P. Fleming, and A. Ning, "Optimization under uncertainty for wake steering strategies," *J. Phys., Conf. Ser.*, vol. 854, no. 1, 2017, Art. no. 012036.

[26] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," 2018, *arXiv: 1801.02610*.

[27] R. Lopez Perez, F. Adamsky, R. Soua, and T. Engel, "Machine learning for reliable network attack detection in SCADA systems," in *Proc. 17th IEEE Int. Conf. Trust Secur. Privacy Comput. Commun.*, 2018, pp. 633–638.

[28] J. Jiang and L. Yasakethu, "Anomaly detection via one class SVM for protection of SCADA systems," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov.*, 2013, pp. 82–88.

[29] K. H. Johansson, "The quadruple-tank process: A multivariable laboratory process with an adjustable zero," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 3, pp. 456–465, May 2000.

[30] D. Yang, A. Usynin, and J. W. Hines, "Anomaly-based intrusion detection for SCADA systems," in *Proc. 5th Int. Top. Meeting Nucl. Plant Instrum. Control Hum. Mach. Interface Technol.*, 2006, pp. 12–16.

[31] N. R. Rodofile, T. Schmidt, S. T. Sherry, C. Djamaludin, K. Radke, and E. Foo, "Process control cyber-attacks and labelled datasets on S7Comm critical infrastructure," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 2017, pp. 452–459.

[32] M. J. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in Optimization and Numerical Analysis*. Berlin, Germany: Springer, 1994, pp. 51–67.

[33] A. Almalawi, A. Fahad, Z. Tari, A. Alamri, R. AlGhamdi, and A. Y. Zomaya, "An efficient data-driven clustering technique to detect attacks in SCADA systems," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 5, pp. 893–906, May 2016.

[34] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[35] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[36] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582.

[37] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1660–1669.

[38] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 135–147.

[39] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[40] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," 2017, *arXiv: 1704.01155*.

[41] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of JPG compression on adversarial images," 2016, *arXiv:1608.00853*.

[42] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1378–1387.

[43] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 582–597.

[44] N. Akhtar, J. Liu, and A. Mian, "Defense against universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3389–3398.

[45] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *Proc. 1st ACM Workshop Cyber-Phys. Syst. Secur.*, 2015, pp. 13–24.

[46] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proc. ACM Symp. Inf. Comput. Commun. Secur.*, 2006, pp. 16–25.

[47] A. J. Bose and P. Aarabi, "Adversarial attacks on face detectors using neural net based constrained optimization," 2018, *arXiv: 1805.12302*.

[48] N. Dalvi *et al.*, "Adversarial classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2004, pp. 99–108.

[49] B. Nelson *et al.*, "Exploiting machine learning to subvert your spam filter," in *Proc. 1st Usenix Workshop Large-Scale Exploits Emergent Threats*, 2008, pp. 1–9.

[50] M. Osadchy *et al.*, "No bot expects the DeepCAPTCHA! introducing immutable adversarial examples, with applications to CAPTCHA generation," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 11, pp. 2640–2653, Nov. 2017.

[51] J. Kos, I. Fischer, and D. Song, "Adversarial examples for generative models," in *Proc. IEEE Secur. Privacy Workshops*, 2018, pp. 36–42.

[52] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 582–597.

[53] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, *arXiv:1611.01236*.

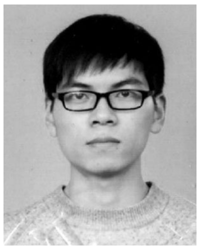
[54] L. Dritsoula, P. Loiseau, and J. Musacchio, "A game-theoretic analysis of adversarial classification," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 12, pp. 3094–3109, Dec. 2017.

Jiming Chen (Fellow, IEEE) received the BSc and PhD degrees in control science and engineering from Zhejiang University, Hangzhou, China, in 2000 and 2005, respectively. He was a visiting researcher with the University of Waterloo from 2008 to 2010. He is currently a changjiang scholars chair professor (MOE) with the College of Control Science and Engineering, the deputy director of the State Key Laboratory of Industrial Control Technology, and a member of the Academic Committee with Zhejiang University. His research interests include the Internet of Things, sensor networks, networked control, and control system security. He was a recipient of the Fok Ying Tung Young Teacher Award of the Ministry of Education and the IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award. He is an IEEE VTS distinguished lecturer.

Xiangshan Gao (Student Member, IEEE) received the BEng degree in computer science from Xidian University, Xi'an, China, in 2018. She is currently working toward the PhD degree in the State Key Laboratory of Industrial Control Technology, Group of Networked Sensing and Control (IIPC-NeSC), Zhejiang University, Hangzhou, China. Her research interests include control system security and machine learning.

Ruilong Deng (Senior Member, IEEE) received the BSc and PhD degrees both in control science and engineering from Zhejiang University, Hangzhou, Zhejiang, China, in 2009 and 2014, respectively. He was a research fellow with Nanyang Technological University, Singapore, from 2014 to 2015; an AITF postdoctoral fellow with the University of Alberta, Edmonton, AB, Canada, from 2015 to 2018; and an assistant professor with Nanyang Technological University, from 2018 to 2019. Currently, he is a professor at the College of Control Science and Engineering, Zhejiang University, where he is also affiliated with the School of Cyber Science and Technology. His research interests include cyber security, smart grid, communication networks, etc. He serves/served as an editor of the *IEEE Transactions on Smart Grid* and *IEEE/KICS Journal of Communications and Networks*, and a guest editor of the *IEEE Transactions on Emerging Topics in Computing*, *IEEE Transactions on Cloud Computing*, and *IET Cyber-Physical Systems: Theory & Applications*. He also serves/served as a symposium chair for IEEE SmartGridComm'19, IEEE GLOBECOM'21, etc. He is a recipient of two best paper awards, AITF postdoctoral fellowship, etc.

Yang He received the BEng degree in automation from Tianjin University, Tianjin, China, in 2016. She is currently working toward the MS degree with the State Key Laboratory of Industrial Control Technology, Group of Networked Sensing and Control (IIPC-NeSC), Zhejiang University, Hangzhou, China. Her research interests include control system security and machine learning.



Chongrong Fang (Student Member, IEEE) received the BEng degree in automation from Zhejiang University, Hangzhou, China, in 2015. He is currently working toward the PhD degree with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China. He visited the School of Computing, Engineering and Mathematics, Western Sydney University, Australia, from May 2017 to August 2017, and the School of Computer Science and Engineering, Nanyang Technological University, Singapore, from November 2018 to August 2019. His research interests include cyber-physical system security.

Peng Cheng (Member, IEEE) received the BSc degree in automation and the PhD degree in control science and engineering, from Zhejiang University, Hangzhou, China, in 2004 and 2009, respectively. From 2012 to 2013, he worked as a research fellow with Information System Technology and Design Pillar, Singapore University of Technology and Design. He is currently a professor with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China. His research interests include networked sensing and control, cyber-physical systems, and control system security.