SRID: State Relation Based Intrusion Detection for False Data Injection Attacks in SCADA

Yong Wang[1,2], Zhaoyan Xu[1], Jialong Zhang[1], Lei Xu[1], Haopei Wang[1], and Guofei Gu[1]

[1] SUCCESS Lab, Texas A&M University, College Station, Texas, USA
[2] Department of Information Security, Shanghai University of Electric Power, Shanghai, China
wybaidu@gmail.com, {z0x0427,jialong,xray2012,haopei,guofei}@cse.tamu.edu

Abstract. Advanced false data injection attack in targeted malware intrusion is becoming an emerging severe threat to the Supervisory Control And Data Acquisition (SCADA) system. Several intrusion detection schemes have been proposed previously [1, 2]. However, designing an effective real-time detection system for a resource-constraint device is still an open problem for the research community. In this paper, we propose a new relation-graph-based detection scheme to defeat false data injection attacks at the SCADA system, even when injected data may seemly fall within a valid/normal range. To balance effectiveness and efficiency, we design a novel detection model, *alternation vectors with state relation graph*. Furthermore, we propose a new inference algorithm to infer the injection point(s), i.e., the attack origin, in the system. We evaluate SRID with a real-world power plant simulator. The experiment results show that SRID can detect various false data injection attacks with a low false positive rate at 0.0125%. Meanwhile, SRID can dramatically reduce the search space of attack origins and accurately locate most of attack origins.

Keywords: Intrusion Detection System, Cyber Security in SCADA, False Data Injection Attack.

1 Introduction

In recent years, we have witnessed that the great technical innovation has intrinsically changed our definition about the data acquisition and control system. Nowadays, when current Supervisory Control And Data Acquisition (SCADA) system starts to connect a great number of sensors to highly-flexible distributed networks, it is no longer *closed* and *single-functional*, but instead *open*, and *complex*. Across the waves of such innovation, cyber security of SCADA systems attracts a lot of research attention. Especially after the Stuxnet[3] worm spread across Iran nuclear infrastructure and occupied the headline of news and media, we naturally ask the question: is our SCADA system ready for the challenges brought by such malware intrusion?

M. Kutyłowski and J. Vaidya (Eds.): ESORICS 2014, Part II, LNCS 8713, pp. 401–418, 2014.

Unfortunately, we may not be confident enough to declare a secure SCADA system: Our existing system is still connected to vulnerable networks which integrate multiple communication protocols and lack proper data validation and authentication [4–7]. It implies that attackers can easily infiltrate the network and compromise the whole control system. Besides that, the potential vulnerability of industrial control system software may also become the Achilles' heel and open a back-door for those malicious attackers. As stated in a vulnerability trends report from Symantec [8], the number of vulnerabilities targeting at SCADA systems has undergone an *exponential* uptrend since 2011. For example, there were over 800% more vulnerabilities discovered in 2012 than the number discovered in 2010.

Therefore, deploying intrusion detection onto SCADA systems becomes a pressing task for security practitioners. Multiple schemes, such as behavior-based scheme [9] and bloom-filter-based scheme [1], have been proposed. However, when we review the recent intrusion incidents conducted by targeted malware, such as the infamous Stuxnet malware, we find these IDS schemes are neither complete nor accurate. In particular, there are two problems causing current IDS systems vulnerable to targeted malware's intrusion:

First, the current design of many IDS systems follows a hierarchical and distributed structure, in an attempt to secure all sensors and devices in the system [10]. However, with too high-level view and always limited resources, this may be ineffective and inefficient to handle some attacks deeply targeting at some specific critical control component. In reality, it has been evidenced that malware shows special interest in some key component, e.g., Stuxnet only infects Siemens S7-3000 devices which control the centrifuge and pumps. Therefore, an IDS scheme that can automatically find and protect the critical control component(s) is a more effective solution to detect targeted malware's intrusion.

Second, many IDS systems follow the idea of traditional intrusion detection schemes, which detect abnormal communication flow(s) between devices. However, in the context of SCADA, our first priority is to ensure the validity of system status. Therefore, we think our detection target should be invalid system *states*. As an illustration, we examine the Stuxnet worm again: when the control device has been compromised, the Stuxnet periodically modifies the frequency to 1410 Hz and then to 2 Hz and then to 1064 Hz, and thus affects the operation of the connected motors by changing their rotational speed. Typically, such behaviors cannot be detected by existing IDS because each of its operation is under the valid threshold of IDS's detection rules. However, the system states, or the relations between continuous states, violate the normal disciplines, and such inconsistency is a clear evidence when intrusion happens in the SCADA system.

In this paper, we propose a novel relation-graph-based intrusion detection scheme, SRID, which aims to detect false data injection attacks in SCADA systems. In particular, given a SCADA system, SRID automatically analyzes the system and extracts independent *components* (each component is a logically independent control system, more details defined in Section 2.3). Then, for each

component, SRID extracts the internal relations among different system variables and derives a graph model to describe valid system states. To achieve a balance between effectiveness and efficiency, we propose *alternation vector* and *state relation graph* as our detection model. To further figure out the origins of attacks, i.e, which device (among all possible data sources) has been compromised, we propose an inference algorithm to deduce what is the possible compromised device causing the inconsistency.

To evaluate the effectiveness and efficiency of SRID, we test it with a real-world power plant simulator with 142 system variables. We then inject malformed data into the simulator with both single-point and multi-point injection schemes. As shown in the result, almost all the injected data and origins can be accurately detected and inferred.

In summary, our paper makes the following contributions:

- We introduce a graph-based scheme for SCADA systems to detect advanced false data injection attacks, even when injected data fall within the valid/normal range of signal specification. SRID is a systematic approach to monitor system states, detect *inconsistent states* and infer the *compromised origins*.
- We propose multiple novel intrusion detection models for SCADA, such as *alternation vectors* and *state relation graphs*, to achieve real-time detection on resource-constrained devices.
- We evaluate our system with a real-world power plant simulator and SRID performs with a 95.83% detection rate and a 0.0125% false positive rate. Meanwhile, our inference module can also achieve a high accuracy in locating injected data origins.

2 Background and Problem Statement

2.1 Background

SCADA(supervisory control and data acquisition) is an advanced control system which collects the data from *system variables* in real-time, operates with encoded signals over communication channels, and provides control of local/remote equipment. These years, as SCADA systems have been widely applied in Smart Grid systems for data monitoring and state estimation [11], cyber security on SCADA has attracted more and more public attention as an emerging, cross-disciplinary research topic. Among those attacks, false data injection attack is of particular interest since it directly affects the reliability and robustness of systems. As defined in related work [11], false data injection is a generalized attack which injects any type of malformed data into the data acquisition system. Such malformed data can be in any form, such as the measurement data from sensors [12], or even the control commands from programmable logic circuits [13].

To detect such injected data, there are two lines of research work. In the first direction, researchers treat the injected data as an injected signal and detect it using bad-data (e.g., data out of normal ranges) processing schemes. As a

common detection strategy, existing studies, such as [14, 15], introduce artificial redundancy to mitigate the effect of injected data. These approaches have been proved to be effective for securing the integrity of small numbers of system variables. The common assumption of these work is that the attacker has only *limited* access to system resources, i.e, altering a small amount of measurement data of sensors or meters. However, when well-crafted malware has been involved in the arm race, such assumption may no longer be held because malware has strong abilities to propagate itself among similar devices and manipulate multiple data signals at the same time.

To defeat malware's intrusion, another line of research concentrates on deploying existing intrusion detection techniques onto SCADA systems. Multiple different detection models, such as statistic-based [16], behavior-based [9] and bloom-filter-based [1], have been proposed. However, as a highly complicated control system, deploying similar or identical rules on different control components is not an ideal choice. For example, the centrifuge control component may require different policies from the gas pump control component. In addition, most existing IDS systems only enforce detection rules on each communication flow, they lack a global view of relationship among different variables to determine whether the whole system is under attack or not.

2.2 Assumption and Approach Overview

In this paper, we focus on detecting false data injection attacks on SCADA systems. To be concrete, we assume the following strong attack model:

- An attacker has the ability to inject single or multiple data at any time.
- The injected data can fall into the valid range of signal specification, which can cause a difficult time for many existing anomaly-based detection schemes.

In this paper, we provide a relation-graph-based intrusion detection scheme for SCADA systems. Our proposal is trying to achieve two tasks complementing the existing intrusion detection schemes:

- **Component-based Policy**: Our scheme automatically analyzes the internal relations among system variables. Then we extract components for each independent control sub-system. Based on each component, we provide different detection rules. Through such component-based policy, we can not only reduce the redundant overhead but also provide more accurate results.
- **Correlation Model**: Our detection model is based on the internal relations among different system variables and states. Hence, we design correlation graphs (for system variables) and state relation graphs (for system states) to describe the valid behavior patterns of our target SCADA system. Therefore, SRID is designed to protect the system from entering invalid running status.

Next, we provide our definition of *component* and *correlation model*.

2.3 Terminology

Definition of Component: In our definition, we define a component as an subsystem which controls an independent set of system variables. As an illustration, we show an example of the controlling component of a power plant boiler in Figure 1. In this system, coal is automatically transported into boiler based on the temperature meter. If the temperature is high, few coal will be transported into the system. Otherwise, more coal is needed. The pressure is proportional to the temperature, which is collected by the temperature meter. Thus, higher temperature generates higher pressure. Since all these system variables affect the control of the boiler, we treat them as a component.

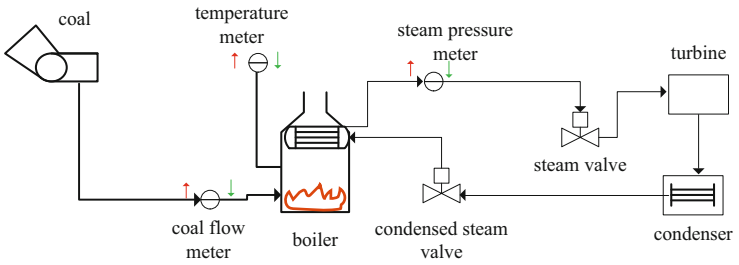


Fig. 1. An Example of False Data Injection

In this example, we find a component:{t(temperature), p(pressure), c(coal)}. The component is normally a subset of the whole control system. In our design, we allow users to customize the system variable subset. Also, we provide another automatic way to extract independent components from a SCADA system.

Definition of Correlation Model: In the above example, suppose that the attacker can compromise the temperature meter to inject fake data into the system. In order to evade bad-data based detection, she keeps changing the value of temperature meter to a valid low value. While the temperature keeps lower, the coal will continuously be transported. Finally, the boiler could blast.

Note that there is no detectable bad data. Hence, bad-data processing scheme will be less effective in this case. However, if we further examine the relation among different meters, we can find the inconsistency: while the attacker can modify the temperature meter to a lower value, the value from the pressure meter is unchanged, which undoubtedly violates the proportional relation between two meters. Our definition of State-Correlation is based on such insight.

Formally, we define the Correlation Model as the correlation between different variables x_i under different time states t_i.

$$c(x_{i,t_i}) = f(x_1, x_2, x_3, ..., x_n) \tag{1}$$

More specifically, we consider two types of variable correlation models in our definition:

Forward Correlation: The forward correlation is a static structure, in which all system variables will not be affected by the time. Thus, the value of one variable

in the current state only depends on the values of other related variables in the simultaneous state. Given the power system variables $\mathbf{x} = (x_1, x_2, ..., x_n)^T$, Equation 2 reflects such forward relation. At any time t, the value of variable i depends on the values of other variables such as variable j, variable k at time t.

$$x_{i,t} = f(x_{j,t}, x_{k,t}...) \tag{2}$$

Since the values of system variables do not depend on time t in the forward correlation structure, Equation 3 can be further simplified as:

$$x_i = f(x_j, x_k...) \tag{3}$$

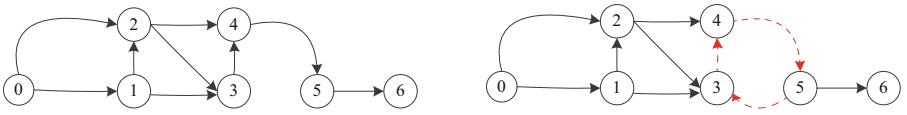
Feedback Correlation: The feedback correlation is a dynamic structure corresponding to the time. In such relation, the value of one variable in the current state depends on not only the values of other related variables in simultaneous states but also the values of some related variables in the previous state. Equation 4 reflects such feedback relationship. At time t, the value of variable i depends on the value of variable j at time t and the value of variable k at time $t-1$.

$$x_{i,t} = f(x_{j,t}, x_{k,t-1}...) \tag{4}$$

Table 1. Functions of forward and feedback data relation structures

structure \ variable	x_0	x_1	x_2	x_3	x_4	x_5	x_6
forward	x_0	$x_0 * x_0$	$x_1 - x_0$	x_2/x_1	$x_2 - x_3$	$\sin(x_4)$	$\cos(x_5)$
feedback	$x_{0,t}$	$x_{0,t} \times x_{0,t}$	$x_{1,t} - x_{t,0}$	$(x_{2,t}/x_{t,1}) + x_{5,t-1}$	$x_{2,t} - x_{3,t}$	$\sin(x_{4,t})$	$\cos(x_{5,t})$

We illustrate another example to describe the correlation model as shown in Table 1. Given system variables $\mathbf{x} = (x_0, x_1, x_2, x_3, x_4, x_5, x_6)^T$, Table 1 shows the relationship of different variables in different types of models. For example, variable x_1 in the forward data relation structure equals the squares of x_0 and variable x_3 equals x_2/x_1; Variable x_3 in the feedback data relation structure equals $x_{2,t}/x_{1,t} + x_{5,t-1}$.



(a) forward relation graph of variables (b) feedback relation graph of variables

Fig. 2. Forward and Feedback Correlation Model

Based on the table description, we further generate a *correlation graph*, which is shown in Figure 2. In the figure, the dash line shows the feedback correlation, in which the value of x_3 depends on the value of x_1 and x_2 from the current state and the value of x_5 from the previous state.

3 System Design

In this section, we present the detailed design of SRID. As seen in Figure 3, there are three basic steps of SRID: *Component Analysis*, *Detection Model Generation* and *Origins Inference*. In the first step, SRID automatically analyzes the internal relations between different variables in the SCADA system. In the second step, we propose a graph-based detection model, *alternation vectors with state relation graphs*, for efficient online intrusion detection. Finally, our inference model traces back the intrusion and infers the possible compromised origins.

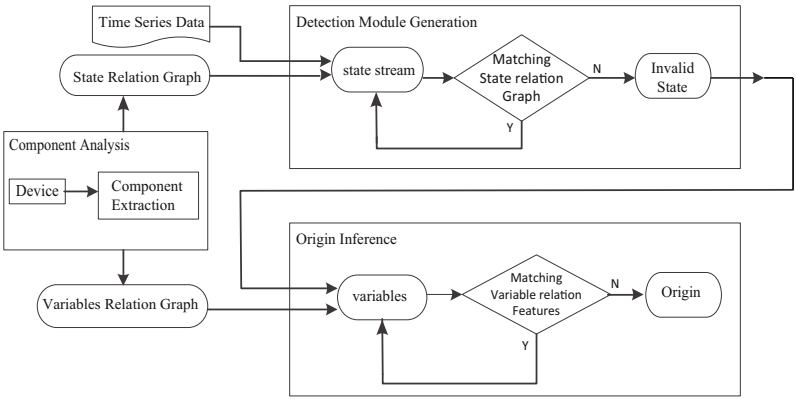


Fig. 3. Architecture of SRID

3.1 Component Analysis

In the first step, our target is to find the internal relations among system variables inside each independent component. As we discussed, the component can be expressed as a set of system variables, such as $x_1, x_2, ...x_n$. The goal of component analysis is to derive the forward and backward correlations, which can be expressed as correlation graphs, illustrated in Figure 2.

To build such relations, the most straightforward and easiest solution is to allow the system designer to specify in advance. With the involvement of human assistance, we can obtain an accurate model which specifies the mathematical relations of each variable. However, since such human effort is tedious and sometimes not available, we can propose another approach for automatic extraction.

The idea is to apply classic control variate method which alters one variable's value at a time. When we alter one variable, we record whether any of other variables has been changed or not. If some variable changes, we build a directed edge from the control variable to the alternated variable in the graph. Then we reset the system, and change another variable in the second round. The process iteratively continues till we find all the relations between variables.

The control variate method can describe the forward correlation between variables. For the feedback correlation, we apply program analysis on the device's

firmware. The idea is to conduct data flow analysis on each variable and determine whether the previous states may affect its current value or not. To achieve that, we collect a set of execution traces for the firmware from t_0 to time window limit t_m. Then we apply the data flow analysis across different traces and find whether the previous variable affects the following states. If so, we draw a feedback line (as seen in Figure 2) in our graph.

One sample output of this step is the correlation graph illustrated in Figure 2. Based on the graph, we can extract all connected components. As we mentioned, we allow users to specify the analyzed component in advance. However, if the user does not specify any component beforehand, we can analyze the whole SCADA system and treat each connected component in the graph as the subject(s) of our further steps.

3.2 Detection Model Generation

In the second step, we study the *changing* pattern of the variables for the component. The task of SRID is to determine, at time t_i, given the current states of component variables, whether the system is under attack or not?

To answer the question, we need to first describe the pattern of normal system operation. In our design, we propose a novel way, *alternation vectors*, to represent real-time states of a component under normal operation.

State Representation Using Alternation Vectors. Suppose a component has n variables $(x_1, x_2, ..., x_n)^T$. At each time t, a state can be represented by a vector of different variable such as $(x_{1,t}, x_{2,t}, ..., x_{n,t})^T$. For that, we have to store the concrete value for each variable and such scheme may consume large amount of memory space when the vector is high dimensional. In our scheme, we apply alternation vector, which only records the alternation relations between two continuous states. It can be expressed with Equation 5:

$$f_1(x_{i,t}) = \begin{cases} 1 & x_{i,t} - x_{i,t-1} > 0 \\ 0 & x_{i,t} - x_{i,t-1} = 0 \quad or \quad t = t_0 \\ -1 & x_{i,t} - x_{i,t-1} < 0 \end{cases} \quad , i \in 1, 2, \cdots, n \tag{5}$$

For the initialization state, t_0, we define $f(x_{i,t0}) = 0$. If the value of variable i increases from the last value, we use 1 to indicate the increase. Also if the value of variable i decreases from the last state, we use -1 to indicate the decrease. If the value keeps the same, we denote it as 0.

Using alteration vectors, we can model a constantly changing component by a series of alternation vectors in a time window from t_0 to t_m. The advantage of alternation vectors is to save memory for each state. Since we only store 2 bits for each variable's state, our scheme is efficient for resource-constraint devices.

Based on the alternation vectors, we discuss our detection scheme using state relation graphs.

State Relation Graph. Our intrusion detection model is based on *state relation graphs*. The state relation graph is a directed graph $G(V, E)$, which describes the

normal states of the component. To construct the relation graph, we need to run SRID and collect all the information about those normal states in a training stage.

Suppose that we run SRID to train the model from a time window t_0 to t_m, and we have n different variables in the analyzed component. At each time slot t_i, we compute the alternation vector based on its previous state at t_{i-1} and current state. For each alternation vector, we create one node in the graph. If we find the node has been created before, we directly use the existing node. Then we create one edge which points from the t_{i-1} state's node to t_i state's node. Each edge is marked with a time stamp t_i. This process continues until we enumerate all the states in the time window.

The state relation graph for our illustration example(in Section 2.3) is shown in Figure 4(a). In theory, there are at most 3^n different alternation vectors in the graph. However, for a practical system, the space is normally limited. For the illustrated example, we have 7 variables and only 37 different state nodes in the graph.

Using the graph, we can build our detection model. The idea for our intrusion detection model is to assure that the variables satisfy the normal changing rule. Intuitively, the malicious injected data, such as Stuxnet's malformed frequency data, can be easily detected since it clearly violates the normal changing rules.

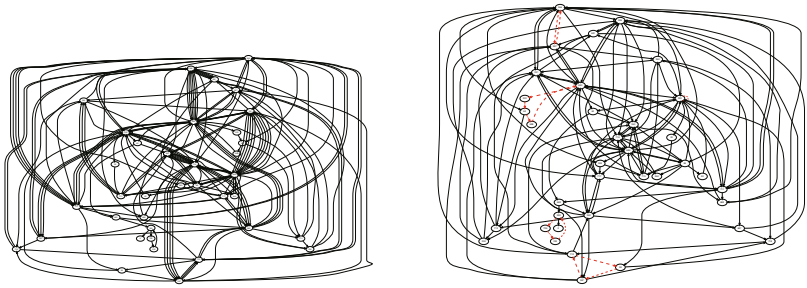
Reduced State Relation Graph. There are two possible problems directly using the state relation graph for detection: First, since we have to maintain the time information for each edge, we have to consume considerable memory to store the trained model. It may complicate the matching process and overburden the resource-constrained device. Secondly, strictly following the transition edges may cause some false positives if some states are not stable.

Therefore, we optimize our state relation graph and remove the time stamp information for each edge. After removing time stamps, there are many duplicated edges existed in the graph. Hence, in the second step, we traverse the whole graph and remove all duplicated edges. The example's reduced state relation graph is shown in Figure 4(b). As we can see, the graph is greatly simplified and, if we apply linked list structure, it only consumes less than 360 bytes to store the whole graph in memory.

Based on the reduced state graph, we summarize the detection steps:

- *Step I:* In detection phase, if we find the new alternation vector is not a node in the graph, we directly generate an alert for *Invalid State*. Since we can store all the nodes in a hashtable, it takes $O(1)$ to fulfill the check.
- *Step II:* If it is a valid state in the graph, we have to check whether it is reachable from the previous state or not. If it is not reachable from the previous state, we generate an alert for *Invalid Transition*. It takes $O(1)$ space to maintain the last transition state and $O(1)$ time to detect the invalid transition (use a hashtable for each node's edge).

In all, SRID can find possible intrusion/false data in $O(1)$ time which is particularly attractive for real-time detection. Meanwhile, with limited number of



(a)Original State Relation Graph (b)Reduced State Relation Graph

Fig. 4. Time-series State Relation Graph Reduction

possible states, it also saves the memory consumption and achieves a better balance between effectiveness and efficiency.

3.3 Attack Origins Inference

The goal of the origins inference model is to infer the possible injection point. Our inference is based on the correlation graph which is generated in Component Analysis. As we introduced befor, the correlation graph describes the internal relations of variables in the component. Therefore, when we detect the violated state, we can trace back the dependence of violated variable(s) and find the possible origin(s).

Next, we analyze our inference algorithms in several possible scenarios.

Scenario I: Known Mathematics Correlation
As we discussed in Section 3.1, we allow users to provide the mathematical relations for the correlation graph. In this case, we have pre-knowledge about the variable relation. We believe it is a valid assumption because existing bad-data processing schemes [14, 15, 17] hold the similar assumption.

With knowing the mathematical correlation, our inference algorithms can precisely locate all possible injection points, no matter whether it is a single-point injection or multiple-point injection. To illustrate that, we go through an

Table 2. Example of Origin Inference

Infer Variable	x_0	x_1	x_2	x_3	x_4	x_5	x_6
math correlation	x_0	$x_0 * x_0$	$x_1 - x_0$	x_2/x_1	$x_2 - x_3$	$\sin(x_4)$	$\cos(x_5)$
Original value	4	16	12	0.75	11.25	-0.9678	0.5671
Injection value				3.5			
Invalid state	4	16	12	3.5	8.5	-0.7985	0.6978
Inference result	0	0	0	1	0	0	0

inference example in Table 2. We first reconstruct the time series data of the whole component. As shown in Table 2, a false data 3.5 is injected to x_3. Such injection will also change the value of variables x_3, x_4, x_5, x_6. To infer the origins of false data injection, we check the inner relation among the variables based on their values in invalid states. For example, the value of x_3 should be $x_2/x_1 = 0.75$, however, the value of x_3 in the invalid state is 3.5. Such inconsistency indicates that the variable x_3 is the attack origin. For variable x_4, $x_4 = x_2 - x_3 = 8.5$, which is the same as the value of x_4 in invalid state, thus x_4 is not the attack origin. In the same way, x_1, x_2, x_4, x_5, x_6 are not the origins of the false data injection attack.

Scenario II: Single Point Injection
When the attacker has only limited access to the system, our inference model can also deduce its source even without the knowledge of mathematical correlation in advance. In the single point injection case, the attacker can only inject one data in the system. The inference algorithm is based on the backward dependence traversal which starts from the violated node in the correlation graph. We backward check whether the previous variable deviates from the normal value or not. If the deviation is above our pre-defined deviation threshold δ, we mark it as a possible compromised variable. We iteratively continue till we find the first valid variable. Since we only have one injection point, the last compromised variable, with its corresponding device, is considered as the injection point.

Scenario III: Multiple Points Injection
When there are multiple injection points, our correlation graph may not locate the precise points, but it can still generate a set of possible compromised variables. It is also computed by the backward dependence traversal till we reach the root node of the dependence chain. We believe our infer algorithm can save a great amount of work in the investigation.

Finally, our overall inference algorithm is presented in Algorithm 1.

4 Evaluation

To evaluate the performance of SRID, we test it with a boiler simulator of a real power plant.

4.1 Data Collection

The boiler system is the core part of a power plant generator, which makes it a popular target for malware. Thus we test our SRID with a common boiler system simulator in a coal power plant. To find the mathematical relationship among different variables, we reverse engineering the boiler simulator system. 142 variables are extracted from the boiler system. Figure 5 shows the relation graph of variables and the dash line represents the feedback structures.

4.2 Overhead Analysis

To evaluate the overhead of SRID, we check how many states generated/maintained by SRID. Since there are 142 variables in the boiler system,

Algorithm 1. SRID: State relation based intrusion detection for false data injection attack

Input: A sequence of power system state $\langle state(1,t), state(2,t) \ldots, state(n,t)\rangle$
Output: When dose the false data $state(k,i)$ injection attack happen? , where is the original injected data $variable(j)$?

△ **injectionDetectModel**
$time \leftarrow t_0$
$stateFeatureGraph[N,N] \leftarrow normalStateFeatures$
while $time < timeEnd$ **do**
  $state[n,time] \leftarrow input(dataFlow)$
  $index \leftarrow 0$
  while $index < N$ **do**
    if $state[index,time] < min(state[index])$ ***or*** $> max(state[index])$ **then**
      return *out of normal range*
    else if $state[index,time] \in stateFeatureGraph$ **then**
      $index \leftarrow index + 1$
      continue;
    else
      $i \leftarrow time$
      $k \leftarrow index$
      return $state(k,i)$
  $time \leftarrow time + 1$
△ **originsInferModel**
$index \leftarrow 0$
$M \leftarrow variableNumber$
$variable[M] \leftarrow input$(state(k,i))
$variableRelationFeature[M,M] \leftarrow normalVariableRelationFeatures$
while $index < M$ **do**
  $variable[index] \leftarrow input(variableDataFlow)$
  if $varialbe[index] \in variableRelationFeature$ **then**
    $index \leftarrow index + 1$
    continue;
  else
    $j \leftarrow index$
    return $variable(j)$
return $state(k,i)$, $variable(j)$

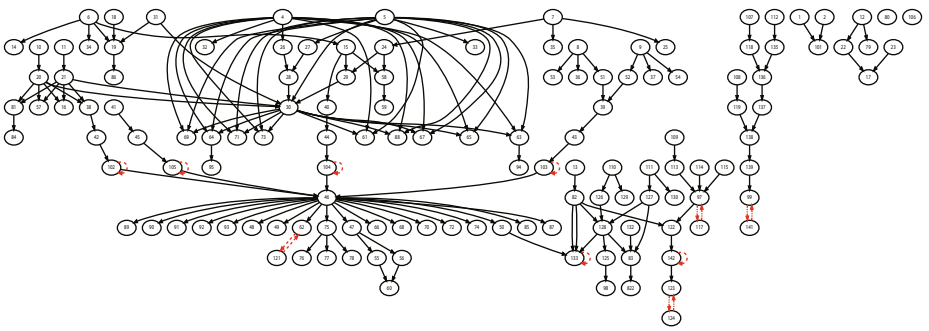


Fig. 5. Variable Relation Graph of the Boiler System in a Power Plant

theoretically there will be 3^{142} states. We train our system with different duration t. Figure 6 shows the number of generated states along with different training duration.

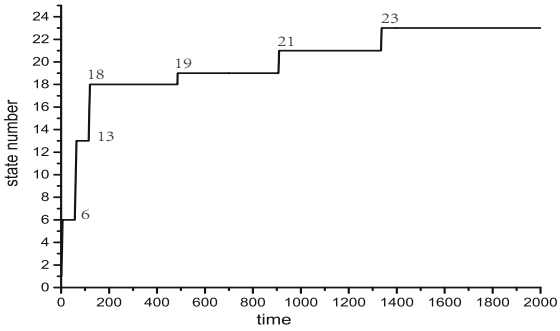


Fig. 6. States Number Distribution

We can see that after 1,366s, there is no more new states generated and the total number of states is only 23, which is much smaller than 3^{142}. Thus such state graph can be stored in memory for realtime analysis.

4.3 Attack Detection Results

To evaluate the detection results, we first run the boiler system for 2,000s (0-2,000s, the values of variables recorded every second, a time granularity used for this evaluation but could be adjusted based on practical need/constraint) to generate the corresponding state relation graph as shown in Figure 7. Then we continue running the boiler system for another 2,000s (2,000-4,000s) to detect when the data injection attacks happen.

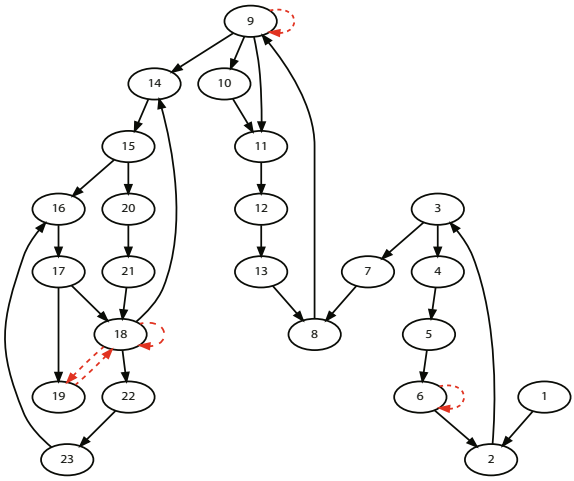


Fig. 7. State Relation Graph of the Boiler

We test SRID with both single-point injection and multiple-point injection. For the single-point injection, each time we randomly choose a variable and inject with arbitrary data that falls in its valid range. Then we randomly inject 6 times in this way during the testing procedure. Thus, among 2,000 total testing data records (one per second), there are 6 false data injection attacks. For the multiple-points injection, instead of injecting false data on a single variable each time, we inject false data on different numbers of randomly chosen variables at the same time. We also launch 6 false injection attacks for each situation during the testing procedure.

Table 3. Detection on false data injection attacks

Types	False data injection		SRID detection	
	Injected variables	Injection attacks	Detected attacks	False positives
Single injection	1	6	6	0
Multi-injection1	2	6	6	0
Multi-injection2	3	6	6	0
Multi-injection3	4	6	5	1

Table 3 shows the detection results. SRID can detect all the single injection attacks without false positives. For multiple-points injection, SRID can still detect all attacks with 2 and 3 variables injection without any false positives and false negatives. However, it failed to detect one multiple-points injection attack with 4 variables. This leads to an overall detection rate of 95.83% (23 out of 24 attacks). Further analysis shows that this missed attack is because two continuous injections on two related states happen to satisfy the learned (normal) alternation relationship. While it is possible, we believe that this is difficult be exploited by attackers, as further discussed in Section 5.

SRID also has one false positive (out of a total of 1994*4 normal data records), which represents a false positive rate of 0.0125%. Further investigation shows that it is because a normal state was not captured by the state relation graph. Thus, SRID reports it as a false data injection attack. However, extending the training time can reduce such false positives.

In summary, in our evaluation SRID has achieved a high overall detection rate of 95.83% for false data injection attacks, with a very lower false positive rate of 0.0125%.

4.4 Attack Origin Inference Results

Once the false data injection is detected in the boiler system, we need to find out what are the origins of those attacks. To evaluate the performance of our attack origin inference algorithm, we use the same test data mentioned above and submit the invalid states from detection results to our inference scheme. Table 4 shows the inference results.

Table 4. Origin inference on data injection attacks (TP means inferred true attack origins, and FP means inferred false positive origins. For Multi-injection3, we only test the attack origin inference on the 5 successfully detected true attacks.)

Types	False data injection		SRID Inference		
	Injected variables	Injection attacks	Total affected variables	Total TP	Total FP
Single injection	1	6	40	6/6	0
Multi-injection1	2	6	94	12/12	0
Multi-injection2	3	6	62	18/18	0
Multi-injection3	4	5	421 (avg. 84 per attack)	13/20	260

SRID accurately infers all the true attack origins without any false positive in the case of single injection, multi-injection1, and multi-injection2. For example, for the single-point injection case, totally 6 variables (1 in each attack) are injected with the false data, which lead to totally 40 affected variables that change their values, and SRID can successfully infer the 6 exact attack origins without false positives. In the case of multi-injection3, SRID does not infer all attack origins (inferred 13/20) and due to the large number of variables affected by the attacks (each randomly injected attack affected 84 variables on average, causing trouble for the inference), SRID unavoidably involves considerable false positives (on average 52 in each multi-injection3 attack inference result). However, we argue that SRID still significantly reduces the search space of affected variables (reducing from 421 to 273) and locates most of true attack origins, which is still useful in practice.

Further analysis shows that if the false data is injected on source variables (i.e., source vertex in the variable relation graph), all the children variables of source variables will change their values according to their relations. Thus, there will not exist any inconsistency among those variables. In this case, SRID can not accurately locate the attack origins.

In summary, SRID can dramatically reduces the search space of attack origins and even accurately locate most of true attack origins.

5 Discussion

In this section, we discuss possible problems and evasions of SRID.

Limitation of Component Analysis: One of our design challenges of SRID is how to handle the case when the system is a blackbox to defenders. To solve that, we present our component analysis to handle the challenge. However, there exists some limitation for the scheme. First, the classic control variate method may not be precise if the relation between variables is not instantly reactive. Second, we apply dynamic program analysis to handle feedback relations. However, if we cannot find the data-flow between different execution traces, we cannot find the accurate model. As a result, it may cause some false positives in the final detection result.

To solve these issues, one possible solution is mentioned in Section 3.3, i.e., applying manual effort to describe the mathematical correlation model. Since it

is a common assumption for many existing bad-data detection schemes [14], we believe it is still feasible in many real-world circumstances.

Possible Evasions: To evade our SRID, attackers can modify the initial state at t_0, and introduce the inconsistency at the beginning of the time series. Such attack cannot be detected by SRID. Hence, we assume we can ensure the integrity at the starting point t_0. We think it is a reasonable assumption because the initial integrity check is required by most of the devices.

Another possible way to evade our system is to inject the data that affects all other related states at the same time. It means the attacker has the *full* control of the component, which, we believe, is not realistic in the practical scenario. In this case, our system cannot detect the data injection attack. However, if the injection data violates the threshold of some variable, which introduces new states in our graph, SRID can still detect it.

Also, it is possible to overburden SRID using Denial of Service (DoS) attacks. It can be achieved by injecting a large amount of false data continuously. Even though such situation rarely happens in practice, we believe that our system can still outperform existing schemes because SRID applies different policies for different components and each component cannot be easily separated for distributed detection.

Complement to Existing Schemes: Last but not least, we need to emphasize that our scheme is not trying to replace existing intrusion detection systems in the field. Our scheme is complementary to existing schemes and especially useful when we try to protect the critical component in the system. However, some anomaly-based [18] and behavior-based [9] schemes provide a consistent protection for the entire SCADA system, which is not the focus of SRID.

6 Related Work

Existing intrusion detection solutions in SCADA can be classified into two categories.

In the first category, existing approaches [1, 5, 12, 13, 19, 20] monitor the abnormal behaviors using predefined rules to detect attacks. SmartAnalyzer [9, 10, 12, 13] detects possible attacks on advanced metering infrastructure (AMI). It applies a verification engine to determine whether the sensor behaviors obey with the predefined threat constraints, such as reachability constraint, security pairing constraint, report schedule constraint, resource constraint, cyber bandwidth constraint, priority delivery constraint and quality of delivery constraint. Another behavior-rule based intrusion detection system, BRIDS [9], is proposed to secure the SCADA system in a distributed way. In [1], an idea of using bloomer-filter to detect intrusion in resource-constrained devices is proposed.

In the second category, researchers applied state estimation to mitigate bad data in the data acquisition sensor [15, 17, 21–26]. This kind of approach is widely used to detect and identify bad data in power systems, such as power flow

analysis [26] and topology errors detection [25]. The detection methods include primary detection for residual detection [21] and inflection point detection [23]. However, the application domains of these approaches are very limited, since they only target at detecting data injection attacks with minor data alternation [11].

Our work is different from all previous work with a stronger attack model, in which we assume powerful attackers can even evade rule-based detection schemes by understanding each variable's threshold in advance. Meanwhile, the state estimation solution also has the limitation of handling arbitrary data injection.

7 Conclusion

In this paper, we propose a novel intrusion detection system, named SRID, to detect intrusion in SCADA systems. Our main defense focus is the false data injection attack and SRID can not only detect such attack but also deduce the possible attack origins in an effective and efficient way. In addition, we propose a new graph-based detection model which combines the state alternation vectors and state relation graph. From the evaluation results, we can see our new design can effectively detect various data injection attacks and infer attack origins.

References

1. Parthasarathy, S., Kundur, D.: Bloom filter based intrusion detection for smart grid scada. In: Proc. of the 25th IEEE Canadian Conference on Electrical & Computer Engineering (CCECE 2012), pp. 1–6 (April 2012)
2. Amin, S., Litrico, X., Sastry, S., Bayen, A.: Cyber security of water scada systems (i) analysis and experimentation of stealthy deception attacks. IEEE Transactions on Control Systems Technology 21(5), 1963–1970 (2013)
3. Stuxnet, `http://en.wikipedia.org/wiki/Stuxnet`
4. Cardenas, A.A., Amin, S., Lin, Z.S., Huang, Y.L., Huang, C.Y., Sastry, S.: Attacks against process control systems: Risk assessment, detection, and response. In: Proc. of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011 (March 2011)
5. Valenzuela, J., Wang, J., Bissinger, N.: Real-time intrusion detection in power system operations. IEEE Transactions on Power Systems 28(2), 1052–1062 (2013)
6. Stouffer, K., Falco, J., Scarfone, K.: Guide to industrial control systems (ics) security. In: NIST Special Publication (2013)
7. Sridhar, S., Hahn, A., Govindarasu, M.: Cyber physical system security for the electric power grid. IEEE Transactions on Power Systems 100(1), 210–224 (2012)
8. Scada vulnerabilities
9. Mitchell, R., Chen, I.: Behavior-rule based intrusion detection systems for safety critical smart grid applications. IEEE Transcations on Smart Grid 4(3), 1254–1263 (2013)
10. Berthier, R., Sanders, W., Khurana, H.: Intrusion detection for advanced metering infrastructures: Requirements and architectural directions. In: Proc. of First IEEE International Conference on Smart Grid Communications (SmartGridComm 2010), pp. 350–355 (October 2010)

11. Liu, Y., Ning, P., Reiter, M.K.: False data injection attacks against state estimation in electric power grids. ACM Transactions on Information and System Security 14(1), 21–32 (2011)
12. Rahman, M., AL-Shaer, E., Bera, P.: A noninvasive threat analyzer for advanced metering infrastructure in smart grid. IEEE Transcations on Smart Grid 4(1), 273–287 (2013)
13. Rahman, M., Bera, P., Al-Shaer, E.: Smartanalyzer: A noninvasive security threat analyzer for ami smart grid. In: Proc. of the 31st IEEE International Conference on Computer Communications (INFOCOM 2012), pp. 2255–2263 (March 2012)
14. Esmalifalak, M., Shi, G., Han, Z., Song, L.: Bad data injection attack and defense in electricity market using game theory study. IEEE Transactions on Smart Grid 4(1), 160–169 (2013)
15. Hagh, M., Mahaei, S., Zare, K.: Improving bad data detection in state estimation of power systems. International Journal of Electrical and Computer Engineering (IJECE 2011) 1(2), 85–92 (2011)
16. Ning, P., Jajodia, S.: Intrusion detection techniques (2003)
17. Xu, W., Wang, M., Tang, A.: On state estimation with bad data detection. In: Proceedings of 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC 2011), pp. 5989–5994 (December 2011)
18. Reeves, J., Ramaswamy, A., Locasto, M., Bratus, S., Smith, S.: Intrusion detection for resource-constrained embedded control systems in the power grid. International Journal of Critical Infrastructure Protection 5(2), 74–83 (2012)
19. McDonald, M.J., Conrad, G.N., Service, T.C., Cassidy, R.H.: A retrofit network intrusion detection system for modbus rtu and ascii industrial control systems. In: Proc. of the 45th Hawaii International Conference on System Science (HICSS 2012), pp. 2338–2345 (January 2012)
20. Diaz, J.: Using snort for intrusion detection in modbus tcp/ip communications (2011)
21. Bi, S., Zhang, Y.: Defending mechanisms against false-data injection attacks in the power system state estimation. In: Proc. of the 2011 IEEE International Workshop on Smart Grid Communications and Networks (GC Wkshps 2011), pp. 1162–1167 (December 2011)
22. Xie, L., Mo, Y., Sinopoli, B.: False data injection attacks in electricity markets. In: Smart Grid Communications, pp. 226–231 (October 2010)
23. Feng, Y., Foglietta, C., Baiocco, A., Panzieri, S., Wolthusen, S.D.: Malicious false data injection in hierarchical electric power grid state estimation systems. In: Proc. of the 4th International Conference on Future Energy Systems (e-Energy 2013), pp. 183–192 (May 2013)
24. Tan, R., Krishna, V.B., Yau, D.K., Kalbarczyk, Z.: Impact of integrity attacks on real-time pricing in smart grids. In: Proc. of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS 2013), pp. 439–450 (November 2013)
25. Pajic, S.: Power System State Estimation and Contingency Constrained Optimal Power Flow-A Numerically Robust Implementation. PhD thesis, Worcester Polytechnic Institute (2007)
26. Lin, J., Yu, W., Yang, X., Xu, G., Zhao, W.: On false data injection attacks against distributed energy routing in smart grid. In: 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems (ICCPS 2012), pp. 183–192 (April 2012)