Chapter 4

COMPARISON OF DESIGN- AND DATA-CENTRIC METHODS FOR DISTRIBUTED ATTACK DETECTION IN CYBER-PHYSICAL SYSTEMS

Jennifer Leopold, Bruce McMillin, Rachel Stiffler and Nathan Lutes

Abstract Cyber-physical systems are vulnerable to a variety of cyber, physical and cyber-physical attacks. The security of cyber-physical systems can be enhanced beyond what can be achieved through firewalls and trusted components by building trust from observed and/or expected behaviors. These behaviors can be encoded as invariants. Information flows that do not satisfy the invariants are used to identify and isolate malfunctioning devices and cyber intrusions. However, the distributed architectures of cyber-physical systems often contain multiple access points that are physically and/or digitally linked. Thus, invariants may be difficult to determine and/or computationally prohibitive to check in real time. Researchers have employed various methods for determining the invariants by analyzing the designs of and/or data generated by cyber-physical systems such as water treatment plants and electric power grids. This chapter compares the effectiveness of detecting attacks on a water treatment plant using design-centric invariants versus data-centric rules, the latter generated using a variety of data mining methods. The methods are compared based on the maximization of true positives and minimization of false positives.

Keywords: Cyber-physical attacks, invariants, data mining, water treatment plant

1. Introduction

Cyber-physical systems (CPSs) typically contain multiple entry points, especially via edge devices (e.g., smart meters, home monitors and cameras) that provide access via service provider core networks. It is often the case that the distributed architectures of cyber-physical systems have access points that are physically and/or digitally linked. As a result, a

computationally-prohibitive number of (combinatorial) conditions need to be checked in real-time to enforce the security of the overall systems.

Efforts to secure cyber-physical systems must contend with a number of challenges:

- Cyber and physical information flows should be combined in a single representational model. However, the semantics of the combined cyber-physical information flows in security domains are often not well understood. A good understanding of the semantics is needed to automate the task of securing information flows.
- Information flows are fundamentally bidirectional between two security domains [9]. An information flow either simultaneously preserves integrity but not confidentiality, or simultaneously preserves confidentiality but not integrity. This needs to be made clear to system designers.
- Mining system behavior by observing system operations may not uncover all the operational modes or may result in an overly complex model with a surplus of generated rules.
- Observations of physical systems may not be timely, accurate or complete due to malicious information, bad data, noisy data and communications delays.
- The proliferation of individual security domains causes high model complexity. Merging the security domains can result in a trivial, all-encompassing security domain.

A promising solution for addressing cyber-physical system vulnerabilities is to enhance the security of cyber-physical systems beyond what can be achieved through firewalls and trusted components by building trust from the observed and/or expected behaviors. These behaviors are encoded as invariants. Redundant, yet inconsistent, information flows that do not satisfy the invariants can help identify and isolate malfunctioning devices and cyber intrusions. Researchers have employed a variety of methods that analyze the designs of and/or data generated by cyber-physical systems such as water treatment plants and electric power grids. This work empirically compares the effectiveness of detecting attacks on a water treatment plant using design-centric invariants versus data-centric rules generated by several data mining methods.

2. Related Work

An invariant is a condition that should hold during the flow of information through a device or process at it moves from one state to

the next. Invariants for a cyber-physical system can be derived using a variety of methods. A design-centric method determines invariants by examining the cyber-physical system design and control algorithms. This method has been employed by Adepu and Mathur [1, 2], with their Distributed Attack Detection (DAD) system [2] being notable for its use of only continuous (i.e., non-discretized) data. In contrast, a data-centric method determines invariants by analyzing the data generated by a cyber-physical system under (labeled) normal and attack conditions. Various data mining and machine learning methods have been employed to generate invariants, including association rule mining, support vector machines, decision trees and neural networks [7, 10].

Umer et al. [10] have compared design-centric and data-centric methods for deriving invariants for a water treatment plant. In the design-centric approach, system component specifications and state condition graphs of the plant were examined and 39 invariants were manually defined (see [3] for a discussion of state condition graphs). In the data-centric approach, association rule mining was employed, which produced hundreds of invariants.

The analysis of Umer et al. yielded two key conclusions. First, the data-centric approach did not find all the invariants defined by the design-centric approach. This was likely due to the loss of information from the data discretization needed to perform association rule mining and the lack of data that tested certain possible/expected conditions. The second conclusion was that the design-centric approach did not include all the invariants defined by the data-centric method, likely due to the comprehensive nature of the data mining method that considered all possible combinations of conditions. Nine invariants were found to be common to the data-centric and design-centric rules. Umer and colleagues mention that an integrated approach for invariant generation might be advantageous, but they did not provide a comparative analysis.

3. Background

This section describes the cyber-physical system testbed used to analyze design-centric and data-centric rules for detecting attacks. It also discusses the metrics used for evaluating the rule sets.

3.1 Secure Water Treatment Plant

The dataset used in this study was obtained from the Secure Water Treatment (SWaT) plant, a testbed for cyber security research [4]. Figure 1 shows the architecture of the plant. The plant produces five gallons/minute of treated water and can operate non-stop, 24 hours per

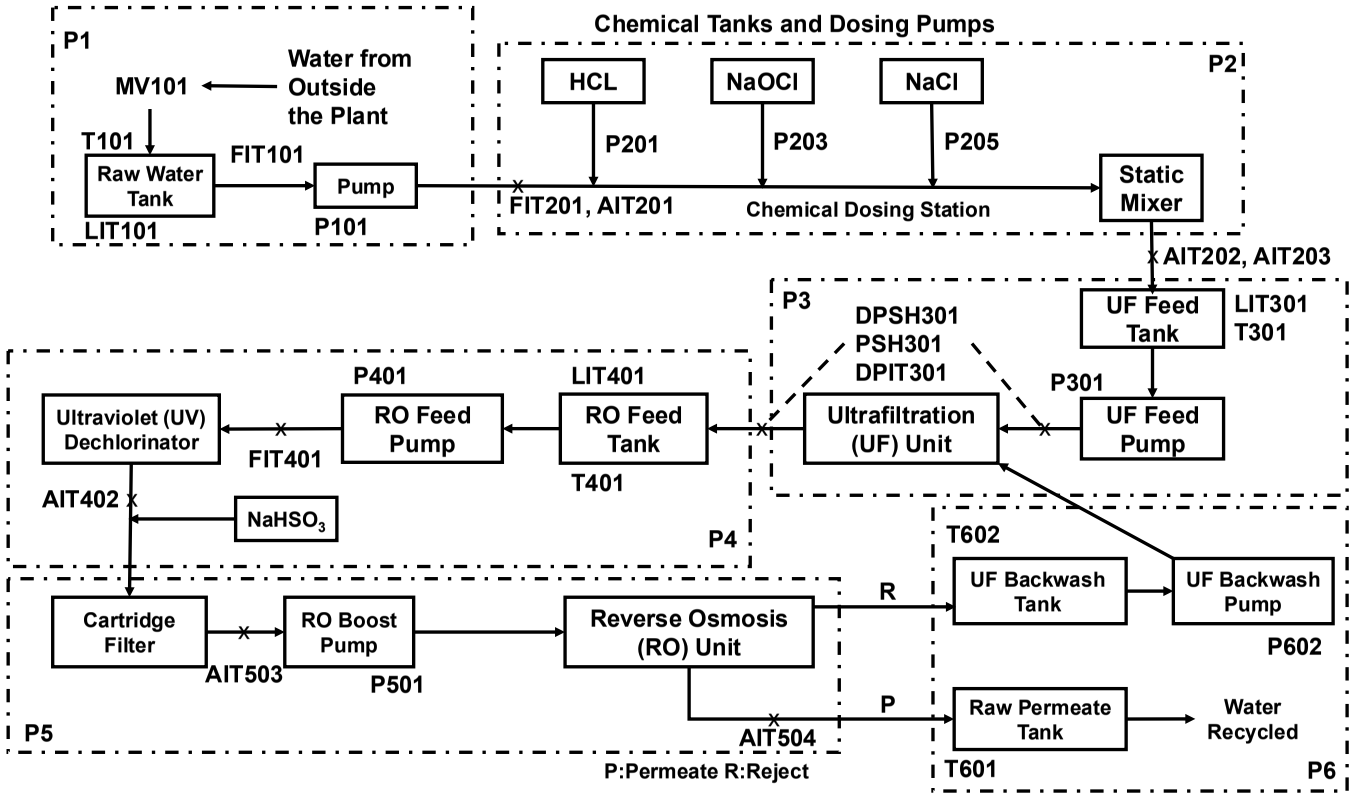

Figure 1. Secure Water Treatment (SWaT) plant architecture [10].

day, seven days a week in a fully autonomous mode. The plant has six stages (processes), each of which is controlled by a programmable logic controller. The states of the stages are measured by sensors and the control actions are performed by actuators. The plant has 68 sensors and actuators; some of the actuators are standby devices that are used only when the primary actuators fail.

A multi-layer ring network supports plant communications. Programmable logic controllers at one level can communicate with sensors and actuators at another level and in different stages. Plant operations are monitored and controlled by an operator at a supervisory control and data acquisition (SCADA) workstation. Physical attacks include replacing or removing sensors, disconnecting wires between components (i.e., sensors, actuators and programmable logic controllers) and interrupting power flow to electronic components.

Each instance in the dataset corresponds to a discrete or continuous value of a sensor and the timestamp denoting when the reading was taken. The readings were taken every second over 24-hour periods, operating for a certain number of days under normal conditions and a certain number of days under attacks.

The original dataset [5] comprised 890,298 instances labeled as sensors operating normally and 54,621 instances labeled as sensors under attack. However, when the dataset was created, the system was operating under a variety of non-standard (i.e., atypical) conditions with no indications of states. As will be discussed in the next section, some of the design-centric rules are sensitive to the system operating state. Also, the values of some attributes in the dataset were not within normal ranges and/or were fluctuating abnormally during certain states or state changes, although the system was not under attack. Without the system state information, the original dataset was inadequate for evaluating design-centric as well as data-centric methods.

The dataset used in this research (currently available at [5]) does not contain data related to atypical states (e.g., initial startup and backwash). Additionally, the dataset maintains state information. It has 13,070 instances labeled as sensors operating normally and 1,926 instances labeled as sensors under attack (a binary attribute is_attack identifies an instance as normal (0) or attack (1)). However, no information is provided about the nature of the attacks.

3.2 Evaluation Metrics

The effectiveness of design-centric versus data-centric rules at detecting attacks was evaluated using the following basic metrics:

- **True Positive (*TP*):** A true positive is an instance labeled as an attack (i.e., is_attack=1) that was correctly detected as an attack.

- **False Positive (*FP*):** A false positive is an instance labeled as normal (i.e., is_attack=0) that was incorrectly detected as an attack.

- **True Negative (*TN*):** A true negative is an instance labeled as normal (i.e., is_attack=0) that was correctly detected as normal.

- **False Negative (*FN*):** A false negative is an instance labeled as an attack (i.e., is_attack=1) that was incorrectly detected as normal.

In the cyber security domain, the highest priority is to maximize the number of true positives and minimize the number of false negatives. A false negative – missing an attack – can have severe consequences. False positives correspond to "crying wolf," so large numbers of false positives are undesirable, but they do carry less severe consequences.

The following metrics are derived from the four basic metrics:

- **Accuracy:** Accuracy is the proportion of the total number of instances that were correctly identified. It is computed as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

- **Precision:** Precision is the proportion of instances that were correctly identified as positive from among the total number of instances that were correctly or incorrectly identified as positive. It is computed as:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- **Sensitivity:** Sensitivity, which is also referred to as recall, is computed as:

$$Sensitivity = \frac{TP}{TP + FN} \tag{3}$$

- **Specificity:** Specificity is the proportion of negative instances that were correctly identified from among the total number of negative instances. It is computed as:

$$Specificity = \frac{TN}{TN + FP} \tag{4}$$

In general, precision and sensitivity are important because they are viewed as metrics of exactness and completeness of testing, respectively.

The kappa statistic is also useful for evaluating a decision model:

- **Kappa Statistic:** The kappa statistic indicates how much better a predictive model is compared with random guessing. It is computed as:

$$Kappa = \frac{Non\text{-}Random\,Successes - Random\,Successes}{N - Random\,Successes} \quad (5)$$

  where *Non-Random Successes* is the number of instances correctly predicted by a non-random model and *Random Successes* is the number of instances correctly predicted by a random model and N is the number of instances in the dataset.

4. SWaT Rule Evaluation

This section compares the effectiveness of design-centric and data-centric rules obtained from the SWaT dataset. Rules produced by a variety of methods are evaluated based on their ability to detect attacks in the SWaT dataset.

4.1 Design-Centric Method

Umer et al. [10] manually generated 39 invariants by examining the SWaT specifications and state condition graphs. However, a number of problems were encountered when attempts were made to test the accuracy of these rules on the SWaT dataset.

Rules are specified using the syntax *Antecedent* → *Consequent*. Each rule was intended to be checked in the following manner where n is the number of seconds to wait for the consequent to become true after the antecedent was satisfied:

```
If the Antecedent is true
   Then wait n seconds
   If the Antecedent and Consequent are true
      Then the system is functioning normally
      Else the system is not functioning normally
   End-if
End-if
```

Interestingly, the situation where the antecedent becomes false during the n-second wait period is not considered, a condition that occurred

seven times in the SWaT dataset. The only concern is that the antecedent and consequent are true at the end of the n-second period. This could be a problem if, for example, a water tank reaches a certain level (satisfying the antecedent), leaks some water on the floor due a malfunction or malfeasance during the wait period (thereby no longer satisfying the antecedent), but refills to a certain level (again satisfying the antecedent) before the end of the wait period.

Umer et al. [10] do not report the time constraints for the design-centric invariants. Therefore, one of the authors (Adepu) was contacted to obtain the time constraint for each rule. Even then, the time constraints for several rules had to be increased by a few seconds to best fit the SWaT dataset (i.e., increase the attack detection accuracy).

Other problems encountered with the original design-centric rules included the specification of attributes that were not in the dataset, unspecified threshold values, reversals of conjunctions and disjunctions, and missing conditions in antecedents and consequents. These problems necessitated the elimination of a few rules (i.e., those involving attributes that were not in the dataset) and resulted in a proliferation of false positives and false negatives. Many of these problems were resolved with the assistance of one of the authors of [10] (Adepu). Other problems were resolved by examining the data and/or specifications in the SWaT operating manual [6].

Tables 1 and 2 show the final set of 31 design-centric rules along with their time constraints in seconds. The order of rules has no significance. Values of 1 and 2 for pumps (e.g., P101) represent off and on, respectively. Values of 1 and 2 for valves (e.g., MV201) represent closed and open, respectively. Attributes with the prefix PLC represent the state of the system (e.g., initial startup and backwash). Interested readers are referred to the SWaT operating manual [6] for more information about the attributes.

The rule set was treated as a set of invariants. Starting from any instance in the dataset, all the rules had to be satisfied or else an attack was considered to have occurred. Because multiple rules with different time constraints had to be satisfied for an attack not to have occurred, it was not possible to compute the true negative rate (i.e., number of labeled normal instances detected as normal). For similar reasons, false negatives (i.e., number of labeled attack instances detected as normal) could not be determined easily. Consequently, the accuracy, specificity and sensitivity values could not be computed.

When the set of design-centric rules was tested on the SWaT dataset, the false positive rate (i.e., instances labeled as normal but detected as attacks) was 55.47% and the true positive rate (i.e., actual attacks

Table 1. Design-centric invariants.

Rule	Rule Specification	Time Constraint (sec)
1	LIT101≤500 → MV101=2	14
2	LIT101≥800 → MV101=1	12
3	LIT101≤250 → P101=1 and P102=1	2
4	LIT301≤800 and PLC1!=1 → P101=2 or P102=2 or PLC1!= 2	12
5	LIT301≥1000 → P101=1 and P102=1 and MV201=1	3
6	LIT301≤800 and PLC2!=1 → MV201=2	10
7	FIT201≤0.5 → P201=1 and P202=1 and P203=1 and P204=1 and P205=1 and P206=1	2
8	AIT201≥260 and FIT201≥0.5 → P201=1 and P202=1	2
9	MV201=2 and FIT201≥0.5 and AIT503≤260 and AIT201≤250 and PLC2!=1 → P201=2 or P202=2	7
10	AIT503≥260 and MV201≥0.5 → P201=1 and P202=1	2
11	MV201=2 and FIT201≥0.5 and AIT503≤260 and PLC2!=1 → P201=2 or P202=2	2
12	AIT202≤6.95 → P203=1 and P204=1	2
13	MV201=2 and AIT202≥7.05 and FIT201≥0.5 and PLC2!=1 → P203=2 or P204=2	8
14	AIT203≥500 → P205=1 and P206=1	2
15	P101=2 or P102=2 and MV201=2 and AIT203≤420 and FIT201≥0.5 and AIT402≤250 and PLC2!=1 → P205=2 or P206=2	6
16	AIT402≥250 → P205=1 and P206=1	2
17	MV201=2 and FIT201≥0.5 and AIT402≤240 and PLC2!=1 → P205=2 or P206=2	6
18	LIT301≤=250 → P301=1 and P302=1	4
19	LIT401≥1000 → P301=1 and P302=1	2
20	LIT301≥250 and LIT401≤800 → P301=2 or PLC3!=7	2

Table 2. Design-centric invariants (continued).

Rule	Rule Specification	Time Constraint (sec)
21	LIT401≤250 → P401=1 and P402=1	4
22	LIT401≤250 → UV401=1	4
23	P401=1 and P402=1 → UV401=1	4
24	FIT401≤0.5 → UV401=1	4
25	P401=1 and P402=1 and PLC4=4 → UV401=1	4
26	AIT402≤240 → P403=1 and P404=1 or PLC4!=4	2
27	AIT402≥250 → P403=2 or P404=2 or PLC4!=4	2
28	P401=1 and PLC5=12 → P501=1	5
29	UV401=1 → P401=1 or P402=1	4
30	FIT401≤0.5 and PLC5=12 → P501=1 or P502=1	4
31	LIT101≥1100 → P601=1	2

correctly detected as attacks) was 44.53%. In all, there were 446 occurrences of false positives, 14 of which were caused by Rule 1 and 432 by Rule 5. These problems may have been due to faulty components. For example, in the case of Rule 5, a motorized valve (MV201) may have had trouble closing; however, increasing the time constraint for Rule 5 only resolved eight of the 432 false positives attributed to the rule.

Of greater concern than the large number of false positives was the unacceptably low rate of true positives. A possible explanation is that the threshold values in the design-centric rules may not have been realistic with regard to the actual performance of the individual components. For example, the LIT301 attribute was required to be less than or equal to 800 in Rule 4; however, in one of the data-centric rule sets, LIT301 had to be less than or equal to 793.7869. This was the case for several attributes common to the design-centric and data-centric rule sets. Another possible explanation for the low true positive rate may have been the time constraints. Manual examination of the dataset revealed several instances where it took longer than the specified amount of time for the consequent in a rule to be satisfied. When the time constraint was increased for some of the rules, the increase in the true positive rate was often accompanied by an increase in the false positive rate.

4.2 Data-Centric Methods

Several data mining methods were employed to generate rules for detecting attacks in the SWaT dataset. Unlike the set of invariants, each rule has as its consequent a single binary condition that indicates whether or not an attack has occurred. Also, unlike the set of invariants, it is not necessary for an instance to satisfy all the rules in the rule set. Additionally, none of the data mining methods took into account the temporal nature of the physical system; specifically, unlike the design-centric rules, they did not consider that a certain amount of time had to transpire between the satisfaction of the rule conditions. For this reason, 10-fold cross validation was used to test the methods. Breaks in time sequences between instances were not as critical as they were in the design-centric testing.

All the data mining methods were implemented using the R programming language. None of the methods required discretization of the SWaT data. However, attributes with values that did not change from row to row were removed before creating the data-centric models (i.e., AIT401 and all of pumps attributes except for P101, P203, P205, P301, P401 and P601). These attributes were not informative and would not have been included in any decision model.

The data mining methods – some of them ensemble methods – that produced the best accuracy were: J48 and C5.0 (both decision tree methods with pruning), rfRules (random forest specifying a maximum of 300 trees), AdaBoost.M1 (boosting with decision trees), naïve Bayesian network (NB), support vector machine (SVM) (linear with a polynomial kernel), JRip (association rules with reduced error pruning) and multilayer perceptron neural network (NNet). Other methods such as linear regression models and association rule methods were attempted. However, the results of these methods are not discussed here because their accuracy was not competitive or their results were not as informative as the other methods. Interested readers are referred to [11] for details about the data mining methods used in this work.

Table 3 compares the results obtained using the eight data mining methods based only on accuracy and the kappa statistic. Figures 2 and 3 show more extensive comparisons in terms of accuracy, kappa statistic, precision, sensitivity and specificity.

The results indicate that the two ensemble methods (rfRules and AdaBoost.M1) did not improve on the individual decision tree methods (J48 and C5.0), and did not justify the additional complexity and memory requirements of an ensemble method. The naïve Bayesian network and neural network were also eliminated from the top candidates due to

Table 3. Accuracy and kappa for eight data mining methods.

Method	Accuracy	Kappa
J48	0.998044	0.991256
Naïve Bayesia Network	0.871880	0.212844
Support Vector Machine	0.998311	0.992461
rfRules	0.984352	0.926389
AdaBoost.M1	0.999289	0.996819
C5.0	0.997866	0.990452
JRip	0.998399	0.992863
Neural Network	0.871533	0.000000

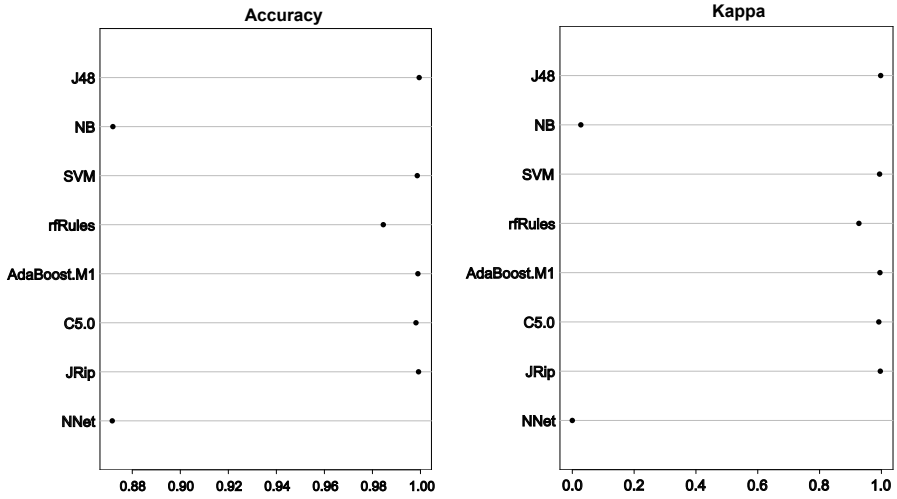

Figure 2. Evaluation metrics for the eight data mining methods.

their lower accuracy, kappa statistic, sensitivity (in the case of the naïve Bayesian network), precision and specificity. Although the support vector machine performed well, it was decided to investigate only the J48, C5.0 and JRip models because they performed well and were the most comparable models in terms of their expression as rule sets.

Stacking was employed using the three selected methods (J48, C5.0 and JRip) as the base classifiers and the best performing of the three, JRip, as the ensemble method. This was done to see if an ensemble of the best three methods could improve on the performance of the individual

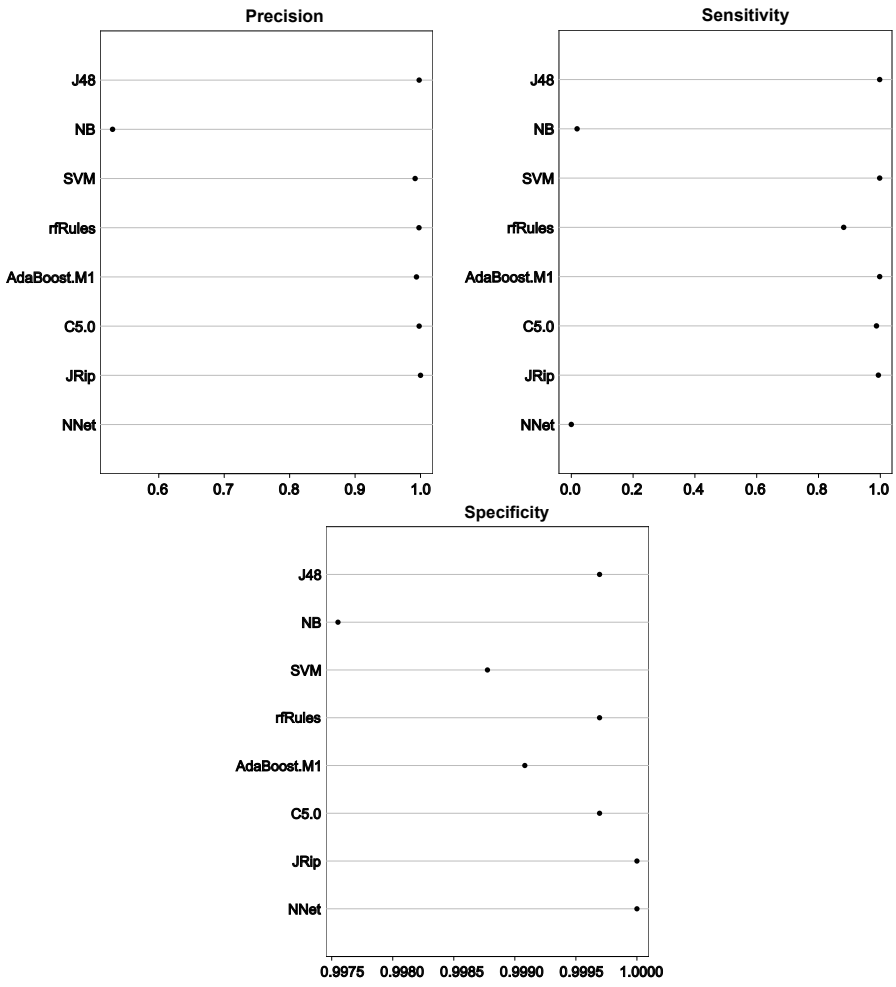

Figure 3. Evaluation metrics for the eight data mining methods (continued).

methods. The accuracy of the ensemble method was 0.998399 and its kappa statistic was 0.992813. Since these results did not improve on those obtained using the individual JRip method, the ensemble method was not considered any further in the interest of memory efficiency.

The J48, C5.0 and JRip predictive models are expressed as rule sets. Even rules derived from pruned decision trees (e.g., J48 and C5.0) can result in unnecessary conditions. Therefore, in order to construct the most efficient rule sets, a program was written in R to check for and drop unnecessary conditions. The strategy that was utilized (originally presented in [8]) is summarized in the following paragraphs.

Table 4. Contingency table for instances satisfying *A'*.

	C	***!C***
X	$Y1$	$E1$
!X	$Y2$	$E2$

Let R be a rule of the form $A \rightarrow C$ where A is the antecedent, C is the consequent and the antecedent A contains more than one condition. Let R' be a more general rule of the form $A' \rightarrow C$ where A' is obtained by deleting one condition X from A'. Consider the contingency table in Table 4, where the counts $Y1$, $Y2$, $E1$ and $E2$ were collected by considering instances in the dataset that satisfy condition A'.

Let $U_{25}(E, S)$ be a binomial distribution using a 25% confidence level for a sample size S with E negative cases in the sample. The error rate of rule R is estimated as $U_{25}(E1, Y1 + E1)$ and the error rate of rule R' is estimated as $U_{25}(E1 + E2, Y1 + Y2 + E2 + E2)$. If the error rate of R' is no greater than that of R, then the condition X is deleted. Using a greedy approach, the condition that yields the lowest error rate for the rule is successively eliminated until no more conditions can be eliminated. The choice of 25% for the confidence level is considered a conservative value for maintaining high accuracy in rule sets [8].

Table 5. Profile of data-centric rules before and after conditions were dropped.

	J48	**JRip**	**C5.0**
No. of rules before dropping conditions	25	11	19
No. of rules with conditions dropped	19	2	1
Average no. of conditions in antecedents	5.68	2.64	2.42

Table 5 shows the number of rules before dropping conditions, number of rules that had conditions dropped and average number of conditions in the antecedent per rule (after dropping conditions) for the J48, JRip and C5.0 methods.

Tables 6 and 7 show the JRip and C5.0 rule sets, respectively. Due to space constraints, the J48 rule set is not shown.

Only a few identical rules were produced by the methods, specifically J48 and C5.0, both of which are decision tree methods. Nevertheless, there are some commonalties between the rule sets.

Table 6. JRip rule set.

Rule	Rule Specification
1	AIT202≤8.931236 and LIT301≥923.2809 → is_attack=1
2	AIT402≤6.613689 and PIT503≥113.800949 and PIT501≤159.590485 and LIT101≥596.9952 and DPIT301≤1.245278 and DPIT301≥1.229272 → is_attack=1
3	AIT402≤6.613689 and AIT402≥5.075622 and LIT301≤829.274536 and LIT101≤769.3148 → is_attack=1
4	AIT402≤6.664958 and LIT301≥938.781738 and AIT203≥262.4968 → is_attack=1
5	LIT301≥1024 and AIT502≥69.21302 and FIT503≥0.610313 and LIT401≤864.932068 → is_attack=1
6	LIT301≥1027.94153 and AIT203≤240.810043 → is_attack=1
7	LIT301≥1112.57532 and AIT202≥9.202961 and AIT201≥126.24968 → is_attack=1
8	LIT401≤787.2597 → is_attack=1
9	FIT601≥0.000384 and LIT301≥1024 and LIT401≥848.936157 → is_attack=1
10	DPIT301≤1.104424 → is_attack=1
11	Else → is_attack=0

Figure 4 shows the attributes that are common and different in the J48, JRip and C5.0 rule sets based on their references in the rule antecedents and consequents. Of particular interest are the six attributes included in all three rule sets: AIT202, AIT203, LIT301, LIT401, AIT402 and AIT201. These attributes could be the primary "canaries in the coal mine" with respect to detecting attacks. In fact, they may have been the direct targets of the attacks.

Not surprisingly, the six common attributes also featured prominently in the design-centric rule set. However, the design-centric rule set was distinctive in that it frequently specified conditions on various pumps (attributes with the prefix P followed by three digits such as P201) and the ultraviolet dechlorinator (UV401), whereas only one pump attribute (P301) was used in the J48 and C5.0 rules and the dechlorinator attribute was never used in J48, C5.0 and JRip rules. In fact, the P203 and P205 attributes have identical values in the SWaT dataset, eliminating the need to test more than one of them.

Table 7. C5.0 rule set.

Rule	Rule Specification
1	AIT202≤9.383043 and LIT301≤923.2008 → is_attack=0
2	LIT301≥923.2008 and LIT401≥878.4286 and AIT201≥126.0895 → is_attack=0
3	AIT402≥6.664958 → is_attack=0
4	LIT301≤923.2008 and LIT401≥954.0246 → is_attack=0
5	AIT202≥8.931236 and LIT401≤805.5627 → is_attack=0
6	AIT202≥8.931236 and PIT503≤113.6728 → is_attack=0
7	P301≥0 and AIT202≥8.931236 and LIT401≤849.5514 → is_attack=0
8	AIT202≥8.931236 and AIT202≤8.996923 → is_attack=0
9	AIT203≤262.3686 and FIT501≥0.80223 and AIT201≤126.0895 → is_attack=0
10	AIT203≤262.3686 and FIT504≥0.209781 and AIT201≤126.0895 → is_attack=0
11	AIT202≤9.020315 and LIT401≥865.4319 → is_attack=0
12	AIT202≥8.931236 and LIT301≥923.2008 and LIT301≤938.3812 → is_attack=0
13	LIT301≤793.7869 → is_attack=0
14	AIT202≥9.070623 and FIT503≤0.606984 → is_attack=0
15	AIT402≤3.409382 and FIT401≥0.802484 → is_attack=0
16	AIT202≥8.996923 and AIT402≤6.664958 and LIT401≥805.5627 and LIT401≤849.5514 → is_attack=1
17	AIT202≥9.383043 and AIT203≥243.2453 and AIT402≤6.664958 and LIT401≤954.0246 → is_attack=1
18	AIT202≥9.39554 and LIT301≥793.7869 and AIT402≤6.664958 → is_attack=1
19	LIT301≥923.2008 and AIT402≤ 6.664958 → is_attack=1

Another point of interest is that, of the six SWaT stages (labeled P1–P6 in Figure 1), the least often referenced attributes in the design-centric and data-centric rules are from stages P1, P5 and P6. Perhaps the attacks targeted the P2, P3 and P4 stages and/or there is a "ripple effect" between the sensors in these three stages.

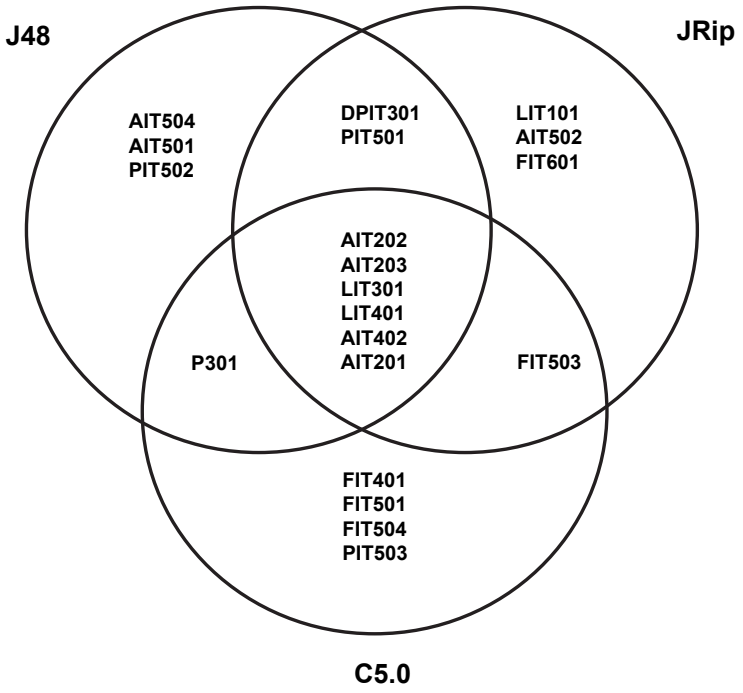

Figure 4. Common and different attributes in the data-centric rule sets.

5. Conclusions

Detecting attacks on cyber-physical systems is an important but challenging task. A promising approach is to enforce invariants, conditions that must hold during the operation of a physical process and, when violated, are indicative of a component fault or cyber attack. Invariants can be generated by manually examining the design of a system (design-centric method) and/or by deducing patterns observed in data collected under normal and abnormal system operating conditions (i.e., data-centric method).

The comparison of the effectiveness of detecting attacks on a water treatment plant using one design-centric rule set and multiple data-centric rule sets generated by data mining methods reveals that the data-centric methods are far better at predicting attacks and minimizing false negatives. The results of this study cannot be generalized and are not intended to indict design-centric invariants. In fact, a design-centric method leverages domain knowledge about how the system should behave under various conditions. However, because the task is time-consuming (typically manual), it may not cover all the contin-

gencies. Furthermore, design-centric rules may require adjustments to threshold values to account for the wear-and-tear on system components. In contrast, a data-centric approach can more easily generate an optimal combination of the numerous conditions that identify anomalous behavior. As demonstrated by the results, a data-centric approach learns (and re-learns) from a real-time system that may no longer correspond to the original design specifications. However, a data-centric rule set is only as complete as the data on which it is based; if the underlying dataset does not cover all possible situations, the resulting rule set will not check for all contingencies. Overall, the best strategy is to consider the information provided by design-centric as well as data-centric methods.

The data-centric rules produced in this study are based on a single dataset and may be subject to overfitting, despite undergoing 10-fold cross validation. In future research, it would be interesting to test these rules on the live SWaT testbed, much like the work reported in [2].

Other structured cyber-physical datasets are available from the iTrust Center for Research in Cyber Security [5], including the Water Distribution (WADI) and Electric Power and Intelligence Control (EPIC) datasets. Future work could compare design-centric and data-centric methods for detecting attacks in these datasets. However, this will require design-centric invariants, which could be generated in a semi-automated manner from system schematics and design documents.

Finally, future research will investigate other data-centric methods for detecting attacks. A promising problem is to model a cyber-physical system as a graph and use dynamic graph anomaly detection algorithms to deduce invariants. Modeling a cyber-physical system as a graph is a natural extension of its interconnected physical design and may provide a bridge to developing automated methods for generating design-centric invariants.

Acknowledgements

This research was performed with the assistance of three Missouri University of Science and Technology students – Nathan Lincoln, Michael Macke and Alex Warhover. The research project was supported by the National Science Foundation under Grant No. CNS-183747.

References

[1] S. Adepu and A. Mathur, Generalized attacker and attack models for cyber-physical systems, *Proceedings of the Fortieth Annual IEEE Computer Software and Applications Conference*, pp. 283–292, 2016.

[2] S. Adepu and A. Mathur, Distributed attack detection in a water treatment plant: Method and case study, to appear in *IEEE Transactions on Dependable and Secure Computing*, 2020.

[3] S. Adepu, A. Mathur, J. Gunda and S. Djokic, An agent-based framework for simulating and analyzing attacks on cyber-physical systems, *Proceedings of the Fifteenth International Conference on Algorithms and Architectures for Parallel Processing*, pp. 785–798, 2015.

[4] J. Goh, S. Adepu, K. Junejo and A. Mathur, A dataset to support research in the design of secure water treatment systems, *Proceedings of the Eleventh International Conference on Critical Information Infrastructures Security*, pp. 88–99, 2016.

[5] iTrust Centre for Research in Cyber Security, Dataset Characteristics, Singapore University of Technology and Design, Singapore (`itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info`), 2020.

[6] iTrust Centre for Research in Cyber Security, Secure Water Treatment, Singapore University of Technology and Design, Singapore (`itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat`), 2020.

[7] K. Junejo and D. Yau, Data driven physical modeling for intrusion detection in cyber-physical systems, *Proceedings of the Singapore Cyber Security Conference*, pp. 43–57, 2016.

[8] J. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, California, 1993.

[9] D. Sutherland, A model of information, *Proceedings of the Ninth National Computer Security Conference*, pp. 175–183, 1986.

[10] M. Umer, A. Mathur, K. Junejo and S. Adepu, Integrating design and data centric approaches to generate invariants for distributed attack detection, *Proceedings of the Workshop on Cyber-Physical Systems Security and Privacy*, pp. 131–136, 2017.

[11] I. Witten, E. Frank and M. Hell, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers, San Francisco, California, 2011.