Delft University of Technology

Domain-Based Fuzzing for Supervised Learning of Anomaly Detection in Cyber-Physical Systems

Wijaya, Herman; Aniche, Maurício; Mathur, Aditya

DOI
10.1145/3387940.3391486

Publication date
2020

Document Version
Accepted author manuscript

Published in
ICSEW'20

Citation (APA)
Wijaya, H., Aniche, M., & Mathur, A. (2020). Domain-Based Fuzzing for Supervised Learning of Anomaly Detection in Cyber-Physical Systems. In *ICSEW'20: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (pp. 237-244). ACM. https://doi.org/10.1145/3387940.3391486

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Domain-Based Fuzzing for Supervised Learning of Anomaly Detection in Cyber-Physical Systems

Herman Wijaya
Singapore University of Technology and Design
Singapore
herman_wijaya@sutd.edu.sg

Maurício Aniche
Delft University of Technology
Delft, Netherlands
M.FinavaroAniche@tudelft.nl

Aditya Mathur
Singapore University of Technology and Design
Singapore
aditya_mathur@sutd.edu.sg

ABSTRACT

A novel approach is proposed for constructing models of anomaly detectors using supervised learning from the traces of normal and abnormal operations of an Industrial Control System (ICS). Such detectors are of value in detecting process anomalies in complex critical infrastructure such as power generation and water treatment systems. The traces are obtained by systematically "fuzzing", i.e., manipulating the sensor readings and actuator actions in accordance with the boundaries/partitions that define the system's state. The proposed approach is tested in a Secure Water Treatment (SWaT) testbed – a replica of a real-world water purification plant, located at the Singapore University of Technology and Design. Multiple supervised classifiers are trained using the traces obtained from SWaT. The efficacy of the proposed approach is demonstrated through empirical evaluation of the supervised classifiers under various performance metrics. Lastly, it is shown that the supervised approach results in significantly lower false positive rates as compared to the unsupervised ones.

KEYWORDS

fuzzing, domain testing, security, cyber physical system, supervised learning, anomaly detection

ACM Reference Format:
Herman Wijaya, Maurício Aniche, and Aditya Mathur. 2020. Domain-Based Fuzzing for Supervised Learning of Anomaly Detection in Cyber-Physical Systems. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3387940.3391486

1 INTRODUCTION

A Cyber-Physical System (CPS), as found in critical infrastructure, e.g., power generation and water treatment plants, is a complex system consisting of distributed computing elements such as sensors, actuators, and Programmable Logic Controllers (PLCs) that interact with the underlying physical processes. Compromising the software and/or the hardware components of such a CPS has the potential to cause a significant damage and service disruption by driving the actuators into states that exceed the physical constrains. Several incidents of attacks on CPS have been reported. In 2000, a disgruntled former employee of Australian's Maroochy Water Services took control of the organization's facilities and released one million litres of untreated sewage into a stormwater drain for days [26]. In 2011, hackers with Russian IP addresses managed to gain access to the network of a water utility in the US city of Illinois and caused damage on the distribution pump by turning it on and off quickly [21, 30]. In 2016, a water utility company with pseudonym Kemuri Water Company was compromised by a hacktivist group with ties to Syria [14, 25]. The group managed to gain access to the plant through a vulnerable web server and changed the amount of chemicals into the water supply which affected the water treatment and production capabilities [13, 14, 25, 28]. The above headlines serve as evidence that fast, accurate, and robust anomaly detection system is needed to protect against potential cyber attacks. With access to the data traces, logs, and CPS model, the physical effects due to the attacks could potentially be detected before any damage is done.

Constructing models of CPS that are accurate enough in practice is a notoriously difficult task due to the tight integration of algorithmic control and complex physical processes. The emergence of Artificial Intelligence (AI) and Machine Learning (ML) offers a practical way of building anomaly detectors for CPS that might be accurate enough to be practical. Most reported algorithms using unsupervised learning result in high false positive rates [19, 20]. On the other hand, obtaining labelled data traces for supervised learning is more difficult, especially for the attack data. Most works using supervised learning rely on either using plant simulator [7] or through manual effort of attacking the physical plant [11] to generate the attack data (i.e., abnormal traces).

The objective of this work is to present a novel approach that uses a software testing technique, i.e., domain testing, to simulate the system under attacks by systematically manipulating sensor readings and actuators actions in accordance with the boundary or partition that defines the system's state. It is started with identifying the equivalence class partitions of different CPS components based on process specification. Combinations of these equivalence classes are used to attack CPS components to generate the abnormal operational traces for learning the CPS anomaly detection model.

Overall, six supervised classifiers that learned from from the normal operational traces and the fuzzing-induced abnormal traces were evaluated. The results demonstrate the effectiveness of the proposed approach to generate abnormal data traces for supervised learning of CPS anomaly detectors as confirmed by empirical evaluation under various performance metrics. Additionally, the results

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3391486

Figure 1: SWaT testbed at SUTD

exhibit a significantly lower false positive rate when compared with that from detectors generated using unsupervised ML. The top performing supervised classifiers in our experiments generate no, or remarkably low, false positives while achieving high precision and recall.

2 THE SECURE WATER TREATMENT (SWAT) TESTBED

The Secure Water Treatment (SWaT) testbed, located at the Singapore University of Technology and Design (SUTD), is a fully operational scaled down water treatment plant with a smaller footprint that is capable of producing five gallons per minute of treated water. This testbed is a replica of bigger water treatment plants such as those found in cities (see Figure 1).

Its main purpose is to enable experimentally-validated research in the design of security and safety of a CPS. SWaT has six main processes that correspond to the physical and control components of a water treatment infrastructure (we refer the reader to [1] for a better understanding of the infrastructure).

All the processes are interdependent, with each process is dependent on the previous one. The cyber portion of SWaT consists of layers communication networks, Programmable Logic Controllers (PLCs), Human Machine Interfaces (HMIs), a Supervisory Control and Data Acquisition (SCADA) workstation, and a historian server. Data from the sensors are available to the SCADA system and recorded by the historian for subsequent analysis. A separate PLC is dedicated for each of the six processes. Each of these PLCs has a pair of a redundant standby PLC for high availability.

The communication in SWaT uses multi-layer communication links between different switches and routers. This communication can take place over either Wi-Fi or Ethernet link running various industrial communication protocols. The sensor readings, as well as actuator commands, are all communicated to the PLC over the wired or wireless communication links. These protocols are vulnerable to network attacks.

In this work, we have successfully launched Man-in-the-Middle (MITM) attacks to inject false sensor readings and actuator commands into the PLC of the raw water process using Pycomm communication library. We use the attacks against the raw water process (stage 1 of the overall pipeline) as a proof of concept of our domain-based fuzzing approach for supervised anomaly detection. Findings from this process can be extended to the other processes in the future.

3 RELATED WORK

One fundamental challenge in supervised learning for anomaly detection arises from the lack of comprehensive training data from the system under attack. Specifically, the attacks on real-world systems are sparse and unsystematic. Moreover, not many studies are reported for supervised learning for anomaly detection as most of them focus on the unsupervised learning approach. The existing works in supervised learning were done either using plant simulator [7] or through manual effort of attacking the physical plant [11] to generate the attack data.

Chen et al. [7] presented the idea of using code mutation to systematically generate the abnormal data traces for supervised learning. The approach was inspired by mutation testing, a fault-based software testing which deliberately introduces errors (small, syntactic transformations called mutations) which is commonly used for assessing the quality of the software test cases. The CPS could be simulated under the original PLC code and different "mutants" from the code mutation to collect sets of normal and abnormal sensor data traces for supervised learning.

Their framework was applied on the simulator of the SWaT testbed to run each mutant code at the same initial configuration for 30 minutes of simulator time unit which is shorter than the physical time. However, while traces generated by the simulator have a somewhat high degree of precision, the physical plant is governed by the law of physics, such as the dynamics of water that affects the reading of the water level sensor, which makes obtaining the accurate abnormal traces to follow the framework becomes more challenging. The time required to run the mutant PLC code on the physical plant is longer than running on the simulator time unit. Given the number of mutants is potentially infinite and considering that each mutant code needs to run for 30 minutes, data collection is too onerous to apply on the physical plant. Due to this constraint, it might be more effective and worth the effort of generating the abnormal traces through manual attack.

In contrast to Chen et al. [7], the work done by Junejo and Goh was directly applied on the physical plant of SWaT [11]. They presented a realistic work of behaviour-based anomaly detection using supervised learning approach on a real CPS plant with all the physical and control components in place. The behaviour-based approach learns the model of the physical process of SWaT from a behaviour perspective to detect and classify cyber-attacks that alter the physical layer.

The strength of their approach is that it learns directly from the empirical data of actual operational data traces, which renders its robustness against incorrect vendor specifications. In particular, this robustness has allowed authors to rectify or more tightly bound the system's real operational limits in comparison to using the vendor's specifications. For example, it is observed that the water level in a tank does not exceed 800 mm under a normal operation, whereas the vendor's specification states an upper bound of 1100 mm and encodes this value in the PLC code.

In addition, Junejo and Goh also evaluated nine supervised learning classifiers using SWaT-generated data from ten different types of attacks [11] which follow the attack model in [2]. They then compared the performance in terms of time to detect an attack, types of attacks that are successfully detected or not, and computation time to build the required detection model. Their work demonstrates that the best classifiers do not only detect almost all the attacks successfully, but they also detect them earlier than the specification-based approach.

Despite the sophisticated approach of supervised learning, however, generating the abnormal data traces was done without automation. The attacks were performed manually by changing the sensor value and actuator information to mislead the PLC. Also, the validity of the classifiers could be questionable as each features vector for the supervised learning consists of sensors reading and actuators status at one particular second which might not give sufficient insight due to lack of meaningful pattern. Nevertheless, the results of their analysis yield valuable insights into the performance potential of supervised behaviour-based anomaly detection.

4 THE PROPOSED APPROACH

The supervised learning approach of behaviour-based anomaly detection yields a performance potential in terms of robustness and accuracy as compared to the unsupervised approach that is known to have higher false positive rates [19, 20]. However, supervised learning requires increased effort to obtain realistic abnormal data traces.

In a small-scale physical plant that consists of a few subprocesses, it is feasible to generate realistic abnormal data traces by manually manipulating the sensor measurements and actuator states. However, for a complex plant with, say, hundreds of subprocesses, attacking sensors and actuators in different processes needs to be done systematically. It could be challenging to manually launch attacks on sensors and actuators across different processes.

To overcome the challenges stated above, a novel approach is proposed for supervised anomaly detection using domain-based fuzzing with equivalence partitioning. This approach simulates the system under attacks by systematically introducing anomalies on sensor readings and actuator actions according to the boundaries or partitions that define the system's state. The approach is inspired by a software testing technique known as domain testing [12]. Domain testing is a type of functional testing that enables the generation of tests to test the program behaviour such that the output of the program is tested with a minimal number of inputs. Domain testing is commonly used as sampling strategy for choosing a few test cases from a nearly infinity of candidates.

The strategy of domain testing is known under several names, such as equivalence partitioning and boundary value analysis. Nevertheless, the essence of it is to partition a domain (that is possibly infinite) into a few sub-domains, known as equivalence classes, such that the program under test evaluates differently and then select a few representatives from each sub-domain as test cases. In the context of a CPS, the domain consists of all possible sensor readings and actuator actions that serve as inputs to the PLC controlling the physical process.

In a nutshell, the proposed approach works as follows: (1) **Threat modelling**: Establish a realistic attack model from a set of attack domains and attacker models; (2) **Equivalence class partitioning**: Applied to CPS components to create the attack domain; (3) **Data collection**: Collect normal traces from the CPS; (4) **Sensor and actuator fuzzing**: Obtain abnormal traces; and (5) **Machine learning**: Learn a classification model from the given dataset.

We apply supervised learning of anomaly detector using domain-based fuzzing with equivalence partitioning on the raw water process (stage 1) of the SWaT testbed as a proof of concept by answering to the following research questions.

RQ1 How well does the domain-based fuzzing with equivalence class partitioning perform when using supervised learning to create anomaly detectors for a CPS?

RQ2 How well do the supervised classifiers perform in detecting anomalies as compared to those detectors obtained from unsupervised learning?

4.1 Threat Modelling

When conducting this threat modelling, we follow the methodology proposed by Adepu and Mathur [3], which consists of 5-stage process to derive the CPS attacks. As a result, we establish the attack domain, attacker, and attack models which will be used for the subsequent domain-based fuzzing.

4.1.1 ***Attack domain model.*** We start off by the identification of items to establish our attack domain. As outlined in [2] and [3], an attack domain is a triple (Cm, Pr, Pe) that consists of three finite sets which are referred as component set (Cm), property set (Pr) and performance set (Pe) respectively. The component set includes CPS elements which can be physical, e.g., level sensor, pump, motorized valve, etc., or cyber, e.g., PLC code, communication network, etc. The property set includes the physical properties of the resulting product being produced or controlled by the CPS, such as the water pH and Oxidation Reduction Potential (ORP). Lastly, the performance set includes one or more performance character of a plant, such as the amount of treated water produced.

However, considering that the raw water process is one of the less complex processes in SWaT with no physical properties affecting the process, as well no performance measure, our attack domain model can be simplified into the component set only, as shown in Table 1.

4.1.2 ***Attacker model.*** According to Adepu and Mathur [2, 3], the attacker model is a pair of (I, DM) where I is a finite set of intents and DM an attack domain model. Definition of intent is a goal or an objective of the attacker. Some possible intents may include damage, learn, and alter. In general, an intent can be considered as a function that is applied by an attacker to one or more elements of an attack domain that defines a CPS.

Based on the attack domain which we have earlier and generalization of intent, Table 2 shows a derivation of our attacker model.

4.1.3 ***Attack model.*** The formalization of attack model requires us to have our attack domain model and attacker model established [2, 3]. Consider an attacker model $AR_{SWaT} = (I, DM)$ for SWaT, where $DM = (Comp, Pr, Pe)$, the attack model for SWaT is denoted as AM_{SWaT} in a form of sextuple (M, G, D, P, S_0, S_e), where M is potentially infinite set of procedures to launch attacks, $G \subseteq I$ is a finite set of attacker intents, D is the domain model for the attacks derived from the domain model DM of C, $P \subseteq Cm$ is a finite set of attack points, S_0 and S_e are possibly infinite sets of

Table 1: Attack domain model of SWaT stage 1

Component	Type	Description
$LIT101$	Sensor	Level indicator transmitter of the raw water tank
$FIT101$	Sensor	Flow indicator transmitter for water inflow from the city water supply
$MV101$	Actuator	Motorized valve for city water supply to the raw water tank
$P101$ / $P102$	Actuator	Pumps to transfer water to the next stage of the plant
$P601$	Actuator	Pump in the RO permeate (*stage 6*) to transfer clean water to the raw water tank
$T101$	Storage	Raw water storage tank
$PLC1$	Controller	PLC controlling the raw water storage of the plant

Table 2: Attacker model of SWaT stage 1

Components	Type	Attacker Intent
$LIT101$	Sensor	Learn, alter
$FIT101$	Sensor	Learn, alter
$MV101$	Actuator	Damage, alter
$P101$ / $P102$	Actuator	Damage, alter
$P601$	Actuator	Damage, alter
$T101$	Storage	Damage
$PLC1$	Controller	Alter

states of C that denote the possible start and end states which is of the interest to the attacker.

As this paper serves as a proof of concept that equivalence class partitioning can be used for fuzzing a CPS to generate abnormal data traces for supervised learning of anomaly detection, we limit the scope of the attack points to include only $LIT101$, $FIT101$, and $MV101$ for simplification reason. Based on our intention to generate attacks for the purpose of supervised learning approach of behaviour-based anomaly detection, we define our attack model for the raw water process as shown in Table 3.

4.2 Equivalence Class Partitioning (ECP)

Domain testing is a form of functional testing where the program is viewed as a function and tested by feeding some inputs and evaluating its outputs [12]. Fundamentally, functional testing can be done through exhaustive testing that includes all possible inputs to the system. However, this is highly impractical. Due to this reason, the concept of *domain testing* was introduced. Instead of testing the program with all possible inputs, the input domain is divided into a set of equivalence classes with an assumption that if the program exhibit a certain behaviour for a certain value then it eventually exhibit the same behaviour for all other values within the same class [17]. This assumption allows the tester to select exactly one test case from each equivalence class resulting in a test suite of exactly N test cases. Therefore, the essence of domain testing is to partition a domain of possibly infinite values into couple of subdomains or equivalence classes which the program evaluates differently and then select some representatives from each subdomain to be tested [12].

For a CPS, the domain is the sensors readings and the actuators actions which serve as inputs to the PLC controlling the physical environment. The set of equivalence classes are defined based on the specification of a particular process. In the case of the raw water process of SWaT, there are two sensors which is the level indicator transmitter ($LIT101$) of the water tank and the flow indicator transmitter ($FIT101$) of the inflow to the water tank. For the actuators, there are the motorized valve ($MV101$) which regulate the inflow to the water tank and two pumps which are configured in active-standby mode to regulate the outflow to the next process.

The specification of the raw water process marks the water level sensor reading into certain predefined constants of LL (Very low), L (Low), H (High) and HH (Very high) as 250 mm, 500 mm, 800 mm and 1000 mm, respectively. In normal operation, the motorized valve $MV101$ should open when the water level reaches L and close when it reaches H. When the water level in the ultra-filtration process is low, the PLC will turn on the pump $P101$ to transfer water to the ultra-filtration tank in the subsequent process. Regardless of whether the ultra-filtration process needs water or not, the pump $P101$ should stop when the water level reaches LL. Likewise, the reading of $FIT101$ should indicate a normal operating flow between 0.0 to 4.40 m^3/h.

With the same reason as mentioned on the earlier part, only $LIT101$, $FIT101$, and $MV101$ are included into the possible combinations of the equivalence class partitions. Therefore, based on the process specification, the water level and flow indicator sensors readings, and the motorized valve actions are partitioned into equivalence classes (details can be found in the online appendix [29]). There are 4 equivalence classes in $LIT101$, 3 equivalence classes in $FIT101$, and 2 equivalence classes in $MV101$. Thus, up to 24 possible combinations of the equivalence class partitions can be obtained, as shown in Table 4.

It is observed that all the attacks defined in Table 3 correlate to at least one combination of equivalence classes from Table 4 (A1: #19, A2: #4, A3: #14, A4: #17, #18, A5: #7, A6: #22). Therefore, this establishes our confidence that 24 possible combination of equivalence classes for the dataset generation would cover the threat model as well as some other anomalies which could be due to faulty sensors or actuators.

4.3 Traces Generation

Supervised learning requires two sets of labelled data traces – normal and attack. The normal traces are generated by running the SWaT under normal conditions while the abnormal traces are generated by running a fuzzer. Python scripts are used to create the fuzzer. The fuzzer uses the Pycomm library developed by Ruscito [24] to communicate with the Allen-Bradley ControlLogix PLC of SWaT (specifically, in this experiment, the $PLC1$ which controls the raw water process). Pycomm is used to emulate realistic attacks against level sensor $LIT101$, motorized valve $MV101$, and the flow rate indicator $FIT101$, by systematically modifying the sensor readings

Table 3: Attack model of SWaT stage 1

Attack ID	Attack Procedure (M)	Intention (G)	Attack Point (P)	Start state of the system (S_0)
A1	Increase $LIT101$ value to above H	Stopping inflow	$LIT101$	Water level is between L and H and $MV101$ is OPEN
A2	Decrease $LIT101$ value to below LL	Stopping outflow	$LIT101$	Water level is between L and H and $P101$ is ON
A3	Change $FIT101$ value to zero	Falsify inflow rate	$FIT101$	$MV101$ is OPEN
A4	Change $FIT101$ value to less than or beyond the normal range (< 0.0 or >= 4.40)	Falsify inflow rate to give impression of faulty sensor or valve leakage	$FIT101$	$MV101$ is OPEN or CLOSED
A5	Override $MV101$ to close	Underflow tank	$MV101$	Water level is between L and H
A6	Override $MV101$ to open	Overflow tank	$MV101$	Water level is between L and H

Table 4: Possible combinations of the equivalence classes

#	$LIT101$	$MV101$	$FIT101$	Anomaly
1	Underflow	Closed	Zero	Yes (*LIT, MV*)
2	Underflow	Open	Zero	Yes (*LIT, FIT*)
3	Underflow	Closed	Flowing	Yes (*LIT, MV, FIT*)
4ˆ	Underflow	Open	Flowing	Yes (*LIT*)
5	Underflow	Closed	Abnormal	Yes (*LIT, MV, FIT*)
6	Underflow	Open	Abnormal	Yes (*LIT, FIT*)
7ˆ	Low	Closed	Zero	Yes (*LIT, MV*)
8	Low	Open	Zero	Yes (*LIT, FIT*)
9	Low	Closed	Flowing	Yes (*LIT, MV, FIT*)
10	Low	Open	Flowing	Yes (*LIT*)
11	Low	Closed	Abnormal	Yes (*LIT, MV, FIT*)
12	Low	Open	Abnormal	Yes (*LIT, FIT*)
13	Normal	Closed	Zero	Yes (*LIT*)
14ˆ	Normal	Open	Zero	Yes (*FIT*)
15	Normal	Closed	Flowing	Yes (*FIT*)
16	Normal	Open	Flowing	Yes (*LIT*)
17ˆ	Normal	Closed	Abnormal	Yes (*FIT*)
18ˆ	Normal	Open	Abnormal	Yes (*FIT*)
19ˆ	High/Overflow	Closed	Zero	Yes (*LIT*)
20	High/Overflow	Open	Zero	Yes (*LIT, MV, FIT*)
21	High/Overflow	Closed	Flowing	Yes (*LIT, FIT*)
22ˆ	High/Overflow	Open	Flowing	Yes (*LIT, MV*)
23	High/Overflow	Closed	Abnormal	Yes (*LIT, FIT*)
24	High/Overflow	Open	Abnormal	Yes (*LIT, MV, FIT*)

in the PLC memory and overriding the commands sent to the actuators following the combinations of equivalence class partitioning as in Table 4. This fuzzer is referred as Equivalence Class Partitioning (ECP) fuzzer. The generated dataset used in this work is available in the online appendix [29].

4.3.1 *Normal traces*.

The normal traces were generated on 29th May 2019 while running SWaT for 4.5 hours under normal conditions. The historian records all sensor measurements and actuator states every one second. Hence, a total of 16,200 data points are in the normal trace.

4.3.2 *Abnormal traces*.

Prior to launching attacks using the ECP fuzzer to generate the abnormal traces, preliminary attacks were launched only on the level sensor $LIT101$ to determine the time required by the system to completely return to normal state from an anomalous state. This time is referred as the "rest interval". It is crucial to have a sufficient rest interval before launching the next attack. The abnormal trace data will be used for supervised learning and thus we do not want to launch another attack when the system is still in an anomalous state for accuracy purposes, as different attacks are generated by different equivalence class partitions of different sensors and actuators.

A few preliminary attacks were launched by changing the $LIT101$ readings to a constant (randomly chosen) anomalous value (between LL to L or H to HH) for 30 seconds, restoring it to the actual value, and giving the system time to recover before launching another attack. We experimented with different rest intervals of 1, 3, and 5 minutes. It turned out that the rest interval of 1minute is sufficient to allow the system to completely return to its normal state before another attack is launched. Based on this finding, rest interval of 1 minute was used in the ECP fuzzer.

The abnormal traces were generated on 19th July 2019 by running the ECP fuzzer for 3-hours while SWaT was running in normal state. The ECP fuzzer generates 24 different attacks on $LIT101$, $MV101$, and $FIT101$ following the combinations of equivalence classes of different sensors and actuators (see Table 4) with each attack lasting for 30 seconds followed by 60-seconds of rest interval before another attack is launched. This resulted in 10,800 datapoints (seconds) in the abnormal trace.

4.3.3 *ECP fuzzer*.

The ECP fuzzer extends the functionality provided by the Pycomm library for overwriting different tags that store the sensor readings and actuator status in the PLC memory.

4.3.4 *Features*.

Following is a list of the measurements and states used by the ECP fuzzer to generate attacks. (i) Motorized valve MV101 (integer; 0: TRANSITION, 1: CLOSED, 2: OPEN) (ii) Water level sensors $LIT101$ and $LIT301$ of the raw water tank (real) (iii) Flow indicator sensor $FIT101$ (real) (iv) Pumps $P101$, $P101$, and $P601$ (integer; 1: OFF, 2: ON)

4.4 Learning the Classifiers

Our machine learning workflow starts with pre-processing of the dataset to obtain the desirable features vector and splitting the dataset for training and testing with proportion of 60:40 respectively. The training dataset is used to train different classifiers such as Multilayer Perceptron [23], Support Vector Machine [4], Logistic Regression (LR), Decision Tree [27], Random Forest [6], and Extremely Randomized Trees [9].

The performance of each classifier is evaluated against the test dataset using various evaluation metrics such as accuracy, precision, recall, and F1. Stratified k-fold cross-validation was also used to validate the classification model against generalization error. The implementation of the entire training process is carried out using Python with the support of Scikit-learn [22], Keras [8], and Pandas [18].

4.4.1 ***Pre-processing of the dataset.*** The dataset consists of both normal and abnormal traces extracted from the historian that collects data from all sensors and actuators every second. As the scope of this study is only within the raw water process, only the features that consist of sensor readings and actuator states status involved in the raw water process, were extracted. Nevertheless, feature reduction (i.e., dimensionality reduction) was performed, as doing so would lead to obvious benefits in terms of reduction of computational resources such in memory usage and CPU time. As the dataset consists of a small number of features, feature reduction can be done manually by removing the components that are not controlled by $PLC1$, or are not part of the attack, or do not influence the control decision of $PLC1$. Thus, seven features were reduced to five by removing $P601$ and $P102$. $P601$ was removed as it is controlled by $PLC6$ in *stage 6* and neither part of the attack influences the control decisions of $PLC1$. Pump $P102$ is not part of the attack despite being controlled by $PLC1$; it serves as a backup when $P101$ fails. During the experiments, no failures of $P101$ were encountered and therefore $P102$ was safely removed from the set of features.

The normal and abnormal traces consist of 16,200 and 10,800 rows, respectively. However, within the abnormal data itself, not all rows correspond to attack data. Recall that the ECP fuzzer launches an attack on a set of sensors and actuators that last for 30 seconds and followed by a rest interval of 60 seconds to allow the system to completely stabilize from an anomalous state. For every 90 seconds of the abnormal traces, the actual attack only occurs for 30 seconds of actual attack and the remaining 60 seconds consist of both anomalous state after the attack and small portion of normal state before the next attack comes in. Due to this reason, the dataset is pre-processed to obtain a desirable feature vector, such that each feature vector on both normal and abnormal datasets consists of every 90 seconds of data (i.e., a vector of 5 features times 90 data points). The following matrix shows the final representation of the feature vector for our machine learning classifiers.

$$\begin{bmatrix} (MV101 \quad FIT101 \quad LIT101 \quad P101 \quad LIT301)_{t=1} \\ (MV101 \quad FIT101 \quad LIT101 \quad P101 \quad LIT301)_{t=2} \\ \vdots \\ (MV101 \quad FIT101 \quad LIT101 \quad P101 \quad LIT301)_{t=90} \end{bmatrix}$$

After obtaining the desirable feature vector, the dataset is split into a training and a testing set. The dataset for training and testing is split in the proportion of 60:40, respectively. After splitting, the dataset contains 108 normal and 72 abnormal data points for training, and 72 normal and 48 abnormal data points for testing.

Optional feature scaling is included to scale the dataset depending on the classification algorithms applied. The feature scaling performs standardization of data values that vary in magnitude and range. It is applied only for the MLP and SVM classifiers. As the feature vector, particularly the abnormal data, contains some outliers of normal system state due to the rest interval before launching another attack which might affect the distance calculation in the MLP and SVM objective functions, the robust scaler is chosen as it is more robust to estimate for the centre and range of the data. For each feature, the robust scaler works by subtracting with the median and dividing it by the interquartile range that makes it more robust against the outliers [10, 15].

4.4.2 ***Supervised classification models.*** The training dataset is to train different classifiers from six different classification algorithms. Three algorithms are discriminative classifiers, namely Multilayer Perceptron (MLP) [23], Support Vector Machine (SVM) [4] and Logistic Regression (LR). The other three are tree-based classifiers, namely Decision Tree (DT) [27], Random Forest (RF) [6] and Extremely Randomized Trees (ExtraTrees) [9].

4.4.3 ***Unsupervised models for performance comparison.*** To compare the performance of the supervised models against the unsupervised approach in detecting anomaly the results from two unsupervised novelty/outlier detectors, the One-Class Support Vector Machine (OC-SVM) [5] and Isolation Forest (IF) [16], are shown. These unsupervised models are trained only with the normal data from the same training dataset used by the supervised classifiers and tested using the normal data of test dataset and the entire abnormal data (of both the training and test dataset).

4.4.4 ***Evaluation.*** Confusion matrix is used to represent the performance of each classifier with some basic information such as True Positive, False Positive, False Negative, and True Negative counts. The following performance evaluation metrics were used in conjunction with the confusion matrix to evaluate the classifiers: accuracy, error rate, precision, recall, and F1 scores (detailed explainations can be found in the online appendix [29]. These are the commonly accepted metrics to summarize and compare the classifiers performance.

All models were also validated using stratified k-fold cross-validation to assess how well the classifiers generalize the dataset. The k-fold cross-validation involves dividing the dataset into k partitions or folds of the same size, training on k-1 partitions and testing on the remaining partition, repeating with respect to different validation partitions. In stratified k-fold cross-validation, the data was arranged such that each fold has a good representation of both normal and abnormal vectors from of the entire dataset to ensure that one class of data is not overrepresented especially when the target variable is unbalanced.

The models were trained on a laptop with 4.1GHz Intel Core i7-8750 processor, 16GB RAM and Nvidia GeForce GTX 1060 Max-Q with 6GB VRAM.

Table 5: Performance summary of the supervised and unsupervised classifiers.

Classifiers	Accuracy	Precision	Recall	F1
MLP	97.50	95.92	97.92	96.91
SVM-RBF	96.67	92.31	**100.00**	96.00
SVM-LNR	87.50	94.60	72.92	82.35
LR	90.83	89.36	87.50	88.42
DT	94.17	90.20	95.83	92.93
RF	98.33	**100.00**	95.83	97.87
ExtraTrees	**99.17**	97.96	**100.00**	**98.97**
OC-SVM	**91.15**	**87.59**	**100.00**	**93.39**
IF	74.48	79.83	79.17	79.50

5 RESULTS

The performance of our supervised (MLP, SVM, DT, RF and ExtraTrees) and unsupervised classifiers (OC-SVM and IF) are shown in Table 5. The model parameters and confusion matrix of each classifier can be found in the online appendix [29].

5.1 RQ1: How well does the domain-based fuzzing with equivalence class partitioning perform when using supervised learning to create anomaly detectors for a CPS?

It is observed that majority of the supervised classifiers show accuracy scores above 90% against the test dataset, except for the SVM-linear and LR, which show lower accuracy. It is also observed that most classifiers have F1 scores which are almost as high as their accuracy scores with some observable gaps between F1 and accuracy are on the SVM-linear, LR and DT. Moreover, SVM-linear and LR score below 90% for F1. The accuracy and F1 scores on the other classifiers such as MLP, SVM-RBF, DT, RF and ExtraTrees noticeably outperform the SVM-linear and LR which are linear models. This means that the features vectors in the dataset are high dimensional and non-linearly separable by linear classifiers.

Only the SVM-RBF and ExtraTrees classifiers manage to accurately classify all the abnormal traces within the test dataset, while the Random Forest set as the only classifier with zero false alarm. The SVM-linear and LR occupy the bottom two with significantly higher false alarm and misdetection rates.

Stratified k-fold cross-validation (with k=5) was applied to obtain k different F1 scores for each classifier, using k different training-test partitions of the dataset. The cross-validated F1 score is the average accuracy of five different models on each classifier, each is obtained by partitioning the dataset set into five, training on four partitions, and validating on the fifth, then repeating with a different validation partition until all the partitions are covered. This is to assess how well the supervised classifiers generalise the dataset (the Tables can be found in the online appendix [29]). The ExtraTrees classifier again comes out as the top performer in cross-validated F1 with score of 98.28%, followed by the MLP at 97.97%, while the two linear models (SVM-linear and LR) are the worst performers with significantly lower scores of around 80%.

Moreover, it can be seen from the F1 score of each fold that these linear models are prone to generalization error. This might due to the features vectors in the dataset are highly dimensional and thus non-linearly separable. Intuitively, the linear classifiers are insufficient because the features vector contains various sensors reading and actuators status in SWaT that are correlated in complicated ways beyond the capability of these linear classifiers.

Summary of RQ1: Results clearly suggest the efficacy of domain-based fuzzing with equivalence class partitioning for supervised learning of anomaly detection in CPS environment. Moreover, ExtraTrees outperforms the other classifiers in almost all metrics and it generalises the dataset very well.

5.2 RQ2: How well do the supervised classifiers perform in detecting anomalies as compared to those detectors obtained from unsupervised learning?

It is observed that the OC-SVM is a clear winner against the Isolation forest in all performance metrics that include accuracy, false positive and false negative rates, precision, recall and F1 scores. The superior performance of the OC-SVM is due to the models was initially designed as a novelty detection algorithm that is specifically trained on a dataset which is free from anomaly, while the IF works by isolating the outliers or anomalies from the dataset instead of profiling the normal datapoints.

Nevertheless, despite the OC-SVM's superior performance as an unsupervised anomaly detection model, it exhibits noticeably lower precision score as compared to the supervised classifiers.

The precision score shows a ratio of the correctly detected abnormal traces versus all traces that are detected as abnormal. Comparing the false positive rates of the supervised classifiers and the unsupervised models, it can be seen that the OC-SVM's lower precision score is due to a significantly higher false positive rate as compared to the supervised classifiers. Nevertheless, it has a perfect recall rate which shows that it has zero false positive or misdetection of abnormal traces, similar to the SVM-RBF and ExtraTrees from the supervised classifiers.

Summary of RQ2: Intuitively, supervised classifiers generally still hold better performance on detection accuracy with significantly lower false positive rates in comparison to the unsupervised models.

6 THREATS TO VALIDITY

Internal validity. (1) The amount of data used to train our models was not large (see Section 4.4.1). However, we are certain that our models are not memorising them as the falsified sensor values and actuators status are randomly generated while satisfying the "abnormal" boundary of the equivalence partitions. The resulting dataset were also validated using 5-fold cross-validation to check if the learned models have good generalization ability. The results of the k-fold validation confirms our confidence that the learned models are not memorizing the attacks. (2) A 90 seconds window of "normal + attack" situation was chosen to generate our dataset (in a way that it consists of 10s of rest + 30s of attack + 50s of rest). We conjectured that this window would better represent a real-world situation, where the system is under control and suddenly receives an (yet not known) attack. We finally chose 90s after short experimentation. Future work should explore the impact of different time windows on the performance of the models.

External validity. Despite the results showing that the supervised approach outperforms the unsupervised approach, the supervised approach is still not as practical on real-world CPS. It is not an option to simulate attacks with the purpose of obtaining abnormal traces on real CPS plants, such as water treatment plant or power grids, given that most of them are critical infrastructures where their operational availability, performance and stability are very sensitive. Future work needs to focus on testing the models built in the testbeds, such as SWaT, in realistic CPS attacks.

7 CONCLUSION

Having reliable and realistic datasets that contain both the normal and abnormal operational traces of a CPS is a fundamental challenge in behaviour-based supervised anomaly detection. In this study, a novel approach is proposed to address this challenge by using a software testing technique, i.e., domain testing using equivalence class partitioning, to systematically derive attacks for attacking the sensors and actuators of the raw water process in the SWaT testbed. This testbed is a replica of real-world water purification plant. These abnormal traces are combined with the traces of normal operation to form a labelled dataset that can be used to train different supervised classifiers.

The reported study demonstrates the effectiveness of using domain-based fuzzing with equivalence class partitioning to generate abnormal data traces for supervised learning of CPS anomaly detection, as confirmed by empirical evaluation under various performance classification metrics. It also shows that supervised learning approach of anomaly detection results in significantly lower false positive rates that intuitively contributes to higher overall accuracy and recall scores of the supervised classifiers as compared to the unsupervised models which have higher false positive rates.

ACKNOWLEDGMENTS

This work is supported by the National Research Foundation, Singapore under its National Satellite of Excellence in Design Science and Technology for Secure Critical Infrastructure programme (NSoE DeST-SCI) (Award No. NRF2018NCR-NSOE005-0001).

REFERENCES

[1] 2018. *Secure Water Treatment (SWaT) Testbed.* Technical Report. https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_swat/
[2] Sridhar Adepu and Aditya Mathur. 2016. An Investigation into the Response of a Water Treatment System to Cyber Attacks. In *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, Vol. 2016-March. IEEE Computer Society, 141–148. https://doi.org/10.1109/HASE.2016.14
[3] Sridhar Adepu and Aditya Mathur. 2016. Generalized Attacker and Attack Models for Cyber Physical Systems. In *Proceedings - International Computer Software and Applications Conference*, Vol. 1. IEEE Computer Society, 283–292. https://doi.org/10.1109/COMPSAC.2016.122
[4] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 2004. A training algorithm for optimal margin classifiers. Association for Computing Machinery (ACM), 144–152. https://doi.org/10.1145/130385.130401
[5] Abdenour Bounsiar and Michael G. Madden. 2014. One-class support vector machines revisited. In *ICISA 2014 - 2014 5th International Conference on Information Science and Applications*. IEEE Computer Society. https://doi.org/10.1109/ICISA.2014.6847442
[6] Leo Breiman. 2001. *Random Forests.* Technical Report. 5–32 pages.
[7] Yuqi Chen, Christopher M. Poskitt, and Jun Sun. 2018. Learning from Mutants: Using Code Mutation to Learn and Monitor Invariants of a Cyber-Physical System. In *Proceedings - IEEE Symposium on Security and Privacy*, Vol. 2018-May. Institute of Electrical and Electronics Engineers Inc., 648–660. https://doi.org/10.1109/SP.2018.00016
[8] François Chollet and Others. 2015. Keras: The Python Deep Learning Library. https://keras.io. https://keras.io
[9] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning* 63, 1 (apr 2006), 3–42. https://doi.org/10.1007/s10994-006-6226-1
[10] Jeff Hale. 2019. Scale, Standardize, or Normalize with Scikit-Learn - Towards Data Science. https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02
[11] Khurum Nazir Junejo and Jonathan Goh. 2016. Behaviour-Based Attack Detection and Classification in Cyber Physical Systems Using Machine Learning. Association for Computing Machinery (ACM), 34–43. https://doi.org/10.1145/2899015.2899016
[12] C. Kaner. 2004. Teaching domain testing: a status report. Institute of Electrical and Electronics Engineers (IEEE), 112–117. https://doi.org/10.1109/csee.2004.1276519
[13] Eduard Kovacs. 2016. Attackers Alter Water Treatment Systems in Utility Hack: Report | SecurityWeek.Com. https://www.securityweek.com/attackers-alter-water-treatment-systems-utility-hack-report
[14] John Leyden. 2016. Water treatment plant hacked, chemical mix changed for tap supplies • The Register. https://www.theregister.co.uk/2016/03/24/water{_}utility{_}hacked/
[15] Jovian Lin. 2017. Feature Scaling - JovianLin.io. https://jovianlin.io/feature-scaling/
[16] Fei Tony Liu, Kai Ming Ting, and Zhi Hua Zhou. 2008. Isolation forest. In *Proceedings - IEEE International Conference on Data Mining, ICDM.* 413–422. https://doi.org/10.1109/ICDM.2008.17
[17] Aditya Mathur. 2013. *Foundations of Software Testing* (2nd ed.). Addison-Wesley Professional. https://learning-oreilly-com.tudelft.idm.oclc.org/library/view/foundations-of-software/9788131794760/xhtml/chapter003.xhtml
[18] Wes McKinney and Others. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Vol. 445. 51–56.
[19] Patric Nader, Paul Honeine, and Pierre Beauseroy. 2014. Lp-norms in one-class classification for intrusion detection in SCADA systems. *IEEE Transactions on Industrial Informatics* 10, 4 (nov 2014), 2308–2317. https://doi.org/10.1109/TII.2014.2330796
[20] Patric Nader, Paul Honeine, and Pierre Beauseroy. 2014. Mahalanobis-based One-Class Classification. *IEEE International Workshop on Machine Learning for Signal Processing* (2014).
[21] BBC News. 2011. Hackers 'hit' US water treatment systems - BBC News. https://www.bbc.com/news/technology-15817335
[22] Fabian Pedregosa, Vincent Michel, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Jake Vanderplas, David Cournapeau, Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Bertrand Thirion, Olivier Grisel, Vincent Dubourg, Alexandre Passos, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830. https://scikit-learn.org
[23] Hassan Ramchoun, Mohammed Amine, Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil. 2016. Multilayer Perceptron: Architecture Optimization and Training. *International Journal of Interactive Multimedia and Artificial Intelligence* 4, 1 (feb 2016), 26. https://doi.org/10.9781/ijimai.2016.415
[24] Agostino Ruscito. 2017. Pycomm. https://github.com/ruscito/pycomm
[25] Marry-Ann Russon. 2016. Hackers hijacking water treatment plant controls shows how easily civilians could be poisoned | International Business Times. https://www.ibtimes.co.uk/hackers-hijacked-chemical-controls-water-treatment-plant-utility-company-was-using-1988-server-1551266
[26] Jill Slay and Michael Miller. 2007. Lessons Learned from the Maroochy Water Breach. In *IFIP International Federation for Information Processing*, Vol. 253. 73–82. https://doi.org/10.1007/978-0-387-75462-8_6
[27] Roman Timofeev. 2004. *Classification and Regression Trees (CART) Theory and Applications.* Ph.D. Dissertation. Humboldt University, Berlin.
[28] Vericlave. 2018. *The Kemuri Water Company Hack - Vericlave.* Technical Report. www.vericlave.com
[29] Herman Wijaya, Mauricio Aniche, and Aditya Mathur. 2020. *Domain-Based Fuzzing for Supervised Learning of Anomaly Detection in Cyber-Physical Systems (Appendix).* Technical Report. https://zenodo.org/record/3699088
[30] Kim Zetter. 2011. H(ackers)2O: Attack on City Water Station Destroys Pump | WIRED. https://www.wired.com/2011/11/hackers-destroy-water-pump/