Ann. Data. Sci. (2018) 5(3):359–386
https://doi.org/10.1007/s40745-018-0141-8

Classifying Categories of SCADA Attacks in a Big Data Framework

Krishna Madhuri Paramkusem[1] · **Ramazan S. Aygun**[1]

Received: 20 July 2017 / Revised: 1 December 2017 / Accepted: 5 January 2018 /
Published online: 15 January 2018

Abstract The supervisory control and data acquisition (SCADA) systems monitor and control industrial control systems in many industrial and economic sectors such as water treatment, power plants, railroads, and gas pipelines. The integration of SCADA systems with the internet and corporate enterprise networks for various economical reasons exposes SCADA systems to attacks by hackers who could remotely exploit and gain access to SCADA systems to damage the infrastructure and thereby harming people's lives. The simplicity of datasets and possible overfitting of models to training data are some of the issues in the previous research. In this paper, we present detecting and classifying malicious command and response packets in a SCADA network by analyzing attribute differences and history of packets using k-means clustering. This study presents a solution to classify SCADA cyber attacks to detect and classify SCADA attacks with high accuracy using a big data framework that comprises of Apache Hadoop and Apache Mahout. Apache Mahout's random forest classification algorithm is applied on SCADA's gas pipeline dataset to categorize attacks. When 70% of the data is used for training the classifier, our approach resulted in 5–17% improvement in accuracy for the classification of read response attacks and 2–8% improvement in accuracy for write command attacks with respect to using the original dataset.

Keywords SCADA attacks · Big Data Mining · Modbus · Hadoop · Mahout

1 Introduction

Supervisory control and data acquisition (SCADA) systems have been prone to cyber attacks in the recent past. These attacks are usually identified from the SCADAnetwork

✉ Ramazan S. Aygun
aygunr@uah.edu

1 Computer Science Department, University of Alabama in Huntsville, Huntsville, AL 35899, USA

packets. The goal of this study is to use the publicly available big data frameworks and libraries to extract useful insights from the SCADA dataset to secure SCADA systems. The nodes in a SCADA network communicate in terms of commands and responses using a communication protocol. These communication packets are captured and analyzed to understand the state of a system. This study presents a solution to classify SCADA cyber attacks using a Hadoop big data framework. There are many classifiers available for categorizing data, but most of them are not suitable for big data framework. As the data are huge, the classifier needs to work on each partition of the data separately. Apache Mahout provides classification algorithms that work with huge datasets. In this study, Apache Mahout's random forest classification algorithm is used to categorize the cyber attacks present in the SCADA system. The random forest ensemble classifier gave good classification results to many problems, and hence is used as a classification technique in this study.

The SCADA cyber attacks are categorized into malicious state command injection (MSCI), malicious parameter command injection (MPCI), denial of service (DoS), näive malicious response injection (NMRI), complex malicious response injection (CMRI), and reconnaissance attacks. These main category of attacks are again sub classified to 35 different specific category of attacks. In this study, we focus on the classification of the read response attacks (NMRI and CMRI) and the write command attacks (MSCI, MPCI, and DoS). The write command attacks corresponds to attacks that occur in the write command packets and the read response attacks correspond to attacks that occur in the read response packets.

The dataset used in this study contains attack packets and normal packets that arrive in a specific manner. A set of normal packets are followed by a set of attack packets, and they are not random in nature. In some cases, the normal packets may be confused for attack packets and vice-versa. Thus, analysis of the payload of just an individual packet for classification is not sufficient. The dataset is analyzed to understand the critical attributes that contribute to attacks. A comparison is made between the critical attributes of the current packet and the corresponding critical attributes of the previous packet(s) to derive additional features. The history of the packets are analyzed with 2-means clustering technique in order to group them into an attack category or a normal category. The difference between the 2-means centroids and the corresponding attribute is used to extract another set of new features. In this way, the attacks can be classified efficiently. The classification accuracy is significantly improved through this approach of using the information present in both the previous packet(s) and the current packet.

The contributions of this paper is summarized as follows:

1. We build attack classification and detection models by creating new attributes for packets by analyzing their history: (a) attribute differences from the previous packets and (b) determining general trend in previous n packets by applying 2-means clustering and using attribute differencing from cluster centers.
2. Our method increases the accuracy of classification using these new attributes without over-fitting since it just does not rely on individual packets.

3. We have performed a comprehensive evaluation of classifiers for different levels of attack categories (i.e., the presence of attack, the main category of attack, and the special category of the main-category of attack).
4. To the best of our knowledge, this is the first evaluation of SCADA attacks on a big data ecosystem.

This paper is organized as follows. The following section describes the background on SCADA attacks and the related work. Section 3 describes the feature extraction and data preprocessing for SCADA dataset. The big data mining approach using Apache Mahout are discussed in Sect. 4. Section 5 explains the experimental results using various performance metrics. The last section concludes our paper.

2 SCADA Attacks and Related Work

In this section, we briefly provide an overview of SCADA attacks and provide related work on the application of machine learning techniques for detecting SCADA attacks.

2.1 Overview of General SCADA Attacks

Cyber attacks have been a growing concern in SCADA systems. There have been various types of attacks: data integrity attacks [26] that manipulate sensor or control signals, database attacks [33] such as SQL injection that manipulate the data input into a web application, deception attacks [1] that include a wrong identity of a command sending device that can enable remote water pilfering from automated canal systems, command injection and response injection attacks [7] that inject false control commands and false responses into a control system. Miller et al. [17] provide a taxonomy of SCADA and critical infrastructure incidents as source sectors (e.g., commercial, government), method of operation (e.g., misuse of resources, user compromise), impact (e.g., disrupt, distort, destruct), and target sectors (e.g., commercial, non-US entity). Zhu et al. [33] provide a taxonomy of SCADA attacks and they categorize attacks into *network layer, transport layer, application layer,* and *attacks on implementation protocols* for the communication stack. The attacks against Modbus protocol are studied under *application layer* attacks.

The risks of big data analytics for power distributed systems are addressed in [32]. The issues are listed below:

1. Lack of innovative use cases and application proposals that convert big data into valuable operational intelligence,
2. Insufficient research on big data analytics system architecture design and advanced mathematic for petascale data, and
3. Failing to adhere to data privacy and data protection standards.

Deka et al. [5] study attacks on power-grids and propose a polynomial graph-based approach just using the grid topology for attacks distorting minimum measurements. Amin et al. [1] study stealthy deception attacks for water SCADA systems and show that these systems are not robust against such attacks by analyzing regulatory-level proportional–integral (PI) control and diagnostic-level unknown input observers

(UIOs) scheme. They emphasize the importance of detecting such attacks since they cannot be detected by basic SCADA attack detection methods. Similarly, Teixeira et al. [28] study stealthy deception attacks for SCADA energy systems. Carcano et al. [4] present intrusion detection system based on critical state analysis and state proximity. They provide a formal model of the system as states. They identify critical states and then try to determine critical state proximity since the goal of an attacker is to take the system from a safe to a critical state by a chain of licit commands. Sridhar et al. [26] model integrity and denial of service attacks for automatic generation control loop in a power grid. They emphasize the necessity of a secure communication link for the system. For replay attacks, Mo et al. [18] provide countermeasures for detection based on a zero-mean Gaussian authentication signal.

2.2 Application of Machine Learning Techniques for SCADA Attacks

This work focuses on detecting malicious SCADA communications with the help of machine learning techniques. The importance of intrusion detection through machine learning and the three objectives of intrusion detection are mentioned below [31]:

1. Detect as many types of attacks as possible (internal malicious and external opportunistic/deliberate attacks).
2. Detect the intrusions as accurately as possible, thereby reducing the number of false alarms.
3. Detect intrusions as early as possible to reduce the amount of damage they incur.

Machine learning has been used to discriminate malicious and anomalous events in intrusion detection for traditional cyber security networks [12,23,25]. Popular machine learning techniques such as rule based approach, hidden Markov model (HMM), support vector machines (SVM), and one class SVM (OCSVM) can be used to prevent SCADA from cyber-attacks. Torrisi et al. [29] use binary SVM for classifying unidirectional and bidirectional attacks for DNP3 communication protocol. Nader et al. [22] utilize support vector data description (SVDD) and the kernel principal component analysis (KPCA) for intrusion detection by studying $l_p - norms$ in radial basis function kernels. Fahad et al. [6] propose a privacy preserving framework for SCADA data publishing by evaluating multi-layer perceptron neural network, J48 decision tree, SVM, 7-nearest neighbor, and naive bayesian classifiers for a water treatment plant and simulated SCADA datasets along with other datasets.

For discriminating power system disturbances, Hink et al. [9] employ five categories of classifiers: probabilistic classification (naive bayesian), rule induction (OneR [10], nearest neighbor with generalization), decision tree learning (random forests), non-probabilistic binary classification (SVM), and boosting (Adaboost). For three classes (no event, attack, and natural distrurbance), JRipper+Adaboost yielded the best performance with low false positive rates. Shosha et al. [24] use feed-forward artificial neural networks for detecting anomalies in SCADA specific protocol traffic.

Almalawi et al. combine unsupervised and supervised techniques for intrusion detection. They use two assumptions: (i) the majority of the data are normal and (ii) the normal data are statistically different from the abnormal data. They assume that the data can be categorized into 3 groups: normal, abnormal, and doubtful data. Each data sample is assigned an anomaly score based on its k-nearest neighbors. Representative normal and abnormal datasets are determined using k-means clustering and these datasets are used to train best-first decision tree (BFTree), J48 decision tree, non-nested generalized exemplars (NNge), projective adaptive resonance theory (PART), and naive bayesian classifiers in WEKA data mining software. We should note that the second assumption is not applicable to our dataset. While a data at an instant could be normal, the same data could be abnormal at another instant. Hence, applying the k-nearest neighbor using the complete dataset for assigniing anomaly scores is not meaningful in our case. Moreoever, we do not have any assumption on the ratio of data packets which could be abnormal.

Machine learning techniques for cyber-attack discrimination in smart power grid framework are discussed in [13]. Maglaras et al. [14] propose K-OCSVM method in which output OCSVM is fed to recursive k-means clustering to determine the most possible/important outliers. Their method does not mention the use of historical context of packets and may fail to detect packets which could be normal or abnormal at different instants. Maglaras et al. [15] propose an ensemble method based on OCSVM. In terms of new attributes, they consider the time between previous packets and the number of packets to a specific destination within a number of previous packets. In our analysis, abnormal packets may have the same temporal behavior. Therefore, we consider that the relative content of the packets are more significant than the number of packets to a destination to determine whether they are abnormal or not.

Machine learning techniques were employed with SCADA data to address security issues in Critical Infrastructure systems (CIS). Naive bayesian, random forests, OneR, J48, and SVM algorithms were used through Weka to detect intrusions in SCADA system. The performance of these classifiers with respect to intrusion detection was evaluated [3]. Nader et al. [21] provided a one class classification technique for categorizing SCADA attacks using SCADA datasets [19]. Their classification accuracy was high (close to 100%). Their datasets were considered unsuitable for intrusion detection system (IDS) research since there were obvious patterns between few parameters present in the dataset which resulted in predictions that gave high accuracies [11].

New SCADA datasets were created to avoid problems present in the old datasets [20]. The random tree classification experiment was done using the entire dataset for both training and testing phases [30]. This resulted in an accuracy of 99.7%. The use of 100% of the data for training and 100% of the data for testing does not represent a realistic performance. This high result might be even due to the overfitting of the data. The experiment was done using the current packet information and no information from the previous packets was utilized.

The gas pipeline dataset developed at the Mississippi State University SCADA Laboratory [30] is used in this research. The dataset includes MSCI, MPCI, and MFCI that fall in the category of command injection attacks and NMRI and CMRI that fall in the category of response injection attacks. These attacks use the state information of the system in order to design attacks that mimic the normal behavior of the system. These attacks can hide system changes and are difficult to detect.

In this study, we use the new datasets [20] with the random forest classification algorithm. There are several factors that distinguish our work from previous work. (1) We do not assume that abnormal packets are different from the normal packets. In our work, the same packet could be normal at one instant and abnormal at another instant. (2) By analyzing a number of previous packets, we expect a different content in malicious packets. (3) The relative content of a packet with respect to the previous packets is more important than the frequency or the number of packets to a destination. Our approach uses big data framework for categorizing SCADA attacks. The information present in both the current and the previous packets were used for analysis.

3 Feature Extraction and Data Preprocessing for SCADA Log Analysis

In this section, we will discuss about the steps required to process the original dataset to make it suitable for analyzing write command attacks and read response attacks. The write command attacks include *malicious state command injection (MSCI), malicious parameter command injection (MPCI),* and *denial of service (DoS)* attacks. These include specific category attacks, 1 through 18 as mentioned in Table 1. The read response attacks include *näive malicious response injection (NMRI) and complex malicious response injection (CMRI)* attacks. These include attacks 25 through 35 as mentioned in Table 1.

We analyzed the attributes to distinguish normal and attack packets and then developed our method. Our method does not look at the semantics of attributes, and attributes could be just labeled as attribute 1, attribute 2, etc. In such a case, our system still works with unknown attributes. A packet at one time could be a normal packet and another time could be an attack packet. The main idea comes from how attack packets are generated. The attacker does not know the actual status of the SCADA system. The attack packets contain just somewhat random values to disrupt the system. It is expected to be different from actual values in the system. Whenever an attack packet is sent, it is expected to deviate from the actual values.

3.1 Feature Extraction

Figure 1 shows a plot between the *pressure measurement* attribute vs the *crcrate* attribute of the read response attacks with labeled categories (normal and attack) of packets. The plot shows the range of pressure values for normal data and attack data. When a classifier is trained against such a dataset, it learns that for a certain range of pressure values there are no attacks and for certain range of pressure values there are attacks. For such cases, we do not need to even train a classifier, instead a simple *if–else* condition might be enough. Nevertheless, values in a range could be normal for certain times and attacks some other times. Forcing the classifier to learn ranges like this causes overfitting. Accuracy might be high for such a dataset, but such a classifier is not desirable due to overfitting. In the pressure range 375 and 395, there are certain pressure values (circled around 375 and 395) that are taken to be valid for an instant of time and invalid for another instant of time, which might result in misclassifications (circled in red in Fig. 1). Hence, new features need to be extracted from the original attributes.

We extract new features by comparing the previous and current packets. In the original dataset, attack packets and normal packets are not random. A set of normal packets are usually followed by a set of attack packets. For example, there is a case

Table 1 Specific categories of cyber attacks in a SCADA system [20]

Attack	Number	Category	Description
Setpoint	1–2	MPCI	Changes the pressure set point outside and inside of the range of normal operation
PID gain	3–4	MPCI	Changes the gain outside and inside of the range of normal operation
PID reset rate	5–6	MPCI	Changes the reset rate outside and inside of the range of normal operation
PID rate	7–8	MPCI	Changes the rate outside and inside of the range of normal operation
PID deadband	9–10	MPCI	Changes the deadband outside and inside of the range of normal operation
PID cycle time	11–12	MPCI	Changes the cycle time outside and inside of the range of normal operation
Pump	13	MSCI	Randomly changes the pump state
Solenoid	14	MSCI	Randomly changes the solenoid state
System mode	15	MSCI	Randomly changes the system mode
Critical condition	16–17	MSCI	Places the system in a critical condition which is not part of the normal activity
Bad CRC	18	DoS	Sends Modbus packets with incorrect CRC values
Clean registers	19	MFCI	Cleans the registers of the slave devices
Device scan	20	Recon	Scans for all possible devices controlled by the master
Force listen	21	MFCI	Forces the slaves to only listen
Restart	22	MFCI	Restart communication on the device
Read ID	23	Recon	Reads ID of the slave device. In this attack the data of the device is actually not recorded, but is shown as if the data is being recorded
Function code scan	24	Recon	Scans for possible functions that are being used on the system. In this attack the data of the device is actually not recorded, but is shown as if the data is being recorded
Rise/Fall	25–26	CMRI	Sends back pressure readings which create trends on the pressure reading graph
Slope	27–28	CMRI	Randomly increases/decreases pressure reading by a random slope
Random value	29–31	NMRI	Random pressure measurements are sent to the master
Negative pressure	32	NMRI	Sends back a negative pressure reading from the slave
Fast	33–34	CMRI	Sends back a high set point then a low set point which changes ‘fast’
Slow	35	CMRI	Sends back a high set point then a low set point which changes ‘slow’

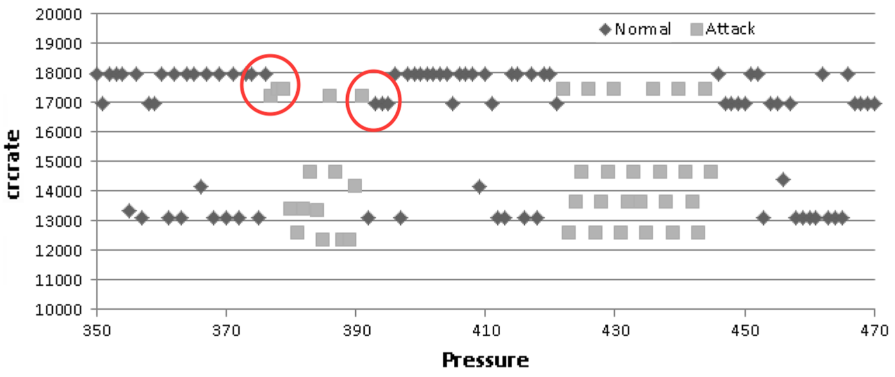


Fig. 1 Over-fitting of data for read response attacks. Certain pressure values (circled in red) in the pressure range 375 and 395 are taken to be valid for an instant of time and invalid for another instant of time, which might result in misclassification. (Color figure online)

where 11 normal packets are followed by 11 attack packets, 11 normal packets, and 26 attack packets and so on. A packet may be classified as normal at one instant of time and the same packet may be classified as an attack at another instant of time. This results in misclassification of packets if packets are analyzed individually. We extract features from packets in two ways:

1. The current packet is compared with its previous packet:

$$n_i = m_i - m_{i-1}, \quad \text{for } i > 1 \tag{1}$$

   In Eq. 1, n_i indicates the new attribute value for the ith packet, m_i indicates the corresponding original attribute value of the ith packet, and m_{i-1} indicates the corresponding original attribute value of the $(i-1)$th packet.
2. The current packet is compared with its previous p packets using the K-means algorithm. The value of K is 2 for the experiments, since we have two categories of data packets: an attack packet and a normal packet. In case the packets are all normal then the two centroids obtains from the 2-means are close to each other.

$$\boldsymbol{C} = kmeans(m(i-p:i-1), 2) \tag{2}$$

   In Eq. 2, C holds the two centroids for the ith attribute value for the attribute m. The difference between the further centroid to the ith attribute value is computed.

$$n_i = \begin{cases} m_i - c_1 & \text{if} \quad |m_i - c_1| > |m_i - c_2| \\ m_i - c_2 & \text{if} \quad |m_i - c_2| \geq |m_i - c_1| \end{cases}$$

The experiments are done by considering the information present in both the current packet and the previous packet(s). As a result, we evaluate three types of datasets:

1. *Original dataset* The dataset with the original attributes.
2. *Attribute-difference dataset* The dataset with additional attributes that are derived by comparing the current packet with its previous packet.

3. *2-means dataset* The dataset with additional attributes that are obtained by comparing the current packet with its previous p packets. In our experiments, we used previous 30 packets for 2-means ($p = 30$).

3.1.1 Discussion on K-means

The K-means algorithm is used for extracting additional features by analyzing history of previous p packets. The SCADA gas pipeline dataset used in this study has a sequence of attack instances followed by a sequence of normal instances. It is reasonable to check the pattern of previous set of packets to understand how the current packet is different from the previous packets. K-means is used only in the feature extraction stage to understand the behavior of packets. The value of K is chosen to be 2, since we either have a normal packet or an attack packet. K-means is not suitable in detecting clusters that are non-spherical in shape or widely different in sizes or densities [27]. If there are normal packets as well as attack packets, two clusters are expected in the dataset: one for normal packets and another for attack packets. The current packet is compared with the centroids of those packets. If all packets are normal, the cluster centroids should be close to each other. The main idea is to compare the current packet with two centroids from previous packets. It is not critical to split packets into normal and attack clusters accurately.

3.1.2 Theoretical Analysis on Feature Extraction

The main goal of feature extraction process is to generate new features that can separate an attack packet from normal packets.

Let C_{np} and C_{ap} be the clusters representing normal packets and attack packets, respectively. Ideally, C_{np} should contain normal packets whereas C_{ap} should include only attack packets. In this domain, 2-means clustering considers a history of previous n packets. If there is *only* one attack packet (p_a) in previous n packets, 2-means clustering will put p_a into C_{ap}. Nevertheless, C_{ap} may also have other normal packets. Regardless, C_{np} will always have normal packets and its mean will be used to generate 2-means attributes. This shows that as long as attack packets go into one cluster regardless of normal packets in that cluster, attributes are generated properly due to a pure cluster of normal packets. This shows that feature extraction is not heavily dependent on incorrect 2-means clustering.

The worst case for 2-means clustering occurs when the normal and attack packets are evenly distributed between these clusters. Although attribute values are more critical than the ratio of packets, the ratio of normal to attack packets is used for simplicity in our analysis by checking this ratio is greater than or equal to a threshold α for safe clustering.

The second aspect of feature creation is the distribution of attack packets in a window frame. If there is a normal packet between every attack packet, the proper attribute values can easily be determined by attribute difference. The worst scenario occurs when there are contiguous attack packets where attacks packets could be considered as new normal. We use Bernoulli distribution with a probability of p for an attack packet and $(1 - p)$ for a normal packet. For safe 2-means clustering of n packets with k of them are attack packets, $\frac{(n-k)}{k} \geq \alpha$ should be satisfied. This implies that if $k \geq \lceil \frac{n}{\alpha+1} \rceil$, the feature extraction process might not be effective. In case of k attack packets, the previous packet should be an attack packet, otherwise attribute difference would be enough to generate new attributes for distinguishing the current packet. Out of k packets, the previous packet is considered to be an attack packet in the worst case scenario. If the binomial distribution is used for the rest of the $(k-1)$ packets in the window, the probability of poor feature extraction becomes

$$f(n, p, \alpha) = \sum_{k=\lceil n/(\alpha+1) \rceil}^{n} \binom{n-1}{k-1} (1-p)^{(n-k)} p^{(k-1)} p$$

For example, for $\alpha = 1$, $p = 0.2$, and $n = 20$, the probability of getting poor attributes for the worst case is 0.0013 assuming that 2-means clustering will definitely fail for α ratio.

3.2 Data Preprocessing for Write Command Attacks

Data preprocessing involves the following steps:

1. The instances whose *function code* is 16 and the Modbus frame *length* is 90 are taken from the original data set. The dataset now contains 64,100 instances.
2. *Data cleaning* The pressure measurement (*14th feature*) is removed from the dataset as it is has missing values.
3. *Feature extraction* The *address, function, length,* and *command response* attributes have the same constant values for all the write command packets, and hence are removed from the dataset. All the packets are ordered in time using the *time* attribute. The *time* attribute is then removed from the dataset. The dataset now contains 11 critical attributes for classifying the write command attacks as shown in the Table 2. Analysis is based on comparisons between the current packet and the previous packet(s). The comparison is done in the following manner:
   (a) Each packet is compared to its previous packet, except for the first one. The comparison is done with respect to each attribute individually (excluding the three labels) in the packet that includes *setpoint, gain, reset rate, deadband, cycletime, rate, system mode, control scheme, pump, solenoid,* and *crcrate*. The difference in attributes of the current packet and the previous packet give new features as shown in the Table 3. The attribute values for these new features is 0 for the first packet in the dataset.
   (b) Each packet is compared with its previous set of packets. This comparison is done using the *kmeans* algorithm. The value of *k* is 2 for this experiment. The attributes *setpoint, gain, reset rate, deadband, cycletime, rate, system mode, control scheme, pump, solenoid,* and *crc rate* are considered here. The new features that result from this process are shown in Table 4.
   The packets are compared as follows:
      i. For the first *p* packets, the current packet is compared with all its previous packets. For example, the 3rd packet is compared with all the previous 2

Table 2 Critical attributes for write command attacks

Attribute name
setpoint
gain
reset rate
deadband
cycletime
rate
system mode
control scheme
pump
solenoid
crcrate

Table 3 Feature extraction for write command attacks—using the previous packet

Original feature	New feature
setpoint	setpoint_diff
gain	gain_diff
reset rate	resetrate_diff
deadband	deadband_diff
cycletime	cycletime_diff
rate	rate_diff
system mode	systemmode_diff
control scheme	controlscheme_diff
pump	pump_diff
solenoid	solenoid_diff
crcrate	crcrate_diff

Table 4 Feature extraction for write command attacks with 2-means—using previous packets

Original feature	New feature
setpoint	setpoint_centroid_diff
gain	gain_centroid_diff
reset rate	resetrate_centroid_diff
deadband	deadband_centroid_diff
cycletime	cycletime_centroid_diff
rate	rate_centroid_diff
system mode	systemmode_centroid_diff
control scheme	controlscheme_centroid_diff
pump	pump_centroid_diff
solenoid	solenoid_centroid_diff
crcrate	crcrate_centroid_diff

packets. For the first packet, the new attribute values are taken to be 0. For the second packet, the new attribute values are equal to the previous attribute values (first packet attribute values).

ii. For the remaining packets (starting from the $(p+1)$th packet), the 2-means algorithm is applied to each of the 11 original attributes of the previous 30 packets individually.

We use the following three datasets for our experiments:

1. The original dataset with 19 attributes.
2. The attribute-difference dataset with 11 derived attributes and 14 original attributes, in total 25 attributes. These 11 attributes are derived by comparing the current packet attribute values with its previous packet attribute values.
3. The 2-means dataset with 11 additional attributes derived from the comparison of the current packet with its previous packets, the 11 new attributes derived in the attribute-difference dataset, and 14 original attributes. Hence, this dataset contains 36 attributes.

The dataset is split to training and testing in the following ways:

1. 100–100 The entire dataset is used for both training and testing.
2. 80–20 80% of the instances (first 51,280 instances) were used for training the random forest model and the remaining 30% of the instances (remaining 12,820 instances) were used for testing the classifier.
3. 70–30 70% of the instances (first 44,870 instances) were used for training the random forest model and the remaining 30% of the instances (remaining 19,230 instances) were used for testing the classifier.

3.3 Data Preprocessing for Read Response Attacks

Data preprocessing involves the following steps:

1. The instances whose *function code* is 3 and the Modbus frame *length* is 46 are taken from the original dataset. The dataset now contains 68,848 instances.
2. *Data cleaning* The attributes with missing values are removed from the instances. These include *setpoint, gain, reset rate, deadband, cycletime, rate, system mode, control scheme, pump* and *solenoid* attributes. These ten features are trimmed off in the pre-processing stage.
3. *Feature extraction* The *address, function, length,* and command response are removed, since they have the same constant value for all the read response packets. The read response packets are also ordered in time, and then the *time* attribute is trimmed off. The dataset contains two attributes namely, *pressure measurement* and *crcrate*, apart from the three labels. The analysis is based on the current packet's *pressure measurement* with the previous packets's *pressure measurement* values, as mentioned in Section 3. This gives two derived attributes namely, *pressuremeasurement_diff* and *pressuremeasurement_centroid_diff*.

We use the following three datasets for experiments. Original dataset with 10 attributes. The attribute-difference dataset contains 1 derived attribute, *pressuremeasurement_diff* and 5 original attributes, in total 6 attributes. The 2-means dataset contains 1 additional attribute derived from the comparison of the current packet with its previous packets, the 1 new attribute derived in the first dataset, and 5 original attributes. Thus, the second dataset contains 7 attributes.

The dataset is split to training and testing in three ways as for write command attacks: 100–100, 80–20 and 70–30. The 80–20 had first 55,078 instances for training and the remaining 13,770 instances for testing. The 70–30 had first 48,194 instances for training and remaining 20,654 instances for testing.

4 Big Data Mining Using Apache Mahout

Mahout is a project supported by Apache Software Foundation [2]. It is a library of scalable machine learning algorithms that works on top of Hadoop. The machine learning algorithms (clustering and classification) are executed as a series of map reduce jobs. In this study, we use random forest ensemble algorithm.

Ensemble methods are techniques that improve classification accuracy by aggregating the predictions of multiple (weak) classifiers. They generally predict the class label of a test record based on the majority vote on the predictions made by the classifier [27].

Random forest is an ensemble classifier of decision trees. A random forest classifier combines the predictions made by multiple decision trees, where each tree is based on randomly selected samples. These random samples are in turn generated from a fixed probability of distributions [27]. Since each decision tree could be generated using random samples independently, random forest is suitable for big data systems. The big data is partitioned into chunks and stored in the HDFS. The classification algorithm works on chunks of data independently to construct multiple base classifiers (decision trees), and then aggregate their predictions when classifying test records.

4.1 Procedure

In traditional classification problems, there is only one class label for the available dataset. Based on the range of this class attribute, the problem can be classified as binary (2-class) classification if there are two classes (or multi-class classification problem if there are more than 2 classes).

In this SCADA domain, we have a hierarchy of class labels and the original dataset contains three levels (labels) of classes. At the first or top level, the class label (18th attribute) indicates if an instance is a normal or an attack sample as in binary classification problems. If the instance is an attack sample, the class label has a sub-category of the attack as the main-category attack at the second level. At the third-level, the specific category of the main category attack is provided. The third label 'specific result' specifies the sub-category of an attack sample. Only one of the three labels is considered as a class while doing the experiments, and the remaining two labels were either ignored or treated as a categorical attribute. The valid combinations of the labels are shown in the Table 5. For example, the third row indicates that only the *categorized*

Table 5 Valid combinations of labels

Label	Binary result	Categorized result	Specific result	Comment
LII	Label	Ignore	Ignore	Detection
CLI	Categorical attribute	Label	Ignore	Main category classification, knowing the presence of an attack
ILI	Ignore	Label	Ignore	Main category classification without knowing the presence of an attack
CCL	Categorical attribute	Categorical attribute	Label	Specific category classification, knowing the main category of an attack
IIL	Ignore	Ignore	Label	Specific category classification, without knowing the presence of an attack

result is used for training, while ignoring the *binary result* and *specific result*. There are three hierarchies of attack categorization:

1. Detecting the presence of an attack (binary result).
2. Classifying the attack into one of the main categories of attacks.
3. Classifying the attack into one of the sub-categories of attacks (specific result) as listed in Table 1.

For example, (1: Attack, 2: NMRI attack, 3: Pipe) attack would be three levels of class labels for a sample packet. This classification indicates that this is an attack packet and its main category of attack is NMRI attack, and the special category of NMRI attack is pipe attack. This allows to train models based on each level of this classification hierarchy. For example, we could build a model for just detecting presence of attacks for the first category by ignoring the other levels of classification. We code the label LII in order of levels of the hierarchy as (1) L: the label of the first level is used, (2) I: the label from the second level is ignored, and (3) I: the label from the third level is ignored.

If we build a model for detecting the main categories of attacks assuming that the packet is an attack packet, it is coded as CLI: (1) C: the presence of attack is provided, (2) L: the model is build based on the second level for the main category, and (3) I: the third level is ignored. If we just focus on finding the specific categories ignoring any knowledge about presence of an attack or the main category of the attack if there is an attack, then it is coded as IIL: (1) I: the presence of attack is ignored, (2) I: the main category of the attacks is ignored, and (3) L: the model is just built for the special category (the third level).

Each of these levels is represented as an attribute in the dataset. Depending on the way classifiers are built, we ignore some of these class levels and use the aforementioned coding scheme for labeling (e.g., IIL).

The commands used for the experiments are based on the information provided in [2].

4.2 Commands for Write Command Attacks

– **mahout describe**

This command generates the descriptor file for the given dataset. It basically creates a metadata for the input dataset. The input parameters to this command would be the *.arff* dataset file, path for the output file, and the attributes type description as shown below:

```
mahout describe -p hdfs://localhost:9000
\/wckmeans_ss3/wcddcent.arff -f
\hdfs://localhost:9000/wckmeans_ss3/iil.info
\ -d 33 N I I L
```

The descriptor (-d) indicates that the dataset has thirty three numerical attributes (33N) followed by ignoring the two labels (I I), and the specific category label (L). This procedure corresponds to the 4th row in Table 5. The output of this command is a *.info*.

– **mahout buildforest**

This command builds the trees using the partial builder implementation with the training data. The input parameters to this command would be the split size in bytes, training dataset, the *.info* file generated by the *describe* command, the number of random attributes to be selected at each node of the tree for splitting, the number of trees to be built, and the output path for the *forest.seq* file as shown below:

```
mahout buildforest -Dmapred.max.splitsize
\ =2655590 -d hdfs://localhost:9000
\ /wckmeans_ss3/wcddcent.arff -ds
\ hdfs://localhost:9000/wckmeans_ss3
\/iil.info -sl 6 -p-t 100 -o hdfs://
\ localhost:9000/wckmeans_ss3/6100iil
```

The number of random attributes to be selected at each node of the tree for splitting is set to the square root of the total number of features. Hence, 4 random attributes were selected at each node with original dataset, 5 random attributes were selected at each node with attribute difference dataset, and 6 random attributes were selected at each node with 2-means dataset. 100 trees were built with all the three datasets.

The number of random attributes selected at each node is usually equal to the square root of the total number of attributes in the dataset excluding the label. The importance of split size will be discussed in the next section.

– **mahout testforest**

Once the model is available, we use it to classify the test data. The input parameters for this command include the test dataset file (.arff file in this experiment), the descriptor *.info* generated by the *describe* command, the path to the model formed by the *buildforest* command, and the output path, as shown below:

```
mahout testforest -i hdfs://localhost:9000
\/wckmeans_ss3/wcddcent.arff -ds
\/iil.info -m hdfs://localhost:9000
\/wckmeans_ss3/6100iil -a -mr -o
\ hdfs://localhost:9000/wckmeans_ss3
\/6100iilout
```

4.3 Commands for Read Response Attacks

The commands for classifying read response attacks are similar to those used for write command attacks, except for the *mahout describe* which takes different parameters for -d option. The following procedure corresponds to the 4th row in Table 5.

```
mahout describe -p hdfs://localhost:9000
\/SCADARRAttacks/SCADARRA_Train.arff
\ -f /SCADARRAttacks/SCADARRA_TrainLII.info
\ -d 3 N C C L
```

3 random attributes were selected at each node with the original dataset, 2 random attributes were selected at each node with the attribute difference dataset, and 3 random attributes were selected at each node with the 2-means dataset. 50 trees were built with all three datasets.

4.4 Split Size

The split size parameter *-Dmapred.max.split.size* is the number of map-reduce tasks that would be set in the partialbuilder class. This parameter affects the performance of Hadoop. As the number of tasks increases, the framework overhead increases, which increases the load balance and lowers the cost of failures. On the other hand, this might even result in resource exhaustion [8]. After analyzing different split sizes for initial experiments, split size for the experiments is chosen as 3. In other words, the data is split into 3 partitions.

5 Experimental Results

In this section, we discuss the results of our experiments in terms of accuracy and false negative rates for all the three datasets for various training and testing splits. While comparing the performance for various splits (70–30, 80–20 and 100–100), we provide analysis for IIL (specific category) experiments.

5.1 Performance Metrics

We provide results of classification using a number of performance metrics [30].

1. *Confusion matrix* The performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model. These counts are

tabulated in a confusion matrix. The format of a confusion matrix for a binary classification problem is depicted in Fig. 2. Each f_{ij} in this figure denotes the number of records from class i predicted to be of class j. f_{11} is the number of *true positives*, f_{10} is the number of *false negatives*, f_{01} is the number of *false positives*, and f_{00} is the number of *true negatives*.

2. *Accuracy* It is the percentage of correct decisions made by the classifier. The problem with accuracy is that it does not consider the distribution of misclassification. Accuracy measure is not discriminative.

$$\text{Accuracy} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}} \tag{3}$$

3. *False negative rate (FNR)* It is the percentage of attack instances that are classified as normal instances.

$$\text{FNR} = \frac{f_{10}}{f_{11} + f_{10}} \tag{4}$$

If a specific category attack is classified as another attack, this does not reduce the FNR. In other words, only attacks that are classified as normal increase the FNR.

4. *Precision* This metric determines the ratio of actual attacks to total attacks classified by the system.

$$\text{Precision} = \frac{f_{11}}{f_{11} + f_{01}} \tag{5}$$

5. *Recall* This is the ratio of the number of instances that are classified correctly as a category of attack to the total number of instances in that category of the attack. It provides a metric to determine the true positive ratio in a category of the attack. This is also known as *sensitivity* or *hit rate*.

$$\text{Recall} = \frac{f_{11}}{f_{11} + f_{10}} \tag{6}$$

6. *F1 score* This is harmonic mean of precision and recall, thus providing a single measurement for a system. It has values between 1 (best case) and 0 (worst case).

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

7. *Kappa* The discrepancy between normal statistics and *kappa* is shown through *kappa* statistics. It is basically a metric to quantitatively show the agreement between two observers. In the context of classification results, it is an agreement between the actual categories and predicted categories forms the basis for calculation of kappa.

8. *Categorical accuracy (Cacc)* We define this accuracy measure for this work. Categorical accuracy is the ratio of correctly classified instances (normal or different main categories of attacks) to total number of instances. The correctly classified instances are those that belong to the same main attack category. It ignores misclassifications within a category. For instance, suppose we have three main classes

Fig. 2 Confusion matrix for a binary (2-class) classification problem

		Predicted Class	
		$Class = 1$	$Class = 0$
Actual Class	$Class = 1$	f_{11}	f_{10}
	$Class = 0$	f_{01}	f_{00}

of attacks, namely a, b, and c. Each of these main classes have three sub categories, namely a_0, a_1, and a_2; b_0, b_1, and b_2; and c_0, c_1, and c_2. If the classifier predicts the classes based on the sub category (a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, and c_2) , then categorical accuracy considers the main category (a, b, and c) to measure accuracy instead of sub category. If an instance is classified as a_1 instead of a_0, it is still considered to be correctly classified by *categorical accuracy*, since it belongs to the same main category a and not b or c.

5.2 Experimental Results for Write Command Attacks

This section provides the results of the experiments for write command attacks for original, attribute difference, and 2-means datasets.

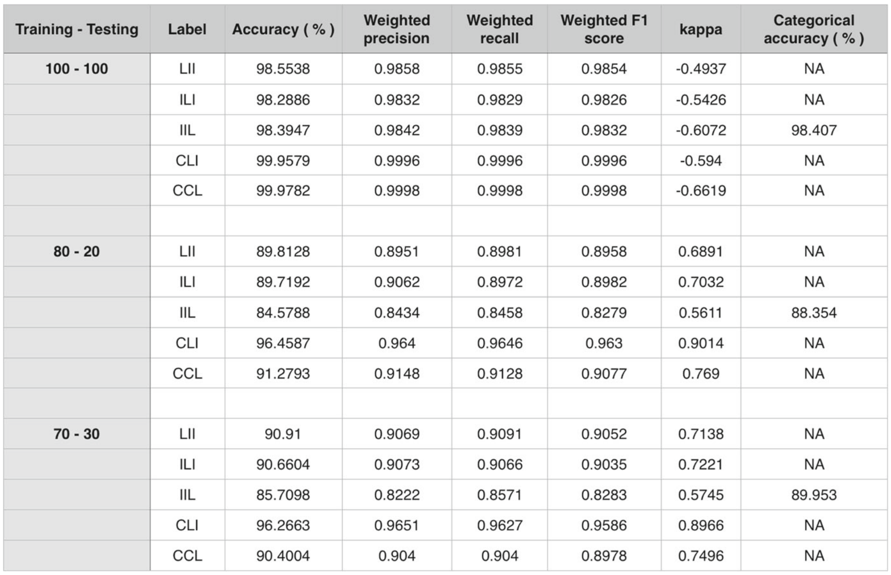
5.2.1 Original Dataset

The results for the 70–30 training–testing sets for the IIL label are discussed here. 98% of the normal packets (0) were classified correctly. *Critical condition* attacks (17), *PID cycle time* attacks (11–12), and PID gain attacks (3–4) were also classified correctly. However, 32% of the *bad crc* (18), 98% of the *system mode* (15), 73% of the *pump* (13), 90% of *PID deadband* (10), and 97% of *PID deadband* (9) attacks were missed (classified as normal). The false negative rate is 37.8%, which is quite high. *Solenoid* attacks (14) were not detected at all. The accuracy of the classifier is 85.7098%.

The summary of classification results for the original dataset for 70–30, 80–20, and 100–100 splits is shown in Fig. 3. The accuracies were high when the entire dataset (100–100) is used for both training and testing the classifier, since the classifier learns well in this case. The accuracies for 70–30 and 80–20 training–testing splits were almost close to each other, and they represent a more realistic performance, as different datasets were used for training and testing the classifier.

5.2.2 Attribute Difference Dataset

The classification results for the 70–30 training–testing sets for the IIL label are discussed here. We have the following observation with respect to the original dataset. 0.7% of the normal packets (0) were classified as attacks. The classification accuracy of *pump* (13) attack is the same. *PID rate* (8), *PID reset rate* (6), *PID deadband* (9–10), system mode (15), and *bad crc* (18) attacks were detected far more better. The false negative rate is reduced from 37.8 to 27.7%. Hence, more number of attacks were detected in this case. While no solenoid attack was detected using the original dataset,



Training - Testing	Label	Accuracy (%)	Weighted precision	Weighted recall	Weighted F1 score	kappa	Categorical accuracy (%)
100 - 100	LII	98.5538	0.9858	0.9855	0.9854	-0.4937	NA
	ILI	98.2886	0.9832	0.9829	0.9826	-0.5426	NA
	IIL	98.3947	0.9842	0.9839	0.9832	-0.6072	98.407
	CLI	99.9579	0.9996	0.9996	0.9996	-0.594	NA
	CCL	99.9782	0.9998	0.9998	0.9998	-0.6619	NA
80 - 20	LII	89.8128	0.8951	0.8981	0.8958	0.6891	NA
	ILI	89.7192	0.9062	0.8972	0.8982	0.7032	NA
	IIL	84.5788	0.8434	0.8458	0.8279	0.5611	88.354
	CLI	96.4587	0.964	0.9646	0.963	0.9014	NA
	CCL	91.2793	0.9148	0.9128	0.9077	0.769	NA
70 - 30	LII	90.91	0.9069	0.9091	0.9052	0.7138	NA
	ILI	90.6604	0.9073	0.9066	0.9035	0.7221	NA
	IIL	85.7098	0.8222	0.8571	0.8283	0.5745	89.953
	CLI	96.2663	0.9651	0.9627	0.9586	0.8966	NA
	CCL	90.4004	0.904	0.904	0.8978	0.7496	NA

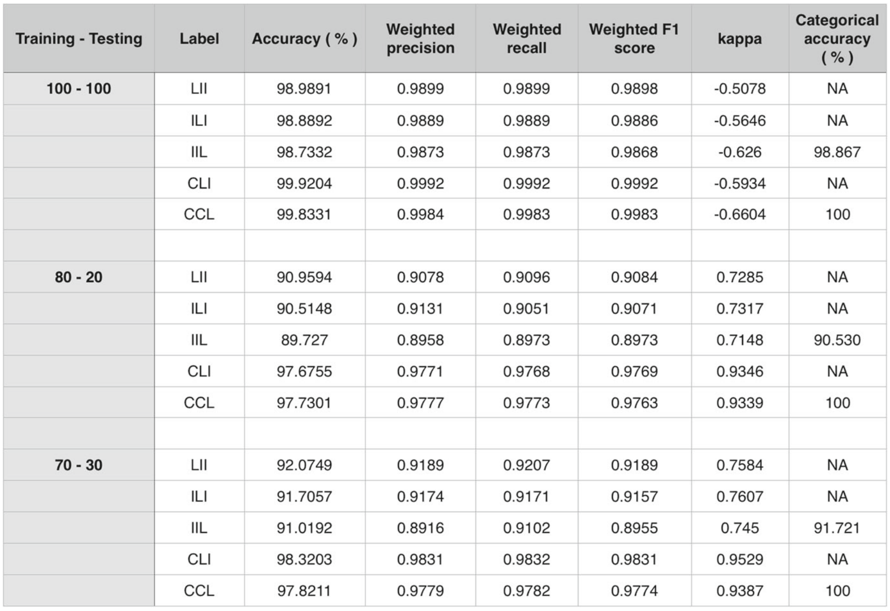
Fig. 3 Performance of random forest classification for write command attacks using the original dataset for different training–testing splits and different labels

only one instance of the solenoid attack could be detected with this dataset. The accuracy of the classifier is 91.019%. The classification accuracy is 5% more than using the original dataset. The summary of classification results for the original dataset for 70–30, 80–20 and 100–100 splits is shown in Fig. 4. The accuracies for 70–30 and 80–20 training–testing splits are close to each other. The results have improved by 7% with the use of attribute difference dataset for classification

5.2.3 2-Means Dataset

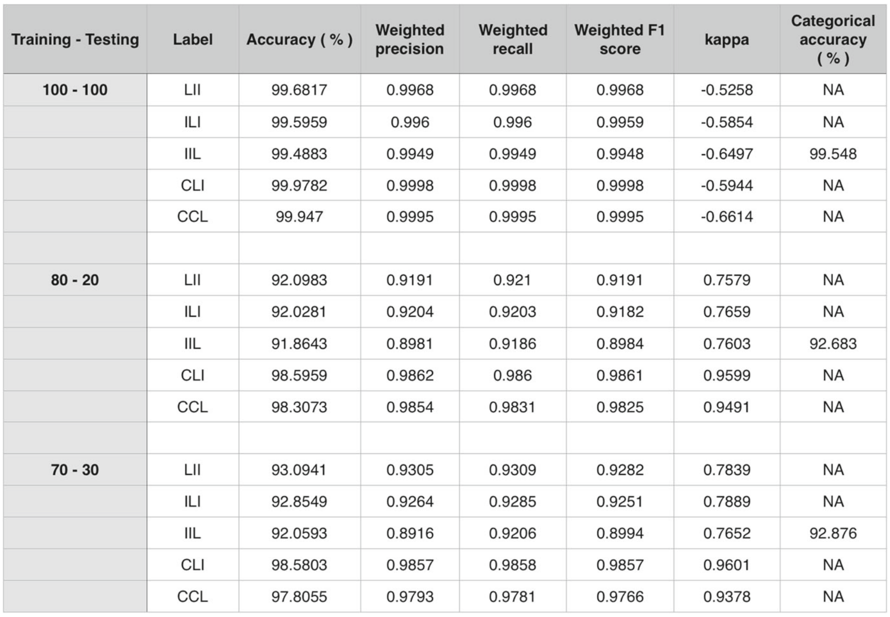
The experimental results for the 70–30 training–testing sets for the IIL label are discussed here. We have the following observation with respect to the attribute difference dataset. 99% of the normal packets (0) were classified correctly. The attack categorization was less when compared to the attribute difference dataset. *Solenoid* attacks (14) could not be detected at all as in using the original dataset. The false negative rate increased from 27.7 to 29.3%. This is still low when compared to the false negative rate of the original dataset and it is slightly more when compared to the false negative rate of the attribute difference dataset. The accuracy of the classifier is 92.0593%. The classification accuracy is 6% more than the original dataset but is slightly higher compared to the attribute-difference dataset. However, more attack packets are missed.

The accuracies for 70–30 and 80–20 training–testing split are close to each other as shown in Fig. 5, except for the ILI (main category) case were 80–20 performed well. The results have improved upto 9% when compared to the original dataset.



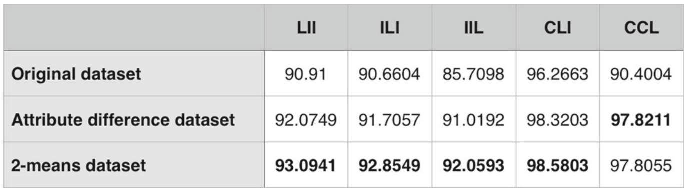
Training - Testing	Label	Accuracy (%)	Weighted precision	Weighted recall	Weighted F1 score	kappa	Categorical accuracy (%)
100 - 100	LII	98.9891	0.9899	0.9899	0.9898	-0.5078	NA
	ILI	98.8892	0.9889	0.9889	0.9886	-0.5646	NA
	IIL	98.7332	0.9873	0.9873	0.9868	-0.626	98.867
	CLI	99.9204	0.9992	0.9992	0.9992	-0.5934	NA
	CCL	99.8331	0.9984	0.9983	0.9983	-0.6604	100
80 - 20	LII	90.9594	0.9078	0.9096	0.9084	0.7285	NA
	ILI	90.5148	0.9131	0.9051	0.9071	0.7317	NA
	IIL	89.727	0.8958	0.8973	0.8973	0.7148	90.530
	CLI	97.6755	0.9771	0.9768	0.9769	0.9346	NA
	CCL	97.7301	0.9777	0.9773	0.9763	0.9339	100
70 - 30	LII	92.0749	0.9189	0.9207	0.9189	0.7584	NA
	ILI	91.7057	0.9174	0.9171	0.9157	0.7607	NA
	IIL	91.0192	0.8916	0.9102	0.8955	0.745	91.721
	CLI	98.3203	0.9831	0.9832	0.9831	0.9529	NA
	CCL	97.8211	0.9779	0.9782	0.9774	0.9387	100

Fig. 4 Performance of random forest classification for write command attacks using the attribute difference dataset for different training–testing splits and different labels



Training - Testing	Label	Accuracy (%)	Weighted precision	Weighted recall	Weighted F1 score	kappa	Categorical accuracy (%)
100 - 100	LII	99.6817	0.9968	0.9968	0.9968	-0.5258	NA
	ILI	99.5959	0.996	0.996	0.9959	-0.5854	NA
	IIL	99.4883	0.9949	0.9949	0.9948	-0.6497	99.548
	CLI	99.9782	0.9998	0.9998	0.9998	-0.5944	NA
	CCL	99.947	0.9995	0.9995	0.9995	-0.6614	NA
80 - 20	LII	92.0983	0.9191	0.921	0.9191	0.7579	NA
	ILI	92.0281	0.9204	0.9203	0.9182	0.7659	NA
	IIL	91.8643	0.8981	0.9186	0.8984	0.7603	92.683
	CLI	98.5959	0.9862	0.986	0.9861	0.9599	NA
	CCL	98.3073	0.9854	0.9831	0.9825	0.9491	NA
70 - 30	LII	93.0941	0.9305	0.9309	0.9282	0.7839	NA
	ILI	92.8549	0.9264	0.9285	0.9251	0.7889	NA
	IIL	92.0593	0.8916	0.9206	0.8994	0.7652	92.876
	CLI	98.5803	0.9857	0.9858	0.9857	0.9601	NA
	CCL	97.8055	0.9793	0.9781	0.9766	0.9378	NA

Fig. 5 Performance of random forest classification for write command attacks using the 2-means dataset for different training–testing splits and different labels



	LII	ILI	IIL	CLI	CCL
Original dataset	90.91	90.6604	85.7098	96.2663	90.4004
Attribute difference dataset	92.0749	91.7057	91.0192	98.3203	**97.8211**
2-means dataset	**93.0941**	**92.8549**	**92.0593**	**98.5803**	97.8055

Fig. 6 Classification accuracies for write command attacks using different datasets with training–testing splits of 70–30 respectively

5.2.4 Summary

The accuracies for 70–30 training–testing splits for all the label combinations are shown in Fig. 6. There is an improvement in classification with the use of attribute difference and 2-means dataset. More number of *PID rate* (8), *PID reset rate* (6), *PID deadband* (9–10), system mode (15), and *bad crc* (18) attacks were classified correctly by attribute difference and 2-means datasets. However, *solenoid* attacks (14) could not be detected by the classifier with all the three datasets (except for one case using the attribute difference dataset).

5.3 Experimental Results for Read Response Attacks

The classification results for the read response attacks are provided for each dataset: original, attribute difference, and 2-means datasets.

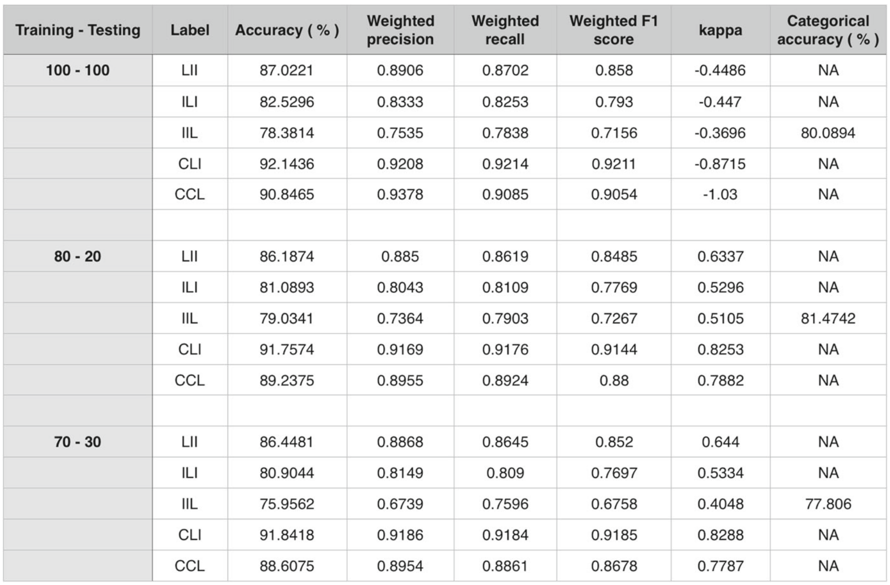
5.3.1 Original Dataset

The experimental results for the 70–30 training–testing sets for the IIL label are discussed here. All the normal packets were classified correctly. The *negative pressure* attacks (32) were classified correctly. The *slope* (27), *random value* (30), *fast* (33), and *slow* (35) attacks were not detected by the classifier. The false negative rate is 56.5%. This is unacceptably low. More than half of the attacks were classified as normal by the classifier. The accuracy of the classifier is 75.9562%.

As shown in the Fig. 7, the accuracies for 70–30 and 80–20 training–testing split are close to each other, except for the ILI (main category) case were 80–20 performed well.

5.3.2 Attribute Difference Dataset

The experimental results for the 70–30 training–testing sets for the IIL label are discussed here. We have the following observation with respect to the original dataset. Few of the normal packets (1%) were classified as attacks. The *slope* (27), *random value* (30), *fast* (33), and *slow* (35) attacks were detected by the classifier with this dataset (which were not detected using the original dataset). The false negative rate is reduced from 56.5 to 7.16% and is quite low when compared to the original dataset. A



Training - Testing	Label	Accuracy (%)	Weighted precision	Weighted recall	Weighted F1 score	kappa	Categorical accuracy (%)
100 - 100	LII	87.0221	0.8906	0.8702	0.858	-0.4486	NA
	ILI	82.5296	0.8333	0.8253	0.793	-0.447	NA
	IIL	78.3814	0.7535	0.7838	0.7156	-0.3696	80.0894
	CLI	92.1436	0.9208	0.9214	0.9211	-0.8715	NA
	CCL	90.8465	0.9378	0.9085	0.9054	-1.03	NA
80 - 20	LII	86.1874	0.885	0.8619	0.8485	0.6337	NA
	ILI	81.0893	0.8043	0.8109	0.7769	0.5296	NA
	IIL	79.0341	0.7364	0.7903	0.7267	0.5105	81.4742
	CLI	91.7574	0.9169	0.9176	0.9144	0.8253	NA
	CCL	89.2375	0.8955	0.8924	0.88	0.7882	NA
70 - 30	LII	86.4481	0.8868	0.8645	0.852	0.644	NA
	ILI	80.9044	0.8149	0.809	0.7697	0.5334	NA
	IIL	75.9562	0.6739	0.7596	0.6758	0.4048	77.806
	CLI	91.8418	0.9186	0.9184	0.9185	0.8288	NA
	CCL	88.6075	0.8954	0.8861	0.8678	0.7787	NA

Fig. 7 Performance of random forest classification for read response attacks using the original dataset for different training–testing splits and different labels

high number of attacks were detected with this dataset. The accuracy of the classifier increased from 75.9562 to 92.0015%.

As shown in the Fig. 8, the accuracies for 70–30 and 80–20 training–testing split are close to each other.

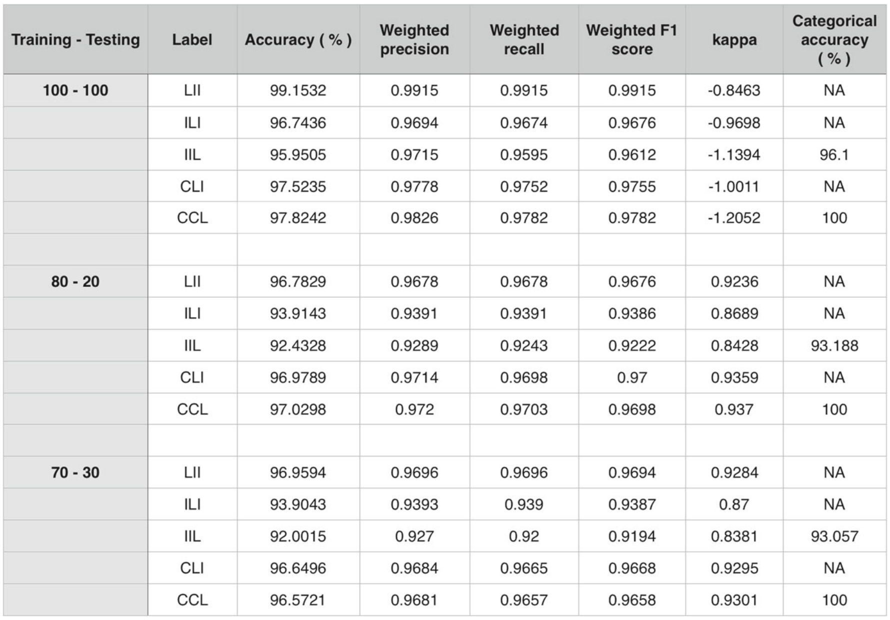
5.3.3 2-Means Dataset

The experimental results for the 70–30 training and testing sets for the IIL label are discussed here. We have the following observation with respect to the attribute difference dataset. The *random value* (39–31), *fast* (33–34), *rise/fall* (25), *slope* (28), and *slow* (35) attacks were detected far better by the classifier. The *rise/fall* (26) were detected more accurately with this dataset. However, the *negative pressure* attacks (32), *slow* (35), and *slope* (27) attacks were detected less accurately with respect to the attribute difference dataset. The false negative rate is 8.9% and is slightly higher compared to the attribute-difference dataset but is quite low when compared to the original dataset. The accuracy of the classifier increased from 92.0015 to 92.7375%.

As shown in the Fig. 9, the accuracies for 70–30 and 80–20 training–testing split are close to each other.

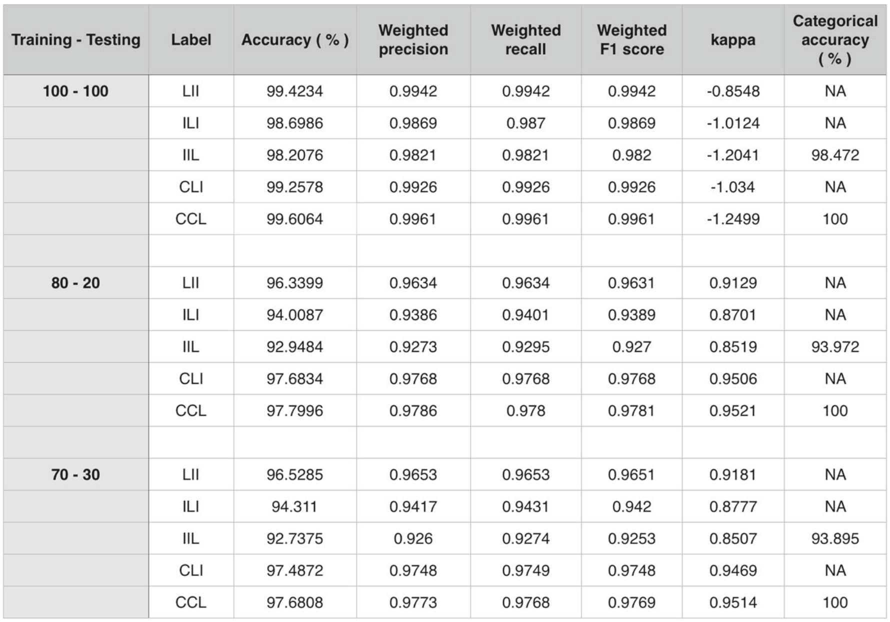
5.3.4 Summary

The accuracies for 70–30 training–testing splits for all the label combinations are shown in Fig. 10. There is an improvement in classification with the use of attribute difference and 2-means datasets. The *slope* (27), *random value* (30), *fast* (33), and



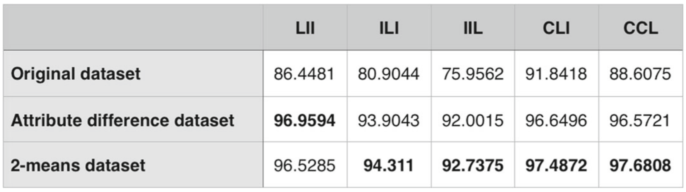
Training - Testing	Label	Accuracy (%)	Weighted precision	Weighted recall	Weighted F1 score	kappa	Categorical accuracy (%)
100 - 100	LII	99.1532	0.9915	0.9915	0.9915	-0.8463	NA
	ILI	96.7436	0.9694	0.9674	0.9676	-0.9698	NA
	IIL	95.9505	0.9715	0.9595	0.9612	-1.1394	96.1
	CLI	97.5235	0.9778	0.9752	0.9755	-1.0011	NA
	CCL	97.8242	0.9826	0.9782	0.9782	-1.2052	100
80 - 20	LII	96.7829	0.9678	0.9678	0.9676	0.9236	NA
	ILI	93.9143	0.9391	0.9391	0.9386	0.8689	NA
	IIL	92.4328	0.9289	0.9243	0.9222	0.8428	93.188
	CLI	96.9789	0.9714	0.9698	0.97	0.9359	NA
	CCL	97.0298	0.972	0.9703	0.9698	0.937	100
70 - 30	LII	96.9594	0.9696	0.9696	0.9694	0.9284	NA
	ILI	93.9043	0.9393	0.939	0.9387	0.87	NA
	IIL	92.0015	0.927	0.92	0.9194	0.8381	93.057
	CLI	96.6496	0.9684	0.9665	0.9668	0.9295	NA
	CCL	96.5721	0.9681	0.9657	0.9658	0.9301	100

Fig. 8 Performance of random forest classification for read response attacks using the attribute difference dataset for different training–testing splits and different labels



Training - Testing	Label	Accuracy (%)	Weighted precision	Weighted recall	Weighted F1 score	kappa	Categorical accuracy (%)
100 - 100	LII	99.4234	0.9942	0.9942	0.9942	-0.8548	NA
	ILI	98.6986	0.9869	0.987	0.9869	-1.0124	NA
	IIL	98.2076	0.9821	0.9821	0.982	-1.2041	98.472
	CLI	99.2578	0.9926	0.9926	0.9926	-1.034	NA
	CCL	99.6064	0.9961	0.9961	0.9961	-1.2499	100
80 - 20	LII	96.3399	0.9634	0.9634	0.9631	0.9129	NA
	ILI	94.0087	0.9386	0.9401	0.9389	0.8701	NA
	IIL	92.9484	0.9273	0.9295	0.927	0.8519	93.972
	CLI	97.6834	0.9768	0.9768	0.9768	0.9506	NA
	CCL	97.7996	0.9786	0.978	0.9781	0.9521	100
70 - 30	LII	96.5285	0.9653	0.9653	0.9651	0.9181	NA
	ILI	94.311	0.9417	0.9431	0.942	0.8777	NA
	IIL	92.7375	0.926	0.9274	0.9253	0.8507	93.895
	CLI	97.4872	0.9748	0.9749	0.9748	0.9469	NA
	CCL	97.6808	0.9773	0.9768	0.9769	0.9514	100

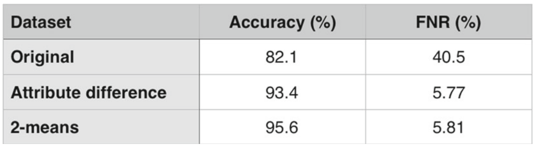
Fig. 9 Performance of random forest classification for read response attacks using the 2-means dataset for different training–testing splits and different labels



	LII	ILI	IIL	CLI	CCL
Original dataset	86.4481	80.9044	75.9562	91.8418	88.6075
Attribute difference dataset	**96.9594**	93.9043	92.0015	96.6496	96.5721
2-means dataset	96.5285	**94.311**	**92.7375**	**97.4872**	**97.6808**

Fig. 10 Classification accuracies for read response attacks using different datasets with training–testing splits of 70–30 respectively

Fig. 11 Performance statistics for read response attacks using Matlab's Ensemble Bagged trees with holdout—30% using different datasets



Dataset	Accuracy (%)	FNR (%)
Original	82.1	40.5
Attribute difference	93.4	5.77
2-means	95.6	5.81

slow (35) attacks that were not detected with the original dataset were detected and classified almost correctly with the attribute difference and 2-means datasets.

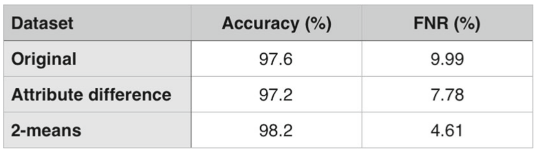
5.4 Summary of Experiments

The attribute difference dataset and the 2-means dataset gave high classification accuracies when compared to the original dataset for both the write command attacks and the read response attacks. The *solenoid* (14) attacks could not be detected by either the original dataset and the 2-means dataset, however a single instance of this attack was detected by the attribute difference dataset. The *pump* (13) attack's classification accuracy was the same for all the three datasets (original dataset, attribute difference dataset, and 2-means dataset). Overall, the classification accuracies were significantly improved with the attribute difference and the 2-means datasets.

5.5 Discussion on Performance of Apache Mahout

The experiments for classification of read response attacks and write command attacks for IIL (specific category) are conducted using Matlab's classification learner with a training–testing split of 70–30. The ensemble bagged trees algorithm is used as the classification algorithm here as it gave good results compared to other classifiers. The holdout validation scheme is used for all the experiments where 30% of data is held out for testing. The test set is picked up randomly by the classifier [16] unlike the time-ordered test set used for Mahout's experiments. Hence, the training–testing sets used in Matlab's experiments are different from those used for Mahout's experiments. The results of these experiments for all the three datasets are shown in Figs. 11 and 12.

For read response attacks, Matlab's ensemble bagged trees method yielded around 3% accuracy better than the Mahout's random forest. For write attacks, Matlab's ensemble bagged trees method performed significantly better than Mahout's random



Dataset	Accuracy (%)	FNR (%)
Original	97.6	9.99
Attribute difference	97.2	7.78
2-means	98.2	4.61

Fig. 12 Performance statistics for write command attacks using Matlab's Ensemble Bagged trees with holdout—30% using different datasets

forest with almost 8% improvement in accuracy. Moreover, the addition of new features (attribute difference dataset and 2-means dataset) significantly improved the performance of the classifier. The false negative rate is reduced from 40.5 to 5.8% in case of read response attacks. For write attacks, the false negative rate is lowered by 2.21% with attribute difference dataset. The 2-means dataset further lowered the false negative rate to 4.61% (compared to false negative rate of 9.99% for the original dataset). The *solenoid* (14) attacks (write command attack) which were not detected by Mahout's random forest were detected with Matlab's ensemble bagged trees algorithm with a classification accuracy of 84.86%. The Mahout's random forest classification algorithm must be improved to give better results, especially for write command attacks. The addition of new features not only improved accuracy of the classifier but also gave low false negative rate values. Thus, the new features are good and are critical for detecting attacks.

6 Conclusion

The principal objective of this study is to categorize the attacks in a SCADA system using a big data ecosystem. The dataset in this study is basically for comparison purposes, and eventually we will be using huge datasets in real time SCADA systems. This study provides a basis on how to deal with SCADA attacks on a big data framework. Apache Mahout's random forest is used to classify the attacks into major categories or specific categories on a Hadoop framework. The partition size should be set to the amount of data that can be loaded into the memory without failing the classification job. Based on the size of the partition, the number of splits can be determined. It may be unnecessary to have too many splits with small partition sizes. Reducing the number of partitions may improve the classification accuracy. The classification is performed by using the information present in the current packet and the previous packet(s). It is difficult to determine whether a packet is normal or not by looking at the attribute values without considering attribute values from previous packets. A packet could be normal at one time and could be attack another time. Thus, it is critical to use the information present in the current packet and the previous packet(s) while categorizing attacks. New features have been extracted from the original attributes. The history of the original attributes is analyzed using K-means clustering. The experiments were conducted using three different datasets: original dataset, attribute difference dataset, and 2-means dataset. A 2% to 18% improvement in classification accuracy was observed for the write command attacks and a 5–17% improvement was observed for the read response attacks for a 70–30 training–testing splits. The classification accuracy is significantly improved through this approach of using the information present in both the previous packet(s) and the current packet.

The experimental results for Matlab's ensemble bagged trees classification algorithm show that Mahout's random forest must be improved to give better results, especially for classification of write command attacks. Mahout's random forest can work with very large data but it should be more powerful than the classifiers running on a single system. The FNR reduced significantly with the addition of new features. Our new features improved the performance of the classifier for both Mahout's random forest and Matlab's ensemble bagged trees algorithm.

Instead of using 2-means, we could find the difference between the current and the most different packet in the history, or we could use the maximum of differences for the history of p packets. This could be studied as future work. Since in our study 2-means generated satisfactory results, we did not perform further experiments. We should note that our method does not rely on proper clustering of normal and attack packets. Adding additional attributes may affect the distance function used in k-means. For proper clustering, it may be critical. Our method just extracts two sets of mean values from the history of p packets. However, the effect of the number of clusters could be studied as future work.

Acknowledgements We would like to thank Dr. Morris and Ian Turnipseed for providing us with the SCADA dataset for our experiments.

References

1. Amin S, Litrico X, Sastry S, Bayen AM (2013) Cyber security of water scada systems. Part I: analysis and experimentation of stealthy deception attacks. IEEE Trans Control Syst Technol 21(5):1963–1970
2. Apache Software Foundation. Mahout. Accessed 24 Oct 2016
3. Beaver JM, Borges-Hink RC, Buckner MA (2013) An evaluation of machine learning methods to detect malicious scada communications. In: 2013 12th International conference on machine learning and applications (ICMLA), vol 2, pp 54–59
4. Carcano A, Coletta A, Guglielmi M, Masera M, Fovino IN, Trombetta A (2011) A multidimensional critical state analysis for detecting intrusions in scada systems. IEEE Trans Ind Inf 7(2):179–186
5. Deka D, Baldick R, Vishwanath S (2014) Optimal hidden scada attacks on power grid: a graph theoretic approach. In: 2014 International conference on computing, networking and communications (ICNC), pp 36–40
6. Fahad A, Tari Z, Almalawi A, Goscinski A, Khalil I, Mahmood A (2014) Ppfscada: privacy preserving framework for scada data publishing. Future generation computer systems, 37(Supplement C):496 – 511. Special Section: innovative methods and algorithms for advanced data-intensive computing special section: semantics, intelligent processing and services for big data special section: advances in data-intensive modelling and simulation special section: hybrid intelligence for growing internet and its applications
7. Gao W, Morris T, Reaves B, Richey D. On SCADA control system command and response injection and intrusion detection. Mississippi State University. Accessed 24 Oct 2016
8. Hadoop (2014) HowManyMapsAndReduces. https://wiki.apache.org/hadoop/HowManyMapsAndReduces. Accessed 24 Oct 2016
9. Hink RCB, Beaver JM, Buckner MA, Morris T, Adhikari U, Pan S (2014) Machine learning for power system disturbance and cyber-attack discrimination. In: 2014 7th International symposium on resilient control systems (ISRCS), pp 1–8
10. Holte RC (1993) Very simple classification rules perform well on most commonly used datasets. Mach Learn 11(1):63–90
11. Hsu J, Mudd D, Thornton Z (2014) Mississippi State University Project Report—SCADA anomaly detection. Accessed 24 Oct 2016

12. Hu W, Liao Y, Vemuri V (2003) Robust support vector machines for anomaly detection in computer security. In: Proceedings of the international conference on machine learning and applications, pp 23–24. Accessed 24 Oct 2016
13. Maglaras LA, Jiang J (2014) Intrusion detection in scada systems using machine learning techniques. Science and information conference (SAI) 2014:626–631
14. Maglaras LA, Jiang J (2014) Ocsvm model combined with k-means recursive clustering for intrusion detection in scada systems. In: 10th International conference on heterogeneous networking for quality, reliability, security and robustness, pp 133–134
15. Maglaras LA, Jiang J, Cruz TJ (2016) Combining ensemble methods and social network metrics for improving accuracy of OCSVM on intrusion detection in SCADA systems. J Inf Secur Appl 30(Supplement C):15–26
16. Matlab Community (2015) MATLAB Answers. https://www.mathworks.com/matlabcentral/answers/251265-holdout-validation-data-taken-randomly-3-questions. Accessed 24 Oct 2016
17. Miller B, Rowe D (2012) A survey scada of and critical infrastructure incidents. In: Proceedings of the 1st annual conference on research in information technology, RIIT '12. ACM, New York, pp 51–56
18. Mo Y, Chabukswar R, Sinopoli B (2014) Detecting integrity attacks on scada systems. IEEE Trans Control Syst Technol 22(4):1396–1407
19. Morris T, Gao W Industrial control system network traffic data sets to facilitate intrusion detection system research. In: Shenoi S, Butts J (eds) Critical infrastructure protection VIII—8th IFIP WG 11.10 International conference, ICCIP 2014, March 17–19, 2014, Revised Selected Papers, vol 441 of IFIP advances in information and communication technology, chapter 1. Springer, Arlington, pp 5–18
20. Morris T, Thornton Z, Turnipseed I (2015) Industrial control system simulation and data logging for intrusion detection system research. In: 7th Annual southeastern cyber security summit, Huntsville, AL
21. Nader P (2015) One-class classification for cyber intrusion detection in industrial systems. Dissertation, University of Technology of Troyes
22. Nader P, Honeine P, Beauseroy P (2014) l_p-norms in one-class classification for intrusion detection in scada systems. IEEE Trans Ind Inf 10(4):2308–2317
23. Perdisci R, Gu G, Lee W (2006) Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems. In: Proceedings of the international conference on data mining, pp 488–498. Accessed 24 Oct 2016
24. Shosha AF, Gladyshev P, Wu SS, Liu CC (2011) Detecting cyber intrusions in scada networks using multi-agent collaboration. In: 2011 16th International conference on intelligent system applications to power systems, pp 1–7
25. Sinclair C, Pierce L, Matzner S (1999) An application of machine learning to network intrusion detection. In: Proceedings of the computer security applications conference, p 371. Accessed 24 Oct 2016
26. Sridhar S, Manimaran G (2010) Data integrity attacks and their impacts on scada control system. In: IEEE PES general meeting, pp 1–6
27. Tan PN, Steinbach M, Kumar V (2006) Introduction to data mining. Pearson Addison Wesley, Boston, San Francisco (Paris). Table des matires l'adresse suivante. http://www.loc.gov/catdir/toc/ecip0510/2005008721.html
28. Teixeira A, Dn G, Sandberg H, Johansson KH (2011) A cyber security study of a scada energy management system: stealthy deception attacks on the state estimator*. In: IFAC Proceedings volumes, 18th IFAC World Congress 44(1):11271–11277
29. Torrisi NM, Vukovi O, Dn G, Hagdahl S (2014) Peekaboo: a gray hole attack on encrypted scada communication using traffic analysis. In: 2014 IEEE international conference on smart grid communications (SmartGridComm), pp 902–907
30. Turnipseed I (2015) A new SCADA dataset for intrusion detection system research. Master's thesis, Mississippi State University
31. Yasakethu SLP, Jiang J (2013) Intrusion detection via machine learning for scada system protection. In: Proceedings of the 1st international symposium on ICS & SCADA cyber security research 2013, ICS-CSR 2013. BCS, UK, pp 101–105
32. Yu N, Shah S, Johnson R, Sherick R, Hong M, Loparo K (2015) Big data analytics in power distribution systems. In: Innovative smart grid technologies conference (ISGT), 2015 IEEE power energy society, pp 1–5

33. Zhu B, Joseph A, Sastry S (2011) A taxonomy of cyber attacks on scada systems. In: 2011 International conference on internet of things and 4th international conference on cyber, physical and social computing, pp 380–388