EDMAND: Edge-Based Multi-Level Anomaly Detection for SCADA Networks

Wenyu Ren*, Timothy Yardley* and Klara Nahrstedt*
* University of Illinois Urbana-Champaign, Urbana, Illinois, USA
Email: {wren3, yardley, klara}@illinois.edu

***Abstract*—Supervisory Control and Data Acquisition (SCADA) systems play a critical role in the operation of large-scale distributed industrial systems. There are many vulnerabilities in SCADA systems and inadvertent events or malicious attacks from outside as well as inside could lead to catastrophic consequences. Network-based intrusion detection is a preferred approach to provide security analysis for SCADA systems due to its less intrusive nature. Data in SCADA network traffic can be generally divided into transport, operation, and content levels. Most existing solutions only focus on monitoring and event detection of one or two levels of data, which is not enough to detect and reason about attacks in all three levels. In this paper, we develop a novel edge-based multi-level anomaly detection framework for SCADA networks named EDMAND. EDMAND monitors all three levels of network traffic data and applies appropriate anomaly detection methods based on the distinct characteristics of data. Alerts are generated, aggregated, prioritized before sent back to control centers. A prototype of the framework is built to evaluate the detection ability and time overhead of it.**

I. Introduction

Supervisory Control and Data Acquisition (SCADA) systems are industrial control systems (ICSs) used for real-time monitoring, data collection, and control for large-scale distributed critical infrastructure systems such as power grids, oil/gas pipelines and refineries, water distribution and treatment. SCADA systems play a critical role in the operation of industrial systems by collecting data from fields sensors located at remote sites and issuing commands to actuators for control purpose. Communication protocols are developed specifically for ICSs like SCADA systems and some of the most commonly used ICS protocols include Modbus (for oil/gas refineries), and DNP3 (for power utilities) [1].

Critical as they are, SCADA systems are subject to a wide range of serious threats in recent years and they could suffer from catastrophic consequences due to successful attacks. Well-known malicious cybersecurity incidents in SCADA systems include the Stuxnet worm attack [2] and the BlackEnergy malware [3]. These attacks exploited the vulnerabilities of SCADA systems and the situation is expected to deteriorate further for several reasons. First, the adoption of cutting-edge communication technologies contributes to the increasing complexity and interconnection of SCADA systems, which potentially provides greater opportunity for attacks from malicious sources. Since corporate intranets can be connected to the internet, SCADA systems connections with corporate intranets may expose their communication weakness to threats of broader aspects. Second, devices in SCADA systems are usually not built with cybersecurity in consideration and lack authentication or encryption mechanisms. To make things worse, the enabling of remote access to these devices via wireless technologies makes them easy to compromise. Third, most ICS protocols lack authentication features and provide no protection for the network traffic. The vulnerabilities of SCADA systems can be exploited from both outside by malicious attackers and inside by disgruntled employees. Besides deliberate attacks, inadvertent events such as natural disasters, device failures, and operator mistakes may also jeopardize SCADA systems due to those vulnerabilities. Therefore, developing techniques to target those vulnerabilities and provide security to SCADA systems is a pertinent topic of particular importance.

In general, two types of analysis are available to provide security for SCADA systems: host-based and network-based. We focus on network-based analysis which monitors and inspects network traffic due to its less intrusive nature. Based on different analysis granularity, data in SCADA network traffic generally can be divided into three levels: transport level, operation level, and content level. Transport level data refers to statistics in IP headers and transport protocol headers. Operation level data refers to operation statistics in ICS protocols. Content level data refers to measurement statistics from field devices. Among all network-based security analysis approaches for SCADA systems, most existing solutions only focus on monitoring and event detection of one or two levels of data, which is not enough to detect and reason about attacks in all three levels. Also, data in each level has its own characteristics, which requires distinct methods to deal with. In this paper, we develop an edge-based multi-level anomaly detection framework for SCADA networks, named EDMAND. EDMAND is located inside the remote substations, which are the edges of the SCADA network. It contains a multi-level anomaly detector to monitor all three levels of network traffic data passing by. Appropriate anomaly detection methods are applied based on the distinct characteristics of data in various levels and alerts are generated, aggregated, prioritized, and sent back to control centers when anomalies are detected.

The remainder of this paper is organized as follows: Section II reviews the related work. Section III introduces the network architecture of SCADA systems and two of our design decisions. Section IV gives an overview of the design of EDMAND. Section V shows the performance evaluation of EDMAND and Section VI concludes the paper.

978-1-5386-7954-8/18/$31.00 ©2018 IEEE

II. Related Work

As we mentioned in the previous section, SCADA network traffic data can be categorized into three levels but most existing network-based intrusion detection only take one or two levels into consideration. [4], [5] focus on flow-level data while [6]–[10] analyze ICS protocol functions. [11], [12] only concentrate on content-level and [13], [14] cover the flow and operation levels. None of these approaches analyze all three levels of data and therefore may fail to detect anomalies in levels not covered. Moreover, a sophisticated multi-step attack may introduce anomalies in multiple levels of traffic data. The whole picture of the attack can be seen only when all three levels of anomalies are detected.

The most similar work to ours is [15]. The authors develop a multiattribute SCADA-specific intrusion detection system. The system uses white lists and behavior-based rules to analyze multiple attributes in transport, operation, and content levels to mitigate various cyberattacks. However, without any method to filter and prioritize alerts, the operator can easily be overwhelmed by false positives or low-priority alerts and miss the high-priority ones. Also, in our framework, alerts in on one level will affect the alert triggering in the other two levels, which is helpful in reducing false alerts.

III. Network Architecture and Design Decision

In this section, we introduce the SCADA network architecture. Then we explain two important design decisions we made for the framework.

A. Network Architecture

A simplified architecture of SCADA network is shown in Figure 1. The major components in SCADA network include the Master Terminal Units (MTUs) in the control centers, remote devices in the substations and the communication network that connects them. The remote devices can be Remote Terminal Units (RTUs), Programmable Logic Controllers (PLCs), or Intelligent Electronic Devices (IEDs), which further connect to and receive measurements from field devices such as sensors or actuators. The MTU in the control center queries the remote devices for system updates and may also issue control commands to them to change the control strategy. To avoid further data collection time and achieve prompt anomaly detection, we deploy EDMAND at the edge of the SCADA network. To be more specific, EDMAND is deployed in each substation between the remote devices and the wide area network. EDMAND monitors all traffic passing by and sends alerts back to control centers when anomalies are detected.

B. Design Decision

We made two important decisions while designing our framework. The first one is to divide traffic data into multiple levels and apply appropriate anomaly detection mechanisms to data in each level based on their characteristics. As we mentioned previously, data in SCADA traffic can be divided into three levels: transport level, operation level, and content level. Data in each level have their own characteristics, which is taken into consideration when we select anomaly detection mechanisms for each level.

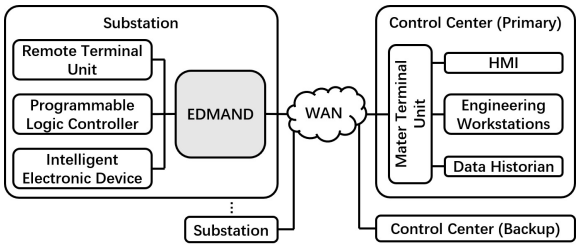


Fig. 1. SCADA network architecture

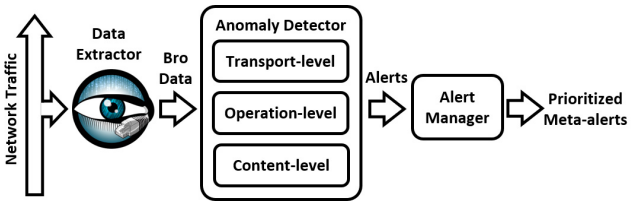


Fig. 2. EDMAND architecture

The second design decision is to introduce the concept of confidence into the anomaly detection process and assign confidence scores to generated alerts. We define an alert's confidence score $CS \in [0, 1]$ to be the confidence that the alert is indeed an anomaly. We calculate the confidence score by $CS = MA \times AS$, where MA is the model accuracy and AS is the anomaly score. The model accuracy measures the accuracy of our anomaly detection model in describing normal behavior and serves as the weight of the anomaly score. We assume that the majority of traffic data is normal data and therefore we can build models with higher accuracy as more samples are observed. In this sense, we estimate the model accuracy by a modified sigmoid function of observed sample number as $MA = 2/(1+e^{-n/N})-1$, where n is the observed sample number by the model and $N = 100$ is a normalization factor. The anomaly score measures how far the current sample deviates from the normal behavior described by the model. Different methods are used to calculate the anomaly score for different data and they are introduced in the next section.

IV. Framework Design

In this section, we present an overview of the modular design of EDMAND. As it is shown in Figure 2, EDMAND consists of 3 main components: (1)*Data Extractor*, (2)*Anomaly Detector*, (3)*Alert Manager*. The data extractor is implemented utilizing a network security monitor called Bro [16]. The data extractor monitors the network traffic passing by and forwards all three levels of network traffic data to the anomaly detector. The anomaly detector contains three levels and each level uses appropriate method to detect anomalies and generates alerts. After that, the alert manager aggregates similar alerts into meta-alerts. Priorities are given to meta-alerts and the alert manager reports meta-alerts to the control center with various frequencies according to their priorities. The anomaly detector and the alert manager will be described in more detail in the following two subsections.

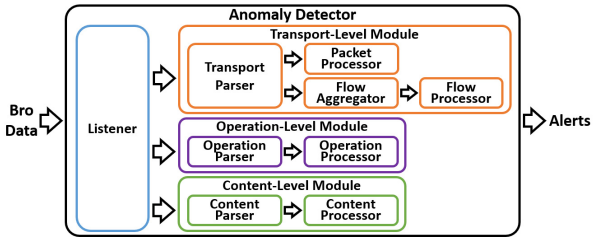


Fig. 3. Multi-level anomaly detector structure

A. Anomaly Detector

The structure of the multi-level anomaly detector is shown in Figure 3. There is a listener which receives Bro data from the data extractor and feeds them to the three modules for three levels. In each module, there is a parser that parses the Bro data corresponding to that level and translates them to standard input data for the processor. The processors implement various anomaly detection mechanisms to detect anomalies and generate alerts. We will introduce the three modules for three levels of data respectively.

1) Transport-Level Module: In the transport-level module, two kinds of analysis at different time scales are applied. A packet processor analyzes each packet for short-term analysis. A flow aggregator aggregates packet statistics every period $T_{flow} = 10$min and forwards to a flow processor for long-term analysis.

The input data to both processors consists of two kinds of fields: the index field which describes the packet or flow related with the input data, and data fields which store statistics for anomaly detection. As it is listed in Table I, the index fields for both processors share the same structure, which is a 4-tuple including originator(IP), responder(IP), transport protocol, and port number. The packet processor has interarrival time IAT and packet size PS as its data fields and the flow processor has packet count PC and average packet size APS as its data fields. Each type of data field has two values, corresponding to statistics of traffic in both directions.

TABLE I
INPUT FIELDS AND ANOMALY DETECTION MECHANISM OF PACKET PROCESSOR AND FLOW PROCESSOR

	Packet Processor	Flow Processor
Index Field	(originator, responder, transport protocol, port number)	
Data Field	interarrival time (IAT) packet size (PS)	packet count (PC) average packet size (APS)
Mechanism	1D-DenStream	Mean-STD

There are two types of anomalies for these two processors. The first type happens when input data with new index field are seen and the anomaly score is set to 1 in this scenario. The second type is abnormal value in data fields and we use various anomaly detection mechanisms to detect anomalies of this type. We mentioned previously that one of the design decision we made is to apply appropriate method to data with different characteristics. Since packet statistics and flow statistics follow quite different distributions, different anomaly detection mechanisms are utilized for the packet processor and flow processor. On the one hand, since traffic in SCADA usually follows periodic patterns [17]–[20], the packet count PC and average packet size APS in a certain period usually follows a unimodal distribution as long as the period is selected properly. Therefore, the mean and standard deviation are good enough to characterize these data fields. We build models for these data fields by calculating the exponentially-weighted mean and standard deviation. The anomaly score AS is calculated as the square of the anomaly score we used in [21] as

$$AS(X,\mu,\sigma) = \begin{cases} \left(1 - \frac{\sigma^2}{|X-\mu|^2}\right)^2 & \text{if } |X-\mu| > \sigma \\ 0 & \text{otherwise} \end{cases}, \quad (1)$$

where μ and σ are the mean and standard deviation stored in the model and X is the data field value of the current input (i.e., PC or APS for the flow processor). For convenience, we call this anomaly detection mechanism Mean-STD in the rest of this paper. On the other hand, the interarrival time IAT and the packet size PS of each packet usually follow multimodal distributions. Consider the following scenario, the control center is sending periodic read requests to a remote device. Each request is followed by a response from the remote device and then a confirmation from the control center. For packets from the control center to the remote device, read requests and confirmations could have big difference in packet size PS but both are considered as normal packets. For this reason, the mean and standard deviation may not be able to characterize these data fields and we utilize a clustering method instead. We use a modified 1D version of the DenStream in [22]. DenStream is an approach to cluster data in an evolving data stream and data is clustered into potential core-micro-clusters and outlier micro-clusters. An alert is generated whenever the new value point is added to an outlier micro-cluster and the anomaly score is calculated as $AS(w,\mu,\beta) = 1 - (w-1)/(\beta\mu - 1)$, where μ and β are predefined parameters and w is the weight of the outlier micro-cluster.

2) Operation-Level Module: The objective of the operation-level module of the anomaly detector is to detect anomalies in operations (e.g., requests and responses) of ICS protocols such as Modbus and DNP3. Similarly, the input data of the operation processor have an index field and a data field. We use a 5-tuple of (originator(IP), responder(IP), ICS protocol, unit id, function code) as the index field and interarrival time IAT as the data field. Here unit id is a ICS protocol specific address which is used to differential devices that share the same IP address. Notice that the IAT in operation level is different from the IAT in transport level. In operation level, the IAT is the difference in timestamps of two consecutive same operations between one pair of nodes (i.e., the two operations share the same index field). In transport level, the IAT is the difference in timestamps of two consecutive packets of the same direction between one pair of nodes which could be different operations or even non-ICS-protocol packets.

As it is shown in Table II, there are mainly three types of anomalies in this level. The first type includes invalid function code and wrong direction of operation. In normal

status, requests should only be sent by the control center and received by remote devices and responses should be sent by remote devices and received by the control center. Wrong direction here stands for unexpected scenarios such as requests initiated by remote devices or responses sent by the control center. For an anomaly of the first type, an alert is generated directly and a confidence scores of 1 is assigned. The second type of anomaly is the emerging of new operation, which is identified when input with new field index is observed. In this case, an anomaly score of 1 is given. The third type of anomalies includes scenarios of periodic operation arriving too early, arriving too late, or disappearing. In SCADA, the IAT of the same operation follows a unimodal distribution since operations are usually periodic. Therefore, the Mean-STD mechanism is used for anomaly detection and AS is calculated by equation 1 where X is replaced by IAT in operation level.

TABLE II
ANOMALY AND DETECTION MECHANISM IN OPERATION LEVEL

Anomaly	Mechanism
invalid function code	CS=1
wrong direction of operation	
new operation	AS=1
early operation	Mean-STD
late operation	
missing operation	

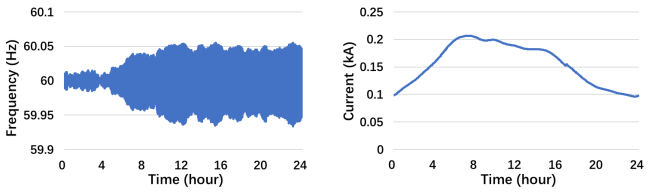
3) Content-Level Module: The content-level module of the anomaly detector is responsible for detecting anomalies in measurement values such as frequencies and voltages which are included in responses to read requests. The input data of the content processor have a 5-tuple of (measure source (IP), ICS protocol, unit id, measurement type, measurement index) as the index field and the measurement value itself as the data field. Depending on the measurement type, different methods are applied for anomaly detection. Let us take DNP3 for example, where the three measurement types are Binary, Analog, and Counter. Here we will discuss the first two which are most commonly seen.

For the Binary measurement type, the intuition behind the detection method is that a binary variable can only take two values (i.e., **0** or **1**) and always one of them is normal and the other is abnormal. Therefore, we can try to identify the normal value by simply counting the **0**s and **1**s in observed samples. The normal value is **0** if the majority of the observed values are **0**s and vice versa. Whenever the abnormal value appears, we calculate the anomaly score by one minus the entropy of observed samples as

$$AS(\gamma) = \begin{cases} 1 + \gamma \log_2 \gamma + (1-\gamma)\log_2(1-\gamma) & \text{if } 0 < \gamma < 1 \\ 1 & \text{if } \gamma = 0 \text{ or } 1 \end{cases}$$

$$\text{where } \gamma = \frac{\text{number of } \mathbf{0}\text{s observed}}{\text{number of samples observed}}.$$

For the analog measurement type, we take the Smart Grid as an example. Frequency, voltage, current, and power are four most common classes of analog measurements and they usually have quite different characteristics. The two subfigures in Figure 4 show one day of simulated frequency and current



(a) Frequency (b) Current

Fig. 4. One day's simulated measurement data of frequency and current

measurements from the Information Trust Institute's testbed [23]. We can see that 'frequency' is always around 60Hz and has a very small relative standard deviation whereas 'current' varies a lot but follows a diurnal pattern. Based on different analog classes' characteristics, we develop a 2-step anomaly detection method to analog measurements. In step 1, we further categorize analog measurements into different analog classes (i.e., frequency, voltage, current, power) and use an appropriate method for each class to detect anomalies in step 2. Notice that if the configuration files of remote devices are available and the specification of each analog index is known to our framework, step 1 can be neglected. Step 1 just provides analog class inference ability to our framework when specification is not given.

More specifically, in step 1 of our analog measurement anomaly detection, we utilize a similar Bayesian inference method as in [24] and build an analog class inference model based on the Bayesian network. Here we use a very simple Bayesian network with one root node and three leaf nodes shown in Figure 5. Each leaf node has a conditional probability table (CPT) representing the prior knowledge of the dependence between the child node and its parent node whose elements are defined by $CPT_{ij} = P(child\_state = j|parent\_state = i)$. The root represents the analog class with four hypothesis states and the leaf nodes represent directly observable evidences and each leaf node has several discrete states. The objective of this model is to calculate the belief in hypotheses of the root, which is decided by the likelihood propagated from its child nodes and ultimately the observed evidences at the leaf nodes. We denote y^k $(k \in 1 \ldots 3)$ as the observation at k^{th} leaf node and x_i $(i \in 1 \ldots 4)$ as the i^{th} analog class at the root node. Let $P(x_i)$ be the prior probability for the hypotheses of the root. The prior probability and CPTs can either be acquired based on domain knowledge or calculated based on training data. The believe in the analog class x_i is represented by the conditional probability of x_i given the observation at all leaf nodes and can be calculated by

$$P\left(x_i|y^1, y^2, y^3\right) = \alpha P(x_i) \prod_{k=1}^{3} P\left(y^k|x_i\right),$$

where $\alpha = 1/P\left(y^1, y^2, y^3\right)$ and can be calculated by $\sum_i P\left(x_i|y^1, y^2, y^3\right) = 1$. If the believe of x_i is larger than a threshold $\theta_b = 0.7$, the inference model infers the analog measurement as class x_i. Since the analog class for the same series of measurements will not change in normal status, the inference model stops analyzing for that series of measurements after its class is successfully inferred.

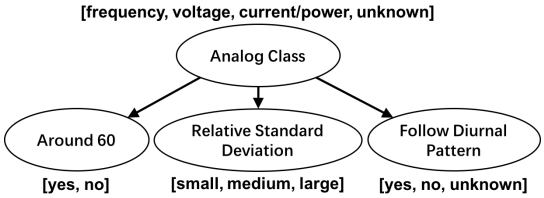


Fig. 5. Analog class inference model

After the analog classes are inferred, different anomaly detection methods are applied as shown in Table III. Mean and standard deviation are used for frequency and voltage. For current or power, we divide 24 hours to multiple time slots and calculate mean and standard deviation in each of the slot throughout multiple days. For analog measurements not belonging to any of the mentioned classes, the 1D-DenStream method for the flow processor is utilized. Notice that this list of analog classes can be extended by incorporating prior knowledge of other analog classes into the inference model and taking their characteristics into consideration while selecting their anomaly detection methods.

TABLE III
ANALOG MEASUREMENT CLASS AND DETECTION MECHANISM

Analog Class	Mechanism
frequency	Mean-STD
voltage	
current/power	slotted Mean-STD
unknown	1D-DenStream

B. Alert Manager

The alerts generated by EDMAND's multi-level anomaly detector which have confidence score higher than a threshold θ_{CS} are forwarded into the alert manager. We use a dynamic mechanism for θ_{CS}. The initial threshold is set at a upper bound value of $\theta_{CS\_H} = 0.95$. For every alert with CS above the current threshold, we calculate the new threshold as $\theta_{CS\_new} = e^{-\lambda}(\theta_{CS\_cur} - \theta_{CS\_L}) + \theta_{CS\_L}$ where $\theta_{CS\_L} = 0.85$ is a lower bound and $\lambda = 0.05$ is a parameter can be tuned. While we exponentially decrease the threshold to approach the lower bound for triggered alerts, we also linearly increases the threshold to the upper bound as time goes by. Keeping an high threshold in normal state helps to decrease false alarms and decreasing the threshold when alarms are triggered helps to detect all anomalies related to the attack and is useful against multi-step attacks. Since the alert from the three levels share the same threshold, alerts in one level will lower the threshold, making it easier to detect simultaneous anomalies in the other two levels if there are any.

Alerts generated by different processors share the following common fields: index field, alert type, timestamp, confidence score, statistical fields and abnormal data. Index field is the same as the index field in the input data of the corresponding processor. Alert type is the description of the anomaly. Statistical fields include statistics in the data field such as current data value, mean, standard deviation, etc. Abnormal data is the original input data of the processor which triggering the alert. As it is shown in Figure 6, the alert manager consists of two components: the *Alert Aggregator* and the *Alert Scheduler*.

Fig. 6. Alert manager structure

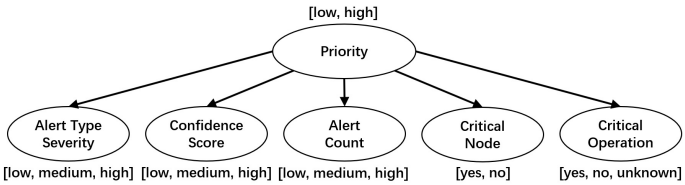


Fig. 7. Alert priority computation model

The objective of the alert aggregator is to aggregate alerts that share the same alert type as well as the index field and have little difference in timestamps. The aggregated alert is called the meta-alert. The meta-alert inherits all the fields from the alerts before aggregation with each field type having its own aggregation rule listed in Table IV. Another count field is added to store the number of alerts aggregated to this meta-alert. Whenever the alert aggregator receives a new alert, it tries to aggregate it to existing meta-alerts. If there is no meta-alert that this alert can merge to, a new meta-alert is created. In this way, consecutive duplicate alerts about the same event are aggregated to one meta-alert, which prevents alert flooding and simplifies further analysis of the alerts.

TABLE IV
META-ALERT FIELDS AND AGGREGATION RULES

Alert Field	Aggregation Rule
index field	shared by all of the aggregated alerts
alert type	
timestamp	keep minimum and maximum
confidence score	keep maximum
statistical fields	inherit from the last alert aggregated
anomaly data	
count	number of aggregated alerts

Every time a meta-alert is created or updated, it is forwarded to the alert scheduler, where its priority score is calculated and its report frequency is decided. The alert priority computation model in Figure 7 is similar to the analog class inference model. It is a Bayesian network with one root node and five leaf nodes. The root represents the alert priority which has two hypothesis states of low and high. Similarly, the leaf nodes represent observable evidence and each has several discrete states. We denote y^k $(k \in 1 \ldots 5)$ as the observation at k^{th} leaf node. We define the priority score PS as $PS = P\left(Priority = high | y^1, y^2, y^3, y^4, y^5\right)$ which can be calculated in a similar way as the analog class inference model. The prior probability of priority, CPTs at leaf nodes, and criteria to categorize observation are set by domain knowledge or system requirements. For example, the critical operation set for DNP3 can be defined by the DNP3 critical request function codes in [25] or entered by utilities according to their own needs.

Different report mechanisms in Table V are applied to meta-alerts based on their priority scores PS. After PS is calculated, the meta-alert is classified as high-priority or low-priority, based on PS and a threshold $\theta_p = 0.7$ as shown

in Table V. High-priority meta-alerts are always reported immediately when first created and reported with a small period if they are updated during that period. Low-priority meta-alerts are not reported immediately upon creation and reported with a large period if they are updated during the period. In the future, we plan to design a causality-based anomaly analyzer in the control center to further correlate and analyze the received alerts.

TABLE V
META-ALERT REPORT MECHANISM

	High-Priority	Low-Priority
Definition	$PS \geq \theta_p$	$PS < \theta_p$
Report when first created	yes	no
Report period	T_h	$T_l (> T_h)$

V. EVALUATION

In this section, we evaluate the performance of EDMAND in two aspects: detection ability and time overhead. The data extractor is implemented by Bro and the anomaly detector as well as the alert manager are implemented in Python. The evaluation is based on a simulated DNP3 traffic set which includes periodic baseline traffic and injected anomalies in transport, operation, and content levels. The baseline traffic consists of 10 days of simulated traffic of one control center sending read requests to two remote devices every 20 seconds. Each read request is followed by a TCP acknowledgement as well as a response from the remote device. After the response which contains the requested measurements is received by the control center, the control center sends a confirmation which is again followed by an acknowledgement from the remote device. Each remote device contains 5 measurements: one binary measurement and four analog measurements including frequency, voltage, current and power. Those measurements are simulated data from Information Trust Institute's testbed.

The analog class inference model correctly identifies all analog classes in the baseline traffic. We inject various anomalies in the three levels listed in Table VI to evaluate EDMANDs anomaly detection ability. EDMAND is able to detect all the anomalies injected with no false alarms. All the anomalies generate 12135 alerts in total, which are aggregated to 22 meta-alerts. We also list in Table VI some related works' detection abilities based on their description for anomalies which we injected. None of them are able to detect all anomalies like EDMAND does.

We also create an simple multi-step attack scenario with simulated traffic. In step 1, the attacker scan several ports in a given IP range to find the target field device and the ICS protocol the SCADA system is using. In step 2, the attacker send a write request to the field device to compromise the device. In step 3, the compromised device send tampered data in responses to read requests from the control center. EDMAND is able to detect all three steps of the attack where the three steps are detected in transport level, operation level, and content levels respectively. A framework only concentrating on one or two levels of data may not be able to see the whole picture of the attack.

One of our design decisions is to apply appropriate anomaly detection mechanisms to data, based on their characteristics. We use the 1D-DenStream mechanism for the packet processor since its data fields (i.e., interarrival time and packet size) follow multimodal distributions. To validate this design decision, we use the Mean-STD mechanism instead for the packet processor. We find that the modified packet processor is unable to detect the padded response and the delayed TCP acknowledgement, and generates tons of false alarms. This proves that selecting appropriate anomaly detection mechanism according to data characteristics is important. We also validate the alert priority computation model by calculating priority scores of meta-alerts triggered by two anomalies. The first anomaly is that the control center suddenly starts to send periodic write request (critical operation) to the remote device, which is considered a critical node. The second anomaly is the delay of one TCP acknowledgement from a remote device which is not a critical node. The meta-alert for the first anomaly has a priority score of 0.995, which is higher than the score of 0.439 for the second anomaly. This is consistent with the fact that the first anomaly is more critical than the second one.

To demonstrate EDMANDs ability to satisfy the real-time requirements of anomaly detection in SCADA systems, we also evaluate the time overhead of data analysis in the three levels and the alert manager. We run our experiments on a Ubuntu 16.04 desktop with 12 Intel Xeon 3.60GHz CPUs and 16GB memory. The data extraction in Bro and the anomaly detection for the three levels run in parallel. The total analysis time (data extraction time + anomaly detection time) per packet of the three levels are 3.87ms for transport level, 6.66ms for operation level, and 1.94ms for content level. Since the common data collection interval in SCADA systems is seconds or even minutes, several milliseconds of time overheads per packet are short enough for the packets to be processed in communication line speed. The average time overhead of the anomaly manager for each alert is 423ms. Since the rate of alerts is far smaller than the rate of packets, the overhead of 423ms for each alert is still practical.

VI. CONCLUSION

In this paper, we present EDMAND, an edge-based multi-level anomaly detection framework for SCADA systems. EDMAND resides in remote substations of SCADA systems and monitors network traffic of flow level, operation level, and content level. Distinct data characteristics are taken into consideration when selecting anomaly detection method for each level. When anomalies are detected, EDMAND generates, aggregates, and prioritizes alerts and send them to control centers. The performance of EDMAND is validated by experiments. EDMAND is only a component of our entire framework. In the future, we are going to build a anomaly analyzer which takes the meta-alerts from the three levels as inputs and use causal reasoning to analyze the causes and consequences of the anomalies. The correlation of anomalies in the three levels will be mainly utilized in the analyzer.

TABLE VI
INJECTED ANOMALIES AND DETECTION ABILITY COMPARISON

Level	Anomaly	Detection Ability					
		EDMAND	[4]	[9]	[11]	[13]	[15]
Transport	add a new node to send several packets to one remote device	yes	yes	no	no	yes	yes
	pad one response from a remote device with more payload	yes	yes	no	no	yes	yes
	delay one TCP acknowledgement from a remote device intentionally	yes	yes	no	no	no	no
	send lots of ICMP packets in a short period to one remote device	yes	yes	no	no	yes	yes
Operation	send one operation with invalid function code to one remote device	yes	no	yes	may	yes	yes
	let one remote device send a control command to the control center	yes	no	yes	may	yes	yes
	delay one response from a remote device intentionally	yes	may	no	no	no	yes
Content	tamper the binary value from on remote device for a short period	yes	no	no	yes	no	yes
	introduce over voltage and under voltage tripping to voltage measurements	yes	no	no	yes	no	yes
	introduce over current tripping to current measurements	yes	no	no	may	no	may
	tamper the frequency value from one remote device for a short period	yes	no	no	yes	no	yes
	tamper the power value from one remote device for a short period	yes	no	no	may	no	may

ACKNOWLEDGEMENT

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000780.

DISCLAIMER

REFERENCES

[1] E. J. Byres, M. Franz, and D. Miller, "The use of attack trees in assessing vulnerabilities in SCADA systems," in *Proceedings of the international infrastructure survivability workshop*, 2004.

[2] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, p. 29, 2011.

[3] D. U. Case, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, 2016.

[4] L. A. Maglaras and J. Jiang, "Intrusion detection in scada systems using machine learning techniques," in *Science and Information Conference (SAI), 2014*. IEEE, 2014, pp. 626–631.

[5] R. Udd, M. Asplund, S. Nadjm-Tehrani, M. Kazemtabrizi, and M. Ekstedt, "Exploiting bro for intrusion detection in a SCADA system," in *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*. ACM, 2016, pp. 44–51.

[6] I. N. Fovino, A. Coletta, A. Carcano, and M. Masera, "Critical state-based filtering system for securing SCADA network protocols," *IEEE Transactions on industrial electronics*, vol. 59, no. 10, pp. 3943–3950, 2012.

[7] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, B. Pranggono, and H. Wang, "Intrusion detection system for IEC 60870-5-104 based SCADA networks," in *Power and Energy Society General Meeting (PES), 2013 IEEE*. IEEE, 2013, pp. 1–5.

[8] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner, "An evaluation of machine learning methods to detect malicious SCADA communications," in *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, vol. 2. IEEE, 2013, pp. 54–59.

[9] H. Lin, A. Slagell, C. Di Martino, Z. Kalbarczyk, and R. K. Iyer, "Adapting bro into scada: building a specification-based intrusion detection system for the dnp3 protocol," in *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*. ACM, 2013, p. 5.

[10] H. R. Ghaeini and N. O. Tippenhauer, "Hamids: Hierarchical monitoring intrusion detection system for industrial control systems," in *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*. ACM, 2016, pp. 103–111.

[11] W. Gao, T. Morris, B. Reaves, and D. Richey, "On SCADA control system command and response injection and intrusion detection," in *eCrime Researchers Summit (eCrime), 2010*. IEEE, 2010, pp. 1–9.

[12] J. Nivethan and M. Papa, "A SCADA intrusion detection framework that incorporates process semantics," in *Proceedings of the 11th Annual Cyber and Information Security Research Conference*. ACM, 2016, p. 6.

[13] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA security scientific symposium*, vol. 46. Citeseer, 2007, pp. 1–12.

[14] N. Saunders, B. Khanna, and T. Collins, "Real-time situational awareness for critical infrastructure protection," in *Smart Grid Communications (SmartGridComm), 2015 IEEE International Conference on*. IEEE, 2015, pp. 151–156.

[15] Y. Yang, K. McLaughlin, S. Sezer, T. Littler, E. G. Im, B. Pranggono, and H. Wang, "Multiattribute SCADA-specific intrusion detection system for power networks," *IEEE Transactions on Power Delivery*, vol. 29, no. 3, pp. 1092–1102, 2014.

[16] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[17] N. Goldenberg and A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63–75, 2013.

[18] N. Erez and A. Wool, "Control variable classification, modeling and anomaly detection in Modbus/TCP SCADA systems," *International Journal of Critical Infrastructure Protection*, vol. 10, pp. 59–70, 2015.

[19] S. Ponomarev and T. Atkison, "Industrial control system network intrusion detection by telemetry analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 252–260, 2016.

[20] R. R. R. Barbosa, "Anomaly detection in SCADA systems: a network based approach," 2014.

[21] W. Ren, S. Granda, T. Yardley, K.-S. Lui, and K. Nahrstedt, "OLAF: Operation-level traffic analyzer framework for Smart Grid," in *Smart Grid Communications (SmartGridComm), 2016 IEEE International Conference on*. IEEE, 2016, pp. 551–556.

[22] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM international conference on data mining*. SIAM, 2006, pp. 328–339.

[23] Cyber-physical experimentation environment for RADICS. https://iti.illinois.edu/research/energy-systems/cyber-physical-experimentation-environment-radics-ceer.

[24] X. Qin, "A probabilistic-based framework for infosec alert correlation," Ph.D. dissertation, Georgia Institute of Technology, 2005.

[25] DNP3 secure authentication. http://www.scadahackr.com/library/Documents/ICS_Protocols/DNP3%20Secure%20Authentication%20v5%202011-11-08.pdf.