The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

# Secure Industrial Control System with Intrusion Detection

**M Rayhan Ahmed Mithu,**[1] **Vadim Kholodilo,**[2] **Rajesh Manicavasagm,**[1]
**Denis Ulybyshev,**[1] **Mike Rogers**[1]
[1]Department of Computer Science, [2]College of Independent Programs
Tennessee Technological University, Cookeville, TN - 38501, United States
{mmithu42, vkholodil42, rmanicava42}@students.tntech.edu, {dulybyshev, mrogers}@tntech.edu

## Abstract

Detecting intrusions and anomalies in Industrial Control Systems at early stages is important to prevent process failure. Operator errors, device or equipment failures, and other non-network events could lead to a critical state. As a result, these events can indirectly lead to anomalous network traffic, and, thus, a manually configured IDS that uses network traffic alone can generate false positives and false negatives. In this paper, we propose a novel approach that uses machine learning and incorporates both network data and device state information to improve the detection accuracy. Our methodology can detect anomalies as well as their root causes, which is essential. To protect device state data, we use a secure data container to store log records for devices in cyber-physical systems and IDS results. The secure data container provides data protection in transit and at rest. It also supports role-based and attribute-based access control.

## Introduction

Industrial Control Systems (ICS) automate the manufacturing process. They are responsible for controlling and managing a potentially large number of field devices. The manufacturing industry has seen a huge rise in the adoption of ICS in recent years, and this growth will continue, according to the study report published by Market Research Future (MRFR) (Online7 2018). Many ICS are deployed in critical infrastructures such as the Smart Grid, health care systems, water purification systems, and nuclear plants. As a result, security breaches and attacks can have significant impact on human lives and are very costly. Securing these systems has become a priority as a result of the increasing number of attacks in this domain.

A primary method of securing an ICS uses intrusion detection systems to attempt to send an alarm when the state of the ICS has been compromised. These systems are often manually configured, and use rules constructed by domain experts that understand the network and device behavior of the complex ICS. Unfortunately, the complexity of configuration often gives less than optimal results that include false positives and false negatives. False positives report attacks that do not exist and false negatives report that traffic is normal when an attack is actually occurring. Furthermore, an Intrusion Detection System (IDS) is usually configured to consider network data only. Network data alone gives only a partial picture of the state of the ICS.

The on-going work presented in this paper attempts to mitigate the weaknesses of typical approaches. In particular, this work attempts to use machine learning techniques that incorporate state information of devices in the ICS to augment network information for intrusion detection. Unfortunately, such state information is vulnerable to manipulation by attackers just as is the network. Therefore, this work also incorporates a method that ensures that state data can be both securely stored and transferred by the intrusion detection system.

The contributions of this work are as follows.

- A Design and implementation of a system architecture for detecting ICS intrusions
- Identification of machine learning techniques to improve intrusion detection over typical IDS that use network data only
- Integration of a secure data-container technology to insure device state is not compromised.

The rest of the paper is organized as follows. Section 2 presents an overview of related work. In Section 3 the core design is presented. Experimental results are discussed in Section 4. Section 5 discusses the future work and section 6 concludes the paper.

## Related Work

The following related work focuses on the state of the art of IDS. The IDS is not only a popular software-based solution that is in practical use, but also has been a focus of research in securing ICS.

Morris, Vaughn, and Dandass (Morris, Vaughn, and Dandass 2012) proposed an IDS that focuses on devices that are connected with serial links. The authors proposed a Snort-based IDS for MODBUS® in both a passive mode and an inline mode that dropped packets.

A signature-based IDS with state-based rules and stand alone IDS rules was presented in a later work in (Gao and

Morris 2014). A critical state analysis-based approach was presented by Carcano et al. (Carcano et al. 2011) in SCADA systems using state information to detect intrusions. The IDS was implemented on the virtual image and an alert was triggered if the current state was critical or moving towards a critical state.

Unfortunately, both of the above mentioned approaches require expert knowledge to model normal and critical device state. They also do not address the problem of integrity of the state information from devices which can be attacked. Our approach uses machine learning algorithms to automatically identify important device state information and correlate it with the network packets. Moreover, we use a secure data container to store the device state information, which provides data privacy and integrity.

A multi-agent IDS was proposed by Tsan and Kwong (Tsang and Kwong 2005) for industrial networks. This implementation shows the improved performance of ant-based clustering. K-Means clustering achieves 89.17% average detection rate and 4.29% false positive rate with the Fast Independent Component Analysis (FastICA) feature extraction. Unfortunately, any false alarms could lead to disastrous consequences.

More recently, (Shen et al. 2018) proposed a solution to use device fingerprinting to enhance intrusion detection in ICS. The proposed solution uses inter-layer communication and network traffic to generate the fingerprints. In another approach (Kleinmann and Wool 2017) uses the highly periodic nature of the communication between HMI and PLC devices to model the communication pattern. The modeling is done with Statechart Deterministic Finite Automaton (DFA) that uses unsupervised learning. These two techniques rely on network traffic to improve the intrusion detection but in our approach, we focus on utilizing the device state information to help the IDS.

(Al-Mamory and Zhang 2009) proposed to use clustering and root cause analysis to reduce alarms generated by an IDS. They believe that, generalized alarms from clustering can suggest root causes. However, it would require expert knowledge of network security and the environment to validate the root causes. Our approach uses the device state information to identify the root cause of an alert by querying information from the secure data container. Further validation by experts are not required in our approach as well.

Ranchal et al (Ranchal et al. 2018) proposed the EPICS solution for web services to protect data throughout the service interaction lifecycle. This solution expands the Active Bundle concept (Lilien and Bhargava 2006), (Othmane and Lilien 2009), (Ranchal 2015) for data protection. Datasets are stored and transferred together with the access control policies and with the data disclosure monitor. Our SDC solution is substantially different since policies are enforced by a Trusted Third Party (TTP) and policy enforcement engine is not stored together with data.

Awad, Lopez, and Rogers (Awad, Lopez, and Rogers 2019) propose a framework for live memory forensics of level 0-1 devices by continually acquiring portions of volatile memory and monitoring changes. This work's primary use is for memory forensics, but could be complementary to our work as a source for generating log records.

## Core Design

Initially, ICS were designed for local processes isolated from Wide Area Networks (WANs) and, therefore, had little or no security. However, as processes became more complex and business needs grew, providing connectivity to WANs became necessary. Unfortunately, the unlimited access of the Internet made the ICS vulnerable to attacks. Furthermore, retrofitting ICS devices with new secure communication protocols is too expensive. In addition, field devices cannot run computationally intensive algorithms due to limited resources.

Therefore, IDS have become a popular security method for detecting attacks. An IDS can be added to an existing network at a strategic point without needing to modify existing devices.

However, in the industrial network it is not always enough to monitor the network traffic alone. For example, a sequence of (valid) control commands, device failures, faulty raw materials, etc. could lead to a critical situation in the process. This critical situation will likely result in unusual network traffic as the control devices attempt to notify the process control personnel. As a result, false positives indicating that the network is being attacked are generated by the IDS. Believing that a network attack is underway, the process control personnel might not respond correctly to the critical state, which could have disastrous results. Similarly, as has been demonstrated with ICS attacks such as StuxNet(Langner 2011), an attacker may modify network traffic to mimic normal behavior even though device state may indicate that something is wrong with the behavior of the physical system. Such an attack results in false negatives. In other words, the IDS will not alert the process control operators, which can also have disastrous results.

We propose a novel approach where the IDS can validate network traffic with the help from the device generating the traffic.The secure ICS architecture is illustrated in Figure 1. Log records of device state information are created and stored. The device state information is kept in an SDC which can be accessed upon request. Log records stored in SDC are protected from being attacked. Uncompromised log records are used to verify IDS findings on whether the network is compromised or not. Machine learning algorithms learn what device state is associated with which network communication patterns, and provide this information as a database to the IDS. The IDS will monitor network patterns and, when an anomaly is suspected, can query SDCs for involved devices, and the database to validate its suspicions. Secure ICS has five major components, illustrated in Figure 2, that are described as follows.

***Embedded Components***. The embedded components of the system monitor ICS devices in real time, create log records, and transfer the log records to the SDC. Two modules execute these actions. The first module is the ***State Monitor*** and the other module is the ***State Router***. The state monitor module is responsible for real time monitoring of the device state. It can collect device state information in various ways, including memory sampling, file sampling, and

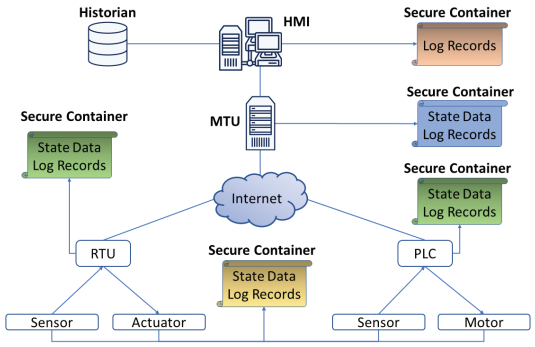


Figure 1: Secure Industrial Control System Architecture

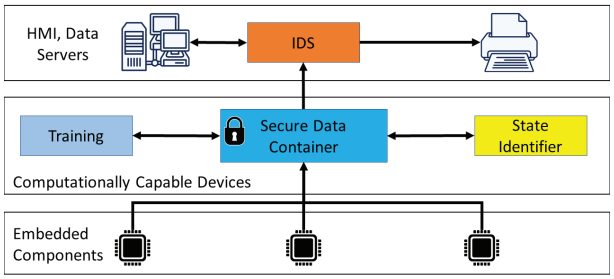


Figure 2: Components of Secure Industrial Control System

querying JTag interfaces. The device state information collected by the state monitor module is transferred by the state router module to the SDC.

***State Identification.*** This system component of the architecture is responsible for identifying and labelling important information about the state of the device. This module uses the information stored in the SDC to produce an image that represents an instantaneous state of the system that may result in a change in the network transmissions of the device.

***Secure Data Container.*** Our proposed solution for data protection in transit and at rest, as well as role-based and attribute-based access control, relies on the *Secure Data Container* (SDC). The SDC is a self-protecting data container that incorporates data in encrypted form, access control policies and metadata with watermarks in encrypted form, as shown in Figure 3. SDC is used to store log files and sensor data snapshots, as well as IDS results. It provides data confidentiality and integrity and guarantees that a client is able to access only those data subsets from an SDC for which the client is authorized. The concept is similar to the Active Bundle (Lilien and Bhargava 2006), (Othmane and Lilien 2009), (Ranchal 2015), (Ulybyshev et al. 2017), (Ranchal et al. 2018) and the Extended Attribute-Aware Active Bundle (EA3B) (Ulybyshev 2019) concepts, but the SDC has the following substantial differences:

- SDC does not store a policy enforcement engine together with data and access control policies. In contrast, policies are enforced on a trusted back-end server
- SDC is implemented both as an encrypted spreadsheet file and as a Java®[1] Archive (JAR), with a separate encryption key per separate data worksheet/subset.
- Different encryption/decryption key generation scheme.

Figure 1, based on the Purdue Enterprise Reference Architecture (Williams 1994), illustrates the SDC application in ICS. Each separate data subset in SDC is encrypted with a separate 256-bit AES key, generated on-the-fly, based on hash values of the following inputs:

- Metadata. It provides protection against attackers who try to modify access control policies and gain privileges in order to access unauthorized data.
- Authentication Server (AS)'s private key. This guarantees that only authorized services whose identity was verified by the AS can derive correct AES keys for decrypting accessible data subsets.
- Data subset's name. It provides fine-grained access control and makes the AES key different for each data subset. Fine-grained access control is useful in case the further forensics investigation needs to be done by authorized users once intrusion has been detected by an IDS.

The SHA-256 hash function is used in the AES key generation process. When the client requests data from an SDC, the client sends HTTP POST request, which contains client's username, hash of its password and what data subset(s) the client wants, to an AS, which validates credentials. If credentials are valid, AS responds to the client with a unique URL for a web service that corresponds to client's role. This URL contains an encrypted and signed authentication token (AT). AT includes the username, the requested data subsets' names, client's IP address and attributes, token expiration time and a hash value of AS's private key. Then the client application sends an HTTP GET request to the unique URL to a back-end server, which runs on the same host as an AS. Back-end server extracts the AT from the request, verifies the signature and extracts AT fields for policies, attributes and metadata evaluation. Not only is the client's role evaluated, but the client's attributes are as well, including the type of the client's device (mobile vs. desktop) and the authentication method (password-based vs. hardware-based). Based on these evaluations, the back-end server generates 256-bit AES keys on-the-fly for the accessible data subsets, retrieves these subsets from an SDC in a JSON form, decrypts them and responds back to the authorized client. Client communicates with an AS and a back-end server over https communication channel. Decryption keys are not stored inside the SDC or on any TTP. They are generated on-the-fly each time when a client requests data from an SDC.

To mitigate the insider's threat issue, SDC contains a digital watermark that is stored as a "magic" string in encrypted form in a "Metadata" worksheet. If this watermark is removed then the hash value of metadata will change. Since this hash value is one of the three inputs for the symmetric key derivation scheme, described above, changed metadata

[1]Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners (Online1 )

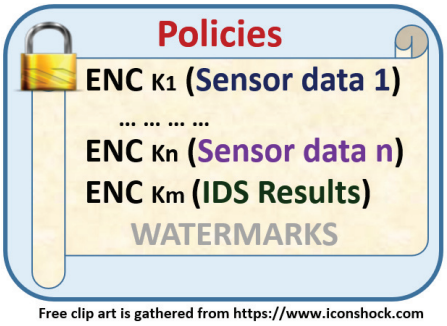


Figure 3: Secure Data Container

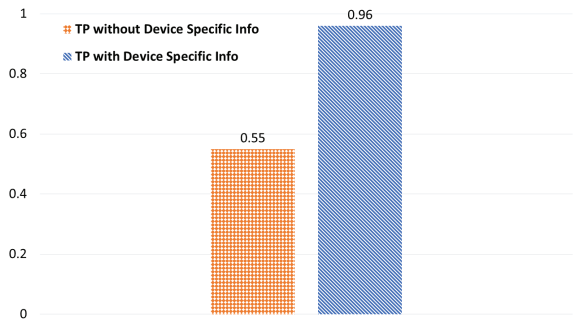


Figure 4: TP Rate for Attack Detection in Water Storage Dataset

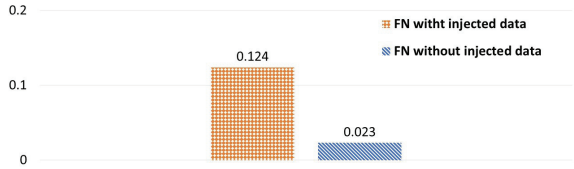


Figure 5: Increased FN with Injected Data

will result in the wrong decryption key derivation. Thus, removed watermark will make the SDC data unreadable. Unauthorized modification of access control policies or watermark removal lead to an incorrect decryption key derivation. SDC implementation as an encrypted spreadsheet file with RESTful API support provides an easier integration into existing ICS and IT infrastructures.

***Training.*** The *training module* is executed off-line to train the system such that it learns what device state results in particular network behavior using the information from the state identifying module. It also identifies the state information which is most likely to make significant changes in the network traffic. The training module is fed with data collected from the state collection component and data collected by sampling the network. The training module matches device state with generated signal state which can be signal over a wire, over serial lines, or over the network as packets, and stores the information in a query-able dataset. Machine learning algorithms are used to model the relationship between device state and specific network traffic. A dataset that represents this relationship is created from the model and stored for future use by the IDS. The IDS can query this information to verify its alerts.

***Intrusion Detection Process.*** In order to determine whether unusual network behavior is caused by an intrusion, the IDS in this component does not rely on network traffic alone. If an alert is generated from abnormal activity in the network, the IDS queries the information stored by the training module. The device state information related to the specific network data is used to verify the alert. This allows the IDS to reduce false positives. The IDS also receives real-time information from the state router and performs analysis to detect compromised device.

Additionally, If the IDS needs to verify its findings, it will be able to query the dataset of relationships that was created by the training module and reconstruct the cause and effect chains to trace the sources of the generated network and device state data. This feature is not currently integrated into our implementation, but will be in future iterations.

## Evaluation

Our initial evaluation includes experiments that show improved IDS results and the performance overhead of the SDC. To show improved IDS results, we derived an experiment that replayed a real dataset (Morris and Gao 2014) created by the Mississippi State University SCADA Security Laboratory and Power and Energy Research laboratory, which is a collection of network data from a water storage tank system. The dataset contains seven different classes of network traffic which includes one normal instance and six attack instances. The attack instances are naive malicious response injection, complex malicious response injection, malicious state command injection, malicious parameter command injection, malicious function command injection, denial-of-service and reconnaissance attacks. The unique features of the water storage system were HH, H, LL, L. H is the higher level and L is the lower level of water storage. The alarm is triggered if the water level is above the high alarm setpoint (HH) or below the low alarm setpoint (LL). Device state was derived based on the behavior of the process. We used the Naive Bayes algorithm to create a model with the dataset. The device state information increases the machine learning algorithm's true positive rate significantly (75%). The comparison is shown in Figure 4. Furthermore, we created a small test dataset with 231 records from the original dataset. 21 of the attack traffic was injected with normal traffic measurement. We performed this evaluation with the Decision Tree algorithm. The most important feature selected by the algorithm was the measurement of water level. The injected traffic had normal water level data which resulted in false negative (FN) predictions by the algorithm. The attack traffic was classified as normal by the algorithm. These shows that just monitoring network traffic can sometimes fool the IDS. The increment of false positive rate is shown in Figure 5.

To measure the performance overhead of the SDC, we collected data request Round-Trip Time (RTT) by measuring the time starting from a web service's data request to

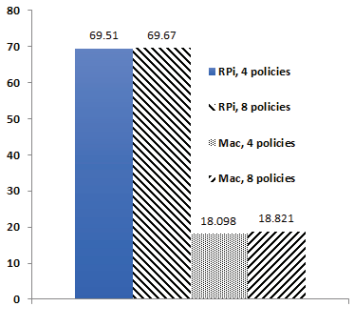


Figure 6: SDC Data Request Round-Trip Time

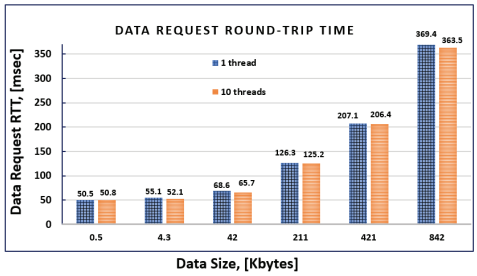


Figure 7: Data Request RTT for a Spreadsheet SDC

SDC and ending with the receipt of the response from SDC. RTT is a sum of times spent for authentication, evaluation of access control policies and the client's attributes, decryption key derivation and data retrieval. The ApacheBench®[2] utility (version 2.3) and web browser developer consoles were used for RTT evaluations. A web service (client) requesting data, an authentication server, and the SDC are running on the same host to exclude network delays from RTT measurements. Experiments were conducted on two system configurations.

1. Hardware: MacBook®[3] Pro, Intel®[4] Core i7 CPU @ 2.2 GHz, 16GB memory; OS: macOS® Sierra 10.12.6.

2. Hardware: Arm7™ [5] Processor rev 4 @1.2GHz, RAM 1GB (Raspberry Pi®[6] 3 Model B); OS: Raspbian GNU/Linux®[7] 9.1

In the following experiment, we measured RTT as an average of 50 requests for 16 bytes of data from SDC, varying hardware that hosts SDC and number of access control policies in SDC, implemented as a JAR® file.

As it can be seen from Figure 6, RTT strongly depends on hardware that hosts SDC, and slightly depends on number of access control policies in SDC. Typically, hardware on level 1 in the secure ICS architecture (see Figure 1) has less computationally powerful hardware to host SDC than on levels 2 and 3. On level 1, represented as a Raspberry Pi®, RTT increases by 284% for 4 policies and by 270.2% for 8 policies in SDC. When the number of access control policies is increased from 4 to 8, RTT increases by 0.23% for Raspberry Pi® and by 4% for MacBook Pro®.

In the next experiment, we measured RTT as an average of 1000 data requests, varying the amount of data requested from SDC and the number of concurrent threads from 1 to 10, requesting data from SDC, implemented as an encrypted spreadsheet file. A client, an authentication server, and the SDC are running on the same host to exclude network delays from RTT measurements. AS listens to an open port for incoming HTTP GET and POST requests. Experiments were conducted on the following system configuration:

- CPU: Intel® Core i5-8350U @ 1.7GHz; RAM: 8GB DDR4
- Microsoft Windows®[8] 10 Pro, 64 Bit
- Web framework: Node.js®[9] server, ver. v10.16.3

As it can be seen from Figure 7, RTT grows by 7.31 times for 1 concurrent data request and by 7.16 times for 10 concurrent requests, when the size of data retrieved from SDC increases from 0.5 to 842 kilobytes. In ApacheBench® utility, having 10 concurrent requests and 1000 total requests makes 10 requests open at a time. Node.js® is a single-threaded server with multiple threads to execute asynchronous code (Tesanovic ). As it can be seen in Fig. 7, the RTT for 10 concurrent threads is smaller than for one thread, except for data request for 0.5 kilobytes of data.

## Future Work

In our current iteration, the relationships between the device state data and the network traffic were manually derived. In the future, we will evaluate machine learning methods to automatically generate descriptions of the relationships between the network data and the important device state data. We also plan to determine how to incorporate learned behavior into the IDS to improve its results and to reduce false positives.

## Conclusions

In this paper, we presented a method to improve IDS true positive rate in ICS using extra information about device state. This extra information is securely stored and transferred to the IDS by employing secure data containers that protect log data and device state information in transit and at

[2]We do not claim association or endorsement of/for/by the Apache Software Foundation (ASF) (Online2 )

[3]This is an independent publication and has not been authorized, sponsored, or otherwise approved by Apple Inc.(Online3 )

[4]Intel is a trademark of Intel Corporation or its subsidiaries.(Online4 )

[5]Arm, Arm7, Arm7TDMI, Arm7TDMI-S and Arm7EJ-S are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. (Online5 )

[6]Raspberry Pi is a trademark of the Raspberry Pi Foundation (Online6 )

[7]Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries. (Online8 )

[8]This paper "Secure Industrial Control System with Intrusion Detection" is an independent publication and is neither affiliated with, nor authorized, sponsored, or approved by, Microsoft Corporation (Online9 )

[9]We do not imply sponsorship or endorsement by Node.js Foundation (Online10 )

rest. We have proposed a system for securely collecting and transporting state data and machine learning techniques that use this data to improve IDS results.

## Acknowledgements

We thank Christian Bare, Bradley Northern and Abhijeet Solanki, who are students at Tennessee Technological University, for their help with SDC implementation as a protected spreadsheet file. We thank Aala Oqab-Alsalem for her help with the experimental setup.

## References

Al-Mamory, S. O., and Zhang, H. 2009. Intrusion detection alarms reduction using root cause analysis and clustering. *Computer Communications* 32(2):419–430.

Awad, R. A.; Lopez, J.; and Rogers, M. 2019. Volatile memory extraction-based approach for level 0-1 cps forensics.

Carcano, A.; Coletta, A.; Guglielmi, M.; Masera, M.; Nai Fovino, I.; and Trombetta, A. 2011. A multidimensional critical state analysis for detecting intrusions in SCADA systems. *IEEE Trans. on Industrial Informatics* 7(2):179–186.

Gao, W., and Morris, T. 2014. On Cyber Attacks and Signature Based Intrusion Detection for Modbus Based Industrial Control Systems. *Journal of Digital Forensics, Security and Law* 9(1).

Kleinmann, A., and Wool, A. 2017. Automatic construction of statechart-based anomaly detection models for multithreaded industrial control systems. *ACM Trans. on Intelligent Systems and Technology (TIST)* 8(4):1–21.

Langner, R. 2011. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy* 9(3):49–51.

Lilien, L., and Bhargava, B. 2006. A scheme for privacy-preserving data dissemination. *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans* 36(3):503–506.

Morris, T., and Gao, W. 2014. Industrial control system traffic data sets for intrusion detection research. *IFIP Advances in Information and Communication Technology* 441:65–78.

Morris, T.; Vaughn, R.; and Dandass, Y. 2012. A retrofit network intrusion detection system for MODBUS RTU and ASCII industrial control systems. *Proc. of the Annual Hawaii Intl. Conf. on System Sciences* 2338–2345.

Online1. Third party usage guidelines for oracle trademarks. www.oracle.com/legal/trademarks.html. [Accessed: 2020-03-09].

Online10. Node.js foundation trademark guidelines for the node.js marks. www.nodejs.org/static/documents/trademark-policy.pdf. [Accessed: 2020-03-09].

Online2. Frequently asked questions about the asf's trademarks and their allowable uses. www.apache.org/foundation/marks/faq/. [Accessed: 2020-03-09].

Online3. Guidelines for using apple trademarks and copyrights. www.apple.com/legal/intellectual-property/guidelinesfor3rdparties.html. [Accessed: 2020-03-09].

Online4. Intel® trademark & company name usage guidelines. www.intel.com/content/www/us/en/trademarks/intel.html. [Accessed: 2020-03-09].

Online5. Using the arm7™ trademark. www.arm.com/company/policies/trademarks/arm-trademark-list/arm7-trademark. [Accessed: 2020-03-09].

Online6. Raspberry pi trademark rules and guidelines. www.raspberrypi.org/trademark-rules/. [Accessed: 2020-03-09].

Online7. 2018. Industrial control systems (ics) market 2018 global analysis, industry size, share leaders, current status by major key vendors and trends by forecast to 2023. www.marketwatch.com/. [Accessed: 2020-03-09].

Online8. The linux foundation trademark attribution. www.linuxmark.org/programs/legal/trademark/attribution. [Accessed: 2020-03-09].

Online9. Publications, seminars, & conferences guidelines. www.microsoft.com/en-us/legal/intellectualproperty/trademarks/usage/publications.aspx. [Accessed: 2020-03-09].

Othmane, L. B., and Lilien, L. 2009. Protecting privacy of sensitive data dissemination using active bundles. In *2009 World Congress on Privacy, Security, Trust and the Management of e-Business*, 202–213. IEEE.

Ranchal, R.; Bhargava, B.; Angin, P.; and Othmane, L. B. 2018. Epics: A framework for enforcing security policies in composite web services. *IEEE Trans. on Services Computing*.

Ranchal, R. 2015. Cross-domain data dissemination and policy enforcement.

Shen, C.; Liu, C.; Tan, H.; Wang, Z.; Xu, D.; and Su, X. 2018. Hybrid-augmented device fingerprinting for intrusion detection in industrial control system networks. *IEEE Wireless Communications* 25(6):26–31.

Tesanovic, V. Multi threading and multiple process in node.js. itnext.io/multi-threading-and-multi-process-in-node-js-ffa5bb5cde98. [Accessed: 2020-03-09].

Tsang, C. H., and Kwong, S. 2005. Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction. *Proc. of the IEEE Intl. Conf. on Industrial Technology* 2005:51–56.

Ulybyshev, D.; Bhargava, B.; Villarreal-Vasquez, M.; Alsalem, A. O.; Steiner, D.; Li, L.; Kobes, J.; Halpin, H.; and Ranchal, R. 2017. Privacy-preserving data dissemination in untrusted cloud. In *2017 IEEE 10th Intl. Conf. on Cloud Computing (CLOUD)*, 770–773. IEEE.

Ulybyshev, D. A. 2019. *Data Protection in Transit and at Rest with Leakage Detection*. Ph.D. Dissertation, figshare.

Williams, T. J. 1994. The purdue enterprise reference architecture. *Computers in industry* 24(2-3):141–158.