Misbehavior Detection of Embedded IoT Devices in Medical Cyber Physical Systems

Ilsun You, Kangbin Yim, Vishal Sharma, Gaurav Choudhary
Information Security Engineering
SoonChunhYang University

Ing-Ray Chen
Department of Computer Science
Virginia Tech

Jin-Hee Cho†
Department of Computer Science
Virginia Tech

ABSTRACT

We propose a lightweight specification-based misbehavior detection technique to efficiently and effectively detect misbehavior of an IoT device embedded in a medical cyber physical system through automatic model checking and formal verification. We verify our specification-based misbehavior detection technique with a patient-controlled analgesia (PCA) device embedded in a medical health monitoring system.

KEYWORDS

Medical cyber physical systems, IoT, behavior rules, zero-day attacks.

1 Introduction

In general, there are three types of misbehavior detection techniques for Internet of Things (IoT): signature-based, anomaly-based and specification-based techniques [12]. Our proposed misbehavior detection technique in this work falls under specification-based detection. We dispose signature-based detection as it cannot deal with zero-day attacks. We consider specification-based techniques rather than anomaly-based techniques for misbehavior detection for efficiency reasons especially for resource-constrained IoT devices by avoiding the high cost associated with profiling and learning anomaly patterns as would be required by anomaly-based techniques. We argue that contemporary anomaly-based misbehavior detection methods for IoT-embedded CPSs based on profiling and machine learning through correlation and statistical analysis of a large amount of data or logs for classifying misbehavior [2, 6-7] will not work for IoT-embedded CPSs because of high memory, run time, communication, and computational overhead, considering that many embedded IoT devices are severely resource-constrained. Specification based misbehavior detection provides a viable approach for misbehavior detection of embedded IoT devices because of light resource requirements for checking misbehaviors against specifications.

The novelty of our work is that we pioneer the use of lightweight behavior rule specification-based misbehavior detection for lightweight IoT devices embedded in a CPS with memory, run time, communication, and computational overhead considerations. Our work is novel compared to the existing specification-based intrusion detection techniques (see Section 2 Related Work for details) in the following aspects: (1) design and implementation of a module for automatically modeling and deriving behavior rules from an embedded IoT device's operational profile specifications [16, 20, 23]; (2) design and implementation of a model checking tool to formally verify that the generated behavior rules are correct and cover all the threats (or meet the security requirements) and that the resulting safe and unsafe states are complete and are generated correctly w.r.t. the specified behavior rules; (3) design and implementation of a module for automatically transforming behavior rules into "attack behavior indicators" and then into a state machine for misbehavior detection at runtime; (4) design and implementation of a lightweight runtime collection module for collecting compliance degree data from runtime monitoring of an IoT device based on its derived state machine; and (5) design and implementation of a lightweight statistical analysis module for misbehavior detection based on experimentally collected misbehavior data at runtime.

The rest of the paper is organized as follows. In Section 2, we survey existing work on misbehavior detection of IoT devices and compare as well as contrast our work with existing work. In Section 3, we discuss the system model. In Section 4, we describe our MedIoT design in detail and apply MedIoT to a patient-controlled analgesia (PCA) device embedded in a medical health monitoring system. In Section 5, we compare MedIoT with an existing anomaly detection method in performance outcomes. In Section 6, we conclude the paper and outline future research areas.

†This work was conducted while Jin-Hee Cho was affiliated with US Army Research Laboratory.

CHASE '18, September 26–28, 2018, Washington, DC, USA

ACM ISBN 978-1-4503-5958-0/18/09$15.00
https://doi.org/10.1145/3278576.327860190

2 Related Work

In this section, we provide an overview of related work in three categories: anomaly-based IoT misbehavior detection, specification-based IoT misbehavior detection, and formal verification of behavior specifications in specification-based detection techniques.

2.1 Anomaly-based IoT Misbehavior Detection

Existing misbehavior detection methods for IoT mostly are designed to detect either routing attacks or Denial of Service (DoS) attacks [24]. More recent works [25] also addressed detecting illegal memory accesses in low-power IoTs. These existing works, however, are based on anomaly-based techniques applying profiling and machine learning through correlation and statistical analysis of a large amount of data or logs for classifying misbehavior [2, 6-7, 10-11, 14-15, 25]. Our motivation of this work is that anomaly-based detection techniques will not work for IoT-embedded CPSs because many embedded IoT devices especially battery-operated ones are severely resource-constrained. Our work is based on lightweight specification-based misbehavior detection of each IoT device embedded in a CPS.

2.2 Specification-based IoT Misbehavior Detection

In the literature, specification-based misbehavior detection has been mostly applied to communication networks [4, 8, 21] and CPS security [1, 9, 17, 18, 26]. DaSilva et al. [4] proposed traffic-based rules to detect network intruders. Ioannis et al. [8] devised packet-forwarding behavior rules to detect blackhole and greyhole attacks. Song et al. [21] proposed specification-based detection rules (i.e., identifying monitored activity) to ensure that the global security requirement is complied with an IP configuration protocol in mobile ad networks. In the context of CPS security, Berthier et al. [1] proposed specification-based misbehavior detection to audit the network traffic among smart meters and access points for protocol compliance. Jokar et al. [9] devised specifications to detect MAC layer attacks in smart grids. Mitchell et al. [17, 18] discussed a conceptual model of behavior rule based intrusion detection for CPSs and conducted a proof-of-concept statistical analysis using pre-generated data following an attacker behavior model. Khan et al. [26] proposed behavior-based executable specification against false data injection attacks for industrial control systems. Unlike the above existing works, we pioneer the use of lightweight specification-based misbehavior detection specifically for resource-constrained IoT devices embedded in a CPS.

2.3 Verification of Specification-based Intrusion Detection

While specification-based detection in general induces a lower false positive rate than anomaly detection, a limitation of specification-based approaches is the difficulty to verify if the specifications are correct and cover all the threats [1]. Toward this end, Song et al. [21] describe a formal reasoning framework to first define a global security requirement and then define the specifications of the behaviors of local nodes to assure the global security property. Berthier et al. [1] followed a similar approach and proposed a formal framework comprising a model of the network, monitoring operations, protocol specifications, and a security policy. Utilizing the Applicative Common Lisp (ACL) theorem prover [27], they verify that all network traces that respect the network model, monitoring operations, and protocol specifications will also respect the security policy. Unlike the above cited work [1, 21], we start with the "operational profile" [16, 20, 23] of an embedded IoT that defines the mission statement of the embedded IoT device to derive the security requirements and hence the threats of the embedded IoT device. Then we derive the behavior rules specifying the intended behavior and verify that the behavior rules are correct and cover all the threats. Since ACL2 [27] is a proven tool, we use it to define behavior specifications and security requirements, all expressed as ACL functions, for formal verification. Lastly unlike [1, 21], MedIoT is specifically designed for misbehavior detection of lightweight IoT devices embedded in a Medical Cyber-Physical Systems (MCPS) with energy consideration.

3 System Model

We refer the readers to [12, 13, 24, 25] for attacker behaviors and intrusion detection mechanisms available for IoT-embedded CPSs. Our behavior-rule based IDS approach relies on the use of monitor nodes. We assume that a monitor node performs misbehavior detection on a target node. One possible design is to have a sensor (actuator) monitor another sensor (actuator respectively) within the same CPS. This may require each sensor (actuator) to have multiple sensing functionalities. Another possibility is that each IoT device is built on top of a secure computational space (e.g., [5]) such that each target IoT device can execute misbehavior detection code in the secure computation space and self-monitor itself, even if the operating kernel has been compromised. The monitoring process is lightweight and will not interfere with the normal operations of the monitor IoT device or the target IoT device (see Section 4.3 for detail).

4 MedIoT Design as Applying to PCA in MCPS

In this section, we provide the detail of our MedIoT design and exemplify MedIoT with patient-controlled analgesia (PCA) devices embedded in a health monitoring MCPS [17]. We consider a PCA device that is programmed to perform analgesic injection in response to the injection button being pressed, with the injection period and dosage controlled by authority.

4.1 Behavior Rule Specification of a PCA

We use the design concept of "operational profile" [16, 20, 23] during the testing and debugging phase of an embedded IoT device when the IoT software is built to identify the complete set of behavior rules. An IoT device's operational profile essentially is a mission assignment during the operational phase of the IoT device. A mission assignment in an embedded IoT device's operational profile explicitly defines a set of security requirements for the mission to be successful, from which a set of threats as well as a set of behavior rules to cope with the threats may be automatically derived.

We consider a PCA in a MCPS with the following operational profile:

Raise an alert to designated personnel and halt analgesic injection if the patient's medical condition is unfit for analgesic injection; raise an alert to designated personnel and halt analgesic injection if the PCA is not ready for analgesic injection; communicate with authorized personnel only regarding the injection rate and dosage of medicine; perform correct IDS functions; when the injection button is pressed, if the patient controlled injection rate is less than or equal to the specified injection rate then inject a specified dose of medicine.

Given this operational profile as input, the security requirements of this PCA may be derived as listed in Table 1.

Table 1: PCA Security Requirements.

ID	Security Requirement
SR 1	The PCA must raise alert to designated personnel and hold analgesic injection if the patient's condition is unfit for analgesic injection
SR2	The PCA must raise alert to designated personnel and hold analgesic injection if the PCA is not ready for analgesic injection
SR 3	The PCA must change its injection rate and medicine dosage upon authorized commands only
SR 4	The PCA must perform correct IDS functions when serving as a monitor node, i.e., providing true recommendations
SR 5	The PCA must perform analgesic injection at the specified dosage without exceeding the allowable injection rate

With the system requirements defined, it is relatively straightforward to identify the threats that will keep this PCA from accomplishing its mission, as listed in Table 2.

Table 2: PCA Threats.

ID	Threat
THREAT 1	The PCA is not able to raise alert and hold analgesic injection when patient is unfit
THREAT 2	The PCA is not able to raise alert and hold analgesic injection when PCA is not ready
THREAT 3	The PCA is not able to follow authorized commands
THREAT 4	The PCA is not able to perform correct IDS functions, i.e., not able to provide true IDS recommendations
THREAT 5	The PCA's analgesic injection rate is above the specified injection rate
THREAT 6	The PCA is not injecting the specified dosage

Next, we derive the behavior rule set for this PCA. Table 3 lists the behavior set without priority order for simplicity. It also lists the security aspect (integrity, confidentiality, or availability) associated with each behavior rule. A behavior rule is typically derived from a threat because a threat specifying a negative event that can lead to an undesired outcome is just opposite to a behavior rule specifying a good behavior or a good event that can lead to a desired outcome.

Table 3: PCA Behavior Rules.

ID	Behavior Rule	Security Aspect
BR 1	Raise alert to designated personnel and hold analgesic injection if patient is unfit	Integrity, confidentiality, availability
BR 2	Raise alert to designated personnel and hold analgesic injection if PCA is not ready	Integrity, confidentiality, availability
BR 3	Accept authorized commands	Integrity, confidentiality, availability
BR 4	Provide true recommendations	integrity
BR 5	Perform analgesic injection without exceeding the specified rate	integrity
BR 6	Perform analgesic injection at the specified dosage	integrity

We conduct automatic model verification of the behavior rules generated by verifying if the behavior rules generated are correct and cover all the threats (or satisfy the security requirements). We leverage ACL2 [27], a theorem prover, to define security requirements (in Table 1) as well as behavior rules (in Table 3) as ACL functions. We complete formal verification by defining a theorem (also an ACL function) that is evaluated to be true, proving that this PCA device will not violate the security requirements if it does not violate the behavior rules.

4.2 Transforming the Behavior Rules to a State Machine for Misbehavior Detection

After the behavior rule set is identified, we transform it to a state machine for lightweight misbehavior detection. The behavior-rule-to-state-machine transformation process is automatic. First, one or more "attack behavior indicators" (ABIs) for each behavior rule is identified. Then, each ABI is expressed as a conjunctive normal form (CNF) predicate to be evaluated to true or false indicating whether the corresponding behavior rule is violated or not. Then, all ABIs are combined together into a disjunctive normal form (DNF) predicate. Lastly the state machine is formed with all ABIs as state components, each taking the value of 1 (true) or 0 (false). When all ABIs take the value of 0, it means that none of the behavior rules is violated and hence the system is in a safe state. Conversely, when any ABI takes the value of 1, it means misbehavior because that particular behavior rule is violated. We describe the behavior rule to state machine transformation process in the following subsections.

4.2.1 Attack Behavior Indicators Expressed as CNF Predicates

Table 4 lists 9 ABIs, each to be evaluated to 1 (true) or 0 (false) at runtime through monitoring, indicating whether the corresponding behavior rule is violated or not. When an ABI is evaluated to true, the PCA is detected as misbehaving against the corresponding behavior rule. Identifying an ABI involves identifying a set of physical variables whose runtime values decide if the corresponding behavior rule is violated or not. For example, ABI 1 in Table 4 has two physical variables, namely, *Patient Pulse*

Rate and *Action*. When the patient's pulse is not normal and the action is not alert-and-hold, it is a violation of BR 1.

ABIs 1, 2, and 3 derive from BR 1 as there are three conditions for defining "when patient is unfit," ABIs 4 and 5 derive from BR 2 as there are two conditions for defining "when the PCA is not ready," ABI 6 derives from BR 3, ABI 7 derives from BR 4, and ABI 8 and ABI 9 derive from BR 5 and BR 6 as there are two conditions for defining the PCA not being able to follow authorized commands to correctly perform analgesic injection.

The 1st ABI (ABI 1 in Table 4) is that this PCA does not alert personnel and hold analgesic injection when the patient's pulse rate is not normal. A normal pulse rate for adults is 60-100 beats per minute. The CNF of the Boolean expression is (Patient Pulse Rate ≠ Normal) ∧ (Action ≠ Alert-and-Hold).

The 2nd ABI (ABI 2 in Table 4) is that this PCA still injects analgesic when the patient's respiration rate is not normal. The normal respiratory rate for adults is 12–20 breaths per minute. The CNF of the Boolean expression is (Patient Respiration Rate ≠ Normal) ∧ (Action ≠ Alert-and-Hold).

The 3rd ABI (ABI 3 in Table 4) is that this PCA still injects analgesic when the patient is being treated with defibrillation. The CNF of the Boolean expression is (Patient Status = Defibrillation) ∧ (Action ≠ Alert-and-Hold).

The 4th ABI (ABI 4 in Table 4) is that this PCA does not alert personnel and hold analgesic injection when the drug reservoir is empty. The CNF of the Boolean expression is (Drug Reservoir = Empty) ∧ (Action ≠ Alert-and-Hold).

The 5th ABI (ABI 5 in Table 4) is that this PCA does not alert personnel and hold analgesic injection when the infusion site is incorrect, e.g., the injection is pulled of the patient's body or the injection is not at the patient's correct infusion point. This is indicated by measuring the infusion pressure being normal or not. The CNF of the Boolean expression is (Infusion Pressure ≠ Normal) ∧ (Action ≠ Alert-and-Hold). This ABI has a local variable called *Infusion Pressure* for measuring the infusion pressure to detect if the infusion site is correct. If image sensors are built inside the PCA, image-sensing the infusion site may directly detect if the infusion site is at the right place.

The 6th ABI (ABI 6 in Table 4) is that a PCA does not accept authorized commands to update its injection rate and medicine dosage. The CNF is (Action ≠ Accept) ∧ (Command = AUTHORIZED).

The 7th ABI (ABI 7 in Table 4) is that a monitor PCA provides false recommendations toward a behaving target PCA (called bad-mouthing attacks), and good recommendations toward a misbehaving target PCA (called ballot-stuffing attacks). This may be detected by detecting recommendation discrepancies among multiple monitor PCAs. The CNF is Target Node Audit ≠ Monitor Node Audit.

The 8th ABI (ABI 8 in Table 4) is that this PCA injects analgesic at a rate exceeding the specified injection rate. The CNF is (Injection Rate > Specified Injection Rate) ∧ (Action = Inject).

Finally, the 9th ABI (ABI 9 in Table 4) is that this PCA does not inject analgesic at the right dosage. The CNF is (Dosage ≠ Specified Dosage) ∧ (Action = Inject).

Table 4: PCA Attack Behavior Indicators in CNF.

ID	Attack Behavior Indicator
ABI 1	(Patient Pulse Rate ≠ Normal) ∧ (Action ≠ Alert-and-Hold)
ABI 2	(Patient Respiration Rate ≠ Normal) ∧ (Action ≠ Alert-and-Hold)
ABI 3	(Patient Status = Defibrillation) ∧ (Action ≠ Alert-and-Hold)
ABI 4	(Drug Reservoir = Empty) ∧ (Action ≠ Alert-and-Hold)
ABI 5	(Infusion Pressure ≠ Normal) ∧ (Action ≠ Alert-and-Hold)
ABI 6	(Command = AUTHORIZED) ∧ (Action ≠ Accept)
ABI 7	Target Node Audit ≠ Monitor Node Audit
ABI 8	(Injection Rate > Specified Injection Rate) ∧ (Action = Inject)
ABI 9	(Dosage ≠ Specified Dosage) ∧ (Action = Inject)

4.2.2 All ABIs are Combined into a DNF Predicate

All 9 ABIs in Table 4 are combined together into a DNF predicate (ABI 1 ∨ ABI 2 ∨ ABI 3 ∨ ABI 4 ∨ ABI 5 ∨ ABI 6 ∨ ABI 7 ∨ ABI 8 ∨ ABI 9) because every ABI if evaluated to true is an indication of misbehavior.

4.2.3 Generated State Machine for Misbehavior Detection

For the PCA state machine, there are 9 Boolean variables (each taking the value of either 1 or 0) in the state representation, resulting in the total number of states being 2^9= 512, out of which only one is a safe state (when all 9 Boolean variables are false or take the value of 0) and all other 511 states are unsafe states. Note that there are many variables in these 9 ABIs. However, these variables are internal variables maintained by a monitor PCA who updates these internal variable values at monitoring intervals to determine the true/false (or 1/0) of the 9 Boolean variables for a target PCA that is being monitored on.

4.3 Runtime Collection of Compliance Degree Data

Unlike anomaly detection which frequently requires heavy resources to profile/learn anomaly patterns, our behavior rule specification-based data collection process is lightweight. By using the transformed state machine, a monitor device only needs to periodically monitor if a target IoT device is in safe or unsafe states without interfering with the normal operation of either the monitor device or the target device. For the target PCA, we label its 512 states in the state machine as states 0, 1, 2, …, 511 with state 0 represented by (0, 0, 0, 0, 0, 0, 0, 0, 0) as the only safe state in which all 9 Boolean variables (ABI 1 – ABI 9) take the value of 0 or false. Hence the monitor node can simply collect an instance of the compliance degree of the target node (to be monitored on) by measuring the proportion of time the target PCA node is in state 0. This collection process is repeated periodically. So by the end of the *n*th monitoring periods, the monitor node would collect the compliance degree

history $c_1, c_2, \ldots, c_n$ of the target PCA. As the state machine has incorporated the knowledge of safe vs unsafe states, this data collection process is extremely lightweight. The monitor node just needs to check which states the target node is in during a monitoring interval and measures the proportional of time the target node is in safe states. To save energy, this monitoring process can be done in discrete time space involving probing the states of the target node at discrete time points. Then an instance of the compliance degree can be measured as the ratio of the number of times in which the target node is found to be in safe states over the total number of times the monitor node probes the status of the target node.

4.4 Statistical Analysis for Misbehavior Detection

Our lightweight statistical analysis does not involve training, that is, we do not partition the "compliance degree" history $c_1, c_2, \ldots, c_n$ collected (see Section 4.3) into the training set and the data set for testing because such heavy profiling and learning at runtime is impractical for resource-constrained IoT devices. Rather, we simply model an IoT device's "compliance degree" by a random variable C following a probability distribution function $G(.)$ with the value of 0 indicating zero compliance and 1 indicating perfect compliance. Once we know the target node's compliance degree distribution function, we can compute the expected value of C to know the average compliance degree of the target node over a time period. This information will allow us to decide if the target node is considered "malicious" based on a binary grading criterion, i.e., if the target node's average compliance degree is less than or just equal to a minimum threshold C_T, we consider the target node as malicious.

In this work we consider $G(.) = Beta(\alpha, \beta)$ as the probability distribution function. The α and β parameters can be parameterized using the target node's compliance degree history $c_1, c_2, \ldots, c_n$ collected during runtime. The computation overhead would be manageable because the monitor node just needs to solve the maximum likelihood equations to parameterize the α and β parameters. In this case, the run time complexity is $O(nlogn)$. If we use a one-parameter $Beta(\beta)$ distribution with α fixed at the value of 1 as the probability distribution function, the run time complexity to parameterize β using the target node's compliance degree history is only $O(n)$ which is extremely lightweight compared with contemporary anomaly based detection methods that would incur the run time complexity of $O(n^p)$ to $O(p^n)$, $p > 1$, where n is the number of data samples because of the need to profile or learn anomaly patterns.

The effectiveness of our lightweight statistical analysis method described above can be measured by the false negative probability P_{fn} and false positive probability P_{fp}. During an experimental run if a seeded "good" node's compliance degree is lower than or just equal to the minimum threshold C_T, we incur a false positive, i.e., treating a good node as a bad node. On the other hand, if during an experimental run a seeded "bad" node's compliance degree is higher than the minimum threshold C_T, we incur a false negative, i.e., treating a bad node as a good node. For the target PCA, since we know the target PCA's compliance degree distribution function $G(.)$ after applying our lightweight statistical analysis method, we can easily compute $P_{fn} = \Pr(C > C_T) = 1 - G(C_T)$ given that the PCA device is "bad" and $P_{fp} = \Pr(C \leq C_T) = G(C_T)$ given that the PCA device is "good" during experimental runs.

5 Performance Comparison

We compare MedIoT with a semi-supervised anomaly-based behavior detection method [22] for classifying patient behaviors in a medical health monitoring system. Their design is based on auditing data collected from a series of events involving least common subsequence (LCS) and non-LCS events with event start times and durations. 70% of the data set is used as training data to learn normal event patterns based on similarity and the remaining 30% used as testing data for evaluating performance.

We setup the testing environment with a good target PCA and a malicious target PCA as in Park's experiment. A monitor PCA is also setup to periodically monitor the good target PCA and the malicious target PCA based on the behavior-rule state machine preloaded into the monitor PCA's memory upon bootstrapping. The environment is characterized by an imperfect monitoring probability of 3% due to coding errors, transient faults and environment noises. That is, the monitor PCA may mis-detect the state a target PCA is in with a 3% error probability due to coding errors, transient faults, or environment noises. The malicious target PCA attacks whenever possible based on the PCA threat conditions listed in Table 2. The monitor PCA collects a target PCA's compliance degree history $c_1, c_2, \ldots, c_n$ subject to the imperfect monitoring probability, based on which it computes the true positive rate $(1 - P_{fn})$ versus false positive rate (P_{fp}) by adjusting the minimum compliance threshold C_T as described in Section 4.4.

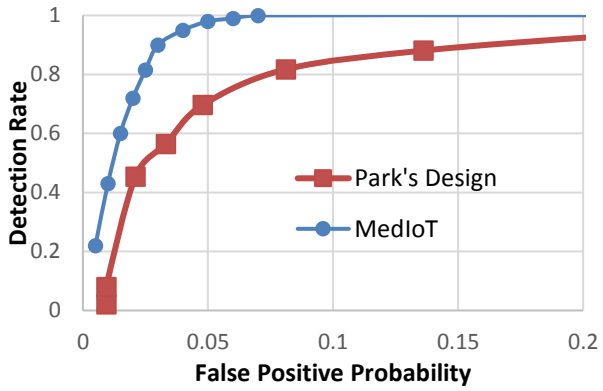

Figure 1: Performance Comparison of MedIoT vs. Park's Design.

We use a Receiver Operating Characteristic (ROC) graph with the detection rate $(1 - P_{fn})$ on the Y coordinate and the false positive rate (P_{fp}) on the X coordinate for performance comparison. The Area Under the ROC curve (AUROC) is especially a well-adopted metric for performance comparison of misbehavior detection methods because it can properly

reflect the tradeoff between false negative rate (P_{fn}) and false positive rate (P_{fp}).

Figure 1 compares the ROC curves for MedIoT and Park's. We can see clearly that the AUROC of MedIoT dominates that of Park's design. In particular, MedIoT always has a better detection rate given the same P_{fp} and always has a better P_{fp} given the same detection rate. By setting the minimum compliance threshold C_T to a high bar at 0.98, MedIoT can achieve a 100% detection rate for detecting the malicious target PCA, while limiting the false positive probability P_{fp} to just 7% for misjudging the good target PCA because of the presence of imperfect monitoring. We attribute the superior performance of MedIoT by the unique design that safe and unsafe-state information is already summarized in the transformed state machine, based on which a monitor node just needs to measure the proportion of time a target node (being monitored on) is in safe states for effective and efficient misbehavior detection.

6 Conclusion

The proposed behavior rule specification-based misbehavior detection technique is generic and can be applied to practical IoT-embedded cyber physical systems for which very lightweight embedded IoT devices (e.g., sensors, actuators, or a combination of both) are an integral part of the overall system design. We illustrated the feasibility of our proposed method with a PCA device embedded in a MCPS where a peer PCA serves the role of a monitor node. We position our behavior rule specification-based misbehavior detection technique as the only feasible solution in terms of low memory, run time, communication, and computation overhead, and high misbehavior detection prediction accuracy to ensure protection of resource-constrained embedded IoT devices against zero-day attacks. This is to be further tested with experimental verification.

ACKNOWLEDGMENTS

This work is partially supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00664, Rule Specification-based Misbehavior Detection for IoT-Embedded Cyber Physical Systems). This work is also supported in part by the U.S. AFOSR under grant number FA2386-17-1-4076.

REFERENCES

[1] R. Berthier and W.H. Sanders. 2011. Specification-based Intrusion Detection for Advanced Metering Infrastructures. *17th IEEE Pacific Rim Int. Symp. Dependable Computing*, 184-193.

[2] A. Bezemskij, G. Loukas, R.J. Anthony, and D. Gan. 2016. Behaviour-based anomaly detection of cyber-physical attacks on a robotic vehicle. *IEEE Symposium on Cyberspace and Security,* 1-8.

[3] I.R. Chen, O. Yilmaz, and I.L. Yen. 2006. Admission control algorithms for revenue optimization with QoS guarantees in mobile wireless networks. *Wireless Personal Communications,* 38 (3), 357-376.

[4] A. DaSilva et al. 2005. Decentralized intrusion detection in wireless sensor networks. *1st ACM inter. workshop on quality of service & security in wireless and mobile networks*., 16–23.

[5] N. Zhang, K. Sun, W. Lou, and Y.T. Hou. 2016. CaSE: Cache-Assisted Secure Execution on ARM Processors. *IEEE Symposium on Security and Privacy*.

[6] J. Hong, C.C. Liu, and M. Govindarasu. 2014. Integrated Anomaly Detection for Cyber Security of the Substations. *IEEE Trans. Smart Grid*, 5 (4), 1643-1653.

[7] S. Huda, et al. 2017. Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data. *Information Sciences*, 379, 211-228.

[8] K. Ioannis, T. Dimitriou, and F. Freiling. 2007. Towards intrusion detection in wireless sensor networks. *13th European Wireless Conference*.

[9] P. Jokar, H. Nicanfar, and V.C.M. Leung. 2011. Specification-based Intrusion Detection for Home Area Networks in Smart Grids. *IEEE Int. Conf. on Smart Grid Communications*.

[10] A.M. Kosek. 2016. Contextual anomaly detection for cyber-physical security in smart grids based on an artificial neural network model. *IEEE Workshop on Cyber-Physical Security and Resilience in Smart Grids*.

[11] C. Kwon, S. Yantek, and I. Hwang. 2016. Real-Time Safety Assessment of Unmanned Aircraft Systems Against Stealthy Cyber Attacks. *Journal of Aerospace Information Systems*, 13 (1), 27-46.

[12] R. Mitchell, and I.R. Chen. 2014. A Survey of Intrusion Detection Techniques in Cyber Physical Systems. *ACM Computing Survey*, 46 (4), article 55.

[13] R. Mitchell and I.R. Chen. 2016. Modeling and Analysis of Attacks and Counter Defense Mechanisms for Cyber Physical Systems. *IEEE Transactions on Reliability*, 65 (1), 350-358.

[14] S. Ntalampiras. 2016. Automatic identification of integrity attacks in cyber-physical systems. *Expert Systems with Applications*, 58, 164-173.

[15] Y. Zhang, L. Wang, W. Sun, R. Green, and M. Alam. 2011. Distributed intrusion detection system in a multi-layer network architecture of smart grids. *IEEE Trans. Smart Grid*, 2 (4), 796–808.

[16] J. Musa. 1993. Operational profiles in software reliability engineering. *IEEE Software*, 14–32.

[17] R. Mitchell and I.R. Chen. 2015. Behavior Rule Specification-based Intrusion Detection for Safety Critical Medical Cyber Physical Systems. *IEEE Transactions on Dependable and Secure Computing*, 12 (1), 16-30.

[18] R. Mitchell and I.R. Chen. 2014. Adaptive Intrusion Detection of Malicious Unmanned Air Vehicles Using Behavior Rule Specifications. *IEEE Transactions on Systems, Man and Cybernetics*, 44 (5), 593-604.

[19] S.T. Cheng, C.M. Chen, and I.R. Chen. 2000. Dynamic quota-based admission control with sub-rating in multimedia servers. *Multimedia Systems*, 8 (2), 83-91.

[20] I.R. Chen and F.B. Bastani. 1991. Effect of Artificial-Intelligence Planning Procedures on System Reliability. *IEEE Trans Reliability*, 40 (3), 364-369.

[21] T. Song, et al. 2006. Formal Reasoning about a Specification-based Intrusion Detection for Dynamic Auto-configuration Protocols in Ad Hoc Networks. *Formal Aspects in Security and Trust,* 16-33.

[22] K. Park, Y. Lin, V. Metsis, Z. Le, and F. Makedon. 2010. Abnormal human behavioral pattern detection in assisted living environments. *3rd ACM Int. Conf. Pervasive Technol. Related Assist. Environments*, 9:1–9:8.

[23] I.R. Chen, B. Gu, S.E. George, and S.T. Cheng. 2005. On failure recoverability of client-server applications in mobile wireless environments. *IEEE Trans. Reliability*, 54 (1), 115-122.

[24] B.B. Zarpelao, R.S. Miani, C.T. Kawakani, and S.C. de Alvarenga. 2017. A Survey of Intrusion Detection in Internet of Things. *Journal of Network and Computer Architecture,* 84, 25-37.

[25] A. Saeed, A. Ahmadinia, A. Javed, and H. Larikani. 2016. Intelligent Intrusion Detection in Low-Power IoTs. *ACM Trans. Internet Technology*, 16 (4), article 27.

[26] M.T. Khan, D. Serpanos, and H. Shrobe. 2017. ARMET: Behavior-based Secure and Resilient Industrial Control Systems. *Proceedings of The IEEE*.

[27] M. Kaufmann and J.S. Moore. 2017. A Computational Logic for Applicative Common Lisp. http://www.cs.utexas.edu/users/moore/acl2/.