Analysis of Attack-Defense Strategies in Autonomous Distributed IoT Systems

Hamid Al-Hamadi
Computer Science
Kuwait University
hamid@cs.ku.edu.kw

Ing-Ray Chen
Dept. of Computer Science
Virginia Tech
irchen@vt.edu

Ding-Chau Wang
Information Management Dept.
Southern Taiwan University of Science and Technology
dcwang@mail.stust.edu.tw

***Abstract*— We develop an analytical model to capture the interplay of attack-defense strategies of an autonomous distributed Internet of Things system (ADIoTS). Every node participates in intrusion detection of a target node of the same type, thus necessitating that every good node plays a set of defense strategies and every bad node plays a set of attack strategies for achieving their own goals. The end product is a methodology for identifying the best defense strategies to maximize the system lifetime.**

Keywords—intrusion detection; attack/defense behavior models; Internet of Things; mission-oriented IoT systems.

1. INTRODUCTION

In this paper, we develop a methodology to capture and analyze the interplay of intrusion detection attack-defense strategies in an autonomous distributed Internet of Things system (ADIoTS). An instance of ADIoTS is a mission-oriented military IoT system populated with autonomous, smart IoT devices including smart sensors, actuators and control nodes, for executing a specific mission. Possible application scenarios may involve a team of unmanned aerial vehicles (UAVs), soldiers, automobiles, or robots monitoring and patrolling a combat area, and relaying critical information to the base for combat advantages. Such IoT devices (called nodes in this paper for short) can be compromised via capture attacks (through physical or cyber space) and turned into insiders performing various malicious attacks with the objective to fail the mission. Thus, an intrusion detection system (IDS) is called for to detect and remove inside attackers in such ADIoTS to ensure successful mission execution.

We design the ADIoTS such that all nodes in the ADIoTS are expected to perform IDS duties. Malicious nodes, however, can choose from a set of attack strategies with the objective to retain malicious nodes (thus causing false negatives) and evict good nodes (thus causing false positives) so as to fail the mission. Good nodes on the other hand choose from a set of defense strategies to prolong the system lifetime. The attack/defense behaviors manifest into the false negative probability (i.e., missing a malicious node as a good node) and false positive probability (i.e., misidentifying a good node as a malicious node) which together affect the system lifetime. Here an attacker refers to an inside attacker and a defender refers to a good node.

While the importance of designing effective IDS strategies for detecting malicious nodes is well recognized, the literature [1-10] is thin in modeling the interplay of attack/defense strategies and their effects on system reliability. Our work follows model-based evaluation. The novelty lies in setting up IDS duties that every node must participate in, thus forcing attack/defense interplay to go in a direction toward the designer's desirable outcome, i.e., prolonging the system lifetime.

Our work has the following unique contributions:

1. We develop a new concept of attack/defense strategies by attackers/defenders while they execute their required IDS functions in the form of voting-based intrusion detection in an ADIoTS. At the *host-level* each node is required to monitor every neighbor node based on preloaded anomaly detection mechanisms to judge if the neighbor node is behaving or misbehaving, At the *system-level,* a group of nodes around a target neighbor node if selected must perform "IDS majority voting" to decide if the target neighbor node is behaving or misbehaving. More specifically, when asked to express its opinion about whether a target node in the neighborhood is behaving, a node must vote "yes" (meaning behaving) or "no" (meaning misbehaving) toward the target node. A malicious node can perform "ballot-stuffing" attacks by voting "yes" toward another malicious node to keep the malicious target node in the system. A malicious node can also perform "bad-mouthing" attacks by voting "no" toward a good node to evict the good target node from the system, especially if doing so does not expose itself as a malicious node. When the majority of votes is "no" the target node is evicted. For the case in which a malicious node is voted "yes" by a majority, the system results in a false negative. For the case in which a good node is voted "no" by a majority, the system results in a false positive. Malicious nodes would apply the "best" attack strategies with the goal to shorten the system lifetime. Good nodes (i.e., defenders) on the other hand would select the "best" defense strategies to prolong the system lifetime. The attack/defense behavior therefore is set up within the context of IDS voting whose effectiveness is measured by the false negative probability and false positive probability which together affect the system lifetime.
2. We develop an analytical model based on Stochastic Petri Net (SPN) modeling techniques [11-23] to describe the dynamics of IDS attack/defense strategies and examine their effect on system lifetime.
3. We develop a novel iterative computational procedure with computational complexity of $O(n)$ where n is the number

TABLE 1: SYSTEM FAILURE TYPES.

System Failure Type	Meaning
Byzantine failure	A Byzantine failure occurs if one third or more IoT devices in the ADIoTS have been compromised as there is no way to reach a consensus for decision making.
Attrition failure	An attrition failure occurs if the ADIoTS does not have enough IoT devices left to carry out its mission.
Resource depletion failure	A resource depletion failure occurs if energy of IoT devices is too depleted to be able to accomplish the mission.

TABLE 2: ATTACK STRATEGIES.

Attack Strategies	
Persistent	Attack with probability 1
Random	Attack with probability P_a to evade detection
Opportunistic	Attack only when it sees bad nodes selected for IDS voting form a majority

TABLE 3: DEFENSE STRATEGIES.

Defense Strategies	
Control the number of voters (m) selected for IDS voting on a suspicious node	Higher m means higher detection strength
Control the detection frequency at which IDS voting is performed, i.e., select the detection interval (T_{IDS})	Smaller T_{IDS} means higher detection frequency

of nodes in an ADIoTS to make it computationally feasible to analyze a large ADIoTS.

The rest of the paper is organized as follows: Section 2 discusses intrusion detection attack-defense strategies. Section 3 develops an analytical model and an iterative computational procedure for quantifying the effect of attack/defense strategies on system lifetime. Section 4 conducts a performance evaluation. Finally, Section 5 summarizes the paper and outlines future work.

2. ATTACK/DEFENSE BEHAVIOR MODELING

2.1 System Failure Types

Table 2 summarizes possible system failure types:

- Byzantine failure [24]: A Byzantine failure occurs if one third or more IoT devices in the ADIoTS have been compromised as there is no way to reach a consensus for decision making.
- Attrition failure: An attrition failure occurs if the ADIoTS does not have enough IoT devices left to carry out its mission.
- Resource depletion failure: A resource depletion failure occurs if energy of IoT devices is too depleted to be able to accomplish the mission.

2.2 Attack Strategies

Table 2 summarizes possible attack strategies used by a malicious node (as an inside attacker) during IDS majority voting:

- Persistent: A malicious node attacks recklessly. When serving as a voter during IDS majority voting, it will always vote "no" to evict a good node (to cause a false positive), and "yes" to retain a bad node (to cause a false negative).
- Random: The attack behavior is the same as a persistent attacker except that a malicious node only attacks randomly with probability p_a (0 to 1) to avoid detection.
- Opportunistic: The attack behavior is the same as a persistent attacker except that a malicious node only attacks opportunistically. That is, when serving as a voter, a malicious node will vote to evict a good node, or to retain a bad node, only if there is a majority of bad nodes among m nodes being selected to perform majority voting.

2.3 Defense Strategies

Table 3 summarizes the defense strategies used by all good nodes (as dictated by the defense system) during IDS majority voting. The defense strength can be controlled by adjusting the following two parameters:

- The number of voters (m) selected from a target node's location for executing IDS majority voting
- The intrusion detection interval (T_{IDS}) to control the detection frequency at which IDS voting is performed.

3. PERFORMANCE MODEL

In this section, we develop a performance model to describe the IDS attack-defense dynamics and analyze the effect of attack/defense strategies executed by attackers/defenders on system lifetime. We also develop an iterative computational procedure to make it computationally feasible for a large ADIoTS consisting of a large number of IoT devices.

A performance model must provide the following two pieces of information to facilitate modeling of attack/defense dynamics:

1. Location: we like to know the probability that node *i* is located in area *l* at time *t*, denoted by $P_{i,l}^{L}(t)$. By inspecting $P_{i,l}^{L}(t)$ and $P_{j,l}^{L}(t)$, we will know if node *i* and node *j* are in the same location at time *t*.
2. Good/Bad/Evicted status: we like to know the probability that node *i* is good, bad, or evicted at time *t*, denoted by $P_i^g(t), P_i^b(t)$and $P_i^e(t)$, respectively, with $P_i^g(t) + P_i^b(t) + P_i^e(t) = 1$. By inspecting $P_i^g(t), P_i^b(t)$and $P_i^e(t)$for node *i*, $P_j^g(t), P_j^b(t)$and $P_j^e(t)$for node *j*, $P_k^g(t), P_k^b(t)$and $P_k^e(t)$ for node *k*, etc. we know the attack/defense strength at time *t*. If a good target node is surrounded by many bad nodes, then there is a high probability that the good target node will be misidentified as a bad node (thus causing a false positive) and a bad target node will be misidentified as a good node (thus causing a false negative).

We use Stochastic Petri Net (SPN) modeling techniques to provide us the above two pieces of information. We utilize a tool called SPNP [11] to define and evaluate SPN node models describing node attack-defense behaviors and status, so as to measure the system performance metrics for performance analysis.

Figure 1 shows the SPN node model for node *i* for modeling the location and status of node *i* over time. It consists of a location subnet (top left) providing the location information of node *I* at time *t*, a timer/energy subnet (top right) providing the energy status of node *i*, and a compromise undetected/detected status subnet (bottom) keeping track of if node *i* has been compromised at time *t* and if the compromise has been detected. These subnets are described in more detail in the following subsections. Each node in the system is separately modeled by a SPN node model. Therefore, there will be many SPN node models in the system (i.e., one for each node), but each can be run and evaluated separately with our hierarchical modeling technique.

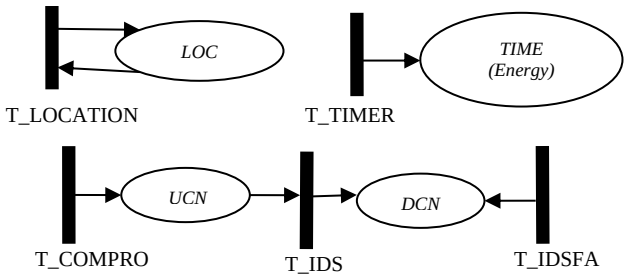

Figure 1: Node SPN Model.

3.1 Modeling Node Status

The location subnet (at the top left of Figure 1) for node *i* provides us information about $P_{i,l}^{L}(t)$. The id of the current location of node *i* is indicated by the number of tokens in place *LOC*. The autonomous distributed IoT environment can be modeled as a *M*×*M* location grid, with the unit length equal to the wireless radio range (*R*) and each location is labeled with a unique location id. We allow each node to have its own mobility pattern specified by a sequence of time-ordered (location id, residence time) tuples, meaning that the IoT device stays at a location with the location id so indicated for this much time with the residence time so indicated. The mobility pattern can be generated by simulating the movement of a node following a mobility model such as the random movement model or the social SWIM mobility model [25]. The transition T_LOCATION is triggered when node *i* moves from its current location to the next location with the transition rate calculated as $1/RT$ where *RT* is the residence time in the current location. Depending on the next location, the number of tokens in place *LOC* is adjusted to reflect the id of the location it resides under (after the movement is made), so by looking at the number of tokens in place *LOC* at time *t* we know the location of node *i* at time *t*.

The compromise undetected/detected status subnet (at the bottom of Figure 1) for node *i* gives us information about $P_i^g(t), P_i^b(t)$ and $P_i^e(t)$. The status of node *i* is indicated by a token which flows from one place to another. Place *UCN* indicates that node *i* is compromised. A node is compromised when transition T_COMPRO with rate λ_{com} fires where λ_{com} is the per-node capture rate. The transition T_COMPRO is enabled if the node is not yet compromised or evicted. When node *i* is compromised, a token goes to *UCN*, meaning that node *i* is now a malicious node not yet detected by IDS, so it may perform persistent, random, or opportunistic attacks. Place *DCN* means that node *i* is evicted. An eviction can occur in two ways. The first way is that node *i* was compromised (i.e., the token was in place *UCN*) and is correctly identified by the system IDS, causing the token to flow from into *DCN* and node *i* to be evicted immediately. The transition rate of T_IDS is $\left(1 - P_{fn}^{IDS}\right)/T_{IDS}$ where P_{fn}^{IDS} (derived in Equation 1 below) is the false negative probability of the system IDS and T_{IDS} is the IDS detection interval. The second way is that node *i* was a good node but is misidentified as a bad node by the system IDS, causing the token to be deposited in place *DCN* and node *i* to be evicted immediately. The transition rate of T_IDSFA is P_{fp}^{IDS}/T_{IDS} where P_{fp}^{IDS}(derived in Equation 1 below) is the false positive probability of the system IDS.

$$P_{fp}^{IDS}(t,l) \text{ or } P_{fn}^{IDS}(t,l) = \sum_{i=0}^{m-m_{maj}} \left[\frac{C\binom{n_{bad}^{a}}{m_{maj}+i} \times C\binom{n_{good} + n_{bad}^{i}}{m-(m_{maj}+i)}}{C\binom{n_{bad}^{a} + n_{bad}^{i} + n_{good}}{m}} \right] + \sum_{i=0}^{m-m_{maj}} \left[\frac{C\binom{n_{bad}^{a}}{i} \times \sum_{j=m_{maj}-i}^{m-i} \left[C\binom{n_{good} + n_{bad}^{i}}{j} \times \omega^{j} \times C\binom{n_{good} + n_{bad}^{i} - j}{m-i-j} \times (1-\omega)^{m-i-j} \right]}{C\binom{n_{bad}^{a} + n_{bad}^{i} + n_{good}}{m}} \right] \quad (1)$$

The timer subnet (at the top right of Figure 1) keeps track of elapsed time in the node SPN model. After T_{IDS} is elapsed, T_TIMER fires and a token is added to place *TIME*. T_TIMER is disabled when the node is evicted (i.e., when a token is in place *DCN*). By looking at the number of tokens in place *TIME*, one can tell the current time. This information allows P_{fp}^{IDS} and P_{fn}^{IDS} to be updated in increment of T_{IDS} dynamically to reflect the effect of IDS attacker/defense dynamics on P_{fp}^{IDS} and P_{fn}^{IDS}. We also use the *timer* subnet as the *energy* subnet with each token deposited in place *TIME* indicating the amount of energy spent by node *i* in an intrusion detection cycle. By knowing the number of IDS cycles elapsed (from place *TIME*) and the percentage of energy spent by node *i* per cycle for executing monitoring, reporting, and performing IDS functions, denoted by P_e, we can estimate the remaining energy of node *i* at time *t*.

3.2 Modeling Attacker/Defender Strategies

An attacker can perform persistent, random, or opportunistic attacks while participating in the majority voting IDS function. The attack strategy chosen affects the system IDS performance measured by the false negative probability (P_{fn}^{IDS}) and the false positive probability (P_{fp}^{IDS}).

We derive the false positive probability ($P_{fp}^{IDS}(t,l)$) and false negative probability ($P_{fn}^{IDS}(t,l)$) for diagnosing a target node at location *l* and time *t* surrounded by $n_{good}(t,l)$ good nodes and $n_{bad}(t,l)$ bad nodes. Henceforth, the notation (t,l) at the end of a symbol is omitted for brevity.

Equation 1 above gives a closed-form solution for P_{fp}^{IDS} and P_{fn}^{IDS} under random attack behavior where $C\binom{a}{b}$ is the # of combinations to select *a* from *b*, n_{bad}^{a} and n_{bad}^{i} are the numbers of "active" and "inactive" bad nodes, given by $n_{bad} \times p_a$ and $n_{bad} \times (1-p_a)$, respectively; m_{maj} is the minimum majority of *m*, e.g., 3 is the minimum majority of 5; and ω is H_{pfp} for calculating P_{fp}^{IDS} and H_{pfn} for calculating P_{fn}^{IDS}. Here H_{pfp} and H_{pfn} are the *host-level* false positive probability and false negative probability, respectively, as a result of each node executing *host-level* IDS duties monitoring behaving or misbehaving of a neighbor node as described earlier. They are given as input at the system start-up time.

Here we note that persistent attack is a special case of random attack with $p_a = 1$. Equation 1 can also be used to model opportunistic attack behavior such that $p_a = 1$ when during IDS voting, more than one half of the nodes selected for IDS voting are bad nodes, thus resulting in $P_{fp}^{IDS} = 1$ and $P_{fn}^{IDS} = 1$. If more than one half of the nodes selected for IDS voting are good nodes, an opportunistic attacker would simply fall back to random attack behavior because there is still a chance good nodes can still vote to evict a good target node (with probability H_{pfp}), or retain a bad target node (with probability H_{pfn}).

3.3 Computational Procedure

The underlying model of a node SPN model as shown in Figure 1 is a continuous-time semi-Markov process with 4 state components, *LOC*, *TIME*, *UCN* and *DCN*, describing the behavior of a node as time progresses.

One could put all node SPN models into one big SPN model and run it in SPNP [11] to yield the system mean time to failure (MTTF) as the performance metric. However, the computational complexity is $O(c^n)$ where $c = 4$ is the number of state components (*LOC*, *TIME*, *UCN* and *DCN*) and n is the number of nodes in the ADIoTS. It is computationally infeasible for a large *n* because of the state explosion problem.

We develop an iterative computational procedure with linear complexity of $O(n)$ to make it computationally feasible for a large ADIoTS. The computational complexity is $O(n)$ because we run each node SPN model one at a time and then integrate their outputs. This computation is performed iteratively until convergence.

The basic idea of our iterative computational procedure is to update the false positive probability $P_{fp}^{IDS}(t)$ and false negative probability $P_{fn}^{IDS}(t)$ iteratively until convergence, as follows:

1) Run each node SPN model for node *i* to completion using SPNP [11] until node *i* is in an absorbing state, i.e., until node *i* is evicted (i.e., a token is in place *DCN*) or until energy is exhausted (i.e., maximum tokens are in place *TIME*). Set $P_{fp}^{IDS}(t)$ and $P_{fn}^{IDS}(t)$ to 5% in the first iteration. Reset them to the new values computed in step 3 in subsequent iterations.

2) For each node SPN model for node *i*, generate the output $P_{i,l}^{L}(t), P_i^{g}(t), P_i^{b}(t)$, and $P_i^{e}(t)$ in increment of T_{IDS}.

3) Compute the false positive probability $P_{fp}^{IDS}(t)$ and false negative probability $P_{fn}^{IDS}(t)$ in increment of T_{IDS} for node *i* based on $P_{i,l}^{L}(t), P_i^{g}(t), P_i^{b}(t)$ and $P_i^{e}(t)$ reported by all nodes in step 2. The time *t* at which the computation is performed can be looked up by inspecting the number of tokens in place *TIME*. Specifically,

$$P_{fp}^{IDS}(t) = \sum\nolimits_{l} P_{i,l}^{L}(t) P_{fp}^{IDS}(t,l) \quad (2)$$

where $P_{fp}^{IDS}(t,l)$ is computed based on Equation 1 with $n_{bad}(t,l) = \sum_{k}^{k\neq i} P_{k,l}^{L}(t) P_k^b(t)$ and $n_{good}(t,l) = \sum_{k}^{k\neq i} P_{k,l}^{L}(t) P_k^g(t)$.

4) Check if the Mean Percentage Difference (MPD) of an important parameter $X_i(t)$ of node i (such as $P_{fn}^{IDS}(t)$) in iteration j and iteration $j+1$ is less than the minimum threshold (set at 1%), i.e., $|X_i^{j+1}(t) - X_i^j(t)|/X_i^j(t) < 1\%$. If no, go to step 1 to continue the iterative computational process. If yes, compute the MTTF of the system based on the failure conditions defined in Table 1 and exit. For *attrition failure*, MTTF can be identified by first sorting the mean time to bad/evicted status for all nodes and then the first time at which the number of good nodes falls below the system allowable minimum threshold (n_{good}^{TH}) is the MTTF. For *Byzantine failure*, the first time at which the number of bad nodes is equal to or greater than 1/3 of the total number of good and bad nodes is the MTTF. For *resource depletion failure*, the first time at which no token is in place *TIME* in the timer subnet is the MTTF.

4. EVALUATION

TABLE 4: PARAMETERS FOR AN ADIOTS.

Parameter	Meaning	Value
n	Number of nodes	128
n_{good}^{TH}	Minimum threshold for attrition failure	32
H_{pfn}	Host IDS false negative probability	5%
H_{pfp}	Host IDS false positive probability	5%
λ_{com}	Per-node capture rate	1/hr
m	Number of voters per IDS voting	3-11
T_{IDS}	IDS interval	0-1400 sec
P_e	Percentage energy spent per T_{IDS}	0.01%
P_a	Random attack probability	[0, 1]
MxM	Operation area	64x64 m^2
R	Radio range	100 m
Mobility	SWIM	Ref [25]

In this section, we conduct an experiment to analyze the effect of the interplay of attack/defense strategies on the MTTF of an ADIoTS characterized by the set of operational and environmental parameter values as listed in Table 4 above.

There are 128 mobile IoT devices randomly deployed in a 64x64 m^2 operational area, each following the SWIM mobility model [25] after deployment. The radio range is 100 m for peer-to-peer communication for the 128 nodes. When there are less than 32 devices in the system, the system is not able to perform its intended function, leading to an attrition failure. At the host level, each device monitors its immediate neighbors with a false negative probability H_{pfn} of 5% and a false positive probability H_{pfp} of 5%. IoT devices are compromised due to capture attacks by which a good device that is being captured is converted into a bad device. The per-node capture rate λ_{com} is 1/hr. Assume that the amount of energy consumed for each IoT device in an IDS period is 0.01%. The performance metric is the system MTTF which is measured when the system fails due to Byzantine, attrition, or energy depletion failure.

Figure 2 shows the system MTTF (s) vs T_{IDS} (s) for the ADIoTS in the case in which the attack strategy is persistent attack (P_a= 1) to quickly fail the system. The defense strategies considered are the number of voters (m) in majority voting IDS and the IDS detection interval (T_{IDS}). With the persistent attack strategy in place, an attacker always performs ballot-stuffing (saying a bad node is a good node) and bad-mouthing attacks (saying a good node is a bad node) whenever it has a chance, so as to cause Byzantine and attrition failures at the fastest pace.

Under this attacker strategy, there exists an optimal T_{IDS} under which the system lifetime is maximized. This is due to the following reasons: When T_{IDS} is too low, the frequency of performing intrusion detection is high, thus causing energy depletion failures to happen early on. When T_{IDS} is too high, it does not perform intrusion detection often enough to detect and remove bad nodes from the system. As a result, many bad nodes remain undetected in the system. This also results in a short lifetime, due to both Byzantine failure (when at least one third of the nodes are bad nodes) and attrition failure (when the number of good nodes falls below n_{good}^{TH}).

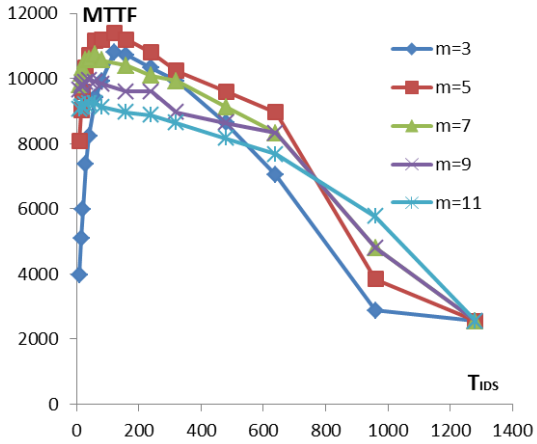

Figure 2: Optimal Defense Strategies for Maximizing the MTTF of an ADIoTS as defined by Table 4.

The effect of the number of voters (m) is clearly demonstrated in Figure 2. We observe that the optimal T_{IDS} depends on m and m = 5 is the best choice of this defense strategy for maximizing the system lifetime for the example ADIoTS. The reason is as follows: When m is high, it tends to deplete energy early on thus causing resource depletion failure. When m is low, it tends to leave too many bad nodes undetected in the system, thus causing Byzantine or attrition failure. Consequently, m = 5 can best balance resource depletion failure versus Byzantine or attrition failure to maximize the system lifetime.

The most striking observation is that an optimal defense strategy exists in terms of the best (T_{IDS}, m) combination that will maximize the system MTTF, given knowledge about the attacker strategy. Figure 2 is for the case in which the attacker strategy is persistent attack. While not reported here due to lack of space, we also observe such best (T_{IDS}, m) combination exists for other types of attacker strategies (random and opportunistic attacks).

5. CONCLUSION

In this work, we developed IDS duties that must be executed by every node of an autonomous distributed IoT system (ADIoTS) with the objective to maximize the system MTTF. We developed SPN-based behavior models as well as a scalable iterative computational procedure with linear complexity in the number of nodes, allowing IDS attack/defense strategies for executing voting-based IDS functions to be specified and analyzed. We demonstrated the applicability with a selected set of attack-defense strategies and identified optimal defense settings in terms of the best (T_{IDS}, m) combination under which the ADIoTS lifetime is maximized.

While we identify the best (T_{IDS}, m) combination that will maximize the system MTTF, given knowledge about the attacker strategy, we fully understand that the attacker strategy is likely to be deceptive and dynamically changed. In the future, we plan to extend this work to a game of mechanism design, allowing IDS attack/defense strategies to be dynamically selected by attackers/defenders to maximize their payoffs, and as a result of the interplay the defense system is able to maximize the MTTF of the ADIoTS as the outcome.

Acknowledgements

This work was partially supported and funded by Kuwait University Research Grant #QS01/18. This work is also supported in part by the U.S. AFOSR under grant number FA2386-17-1-4076.

REFERENCES

[1] A. B. Sharma, F. Ivancic, A. Niculescu-Mizil, H. Chen, and G. Jiang, "Modeling and Analytics for Cyber-Physical Systems in the Age of Big Data," *ACM Sigmetrics*, 2013.

[2] E. Benkhelifa, T. Welsh, and W. Hamouda, "A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Towards Universal and Resilient Systems," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, 2018, pp. 3496-3509.

[3] I. You, K. Yim, V. Sharma, I.R. Chen, and J.H. Cho, "On IoT Misbehavior Detection in Cyber Physical Systems," *The 23rd IEEE Pacific Rim International Symposium on Dependable Computing* (PRDC 2018), Dec 2018.

[4] I. You, K. Yim, V. Sharma, I.R. Chen, and J.H. Cho, "Misbehavior Detection of Embedded IoT Devices in Medical Cyber Physical Systems," *3rd ACM Chase Workshop on Security, Privacy, and Trustworthiness in Medical Cyber-Physical Systems*, Washington DC, Sept 2018.

[5] R. Mitchell and I.R. Chen, "Modeling and Analysis of Attacks and Counter Defense Mechanisms for Cyber Physical Systems," *IEEE Trans. Reliability*, vol. 65, no. 1, 2016, pp. 350-358.

[6] R. Mitchell and I.R. Chen, "Effect of intrusion detection and response on reliability of cyber physical systems," *IEEE Trans. Reliability*, Vol. 62, No. 1, 2013, pp. 199-210.

[7] H. Al-Hamadi and I.R. Chen, "Trust-Based Decision Making for Health IoT Systems," *IEEE Internet of Things Journal*, vol. 4, no. 5, Oct. 2017, pp. 1408-1419.

[8] R. Mitchell and I.R. Chen, "Behavior Rule Specification-based Intrusion Detection for Safety Critical Medical Cyber Physical Systems," *IEEE Trans. Dependable and Secure Computing*, vol. 12, no. 1, 2015, pp. 16-30.

[9] H. Al-Hamadi and I.R. Chen, "Adaptive Network Management for Countering Smart Attack and Selective Capture in Wireless Sensor Networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, 2015, pp. 451-466.

[10] Jin-Hee Cho and Ing-Ray Chen, "PROVEST: Provenance-based Trust Model for Delay Tolerant Networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 1, 2018, pp. 151-165.

[11] G. Ciardo, R.M. Fricks, J.K. Muppala and K.S. Trivedi, *Stochastic Petri Net Package (SPNP)*, Dept. Electrical Engineering, Duke University, 1999.

[12] I.R. Chen and D.C. Wang, "Analyzing dynamic voting using Petri nets," *15th Symposium on Reliable Distributed Systems*, 1996, pp. 44-53.

[13] B. Gu and I. R. Chen, "Performance analysis of location-aware mobile service proxies for reducing network cost in personal communication systems," *Mobile Networks and Applications*, vol. 10, no. 4, 2005, pp. 453-463.

[14] I.R. Chen, T.M. Chen, and C. Lee, "Performance evaluation of forwarding strategies for location management in mobile networks," *The Computer Journal*, vol. 41, no. 4, 1998, pp. 243-253.

[15] I. R. Chen, T.M. Chen, and C. Lee, "Agent-based forwarding strategies for reducing location management cost in mobile networks," *Mobile Networks and Applications*, vol. 6, no. 2, 2001, pp. 105-115.

[16] I.R. Chen, B. Gu, S.E. George, and S.T. Cheng, "On failure recoverability of client-server applications in mobile wireless environments," *IEEE Trans. Reliability*, vol. 54, no. 1, 2005, pp. 115-122.

[17] I.R. Chen and D.C. Wang, "Analysis of Replicated Data with Repair Dependency," *The Computer Journal*, vol. 39, no. 9, 1996, pp. 767-779.

[18] I.R. Chen and F.B. Bastani, "Effect of Artificial-Intelligence Planning Procedures on System Reliability," *IEEE Trans Reliability*, vol. 40, no. 3, pp. 364-369, 1991.

[19] F.B. Bastani, I.R. Chen, and T. Tsao, "Reliability of Systems with Fuzzy-Failure Criterion," *Annual Reliability and Maintainability Symposium*, 1994, pp. 442-448.

[20] I.R. Chen, O. Yilmaz, and I.L. Yen, "Admission control algorithms for revenue optimization with QoS guarantees in mobile wireless networks," *Wireless Personal Communications*, vol. 38, no. 3, 2006, pp. 357-376.

[21] S.T. Cheng, C.M. Chen, and I.R. Chen, "Dynamic quota-based admission control with sub-rating in multimedia servers," *Multimedia Systems*, vol. 8, no. 2, 2000, pp. 83-91.

[22] S.T. Cheng, C.M. Chen, and I.R. Chen, "Performance evaluation of an admission control algorithm: dynamic threshold with negotiations," *Performance Evaluation*, vol. 52, no. 1, 2003, pp. 1-13.

[23] O. Yilmaz and I.R. Chen, "Utilizing Call Admission Control for Pricing Optimization of Multiple Service Classes in Wireless Cellular Networks," *Computer Communications*, vol. 32, no. 2, 2009, pp. 317-323.

[24] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, 1982, pp. 382-401.

[25] S. Kosta, A. Mei, and J. Stefa, "Large-Scale Synthetic Social Mobile Networks with SWIM," *IEEE Trans. Mobile Computing*, vol. 13, no. 1, pp. 116-129, 2014.