IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SECURITY AND PRIVACY IN APPLICATIONS AND SERVICES FOR FUTURE INTERNET OF THINGS

Received March 10, 2017, accepted April 11, 2017, date of publication April 24, 2017, date of current version May 17, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2696031

IoTRiskAnalyzer: A Probabilistic Model Checking Based Framework for Formal Risk Analytics of the Internet of Things

MUJAHID MOHSIN, MUHAMMAD USAMA SARDAR, OSMAN HASAN (Senior Member, IEEE), AND ZAHID ANWAR

School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan

Corresponding author: Mujahid Mohsin (mujahid.mohsin@seecs.nust.edu.pk)

ABSTRACT The Internet of Things (IoT) is being deployed for a plethora of use-case scenarios. In any deployment, a number of configuration choices are available that achieve the mission goal. However, IoT security incidents have demonstrated that different configurations are vulnerable to varied risk levels. We propose the *IoTRiskAnalyzer* framework to formally and quantitatively analyze these risks using probabilistic model checking. *IoTRiskAnalyzer* takes vulnerability scores, candidate IoT configurations, and attacker's capabilities as inputs. It then generates the system and threat models to compute attack likelihood and attacker cost for each configuration. Evaluation indicates that *IoTRiskAnalyzer* is efficient and automatically prioritizes the input configurations on the basis of risk exposure.

INDEX TERMS IoT risk analytic, formal risk modeling, probabilistic model checking, Markov decision process, threat assessment, secure configuration planning, PRISM model checker.

I. INTRODUCTION

The recent years have witnessed an explosive proliferation of the Internet of Things (IoT); thanks to its ever-rising acceptability and express-pace advancements in its enabling technologies. The future IoT prospects are also being nurtured by an overwhelming growth predictions, estimating more than 50 Billion connected 'things' [1], [2], generating $ 7.1 Trillions of market revenues [3] and producing 44 Zettabytes of data [4], by the year 2020.

This exponential IoT growth has also opened new doors for the attackers to conduct malicious activities, as unprofessionally-configured and poorly-protected IoT systems can facilitate their nefarious goals. Gartner, Inc. [5] predicts that by the year 2020, more than 25 percent of the cyber attacks in enterprises will involve IoT and many of the recent security breach incidents, research studies and practical demonstrations endorse this claim. For example, in a proof-of-concept attack analysis of leading smart home products [6], the cyber-security researchers demonstrated that their framework design can be exploited to create spare door keys, steal existing keys and force fire alarms to go off. A number of recently unfolded Distributed Denial of Service (DDoS) attack incidents [7] were ranked amongst the largest known attacks till date, as they recruited millions of connected things as 'thing-bots'. Other studies highlighted vulnerabilities in the Zigbee [8] and Z-Wave [9] implementations (two of the most popular IoT protocols), capable of inducing rapidly spreading IoT worms and causing other digital and physical security threats.

The modern day IoT systems are "*riddled with vulnerabilities and there are no good ways to patch them*" due to a number of practical reasons, as highlighted by Bruce Schneier [10]. Moreover, some vulnerabilities either cannot be completely countered (such as wireless jamming) or require dedicated security controls with specialized management skill set and prohibitive budget overheads (such as firewalls). These limitations have led the IoT planners and consumers to live with such vulnerabilities for long periods, even stretching up the useful life of these systems.

It is noteworthy that the risk of exploiting IoT vulnerabilities largely depends on system configurations. Thus, an optimal configuration can be used to significantly reduce the likelihood of system-level attacks. An IoT architect has several configuration choices in terms of device and technology selection, connectivity and redundancy, each serving the same mission goal but with varied risk levels. Therefore, it is

important to comprehend, quantify and analyze such risks for subsequent configuration optimizations from a security viewpoint.

System-level risk analysis mainly relies on likelihood of exploitation of a given set of vulnerabilities, computed on the basis of well-established risk analysis models, and is, therefore, probabilistic in nature. Moreover, in a hostile IoT scenario, environmental events and attacker behavior (such as her preferences and sequence of exploiting vulnerabilities) are mostly unpredictable, thus making pure deterministic analysis unrealistic. On the other hand, simulation-driven IoT risk assessment approaches [11], [12] cannot exhaustively quantify and analyze the risk exposure scores for complex and safety-critical IoT systems, owing to the incomplete coverage of all possible input vectors. Citing these requirements in this work, we present a novel probabilistic model checking [13] based framework called *IoTRiskAnalyzer*, developed using a Markov Decision Process (MDP) [14] model. A formal model-driven verification approach, as utilized by *IoTRiskAnalyzer*, offers a powerful means to verify all possible behaviors of reference model using a finite state space and thus, can precisely assess the cause and degree of security risks. The MDP models generated by our framework are used in the PRISM model checker [15] to automatically analyze the system-level risk profiles. PRISM, an MDP-supported model checker, has already been extensively employed to verify the security and safety aspects of a wide variety of systems, including aerodynamics [16], smart grid [17] and secure product design [18].

Our earlier work, *IoTSAT* [19], utilized Satisfiability Modulo Theories (SMT) [20] to formally model and analyze the threat resiliency and tactics for generic IoT systems. SMT solvers primarily follow a constraint satisfaction approach and therefore, cannot precisely capture the probabilistic nature of risk-assessment and temporal behaviors of threat verification, as demonstrated by *IoTRiskAnalyzer*. On the other hand, the SMT models, being highly expressive due to the rich set of supported theories, can be used to accurately model the low-level details of complex IoT systems. Consequently, the formalisms adopted by *IoTSAT* and *IoTRiskAnalyzer* are complimentary in nature and both the frameworks can be used in conjunction to plan, verify and develop a holistic security picture for complex IoT systems.

A. PAPER CONTRIBUTIONS

The key contribution of this work is to present a framework, i.e. *IoTRiskAnalyzer*, to *realistically model and formally verify the risk exposure to complex IoT Systems*. As explained further in Section IV, *IoTRiskAnalyzer* takes as input: (1) a set of software, hardware, data and communication vulnerability scores from relevant IoT security literature, (2) a set of candidate IoT configurations for achieving a mission and (3) the attacker behavior and capabilities. *For each candidate configuration, the framework generates the system and threat models, which are utilized to formally compute the likelihood and attacker cost for exploiting individual vulnerabilities to achieve the system-level attack objectives.* The framework thus produces an ordered set of configurations, prioritized on the basis of risk exposure probabilities to different system-level attack scenarios.

B. PAPER ORGANIZATION

The rest of the paper is organized as follows: Section II covers the existing work in the relevant domains. In Section III, we present a background of probabilistic model checking and the PRISM tool. Section IV gives an overview of the research approach used by *IoTRiskAnalyzer*. Section V presents a small-scale case study of a home security system, which is used to explain the working of our framework. Lastly, Section VI explains the implementation and evaluation aspects and Section VII concludes the paper while identifying some directions for the future work.

II. RELATED WORK

To the best of our knowledge, this is the first formal framework to probabilistically quantify the risk of exposures to complex system-level attacks on IoT systems, as a function of individual device-level vulnerabilities and attacker behavior. Nevertheless, our framework leverages existing efforts towards IoT-specific risk analysis and the use of formal techniques for analyzing the security and consistency of the IoT systems. This section critically analyzes the existing efforts in the associated fields.

A. IoT-SPECIFIC RISK ANALYSIS

With the rise in IoT-specific security breach incidents, the field of risk assessment and management for IoT related threats has also emerged as a dedicated research area. Liu *et al.* [21] proposed a dynamic risk assessment methodology for the IoT, inspired by the artificial immune system. Their approach computed the changing risk value of an IoT system based on attack intensity as measured by different attack detection agents. Roman *et al.* [22] discussed the security risks being contributed by the ever-increasing influx of IoT devices. The authors critically analyzed such emerging risks, their root causes, and viable mitigation techniques. Podgórski *et al.* [23], besides presenting a comprehensive literature review of related fields, also proposed a conceptual framework for risk management in the domain of occupation safety and health under smart working environments. Djemame *et al.* [24] presented an implementation of a risk assessment framework for cloud service eco-systems with capabilities to identify, evaluate and mitigate the risks. Their key contribution is the risk assessment model, comprising of four risk categories, namely technical, policy, legal and general. The literature, discussed above, dictates that the IoT systems follow a dynamic and formidable risk posture, which can be tackled using diverse approaches. However, none of these efforts employed a formal model checking based verification approach for risk analysis.

A questionnaire-driven empirical study is another way of quantifying the security risks. Chang *et al.* [25] utilized this

approach to investigate enterprise risk factors for governing the risk of IoT environments. Other works [26], [27] conducted empirical risk analyses for smart home automation systems. Still, other methodologies employed scenario [27], [28] and product-based [29] approaches for characterizing risks. These efforts focus on risk analysis of individual devices or general category of devices and their findings are based on expert opinions, experiences or domain-specific security incidents. Use of formal methods for automated risk assessment, as presented by *IoTRiskAnalyzer*, can leverage and extend such manual methods to automatically reason about risk applicability and countermeasures, not only on individual IoT entities but also for complex and large-scale IoT systems.

B. FORMAL APPROACHES TOWARDS IoT SECURITY

Corno and Sanaullah [30] surveyed and critically analyzed the research contributions towards design-time formal verification for smart environments. The authors categorized the surveyed papers based on various factors and formalisms and concluded that *"no surveyed technique maintains a holistic [modeling] perspective"*. Another indirect inference from their work is that the existing efforts mostly focus on formally verifying the correctness and stability of entity interactions and controls and a very limited literature exists towards the use of formal methods for security analytic and risk verification of such systems.

Mundhenk *et al.* [31] proposed system-level security analysis of smart automotive architectures using a Continuous-Time Markov Chain (CTMC) [32] model. Their approach was focused at design-time system verification, with a premise that specific vulnerabilities are not known a priori. Moreover, CTMC being purely stochastic, cannot model the non-deterministic behavior of attacker, as demonstrated by our work. A model checking approach was used by Kang *et al.* [33] for performing security property verification of a water treatment system. The scope of their research did not cover risk verification and concentrated on exploring a particular system, and is not generalizable. Furthermore, their work examined malicious alteration of sensing and actuation data only, and did not cater for the networking aspects and associated attack patterns, as demonstrated by our framework. Prior works in this area encompass static analysis of security considerations in Computer Supported Cooperative Work (CSCW) systems [34] and a formal analysis technique to uncover stealthy attacks in cyber-physical systems, while utilizing the verification engine of Matlab/Simulink [35].

C. FORMAL APPROACHES TOWARDS IoT CONSISTENCY

Use of formal techniques for proving the consistency of IoT systems is another popular research area. Guilly *et al.* [36] extended the Event Condition Action (ECA) language in Timed Automata for isolating system anomalies and safety hazards in smart home applications. Augusto and Hornos [37] leveraged Linear Temporal Logic (LTL) to model and verify the behavior of smart systems. Corno and Sanaullah [38] proposed an approach to formally verify the correctness, reliability and safety of smart environments at the design stage, using model checking. Coronato and Pietro [39] extended Ambient Logic and Ambient Calculous to formally specify the requirements and verify their correctness in safety-critical ubiquitous and pervasive systems.

The research contributions discussed above aim to verify the anticipated behavior of targeted systems under non-malicious situations. Therefore, these efforts do not cover the threat analytics and risk verification for active adversarial attacks. Contrarily, the system model generated by *IoTRiskAnalyzer* is based on an assumption that the input IoT configurations are consistent and stable under non-malicious scenarios and can only be compromised by an active adversary through the exploitation of components' vulnerabilities, as formalized by the threat model.

III. BACKGROUND

A. PROBABILISTIC MODEL CHECKING

Model checking [40] is a well-recognized and widely-adopted formal technique to verify functional, safety, security and reliability requirements in a number of application domains. The key idea is to model the system as a state transition and express the desired system properties as formulas in temporal logic. The main benefit of model checking is the automatic verification of the properties of interest. In case a property does not hold for a given model of the system, it also provides a counter trace for debugging.

Probabilistic model checking [13] is an advanced model checking technique in which uncertainties and randomized behaviors of stochastic systems are modeled by assigning probability values to the transitions in the state transition model of the targetted system. The behavior of probabilistic systems can be modeled as discrete-time Markov chains (DTMCs), CTMCs [32] or MDPs [14]. DTMC and CTMC are used for modeling the systems where the events are discrete and continuous, respectively, with respect to time. MDP is used to model the non-deterministic behavior of the systems.

Since the attacker behavior and the IoT environmental events are non-deterministic in nature, we have modeled our system as an MDP. Each transition in MDP from the current state to the next state is probabilistic and depends on the current state of the system. Mathematically, the probability of the transition from a current state S to a next state S' is expressed as [41]:

$$P_a(S, S') = P_r(S_{t+1} = S' | S_t = S, a_t = a)$$

where P_r represents the probability to transition from state S to S' and a denotes the corresponding action, which triggered that transition. The transition probabilities of all the state transitions is represented by a Transition Probability Matrix P. The probability of the next state is then expressed as [41]:

$$P_r(S') = P_r(S) * P$$

A variety of probabilistic model checkers, such as MRMC [42], Vesta [43], Ymer [44], PRISM [15] and ETMCC [45], are available to formally model and verify those systems, which exhibit random or probabilistic behaviors. We choose the PRISM model checker because of its support for MDP models (not all probabilistic model checkers support MDP) and time and memory efficiency as compared to the other available tools [46].

B. PRISM MODEL CHECKER

PRISM facilitates several categories of probabilistic models, such as DTMCs, CTMCs [32], MDPs [14], probabilistic automata (PAs) [47], probabilistic timed automata (PTAs) [48] as well as extensions of these models with rewards (or costs), referred to as continuous or discrete-time Markov reward models and priced PTAs. The system models are formally encoded in the *PRISM language*, which is a state-driven language, founded on Alur's Reactive Modules formalism [49]. The PRISM language primarily consists of modules and variables. The encoded model comprises of a concurrent composition of independent yet interacting modules. A module consists of local variables and guarded commands. At any given time, the values assigned to these variables represent the state of the modules and the guarded commands mimic the behavior of these modules. The local state of independent modules is integrated to determine the holistic state of the entire model. The syntax of a PRISM command is as follows:

```
[act] Guard ⇒ Pr_1 : Update_1
              + ...
              + Pr_n : Update_n;
```

where `act` refers to an optional synchronization label and the `Guard` represents a predicate, which can be defined using all the variables contained in the model (inclusive of variables from other modules). The `Update` expression assigns new values to the module variables and `Pr` represents a probability (or rate) value assigned to the corresponding transition taken by the module, after the `Guard` condition is met. Non-determinism can be modeled by using the same `Guard` for multiple PRISM commands, so that all of them are enabled at the same time (whenever their common pre-condition is met) and only one of them is non-deterministically selected.

In order to verify and analyze the behavior of a given system, the desired functionality has to be expressed as a property in a suitable probabilistic logic using *property specification language*. The property specification language used by PRISM is founded on temporal logic and subsumes LTL, PCTL* and PCTL [50] for MDPs. The `Pmin` and `Pmax` operators are used to reason about the minimum and maximum probabilities over all possible resolutions of non-determinism. They can be used to verify quantitative properties that take the form:

```
Pmax=? [ Path-Prop ]
```

where `Path-Prop` is a path property using temporal operators `X` (next state), `U` (until), `F` (eventually), `G` (globally), `W` (weak-until), `R` (release) and their complex combinations. The property mentioned above represents the maximum probability that `Path-Prop` is satisfied by the paths from the current state, for all possible resolutions of nondeterminism [14]. The minimum probability is computed similarly, by using the PRISM keyword `Pmin`.

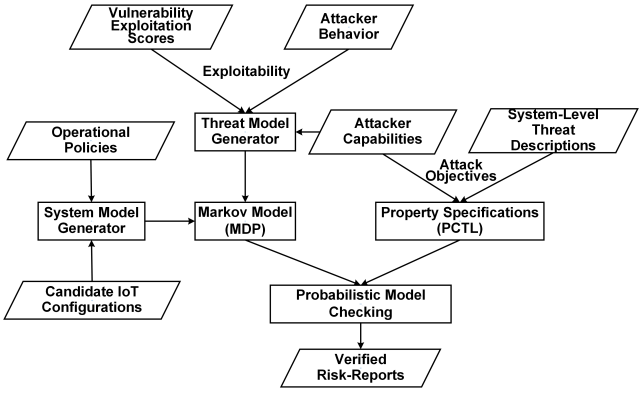

FIGURE 1. *IoTRiskAnalyzer.*

IV. *IoTRiskAnalyzer*: RESEARCH APPROACH

The proposed approach, followed by *IoTRiskAnalyzer*, is depicted in Figure 1. Firstly, a Markov model is developed based on the *system* and *threat* models. The *system model* captures the behavior of: (i) A set of candidate IoT configurations, defining IoT entities and their network, functional and environmental coupling requirements. (ii) *Operational policies* (user-defined), depicting the rules based on which the sensing data is processed and actuation commands are triggered. The *threat model* formally defines the non-deterministic nature of attacker (i.e. *attacker behavior*) while exploiting a chain of vulnerabilities to achieve the attack objectives. This behavior is enriched by the *attacker capabilities* and *vulnerability exploitation scores* of individual IoT components. These vulnerability scores can be extracted from well-established and widely accepted risk assessment models, which quantify the exploitation probabilities for such vulnerabilities.

After the generation of the Markov model in *IoTRiskAnalyzer*, the appropriate properties for risk verification are specified. These properties are developed using Probabilistic Computation Tree Logic (PCTL), after translating the identified system-level threats into *attack objectives* and punctuating them with realistic attacker capabilities. These properties are then checked against the Markov model to extract verified risk reports using the principles of probabilistic model checking. The reports can be analyzed to answer different questions such as: "*What is the maximum likelihood of an attack within the defined attacker's capabilities? Which configuration offers the maximum protection against a range of threats? or which policy suits best the input configurations from a security viewpoint?*"

A. SYSTEM MODEL

The proposed framework generates a dedicated system model for each IoT configuration, where every candidate configuration uniquely defines the network, policy, functional and environmental relationships of registered IoT entities.

Mathematically, an IoT system configuration is defined as a quadruple $< F, N, L, P >$ where:

- F Environmental Features;
- N Nodes
  - $N = \{(H \cup G) \wedge (H \cap G = \varnothing)\}$ where;
  - $G =$ Network devices (gateways, routers) and
  - $H = \{S \cup C \cup D \cup M\}$ are Hosts, where;
  - $S =$ Sensors
  - $C =$ Controllers
  - $D =$ Cloud Servers (Aggregators)
  - $M =$ Actors $\mid M = \{(A \cup R) \wedge (A \cap R = \varnothing)\}$ where;
  - $A =$ Controllable Actuating Devices (Actuators)
  - $R =$ Response Actors
- L Links $\mid L \subseteq N \times N$;
- P A set of operational policies where each policy is a set of rules;

and a set of entity mappings, including;

Network Mapping (NM): Defines the link to device mappings to form a connected graph, comprising of *Nodes* ($b_i \in N$) and connected by *Links* ($l_i \in L$) as edges.

$$\underset{b_i \in N}{NM(b_i) : b_i \Rightarrow \mathbb{M}(L)}$$

Policy Mapping (PM): Defines a one-to-one mapping of *Controllers (C)* and *Response Actors (R)* to the corresponding policy set $p_i \in P$ hosted by them.

$$\underset{b_i \in C \cup R}{PM(b_i) : b_i \Rightarrow \mathbb{M}(P)}$$

Functional Mapping (FM): Defines the functional relationships among the IoT devices in accordance with the corresponding policy defined by PM.

$$\underset{b_i \in H}{FM(b_i) : b_i \Rightarrow \mathbb{M}(C \cup D)}$$

Environment Mapping (EM): Defines the relationship of *Sensors* and *Controllable Actuating Devices* (later referred as actuators) to the corresponding environmental feature $f_i \in F$, which they observe or impact, respectively.

$$\underset{b_i \in S \cup A}{EM(b_i) : b_i \Rightarrow \mathbb{M}(F)}$$

B. THREAT MODEL

For a given scenario, the techniques followed by an attacker to achieve her objectives rely on: (i) IoT vulnerabilities and their exploitation probabilities, (ii) attacker capabilities and priorities and (iii) the IoT system and policy configurations defined by the *system model*. Our threat model is built upon the threat classifications and relations, formally defined in our earlier work [19]. This paper classifies IoT threats as context (sensing), trigger (controlling) and actuation threats and then formally relates them as an interconnected threat propagation tree. The reference threat model [19] is based on the fact that the attacker can meet her end objectives in multiple ways by exploiting the vulnerable assets in conjunction with the intrinsic IoT couplings. However, for meaningful risk analysis, *IoTRiskAnalyzer* extends this threat model by evaluating the pre-defined threat relationships in the light of (i) component vulnerabilities, (ii) their exploitation probabilities and (iii) a non-deterministic attacker behavior with finite capabilities. Our threat model is, therefore, composed of the following building blocks.

1) VULNERABILITY SET (V) AND MAPPINGS (VM)

The threat model receives a finite set of vulnerabilities $V(|V| = Z)$ and a Vulnerability Mapping (VM) function, relating the IoT entities with the corresponding vulnerabilities, hosted by them.

$$\underset{b_i \in N \cup L}{VM(b_i) : b_i \Rightarrow \mathbb{M}(V)}$$

2) VULNERABILITY EXPLOITATION SCORES (E)

Each vulnerability ($v_i \in V$) is assigned a normalized value of Vulnerability Exploitation Score ($e_i \in E \mid 0 < e_i \leq 1$). The data regarding these scores is obtained from the widely acknowledged risk assessment studies, which takes into account several metrics, such as how similar vulnerabilities are exploited in the past, type and difficulty of access required, strengths of respective security barriers (encryption, authentication, etc.) required to be compromised and availability of attack code and tools needed for exploitation.

3) VULNERABILITY TO THREAT MAPPING (TM)

Each vulnerability induces a specific type of Threat(s) (set T). We use the threat definitions from our previous work [19] and align them to the corresponding set of vulnerabilities (V), using the mapping function TM.

$$\underset{b_i \in V}{TM(b_i) : b_i \Rightarrow \mathbb{M}(T)}$$

4) ATTACKER CAPABILITY (Cap)

We assume a non-global adversary with finite capabilities. The attacker is mainly interested in active attacks (such as modification, fabrication and disruption), which she can achieve by compromising only a limited and pre-defined set of IoT entities, through the exploitation of their respective vulnerabilities. The permissible threat vectors strictly follow the vulnerability definitions and vulnerability to threat mappings (TM). The term 'attacker capability (Cap)' is defined as the maximum number of vulnerabilities, which can be exploited by an attacker ($0 < Cap \leq Z$). Our framework supports assigning a constant Cap value or can even evaluate system security against a sweeping value of Cap, within the defined limits.

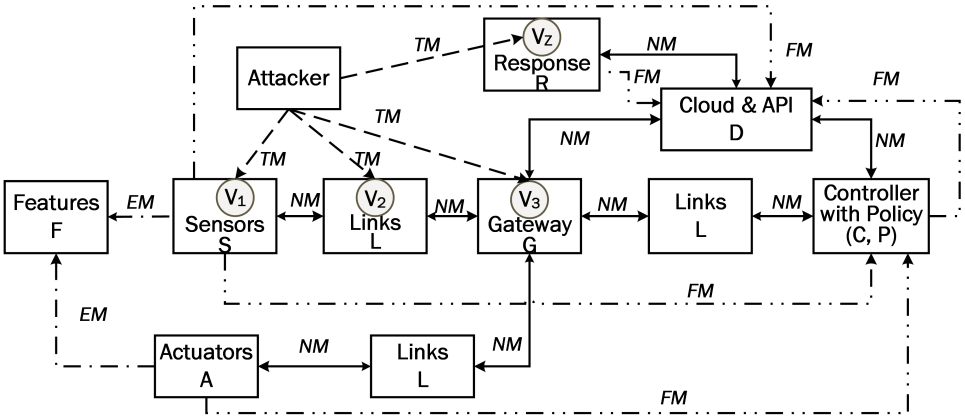

FIGURE 2. An overview of the MDP modules.

5) ATTACKER BEHAVIOR

The generic behavior of the attacker is depicted in Figure 3 as a finite state machine. This behavioral modeling is based on a probabilistic nature of exploitation of individual vulnerabilities and a non-deterministic approach of the attacker towards their exploitation. We assume a reasonable threat model, in which the attacker is aware of the system vulnerabilities. She may attempt an exploit but fail or even choose not to target a known vulnerability, based on her preferences, such as conserving resources, avoiding detection or paying the required cost.

Hence, at a given entry point, the choice of whether an attacker attempts to exploit this vulnerability is modeled as a non-deterministic decision. If a vulnerability (v_i) is targeted by an attacker, there is a probability e_i that it will be exploited and $1-e_i$ that the exploit will fail. For each attempted exploit the attacker has to pay an overhead, modeled as the *Cost* variable ($0 \leq Cost \leq Cap$). It is initialized with zero and is sequentially incremented with each exploit attempted by the attacker.

C. MARKOV MODEL

The proposed framework models the IoT architecture, along with the attacker components, as an MDP. We utilize a modular approach for generating the Markov model as depicted in Figure 2.

IoTRiskAnalyzer generates dedicated modules for different IoT entities, as defined by the *system model*. The feature modules (one for each feature $f_i \in F$) model the non-deterministic nature of environmental features being observed or impacted by the system. Each $f_i \in F$ refers to a unique and physically non-overlapping feature such as temperature, humidity and motion. The sensor modules (one for each sensor $s_i \in S$) define the concurrent sensing behavior, where each sensor can observe a single feature (as per environment mappings *EM*), however, multiple sensors can be deployed to observe the same feature (e.g., having multiple smoke sensors in the same room). The sensed values are then communicated to the respective controllers (C) and cloud modules (D) via the link and gateway modules, in accordance with the functional (*FM*) and network (*NM*) mappings, defined by the *system model*. The controller modules also implement the respective *operational policies (P)* and accordingly, issue the commands for the actuator (A) modules, while following the functional mappings defined among them. The actuators, in turn, change the linked features as per their respective environment mappings. The response actor (R) modules represent authorized external agencies, which can access and respond to the sensing data stored in the cloud (D), through the corresponding APIs.

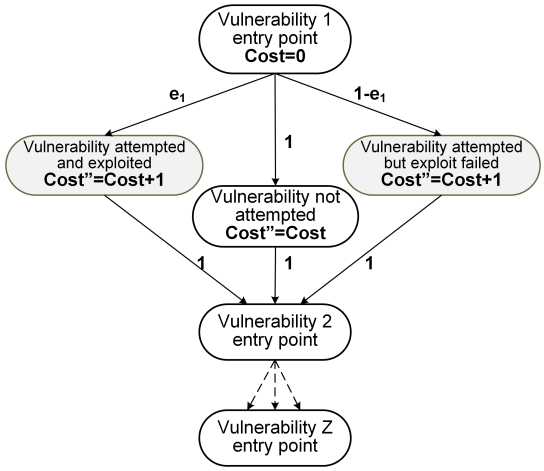

FIGURE 3. Attacker FSM.

A dedicated *attacker* module is used to model the non-deterministic attack behavior. It injects the threat vectors to vulnerable system modules in accordance with the threat definitions and mappings (*TM*), while following the attacker behavior defined in Figure 3.

A typical IoT system is composed of both the sequential and concurrent elements. Concurrency occurs due to the parallel sensing, actuation, controlling and communication aspects of multiple entities; whereas the sequential behavior originates due to the inherent dependencies and interaction

requirements (such as a controller can issue actuation commands only after receiving the desired observations from sensors). To address this aspect, we have used synchronization labels for modeling concurrency and flags and counters for the sequential flow.

V. CASE STUDY: A HOME SECURITY SCENARIO

In order to illustrate the verification methodology used by our framework, we consider a typical home security automation scenario where a tenant, who is a frequent out-of-town traveler, wishes to automate his house. The tenant is especially concerned about the physical security and safety of his property during his absence in this scenario.

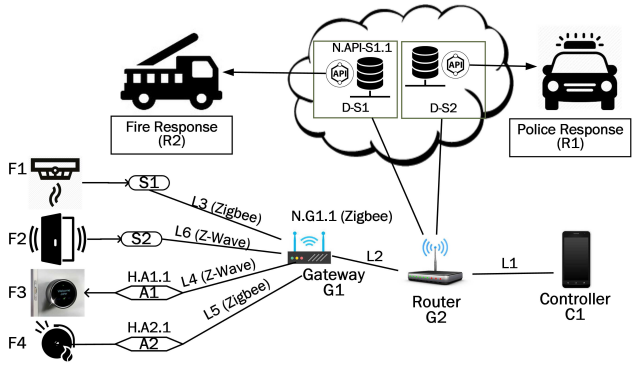

FIGURE 4. Smart home example scenario (Config-1).

Figure 4 portrays a basic system configuration of the planned automation system, comprising of two sensors (smoke (`S1`) and door (`S2`)) and two actuators (door lock (`A1`) and fire alarm (`A2`)). The data from the sensors is processed by the mobile apps, installed on the smartphone controller (`C1`) of the tenant. The same information is also accessible to the law enforcement and emergency response agencies (*response actors*), through respective cloud APIs, as shown in Figure 4. The controller (`C1`) is used to implement local policies, governing the behavior of smart home services. Similarly, emergency response policies are also established at respective agencies as a part of a safe-city project. For this scenario, we consider that the traveling tenant is interested in analyzing the risks associated with two services, namely physical security and fire-response. The initial set of service policies is introduced below:

- **Controller Policy (C1-P1):** If smoke is reported by the smoke sensor (`S1`) then the alarm (`A2`) is activated and the door (`A1`) is unlocked to facilitate fire-fighting and evacuation measures.
- **Police Response Policy (R1-P1):** If the door-lock is armed (house is unoccupied) and the door sensor (`S2`) reports an open door (through the cloud module `D-S2`), then a signal is transmitted to the police-petrol to investigate a potential theft situation.
- **Fire Response Policy (R2-P1):** If smoke is reported by `S1` (through `D-S1`), then a fire-tender has to be sent to tackle the situation.

The vulnerability analysis of the procured components reveals a few weaknesses in individual modules, as labeled in Figure 4. The selected vulnerabilities have repeatedly been discovered in several practically deployed home automation devices [6], [8], [27], [51], [52]. The vulnerability labels as well as their exploitation probabilities (after normalization) are extracted from the empirical risk analysis study of smart home systems, presented by Jacobsson *et al.* [26], and are given in Table 1. This work [26], presented a categorization of the smart home system vulnerabilities with reference to the attack surface (hardware, software, data, communication and human) and entry points (device, gateway, cloud, API and Apps) and assigned risk likelihood scores to each of the identified vulnerabilities. Table 1 also enumerates the threats associated with these vulnerabilities, in coherence with the threat classifications and mappings (*TM*), explained in the leveraged literature [19], [26].

TABLE 1. Vulnerabilities, their exploitation probabilities and associated threats for the considered scenario.

Vuln. Label	Exploit. Prob.	Associated Threat
H.A1.1, H.A2.1	0.40	*Denied Actuation*: Receivers of both the actuators can be jammed
N.G1.1 (Zigbee)	0.40	*Tailored Context / Incorrect Actuation*: Zigbee implementation of G1 can be compromised to modify traffic
N.API-S1.1	0.20	*Tailored Context*: API-S1 protocol vuln. may lead to data manipulation

TABLE 2. Identified attacks, their impacts and pre-conditions for the considered scenario.

Attack	Impact	Attack Pre-conditions
Theft	High	Door unlocked without triggering alarm when smoke not sensed by S1
Missed both alarms	High	The fire incident neither triggered the alarm nor reported to fire deptt.
Missed either alarm	Med.	The fire incident not reported to fire deptt. or did not trigger the alarm, but not both
False alarm	Low	Alarm triggered without actual fire
Compromised evacuation	Low	Door remain locked during a fire incident

The IoT architect wishes to configure the system in such a way that it offers maximum resistance to all possible attack scenarios, in the order prioritized by the tenant. The list of identified attacks are summarized in Table 2. It mentions two high-impact attacks (theft and missed both alarms), one medium-impact attack (missed either alarm) and two low-impact attacks (false alarm and compromised evacuation), prioritized by the tenant based on the attack consequences. For example, in case of a fire incident, the attack where both the actors (*A*2 and *R*2) failed to respond is graded more severe than the '*missed either alarm*' attack (i.e. exactly one of the two actors respond to the fire).

A. MODEL GENERATION

The formal system and threat models of the example scenario are generated by *IoTRiskAnalyzer*, through instantiation

of the generic modules, described in Section IV. We have released the reusable and formally verifiable PRISM code of this example as an open-source [53]. This source code can be adapted for general understanding, training and risk verification of similar smart home architectures through minimal tweaks. The model generation consumes the following information: (i) System configurations given in Figure 4, (ii) Operational policies of controller (`C1`) and response actors (`R1, R2`), stated above and (iii) Component vulnerabilities, their exploitability scores and threat mappings, as summarized in Table 1. The sensing elements are modeled using `sensor_smoke_S1` and `sensor_door_S2` modules, which transmit their observations to `C1` and the respective cloud modules through `gateway_G1` and corresponding links' modules. The actuation commands by `C1` are transmitted (using associated network components) and acted upon (by `actuator_door_A1` and `actuator_alarm_A2` modules), in parallel. The response actors of this scenario are modeled by two modules, namely `police` and `fire_deptt`, each implementing their respective policies.

```
[] link_L4_Jam=0 & cost<=cap & counter=3 &
G1_A1A2_Tx_flag=1 ⟹ 0.4:(link_L4_Jam'=1)
&(cost'=cost+1)&(counter'=4)+0.6:(link_L4_
Jam'=0)&(cost'=cost+1)&(counter'=4);

[] link_L4_Jam=0 & counter=3 & G1_A1A2_Tx_
flag=1 ⟹ (link_L4_Jam'=0) & (counter'=4);
```

Listing 1. Commands for L4 jamming (attacker module).

The `attacker` module models the non-deterministic attacker behavior of exploiting the identified vulnerabilities (Table 1), as described in Figure 3. For instance, the PRISM commands to model the jamming of the link `L4` are given in Listing 1. Here, the variable `link_L4_Jam` represents the status of the link `L4`. `G1_A1A2_Tx_flag` is the flag for sequential flow, representing whether or not the Gateway `G1` has sent the data to actuators `A1` and `A2`. The `counter` variable is used for logical ordering of the possible attacks and `cost` defines the total number of vulnerabilities attempted, which is always less than the attacker capability (*cap* variable). The first command models the scenario when an attacker attempts to jam the link `L4` with the success probability of 0.4, while the second command captures the situation when the attacker chooses not to attempt the `L4` jamming. The antecedent (left-hand side) of the implication specifies the satisfying conditions for the activation of that command and it is same for both the commands in Listing 1 to model the non-determinism.

B. FORMALIZING PROPERTIES

As a next step, the scenario-specific formal model is tested against suitable properties, developed using the PCTL logic. We classify these properties as system and attack properties, as defined below:

```
Property-1: Theft
Pmax=?[smoke=0 U (door=1&alarm=0&cost<=cap)]
Property-2: Compromised Evacuation
filter(max, Pmax=? [ G door!=1&cost<=cap],
smoke=1)
Property-3: Missed Either Alarm
filter(max, Pmax=? [ F ((alarm=1&fire_t=0)|
(alarm=0&fire_t=1))&cost<=cap ], smoke=1)
```

Listing 2. Sample attack properties.

1) **System Properties** are used to verify the soundness of our MDP model (e.g., deadlock freeness and state reachability).
2) **Attack Properties** are defined to get the maximum likelihood of satisfying the pre-conditions (Table 2), which can achieve the corresponding attack objectives.

Some of the attack properties are presented in Listing 2. `Property-1` verifies the likelihood of theft, where an attacker is able to unlock the door while blocking the alarm, even if there is no smoke. Similarly, `Property-2` and `Property-3` verify the likelihood of 'compromised evacuation' and 'missed either alarm' attacks, respectively.

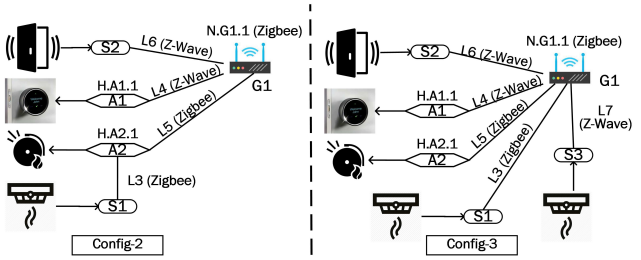

FIGURE 5. Alternate configurations of example scenario.

VI. EVALUATION AND IMPLEMENTATION

In this section, we present and analyze the verification results of the example scenario (Section V), with an aim to establish that the risk exposure scores significantly depend on the candidate system configurations as well as the operational policies. To achieve this, we evaluated three candidate configurations, using our *IoTRiskAnalyzer* framework. We denote the reference configuration (Figure 4) as *Config-1*, and present the architecture of other two configurations in Figure 5. In *Config-2*, the location of the smoke sensor (`S1`) has been changed to form a mesh network over the Zigbee technology. This configuration does not require any extra budget as it only involves connectivity adjustments (*NM*) of existing devices. Contrary to that, we have introduced a redundancy in *Config-3* by adding a new smoke sensor (`S3`), which works on the Z-wave technology. The corresponding policies (i.e. *C1-P1* and *R2-P1*) for *Config-3* were also changed as; "`Config-3P1`: *Unlock the door and send the fire-tender only if both sensors (S1 and S3) report smoke but trigger alarm if any of the two sensors report smoke*". The security of all these three configurations was tested against attacks mentioned in Table 2, by defining suitable properties as demonstrated in Listing 2.

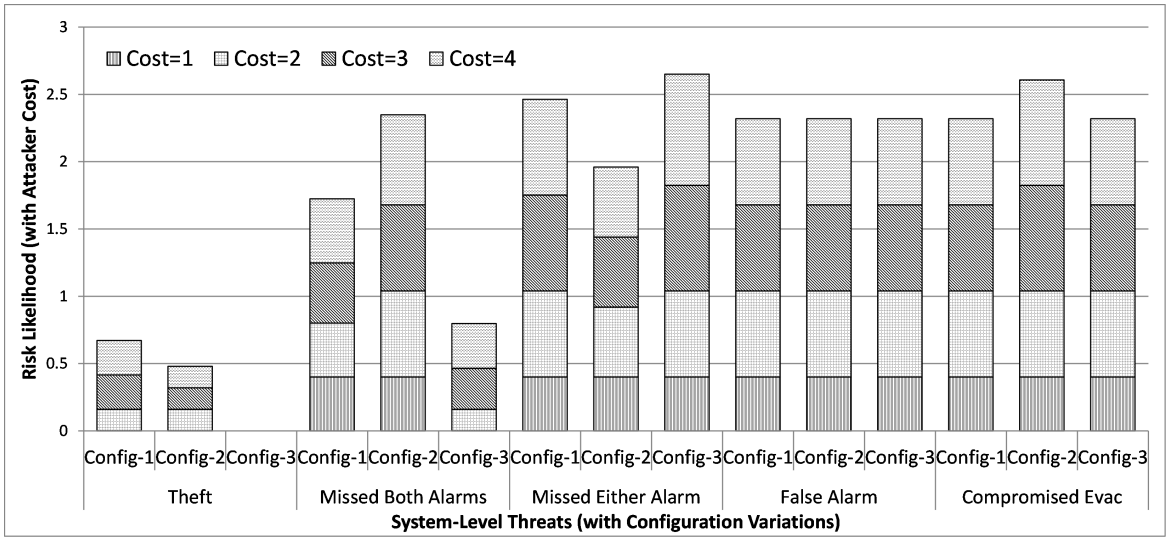

FIGURE 6. **Verified risk exposure scores (`Config-3` utilizes `Config-3P1` policy).**

A. EXPERIMENTAL RESULTS

1) RISK Vis-à-Vis SYSTEM CONFIGURATION

Based on the input MDP model and customized PCTL properties, as discussed in Section V, our framework generated the risk exposure scores to different attacks in the form of comparative graphs. The results are plotted in Figure 6. The results revealed that *Config-2* is more resilient to theft and 'missed either alarm'. The main reason behind this finding is that if the attacker chooses to jam the Link `L5` to block the fire alarm, she will not be able to modify the sensing values at the gateway as well (because `S1` values will also fail to reach `G1`). Contrarily, *Config-2* is more risk-prone towards compromised evacuation and missing both alarms in case of fire, due to the additional dependency of `S1` over `A2` for transmitting its observations. *Config-3* is observed to be more resilient against both the high-impact attacks due to the introduced redundancy (Sensor `S3`).

2) RISK Vis-à-Vis OPERATIONAL POLICY

Next, we evaluated the impact of different *C1* policies over the threat-resiliency for *Config-3*. To achieve this, we evaluated the following three $C1$ policies:

- **Config-3P1**: If smoke is reported by any of the two sensors ($S1$ or $S3$) then the alarm ($A2$) is triggered but the door ($A1$) is unlocked, only if both the sensors report smoke.

$$S1 \cap S3 \Rightarrow A1; \quad S1 \cup S3 \Rightarrow A2$$

- **Config3-P2**: If the smoke is reported by any of the two sensors then the door ($A1$) is opened but the alarm ($A2$) is activated only if both the sensors report the presence of smoke.

$$S1 \cup S3 \Rightarrow A1; \quad S1 \cap S3 \Rightarrow A2$$

- **Config3-P3**: If the smoke is reported by any of the two sensors then both the door ($A1$) and the alarm ($A2$) are actuated.

$$S1 \cup S3 \Rightarrow A1 \cap A2$$

Figure 7 plots the verified risk-scores of these three policies with reference to the threats under consideration and the changing attacker cost. The results revealed that there are no chances of *theft* for policy `P1` (even with a very powerful attacker, i.e., `cost`=4), through the exploitation of the known set of vulnerabilities. Moreover, `P1` poses a minimum risk exposure to the other high-impact attack as well (i.e., *Missed both alarms*). However, this policy is relatively more risk-prone to the medium and low-impact attacks. Contrarily, policy `P2` extends a maximum resilience against the medium and low-impact attacks but has worst scores for both the high-impact attacks. In the case of Policy `P3`, the attacker is required to exploit at least two vulnerabilities to meet the objectives of both the high-impact attacks and hence, this policy offers good resistance against a weak adversary (i.e. $Cap = 1$). It is also noteworthy that the risk exposure for some attacks is independent of the attacker capabilities (e.g., 'compromised evacuation' for `P2` and `P3`).

B. DISCUSSION

The emerging IoT systems comprise of complex functional couplings and cascaded dependencies and host several safety-critical services. Security analysis of such critical systems cannot rely on manual or simulation-driven approaches, due to their incomplete coverage. *IoTRiskAnalyzer* offers a provable risk verification framework, through exhaustive testing of all possible behaviors of input system and threat models, including the 'corner-cases', which may otherwise be overlooked by the traditional approaches.

We utilized *IoTRiskAnalyzer* for verifying the risk exposure of different IoT configurations while manipulating

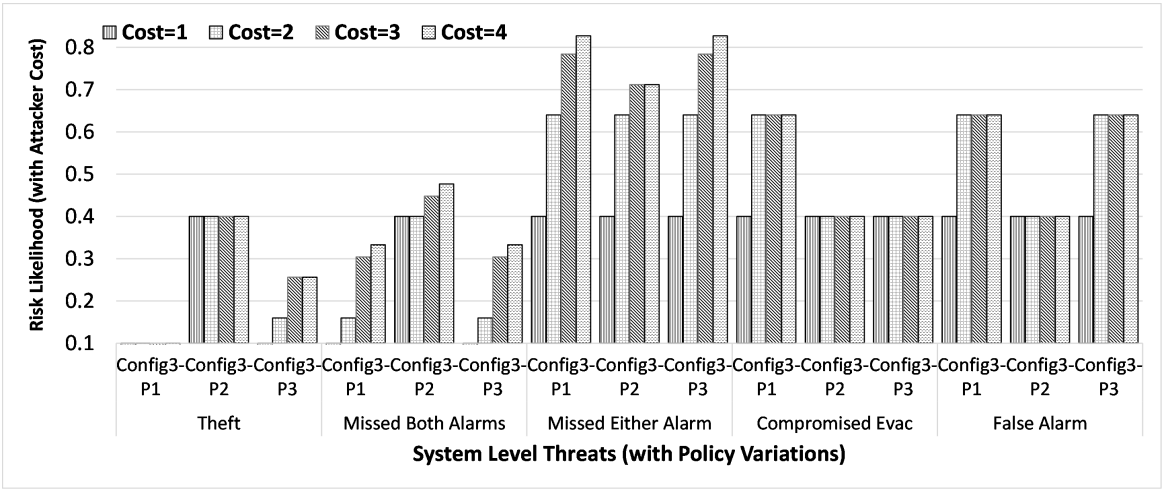

FIGURE 7. Impact of policy on risk exposure.

the input parameters, such as connectivity, redundancy, vulnerable entry points and operational policies. During our experiments, *IoTRiskAnalyzer* exposed several such complex attack vectors, which were otherwise difficult to comprehend through traditional approaches. The key findings of these experiments are summarized here:

1) Given a known set of vulnerabilities, the amount of risk to system-level security threats vary significantly with system and policy configurations (as clear from Figures 6 and 7).
2) Introducing redundancy must equally be complemented by implementing a right set of policies for reducing risk exposure. As a counter-example, consider that in case of theft, the risk scores for `Config3-P2` were even higher than both the `Configs-1` and `2`, despite the introduction of redundancy.
3) Some candidate system and policy configurations offer a risk-tradeoff for different attack situations. These situations can be scrutinized through attack impact analysis, as demonstrated by our work. For example, `Config-3` is a preferred solution for the designer as it offers less risk to both high-impact attacks, despite being weak against the medium-impact attack (i.e., 'missed either alarm'). Moreover, within `Config-3`, `P1` is the preferred policy, owing to its minimum risk scores for the high-impact attacks.
4) Due to the inherent functional dependencies among IoT devices, individual vulnerabilities may be exploited to cause a cascaded impact. For example, a secure actuator can be compromised by exploiting a vulnerable sensor, just because its controller is dependent upon the context information produced by that sensor.

C. IMPLEMENTATION

We have used the Java API for implementing the *IoTRiskAnalyzer* framework. The API reads the input IoT configurations from a text file and automatically generates the MDP model, in compliance with the language syntax used by the PRISM model checker. This model can then be analyzed against suitable system and attack properties, using the PRISM tool. Our framework utilizes the default Hybrid engine of PRISM Version 4.3.1 for verification of input model. It employs the module renaming feature of PRISM, where ever applicable, for ensuring modeling scalability, as the generic system modules can be utilized to model multiple instances of similar entities.

The performance of *IoTRiskAnalyzer* directly depends on the system size, the complexity of configuration requirements and the number of vulnerabilities and policies. To optimize the performance of *IoTRiskAnalyzer*, we implemented model-abstractions and model-decompositions at different layers, as discussed ahead.

The role of a given component in a model generated by *IoTRiskAnalyzer* depends on: (a) Whether the component contains any vulnerability or (b) Whether the component actively transforms the state due to functional requirements. We applied an abstraction by omitting the modules not satisfying the above-stated requirements. For example, in our reference scenario, router `R1` and links `L1` and `L2` were not modeled, since they were transparent for the system and inaccessible to the attacker. Another level of abstraction was applied over individual modules based on the policy. For example, a realistic temperature sensor may require a range of integers to precisely model its behavior. However, if the policy requires the decision to be made only at a threshold of $77°F$, then the sensor can be modeled to observe and report boolean values (i.e., $T > 77$ and $T \leq 77$).

In addition to abstractions, model-decompositions were performed by splitting large system models into multiple independent sub-systems, based on different types of entity-mappings, as defined in Section IV. These abstractions and decompositions considerably reduced the state space and significantly improved the performance. As an example, the framework, while running on a Core-i7 machine with 8 GB of RAM, consumed small fractions of seconds, both for model construction and property verification, for all configurations of our case study.

VII. CONCLUSION

In this paper, we presented *IoTRiskAnalyzer*, which is a novel framework for automated verification and probabilistic quantification of attack likelihoods against generic IoT system configurations. The reports delivered by *IoTRiskAnalyzer* can help IoT engineers to select the best possible system and policy configurations from a security standpoint. The framework can also assist in analyzing the impact of component-level vulnerabilities over system-level threats. In the future, we plan to integrate and extend the contributions made by the IoTSAT [19] and IoTRiskAnalyzer frameworks, towards budget constrained security planning of IoT systems. This envisaged tool-chain will assist non-expert IoT designers to plan, verify and optimize the security of their configurations, within the affordable budget, after putting minimal technical efforts.

REFERENCES

[1] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, Eds., *Vision and Challenges for Realising the Internet of Things*. rue Mercier, Luxembourg: Publications Office of the European Union, 2010. [Online]. Available: http://www.eurolibnet.eu/3/72/&for=show&tid=7944

[2] D. Evans. The Internet of Things—How the next evolution of the Internet is changing everything. Cisco, Inc., accessed on Mar. 9, 2017. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[3] D. Lund, C. MacGillivray, V. Turner, and M. Morales, "Worldwide and regional Internet of Things (IoT) 2014–2020 forecast: A virtuous circle of proven value and demand," Int. Data Corp., Framingham, MA, USA, Tech. Rep. IDC#248451, Dec. 2014, doi: 10.2824/33134.

[4] V. Turner, J. F. Gantz, D. Reinsel, and S. Minton, "The digital universe of opportunities: Rich data and the increasing value of the Internet of Things," IDC Anal. Future, Framingham, MA, USA, Tech. Rep., 2014.

[5] Gartner, Inc. (2016). *Gartner Says Worldwide IoT Security Spending to Reach $ 348 Million in 2016*, accessed no Mar. 9, 2017. [Online]. Available: http://www.gartner.com/newsroom/id/3291817

[6] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proc. IEEE Symp. Secur. Privacy*, May 2016, pp. 636–654.

[7] K. Angrishi. (Feb. 2017). "Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV): IoT Botnets." [Online]. Available: https://arxiv.org/abs/1702.03681

[8] E. Ronen, C. O'Flynn, A. Shamir, and A.-O. Weingarten. (Nov. 2016). *IoT Goes Nuclear: Creating a ZigBee Chain Reaction*, accessed on Mar. 9, 2017. [Online]. Available: https://eprint.iacr.org/2016/1047

[9] B. Fouladi and S. Ghanoun, "Security evaluation of the Z-wave wireless protocol," *Black Hat USA*, vol. 1, pp. 1–6, Aug. 2013.

[10] B. Schneier. *The Internet of Things is Wildly Insecure-and Often Unpatchable*, accessed on Mar. 9, 2017. [Online]. Available: https://www.schneier.com/essays/archives/2014/01/the_internet_of_thin.html

[11] M. Hamdi and H. Abie, "Game-based adaptive security in the Internet of Things for eHealth," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2014, pp. 920–925.

[12] R. Zheng *et al.*, "An IoT security risk autonomic assessment algorithm," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 11, no. 2, pp. 819–826, 2013.

[13] T. Nipkow, "Advances in probabilistic model checking," in *Software Safety and Security: Tools for Analysis and Verification*, vol. 33. Amsterdam, The Netherlands: IOS Press, 2012, pp. 126–151.

[14] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 1994

[15] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification* (Lecture Notes in Computer Science), vol. 6806. Berlin, Germany: Springer, 2011, pp. 585–591.

[16] M. U. Sardar, N. Afaq, K. A. Hoque, T. T. Johnson, and O. Hasan, "Probabilistic formal verification of the SATS concept of operation," in *NASA Formal Methods*, vol. 9690. New York, NY, USA: Springer, 2016, pp. 191–205.

[17] M. Q. Ali and E. Al-Shaer, "Probabilistic model checking for AMI intrusion detection," in *Proc. IEEE SmartGridComm*, Oct. 2013, pp. 468–473.

[18] S. Ouchani, O. A. Mohamed, and M. Debbabi, "A security risk assessment framework for SysML activity diagrams," in *Proc. IEEE 7th Int. Conf. Softw. Secur. Rel.*, Jun. 2013, pp. 227–236.

[19] M. Mohsin, Z. Anwar, G. Husari, E. Al-Shaer, and M. A. Rahman, "IoTSAT: A formal framework for security analysis of the Internet of Things (IoT)," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2016, pp. 180–188.

[20] L. de Moura and N. Bjørner, "Satisfiability modulo theories: Introduction and applications," *Commun. ACM*, vol. 54, no. 9, pp. 69–77, Sep. 2011.

[21] C. Liu, Y. Zhang, J. Zeng, L. Peng, and R. Chen, "Research on dynamical security risk assessment for the Internet of Things inspired by immunology," in *Proc. IEEE 8th Int. Conf. Natural Comput. (ICNC)*, May 2012, pp. 874–878.

[22] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things," *Computer*, vol. 44, no. 9, pp. 51–58, Sep. 2011.

[23] D. Podgórski, K. Majchrzycka, A. Dabrowska, G. Gralewicz, and M. Okrasa, "Towards a conceptual framework of OSH risk management in smart working environments based on smart PPE, ambient intelligence and the Internet of Things technologies," *Int. J. Occupat. Safety Ergon.*, vol. 23, no. 1, pp. 1–20, 2016.

[24] K. Djemame, D. Armstrong, M. Kiran, and M. Jiang, "A risk assessment framework and software toolkit for cloud service ecosystems," in *Cloud Computing*. Wilmington, DE, USA: Xpert Publishing Services, 2011, pp. 119–126.

[25] S.-I. Chang, A. Huang, L.-M. Chang, and J.-C. Liao, "Risk factors of enterprise internal control: Governance refers to Internet of Things (IoT) environment," in *Proc. RISK*, 2016, pp. 1–11.

[26] A. Jacobsson, M. Boldt, and B. Carlsson, "A risk analysis of a smart home automation system," *Future Generat. Comput. Syst.*, vol. 56, pp. 719–733, Mar. 2016.

[27] D. Barnard-Wills, L. Marinos, and S. Portesi, "Threat landscape and good practice guide for smart home and converged media," Eur. Union Agency Netw. Inf. Secur., Heraklion, Greece, Tech. Rep., Dec. 2014.

[28] T. Denning, T. Kohno, and H. M. Levy, "Computer security and the modern home," *Commun. ACM*, vol. 56, no. 1, pp. 94–103, 2013.

[29] D. Pishva and K. Takeda, "Product-based security model for smart home appliances," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 23, no. 10, pp. 32–41, Oct. 2008.

[30] F. Corno and M. Sanaullah, "Design-time formal verification for smart environments: An exploratory perspective," *J. Ambient Intell. Humanized Comput.*, vol. 5, no. 4, pp. 581–599, 2014.

[31] P. Mundhenk, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty, "Security analysis of automotive architectures using probabilistic model checking," in *Proc. ACM/EDAC/IEEE Design Autom. Conf.*, Jun. 2015, pp. 1–6.

[32] V. Kulkarni, *Modeling and Analysis of Stochastic Systems*. London, U.K.: Chapman & Hall, 1995

[33] E. Kang, S. Adepu, D. Jackson, and A. P. Mathur, "Model-based security analysis of a water treatment system," in *Proc. 2nd Int. Workshop Softw. Eng. Smart Cyber-Phys. Syst.*, 2016, pp. 22–28.

[34] T. Ahmed and A. R. Tripathi, "Static verification of security requirements in role based CSCW systems," in *Proc. 8th ACM Symp. Access Control Models Technol. (SACMAT)*, New York, NY, USA, 2003, pp. 196–203. [Online]. Available: http://doi.acm.org/10.1145/775412.775438

[35] N. Trcka, M. Moulin, S. Bopardikar, and A. Speranzon, "A formal verification approach to revealing stealth attacks on networked control systems," in *Proc. 3rd Int. Conf. High Conf. Netw. Syst. (HiCoNS)*, New York, NY, USA, 2014, pp. 67–76. [Online]. Available: http://doi.acm.org/10.1145/2566468.2566484

[36] T. L. Guilly, J. H. Smedegard, T. Pedersen, and A. Skou, "To do and not to do: Constrained scenarios for safe smart house," in *Proc. IEEE Int. Conf. Intell. Environ. (IE)*, Jul. 2015, pp. 17–24.

[37] J. C. Augusto and M. J. Hornos, "Software simulation and verification to increase the reliability of Intelligent Environments," *Adv. Eng. Softw.*, vol. 58, pp. 18–34, Apr. 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/aes/aes58.html#AugustoH13

[38] F. Corno and M. Sanaullah, "Modeling and formal verification of smart environments," *Secur. Commun. Netw.*, vol. 7, no. 10, pp. 1582–1598, 2014. [Online]. Available: http://dx.doi.org/10.1002/sec.794

[39] A. Coronato and G. D. Pietro, "Formal specification and verification of ubiquitous and pervasive systems," *ACM Trans. Auto. Adapt. Syst.*, vol. 6, no. 1, p. 9, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/taas/taas6.html#CoronatoP11
[40] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008
[41] U. Pervez, A. Mahmood, O. Hasan, K. Latif, and A. Gawanmeh, "Improvement strategies for device interoperability middleware using formal reliability analysis," *Scalable Comput., Pract. Exper.*, vol. 17, no. 3, pp. 150–170, 2016.
[42] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen, "The ins and outs of the probabilistic model checker MRMC," *Perform. Eval.*, vol. 68, no. 2, pp. 90–104, 2011.
[43] K. Sen, M. Viswanathan, and G. Agha, "VESTA: A statistical model-checker and analyzer for probabilistic systems," in *Proc. 2nd Int. Conf. Quant. Eval. Syst.*, vol. 5. Sep. 2005, pp. 251–252.
[44] H. L. Younes, "Ymer: A statistical model checker," in *Computer Aided Verification*, vol. 3576. Berlin, Germany: Springer, 2005, pp. 429–433.
[45] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle, "ETMCC: Model checking performability properties of Markov chains," in *Proc. DSN*, 2003, p. 673.
[46] D. N. Jansen, J.-P. Katoen, M. Oldenkamp, M. Stoelinga, and I. Zapreev, "How fast and fat is your probabilistic model checker? An experimental performance comparison," in *Hardware and Software: Verification and Testing*, vol. 4899. Berlin, Germany: Springer, 2008, pp. 69–85.
[47] R. Segala and N. Lynch, "Probabilistic simulations for probabilistic processes," *Nordic J. Comput.*, vol. 2, no. 2, pp. 250–273, 1995.
[48] D. Beauquier, "On probabilistic timed automata," *Theor. Comput. Sci.*, vol. 292, no. 1, pp. 65–84, 2003.
[49] R. Alur and T. A. Henzinger, "Reactive modules," *Formal Methods Syst. Design*, vol. 15, no. 1, pp. 7–48, 1999.
[50] A. Bianco and L. de Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Foundations of Software Technology and Theoretical Computer Science*, vol. 1026. Berlin, Germany: Springer, 1995, pp. 499–513.
[51] S. Curtis. *Home Invasion 2.0: How Criminals Could Hack Your House*, accessed on Mar. 9, 2017. [Online]. Available: http://www.telegraph.co.uk/technology/internet-security/10218824/Home-invasion-2.0-how-criminals-could-hack-your-house.html
[52] C. B. Review. *Veracode Warns IoT a Pathway for Cybercrime*, accessed on Mar. 9, 2017. [Online]. Available: http://www.cbronline.com/news/internet-of-things/consumer/veracode-warns-iot-a-pathway-for-cybercrime-4548343
[53] M. Mohsin, M. U. Sardar, O. Hasan, and Z. Anwar. *IoTRiskAnalyzer*, accessed on Mar. 9, 2017. [Online]. Available: https://github.com/mujahidmohsin/IoTRiskAnalyzer

MUJAHID MOHSIN received the M.S. degree (Hons.) in information security from the National University of Sciences and Technology (NUST), Pakistan, in 2010, where he is currently pursuing the Ph.D. degree in computer and communication security, under the supervision of Dr. Z. Anwar. He was a Researcher with CERN and the Cyber Defense and Network Assurability Center, University of North Carolina at Charlotte, USA. He is also a Research Assistant with the Systems Research Group, NUST. His current research interests include automated security analytics, the Internet of Things security, formal methods, and actionable cyber threat intelligence. He was a recipient of the Ph.D. Scholarship from the Higher Education Commission, Pakistan, and the Rector's NUST High Achiever Certificate in 2015.

MUHAMMAD USAMA SARDAR received the B.Sc. degree in electronics engineering from the Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan, in 2009, and the M.S. degree (Hons.) in electrical engineering from the National University of Sciences and Technology (NUST), Pakistan, in 2015. He was a Researcher with the Chair of Embedded Systems, Karlsruhe Institute of Technology, Germany. He is currently a Research Assistant with the System Analysis and Verification Laboratory, NUST. His main research interests include probabilistic model checking-based formal verification of safety-critical systems. His research work has resulted in publications at top international forums, such as the *Journal of Parallel and Distributed Computing* and the NASA Formal Methods Symposium.

OSMAN HASAN (S'07–M'11–SM'14) received the B.Eng. degree (Hons.) from the University of Engineering and Technology, Pakistan, in 1997, and the M.Eng. and Ph.D. degrees from Concordia University, Montreal, Canada, in 2001 and 2008, respectively. He was an ASIC Design Engineer with LSI Logic Corporation, Ottawa, Canada, from 2001 to 2003, and a Research Associate with Concordia University, Montreal, Canada, from 2008 to 2009. He is currently an Assistant Professor with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad, Pakistan. He is the Founder and Director of the System Analysis and Verification Laboratory, NUST, which mainly focuses on the design and formal verification of safety-critical systems, including e-health and digital systems. He is a member of the ACM, the Association for Automated Reasoning, and the Pakistan Engineering Council. He was a recipient of several awards and distinctions, including the Pakistan's Higher Education Commission's Best University Teacher in 2010 and the Best Young Researcher Award in 2011, and the President's Gold Medal for the best teacher of the University from NUST in 2015.

ZAHID ANWAR received the Ph.D. and M.S. degrees in computer sciences from the University of Illinois at Urbana–Champaign, USA, in 2008 and 2005, respectively. He was a Software Engineer and a Researcher with IBM, Intel, Motorola, the National Center for Supercomputing Applications, xFlow Research, and CERN on projects related to information security and data analytics. He was a Post-Doctoral Fellow with Concordia University, Canada, and a Faculty Member with the University of North Carolina at Charlotte, USA, Fontbonne University, USA, and the National University of Sciences and Technology (NUST), Pakistan. He is currently the Laboratory Director of the Systems Research Group, NUST.