A Language for Analyzing Security of IOT Systems

Delphine Beaulaton, Najah Ben Said, Ioana Cristescu, Régis Fleurquin, Axel Legay, Jean Quilbeuf, Salah Sadou

A Language for Analyzing Security of IOT Systems

Delphine Beaulaton*, Najah Ben Said†, Ioana Cristescu†, Régis Fleurquin*,
Axel Legay†, Jean Quilbeuf † and Salah Sadou*
*Univ. South Brittany, Irisa, Vannes, France
†INRIA - France
Email: * name.surname@irisa.fr, † name.surname@inria.fr

***Abstract*—The control and protection of a user data is a very important aspect in the design and deployment of the Internet of Things (IoT). In this paper we propose a security-based modelling language for IoT systems, which explicitly represents data access controls. The language leverages the analysis of potential security failures resulting from a series of interactions between heterogeneous components of a system. We implemented a tool that automatically transforms IoT models into BIP models, which can then be simulated and analyzed for security guarantees. We illustrate the features of our language with a use-case inspired by an industrial scenario.**

I. Introduction

The IoT systems have a profound impact on our daily lives, including technologies for the home, for health, for transportation, and for managing our natural resources. It provides a new range of services but it also leaves the systems more vulnerable to attacks [1]. Many strategies are put in place aiming to build robust and practical IoT systems with great concerns about security privacy and public policy [2].

A striking example of IoT vulnerability is the mirai botnet [1]. One part of mirai infects IoT devices by trying to connect with a list of about 20 common or default passwords. Upon success, a malicious binary containing a server is uploaded and run. This binary registers itself to a central server and then waits for orders from this server. Whenever the attacker wants to launch a denial of service attack, they simply issue the command through the central server, which causes all infected IoT devices to start sending packets.

IoT vulnerabilities may have tremendous consequences. Consider Hospitals system as another example. An increasing number of connected devices expands the actions of the medical staff by allowing them to treat patients at home, making medical files available for remote consultancy, etc. In turn, these services come with the cost of increasing the attack surface of the system. In 2016 such a network of connected devices was exploited to launch a series of attacks which paralyzed several UK member hospitals. These medical device hijacks, baptized ”MEDJACK” by the TrapX organization [3], caused major disturbances in the functioning of the hospitals.

Hospitals databases can also be the target of attacks on their patients. Indeed, medical databases may be of interest for various attackers wanting to get access to medical secrets and could also have dramatic consequences for patients if they are tampered with. For instance, if the indication that a patient suffers from a severe allergy is removed, such a patient could die.

In order to prevent attacks on IoT in general, several approaches exist. These approaches range from best security practices (such as changing the default password of IoT devices) to intrusion detection systems (IDS) deployed over a set of IoT devices. In this paper, we focus on the analysis of systems at design stage.

At the design stage of a IoT system, we know the IoT devices of the systems, we have a specification for their behaviour and how they are supposed to communicate with each other.

The main contributions of our paper are as follows:

- We propose a Model-based security language of IoT systems that enables users to create models of their IoT systems and to make analysis of the likelihoods of cyber-attacks to occur and succeed. The modeling language describes the interactions between different entities. The entities can either be human actuators or “Things” (i.e, hardware, sensors, software tools, ..). The main trigger for security problems is human behaviour, either unintentional or malicious. The data known by an entity influences the actions it can execute. For instance, an attacker can launch a phishing attack via email, only if it knows the email address of the target. Another feature of our modeling language is that security failures are modeled as a sequence of simpler steps, in the spirit of attack tree [4]: several interactions need to occur before the attacker can complete its attack. The language we provide is a simple language for security experts to help them assess the vulnerabilities of an IoT system. This language is not only IoT specific, but it can also be used to model systems with knowledge based interactions.
- We propose a transformation from our modeling language to a component-based model (in our case BIP). Hence a security analysis of a composite model of systems can take place as well as the application to these systems of formal verification techniques developed for BIP, e.g deadlock detection. Furthermore, we proved the correctness of our transformation.
- We implemented a transformation tool and we illustrate its use on an example involving cyber-attacks on a smart hospital. The attacker tries to access the confidential data of the organization (such as employee credentials, patient files, etc).

The paper is structured as follows. Section II presents the syntax and the semantics of the IoT modeling language. The transformation from IoT models to BIP models is presented in Section III. Section IV presents the implementation tool and a validation of the approach using a Smart Hospital running example. Finally, Section V discusses the related work and in Section VI we conclude and we propose directions for future work.

II. Language

The language we present in this section aims to model the IoT paradigm in a simple way. The IoT systems are composed of *entities* having some communication capabilities between each other. Two entities can communicate if (i) they are connected through a *communication protocol* and (ii) they satisfy some constraints imposed by the protocol. For instance, protocols can ask the participants to perform some identity checks. In our language, each entity has a *knowledge* and the protocols ensures that entities only communicate if they have the "knowledge" necessary (i.e. passwords, keys, . . .). In our introductory example, the url of a website is part of the knowledge. Users can only access a website if they have the url in their knowledge base. For simplicity, we model an entity's knowledge as a set of *values*.

Here after we give a formal definition of the IoT core model.

A. Model Basics

We formally define the IoT model as a set of heterogeneous entities E that have a unique identifier (actors and physical devices), ranged over by e_1, e_2,. . . , and communicating between each others through a set of protocols C, ranged over by c_1, c_2,. . . . Let Val be a set of *values* (for example string or integer constants), ranged over v_1,v_2, ... The behaviour of an entity is modeled by a CCS-like [5]. Their syntax is given in Figure 1.

$$
\begin{array}{rll}
\mathit{Process} & P,Q \ ::= \ 0 \mid a.P \mid a.P + b.Q \mid A & \\
\mathit{Action} & a,b \ ::= e \xrightarrow[v]{c} e' & \mathit{Send} \\
 & |e \xleftarrow{c} e' & \mathit{Receive} \\
 & |e \underset{v}{\rightarrowtail} e' & \mathit{Leak} \\
 & |e \leftarrowtail e' & \mathit{Collect} \\
 & |\tau & \mathit{Internal} \\
\mathit{Definition} & A \overset{\text{def}}{=} P &
\end{array}
$$

Fig. 1. Syntax of the core IoT-calculus

The language includes 0, that performs no action. The $a.P$ performs an action a and continues as P. $a.P + b.Q$ behaves as either $a.P$ or as $b.Q$ [1]. A is a definition of a (potentially recursive) . es communicate using the Send, $\mathit{Receive}$, Leak and $\mathit{Collect}$ actions. During a communication entities can exchange values. In order to add the received values under the right protocol in the receiver's knowledge, we define a function $\mathsf{protocol} : \mathit{Val} \to C$.

A of an entity e can send to another entity e' a value v using the protocol c. A which receives a value has only to specify the identity of the participants, e and e', and the protocol c. A leak action is another type of send, where the participants do not need to agree on a protocol. Finally, a collect is the counterpart of receiving in the case of a leak.

τ is an internal action, which is useful in our language for the sum construct. When a has a choice, as in $a.P + \tau.Q$, it either can do the action a or not. In the latter, the internal τ action is performed instead. However, nothing prevents a user to write es of the form $\tau.P$.

B. State of the System

A *knowledge* function associates to each entity and each protocol a subset of values, $K : E \times C \to \mathcal{P}(\mathit{Val})$. For simplicity we write k_i for the function $K(e_i) : C \to \mathcal{P}(\mathit{Val})$ and k_i^c for the set of values "known" by entity i under the protocol c.

We define a *congruence* relation on es $\equiv_P \subseteq P \times P$ as the smallest equivalence relation which includes:

- the abelian monoid laws for $+$:

$$P + Q \equiv_P Q + P \quad (P + Q) + R \equiv_P P + (Q + R)$$
$$P + 0 \equiv_P P$$

- the unfolding law: $A \equiv_P P$ if $A \overset{\text{def}}{=} P$.

Each entity, has at any state of its computation, a running and a knowledge. States compose using the parallel composition constructor $|$:

$$s \ ::= \emptyset \ \Big| \ \langle P, k \rangle \ \Big| \ s \mid s.$$

We also introduce a congruence relation on states $\equiv_s \subseteq s \times s$, as the smallest equivalence relations which

- generalizes the congruence on es

$$\frac{P \equiv_P Q}{\langle P, k \rangle \equiv_S \langle Q, k \rangle}$$

- includes the Abelian monoid laws for $|$:

$$s|t \equiv_s t|s \quad (s|t)|q \equiv_s s|(t|q) \quad s|\emptyset \equiv_s s$$

The (global) state of the system consists of the parallel composition of all entities states i.e. $s_1 \mid \cdots \mid s_n$, where s_i is the current state of the entity e_i.

C. Operational semantics

The operational semantics of the system is defined by the inference rules of Figure 2. The system can execute a $\mathit{SendReceive}$ interaction between two entities e_1 and e_2 if they agree on the protocol c and if they satisfy the constraint imposed by the protocol: the sender has a value in its knowledge that is also known by the receiver. The values received

[1] We use *guarded sum* here (i.e. $a.P + b.Q$ instead of $P + Q$), which is motivated by the fact that as future work, the language will be augmented with probabilities, similarly to [6].

$$\textsc{SendReceive}\ \frac{\exists v \in k_1^c \text{ s.t. } v \in k_2^c \qquad c' = \mathsf{protocol}(v')}{\langle e_1 \xrightarrow[v']{c} e_2.P_1, k_1\rangle | \langle e_2 \xleftarrow{c} e_1.P_2, k_2\rangle \to \langle P_1, k_1\rangle | \langle P_2, k_2^{c'} \uplus \{v'\}\rangle}$$

$$\textsc{LeakCollect}\ \frac{c' = \mathsf{protocol}(v')}{\langle e_1 \underset{v'}{\rightarrowtail} e_2.P_1, k_1\rangle | \langle e_2 \leftarrowtail e_1.P_2, k_2\rangle \to \langle P_1, k_1\rangle | \langle P_2, k_2^{c'} \uplus \{v'\}\rangle}$$

$$\textsc{Internal}\ \langle \tau.P, k\rangle \to \langle P, k\rangle$$

$$\textsc{Sum}\ \frac{\langle P_i, k_i\rangle | \langle P_j, k_j\rangle \to \langle P_i', k_i'\rangle | \langle P_j', k_j'\rangle}{\langle P_i + Q_i, k_i\rangle | \langle P_j + Q_j, k_j\rangle \to \langle P_i', k_i'\rangle | \langle P_j', k_j'\rangle}$$

$$\textsc{Par}\ \frac{s \to s'}{s \mid t \to s' \mid t}$$

$$\textsc{Congruence}\ \frac{s \equiv_s t \to s' \equiv_s t'}{s \to s'}$$

Fig. 2. The operational semantics of an IoT system

by entity e_2 is added to its knowledge under the protocol c'. The interaction *LeakCollect* is similar, except that there are no protocols or constraints to be checked. Note that both the identity of the sender and of the receiver are specified in an interaction. The rules *Internal* and *Sum* allow a to do an internal action and make a non deterministic choice, respectively.

The two remaining rules *Par* and *Congruence* are on states and ensure that the rules can proceed regardless of the syntactical form of the system.

Example 1. Let $E = \{attacker, hospital, employee\}$ be three entities which communicate with each other using three protocols $C = \{url, message, mail\}$. The *attacker* has the Attacker as initial , and the initial knowledge $k_{attacker}$ and similarly for the *hospital* and the *employee* (see Figure 3).

The attacker starts by sending a message to the hospital:

$$\langle \text{Attacker}, k_{attacker}\rangle \mid \langle \text{Hospital}, k_{hospital}\rangle \to \langle \text{Attacker}, k_{attacker}\rangle \mid \langle P_H, k'_{hospital}\rangle$$

where $P_H = hospital \underset{\text{emailEmployee}}{\rightarrowtail} attacker.\text{Hospital} + \tau.\text{Hospital}$ and where only the knowledge of the hospital changed:

$$k'_{hospital} = \{url = \{\text{urlHospital}\}, message = \{\text{giveEmail}\}, mail = \{\text{emailEmployee}\}\}.$$

At this point the hospital can either do a leak by sending to the attacker the employee's email:

$$\langle \text{Attacker}, k_{attacker}\rangle \mid \langle P_H, k'_{hospital}\rangle \to \langle \text{Attacker}, k'_{attacker}\rangle \mid \langle \text{Hospital}, k'_{hospital}\rangle$$

or not do the leak:

$$\langle \text{Attacker}, k_{attacker}\rangle \mid \langle P_H, k'_{hospital}\rangle \to \langle \text{Attacker}, k_{attacker}\rangle \mid \langle \text{Hospital}, k'_{hospital}\rangle$$

Note that the knowledge of the attacker is changed only in the first case:

$$\begin{aligned} k_{attacker} = \{&url = \{\text{urlHospital}\},\\ &message = \{\text{giveEmail, giveCredential}\},\\ &mail = \{\text{emailEmployee}\}\}. \end{aligned}$$

The attacker can now communicate with the employee, as she has *emailEmployee* in her knowledge.

We conclude this section with a remark: a state of a system that is *syntactically* correct is not necessarily *semantically* correct. For example the of an entity e_1 can contain the send action $e_2 \xrightarrow[v]{c} e_2$, for some protocol c and value v and for some entities e_2, e_3 both distinct from e_1. While such es can execute, it has no meaning in our model. In the implementation, a syntactic check of the system raises errors for semantically incorrect es.

III. Transformation

The *BIP* framework is a component based model introduced in [7]. *BIP* stands for *Behavior*, *Interaction* and *Priority*, that is, the three layers used for the definition of components and their composition in this framework. *BIP* allows the construction of complex, hierarchically structured models from atomic components characterized by their behavior and their interfaces. Such components are transition systems enriched with data. Each time a transition is taken, component data (variables) may be assigned new values, computed by user-defined functions (in C). Atomic components are composed by layered application of interactions and priorities. Interactions express synchronization constraints and data transfer between the interacting components. Priorities are used to filter among possible interactions and to steer system evolution so as to meet performance requirements e.g., to express scheduling policies.

First we briefly recall the key concepts of *BIP* which are further relevant for dealing with security. Then we define a transformation of the IoT modeling language into BIP systems, which can be simulated and analyzed for possible security attacks using statistical model checkers [8].

A. The target language: BIP

Let us give a formal definition of BIP, in particular of atomic components and their composition through multiparty interactions. Priorities are not considered in this work.

Definition 1 (Atomic component)**.** *An atomic component B is a triple (X, P, S) where*

- *X is a set of local variables,*
- *P is a set of ports (or action names). We distinguish respectively input ports $P^{in} \subseteq P$ and output ports $P^{out} \subseteq P$ and we assume they are disjoint, $P^{in} \cap P^{out} =$*

$$
\begin{aligned}
\text{Attacker} =& \mathit{attacker} \xrightarrow[\text{giveEmail}]{\mathit{url}} \mathit{hospital}.\text{Attacker} + \mathit{attacker} \xrightarrow[\text{giveCredential}]{\mathit{mail}} \mathit{employee}.\text{Attacker}+\\
&(\mathit{attacker} \leftarrowtail \mathit{hospital}.\text{Attacker} + \mathit{attacker} \leftarrowtail \mathit{employee}.\text{Attacker})\\
k_{attacker} =& \{\mathit{url} = \{\text{urlHospital}\}, \mathit{message} = \{\text{giveEmail, giveCredential}\}, \mathit{mail} = \emptyset\}\\
\text{Hospital} =& \mathit{hospital} \xleftarrow{\mathit{url}} \mathit{attacker}.(\mathit{hospital} \underset{\text{emailEmployee}}{\rightarrowtail} \mathit{attacker}.\text{Hospital} + \tau.\text{Hospital})\\
k_{hospital} =& \{\mathit{url} = \{\text{urlHospital}\}, \mathit{message} = \emptyset, \mathit{mail} = \{\text{emailEmployee}\}\}\\
\text{Employee} =& \mathit{employee} \xleftarrow{\mathit{mail}} \mathit{attacker}.(\mathit{employee} \underset{\text{emailEmployee}}{\rightarrowtail} \mathit{attacker}.\text{Employee} + \tau.\text{Employee})\\
k_{employee} =& \{\mathit{url} = \emptyset, \mathit{message} = \emptyset, \mathit{mail} = \{\text{emailEmployee}\}\}
\end{aligned}
$$

Fig. 3. An example of a IoT system

$\emptyset$. For every input or output port $p \in P^{in} \cup P^{out}$ we denote by X_p the subset of variables exported and available for interaction through p.

- *$S = (Q, T)$ is a labeled transition system where :*
  - *Q is a finite set of states,*
  - *T is a finite set of labeled transitions where a transition $t \in T$ is a tuple (q, p, g, f, q') such that $q \triangleq src(t), q' \triangleq dst(t) \in Q$ are respectively the source and the target states, $p \triangleq port(t) \in P$ is a port, $g \triangleq guard(t) \in \mathbb{E}xpr[X]$ is the enabling condition and $f \triangleq asgn(t) \in \mathbb{A}sgn[X]$ is the update of t.*

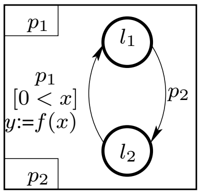

Fig. 4. Atomic Component in *BIP*

Figure 4 provides an example of an atomic component. It contains two states l_1 and l_2 and two ports p_1 and p_2. The transition labeled with p_1 can take place only if the guard $(0 < x)$ is true. When the transition takes place, the variable y is recalculated as some function of x.

Definition 2 (Composite component)**.** *In our model, atomic components have exclusive access on their variables. Interactions between components take place only through explicit input/output binary connectors. An interaction defines a static communication channel from one output port p_1^{out} of a sender component B to an input port p_2^{in} in a receiver component $B' \neq B$. The interaction is denoted Int$_{12}$ = $((p_1^{out}, p_2^{in}), G_{12}, F_{12})$, where G_{12} is a global guard and F_{12} is a global update function. Intuitively, when communication takes place, the value of $var(p^{out})$, is assigned to $var(p^{in})$.*

We denote by $\Gamma(B_1, ..., B_n)$ the composition of a set of atomic components $B_i = (X_i, P_i, (Q_i, T_i))_{i=1,n}$ through a set of interactions Γ. For the sake of simplicity, we tacitly assume that every input and output port of every B_i is used in exactly one connector in Γ. The operational semantics of a composition is defined as a labeled transition system $(\mathcal{Q}, \mathcal{L}, \rightarrow)$ where states correspond to system configurations and transitions to internal steps or communication through interactions. A system configuration $\langle \vec{q}, \vec{V} \rangle$ in $\mathcal{Q}$ where $\vec{q} = (q_1, ..., q_n)$, $\vec{V} = (V_1, ..., V_n)$ is obtained from component configurations (q_i, V_i) where $q_i \in Q_i$ and V_i is a valuation of X_i, for all $i = 1, n$. A label in $\mathcal{L}$ is either a pair of ports or τ, to denote either a communication or an internal action. The set of transitions $\rightarrow \subseteq \mathcal{Q} \times \mathcal{L} \times \mathcal{Q}$ between configurations are defined by the two rules of Figure 5.

The system evolves either by performing asynchronously an internal step of some component B_i (INTER rule) or by performing a synchronous communication between two components B_i, B_j involving respectively ports p_i^{out}, p_j^{in} related by an interaction in Γ (COMM rule). Transitions are executed only if guards are evaluated to *true* in the current configuration. As usual, next configurations are obtained by taking into account variable assignments and communication.

Figure 6 presents a classical *Producer-Buffer-Consumer* example modeled in *BIP*. It consists of three atomic components, namely *Producer*, *Buffer* and *Consumer*. The *Buffer* is a shared memory placeholder, which is accessible by both the *Producer* and the *Consumer*. It holds into the local variable x the number of items available. The *Buffer* interacts with the *Producer* (resp. *Consumer*) on the *put* (resp. *get*) interaction. On the *put* interaction, an item is added to the *Buffer* and x is incremented. On the *get* interaction, the *Consumer* removes an item from the *Buffer*, if at least one exists (the guard $[x \geq 1]$), and x is decremented. Finally, the transitions labeled *produce* and *consume* do not require synchronization - they are executed alone (on singleton port interactions) by their respective components.

B. From IoT to Component-Based Model

With consideration to the IoT system language previously defined in Section II-C, we transform the different entities of our model to a set of atomic components where the communication between them are represented with interactions.

$$\textsc{Inter}\ \frac{(q_i, p_i, g_i, f_i, q_i') \in T_i \quad p_i \notin P_i^{in} \cup P_i^{out} \qquad g_i(V_i) = \textit{true} \quad V_i' = f_i(V_i) \quad \forall k \neq i.\ (q_k', V_k') = (q_k, V_k)}{\langle (q_1, ..., q_n), (V_1, ..., V_n) \rangle \xrightarrow{\tau} \langle (q_1', ..., q_n'), (V_1', ..., V_n') \rangle}$$

$$\textsc{Comm}\ \frac{\begin{array}{c}((p_i^{out}, p_j^{in}), G_{ij}, F_{ij}) \in \Gamma \quad (q_i, p_i^{out}, g_i, f_i, q_i') \in T_i \quad (q_j, p_j^{in}, g_j, f_j, q_j') \in T_j \quad g_i(V_i) = g_j(V_j) = \textit{true} \quad G_i(\{x_{p_i^{out}}, x_{p_j^{in}}\}) = \textit{true} \\ u = V_i(var(p_i^{out})) \quad V_i' = f_i(V_i) \qquad V_j' = f_j(V_j[u/var(p_j^{in})]) \quad \forall k \neq i, j.\ (q_k', V_k') = (q_k, V_k)\end{array}}{\langle (q_1, ..., q_n), (V_1, ..., V_n) \rangle \xrightarrow{p_i^{out} p_j^{in}} \langle (q_1', ..., q_n'), (V_1', ..., V_n') \rangle}$$

Fig. 5. The operational semantics of a BIP system

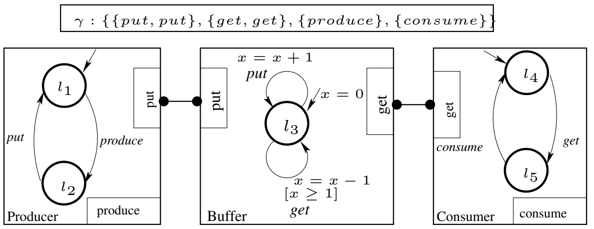

Fig. 6. *BIP* model of the *Producer-Buffer-Consumer* example

In our transformation we use a labeling function on states $\ell : Q \to Process$. To simplify the presentation, we identify states have congruent labels, i.e. $\ell(q_1) \equiv_P \ell(q_2) \iff q_1 = q_2$. We write q_P when $\ell(q) = P$.

We use an auxiliary function in our transformation from an IoT entity to a BIP component, denoted $[\![P]\!]_s \subseteq Q$ that retrieves a set of states from a process:

$$\begin{aligned} [\![a.P]\!]_s &= [\![P]\!]_s \cup \{q_{a.P}\} \\ [\![P_1 + P_2]\!]_s &= [\![P_1]\!]_s \setminus q_{P_1} \cup [\![P_2]\!]_s \setminus q_{P_2} \cup \{q_{P_1+P_2}\} \\ [\![A]\!]_s &= \emptyset \\ [\![0]\!]_s &= \{q_0\}. \end{aligned}$$

Definition 3 (Atomic component)**.** *Let e be an entity in an IoT system with the initial state $s_0 = \langle P, k \rangle$ and $\mathcal{A} = \{A_1, \ldots, A_n\}$ is a set of processes definitions used (recursively) by P. We denote Act the set of actions in P and in def(A_i), for $i \in \{1 \ldots n\}$. We define the transformation of e as the atomic component $B_e = (X_e, P_e, (Q_e, T_e))$ where:*

- *$X_e = \{x_c = K(e, c) | \forall c \in C\}$ is a set of variables where for each protocol c we define a variable x_c initialized to the set of values $K(e, c)$;*
- *$P_e = \{p_a | \forall a \in Act\}$ is a set of ports where each port corresponds to an action in Act and where $P^{out} = \{p_a | a$ is a send or a leak$\}$, $P^{in} = \{p_a | a$ is a receive or a collect$\}$. Moreover, for each $p \in P_e$, $X_p = X_e$.*
- *$Q_e = \{Q_{A_i} | \forall A_i \in \mathcal{A}\} \cup [\![def(A_i)]\!]_s$ is a set of states;*
- *$T_e = \bigcup_{A_i \in \mathcal{A}} [\![def(A_i)]\!]_t$ is a set of transitions where for each definition $A_i \in A$ we define a set of transitions as follows:*

$$\begin{aligned} [\![a.P]\!]_t &= \{(q_{a.P}, a, g, f, q_P)\} \cup [\![P]\!]_t \\ [\![0]\!]_t &= \emptyset \\ [\![a.P_1 + b.P_2]\!]_t &= \{(q_P, a, g, f, q_{P_1}), (q_P, b, g, f, q_{P_2})\} \cup \\ & \quad [\![P_1]\!]_t \cup [\![P_2]\!]_t \ \textit{with}\ P = a.P_1 + b.P_2 \end{aligned}$$

where g is the constant true and f updates the variables in X_e with the values received during an interaction.

In a BIP system each interaction is performed in two steps: a first step where the two communicating entities exchange values and a second step where each entity updates locally its variables. The second step is usually modeled using a local update function f. Using this trick the second step is hidden from the semantics.

Communications between two entities e_1 and e_2 in the IoT language can either perform a *SendReceive* or a *LeakCollect* communication through a certain protocol c. In the generated BIP model, these communications are transformed into a set of guarded interactions between components B_{e_1} and B_{e_2} corresponding respectively to e_1 and e_2.

Definition 4 (Interactions)**.** *For each action $a \in Act$ of type send or leak if there exists $a' \in Act$ such that:*

- *if $a = e_1 \xrightarrow[v']{c} e_2$ then $a' = e_2 \xleftarrow{c} e_1$,*
- *if $a = e_1 \underset{v'}{\rightarrowtail} e_2$ then $a' = e_2 \leftarrowtail e_1$*

then we define an interaction $Int_{aa'}$=$(P_{aa'}, G_{aa'}, F_{aa'})$ where $G_{aa'}$ is a guard, $F_{aa'}$ is an update function and $P_{aa'}$ is a pair of ports as follows

- *$P_{aa'} = \{p_a, p_{a'}\}$;*
- *if $a = e_1 \xrightarrow[v']{c} e_2$ then $G_{aa'} = (\exists v_1 \in V_1(x_c^1)$ and $v_2 \in V_2(x_c^2)$ such that $v_1 = v_2)$, otherwise $G_{aa'} = true$;*
- *$F_{aa'}(x_{c'}^2) = x_{c'}^2 \cup \{v'\}$ updates $x_{c'}^2$, with $\mathsf{protocol}(v') = c'$*

and where $x_c^1 \in X_{e_1}$, $x_c^2 \in X_{e_2}$ are the variables of e_1 and e_2 respectively, associated to the protocol c.

Note that interactions are only defined for consistent pairs of *SendReceive* and *LeakCollect*. Interactions reflect then the rules SENDRECEIVE and LEAKCOLLECT, respectively from the operational semantics of Figure 2.

During interactions in an IoT systems, values are added to the knowledge of the entities. In the BIP system the knowledge of different components is stored as sets of values. For each protocol the transformation introduces a variable of type set. Ports export all variables of a component, which can be exchanged or checked during an interaction. It is the role of the global guards and functions to use the pertinent variables for exchange and for checking global guards.

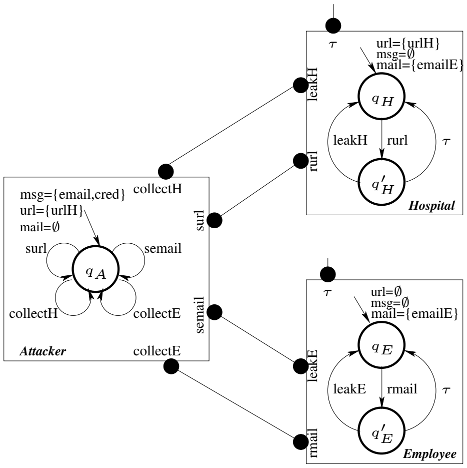

Fig. 7. Transformation of the Smart-Hospital example

Example 2. In Figure 7 we give the transformation of the Smart-Hospital example presented as a running example. For the initially defined entities, we generate three atomic components that are $Attacker$, $Employee$ and $Hospital$. For each defined sequence of processes, we recursively create a state, where for instance, we define q_H for the $Hospitol$ process and q'_H for the process ($hospital \underset{\text{emailEmployee}}{\rightarrowtail} attacker$.Hospital $+$ τ.Hospital) . Each atomic component contains three variables: *url*, *msg* and *mail*. The interaction *surl-rurl* can occur between the $Attacker$ and the $Hospital$ components since the value *urlH* is contained in both components. The attacker can collect the *Email* value from the $Hospital$ component since there is no restriction on the execution of the *LeakH-CollectH*.

C. Correctness of the transformation

In this section we show that there exists an operational correspondence between an IoT system and its transformation in a BIP system. Formally, we state this in Theorem 1. First we need an intermediate lemma.

Lemma 1. Any two congruent IoT states have the same transformation in BIP systems.

Theorem 1. Let S=$e_1 \mid \cdots \mid e_n$ be an IoT system and its transformation B=$\Gamma(B_{e_1}, \cdots B_{e_n})$. There exists a bisimulation between s and B.

Proof sketch. We define a relation $\mathcal{R}$ between the global states s=$\langle s_1, K_1\rangle | \ldots | \langle s_n, K_n\rangle$ of S and a state q=$((q_{p_1}, \ldots, q_{p_n}), (V_1, \ldots, V_n))$ of B as follows: $\mathcal{R}$=$\Big\{(s, q)$ where s=$\langle P_1, k_1\rangle \ |\cdots|\ \langle P_n, k_n\rangle$ and $\langle (q_{P_1}, \cdots, q_{P_n}), (V_1, \cdots, V_n)\rangle$ are respectively the current states of S and B, where V_i is the transformation of k_i, for all $i \in \{1, \cdots, n\}\Big\}$.

We prove that $\mathcal{R}$ is a bisimulation. Due to lack of space, we show here only one direction: if $s \to s'$ then B can also do a transition from its current state q to another q' such that $(s', q') \in \mathcal{R}$.

We consider that an IoT system executes an interaction $SendReceive$ or $LeakCollect$ between the two entities e_1 and e_2 where

$$s = \langle a_1.P_1 + Q_1, K_1\rangle | \langle a_2.P_2 + Q_2, K_2\rangle | \cdots \to$$
$$s' = \langle P_1, K'_1\rangle | \langle P_2, K'_2\rangle | \ldots$$

Note that we use Lemma 1 to rewrite s as above. From the encoding in Definition 3 we have that in the transition system of B_{e_1} there exists the states $q_{a_1.P_1+Q_1}$, q_{P_1} and a transition between them labeled by the action a_1. Similarly, for the component B_{e_2} and the states $q_{a_2.P_2+Q_2}$, q_{P_2} with a transition labeled a_2. Moreover B_{e_1}, B_{e_2} have two ports p_{a_1} and p_{a_2}, respectively. Let us suppose w.l.o.g. that a_1 is the send or leak and a_2 is the receive and collect, respectively. From the Definition 4, we have that $Int_{a_1a_2}$=$(\{p_{a_1}, p_{a_2}\}, G_{a_1a_2}, F_{a_1a_2}) \in \Gamma$ with a guard $G_{a_1a_2}$ which is either always true, (for the $LeakCollect$ case) or $G_{a_1a_2} = (\exists v_{1c} \in V_1(x_c)$ and $v_{2c} \in V_2(x_c)$ where $v_{1c} = v_{2c})$ (in the case of $SendReceive$). In the latter, the guard holds by the hypothesis of the rule SendReceive and from the correspondence between K_i and V_i.

Then the atomic component B will also execute an interaction

$$q = ((q_{a_1.P_1+Q_1}, q_{a_2.P_2+Q_2}, \ldots), (V_1, V_2, \ldots))$$
$$\xrightarrow{(p_{a_1}, p_{a_2})} q' = ((q_{P_1}, q_{P_2}, \ldots), (V'_1, V'_2, \ldots))$$

using rule Comm. Remains to show that V_i is updated in the same manner as the knowledge function k_i, for $i \in \{1, 2\}$. It follows trivially from the rules SendReceive, LeakCollect and from the Definition 4. We have that s' and q' are in $\mathcal{R}$.

The case where the entity e_1 of the IoT system does a τ transition as well the case where the IoT system has to simulate its corresponding BIP system is obtainable in the Technical Report, available to download at https://gitlab.inria.fr/IOTLanguage/IoTCompilertoBIP.

□

IV. Implementation

A. Tool

The presented IoT language as well as the transformation to component based model, described in both Sections II and III, are implemented and available to download at https://gitlab.inria.fr/IOTLanguage/IoTCompilertoBIP. The tool implementation is constituted of a parser that carries out the lexical and syntactic analysis of the IoT system model and of a transformation algorithm for the conversion towards the BIP model. The grammar of the IoT language is defined using extended Backus-Naur form (EBNF) language from which we generate a Lexer and a Parser. Finally, BIP code is generated as defined in Section III. The tool work-flow is presented in Figure 8.

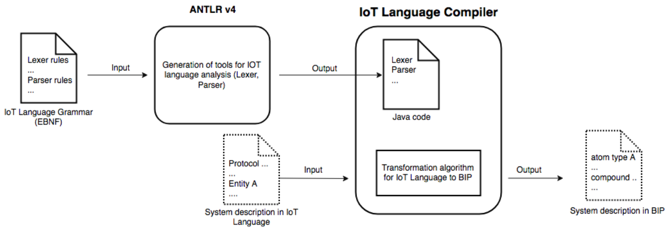
ANTLR v4
IoT Language Compiler
Lexer rules
...
Parser rules
...
IoT Language Grammar (EBNF)
Input
Generation of tools for IOT language analysis (Lexer, Parser)
Output
Lexer
Parser
...
Java code
Protocol ...
...
Entity A
....
System description in IoT Language
Input
Transformation algorithm for IoT Language to BIP
Output
atom type A
...
compound ..
...
System description in BIP

Fig. 8. Implementation of the IoTtoBIP compiler

B. Use-Case

To illustrate the use of our approach and tool we consider the example of a phishing attack to get access to a Smart Hospital infrastructure. The system model is composed of two internal actors: an *Employee* and a *Hospital IT infrastructure*. External actors which are an *Attacker* and a malicious *Malware*, can interact with the internal actors in the goal to access confidential information. We instrument our models so that we detect whenever a confidential value (i.e. the secret) becomes known to the attacker, that is, it becomes part of its knowledge set.

In the first case, the *Attacker* is able to retrieve the secret value. Indeed, the first system represents a successful attack where an employee gets his device infected by the *Malware* after downloading a malicious program sent by the *Attacker* as an attached file on an email. Once it is installed on the victim device, the *Malware* sends a confirmation to the *Attacker*. Hence, the *Attacker* can communicate with it via a ssh protocol. First, the *Malware* sets up a backdoor, which allows the *Attacker* to receive and send files to a remote location. Then, it installs additional software to establish command and have control over the device. Using the *Malware*, the *Attacker* can now go through the system, send requests to connected devices and look for confidential data. Figure 9 illustrates the described successful attack.

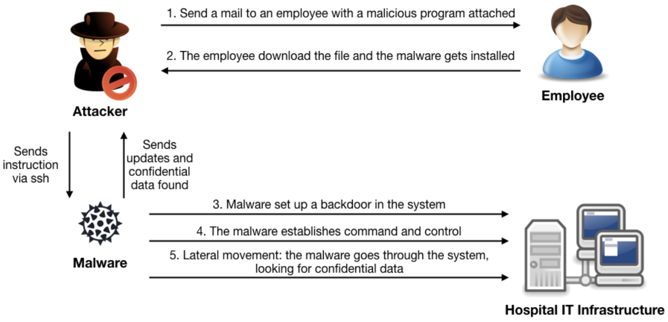
1. Send a mail to an employee with a malicious program attached
2. The employee download the file and the malware gets installed
Attacker
Employee
Sends instruction via ssh
Sends updates and confidential data found
3. Malware set up a backdoor in the system
4. The malware establishes command and control
5. Lateral movement: the malware goes through the system, looking for confidential data
Malware
Hospital IT Infrastructure

Fig. 9. System 1 : Successful attack

In the second case, no trace in the system representing a serious threat is found. The *Attacker* tries to synchronize with the system by sending an infected email but the employee doesn't answer the email and then the system is not infiltrated. The main benefit from using our modeling approach is the evaluation of each entity's knowledge at each execution step in the trace. Such evaluation can be more invested with quantitative analysis of the system to calculate the distance that separates an attacker to reach the main goad, which is access the patient confidential information.

V. Related Work

In this section we present other approaches proposed in the literature for modeling and analyzing the security of IoT systems. We focus (i) on whether the modeling language is adapted (expressive, robust, unambiguous) for IoT systems and their security failures, and (ii) on the analysis tools proposed to guarantee security properties on models of IoT systems.

MAPL (Multi-Attribute Prediction Language) [9] aims to model and analyze *System of Systems*. It focuses on quality analysis defined by the availability and data accuracy of the involved software. However this language fall short to properly handle security aspects and system heterogeneity. A more adopted and popular security oriented modeling language for IoT system architecture is CySeMol [10]. With this language users can model a system's architecture on an enterprise level. A model in CySeMol, called a *Probabilistic Relational Model* is equipped with probabilities. CySeMol's analysis is therefore able to give the probabilities of success for different attacks. Entities in this language are *software instances* modeled in UML that can be of different types and that can communicate. Protocols and access control primitives are supervising these exchanges. CySeMol has been developed with the objective of improving decision making in the industrial field. CySeMol's modeling language is UML which is low level and has an ad-hoc semantics. Compared to CySeMol, we present a sound and robust modeling language with a formally proved transformation to a component based model where we can run analysis of security properties[11] and simulate attacks. The proposed model handles both heterogeneity of systems and probabilistic aspects [8]. Another similar approach was proposed for WAMC (Wide Area M Control) systems [12]. The aim is to detect failures in the interoperability and cyber-security aspects of WAMC systems. Their language is UML-based and can model various components of the system such as WAMC components, data flows, protocols or network zones through which the communication goes. The cyber security aspect of this model lies in the integrity of the data flow. Attacks and countermeasures are represented as attributes for each component and a successful attack results from the success of a series of different attacks. As in the previous approaches, the model is equipped with probabilities which allows an analysis of security properties on a model's simulations. Our approach is similar, but more high level and applicable to more general IoT systems, not just to WAMC.

VI. Conclusion

In this paper we proposed a formal language for modeling IoT systems. Our motivation is to model security failures that arise from a sequence of ”bad” decisions and interactions within a distributed system (such as the security failures described by attack trees).

As future work, we plan to extend the IoT language with probabilities over the system's actions. Higher sophisticated attackers can infiltrate into the systems by combining several atomic steps. In a well designed system, the success of such attacks is considered a *rare event*. Hence, we will develop a statistical quantitative analysis approach of our model using an adequate statistical model checking, based on *importance splitting* [13] to automate the detection of such failures.

Another direction for future work is to extend the expressiveness of the IoT language. For instance protocols can perform more complicated checks.

References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017.

[2] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things," *Future Gener. Comput. Syst.*, vol. 78, 2018.

[3] "Anatomy of an attack, medjack (medical device attack)," Tech. Rep., 05 2015.

[4] B. Schneier, *Secrets & Lies: Digital Security in a Networked World*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[5] R. Milner, *A Calculus of Communicating Systems*, 1982.

[6] "Reactive, generative, and stratified models of probabilistic processes," *Information and Computation*, vol. 121.

[7] A. Basu, M. Bozga, and J. Sifakis, "Modeling heterogeneous real-time components in BIP," in *Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM 2006), 11-15 September 2006, Pune, India*.

[8] S. Bensalem, M. Bozga, B. Delahaye, C. Jegourel, A. Legay, and A. Nouri, "Statistical model checking qos properties of systems with sbip," in *Proceedings of the 5th International Conference on Leveraging Applications of Formal Methods, Verification and Validation: Technologies for Mastering Change - Volume Part I*.

[9] P. Johnson, R. Lagerstrom, M. Ekstedt, and U. Franke, "Modeling and analyzing systems-of-systems in the multi-attribute prediction language (mapl)," in *2016 IEEE/ACM 4th International Workshop on Software Engineering for Systems-of-Systems (SESoS)*, May 2016, pp. 1–7.

[10] T. Sommestad, M. Ekstedt, and H. Holm, "The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures," *IEEE Systems Journal*, vol. 7, no. 3, pp. 363–373, Sept 2013.

[11] N. Ben Said, T. Abdellatif, S. Bensalem, and M. Bozga, "Model-driven Information Flow Security for Component-Based Systems," in *From Programs to Systems. The Systems perspective in Computing - ETAPS Workshop, FPS 2014*, 2014, pp. 1–20.

[12] H. Holm, K. Shahzad, M. Buschle, and M. Ekstedt, "P^2 cysemol: Predictive, probabilistic cyber security modeling language," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 6, pp. 626–639, Nov 2015.

[13] C. Jégourel, A. Legay, S. Sedwards, and L. Traonouez, "Distributed verification of rare properties with lightweight importance splitting observers," *CoRR*, vol. abs/1502.01838, 2015.