On Service Security Analysis for Event Log of IoT System Based on Data Petri Net

Mohd Anuaruddin Bin Ahmadon, Shingo Yamaguchi
Graduate School of Sciences and Technology
for Innovation, Yamaguchi University,
2-16-1 Tokiwadai, Ube, 755-8611, Japan
Email: {anuar,shingo}@yamaguchi-u.ac.jp

Sharifah Saon, Abd Kadir Mahamad
Faculty of Electrical and Electronic Engineering
Universiti Tun Hussein Onn Malaysia
86400 Parit Raja, Batu Pahat, Johor, Malaysia
Email: {kadir,sharifa}@uthm.edu.my

***Abstract*—The Internet of Things (IoT) has bridged our physical world to the cyber world which allows us to achieve our desired lifestyle. However, service security is an essential part to ensure that the designed service is not compromised. In this paper, we proposed a security analysis for IoT services. We focus on the context of detecting malicious operation from an event log of the designed IoT services. We utilized Petri nets with data to model IoT service which is logically correct. Then, we check the trace from an event log by tracking the captured process and data. Finally, we illustrated the approach with a smart home service and showed the effectiveness of our approach.**

I. Introduction

The Internet of Things (IoT) enabled us to move towards the super smart society transformation where every need for individuals are met. Our life will become easier and open to the automated services around us. IoT systems constructed by cyber-physical systems are now becoming more rapidly deployable with enabling technologies such as service orchestration [1]. However, the system exposure to possible device malfunctions or cyber threats is undeniable. Concretely, the threats may be caused by false reading or bad response from devices and possible cyber attack on physical devices.

Most works related to the security of IoT systems focus on the network level. Singh et al. proposed a secure version of device to device (D2D) protocol based on the popular MQTT protocol [2]. The secure D2D protocol is expected to protect the communication but previous incidents have proved that IoT devices are still prone to malfunction and false response. Therefore, some previous works had given attention on modeling and analyzing the security of cyber-physical systems. We [3] proposed an intrusion detection system (IDS) framework that fuses the model of known attacks into one integrated model based on Petri net. Wang et al. [4] proposed a formal analysis method of security properties for cyber-physical system which focuses on the physical processes based on timed-automata. Howser et al. [5] also proposed an approach using information-flow methods to analyze the security of cyber-physical systems to ensure the confidentiality and integrity of the system. The given approaches focus on the detection of known malicious activity flow but do not mainly consider data flow in their modeling and analysis technique. Data tracking is important to ensure that the handled data state is within the authorized boundaries in case of failure of preserving the correct operation after implementation of IoT systems in the real world. External threats such as cyber attack,

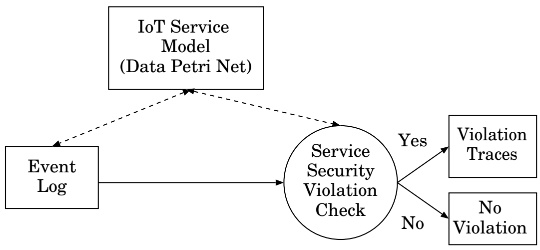

Fig. 1: Overview of our approach.

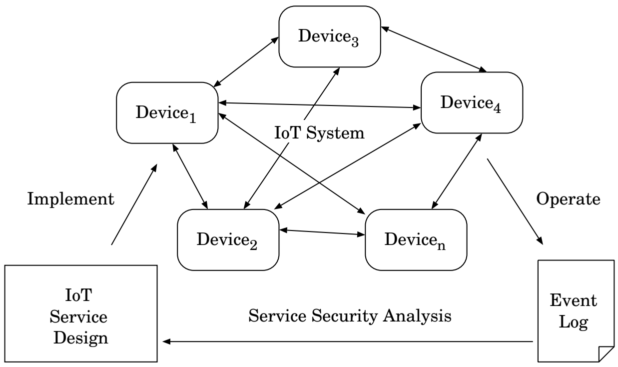

Fig. 2: An overview of IoT service framework.

software bug or hardware malfunction can lead to unintended illegal operation.

In this paper, we propose a security analysis for the IoT services. We focus on the context of detecting malicious operation from an event log of the designed services (See Fig. 1). We utilized a control and data flow graph called as Petri nets with data to model IoT service. It allows us to track both process and data of the modeled system. Figure 2 shows a framework of IoT service design, implementation, and security analysis. An IoT service model implemented by interconnected IoT devices will operate and record all activities into the event log. We can check the trace from an event log by checking the firing sequences and the data state. Finally, we illustrated the approach with a smart home service and showed its effectiveness. After Sect. 1, we give the definition of Petri nets with data in Sect. 2. In Sect. 3, we show the modeling techniques with Petri nets with data. In Sect. 4, we give our service security analysis approach. In Sect. 5 we give the example and evaluation. Finally, we give the conclusion and future work in Sect. 6.

II. Preliminary

a) Petri Net: A Petri net [6] is a four tuple $N{=}(P,T,F,\ell)$, where P, T, and A $(\subseteq(P{\times}T)\cup(T{\times}P))$ are finite sets of *places*, *transitions*, and *arcs*, respectively. Let x be a place or transition. $\overset{N}{\bullet}x$ and $x\overset{N}{\bullet}$ denote $\{y|(y,x){\in}F\}$ and $\{y|(x,y){\in}F\}$, respectively. They are also extended to a subset X of P or T: $\overset{N}{\bullet}X{=}\bigcup_{x\in X}\overset{N}{\bullet}x$, $X\overset{N}{\bullet}{=}\bigcup_{x\in X}x\overset{N}{\bullet}$. ℓ is a labelling function of transitions. A marking (state) is a multiset of its place of i.e. $M{:}P{\to}\mathbb{N}$, which is denoted by $M{=}[p^{M(p)}|p{\in}P, M(p){>}0]$. Let M_X and M_Y be markings. $M_X{=}M_Y$ denotes that $\forall p{\in}P : M_X(p){=}M_Y(p)$. $M_X{\geq}M_Y$ denotes that $\forall p{\in}P : M_X(p){\geq}M_Y(p)$. A transition t is said to be firable in a marking M if $M{\geq}\overset{N}{\bullet}t$. This is denoted by $M[N,t\rangle$. Firing t in M results in a new marking M' $({=}M{\cup}t\overset{N}{\bullet}{-}\overset{N}{\bullet}t)$ denoted by $M[N,t\rangle M'$. A marking M_n is said to be *reachable* from a marking M_0 if there exists a transition sequence σ $({=}t_1t_2\cdots t_n)$ such that $M_0[N,t_1\rangle M_1[N,t_2\rangle M_2\cdots[N,t_n\rangle M_n$. This is denoted by $M_0[N,\sigma\rangle M_n$ or simply $M_0[N,*\rangle M_n$. σ is called a firing sequence which transforms M_0 to M_n. The set of all possible markings reachable from M_0 in (N,M_0) is denoted by $R(N,M_0)$.

A labeled Petri net $N{=}(P,T,F,\ell)$ is a WF-net [8] iff (i) N has a single source place p_I ($\overset{N}{\bullet}p_I = \emptyset$ and $\forall p \in (P\backslash\{p_I\})$: $\overset{N}{\bullet}p \neq \emptyset$) and a single sink place p_O ($p_O\overset{N}{\bullet} = \emptyset$ and $\forall p \in (P - \{p_O\})$: $p\overset{N}{\bullet} \neq \emptyset$); and (ii) every node is on a path from p_I to p_O.

b) Petri Net with data: A Petri net with data or Data Petri Net (DPN for short) [7] is a 6-tuples $DPN = (N,V,U,R,W,G)$. It consists of a Petri net $N{=}(P,T,F,\ell)$, a set V of variables, a function U that defines the values admissible for each variable $v \in V$, a read function R that labels each transition with the set of variables that it reads, a write function W that labels each transition with the set of variables that it writes, a guard function G that denotes a guard with each transitions. A transition $t \in T$ is enabled if its guard $G(t)$ returns true. If t has no guard then $G(t){=}true$. We define a subclass of DPN called as Data Workflow Net (DWF-net for short). A DPN is a DWF-net if N is a workflow net [8] with single input place p_I and single output place p_O.

A trace of transition bindings execution sequences of DPN is denoted by a pair (t,ϕ) where $t \in T$ and ϕ is value assignment function for some variables $v \in V$, i.e $v^w{=}x, v^r{=}y$ where $v^w{=}x$ ($v^r{=}y$) denotes writing (reading) value x (y) to (from) variable $v \in V$. We can write the trace such as $\sigma{=}\langle t_1\{\phi_1\}, t_2\{\phi_2\}, \cdots, t_n\{\phi_n\}\rangle$. A valid firing of (t,ϕ) satisfies state transition rule in Def. 4 of Ref. [7].

The marking (state) of DPN is denoted by a pair (M,A) and let $D{=}\bigcup_{v\in V}U(v)$. M is the marking of Petri net (P,T,F,ℓ). The function A assigns a value to each variable, i.e. $A : V \to D{\cup}\{\perp\}$, with $A(v) \in U(v){\cup}\{\perp\}$. We use a symbol $\perp$ if no value is assigned, i.e $A(v){=}\perp$. The initial marking is denoted by (M_0,A_0) where $M_0{=}[p_I], \forall v \in V : A_0(v){=}\perp$. The final marking is denoted by (M_{final},A_{final}) where $M_{final}{=}[p_O]$ and A_{final} is the latest assignment of value $v{\in}V$ after reaching M_{final}.

A firing in DPN is denoted by transition binding (t,r,w) where $t \in T$, $r \subseteq V$ is the set of variables that are read and $w : w \not\subseteq V$ is the set of variables that are written with the respective values. A transition firing (t,r,w) in the state M_A is said to be valid if (i) the input places has at least one token such that i.e. $\forall p \in \overset{DPN}{\bullet}t{:}M(p){>}0$; (ii) the transition reads and writes all and only the variables that was prescribed to it, i.e. $r{=}R(t)$ and $dom(w){=}W(t)$[1]; (iii) the value assigned to each variable is valid, i.e $\forall v \in dom(w).\ w(v) \in U(v)$; (iii) the guard $G(t)$ returns true with respect to the assignment A of values to process variables.

A valid firing (t,r,w) in state (M',A'), where

$$M'(p) = \begin{cases} M(p)-1 & if \quad p \in \overset{DPN}{\bullet}t\backslash t\overset{DPN}{\bullet} \\ M(p)+1 & if \quad p \in t\overset{DPN}{\bullet}\backslash\overset{DPN}{\bullet}t \\ M(p) & otherwise \end{cases}$$

and

$$A'(v) = \begin{cases} A(v)-1 & if \quad v \in V\backslash W(t) \\ w(v) & if \quad v \in W(t) \end{cases}$$

$\mathcal{L}(DPN,[p_I],A_0)$ denotes the set of all possible traces σ that transforms initial marking $[p_I]$ to final marking $[p_0]$ where N is a WF-net.

III. IoT Service Model

We utilize DPN to design our IoT service model. It acts as the specification of the control flow and the data flow of a service.

A. Service Modeling with Data Petri Net

Using DPN we can describe the control flow with classic Petri net and the data flow by adding additional variables node and read/write arc. The process and data can be captured at the same time once an action is executed. We can also separate the control flow and data flow by omitting the variable nodes and read/write arc so that existing Petri net analysis techniques can be applied.

We give an example of a DWF-net DPN_X that represents an IoT service as shown in Fig. 3. It describes a window control service based on temperature and humidity condition. The window will open when temperature is between 20°C and 28°C and humidity is between 40% and 65%. We give the components of DPN_X as transitions $T_x{=}\{getTemp, getHumid, openWindow\}$, variables $V_X{=}\{v_1,v_2\}$ and guard $G(openWindow){=}(20{\leq}v_1{\leq}28) \wedge (0.4{\leq}v_2{\leq}0.65)$. For transitions $getTemp$ and $getHumid$, we set no guards so any value is admissible.

As another example, DPN_Y shown in Fig. 4 describes a refrigerator food stock management service. First, the expiry date and food stock in the refrigerator is checked. If the stock is less than one or the expiry date is near or expired, the refrigerator will automatically order the ingredient and notify the user. We give the transitions, variables and guard function of DPN_Y as $T_Y = \{orderFood, checkStock, checkExpiry, notifyUser\}$, $V_Y = \{v_1,v_2\}$ and $G(orderFood) = (v_1^r{=}true) \vee (v_2^r{\leq}1)$.

We must ensure that the design of model satisfies logical correctness called as soundness. The property of soundness ensures a proper termination. We can say that DPN_X and DPN_Y is sound. We give the definition as follows:

[1] $dom(w)$ denotes the domain of function w.

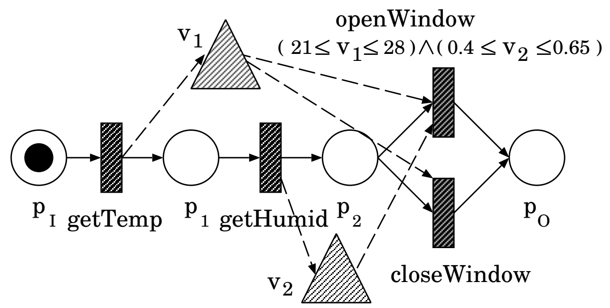

Fig. 3: An instance (DPN_X, M_0, A_0).

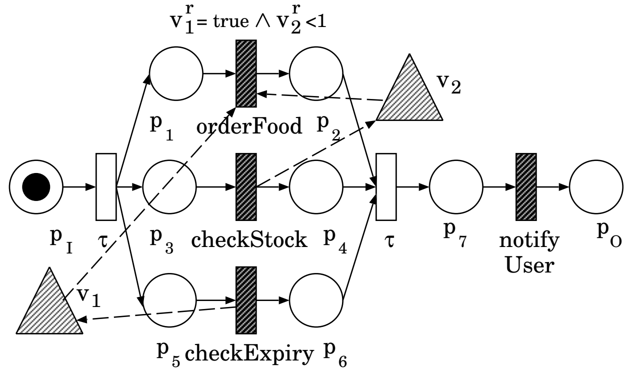

Fig. 4: An instance (DPN_Y, M_0, A_0).

Property 1 (Soundness of Data Workflow Net): A DWF-net (DPN, M_0, A_0) is said to be sound iff

- (i) DPN will terminates eventually, i.e. $\forall(M, A) \in R(DPN, M_0, A_0){:}\exists(M', A'){\in}R(DPN, M, A){:}M'{\geq}[p_O]$;
- (ii) After termination there is no token in other places i.e. $\forall(M, A) \in R(DPN, M_0, A_0) : M \geq [p_O] \Rightarrow M{=}[p_O]$;
- (iii) There is no dead transition in (DPN, M_0, A_0). ■

The net DPN_X and DPN_Y are both sound DWF-nets.

IV. Service Security Analysis

Once we designed a service model with DPN, the captured event log will be verified. We propose an approach to verify the trace of the event logs. We take example of a service DPN_X. We check a trace σ^X of event log E_X. We obtained the trace:

$\sigma^X{=}\langle getTemp\{v_1^w{=}15\}, getHumid\{v_2^w{=}0.5\}, openWindow\{v_1^r{=}25, v_2^r{=}0.5\}\rangle$.

Let us execute each transition binding (t, ϕ). There is no problem for σ^X in terms of valid firing and reaching the final marking. However, let us confirm the variables of σ^X. Variable v_1 is written with 15 when *getTemp* is fired, but v_1 was read with 25 when *openWindow* is fired. *openWindow* should not fire if v_1 was previously written with a value that does not satisfy guard $G(openWindow)$. This shows us that an anomaly exists in E_X.

Next, we take example of a service DPN_Y. We check a trace σ^Y of event log E_Y. We obtained the trace:

$\sigma^Y{=}\langle checkExpiry\{v_1^w{=}true\}, orderFood\{v_1^r{=}true, v_2^w{=}0\}, checkStock\{v_2^r{=}3\}\rangle$.

Let us execute each transition binding (t, ϕ). There is no problem for σ^X in terms of valid firing and reaching the final marking when not concerning the data variables v_1 and v_2 because the process flow for each *checkExpiry*, *orderFood* and *checkStock* can be executed in parallel (the firing can be in any sequence). However, let us confirm the data of σ^Y. Variable v_1 is written with *true* when *checkExpiry* is fired. Then, *orderFood* is fired because the read value of $v_1{=}true$ and $v_2{=}3$ that satisfy the guard $G(orderFood)$. However, the variable v_2 should be written after *checkStock* is fired. In the trace, *checkStock* writes $v_2^w{=}0$. *orderFood* should be fired after *checkStock* and the value should be read with $v_2{=}0$ instead of $v_2{=}3$. This shows us that without concerning the data value, we cannot identify the anomaly that exists in E_Y.

From the situation given on DPN_X and DPN_Y, although we can ensure that the design is logically correct, the system is not guaranteed to be secured during implementation and operation of the IoT systems. This is because we can expect external factors such as malfunction of devices, software bug and cyber-attack. The best way is to detect the anomaly that exists in the system by monitoring the event log and address the problem to the system administrator. We give the definition of anomaly called as violation trace in an event log E:

Definition 1 (Valid Trace): A trace σ is said to be valid for a DWF-net DPN if $\sigma \in \mathcal{L}(DPN, [p_I], A_0)$ ■

We propose a security violation check method by searching the event log for invalid traces. Therefore, we formalize the following problem:

Definition 2 (Service Security Violation Problem):
Input: Service Model (DPN, M_0, A_0), Event log E
Output: Does E of (DPN, M_0, A_0) contain any invalid trace σ?

In order to check any invalid traces, we need to verify each traces in the event log. Therefore, we can deduce the following theorem to check for valid traces:

Theorem 1 (Valid Trace): For a trace σ of an event log E where $\sigma \in E$, σ is valid for a DWF-net DPN iff:

- (i) All firing of (t, ϕ) in σ is a valid firing based on Def. 4 in Ref. [7].
- (ii) σ transforms (M_0, A_0) to (M_{final}, A_{final}) [2] such that $(DPN, M_0, A_0)[\sigma\rangle(DPN, M_{final}, A_{final})$.
- (iii) Each (t, ϕ) in σ satisfies the state transition, i.e. $\forall v{\in}V{:}A'(v){=}A(v)$ where $v{\in}W(a){\cap}R(b), \{a, b\}{\in}T$, $(M, A)[a\sigma' b\rangle(M', A'){:}a\sigma' b{\in}\sigma$. ■

The verification is trivial by only executing all (t, ϕ) sequences. The captured event log can be compared with the specification of the service model by executing the traces on the service model. The condition of Theorem 1 can be verified with our procedure. Therefore, we propose the following procedure :

≪Service Security Violation Check≫

Input: Service Model (DPN, M_0, A_0), Event log E
Output: Trace σ for (DPN, M_0, A_0).

[2] (M_{final}, A_{final}) is the final state where $M_{final}{=}[p_o]$ and A_{final} is the latest variable state after reaching M_{final}

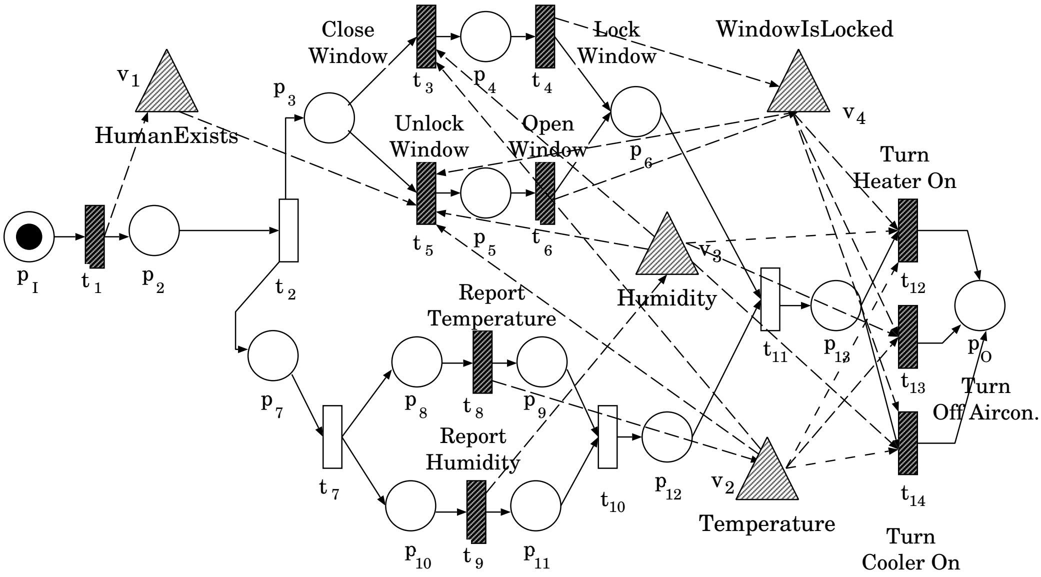

Fig. 5: An example of a smart home service model DPN_Z.

1° For all $\sigma \in E$, check Condition (i), (ii) and (iii) of Theorem 1.
  a) Check if each $(t, \phi) \in \sigma$ is valid based on Def. 4 in Ref. [7]. If no output σ and stop.
  b) Check if $[p_I]$ transforms to $[p_O]$. If no output σ and stop.
  c) Check if for each (t, ϕ) satisfies the state transition, i.e. $(DPN, M, A)[(a, \phi_a)\rangle(DPN, M', A')$ then $\exists(DPN, M'', A'')[(b, \phi_b)\rangle(DPN, M''', A''')$, where $A(v){=}A'''(v)$ such that $v \in \phi_a \cap \phi_b$. If no output σ and stop.

2° Output no and stop.

The given procedure simply checks the firing of the traces one-by-one. The traces are verified by first checking whether each trace enables valid firing or not. If yes, then we need to confirm they can transform the initial marking to the final marking. Then, we check whether the data wrote on the previously fired transition can be read with the same value when another transition is executed. Finally, we output the result.

V. Application Example

In this section, we will show that we can verify two types of anomaly in the event log: (i) invalid data transition and (ii) wrong execution sequence. We illustrate the approach with a real-world model DPN_Z given in Fig. 5 describes another part of a real-world smart home service. The services show a climate control service of the smart house. First, the smart house will detect a human presence. Then, if a human is in the house the temperature and humidity sensor will read the in-room temperature and humidity. The window will be unlocked and opened if a human exists and temperature more than 28° Celcius and humidity value is less than 40 percent. In contrast, the window will be closed and locked if the temperature is lower than 16° Celcius. Then, if the temperature is cold (less than 16° Celcius) and the window is closed, the heater will be turned on. If the temperature is high (more than 28° Celcius) the cooler will be turned on. If the temperature is suitable to open the window then the air conditioner will be turned off. Table I shows the list of variables and their ranges and Table II shows the list of transitions assigned with guards of DPN_Z.

TABLE I: List of variables.

Variables	Range of Data
$v_1(HumanExist)$	$true, false$
$v_2(Temperature)$	$-30 \leq temp \leq 100$
$v_3(Humidity)$	$0 \leq humid \leq 100$
$v_4(WindowIsLocked)$	$true, false$

TABLE II: List of transitions and guard functions.

Transition	Guard
t_1	$v_1^w{=}true$
t_2	$v_1^r{=}true$
t_3	$v_2^r{\leq}16 \wedge v_3^r{\leq}50$
t_4	$v_4^w{=}true$
t_5	$v_1^r{=}true \wedge v_4^r{=}true \wedge (28{<}v_2^r \vee v_2^r{\leq}16) \wedge (40{\leq}v_3^r)$
t_6	$v_4^w{=}false$
t_8	$v_2^w{=}\phi$
t_9	$v_3^w{=}\phi$
t_{12}	$v_4^r{=}true \wedge v_2^r{\leq}16 \wedge v_3^r{\leq}50$
t_{13}	$(v_4^r{=}false \wedge 17{\leq}v_2^r{\leq}28 \wedge 30{\leq}v_3^r{\leq}50$
t_{14}	$v_4^r{=}true \wedge v_2^r{\geq}29 \wedge v_3^r{\geq}50$

We take example of a service DPN_Z. We check some traces from the event log E_Z. To show the application example we give both bad and good examples. Let us say we obtained the following traces:

$$\sigma^{Z_1}{=}\langle t_1\{v_1^w{=}true\}, t_8\{v_2^w{=}30\}, t_5\{v_1^r{=}true, v_2^r{=}30, v_4^w{=}40\}, t_6\{v_4^w{=}false\}, t_{13}\{v_2^r{=}30, v_3^r{=}40, v_4^r{=}false\}\rangle$$

$$\sigma^{Z_2}{=}\langle t_1\{v_1^r{=}false\}, t_8\{v_2^w{=}30\}, t_9\{v_3^w{=}40\}, t_5\{v_1^r{=}true, v_4^r{=}true, v_2^r{=}30, v_3^r{=}40\}, t_6\{v_4^r{=}false\}\rangle$$

$\sigma^{Z_3}=\langle t_1\{v_1^w=true\}, t_3\{v_2^w=30, v_3^w=40\}, t_4\{v_4^r=false\}, t_8\{v_2^w=30\}, t_9\{v_3^w=40\}\rangle$

First, we check σ^{Z_1}. We apply ≪Service Security Violation Check≫ on DPN_Z. The input is (DPN_Z, M_0, A_0) and trace σ^{Z_1}. In Step 1°, we check condition (i), (ii) and (iii) of Theorem 1. In 1°(a), we check if each (t, ϕ) in σ^{Z_1} is valid based on Def. 4 in Ref. [7]. Each firing transforms the marking (M, A) to (M', A') where there is no guard violations. Therefore, all firing in σ^{Z_1} is valid. In Step 1°(b), the trace allows a proper termination from initial marking to final marking such that $[p_I][\sigma^{Z_1}\rangle[p_O]$. Next, in Step 1°(c), we check if each (t, ϕ) satisfies the state transition. The state transition is satisifed because all data variables was read and wrote at the same value with the previously fired transitions i.e. the values $v_2^w=true$ when t_8 was fired is read as $v_2^r=30$ when t_5 fires. It is similar with v_1, v_2 and v_3. For example, $(DPN, [p_3, p_8, p_{10}], A)[(t_8, v_2=30), (t_9, v_3=40)\rangle(DPN, [p_3, p_9, p_{11}], A')$ allows $(DPN, [p_3, p_9, p_1 1], A'')[(t_5, \{v_2=30, v_3=40, v_4=true\})\rangle(DPN, [p_5, p_9, p_{11}], A''')$. In Step 2°, the procedure outputs no and stops. This means that there is no anomaly detected in σ^{Z_1}.

Next, we check σ^{Z_2}. In Step 1°, we check condition (i), (ii) and (iii) of Theorem 1. In 1°(a), each firing transforms the marking (M, A) to (M', A') where there is no guard violations. Therefore, all firing in σ^{Z_2} is valid. In Step 1°(b), the trace allows a proper termination from initial marking to final marking such that $[p_I][\sigma^{Z_2}\rangle[p_O]$. Next, in Step 1°(c), we check if each (t, ϕ) satisfies the state transition. The state transition is not satisifed because the values $v_1^w=false$ when t_1 was fired is read as $v_1^r=true$ when t_5 is fired. In Step 1°(c), the procedure outputs σ^{Z_2}. This means that there is an anomaly of invalid data transition detected in σ^{Z_2}.

Finally, we check σ^{Z_3}. In Step 1°, we check condition (i), (ii) and (iii) of Theorem 1. In 1°(a), each firing transforms the marking (M, A) to (M', A') where there is no guard violations. Therefore, all firing in σ^{Z_3} is valid. In Step 1°(b), the trace allows a proper termination from initial marking to final marking such that $[p_I][\sigma^{Z_3}\rangle[p_O]$ is not satisfed because t_3 and t_4 should only be fired before t_8 and t_9 in order to obtain the value of v_2 and v_3. In Step 1°(b), the procedure outputs σ^{Z_3}. We found that the firing sequence is wrong. The correct firing sequence should be $t_1\{v_1^w=true\}, t_8\{v_2^w=30\}, t_9\{v_3^w=40\}, t_3\{v_2^w=30, v_3^w=40\}, t_4\{v_4^r=false\}$. This means that there is an anomaly of wrong firing sequence detected in σ^{Z_3}.

VI. Conclusion

In this paper, we discussed analysis method for service security. We proposed a modeling technique based on data Petri net and conformance of valid trace from the event log. Prior to the analysis procedure, we proposed a soundness property for DPN that ensures the logical correctness of the model. Service security analysis is important for the detection of unintended illegal operation which can possibly be caused by device malfunction or cyber-attack in IoT systems.

In the examples given, if we do not consider the data flow in the design of IoT services, we can only detect wrong firing sequence of sequentially connected transitions in the DPN model. We showed that we can improve the false positive detection by considering parallel firing sequence and the data state transitions in the DPN model.

In the future work, we will propose a tool to verify the service security and test it for large-scale IoT systems. We will also improve the detection procedure with fast analysis algorithm using data mining techniques.

Acknowledgement

This work was partially supported by Interface Corporation, Japan and Yamaguchi University's KAKEN TRY (Research Grant for Young Scientist) Project 2017.

References

[1] M. A. B. Ahmadon, S. Yamaguchi, "On Service Orchestration of Cyber Physical System and its Verification Based on Petri Net," 2016 IEEE 5th Global Conference on Consumer Electronics, Kyoto, pp. 1–4, 2016.

[2] M. Singh, M. A. Rajan, V. L. Shivraj, P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)" 2015 Fifth Int. Conf. on Communication Systems and Network Technologies, Gwalior, pp. 746–751, 2015.

[3] M.A.B. Ahmadon, Z.Gou, S. Yamaguchi, B.B. Gupta, "Detection and Update Method for Attack Behavior Models in Intrusion Detection Systems," Proc. of INDIACom 2016, pp. 5637–5642, 2016.

[4] T. Wang, Q. Su, T. Chen, "Formal Analysis of Security Properties of Cyber-Physical System Based on Timed Automata," 2017 IEEE Second Int. Conference on Data Science in Cyberspace (DSC), Shenzhen, pp. 534–540, 2017.

[5] G. Howser, B. McMillin, "Using Information-Flow Methods to Analyze the Security of Cyber-Physical Systems," in Computer, vol. 50, no. 4, pp. 17–26, 2017.

[6] T. Murata, "Petri nets: Properties, analysis and applications," Proc. of the IEEE, vol.77, no.4, pp.541–580, 1989.

[7] M. de Leoni, W.M.P. van der Aalst, "Data-aware process mining : discovering decisions in processes using alignments," Proc. of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, pp.1454–1461, 2013.

[8] W.M.P. van der Aalst, K. M. van Hee, *Workflow Management: Models, Methods, and Systems*, The MIT Press, 2002.