DOI 10.15622/ia.21.2.1

F. Mercaldo, F. Martinelli, A. Santone

MODEL CHECKING FOR REAL-TIME ATTACK DETECTION IN WATER DISTRIBUTION SYSTEMS

F. Mercaldo, F. Martinelli, A. Santone **Model Checking for Real-Time Attack Detection in Water Distribution Systems.**

Abstract. Water distribution systems represent critical infrastructures. These architectures are really critical, and irregular behaviour can be reflected in human safety. As a matter of fact, an attacker obtaining control of such an architecture is able to perpetrate a plethora of damages, both to the infrastructure and people. In this paper, we propose an approach to identify irregular behaviours focused on water distribution systems. The designed approach considers a formal verification environment. The logs retrieved from water distribution systems are parsed into a formal model and, by exploiting timed temporal logic, we characterize the behaviour of a water distribution system while an attack is happening. The evaluation, referred to a water distribution system, confirmed the effectiveness of the designed approach in the identification of three different irregular behaviours.

Keywords: critical infrastructure, SCADA, formal verification environment, formal methods, timed automaton, safety, security.

1. Introduction. The networks of many critical infrastructures in countries depend strategically on SCADA systems. To provide some examples, energy generation and transport, gas and oil pipelines, communication systems and aqueducts are now largely managed through industrial automation technologies. [1]. The acronym SCADA (Supervisory Control And Data Acquisition) indicates a computer system for electronic monitoring and control of industrial systems. In a nutshell, with the term SCADA, we are referring to a system for the critical infrastructure management consisting of computers and networked data [2]. Typically, these architectures exploit peripheral devices, for instance, programmable logic controller [3]. The attendant is able to control the critical architectures through the critical infrastructure management system [4]. Sensors and computers cooperate with the aim to guarantee the service provided by the critical infrastructure: the problem is that sensors and computers expose the critical infrastructure to potential irregular behaviours [5].

An attack targeting a SCADA system can easily generate significant physical damage; for this reason attackers are increasingly interested in SCADA systems [6]. For example, in 2003, the Davis-Besse nuclear power plant and the CSX company in the US were victims of the Slammer and Sobig worms respectively. Another attack, Slammer, caused a denial of service that slowed the network down, while Sobig sent spam via email [7]. At the same time, the Sobig malware has infected a computer at CSX headquarters by blocking among other reporting and delivery systems, thus causing train delays [8].

SCADA attack can also afflict damages in airplane passengers. In fact, in 2004, transport companies, such as British Airways, Railcorp and Delta Airlines, were hit by the Sasser worm, which exploited a buffer overflow vulnerability to spread to other systems [9]. Some aggressive variants may have caused network overcrowding. The consequences were delays of trains and planes, in some cases, cancellation of flights [10].

Another critical infrastructure is represented by the oil companies, which have been attack targets. In fact, in 2009, companies operating in the Oil & gas and petrochemical sectors, such as Exxon, Shell and BP, were affected by the Night Dragon malware, distributed using spearphishing technologies. This malware allowed criminals to take remote control of infected computers [11].

Attacks on SCADA systems can also offer to attackers the opportunity not only to perpetrate harm to the population, but also to extract sensitive information. The case of the Stuxnet worm, appeared in 2010; it was a worm engaged in espionage and reprogramming of industrial systems at the Natanz nuclear plant in Iran. The virus intercepted and modified data within a Programmable Logic Controller (PLC). As a consequence, sensible data were gathered, and a fifth of Iranian nuclear centrifuges were destroyed [12].

The most common attacks are aimed at silencing safety warnings or alarms, muting attacks. The typical attack scenario is when the verification of the pressure inside a joint of a gas pipeline is silenced [13]. Much more complex are the attacks that modify the behavior of a SCADA system, for example, altering the pressure levels that the system of a pipeline considers normal [14].

Moreover, considering the obsolete architecture of many systems, it is possible to search through specific scanning tools, systems exposed on the web knowingly or not.

In the identification of cyber-attacks in critical infrastructure are currently exploited features gathered from network protocols: in this context, high-level features for cyber-attacks identification are not explored. For these reasons, in this paper, we design an approach to identify irregular behaviours targeting SCADA systems (with particular regard to water distribution systems), by considering features that we retrieve from a water distribution systems.

We consider a formal verification environment [15–18] to check if properties are satisfied on the modeled water distribution system. Clearly, when a property representing an irregular behaviour is verified, the irregular behaviour is in progress on the considered model.

This paper represents an extension of a preliminary work [19] presented at the 28th International Conference on Enabling Technologies: Infrastructure

for Collaborative Enterprises (WETICE). With respect to the paper in [19] below, we depict the contributions of this paper:

– we propose a property to identify the reply irregular behaviour;

– we extend the experiment to validate better the *underflow* and the *overflow* properties presented in [19];

– we introduce a new property for the identification of a new anomaly, i.e., the reply irregular behaviour.

The work proceeds as follows: the next section presents background concepts about model checking timed automata, Section 3 describes the designed approach, Section 4 discusses the performed experiment aimed to evaluate the approach and, finally, in Section 5 conclusions and future research directions are drawn.

2. Background. In this section, fundamental concepts related to the formal methods technique adopted by the proposed method are provided, i.e., the model checking timed automata.

The toolchain to apply the model-checking technique is composed of a *formal model* and a *temporal logic*: both of them are described in this section.

We consider timed automata, proposed by Alur and Dill [20, 21], as a formal verification technique for real-time systems as, for instance, the water distribution systems. In a nutshell, a timed automaton is composed of a classical finite automaton able to manage clocks, evolving continuously and synchronously with respect to absolute time. Each transition of a timed automaton is labelled by a guard or constraint overclock values, indicating the time in which the transition can be fired, and a set of clocks to be reset when the transition is fired. Each location is constrained by an invariant, which restricts the possible values of the clocks for being in the state, which can then enforce a transition to be taken [22–24].

In this paper, we model a sequence of logs gathered from a water distribution systems as a network of timed automata, i.e., a finite-state machine extended with clock variables [21]. The model is extended with bounded discrete variables that are part of the state. The state of the system is defined by the location of all automata, the clock values and the values of the discrete variables. Automaton may fire an edge (i.e., perform a transition) separately or synchronise with another automaton with the aim to lead a new state.

Below we provide the definition of the syntax and semantics for the basic timed automata. We exploit the following notation: C is a set of clocks and $B(C)$ is the set of conjunctions over simple conditions of the form $x \bowtie c$ or $x - y \bowtie c$, where $x, y \in C$, $c \in \mathbb{N}$ and $\bowtie \in \{ <, \leqslant, =, >, \geqslant \}$. A timed automaton is a finite directed graph annotated with conditions over and resets of non-negative real-valued clocks [21].

Definition 1. *Timed Automaton. A timed automaton is a tuple* (L, l_0, C, A, E, I) *where* L *is a set of locations,* $l_0 \in L$ *is the initial location,* C *is the set of clocks,* A *is a set of actions, co-actions and internal* τ*-action,* $E \subseteq L \times A \times B(C) \times 2^c \times L$ *is a set of edges between locations with an action, a guard and a set of clocks to be reset, and* $I : L \to B(C)$ *assigns invariants to locations.*

In the following, we define the semantics of a timed automaton. A clock valuation is a function $u : C \to \mathbb{R}_{\geqslant 0}$ from the set of clocks to the non-negative reals. Let $\mathbb{R}^C$ be the set of all clock valuation. Let $u_0(x) = 0$ for all $x \in C$. We will abuse the notation by considering guards and invariants as sets of clock valuations, writing $u \in I(l)$ to mean that u satisfies $I(l)$.

Definition 2. *Semantics of Timed Automaton. Let* $= (L, l_0, C, A, E, I)$ *be a timed automaton. The semantics is defined as a labelled transition system* $\langle S, s_0, \to \rangle$*, where* $S \subseteq L \times \mathbb{R}^C$ *is the set of states,* $s_0 = (l_0, u_0)$ *is the initial state, and* $\to \subseteq S \times (\mathbb{R}_{\geqslant 0} \cup A) \times S$ *is the transition relation such that:*

- $(l, u)d\ (l, u + d)\ if\ \forall d' : 0 \leqslant d' \leqslant d \implies u + d' \in I(l),\ and$
- $(l, u)a\ (l', u')$ *if there exists* $e = (l, a, g, r, l') \in E$ $s.t.\ u \in g, u' = [r \mapsto 0]u,\ and\ u' \in I(l')$,

where for $d \in \mathbb{R}_{\geqslant 0}$*,* $u + d$ *maps each clock* x *in* C *to the value* $u(x) + d$*, and* $[r \mapsto 0]u$ *denotes the clock valuation which maps each clock in* r *to 0 and agrees with* u *over* $C \backslash r$.

Timed automata are often composed into a *network of timed automata* over a common set of clocks and actions, consisting of n timed automata $_i = (L_i, l_i^0, C, A, E_i, I_i)$, $1 \leqslant i \leqslant n$. A location vector is a vector $\bar{l} = (l_1^0, \ldots, l_n^0)$. We compose the invariant functions into a common function over location vectors $I(\bar{l}) = \wedge_i I_i(l_i)$. We write $\bar{l}\ [l_i^{'}/l_i]$ to denote the vector where the i-th element l_i of $\bar{l}$ is replaced by $l_i^{'}$.

In the following, we define the semantics of a network of timed automata.

Definition 3. *Semantics of a network of Timed Automata: Let* $= (L, l_i^0, C, A, E_i, I_i)$ *be a network of* n *timed automata. Let* $\bar{l_0} = (l_1^0, \ldots, l_n^0)$ *be the initial location vector. The semantics is defined as a transition system* $\langle S, s_0, \to \rangle$*, where* $S = (L_1 x \ldots x L_n) \times \mathbb{R}^C$ *is the set of states,* $s_0 = (\bar{l_0}, u_0)$

is the initial state, and $\rightarrow \subseteq S \times S$ *is the transition relation defined by:*

- $(\bar{l}, u) \rightarrow (\bar{l}, u + d)\ if\ \forall d' : 0 \leqslant d' \leqslant d \implies u + d' \in I(\bar{l}),\ and$
- $(\bar{l}, u) \rightarrow (\bar{l}[l_i^{'}/l_i], u')$ *if there exists* $l_i \tau g r l_i^i\ s.t.\ u \in g, u' = [r \mapsto 0]u\ and\ u' \in I(l')$.
- $(\bar{l}, u) \rightarrow (\bar{l}[l_j^{'}/l_j, l_i^{'}/l_i], u')$ *if there exists* $l_i c?g_i r_i l_i^i\ and\ l_j c!g_j r_j l_j^j s.t. u \in (g_i \wedge g_j)$, $u' = [r_i \cup r_j \mapsto 0]u\ and\ u' \in I(l')$.

Modelling languages usually extend timed automata with the following additional features:

– *binary synchronisation*: channels are declared as chan c. An edge labelled with c! synchronises with another labelled c?. A synchronisation pair is chosen non-deterministically if several combinations are enabled;

– *broadcast channels*: are declared as broadcast chan c. In a broadcast synchronisation one sender c! can synchronise with an arbitrary number of receivers c?. Any receiver that can synchronise in the current state must do so. If there are no receivers, then the sender can still execute the c! action, i.e. broadcast sending is never blocking;

– *initialisers*: are used to initialise integer variables and arrays of integer variables. For instance, $int\ i := 2$; or $int\ i[3] := \{1, 2, 3\}$.

Once defined the formal model, we need to verify the model with regard to a requirement specification. Similarly to the model, the requirement specification must be expressed in a formally well-defined language. Several such logics exist in the scientific literature. In this paper, we consider the Timed Computational Temporal Logic (TCTL) [21, 25], which extends the classical untimed branching-time logic CTL [26] with time constraints on modalities.

The TCTL syntax is defined by the following grammar:

$$\phi \quad ::= \quad a \mid \neg\phi \mid \phi \vee \phi \mid \mathbf{E}\phi\ \mathbf{U}_I\phi \mid \mathbf{A}\phi\ \mathbf{U}_I\phi,$$

where $a \in AP$ (we denote with AP a set of atomic propositions), and I is an interval of $\mathbb{R}_+$ with integral bounds.

There are two possible semantics for TCTL, one which is said *continuous*, and the other one which is more discrete and is said *pointwise*. We consider the second one, i.e., the *pointwise* semantic (Table 1).

Where $\varrho[\pi]$ is the state of ϱ at position π, and duration $\varrho \leqslant_\pi$ is the prefix of ϱ ending at position π, and duration($\varrho \leqslant_\pi$) is the sum of all delays along ϱ up to position π.

Table 1. Timed temporal logic semantics

$(l,u) \models a \iff a \in (l)$
$(l,u) \models a \neg\phi (l,v) \not\models \phi$
$(l,u) \models \phi \vee \psi \iff (l,u) \models \phi \; or \; (l,u) \models \psi$
$(l,v) \models \mathbf{E}\phi \, \mathbf{U}_I \psi \iff there \; is \; an \; infinite \; run \; \varrho \; in \;\; from \; (l,u)$ $such \; that \; \varrho \models \phi \, \mathbf{U}_I \psi$
$(l,u) \models \mathbf{A}\phi \, U_I \psi \iff any \; infinite \; run \; \varrho \; in \;\; from \; (l,u)$ $is \; such \; that \; \varrho \models \phi \mathbf{U}_I \psi$
$\varrho \models \phi \, \mathbf{U}_I \psi \iff there \; exists \; a \; position \; \pi > 0 \; along \; \varrho \; such \; that$ $varrho[\pi] \models \psi, for \; every \; position \; 0 < \pi' < \pi, \varrho[\pi'] \models \psi,$ $andduration(\varrho \leqslant_\pi) \in I.$

In the pointwise semantics, a position in a run:

$$\varrho = s_0 \tau_1, e_1 s_1 \tau_2, e_2 s_2 \ldots s_{n-1} \tau_n, e_n s_n,$$

is an integer i and the corresponding state s_i. In this semantics, formulas are checked only right after a discrete action has been done. Sometimes, the pointwise semantics is given in terms of actions and timed words, but it does not change anything.

As usually in CTL, TCTL: $\equiv a \vee \neg \, a$ standing for true, $\equiv \neg$ standing for false, the implication $\phi \rightsquigarrow \psi \equiv (\neg \, \phi \vee \psi)$, the eventual operator $F_I \phi \equiv$ tt $U_I \phi$ and the global operator $G_I \phi \equiv \neg \, (F_I \neg \, \phi)$.

Formulae can be classified into *reachability*, *safety* and *liveness*. Figures 1, 2, 3 and 4 shows examples of different path formulae.

Reachability properties: they are looking if whether a given state formula, φ, possibly can be satisfied by any reachable state. Reachability properties are often used while designing a model to perform sanity checks. We express that some state satisfying φ should be reachable using the path formula E φ.

Safety properties are expressed in the following form: "something bad will never happen". These properties are usually formulated positively, for example, something good is invariantly true. For instance, let φ a state formula, we express that φ should be true in all states that are reachable with the path formulae A[ ] φ.

Liveness properties are of the following form: something will eventually happen. Liveness is expressed with the path formula A φ meaning φ is eventually satisfied. A useful form is the leads to or response property, written $\psi \rightsquigarrow \varphi$ which is read as whenever ψ is satisfied, then eventually φ will be satisfied.

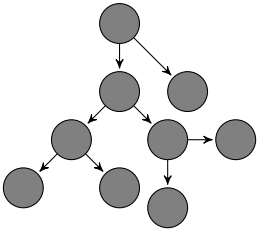

Fig. 1. A[ ] φ

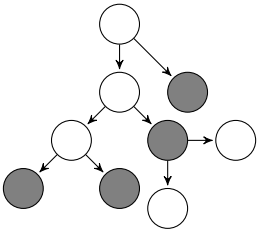

Fig. 2. A φ

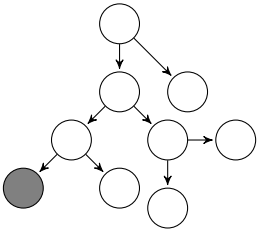

Fig. 3. E φ

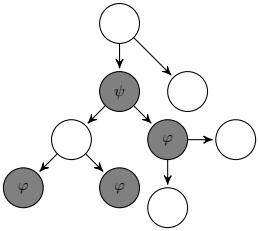

Fig. 4. $\psi \rightsquigarrow \varphi$

Once the model and temporal logic properties are defined, we need something enabling us to check whether the timed automata network (i.e., the formal model) satisfies the defined properties. To this aim, we consider formal verification, a system process exploiting mathematical reasoning to verify whether a system under analysis (i.e., the model) satisfies some requirements (i.e., the timed temporal properties).

Several verification techniques have been proposed in the last years. In this paper, we resort to model checking [24, 27].

In the model checking technique the properties are formulated in temporal logic: each property is evaluated against the system. The model checker accepts as input a model and a property, it returns "true" whether the system satisfies the formula and "false" otherwise. The performed check is an exhaustive state space search that is guaranteed to terminate since the model is finite. In this paper, we consider as model checker UPPAAL[1] [22, 23, 28], an integrated tool environment for modeling, validation and verification of real-time systems modeled as timed automata networks. Thus, the syntax of UPPAAL expression is given in Table 2.

Table 2. Syntax of expressions in BNF

$$
\begin{aligned}
Expression \rightarrow\ & ID \mid NAT \\
& \mid Expression\,'['Expression']' \\
& \mid '('Expression')' \\
& \mid Expression\,'++' \mid '++'\,Expression \\
& \mid Expression\,'--' \mid '--'\,Expression \\
& \mid Expression\ AssignOp\ Expression \\
& \mid UnaryOp\ Expression \\
& \mid Expression\ BinaryOp\ Expression \\
& \mid Expression\,'?'\ Expression\,':'\ Expression \\
& \mid Expression\,'.'\ ID \\
UnaryOp \rightarrow\ & '-' \mid '!' \mid 'not' \\
BinaryOp \rightarrow\ & '<' \mid '\leqslant' \mid '==' \mid '!=' \mid '\geqslant' \mid '>' \\
& \mid '+' \mid '-' \mid '*' \mid '/' \mid '\%' \mid '\&' \\
& \mid '|' \mid '\hat{}' \mid '\ll' \mid '\gg' \mid '\%\%' \mid '\|' \\
& \mid '<?' \mid '>?' \mid '\wedge' \mid '\vee' \mid '\rightsquigarrow' \\
AssignOp \rightarrow\ & ':=' \mid '+=' \mid '-=' \mid '*=' \mid '/=' \mid '\%=' \\
& \mid '|=' \mid '\%=' \mid '\hat{}=' \mid '\ll=' \mid '\gg='
\end{aligned}
$$

[1] http://www.uppaal.org/

Expressions are used with the following labels:

– *guard*: a particular expression satisfying the following conditions: (i) it is side-effect free; (ii) it evaluates to a boolean; (iii) only clocks, integer variables, and constants are referenced (or arrays of these types); (iv) clocks and clock differences are only compared to integer expressions; (v) guards overclocks are essentially conjunctions (disjunctions are allowed over integer conditions);

– *synchronisation*: a synchronisation label is either on the form Expression! or Expression? or is an empty label. The expression must be side-effect free, evaluate to a channel, and only refer to integers, constants and channels;

– *assignment*: an assignment label is a comma-separated list of expressions with a side-effect; expressions must only refer to clocks, integer variables, and constants and only assign integer values to clocks. *invariant*: an expression that satisfies the following conditions: it is a side-effect free; only clock, integer variables, and constants are referenced; it is a conjunction of conditions of the form $x < e$ or $x \leqslant e$ where x is a clock reference and e evaluates to an integer.

3. The Designed approach. In the designed approach, we consider the cistern level gauging as features. In details, if the water distribution system is formed by two cisterns, i.e., two features describing the water gauging in the two cisterns are considered.

The designed approach consists of two main steps: the *Formal Model Creation* (Figure 5) and the *Formal Model Verification* (Figure 8).

From the water distribution system under analysis and the technical report, a technician marks the specific day log as *irregular*. The features gathered from the cistern levels and the label to mark a specific trace as *irregular* are the input for the *one day log* stored in CSV files, containing the cistern gauging for one day at a fixed range time (1 hour).

The *Discretisation* is aimed to discretise each cistern level feature. The numeric values are split into 3 different ranges [29]. Furthermore, each discretised feature previously gathered is converted into a timed model. The discretisation process is aimed to convert continuous values into discrete ones. A plethora of methods were proposed by the research community for numeric values discretisation: in our approach, we exploit one discussed by researchers in [29]. The idea behind this method divides the numeric features into three intervals by exploiting the equal-width partitioning that basically divides the values of a certain numeric value into three equal-size intervals. In our case, the cistern level continuous values are divided into one of three different classes (*Up*, *Basal*, and *Low*).

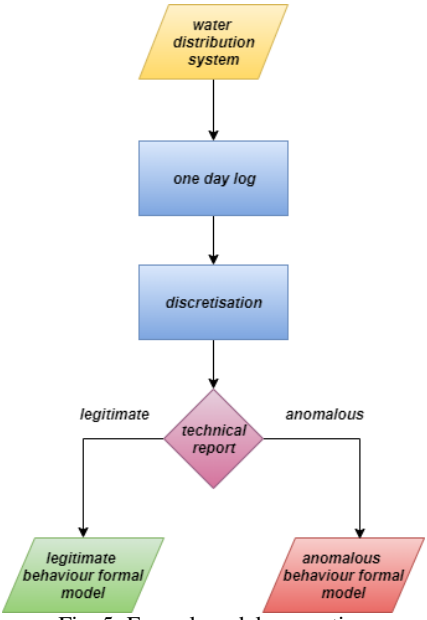

Fig. 5. Formal model generation

We compute the interval width in the following way: $W = (Max - Min)/3$, where Max and Min indicate the maximum and the minimum values. The partitioning is applied to all features (i.e., the cisterns). Moreover, once obtained the discrete values from the discretisation process, from each feature is generated a timed automaton (i.e., *formal model* in Figure 5).

With the aim to better explain how the timed automaton is built, let us consider the following example. In particular, in Table 3 we represent a fragment of a discretised feature.

With the first column (i.e., *Time* in Table 3) we are referring to the interval time (in the fragment $1 \leqslant t \leqslant 6$), while with the F_1 and F_2 columns, we are referring to the two considered features (i.e., the cistern levels). By considering the fragment in Table 3, in the t_3 time interval, the F_1 reaches the Up value, while the F_2 feature reaches the Low value. Once obtained the discretised values for the features (i.e., the cistern values), it is possible to generate the formal model. In detail, we build a timed automata network: in a nutshell, for each discretised feature, we generated an automaton.

Table 3. A fragment related to the feature discretisation

Time	F_1	F_2
t_1	*Up*	*Up*
t_2	*Up*	*Low*
t_3	*Up*	*Low*
t_4	*Basal*	*Up*
t_5	*Low*	*Basal*
t_6	*Up*	*Basal*

By considering the discretised fragment, we shown in Table 3 the timed automata network that is built in the following way: if the same discrete value is repeated in a consecutive time interval, the automaton related to the feature under analysis exhibits a loop: for instance, the automaton built from the F_1 discretised feature contains a loop related to the t_1, t_2 and t_3 time intervals (because the *Up* discretised values are repeated three times). Moreover, the resulting automaton for the F_2 feature shows two loops: the first loop is related to the t_2 and t_3 time intervals (*Low* is the repeated discrete value), while the second loop is related to the t_5 and t_6 time intervals, and in this last case the repeated discrete value is *Basal*).

In order to exit from the loops, a guard is exploited. Differently, with regard to the entering condition, an invariant is considered. Moreover, for each automaton, we consider a number of clocks equal to two: the first clock (x) to ensure the entering condition into the loop and the second one (y) to ensure the exit condition. Moreover, each automaton is responsible for storing the count related to the respective *Up*, *Basal* and *Low* values.

The values for the F_x automaton are labelled with a subscript $x \in \{1, 2\}$ (we consider two features). The s channel permits the automata synchronisation and, for this reason, is not stored locally. The aim of the channels is to ensure the automata network progression. This mechanism basically is a hand-shaking synchronization: two processes take a transition at the same time, the first one will have an $s!$, while the other an $s?$, in order to ensure the synchronization. As a matter of fact, one sender event $s!$ is able to synchronise with a number of $s?$ receiver events. We resort to channels in order to avert incoherence from the discretised feature values and the interval times. For instance, the automaton related to the first feature is indicated with F_1 and its variables are u_1, b_1 and l_1 respectively for *Up*, *Basal* and *Low* values. As a consequence, the automaton for the second variable is F_1, and its variables are u_2, b_2 and l_2, respectively for *Up*, *Basal* and *Low* values of this second automaton.

Figures 6 and 7 indicate the model, respectively, gathered from the F_1 and F_2 discretisation. In both the figures, we indicate with *u* the *Up* value, with *b* the *Basal* value, and with *l* the *Low* value, all the three values are present in Table 3.

Below we provide a brief explanation about the models represented in Figures 6 and 7: the F_1 automaton is iterating in the loop (node 1 in Figure 6) for three-time intervals (i.e., $y_1 < 3$), while the F_2 automaton after the increment of the u_1 variable (node 1 in Figure 7) is iterating for two-time intervals (i.e., $y_2 < 2$). Subsequently, the F_1 automaton does not exhibit any loops, while the F_2 automaton is iterating for two-time intervals in the loop in node 3 in Figure 7, and then it continues with the last node.

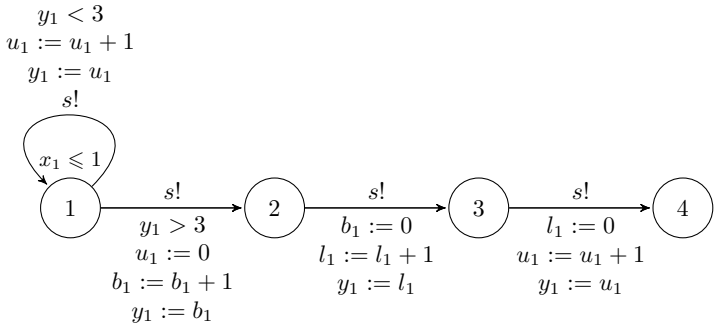

Fig. 6. The F_1 model

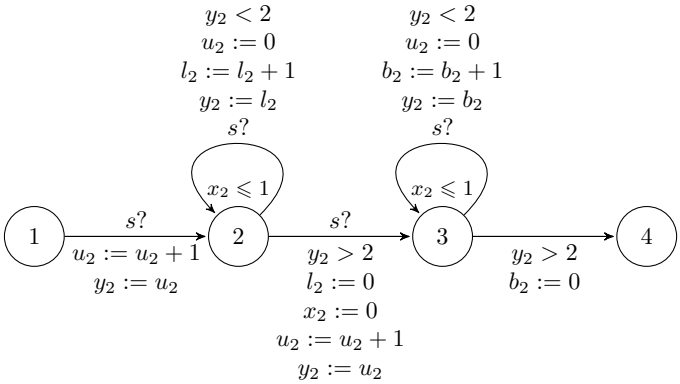

Fig. 7. The F_2 model

Once generated the formal model, the *Formal Model Verification* step (Figure 8) is aimed at checking if the modeled water distribution system is violated.

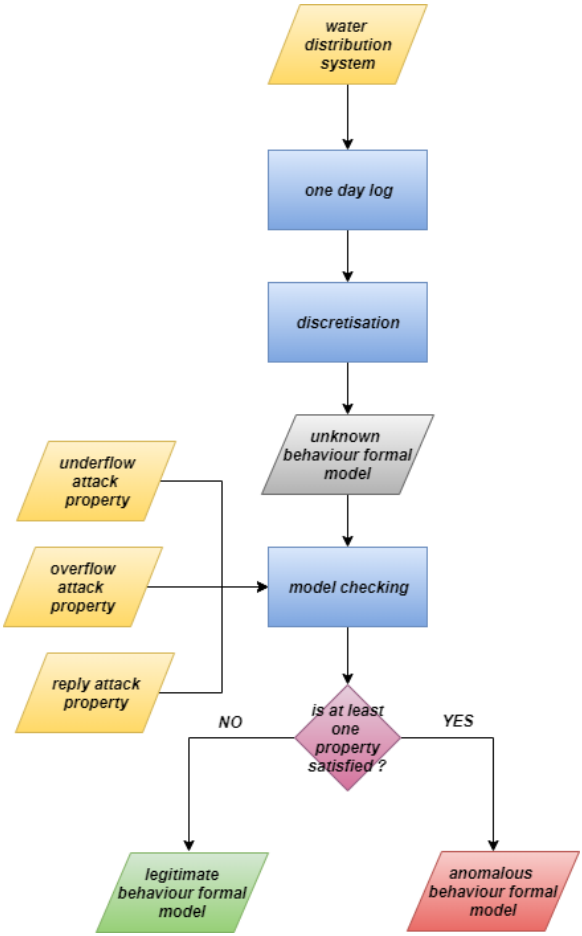

Fig. 8. Formal model verification

The *Formal Model Verification* receives as input a formal model (*formal model*) built in the previous step and a set of properties. The set of logic properties is checked against the formal model generated (*formal verification environment*) using the UPPAAL (an acronym based on a combination of UPPsala and AALborg universities) formal verification environment[2]. UPPAAL represents an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata.

4. Experimental evaluation. From the physical layout point of view, the following generic scenario is considered: the analysed SCADA water distribution system is composed by a generic water distribution system operator where, recently, has introduced a new technology to enable remote data collection from sensors and remote control of actuators. After the technology was introduced, anomalous levels in two tanks were observed; for instance, water overflow in one tank occurred. By looking for the causes, experts domain suspect potential cyberattacks. In particular, they consider some kind of malicious behaviours [30, 31] aimed to activate and deactivate the actuators.

The dataset[3] exploited in the experimentation of the designed approach contains one-day logs from a water distribution system composed from the water levels of the two cisterns (i.e., *ct1* and *ct2*).

We have hourly gauging of the water under *regular* operating conditions, and when an *overflow* (OF), an *underflow* (UF) or a *reply* (R) irregular behaviours targeting, respectively, the *ct1* and *ct2* cisterns are happening. Logs referred to 30 days of gauging are considered: ten marked with the OF irregular behaviour on *ct1*, ten with the UF irregular behaviour on *ct2* and the last ten days with *reply* (R) irregular behaviour on both the *ct1* and *ct2* cisterns (irregular behaviours have happened each day), while the remaining ten days without irregular behaviours [5, 32, 33]. For each day log a formal model is built. In total, we consider 40 days.

To summarise, in the experiment, we consider 40 days: 30 days related to irregular behaviours and 10 days with legitimate behaviours. The 30 days contain the following irregular behaviours: 10 days with overflow attack each day, 10 days with underflow attack each day and 10 days with reply attack each day.

4.1. The Properties. Table 4 indicates the OF, UF and R irregular behaviour identification properties.

[2] http://www.uppaal.org/
[3] http://www.batadal.net

Table 4. Timed temporal logic property for OF, UF and R water distribution irregular behaviour identification

$E\varphi$, $where\ \varphi$	=	$h_1 \geqslant 11$
$E\chi$, $where\ \chi$	=	$l_2 \geqslant 9$
$E\psi$, $where\ \psi$	=	$b_1 \geqslant 12 \wedge\ b_2 \geqslant 12$
$E\varphi\ \vee\ \chi\ \vee\ \psi$		

The φ property, referred to the OF irregular behaviour, is able to verify if an OF is in progress (i.e., when the *ct1* level exhibits an *up* value at least 11 times). The UF irregular behaviour, described by the χ property, is able to verify if an UF is in progress (i.e., when the *ct2* level exhibits a *low* value at least 9 times). The last property, i.e., ψ (for the R irregular behaviour identification), is aimed to verify if both the *ct1* and the *ct2* cisterns present the same value for more than 12 times. We want to verify if the φ, the χ and the ψ properties, possibly, can be satisfied by a reachable state. We express that some state satisfying φ should be reachable using the path property $E\varphi$. Similar considerations can be done for the χ and ψ properties. Furthermore, provide the property expressing that can happen an OF, or UF or a R irregular behaviours (i.e., $E\varphi\ \vee\ \chi\ \vee\ \psi$).

The properties were formulated with the help of domain experts. As a matter of fact, the properties are aimed to explain the domain experts knowledge. In particular, the domain experts suggest that whether *ct1* exhibits for a number of times equal or greater than 11 an increasing value, this is reflecting of an overflow attack in progress, while if *ct2* exhibits a value that is decreasing for a number of times equal or greater than 9, this is reflecting in an underflow attack in progress. With regard to the reply attack, expert domains suggested that whether both *ct1* and *ct2* are showing basal values for a number of times equal or greater than 12 this is symptomatic for this kind of attack in progress.

4.2. The Experiment. The property verification outcomes are indicated in Table 5.

With L^x we mark the log of the water cisterns for a single log, where $L \in \{irregular, regular\}$, x identifies the log under analysis. The dataset comprises 40-day log, 10 exhibiting normal behaviour, while the remaining 30 are afflicted by OF, UF and R irregular behaviours. The column (φ) identifies the models verified as *true* (✓ in Table 5) or *false* (✗ in Table 5) to the OF

property, while the column (χ) identifies the models verified as *true* or *false* to the UF property.

From the formal verification environment output indicated in Table 5 we can state the φ and the χ are able to identify all the models afflicted by the OF irregular behaviour (i.e., $irregular_o^1$, $irregular_o^2$, $irregular_o^3$, $irregular_o^4$, $irregular_o^5$, $irregular_o^6$, $irregular_o^7$, $irregular_o^8$, $irregular_o^9$ and $irregular_o^{10}$). The χ property is able to identify all the models afflicted by the UF irregular behaviour (i.e., $irregular_u^1$, $irregular_u^2$, $irregular_u^3$, $irregular_u^4$, $irregular_u^5$, $irregular_u^6$, $irregular_u^7$, $irregular_u^8$, $irregular_u^9$ and $irregular_u^{10}$). The ψ property is able to identify all the models afflicted by the R irregular behaviour (i.e., $irregular_r^1$, $irregular_r^2$, $irregular_r^3$, $irregular_r^4$, $irregular_r^5$, $irregular_r^6$, $irregular_r^7$, $irregular_r^8$, $irregular_r^9$ and $irregular_r^{10}$). Furthermore, all the models without irregular behaviour (i.e., $regular^1$, $regular^2$, $regular^3$, $regular^4$, $regular^5$, $regular^6$, $regular^7$, $regular^8$, $regular^9$ and $regular^{10}$) are marked as *false* by φ, χ and ψ properties.

Moreover, we consider four different metrics to evaluate the performance of the proposed approach: Precision, Recall, F-Measure and Accuracy.

The precision has been computed as the proportion of the observations that truly belong to investigated logs among all those which were assigned to the specific attack. It is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved:

$$lclPrecision = \frac{tp}{tp + fp},$$

where *tp* indicates the number of true positives (for instance, whether w e are evaluating the φ formula, this value represents the number of one-day logs whose related *overflow attack* model is correctly labelled as *true* by the formal verification environment) and *fp* indicates the number of false positives (for instance, whether we are evaluating the φ formula, this value represents the number of models whose related *legitimate* model is wrongly labelled as *true* by the formal verification environment).

Table 5. Property verification

Performances / Model	φ	χ	ψ
$irregular_g^1$	✓	✗	✗
$irregular_g^2$	✓	✗	✗
$irregular_g^3$	✓	✗	✗
$irregular_g^4$	✓	✗	✗
$irregular_g^5$	✓	✗	✗
$irregular_g^6$	✓	✗	✗
$irregular_g^7$	✓	✗	✗
$irregular_g^8$	✓	✗	✗
$irregular_g^9$	✓	✗	✗
$irregular_g^{10}$	✓	✗	✗
$irregular_y^1$	✗	✓	✗
$irregular_y^2$	✗	✓	✗
$irregular_y^3$	✗	✓	✗
$irregular_y^4$	✗	✓	✗
$irregular_y^5$	✗	✓	✗
$irregular_y^6$	✗	✓	✗
$irregular_y^7$	✗	✓	✗
$irregular_y^8$	✗	✓	✗
$irregular_y^9$	✗	✓	✗
$irregular_y^{10}$	✗	✓	✗
$irregular_r^1$	✗	✗	✓
$irregular_r^2$	✗	✗	✓
$irregular_r^3$	✗	✗	✓
$irregular_r^4$	✗	✗	✓
$irregular_r^5$	✗	✗	✓
$irregular_r^6$	✗	✗	✓
$irregular_r^7$	✗	✗	✓
$irregular_r^8$	✗	✗	✓
$irregular_r^9$	✗	✗	✓
$irregular_o^{10}$	✗	✗	✓
$regular^1$	✗	✗	✗
$regular^2$	✗	✗	✗
$regular^3$	✗	✗	✗
$regular^4$	✗	✗	✗
$regular^5$	✗	✗	✗
$regular^6$	✗	✗	✗
$regular^7$	✗	✗	✗
$regular^8$	✗	✗	✗
$regular^9$	✗	✗	✗
$regular^{10}$	✗	✗	✗

The recall has been computed as the proportion of attacks that were assigned to a given class among all the attacks that truly belong to the class. It is the ratio of the number of relevant records retrieved to the total number of relevant records:

$$lclRecall = \frac{tp}{tp + fn},$$

where *tp* indicates the number of true positives and *fn* indicates the number of false negatives (for instance, whether we are evaluating the φ formula, this value represents the number of one-day log whose related *overflow attack* model is wrongly labelled as *false* by the formal verification environment).

The F-Measure is a measure of a test's accuracy. This score can be interpreted as a weighted average of the precision and recall:

$$lclF\text{-}Measure = 2 * \frac{Precision * Recall}{Precision + Recall},$$

where *Precision* and *Recall* are obtained by following the formulae above explained.

The Accuracy is the fraction of the model correctly identified, and it is computed as the sum of true positives and negatives divided all the evaluated models:

$$lclAccuracy = \frac{tp + tn}{tp + fn + fp + tn},$$

where *tn* indicates the number of true negatives (for instance, whether we are evaluating the φ formula, this value represents the number of one-day logs whose related *legitimate* model is correctly labelled as *false* by the formal verification environment).

By the metrics computation, we reach a value equal to 1 for all metrics we have considered (i.e., Precision, Recall, F-Measure and Accuracy). This is symptomatic that the designed approach is able to correctly identify the several irregular behaviours we considered in the experiment with no false positive.

5. Conclusion and future work. The risks of an attack on SCADA systems are concrete and real: these are very sensitive objectives given the importance of the processes they govern and the impact that a disservice can cause on the community. Precisely, because of their structure distributed over the territory, hitting a SCADA system can really affect hundreds or thousands of other sites or plants, which in turn become unmanageable and therefore dangerous. There are countless organizations and goals that move interests in carrying out or making increasingly sophisticated actions and attack strategies, ranging from fraud and physical to creating disservices.

For this reason, to mitigate the irregular behaviours in critical infrastructure and, in particular, in SCADA water distribution systems, in this paper, we design a timed automata-based approach to identify OF, UF and R irregular behaviours on the water distribution system.

We propose formal model logs obtained from SCADA systems in terms of a timed automata network by exploiting the UPPAAL formal verification environment.

An accuracy equal to 1 is reached, symptomatic of the effectiveness of the proposed approach in attack detection in water distribution systems.

The proposed method can be adopted for real-time irregular behaviour detection in critical systems while the monitored SCADA system is under attack. As a matter of fact, in critical contexts, it is of fundamental importance to identify a threat when the latter is in progress in order to minimize data and put the system in a position to work safely. Moreover, considering that the proposed approach exploits formal methods and temporal logic formula, it is possible to understand the reason why a certain property is resulting false using the counterexample. Although the principal aim of the counterexample is to assist the designer in finding the source of the error in complex systems design, the counterexample can be exploited for other purposes. For instance, in the context of the proposed method can be used to understand how many time intervals the property will become true, thus providing a sort of probability of risk that irregular behavior may occur. This aspect can be of interest because it can be considered to forecast future irregular behaviours, thus taking measures before they can possibly take place.

As future work, we plan to evaluate the proposed of other SCADA critical systems (for instance, relating to oil/gas or power management and distribution) with the aim to validate the proposed method in another critical environment. Moreover, we plan to evaluate the proposed method with systems with much faster dynamics (for instance, power distribution/generation), i.e., with significantly larger models.

References

1. S. Cheruvu, A. Kumar, N. Smith, and D.M. Wheeler, “Conceptualizing the secure internet of things,” in Demystifying Internet of Things Security, pp. 1–21, Springer, 2020.
2. K. Jia, J. Xiao, S. Fan, and G. He, “A mqtt/mqtt-sn-based user energy management system for automated residential demand response: Formal verification and cyber-physical performance evaluation,” Applied Sciences, vol. 8, no. 7, p. 1035, 2018.
3. S.A. Boyer, SCADA: supervisory control and data acquisition. International Society of Automation, 2009.
4. B. Miller and D.C. Rowe, “A survey scada of and critical infrastructure incidents.,” RIIT, vol. 12, pp. 51–56, 2012.
5. R. Taormina, S. Galelli, N.O. Tippenhauer, E. Salomons, A. Ostfeld, D.G. Eliades, M. Aghashahi, R. Sundararajan, M. Pourahmadi, M.K. Banks, et al., “Battle of the attack

detection algorithms: Disclosing cyber attacks on water distribution networks", Journal of Water Resources Planning and Management, vol. 144, no. 8, p. 04018048, 2018.

6. P.K. Hajoary and K. Akhilesh, "Role of government in tackling cyber security threat," in Smart Technologies, pp. 79–96, Springer, 2020.
7. R. Meyur, "A bayesian attack tree based approach to assess cyber-physical security of power system," in 2020 IEEE Texas Power and Energy Conference (TPEC), pp. 1–6, IEEE, 2020.
8. I.N. Fovino, A. Carcano, M. Masera, and A. Trombetta, "An experimental investigation of malware attacks on scada systems," International Journal of Critical Infrastructure Protection, vol. 2, no. 4, pp. 139–145, 2009.
9. A. Carcano, I.N. Fovino, M. Masera, and A. Trombetta, "Scada malware, a proof of concept," in International Workshop on Critical Information Infrastructures Security, pp. 211–222, Springer, 2008.
10. T. Alladi, V. Chamola, and S. Zeadally, "Industrial control systems: Cyberattack trends and countermeasures," Computer Communications, 2020.
11. F. Daryabar, A. Dehghantanha, N.I. Udzir, S. bin Shamsuddin, et al., "Towards secure model for scada systems," in Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), pp. 60–64, IEEE, 2012.
12. T. Wu, J.F.P. Disso, K. Jones, and A. Campos, "Towards a scada forensics architecture," in 1st International Symposium for ICS SCADA Cyber Security Research 2013 (ICS-CSR 2013) 1, pp. 12–21, 2013.
13. D. Upadhyay and S. Sampalli, "Scada (supervisory control and data acquisition) systems: Vulnerability assessment and security recommendations," Computers Security, vol. 89, p. 101666, 2020.
14. M. Gaiceanu, M. Stanculescu, P.C. Andrei, V. Solcanu, T. Gaiceanu, and H. Andrei, "Intrusion detection on ics and scada networks," in Recent Developments on Industrial Control Systems Resilience, pp. 197–262, Springer, 2020.
15. M.G. Cimino, N. De Francesco, F. Mercaldo, A. Santone, and G. Vaglini, "Model checking for malicious family detection and phylogenetic analysis in mobile environment," Computers Security, vol. 90, p. 101691, 2020.
16. L. Brunese, F. Mercaldo, A. Reginelli, and A. Santone, "Formal methods for prostate cancer gleason score and treatment prediction using radiomic biomarkers," Magnetic resonance imaging, vol. 66, pp. 165–175, 2020.
17. L. Brunese, F. Mercaldo, A. Reginelli, and A. Santone, "An ensemble learning approach for brain cancer detection exploiting radiomic features," Computer methods and programs in biomedicine, vol. 185, p. 105134.
18. L. Brunese, F. Mercaldo, A. Reginelli, and A. Santone, "Prostate gleason score detection and cancer treatment through real-time formal verification," IEEE Access, vol. 7, pp. 186236–186246, 2019.
19. F. Mercaldo, F. Martinelli, and A. Santone, "Real-time scada attack detection by means of formal methods," in 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 231–236, IEEE, 2019.
20. R. Alur and D. Dill, "Automata for modeling real-time systems," in International Colloquium on Automata, Languages, and Programming, pp. 322–335, Springer, 1990.
21. R. Alur and D.L. Dill, "A theory of timed automata," Theoretical computer science, vol. 126, no. 2, pp. 183–235, 1994.
22. G. Behrmann, K.G. Larsen, O. Moller, A. David, P. Pettersson, and W. Yi, "Uppaal-present and future," in Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228), vol. 3, pp. 2881–2886, IEEE, 2001.

23. G. Behrmann, A. David, and K.G. Larsen, “A tutorial on uppaal 4.0,”Department of computer science, Aalborg university, 2006.
24. G. Behrmann, A. David, and K.G. Larsen, “A tutorial on uppaal,” in Formal methods for the design of real-time systems, pp. 200–236, Springer, 2004.
25. P. Bouyer, “Model-checking timed temporal logics,” Electronic Notes in Theoretical Computer Science, vol. 231, pp. 323–341, 2009.
26. E.M. Clarke, E.A. Emerson, and A.P. Sistla, “Automatic verification of finite-state concurrent systems using temporal logic specifications,” ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 8, no. 2, pp. 244–263, 1986.
27. N.D. Francesco, G. Lettieri, A. Santone, and G. Vaglini, “Heuristic search for equivalence checking,” Software and System Modeling, vol. 15, no. 2, pp. 513–530, 2016.
28. H.E. Jensen, K.G. Larsen, and A. Skou, “Scaling up uppaal,” in International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, pp. 19–30, Springer, 2000.
29. J. Dougherty, R. Kohavi, and M. Sahami, “Supervised and unsupervised discretization of continuous features,” in Machine Learning Proceedings 1995, pp. 194–202, Elsevier, 1995.
30. F. Mercaldo and A. Santone, “Deep learning for image-based mobile malware detection,” Journal of Computer Virology and Hacking Techniques, pp. 1–15, 2020.
31. A. De Lorenzo, F. Martinelli, E. Medvet, F. Mercaldo, and A. Santone, “Visualizing the outcome of dynamic analysis of android malware with vizmal,” Journal of Information Security and Applications, vol. 50, p. 102423, 2020.
32. R. Taormina, S. Galelli, N.O. Tippenhauer, A. Ostfeld, and E. Salomons, “Assessing the effect of cyber-physical attacks on water distribution systems,” in World Environmental and Water Resources Congress 2016, pp. 436–442, 2016.
33. E. Salomons and A. Ostfeld, “Simulation of cyber-physical attacks on water distribution systems with epanet,” in Proceedings of the Singapore Cyber-Security Conference (SGCRC) 2016: Cyber-Security by Design, vol. 14, p. 123, 2016.

Mercaldo Francesco — Ph.D., Researcher, University of Molise. Research interests: artificial intelligence, mobile computing, Android (operating system), invasive software, security of data, biomedical MRI, medical image processing, pattern classification. The number of publications — 61. francesco.mercaldo@iit.cnr.it; 1, Via Francesco De Sanctis St., 86100, Campobasso, Italy; office phone: +3908744041, 171.

Martinelli Fabio — Ph.D., Dr.Sci., Research director, institute for informatics and telematics, National Research Council of Italy. Research interests: formal analysis of network and system security, security protocols, secure service, information flow analysis, PKI and trust management, access and usage control, web and GRID services, mobile devices, formal analysis of concurrent, distributed, mobile systems, program synthesis. The number of publications — 166. fabio.martinelli@iit.cnr.it; 1, Via Giuseppe Moruzzi St., 56124, Pisa, Italy; office phone: +390503153425, 443.

Santone Antonella — Ph.D., Associate professor, Department of engineering, University of Molise. Research interests: formal description techniques, temporal logic, concurrent and distributed systems modeling, heuristic search, formal methods for systems biology and for security engineering. The number of publications — 138. antonella.santone@unimol.it; 1, Via Francesco De Sanctis St., 86100, Campobasso, Italy; office phone: +3908744041, 171.

DOI 10.15622/ia.21.2.1

Ф. Меркальдо, Ф. Мартинелли, А. Сантоне

ПРОВЕРКА МОДЕЛИ ДЛЯ ОБНАРУЖЕНИЯ В РЕАЛЬНОМ ВРЕМЕНИ АТАК В СИСТЕМАХ РАСПРЕДЕЛЕНИЯ ВОДЫ

Меркальдо Ф., Мартинелли Ф., Сантоне А. **Проверка модели для обнаружения в реальном времени атак в системах распределения воды.**

Аннотация. Системы распределения воды представляют собой критическую инфраструктуру. Эти архитектуры очень важны, и нестандартное поведение может отразиться на безопасности человека. Фактически, злоумышленник, получивший контроль над такой архитектурой, может нанести множество повреждений как инфраструктуре, так и людям. В этой статье мы предлагаем подход к выявлению нестандартного поведения, ориентированного на системы распределения воды. Разработанный подход рассматривает формальную среду проверки. Журналы, полученные из систем распределения воды, анализируются в формальную модель, и, используя временную логику, мы характеризуем поведение системы распределения воды во время атаки. Оценка, относящаяся к системе распределения воды, подтвердила эффективность разработанного подхода при выявлении трех различных нестандартных режимов работы.

Ключевые слова: критическая инфраструктура, SCADA, формальная среда верификации, формальные методы, таймер, безопасность, охрана.

Меркальдо Франческо — Ph.D., научный сотрудник, Университет Молизе. Область научных интересов: искусственный интеллект, мобильные вычисления, Android (операционная система), инвазивное программное обеспечение, безопасность данных, биомедицинская МРТ, обработка медицинских изображений, классификация образов. Число научных публикаций — 61. francesco.mercaldo@iit.cnr.it; Виа Франческо де Санктис, 1, 86100, Кампобассо, Италия; р.т.: +3908744041, 171.

Мартинелли Фабио — Ph.D., Dr.Sci., директор по исследованиям, институт информатики и телематики, Национальный исследовательский совет Италии. Область научных интересов: формальный анализ сетевой и системной безопасности, протоколы безопасности, безопасный сервис, анализ информационных потоков, PKI и доверительное управление, контроль доступа и использования, веб- и GRID-сервисы, мобильные устройства, формальный анализ параллельных, распределенных, мобильных систем, программный синтез. Число научных публикаций — 166. fabio.martinelli@iit.cnr.it; Виа Джузеппе Моруцци, 1, 56124, Пиза, Италия; р.т.: +390503153425, 443.

Сантоне Антонелла — Ph.D., доцент, инженерный факультет, Университет Молизе. Область научных интересов: методы формального описания, темпоральная логика, моделирование параллельных и распределенных систем, эвристический поиск, формальные методы системной биологии и техники безопасности. Число научных публикаций — 138. antonella.santone@unimol.it; Виа Франческо де Санктис, 1, 86100, Кампобассо, Италия; р.т.: +3908744041, 171.

Литература

1. S. Cheruvu, A. Kumar, N. Smith, and D.M. Wheeler, "Conceptualizing the secure internet of things," in Demystifying Internet of Things Security, pp. 1–21, Springer, 2020.
2. K. Jia, J. Xiao, S. Fan, and G. He, "A mqtt/mqtt-sn-based user energy management system for automated residential demand response: Formal verification and cyber-physical performance evaluation," Applied Sciences, vol. 8, no. 7, p. 1035, 2018.
3. S.A. Boyer, SCADA: supervisory control and data acquisition. International Society of Automation, 2009.
4. B. Miller and D.C. Rowe, "A survey scada of and critical infrastructure incidents.," RIIT, vol. 12, pp. 51–56, 2012.
5. R. Taormina, S. Galelli, N.O. Tippenhauer, E. Salomons, A. Ostfeld, D.G. Eliades, M. Aghashahi, R. Sundararajan, M. Pourahmadi, M.K. Banks, et al., "Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks", Journal of Water Resources Planning and Management, vol. 144, no. 8, p. 04018048, 2018.
6. P.K. Hajoary and K. Akhilesh, "Role of government in tackling cyber security threat," in Smart Technologies, pp. 79–96, Springer, 2020.
7. R. Meyur, "A bayesian attack tree based approach to assess cyber-physical security of power system," in 2020 IEEE Texas Power and Energy Conference (TPEC), pp. 1–6, IEEE, 2020.
8. I.N. Fovino, A. Carcano, M. Masera, and A. Trombetta, "An experimental investigation of malware attacks on scada systems," International Journal of Critical Infrastructure Protection, vol. 2, no. 4, pp. 139–145, 2009.
9. A. Carcano, I.N. Fovino, M. Masera, and A. Trombetta, "Scada malware, a proof of concept," in International Workshop on Critical Information Infrastructures Security, pp. 211–222, Springer, 2008.
10. T. Alladi, V. Chamola, and S. Zeadally, "Industrial control systems: Cyberattack trends and countermeasures," Computer Communications, 2020.
11. F. Daryabar, A. Dehghantanha, N.I. Udzir, S. bin Shamsuddin, et al., "Towards secure model for scada systems," in Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), pp. 60–64, IEEE, 2012.
12. T. Wu, J.F.P. Disso, K. Jones, and A. Campos, "Towards a scada forensics architecture," in 1st International Symposium for ICS SCADA Cyber Security Research 2013 (ICS-CSR 2013) 1, pp. 12–21, 2013.
13. D. Upadhyay and S. Sampalli, "Scada (supervisory control and data acquisition) systems: Vulnerability assessment and security recommendations," Computers Security, vol. 89, p. 101666, 2020.
14. M. Gaiceanu, M. Stanculescu, P.C. Andrei, V. Solcanu, T. Gaiceanu, and H. Andrei, "Intrusion detection on ics and scada networks," in Recent Developments on Industrial Control Systems Resilience, pp. 197–262, Springer, 2020.
15. M.G. Cimino, N. De Francesco, F. Mercaldo, A. Santone, and G. Vaglini, "Model checking for malicious family detection and phylogenetic analysis in mobile environment," Computers Security, vol. 90, p. 101691, 2020.
16. L. Brunese, F. Mercaldo, A. Reginelli, and A. Santone, "Formal methods for prostate cancer gleason score and treatment prediction using radiomic biomarkers," Magnetic resonance imaging, vol. 66, pp. 165–175, 2020.
17. L. Brunese, F. Mercaldo, A. Reginelli, and A. Santone, "An ensemble learning approach for brain cancer detection exploiting radiomic features," Computer methods and programs in biomedicine, vol. 185, p. 105134.

18. L. Brunese, F. Mercaldo, A. Reginelli, and A. Santone, “Prostate gleason score detection and cancer treatment through real-time formal verification,” IEEE Access, vol. 7, pp. 186236–186246, 2019.
19. F. Mercaldo, F. Martinelli, and A. Santone, “Real-time scada attack detection by means of formal methods,” in 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 231–236, IEEE, 2019.
20. R. Alur and D. Dill, “Automata for modeling real-time systems,” in International Colloquium on Automata, Languages, and Programming, pp. 322–335, Springer, 1990.
21. R. Alur and D.L. Dill, “A theory of timed automata,” Theoretical computer science, vol. 126, no. 2, pp. 183–235, 1994.
22. G. Behrmann, K.G. Larsen, O. Moller, A. David, P. Pettersson, and W. Yi, “Uppaal-present and future,” in Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228), vol. 3, pp. 2881–2886, IEEE, 2001.
23. G. Behrmann, A. David, and K.G. Larsen, “A tutorial on uppaal 4.0,”Department of computer science, Aalborg university, 2006.
24. G. Behrmann, A. David, and K.G. Larsen, “A tutorial on uppaal,” in Formal methods for the design of real-time systems, pp. 200–236, Springer, 2004.
25. P. Bouyer, “Model-checking timed temporal logics,” Electronic Notes in Theoretical Computer Science, vol. 231, pp. 323–341, 2009.
26. E.M. Clarke, E.A. Emerson, and A.P. Sistla, “Automatic verification of finite-state concurrent systems using temporal logic specifications,” ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 8, no. 2, pp. 244–263, 1986.
27. N.D. Francesco, G. Lettieri, A. Santone, and G. Vaglini, “Heuristic search for equivalence checking,” Software and System Modeling, vol. 15, no. 2, pp. 513–530, 2016.
28. H.E. Jensen, K.G. Larsen, and A. Skou, “Scaling up uppaal,” in International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, pp. 19–30, Springer, 2000.
29. J. Dougherty, R. Kohavi, and M. Sahami, “Supervised and unsupervised discretization of continuous features,” in Machine Learning Proceedings 1995, pp. 194–202, Elsevier, 1995.
30. F. Mercaldo and A. Santone, “Deep learning for image-based mobile malware detection,” Journal of Computer Virology and Hacking Techniques, pp. 1–15, 2020.
31. A. De Lorenzo, F. Martinelli, E. Medvet, F. Mercaldo, and A. Santone, “Visualizing the outcome of dynamic analysis of android malware with vizmal,” Journal of Information Security and Applications, vol. 50, p. 102423, 2020.
32. R. Taormina, S. Galelli, N.O. Tippenhauer, A. Ostfeld, and E. Salomons, “Assessing the effect of cyber-physical attacks on water distribution systems,” in World Environmental and Water Resources Congress 2016, pp. 436–442, 2016.
33. E. Salomons and A. Ostfeld, “Simulation of cyber-physical attacks on water distribution systems with epanet,” in Proceedings of the Singapore Cyber-Security Conference (SGCRC) 2016: Cyber-Security by Design, vol. 14, p. 123, 2016.