Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 621203, 10 pages
http://dx.doi.org/10.1155/2013/621203

Research Article

A Novel Algorithm for Intrusion Detection Based on RASL Model Checking

Weijun Zhu,[1,2] Qinglei Zhou,[1] Weidong Yang,[2] and Haibin Zhang[3]

[1] *School of Information Engineering, Zhengzhou University, Zhengzhou, Henan 450001, China*
[2] *MOE Key Laboratory of Grain Information Technology & Control, Henan University of Technology, Zhengzhou, Henan 450001, China*
[3] *School of Computer Science, Xidian University, Xi'an, Shaanxi 710071, China*

Correspondence should be addressed to Weijun Zhu; zhuweijun76@163.com

Received 29 November 2012; Revised 6 February 2013; Accepted 15 February 2013

Academic Editor: Jitao Sun

The interval temporal logic (ITL) model checking (MC) technique enhances the power of intrusion detection systems (IDSs) to detect concurrent attacks due to the strong expressive power of ITL. However, an ITL formula suffers from difficulty in the description of the time constraints between different actions in the same attack. To address this problem, we formalize a novel real-time interval temporal logic—real-time attack signature logic (RASL). Based on such a new logic, we put forward a RASL model checking algorithm. Furthermore, we use RASL formulas to describe attack signatures and employ discrete timed automata to create an audit log. As a result, RASL model checking algorithm can be used to automatically verify whether the automata satisfy the formulas, that is, whether the audit log coincides with the attack signatures. The simulation experiments show that the new approach effectively enhances the detection power of the MC-based intrusion detection methods for a number of telnet attacks, p-trace attacks, and the other sixteen types of attacks. And these experiments indicate that the new algorithm can find several types of real-time attacks, whereas the existing MC-based intrusion detection approaches cannot do that.

1. Introduction

Intrusion detection (ID) is an important network security technique. ID can be divided into anomaly intrusion detection and misuse intrusion detection in terms of the different principles of ID. The former can find unknown types of attacks. However, false positives rate of anomaly intrusion detection is often very high. In contrast, a misuse intrusion detection system has a comparatively low false positives rate with regard to known types of attacks. This is due to the principle of misuse intrusion detection: IDS developers predefine their known types of attacks, use appropriate language to describe these types, and establish libraries of attack patterns (called misuse signatures). The system will monitor the audit log. Once a data stream in the log is found to match with certain attack type, it means that an attack is found.

However, such a class of detection methods based on pattern matching (PM) suffers from their inherent problems. First, affected by intruders' subjective wishes or other random factors, the logical relationship among its atomic actions associated with attacks of the same pattern launched by different intruders may present different features [1, 2], where an atomic action means a minimum operation step in an attack. It is hard to depict precisely so vastly different attacks with a relatively small-scale attack pattern library. Second, a large-scale coordinated attack requires an intrusion detection algorithm to handle a large volume of network data in a short period of time. To address these issues, a series of intrusion detection methods based on model checking have been developed.

A relatively comprehensive algorithm has been presented, and it is based on linear temporal logic (LTL) model checking [1]. Its basic principle can be formulated as follows: (1) use an LTL formula to describe an attack pattern as well as an automaton to record what happened in the audit log, and (2) use a model checking algorithm to check whether the automaton satisfies the formula (i.e., whether the records in

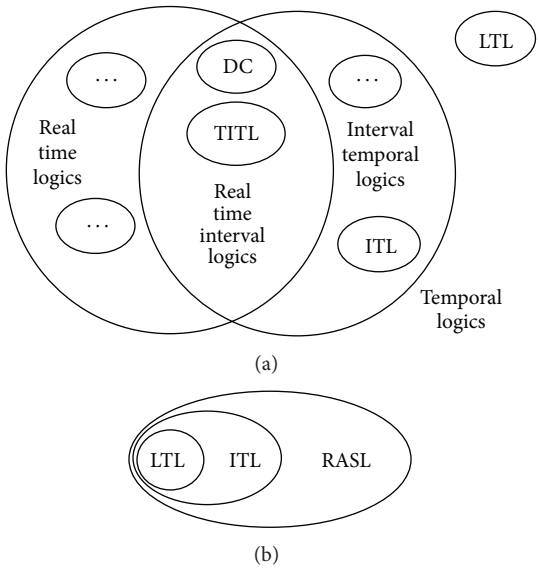


Figure 1: (a) Some temporal logics and their classification. (b) Some logics and their expressive power.

log match the attack pattern). Since current model checking algorithms have been able to check up to 10^{120} states, they are particularly suitable for the large-scale attack detection [3], and the operators in LTL formulas can flexibly describe various logical relationships between atomic attack actions.

Compared with the PM-based approaches, the MC-based ones can effectively portray the ever-changing attack patterns [1, 3]. Furthermore, the MC-based approaches have an important advantage for intrusion detection over the PM-based ones. Pattern matching is usually applicable to detect inconsistencies between data while automata, temporal logic formulas, and model checking techniques are applicable to detect inconsistencies of behaviors. Thus, the MC-based methods can do something more than the PM-based ones since intrusion attacks involve complex behaviors besides the comparatively simple data.

However, the algorithm in [1] can realize the automatic detection for neither concurrent attacks nor real-time (i.e., time constraint relation) attacks because LTL formulas cannot be used to describe multiprocess activities or time constraint relationships between attack actions or attack action sequences. As the first attempt to address these issues, a method based on ITL model checking was presented in [2], and it can describe and detect concurrent attacks, since ITL has more power than LTL. However, ITL-model-checking-based methods still cannot describe and detect real-time attacks. For example, there are a large number of attacks with the following characteristic in a real network intrusion: No more than n seconds after action (sequence/process) A occurs, action (sequence/process) B occurs. Here, the condition "no more than" can be replaced by more than, less than, no less than, or equal to. The existing MC-based algorithms cannot find these attacks.

Therefore, motivated by addressing both concurrent attacks and real-time attacks simultaneously, we, in this paper, present a new interval temporal logic to describe conveniently the real-time attack signatures and also put forward a new MC-based approach to automatically detect the various changing modes of real-time attacks.

We conducted some simulation experiments and a benchmark test (see Section 7). The detection of several groups of attacks, such as telnet attacks and p-trace attacks, is simulated on MATLAB. The experiment results verify that (1) the new algorithm finds more attacks than the existing MC-based algorithms; (2) the new algorithm finds real-time attacks. This is the main contribution of this paper.

The remainder of this paper is organized as follows. Section 2 illustrates some related works and compares them with the new approach. Section 3 defines a new logic, RASL, and gives its formal syntax and semantics. Section 4 uses RASL formulas to establish some models for attack patterns. Several examples of models are given in Section 5. Section 6 formalizes a RASL model checking algorithm based on a new data structure called timed normal form graph (TNFG), and a misuse intrusion detection algorithm is presented. Section 7 presents several groups of experiments and compares the new algorithm with the existing ones with regard to the description capabilities and detection capabilities for intrusion attacks. Section 8 draws the conclusions of this paper.

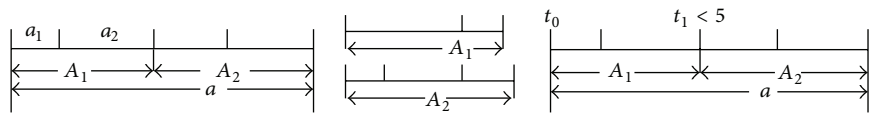
2. Related Works

2.1. Detect Various Attack Types Using Model Checking Linear Temporal Logics. A tool called ORCHIDS was developed [3], which fulfilled the LTL-model-checking-based method for intrusion detection in reality [1]. In one experiment, ORCHIDS found some p-trace attacks [4] which usually exploit the flaws in process calls to inject malicious code. It is difficult for traditional intrusion detection systems to find this type of attacks because they only match individual events [4]. The ORCHIDS was improved in [5]. In a real environment, it successfully detected a series of wireless network attacks [5], including deauthentication flooding, rogue access points, and Chop-Chop. This is the first IDS to successfully detect Chop-Chop attacks [5]. Furthermore, to avoid repeated verifications needed by the algorithm in [1], an improved algorithm was put forward in [6], which is able to compute the number of guesses in password attacks.

Compared with the methods mentioned above, the new algorithm can be used to detect complex concurrent attacks and real-time attacks (See Section 7).

2.2. Detect Various Attack Types Using Model Checking Interval Temporal Logics. ITL was put forward in [7]. With its successful and broader adoption and adaptation [8–11], ITL is becoming a class of logics, including some non-real-time interval logics [7] and some real-time interval logics [12–14]. Figure 1(a) illustrates the relationships between some temporal logics.

There are some studies that use interval temporal logics to describe attack patterns so that more intrusion behaviors can be expressed [15–17]. However, these papers do not mention how to detect these attacks automatically. The method presented in [2] can do it automatically, but it can



(a) Sequential relation $a = A_1, A_2$ (b) Concurrent relation $a = A_1 \mid A_2$ (c) Time relation

FIGURE 2: Different relationship between behaviors in an attack a.

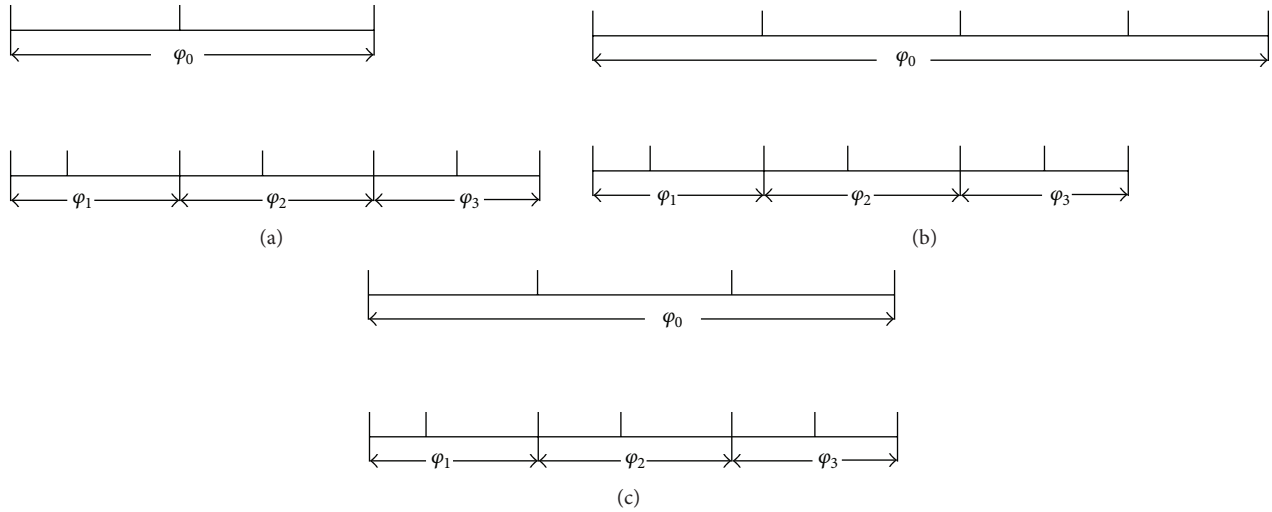


FIGURE 3: Three cases on different length of projection.

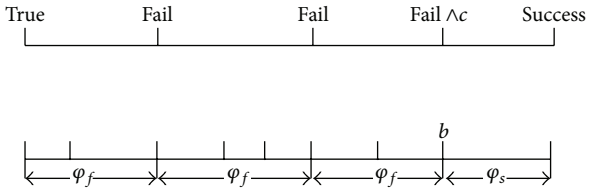


FIGURE 4: Success after connection failed three times.

only find concurrent attacks rather than real-time attacks. In contrast, as a real-time interval logic, RASL has more expressive power (see Figure 1(b)), which can be used to describe the time relationships among attack activities, and our model checking algorithm can find real-time intrusion attacks in a fully automatic manner (See Section 7).

3. RASL

Definitions 1 and 2 give the formal description of the syntax of RASL, whereas the other definitions present its semantics. Compared with ITL [11, 18, 19], the additional operator denoted as "$;_I$" in RASL is appended for the description of time constraints between intervals.

Definition 1. RASL formulas have the following syntax given in the Backus-Naur form:

(1) terms $t ::= T \mid T_f$,

(2) constraint formulas $\delta ::= T_f \le c \mid T_f < c \mid T_f > c \mid T_f \ge c \mid \delta_1 \wedge \delta_2$,

(3) interval formulas $\varphi ::= p \mid skip \mid (\varphi_1, \ldots, \varphi_m)\ prj\ \varphi_0 \mid (\varphi_1, \ldots, (\varphi_i, \ldots, \varphi_j)^{\Theta}, \ldots, \varphi_m) prj\ \varphi_0 \mid \varphi_1; \varphi_2 \mid \varphi^* \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 || \varphi_2$,

(4) timed formulas $\psi ::= \varphi \mid \delta \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \psi_1; \psi_2$

Definition 2. The derived formulas are defined as follows:

(1) $\bigcirc p ::= skip; p$, (2) $more ::= \bigcirc true$, (3) $empty ::= \bigcirc false$, (4) $p;_I q ::= (p \wedge T_f \in I); q$

Definition 3. A state s is a tuple (a, t), where $a = \{x \mid x \in AP \wedge x = true\}$ and $t : T \to N$ denotes the absolute time of the current state.

Definition 4. A timed sequence of states is defined and also denoted as $\sigma = \langle s_0, s_1, \ldots, s_i, \ldots \rangle$, where s_i is a state.

Definition 5. An interpretation is a quadruple $I = (\sigma, i, k, j)$, where σ is a timed sequence of states over $\langle s_i, \ldots, s_k, \ldots, s_j \rangle$, $i, k, j \in N$, and s_k is the current state. We use the notation $len(\sigma) = |\sigma| = j - i$ for the number of states in interval and $len_t(\sigma) = s_t(j) - s_t(i)$ for the time distance between the endpoints in interval, where $s_t(i)$ is the absolute time of state s_i.

Definition 6. Let $\sigma = \langle s_0, s_1, \ldots, s_{|\sigma|} \rangle$ an interval, $r_1, \ldots, r_h$ be integers, and $0 \le r_1 \le r_2 \le \cdots \le r_h \le |\sigma|$. We use notation $\sigma \downarrow (r_1, \ldots, r_h) = \langle s_{t1}, s_{t2}, \ldots, s_{tl} \rangle$ to denote a projection from σ to

Table 1: The way of constructing models for attacks: behaviors and their RASL formulas.

Attack behaviors	RASL formulas defined for description of behaviors
a_j means an atomic action	A state formula w_j
$A_i = a_1, \ldots, a_m$ means a phase in an attack	$\varphi_i = ((w_1 \wedge skip); true); \ldots; ((w_m \wedge skip); true)$
$a = A_1, \ldots, A_n$ means an attack	$\varphi = (\varphi_1 \wedge skip); true; \ldots; (\varphi_n \wedge skip); true$
a_j is repeated k times	$\varphi = len(k) \wedge \left(w_j \wedge skip\right)^*$
A_i is repeated k times	$\varphi = ((\varphi_i \wedge skip)^{\Theta}, b)\ prj\ (((true \wedge skip)^* \wedge len(k+1); c)) \wedge \Box(b \leftrightarrow c)$
Result of a_j is $tf_j, tf_j \in \{fail, success\}$	$\varphi = (w_j \wedge empty)\ prj\ tf_j$
$a = A_1, \ldots, A_n$, where result of A_i is tf_i	$\varphi = \left((\varphi_1 \wedge skip); true, \ldots, (\varphi_n \wedge skip); true\right)\ prj\ (tf_1 \wedge \bigcirc tf_2 \wedge \ldots \wedge \bigcirc^{n-1} tf_n)$
$a = A_1 \mid \ldots \mid A_n$	$\varphi = \varphi_1 \parallel \ldots \parallel \varphi_n$
A_1 and A_2 are two ways of the attack	$\varphi = \varphi_1 \vee \varphi_2$
Time constraint between A_i and A_{i+1} is I_i	$\varphi = \varphi_1;_{I_1} \ldots;_{I_{n-1}} \varphi_n$

Table 2: Some notations presented in Algorithm 1.

Notations	Semantics
$Unext(P)$	The set of all subformulas which do not occur in the domain of any operator next of normal form of $P \in L_{\mathrm{RASL}}$
$pre(N)$	All the nodes which are parents of N in G, where G is a TNFG and N is a node of G
$Sub(P)$	The set of all subformulas of $P \in L_{\mathrm{RASL}}$
$Unchop(P)$	A Boolean variable and $Unchop(P) = true$ if and only if there exists no subinterval in $P \in L_{\mathrm{RASL}}$

$r_1, \ldots, r_h$, where $t_1, \ldots, t_l$ is obtained by deleting the duplicate numbers from $r_1, \ldots, r_h$.

Definition 7. Let $c \in N$ and $s_p(k)$ be the true value of $p \in AP$ in state s_k. The satisfaction relation is inductively defined as follows:

(1) $T = s_t(k)$,

(2) $T_f = s_t(j) - s_t(k)$,

(3) $I \vDash T_f \leq C$ if and only if $s_t(j) - s_t(k) \leq C$,

(4) $I \vDash T_f \geq C$ if and only if $s_t(j) - s_t(k) \geq C$,

(5) $I \vDash T_f < C$ if and only if $s_t(j) - s_t(k) < C$,

(6) $I \vDash T_f > C$ if and only if $s_t(j) - s_t(k) > C$,

(7) $I \vDash \delta_1 \wedge \delta_2$ if and only if $I \vDash \delta_1$ and $I \vDash \delta_2$,

(8) $I \vDash p$ if and only if $s_p(k) =$ true,

(9) $I \vDash \varphi_1 \wedge \varphi_2$ if and only if $I \vDash \varphi_1$ and $I \vDash \varphi_2$,

(10) $I \vDash \varphi_1 \vee \varphi_2$ if and only if $I \vDash \varphi_1$ or $I \vDash \varphi_2$,

(11) $I \vDash \varphi_1; \varphi_2$ if and only if $\exists r, k \leq r \leq j$, such that $(\sigma, i, k, r) \vDash \varphi_1$ and $(\sigma, i, k, r) \vDash \varphi_2$,

(12) $I \vDash skip$ if and only if $len(\sigma) = 1$,

(13) $I \vDash \varphi_1 || \varphi_2$ if and only if $\varphi_1 \wedge (\varphi_2; true) \vee \varphi_2 \wedge (\varphi_1; true)$,

(14) $I \vDash \varphi^*$ if and only if (i) there exist finite many $r_0, \ldots, r_n \in N_\omega$, such that $k = r_0 \leq r_1 \leq \cdots \leq r_{n-1} \leq r_n = j$, $(\sigma, i, k, r_0) \vDash \varphi$, and for every $1 \leq l \leq n$, $(\sigma, r_{l-1}, r_{l-1}, r_l) \vDash \varphi_0$ (ii) or $k = j$,

(15) $I - prj : I \vDash (\varphi_1, \ldots, \varphi_m)\ prj\ \psi$, if and only if there exist integers $k = r_0 \leq r_1 \leq \cdots \leq r_m \leq j$ and $(\sigma, i, k, r_1) \vDash \varphi_1, \ldots, (\sigma, r_{n-1}, r_{n-1}, r_n) \vDash \varphi_n$, where $1 < n \leq m$, such that for σ' in the two cases mentioned below, we have $(\sigma', 0, 0, |\sigma'|) \vDash \psi$ — (i) $r_m < j$ and $\sigma' = \sigma \downarrow (r_0, \ldots, r_m) \cdot \sigma(r_m + 1, \ldots, j)$, (ii) $r_m = j$ and $\sigma' = \sigma \downarrow (r_0, \ldots, r_h), 0 \leq h \leq m$,

(16) $I \vDash (\varphi_1, \ldots, (\varphi_i, \ldots, \varphi_j)^{\Theta}, \ldots, \varphi_m)\ prj\ \varphi_0$ if and only if there exists $n \in N_0$, such that $I \vDash (\varphi_1, \ldots, (\varphi_i, \ldots, \varphi_j)^n, \ldots, \varphi_m) prj\ \varphi_0$,

(17) $I \vDash \psi_1 \wedge \psi_2$ if and only if $I \vDash \psi_1$ and $I \vDash \psi_2$,

(18) $I \vDash \psi_1 \vee \psi_2$ if and only if $I \vDash \psi_1$ or $I \vDash \psi_2$,

(19) $I \vDash \psi_1; \psi_2$ if and only if $\exists r, k \leq r \leq j$, such that $(\sigma, i, k, r) \vDash \psi_1$ and $(\sigma, i, k, r) \vDash \psi_2$,

4. Construct Signatures with RASL Formulas

We can use RASL formulas to construct signatures, that is, specifications of attack patterns. Compared with linear temporal logic, RASL has been additionally equipped with interval semantics. So, a phase, that is, a sequence of atomic actions, in an attack can be described with an interval in a RASL formula, while various steps in the phase can be described with various points in the interval [2]. Temporal relationship between steps in an attack can be described with temporal operators. Logical relationship between various phases can be described with operator ";" [2]. And a concurrent attack can be described with a formula with the operator "$\parallel$". Compared with ITL, RASL can express more. Particularly, repeated attacks can be described with operator "$*$" or "$prj\Theta$", and a time constraint between phases or steps in an attack can be described with operator "$;_I$".

Table 3: A comparison of different MC-based approaches for detecting telnet attacks.

Attack actions\detection results (if attacks are found)	ITL [2]	RASL	LTL [1]
Make a backdoor after/before/while close firewall	Yes	Yes	No
There exists an/no overlap timed interval between two actions	Yes	Yes	No
The time distance between two actions is less than n seconds	No	Yes	No

Table 4: Another comparison of different MC-based approaches for detecting telnet attacks.

Attack actions\detection results (if attacks are found)	ITL [2]	RASL	LTL [1]
Two attacks are launched at the same time point	Yes	Yes	No
The same attack is launched repeatedly n times (n is a variable)	Yes	Yes	No
The same attack is launched repeatedly k times (k is a given large constant)	Yes	Yes	No
The time distance between two actions is less than m seconds	No	Yes	No

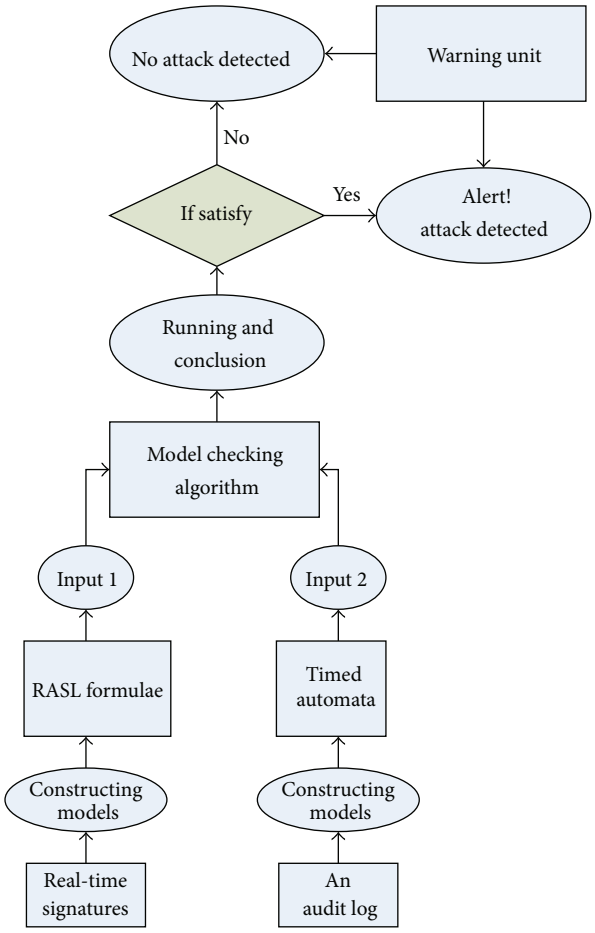


Figure 5: Algorithm for intrusion detection based on RASL model checking (the main idea of this paper).

Table 1 presents how to construct formal models for intrusion attacks with RASL formulas. And Figure 2 illustrates sequential relationships, concurrent relationships, and time relationships between behaviors in an attack.

Definition 8 (See [1]). A record in a log library is modeled by a finite state automaton A.

Theorem 9. *A record of a log can be modeled by a timed automaton A'.*

Proof. According to Definition 8, we know that a record of a log can be modeled by a finite state automaton A. For every transition e of A, we add time constraint "*true*". For every state s of A, we extend s to (s, t), where t denotes absolute time. So, finite state automaton A is turned to timed automaton A'. The theorem holds. □

5. A Case Study

As a case study, we discuss several examples to show the expressive capability of the above proposed models.

Example 10. Password cracking inconsecutive attack: failure. The RASL formula is

$$((connect \wedge empty\ \ prj\ \ fail)\,; true)^{*}, \tag{1}$$

where connect means that an intruder is trying to connect. The intruder could launch another concurrent process before the end of current connection process. Thus, the subinterval that describes current execution of the concurrent connection process is over, and it can be described with operators *before prj*. The sub-interval that describes the result is over while this connection process fails, and it can be described with the operator *fail after prj*. The intruder repeatedly tries connection, and it can be described with "$*$". Inconsecutive phenomenon between connections can be described with "$; true$".

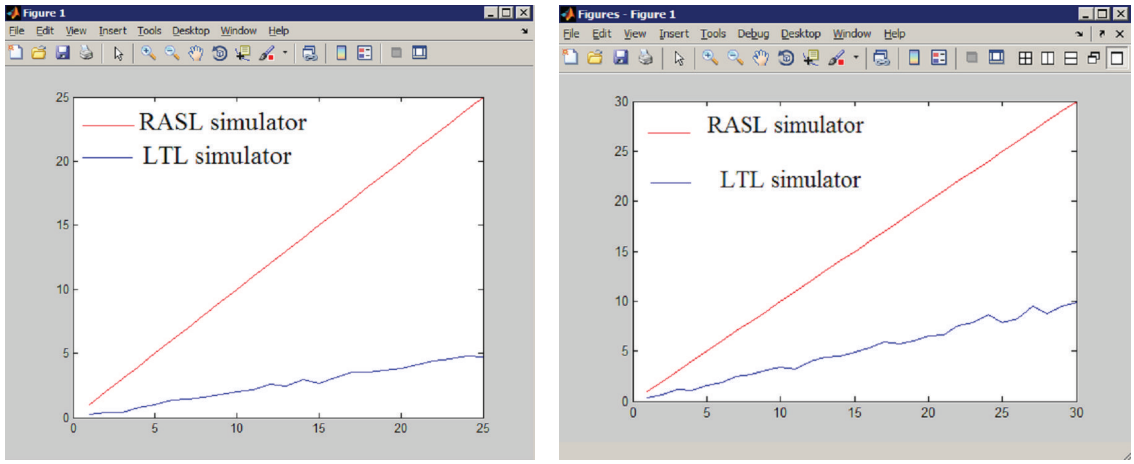
Example 11. Password cracking inconsecutive attack: success after connection failed $k - 1$ times.

At first, one time failure in connection can be described as $\varphi_f := (connect \wedge empty\ prj\ fail); true$. And, then, a successful trial can be described as $\varphi_s := connect \wedge empty\ prj\ success$.

The formula that describes $k - 1$ times failures in connections can be defined as $\varphi_{f'} := (\varphi_f{}^{\Theta}, b)\, prj((true; skip); ((fail \wedge skip)^{*} \wedge (len(k-1)); c)) \wedge \Box(b \leftrightarrow c)$.

The formula that describes the attack can be defined as

$$\varphi := \left(\varphi_{f'} \wedge skip\right); \varphi_s. \tag{2}$$



(a) (b)

Figure 6: Comparisons of average number of attacks found by different MC-based approaches (x: the number of kinds of simulation attacks and y: the different number of kinds of attacks found by the different simulators). (a) For telnet attacks. (b) For p-trace attacks.

Table 5: Comparison of different MC-based approaches for detecting password attacks.

Attack actions\detection results (if attacksare found)	ITL [2]	RASL	LTL [1]
Consecutive/inconsecutive attack: success after connection failed $k-1$ times (k is a given large constant)	Yes	Yes	No
Every time distance between m attacks is less than n seconds (k is a given small constant)	No	Yes	No

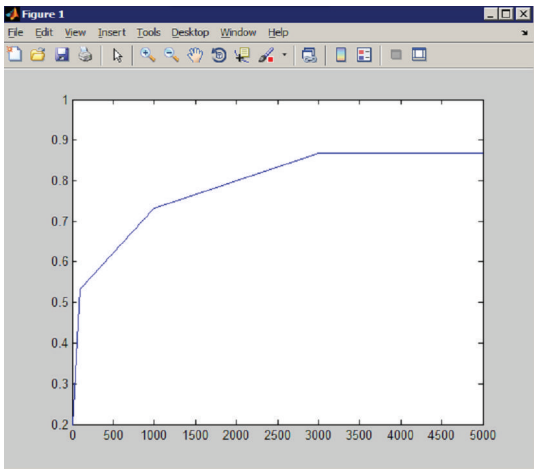


Figure 7: A comparison of detection ability for p-trace attacks using different MC-based approaches (x: the average time distance between atomic attack actions and y: the average number of attacks found by ITL simulator/average number of attacks found by RASL simulator).

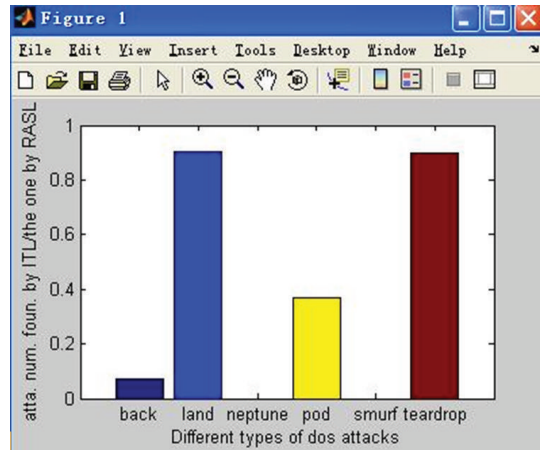


Figure 8: Comparison for DOS attacks.

As shown in Figure 3, the definition of $\varphi_{f'}$ is illustrated, where $k = 4$, that is, $\varphi_{f'}$ denotes three times failures in connections. As shown in (a), (b), and (c) of Figure 3, there are three cases on the length of interval $(\varphi_1, \varphi_2, \varphi_3)$ prj φ_0 in RASL formula. In each of the $k-1$ failures in connections, there exists a one-to-one map between attack actions and their results. That is to say, the number of φ_fs which describe attack actions is equal to the number of $(fail)$s that describe their results. This number is three, so only (c) of Figure 3 is correct. To this end, we can append atomic proposition b to the formula, and let b follow $\varphi_f{}^{\Theta}$. Furthermore, we can append atomic proposition c to the formula when subinterval $fail^*$ is over. The number of φ_fs is equal to the number of $(fail)$s if $\Box(b \leftrightarrow c)$ holds, as shown in (c) of Figure 3.

Subinterval $fail^*$ is executed repeatedly $k-1$ times to guarantee $k-1$ times cycles of φ_f, as shown in Figure 4. We need two states in current subinterval $fail$ to make sure that the first state of the next subinterval $fail$ is the next state of the final state of the current subinterval. So, we replace $fail^*$ with $(fail \wedge skip)^*$.

Example 12. Phases of a telnet attack are observed as follows.

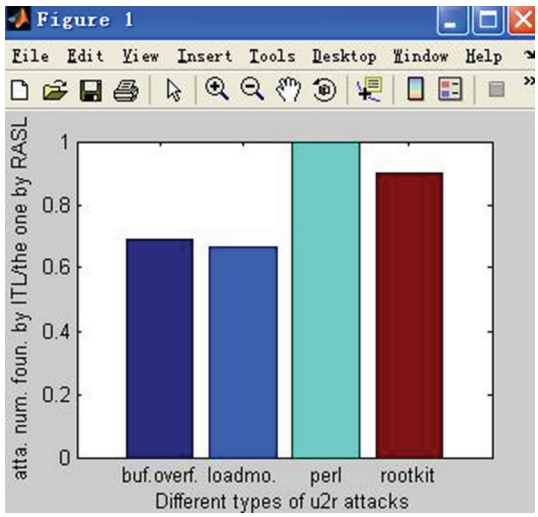


Figure 9: Comparison for U2R attacks.

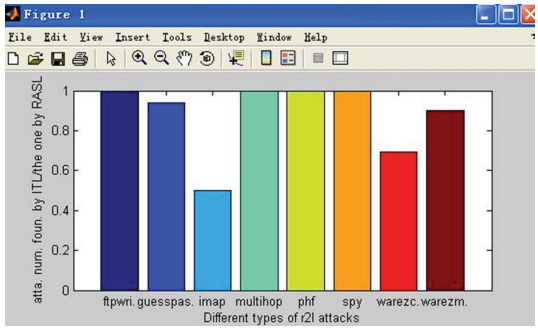


Figure 10: Comparison for R2L attacks.

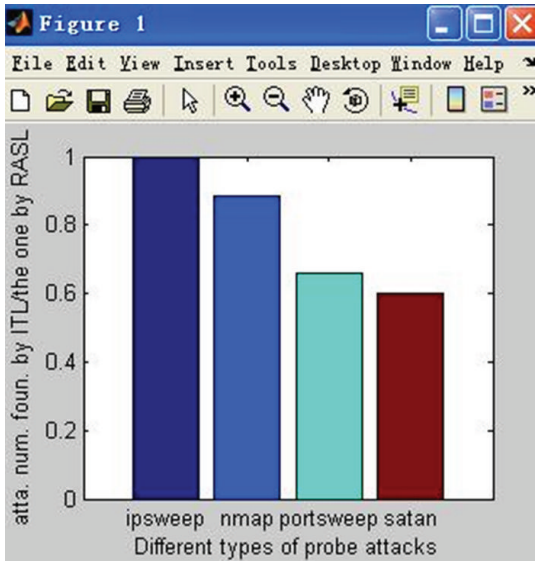


Figure 11: Comparison for Probe attacks.

Phase 1: the telnet service is started, and it is described as atomic formula q.

Phase 2: the intruder closes firewall. There are three steps in this phase. At first, the intruder accesses C: *windows* in order to find program *aproman*. It is described as RASL atomic formula s_1. And, then, the intruder executes command *aproman-a*[PID] and monitors all processes in order to find PID of firewall process. It is described as RASL atomic formula s_2. At last, the intruder executes command *aproman-t*[PID] to close firewall. It is described as RASL atomic formula s_3. The intruder performs the three steps of this phase in sequence with a gap between each step. Each of the two delays is less than n seconds, and it is described as $;_{I_1 \in [0,n)}$.

Phase 3: in order to login the system again in the future, the intruder makes a backdoor. There are two steps in this phase. The first step is to access directory in which file *instsrv.exe exists*, and step 2 is to execute command *instsrv.exe SYSHE-ALTHC* : *windows* \ *SYSTEM*32 \ $t\ln tsvr.exe$ in order to setup *SYSHEAHTH* service which is a backdoor. The former can be denoted as a RASL atomic formula t_1, and the later can be denoted as a RASL atomic formula t_2. The intruder performs the two steps of this phase in sequence with a gap between each step. The delay is less than m seconds, and it is described as $;_{I_2 \in [0,m)}$.

In summary, the timed formula for the telnet attack is formulized as follows:

$$\begin{aligned} &(((q \wedge empty); true);\ (((((s_1 \wedge empty); true);_{I_1 \in [0,n)}\ , \\ &((s_2 \wedge empty); true);_{I_1 \in [0,n)}\ ((s_3 \wedge empty); true))\|\ , \\ &(((t_1 \wedge empty); true);_{I_2 \in [0,m)}\ ((t_2 \wedge empty); true)))). \end{aligned} \tag{3}$$

In Formula (3), ";" is used to express a piecewise action, "$\|$" is used to express a concurrent action, and "$;_I$" is used to express a time constraint relationship.

6. RASL Model Checking Algorithm and Intrusion Detection Algorithm

We can give a subset of RASL called ASL, which is obtained by deleting all of the time constraints in RASL. Reference [18] gives a data structure called normal form graph (NFG) as well as a procedure called PRO(P) to construct the NFG model denoted as G_P for an ASL formula P. Thus, an ASL model checking algorithm was obtained in [18]. Based on this work, we can obtain a RASL model checking algorithm and its intrusion detection algorithm.

First, Definition 13 presents a data structure called TNFG, which is a timed version of NFG.

Definition 13. For a formula $P \in L_{\mathrm{RASL}}$, the TNFG of P is defined as a tuple $G = (\mathrm{CL}(P), \mathrm{EL}(P), C)$, where C is a finite

```
TNFG(P)
Begin
  PRO(Untime(P)); CL(P):= CL(Untime(P)); EL(P):= EL(Untime(P));
            /* produce the NFG in which no clock constraint exists, where CL(Untime(P)) is the set of nodes
            of the NFG of ASL formula Untime(P), and EL(Untime(P)) is the set of edges of the NFG of ASL
            formula Untime(P)
  T := φ /* T is defined as a set, and it consists of all the nodes which have been converted from NFG to TNFG.
  for all Qs of CL(P) in the order of building NFG's nodes, do the following:
    T := T ∪ {Q}
    for all M; N ∈ Unext(Q) and Unchop(M) and M; N ∉ Sub(t ∈ T)
    and M;_I N ∈ Sub(P), do   /* for all the clock constraints holding the conditions
      new(x); /* allocate a new clock
      for all EL(Q) = (Q, Q_e, ε), do EL(Q) := (Q, Q_e, ε, I_{x=0}, {x})
            /* I_{x=0} means that constraint I is satisfied when x = 0, where I = δ
      for all EL(Q) = (Q, Q_i, Q_i'), do EL(Q) := (Q, Q_i, Q_i', true, {x})
            /* the edges of NFG is converted to the ones of TNFG
      for all EL(pre(w ∧ N)) do /* the current interval is over, where w is a state formula which may be
      empty or not.
      if EL(pre(w ∧ N)) = (R, R_i, R_i') then EL(pre(w ∧ N)) := (R, R_i, R_i', I_x, {x})
        /* in the TNFG, the clock constraint is appended if it is satisfied.
      end for
    end for
  end for
  for all of e in EL(P) do   /* for all the converts which deal with no clock constraint
    if e = (Q, Q_i, Q_i'), e := (Q, Q_i, Q_i', true, φ)   /* the edges of NFG is converted to the ones of TNFG
  end for
end TNFG
```

Algorithm 1: Translation from a RASL formula to a TNFG.

clock set, and the set CL(P) of nodes and the set EL(P) of edges are inductively defined as follows:

(1) $Untime(P) \in \mathrm{CL}(P)$, where $Untime(P)$ is an ASL formula in which all *CHOPIs* in P are replaced by *CHOP*,

(2) for every $Q \in \mathrm{CL}(P) \setminus \{\varepsilon, false\}$, if $Q \equiv Q_e \wedge empty \vee V_{i=1}^{r}(Q_i \wedge \bigcirc Q_i')$, $\varepsilon \in \mathrm{CL}(P)$, $(Q, Q_e, \varepsilon, \delta, \lambda) \in \mathrm{EL}(P)$, and for every i, $1 \le i \le r$, we have $Q_i' \in \mathrm{CL}(P)$, $(Q, Q_i, Q_i', \delta, \lambda) \in \mathrm{EL}(P)$, where set λ gives the clocks to be reset and δ is a clock constraint,

(3) CL(P) and EL(P) are produced by (1) and(or) (2) only.

Second, Algorithm 1 constructs TNFG models for RASL formulas. Some notations presented in the algorithm are explained in Table 2.

Third, if we append accepted conditions to TNFGs, we will obtain discrete timed automata models of RASL formulas. It is illustrated by Algorithm 2.

Algorithm 2 gives a procedure to compute discrete timed automaton A, that is, the model of RASL formula P. The model of P is the formal language accepted by the automaton.

Last, we can use A to describe an attack signature and another discrete timed automaton B to a record of the audit log. If $\neg A \cap B = \phi$, the result of model checking algorithm is that B satisfies P, else the result is that B does not satisfy P. We can surely say that IDS finds an attack if B satisfies P. Thus, the intrusion detection algorithm is obtained, as shown in Figure 5.

The inherent complexity of interval temporal logic model checking problem is nonelementary. The number of exponential order is proportional to the number of embedded not operators. The approach based on an NFG or TNFG reaches the lower bound of this problem [14, 19]. There is only one occurrence of the operator not in the new model checking algorithm. So, both the inherent complexity of the intrusion detection problem based on RASL model checking and the complexity of our algorithm, in the worst case, are exponential.

7. Simulation Experiments

In order to compare the existing approaches with our new algorithm, we conducted experiments by simulating and detecting telnet attacks and password attacks mentioned above as well as other types of attacks. The platform used is a PC with Dual core 3.2 GHz, 8 GB, and Windows XP SP3, along with MATLAB 2010. The results on detection ability are shown in Tables 3, 4, and 5. The different results are due to the different expressive powers of the different logics.

In order to compare the LTL-model-checking-based approaches in [1, 3, 5] with our RASL-model-checking-based algorithm, we simulate and detect some telnet attacks by using MATLAB. We randomly produce 25 kinds of telnet attacks and repeat 80 times for every of these attacks. On average, less than 5 kinds of attacks are reported by the LTL-based simulator, whereas almost 100 percent of these attacks are found by the RASL-based simulator, as shown in

function CONSTRUCT (G)
/* pre-condition: $G = (\mathrm{CL}(P), \mathrm{EL}(P), X)$ is an TNFG of RASL formula P */
/* post-condition: CONSTRUCT constructs a timed automaton from TNFG of formula P */
begin function
$S := \{P\}; S_0 := \{P\}; \Sigma := \phi; E := \phi;$
$\mathrm{CL} := \mathrm{CL}(P)$ /* CL denotes acceptance state set */
For all $Q \in \mathrm{CL}(P)$ do If Q is labeled F then $\mathrm{CL} := \mathrm{CL}/\{Q\}$ /*Acceptance state set doesn't contain the circulate nodes passed by finite times */
$\mathrm{CL} := \mathrm{CL}/\{\varepsilon\}$ /*Acceptance state set doesn't contain the final states */
while $\exists el = (Q, Q_{ci}, Q_{ni}, \delta, \lambda) \in \mathrm{EL}(P)$ /*for every non-terminal edge of TNFG*/
do $s := Q; s' := Q_{ni}; a := Q_{ci}; e := (s, s', a, \delta, \lambda); E := E \cup \{e\};$ /*add transition rules to timed automaton*/
$S := S \cup \{s, s'\}; \Sigma := \Sigma \cup \{a\};$ /*add state and input alphabet to timed automaton
end while
while $\exists el = (Q, Q_e, \varepsilon, \delta, \lambda) \in \mathrm{EL}(P)$ /*for every terminal edge of TNFG*/
do $s := Q; s' := \varepsilon; a := Q_e; e := (s, s', a, \delta, \lambda); E := E \cup \{e\};$ /*add transition rule to timed automaton*/
$S := S \cup \{s, s'\}; \Sigma := \Sigma \cup \{a\};$ /*add state and input alphabet to timed automaton*/
end while
$F := \{\varepsilon\}; AC := \mathrm{CL};$ /*set of final states and set of acceptance states*/
return $A = (\Sigma, S, S_0, E, X, F, AC);$
end function
Construction Procedure (P)
(1) Build the TNFG of P, $G = (\mathrm{CL}(P); \mathrm{EL}(P), X)$, by algorithm TNFG($P$);
(2) Obtain the timed automaton, $A = (\Sigma, S, S_0, E, X, F, AC)$, by algorithm CONSTRUCT(G').

Algorithm 2: Constructing discrete timed automata for RASL formula.

Figure 6(a). The simulation results indicate that the model checking technique itself cannot make an IDS stronger, but this technique, when employing a stronger temporal logic, such as RASL, to describe attacks, can.

We simulate and detect some p-trace attacks by using MATLAB. We randomly produce 30 kinds of p-trace attacks, and repeat 100 times for each of these attacks. On average, less than 10 kinds of attacks are reported by the LTL-based simulator, whereas almost 100 percent of kinds of attacks are found by the RASL-based simulator, as shown in Figure 6(b). The results indicate that the RASL-based algorithm enhances the detection power for p-trace attacks, compared with the LTL-based algorithm. Clearly, this is due to the stronger expressive power of RASL.

Suppose that the standard time unit is a second; Figure 7 illustrates a comparison between the ITL-model-checking-based approach in [2] and our RASL-model-checking-based algorithm. We randomly produce some attacks including real-time attacks and non-real-time attacks. Compared with the ITL-based simulator, the RASL-based simulator raises the average number of detected attacks by as high as 400%, where the average time distance (or time constraints) between two atomic actions in the same real-time attack is only five seconds. The average number will still be raised by 15% even in the worst case, that is, the time distance is more than three thousand seconds. These results indicate that the RASL-based algorithm further raises the power of detection for p-trace attacks, compared with the ITL-based algorithm, again, due to the stronger expressive power of RASL.

In order to give a comparison of the detection ability for more types of attacks between the ITL-model-checking-based approach [2] and the RASL-model-checking-based one, we tried to conduct a Benchmark test on KDD CUP 99 [20]. We used a behavior version of a sample subset of this standard benchmark set [20] to evaluate our research in intrusion detection. Attacks fall into four main categories [20], that is, DOS, R2L, U2R, and Probe, including totally twenty-two types of attacks, as shown in Figures 8, 9, 10, and 11. In each of these four figures, the y-axis means the ratio between the number of attacks found by ITL-based simulator and the number of attacks found by RASL-based simulator, whereas the x-axis means different types of attacks.

As shown in the figures, all of the ratios range between 0 and 1. For some types of attacks, such as perl and ftp write, et al. the ITL-based simulator finds equal number of attacks when the new simulator does. And for other types of attacks, such as back, Neptune, and smurf, et al., the ITL-based simulator almost does nothing, whereas the RASL-based one does more. This is due to the strong expressive power of RASL again.

8. Conclusions

This paper defined a new real-time interval temporal logic—RASL. Based on it, we presented a RASL model checking algorithm and its intrusion detection algorithm. This enables us to employee MC-based approaches for detecting real-time attacks. P-trace attacks especially are hard to be detected by the existing IDS [4] except the LTL-based algorithm [1, 3], the ITL-based algorithm [2], and the new RASL algorithm. The new algorithm has detected some real-time p-trace attacks in our simulation experiments. To the best of our knowledge, this is the only method to report this type of attacks. It is the benefit of using the new approach.

Conflict of Interests

The authors certify that they have no conflict of interests with any trademark included in this paper.

Acknowledgments

The first author of this paper would like to thank Dr. Kevin Lu at Brunel University, UK, for his constructive suggestions on this paper. This work has been partially supported by the National Natural Science Foundation of China (No. 61250007, no. U1204608, no. 61003079, and no. 61202099), the China Postdoctoral Science Foundation (no. 2012M511588), the SRFDP (no. 20100203120012), and the Fundamental Research Funds for the Central Universities in China (no. K5051203019).

References

[1] M. Roger and J. Goubault-Larrecq, "Log auditing through model-checking," in *Proceedings of the 14th IEEE workshop on Computer Security Foundations (CSFW '01)*, pp. 220–234, IEEE Computer Society, Washington, DC, USA, June 2001.

[2] W. Zhu, Z. Wang, and H. Zhang, "A novel algorithm for intrusion detection based on model checking interval temporal logic," *China Communications*, vol. 8, no. 3, pp. 66–72, 2011.

[3] J. Olivain and J. Goubault-Larrecq, "The ORCHIDS intrusion detection tool," in *Proceedings of the 17th International Conference on Computer Aided Verification (CAV '05)*, Lecture Notes in Computer Science, pp. 286–290, Springer, Edinburgh, UK, July 2005.

[4] J. Goubault-Larrecq and J. Olivain, "A smell of orchids," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5289, pp. 1–20, 2008.

[5] R. Ben, G. Tremblay, and G. Bégin, "Extending orchids for intrusion detection in 802.11 wireless networks," in *Proceedings of the 8th international conference on New technologies in distributed systems (NOTERE '08)*, pp. 1–12, New York, NY, USA, June 2008.

[6] Y. Zhang, Y. Fu, and X. Sun, "A method of intrusion detection based on model-checking," *Wuhan University Journal of Natural Sciences*, vol. 51, no. 3, pp. 319–322, 2005 (Chinese).

[7] B. Moszkowski, *Reasoning about digital circuits [Ph.D. thesis]*, Department of Computer Science, Stanford University, Stanford, Calif, USA, 1983.

[8] M. Hira, "Verification of tempura specification of sequential circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 4, pp. 362–375, 1997.

[9] M. Solanki, A. Cau, and H. Zedan, "Semantically annotating reactive web services with temporal specifications," in *Proceedings of the IEEE 2nd International Workshop on Semantic and Dynamic Web Processes (ICWS '05)*, 2005.

[10] H. Bowman, H. Cameron, P. King, and S. Thompson, "Mexitl: multimedia in executable interval temporal logic," *Formal Methods in System Design*, vol. 22, no. 1, pp. 5–38, 2003.

[11] B. Moszkowski, "Using temporal logic to analyse temporal logic: a hierarchical approach based on intervals," *Journal of Logic and Computation*, vol. 17, no. 2, pp. 333–409, 2007.

[12] Z. Chaochen, C. A. R. Hoare, and A. P. Ravn, "A calculus of durations," *Information Processing Letters*, vol. 40, no. 5, pp. 269–276, 1991.

[13] Z. Duan, *Modeling of Hybrid Systems*, Science Press, Beijing, China, 2004.

[14] W. J. Zhu, H. B. Zhang, and Q. L. Zhou, "On the decidability of satisfiability of discrete TITL formulae," *Tien Tzu Hsueh Pao/Acta Electronica Sinica*, vol. 38, no. 5, pp. 1039–1045, 2010.

[15] M. G. Ouyang, F. Pan, and Y. T. Zhang, "ISITL: intrusion signatures in augmented interval temporal logic," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, vol. 3, pp. 1630–1635, November 2003.

[16] E. Nowicka and M. Zawada, "An interval temporal logic-based matching framework for finding occurrences of multi-event attack signatures," *Computer Network Security*, vol. 1, part 5–8, pp. 272–285, 2007.

[17] M. G. Ouyang and Y. B. Zhou, "ISDTM: an intrusion signatures description temporal model," *Wuhan University Journal of Natural Sciences A*, vol. 8, no. 2, pp. 373–378, 2003.

[18] Z. Duan, C. Tian, and L. Zhang, "A decision procedure for propositional projection temporal logic with infinite models," *Acta Informatica*, vol. 45, no. 1, pp. 43–78, 2008.

[19] C. Tian and Z. Duan, "Complexity of propositional projection temporal logic with star," *Mathematical Structures in Computer Science*, vol. 19, no. 1, pp. 73–100, 2009.

[20] "KDD Cup 1999 Data," 2007, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html .