Modeling Temporal Properties of Multi-event Attack Signatures in Interval Temporal Logic

Elżbieta Nowicka
Chair of Systems and Computer Networks
Wrocław University of Technology
Janiszewskiego 11/17, 50–370 Wrocław, Poland
Email: elzbieta.nowicka@plusnet.pl

Marcin Zawada
Institute of Mathematics and Computer Science
Wrocław University of Technology
Janiszewskiego 14a, 50–370 Wrocław, Poland
Email: marcin.zawada@pwr.wroc.pl

***Abstract*— In this paper we address the problem of modeling different temporal relationships between events in multi-event attack signatures. In many existing signature representations encoding and interpreting such temporal properties can be difficult tasks. There is a growing need for a formalism which is both expressive and manageable, to support and simplify these tasks. To this end, we propose a new formal conceptual model based on Interval Temporal Logic (ITL). We show how temporal properties of multi-event attack signatures can be conveniently expressed in our model, giving a number of illustrative examples.**

I. Introduction

The increasing number of computer attacks and intrusions in recent years, as well as their growing diversity, has demonstrated the need for intensive development of various security technologies. In the research area of the design of Intrusion Detection Systems (IDS systems) two techniques are being investigated: *misuse detection* and *anomaly detection* [5]. In this paper we address a misuse detection approach which assumes attacks to be well-known sequences of actions. In this approach attack scenarios are represented in the form of special patterns, called *attack signatures*, that are used to match suspicious events in system logs or in network traffic. Attack signatures can be either *mono-event* or *multi-event*, depending on the number of steps in the corresponding attack scenarios.

Still mostly mono-event signatures are used in common intrusion detection tools (e.g. in Snort [16]). The problem is that such tools are not sufficient for identification of multi-step scenarios, but an idea of an application of more advanced, multi-event detection tools is not easy to implement in practice.

Some limiting factors on the use of multi-event signatures are requirements that are not satisfied or not easily handled by existing attack specifications. Some of such requirements concern flexibility in defining different features of multi-event signatures, their precise interpretation and manageability. Satisfaction of them is very important, because it is a necessary condition for the development of high quality signatures, faster definition of new signatures and updating of existing ones. It also has a strong influence on the success rate of detection. However, still, defining an expressive formalism for specifying multi-event signatures, that would be both clear and easy-of-use still remains a difficult challenge.

Much research in the literature has been devoted to the development of multi-event signatures specifications and to date various attack languages (detection languages [3]) have been proposed, multi-event signature-based tools. Generally, the languages designed can be classified into low-level and high-level languages. The latter represent a relatively new trend that aims at defining attack signatures in a declarative form. This approach is motivated by the fact that low-level implementation formalisms, although they provide high expressiveness, also imply high development costs. Declarative specifications, which use data structures instead of programs for attack signature descriptions, reduce development cost as they provide rules that tend to be more concise and readable since functional details are not explicitly specified. However, in some cases using declarative structures can lead to limited expressiveness. Among existing languages we can find: rule-based Russel [12], Colored Petri Nets-based specification proposed in [5], state-transition diagram-based STATL [3], regular expression-based REE [19], regular expressions with back referencing representation applied in GASSATA [7], relation algebra-based MuSig [6], event calculus algebra-based LAMBDA [2], procedural ADeLe [9, 18], declarative transition-based Sutekh [15] and temporal languages based on different linear temporal logic formalisms [17, 14, 1].

Our work is particularly motivated by the fact that a great deal of the problems with specifying multi-event signatures concern modeling and interpreting various temporal properties i.e. time-relations between events. And though such features are of special importance in multi-event signatures, specifying and analyzing non-trivial temporal properties such as partial ordering of events and non-occurrence (i.e. an event that invalidates another earlier event) are either not supported by many existing languages or defining them leads to difficulties [15, 8].

In this paper we would like to focus on developing new possibilities for facilitating modeling of such temporal properties as well as their better understanding. The goal of this paper is to present a theoretical model for defining a selected class of multi-step attacks in which steps can be temporarily ordered or specified. Our model is based on high-level declarative

Interval Temporal Logic (ITL) [11, 12] that is an expressive well-studied formalism successfully applied to different areas of specification and verification of real-time systems. Our choice of using interval-based ITL is mostly motivated by the fact that it provides a straightforward way to define more complex temporal features and at the same time it also has an intuitive advantage over point-based linear temporal logic that has been used in attack signature specifications. We believe our formal model and some presented ideas might be useful for deeper analyzing some modeling problems in the context of multi-event signatures. To our knowledge no such a formal mathematical model has been proposed yet.

The structure of this paper is as follows. Section II presents our formal model for specifying temporal properties of multi-event signatures. Section III provides a set of basic modeling rules. Section IV presents a study of more complex modeling cases and provides illustrative examples. Section V concludes the paper.

II. Formal SigITL Model

In this section we present a formal model that we shall use later for specifying temporal properties of multi-event attack signatures. We begin with a brief introduction to general concepts of temporal logic.

Temporal logic extends propositional logic with a notion of time by introducing special temporal operators e.g. *always* ($\Box$), *sometime* ($\Diamond$), *at the next moment* ($\bigcirc$). Adding them allows us to define when statements are true. For example, if p is some statement, an intuitive interpretation of $\Box p$ is "statement p is always true".

There are many different temporal logics that may differ about the underlying nature of time which is assumed. In the context of misuse detection problem it is natural to consider time as a *discrete* and *linear* structure, because misuse detection techniques base on analyzing a set of discrete, time-sequenced events (arranged by their timestamps) from audit trails or network traffic.

For more convenient expressing different temporal properties of multi-event signatures we propose to consider the use of *Interval Temporal Logic* (ITL) [11]. ITL can be viewed as an extension of linear time point-based logic, where the notion of the satisfaction by a point is replaced by a notion of the satisfaction by an *interval*, that is a finite discrete linear structure. Interpretation over intervals greatly simplifies formulation of temporal formulas. Our model we would like to propose is a slightly modified subset of ITL. We called it *Sig*ITL (*Signature* ITL). Let P be a set of atomic propositions. Let us present formal syntax and semantics of *Sig*ITL.

Syntax. Formulas in *Sig*ITL are defined inductively as follows

$$\varphi ::= P \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \bigcirc\varphi \mid \varphi\,;\,\varphi \mid \varphi \oplus \varphi \mid \varphi^n.$$

Semantics. Let π be a finite sequence of events. By $\pi(i)$ we denote i^{th} event of a trace π. The term $\pi[i,j]$ denotes the interval of a trace π from position i to position j, where $i \leq j$. If a *Sig*ITL formula φ holds in the interval $\pi[i,j]$, we denote this by $\pi[i,j] \models \varphi$, and say that an interval $\pi[i,j]$ satisfies a formula φ. Let φ, ψ be *Sig*ITL formulas. Given a trace π, satisfaction is defined as follows

$$\begin{aligned}
\pi[i,j] &\models p \quad \text{iff} \quad \pi(i) = p,\ p \in P \\
\pi[i,j] &\models \neg\varphi \quad \text{iff} \quad \neg(\pi[i,j] \models \varphi) \\
\pi[i,j] &\models \varphi \wedge \psi \quad \text{iff} \quad \pi[i,j] \models \varphi \wedge \pi[i,j] \models \psi \\
\pi[i,j] &\models \varphi \vee \psi \quad \text{iff} \quad \pi[i,j] \models \varphi \vee \pi[i,j] \models \psi \\
\pi[i,j] &\models \Box\varphi \quad \text{iff} \quad (\forall k \in [i,j])(\pi[k,j] \models \varphi) \\
\pi[i,j] &\models \Diamond\varphi \quad \text{iff} \quad (\exists k \in [i,j])(\pi[k,j] \models \varphi) \\
\pi[i,j] &\models \bigcirc\varphi \quad \text{iff} \quad \pi[i+1,j] \models \varphi \wedge i < j \\
\pi[i,j] &\models \varphi\,;\,\psi \quad \text{iff} \quad (\exists k \in [i,j))(\pi[i,k] \models \varphi \\
&\qquad\qquad\qquad\quad \wedge \pi[k+1,j] \models \psi) \\
\pi[i,j] &\models \varphi \oplus \psi \quad \text{iff} \quad \pi[i,j] \models (\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi) \\
\pi[i,j] &\models \varphi^n \quad \text{iff} \quad \pi[i,j] \models \underbrace{\varphi\,;\,\varphi\,;\,\ldots\,;\,\varphi}_{n}
\end{aligned}$$

In the context of multi-event signature approach a trace π corresponds to a sequence of events from system logs or network traffic and temporal formulas to attack signatures.

III. Basic Modeling Rules in SigITL

In this section we formally define a set of modeling rules for specifying different relationships between events in multi-event signatures. Let $p_1, p_2, \ldots, p_n$ be atomic propositions of *Sig*ITL which are assumed to be mutually exclusive. Then interesting temporal properties are defined as follows.

Sequence. A condition that events must occur in a fixed sequential order is expressed as follows

$$\Diamond p_1\,;\,p_2\,;\,\ldots\,;\,p_n.$$

Note that the above formula is equivalent to a formula $\Diamond(p_1\,;\,p_2\,;\,\ldots\,;\,p_n)$.

Any order. If an order in which events should occur is not important is modeled in the following way

$$\Diamond p_1 \wedge \Diamond p_2 \wedge \ldots \wedge \Diamond p_n.$$

Exclusive choice. Exclusive choice between two or more alternative events is represented by the operator $\oplus$ and defined as follows

$$\Diamond(p_1 \oplus p_2 \oplus \ldots \oplus p_n).$$

Non-occurrence. The absence of an event of a special type between occurrences of two others is specified by the following formula

$$\Diamond(p_1 \wedge \Box\neg p_3)\,;\,p_2,$$

which means that an occurrence of p_1 is followed eventually by an occurrence of p_2 and p_3 never occurs between p_1 and p_2.

Repetition. Occurrence of at least n repetitions of a particular event type is expressed by the following iterative formula

$$\Diamond(p_1^n).$$

Note that we add operator $\Diamond$ in the front of each rule. This is due to the fact we are not necessarily interested in the occurrence of any event from a multi-event signature at the first time point only. The above basic operations can be combined to get more complex temporal properties, such as partial ordering of events. These properties will be studied and presented in the next section.

Now let us provide a few simple examples how the above modeling rules can be used. Let us fix a finite alphabet $\Sigma = \{A, B, ..., Z\}$ of elements that refer to possible types of events that can occur in system logs or in network traffic. Let each element from Σ correspond to some propositional property. Consider three distinct events of types: A, B and C. A sequence of them can be expressed as a formula: $\Diamond A\,;\, B\,;\, C$. If any of the possible orders of these events is acceptable, it can be defined as a formula: $\Diamond A \wedge \Diamond B \wedge \Diamond C$. In order to specify the case of an exclusive choice of them i.e. "one event, of type A or B or C, must occur" we may use a formula: $\Diamond(A \oplus B \oplus C)$. A formula $\Diamond(A \wedge \Box\neg C)\,;\, B$ represents the condition "between events of type A and B must not occur an event of type C". If we would like to specify a thrice-repeated occurrence of an event of type A we can use a formula: $\Diamond(A^3)$.

IV. Advanced Modeling Rules in SigITL with Examples

Specifying more complex temporal properties of multi-event signatures in *Sig*ITL can be done according to the rules introduced in the previous section. However, there are particular cases when specifying some of them might be a bit problematic. For this reason, in this section we study such cases, presenting and explaining them in more details. Furthermore, we introduce some of our ideas that we think can simplify these tasks. Due to the limited space in this paper we present our ideas informally; suitable formal definitions and proofs will be published at a later date. We start with providing a notation that could support precise interpretation of different continuity modes of an event sequence in description of multi-event signatures that have been distinguished in [5] and [8]. We follow partially the terminology from [5].

Continuity in the Sequence. In a proposed model we can consider at least three different semantics for an event sequence, namely: (a) *immediately follows*, (b) *follows* and (c) *follows with constraints*. To illustrate them, consider again three different events of type: A, B and C. We assume they are mutually exclusive. A mode (a) addresses cases in which we are interested in an immediate occurrence of events one after another in a totally specified order. For example, $\Diamond(A \wedge \bigcirc B \wedge \bigcirc\bigcirc C)$ means that an event of type A is immediately followed by an event of type B that is immediately followed by an event of type C. Another equivalent form of this formula is: $\Diamond(A \wedge \bigcirc(B \wedge \bigcirc C))$. In mode (b) strict order constraints of the events considered are given, but insertions of other events between them are allowed. The corresponding formula is $\Diamond A\,;\, B\,;\, C$. With the last mode (c) we can indicate that between the events considered, events of some type must not occur. For example, $\Diamond(A \wedge \Box\neg C)\,;\, (B \wedge \Box\neg A)\,;\, C$ enforces that there must be no event of type C between events of type A and B and there must be no event of type A between events of type B and C.

It is important to note that in the context of multi-event signature detection problem, the modes of interest are (b) and (c) rather than (a), because it is common that relevant events (corresponding to an attack scenario) can be interlaced in an analyzed event stream with unrelated ones [5].

Partial ordering. A rule for "partial order" of events is in fact a rule built up from basic rules: "sequence" and "any order". In order to illustrate how to model partial order in our model, let us consider an old classical attack [13, 5, 17] in which an attacker exploits an old sendmail bug in Unix. It has the advantage of being well-known and simple, but sufficiently illustrative for two different rules we would like to propose. This *mail* attack scenario consists of the following steps. An attacker copies (**cp**) some shell (**/bin/sh**) to **/usr/spool/mail/root**, then he sets its setuid bit on (**chmod 4755 /usr/spool/mail/root**), creates an empty file **x** (**touch x**) and sends the fake mail message to **root** (**mail root** $<$ **x**). In old implementations, when root tried to read his mail, the ownership of **/usr/spool/mail/root** was switched to root, giving the attacker root privilege.

A corresponding multi-event attack signature to the above partially ordered scenario needs to include the following information: cp must precede chmod and chmod must precede mail, but the only condition for touch is that it occurs before mail. Let A refers to an event type **cp**, B to **chmod**, C to **mail** and D to **touch**. Symbolic illustration of corresponding signature is given in Fig. 1.

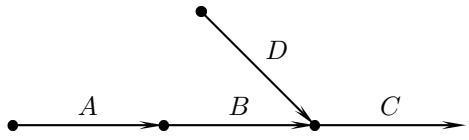

Fig. 1. A graphical representation of a partial order of events in a *mail* attack

A formula that specifies a considered signature can be as follows:

$$(\Diamond(A\,;\, B) \wedge \Diamond D)\,;\, C. \tag{1}$$

Since specifying more complex partial orderings, especially of a more numerous set of events, may become very complicated, we propose a rule to define directly suitable formulas. Informally, the rule states that a temporal formula that defines a partially-ordered set of events in a multi-event signature can be defined directly, thanks to the existence of maximal chains[1] of an underlying partially ordered set. Thanks to this

[1]In relation algebra a maximal chain of a finite partial order is defined as a chain (i.e. subset of the partial order that is totally ordered) that is not a proper set of any other chain. For more complex partial orders, a more convenient way is to read maximal chains from acyclic directed graphs (standard representation of partially ordered sets). Here, to be consistent with our further examples of non-occurrence, we have used a bit different graphical illustration.

rule, a resulting formula can be defined as a conjunction of temporal formulas that correspond to all maximal chains of this set. We have called the rule a *maximal chain rule*. To give a general view how this rule works, let us consider again a partial order in a signature of the *mail* attack. It has two maximal chains, for which corresponding formulas are $\varphi = \Diamond A\,;\, B\,;\, C$ and $\psi = \Diamond D\,;\, C$. Thereby, using our mentioned rule, we can immediately get the following formula: $\varphi \wedge \psi = (\Diamond A\,;\, B\,;\, C) \wedge (\Diamond D\,;\, C)$ that is semantically equivalent to the shorter formula (1)). The proposed rule works for any finite partially-ordered set of events. Both rules, for defining and shortening formulas that correspond to partially-ordered signatures, we shall present formally in a forthcoming report. Let us just mention here that these formal rules can support *overlap* and *nonoverlap* semantics proposed in the model for the semantics of attack signatures presented in [8].

As another example, suppose that an attacker can use a different Unix command to create an empty file **x** in a *mail* attack scenario under consideration. Therefore instead of an event of type D (**touch**) we have an alternative event of type F (e.g. **cat**). We can encode such a possible variant as follows

$$(\Diamond(A\,;\, B) \wedge \Diamond(D \oplus F))\,;\, C.$$

Single and nested non-occurrence. To illustrate the problem of non-occurrence, let us consider another possible variant of executing a *mail* attack. Suppose that the setuid bit can be cleared before a fake message is sent to **root**. In such a case an alarm must not be reported, because it would certainly be a false alarm. On the other hand, we can consider the possibility that clearing the setuid bit is only temporary - it can be a technique used to elude detection and allow an attacker to deceive the IDS system. So, in order not to omit a real attack, we need to continue monitoring to see whether the setuid bit is set again. This case was considered in [13, 17]. The problem in encoding possibilities of "setting-clearing" the setuid bit in temporal formula is that this situation can repeat many times. The idea proposed in [17] was to do tests of evenness, i.e. checking "parity conditions" (represented as a loop in Fig. 2a). As noted, this cannot be directly expressed in linear time temporal logic without adding an auxiliary variable, so using a more expressive but also more complex logic has been proposed in [17]. We would like to look at this problem in a different way. Note that in fact we are not interested in expressing whether the number of times the setuid bit has been set is even or odd; we would prefer to carry out a different test with the same effect here. Let B refer to setting the bit setuid and E to clearing it. We could consider reporting an attack only if events of type A, B and C occurs in a given order under the condition that there is no event of type E (non-occurrence) between events of type B and C. In other words, in our temporal formula we require $\neg E$ before C occurs. For a moment we do not consider an event D. We can check the necessary condition we described with the following formula

$$\Diamond A\,;\, (B \wedge \Box\neg E)\,;\, C. \qquad (2)$$

For better illustration, let us consider the following input sequence of events

$$[--\underline{A} - BE - -B - E - -\underline{B} - -\underline{C}-]$$

where by '$-$' we mark unrelated types of events. The formula (2) holds in this interval, so an attack would be successfully detected (events that refer to an attack are underlined).

So far we have dealt with encoding a *single* non-occurrence. The same idea can be also applied to the case of several *nested* non-occurrences. Let us consider a signature of a "*ftp-write*" attack described in [3]. Its graphical illustration is presented in Fig. 2b. We focus only on giving the general view of this signature (details can be found in [3]). In this signature we have two non-occurrences D and one non-occurrence E. We can see that an event of type A is followed by an event of type B, that is followed by an event of type C and D invalidates an occurrence of an earlier event A and E invalidates an occurrence of an earlier event B. Using similar analysis and ideas proposed to the single non-occurrence problem, we can easily get for this signature the following formula

$$\Diamond(A \wedge \Box\neg D)\,;\, (B \wedge \Box\neg E \wedge \Box\neg D)\,;\, C.$$

Note that in defining non-occurrence, it is much more convenient to define the formula from right to left.

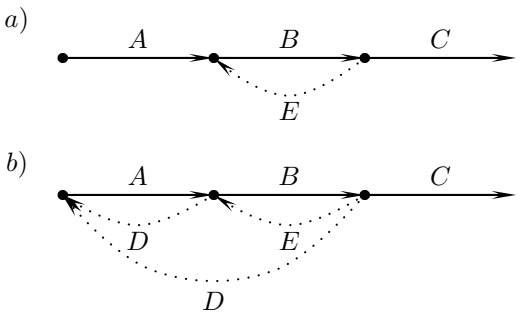

Fig. 2. A graphical illustration of multi-event signatures with non-occurrence: a) a single non-occurrence b) nested non-occurrences (non-occurences are marked by a dotted line)

Let us just to mention here that non-occurrences correspond to so-called unwinding transitions in state-transition diagrams considered in [3].

Partial ordering and non-occurrence. Let us consider again a partially ordered signature of the *mail* attack presented in Fig. 1. Assume we include in this signature a non-occurrence E (compare Fig. 2a). We can express such a form of this signature by this formula

$$((\Diamond A\,;\, (B \wedge \Box\neg E)) \wedge \Diamond D)\,;\, C.$$

V. Conclusions

In this paper we have proposed a formal model intended to support a task of modeling time relations between events in multi-event attack signatures. Modeling such properties is an important part of attack signature specification.

The model proposed is based on an interval temporal logic formalism which has already proved to be a convenient and

expressive modeling tool in areas of real-time systems where different temporal properties need to be specified. Our model proposed here is a simplified version adapted to modeling of time-related features in multi-event signature representation.

In particular, we have presented rules and our modeling ideas to facilitate modeling some problematic cases such as single and nested non-occurrences, and complex partial orders with or without associated non-occurrences. The model is flexible enough for these tasks, and in addition it clarifies the semantics for interpreting temporal properties of multi-event signatures thanks to rigorous mathematical notations.

REFERENCES

[1] M. Couture, B. Ktari, M. Mejri, F. Massicotte, *A Declarative Approach to Stateful Intrusion Detection and Network Monitoring*, The Second Annual Conference on Privacy, Security and Trust (PST 2004), October 2004.

[2] F. Cuppens, R. Ortalo, *LAMBDA: A Language to Model a Database for Detection of Attacks*, In Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID 2000), October 2000.

[3] S. T. Eckmann, G. Vigna, R. A. Kemmerer, *STATL: an Attack Language for State-based Intrusion Detection*, In Proceedings of the ACM Workshop on Intrusion Detection, Athens, Greece, November 2000.

[4] K. Ilgun, R. A. Kemmerer, P. A. Porras, *State Transition Analysis: Rule-Based Intrusion Detection Approach*, IEEE Transactions on Software Engineering, Vol. 21, No. 3, pp. 181-199, March 1995.

[5] S. Kumar, *Classification and Detection of Computer Intrusions*, PhD Thesis, Purdue University, 1995.

[6] J. L. Lin, X. S. Wang, S. Jajodia, *Abstraction-Based Misuse Detection: High-Level Specifications and Adaptable Strategies*, In Proceedings of the 11th Computer Security Foundation Workshop, IEEE Computer Society Press 1998.

[7] L. Me, *A Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis*, In Proceedings of the 1st Iinternational workshop on the Recent Advances in Intrusion Detection (RAID98) 1998

[8] M. Meier, *A Model for the Semantics of Attack Signatures in Misuse Detection Systems*, In Proceedings of the 7th International Information Security Conference (ISC 2004), pp. 158- 169, USA, September 2004.

[9] C. Michel, L. Me, *ADeLe: an Attack Description Language for Knowledge-based Intrusion Detection*, In Proceedings of the 16th International Conference on Information Security, Kluwer, June 2001.

[10] B. Morin, H. Debar, *Correlation of Intrusion Symptoms: an Application of Chronicles*, In Proceedings of the 6th symposium on Recent Advances in Intrusion Detection (RAID 2003), September 2003.

[11] B. Moszkowski, *Executing Temporal Logic Programs*, Cambridge University Press, Cambridge, England, 1986.

[12] B. Moszkowski, *A Hierarchical Completeness Proof for Propositional Interval Temporal Logic with Finite Time*, In Special Issue of a Journal of Applied Non-Classical Logics on Interval Temporal Logics and Duration Calculi, Vol. 14, No. 1-2, pp. 55-104, 2004.

[13] A. Mounji, *Languages and Tools for Rule-Based Distributed Intrusion Detection*, PhD thesis, Computer Security Institute, University of Namur, Belgium, September 1997.

[14] P. Naldurg, K. Sen, P. Thati, *A Temporal Logic Based Framework for Intrusion Detection*, In Proceedings of the 24th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2004), August 2004.

[15] J. P. Pouzol, M. Ducasse, *From Declarative Signatures to Misuse IDS*, In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001), 2001.

[16] M. Roesch, `http://www.snort.org`

[17] M. Roger, J. Goubault-Larrecq, *Log auditing through model checking*, In Procedings of the 14th IEEE Computer Security Foundations Workshop, June 2001.

[18] E. Totel, B. Vivinis, L. Me, *A Language Driven Intrusion Detection System for Events and Alerts Correlation*, In Proceedings of the 19th IFIP International Information Security Conference, Kluwer Academic, August 2004.

[19] P. Uppuluri, R. Sekar, *Experiences with Specification Based Intrusion Detection System*, In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection Recent Advances in Intrusion Detection (RAID 2001), October 2001