Inference of Security Hazards from Event Composition Based on Incomplete or Uncertain Information

Segev Wasserkrug, Avigdor Gal, and Opher Etzion

Abstract— **A quick recognition of security hazards is challenging. Information sources are often insufficient to infer the occurrence of hazards with certainty, requiring recognition to be based on patterns of occurrences distributed over space and time. We introduce a generic framework that supports a quick response to changes in patterns of occurrences, using multiple inferencing techniques. We demonstrate, with a case study of detecting DoS attacks, that our approach is more flexible and accessible than custom coded solutions, supporting multiple statistical inferencing techniques when such results are available.**

I. Introduction

Security applications, such as those responding to DoS attacks, face a trade-off between identifying security hazards with certainty, and the need to provide a quick response to threats. Waiting too long to respond may mean that mitigating actions have little effect once they are applied, while wrongly determining that a DoS attack has occurred may result in unjustifiable denial of service.

Event Composition Systems (*e.g.*, [1], [2], [3]) have been proposed in the literature for analyzing data and detecting situations of interest. These general purpose systems infer, in real time, the occurrence of *events*, based on the occurrence of other events using rules. The declarative nature of rules, combined with an optimized inference mechanism, allows quick response to new and evolving situations by simply changing a set of rules rather than by making changes in code. However, existing event composition systems are by-and-large unable to handle incomplete or uncertain information.

In this work we describe an uncertainty management extension of event composition systems, using a quantitative approach based on probability theory that associates a certainty measure with inferred events. Quantitative decision-making enables assessing the certainty of threats and deciding on mitigating activities, even in the absence of complete information. To the best of our knowledge, our proposed solution is the most comprehensive for handling inference with uncertain and incomplete information for event composition systems. This makes it an ideal candidate for security informatics, as illustrated in the case of DoS attack detection described in Section IV.

IBM Haifa Research Labs; Techion - Israel Institute of Technology.
Technion - Israel Institute of Technology.
IBM Haifa Research Lab.

II. Related Work

Network attack detection, the focus of the case study in this paper, is analyzed in [4], [5] using BGP routing instabilities and describing how distributed events may be correlated to detect attack occurrences (using a Bayesian network in [4]). The need for inference mechanisms to handle incomplete information and uncertainty is further motivated in [6], [7]. These works tailor probabilistic or statistical models to a specific security application, while our work proposes a generic framework for the inference of security events. We also overcome the inability of application-specific tailored models to combine the results of different models.

Contemporary event composition systems (*e.g.*, [2] and [3]) reason only about deterministic knowledge regarding the relationships between events. The language and inference algorithm of [8] manage uncertainty, yet are too limited to enable the definition of the types of rules required for the applications discussed in this work. For example, the predicates of that language are limited, and the notion of a lifespan, which is used extensively in the case study, does not appear in [8].

III. Uncertain Inference Framework

In this section we provide a generic formal framework for uncertain event inferencing that can be efficiently and effectively utilized for security applications, using probability theory as an underlying mechanism to represent uncertainty. The framework has three main components, namely *Event Model*, *Inference Language*, and *Inference Algorithms*.

The event model is used to represent an event, as well as the uncertainty regarding an event occurrence (*i.e.*, whether an event occurred) and event attributes, *e.g.*, when it is clear that an event has occurred, but uncertainty exists regarding its exact occurrence time. The proposed framework includes a formal probabilistic definition of the semantics of the relevant uncertainties (see Section III-A).

The inference language enables *inferring* events under uncertainty using *rules*. Rules use formal probabilistic semantics, and can involve complex temporal predicates.

The third main component involves a set of inference algorithms to calculate the probabilities of inferred events, given events and rules. The algorithms ensure that a valid probability space is constructed, based on the semantics of the event model and the language.

A. Event Model

In the context of active systems, an *event* is defined as an occurrence that is significant (falls within a domain of interest to the system), instantaneous (takes place at a specific point in time), and atomic (either occurs or not). Examples of events include BGP advertisement notifications and ICMP unreachable messages in the network domain, which, as discussed in the sequel, are also important for the detection of DoS attacks.

A variety of data can be associated with the occurrence of an event, *e.g.*, the IP address of a router generating BGP advertisement messages. Two events that share the same data types are of the same *event type*. For example, consider events that provide the hourly number of advertisements of a BGP protocol in a network. With each instance of this type of event, the relevant information consists of all the details that such an advertisement must include according to the BGP protocol. An example of such information is the exact hour for which this event was generated. All such events can be said to belong to the event type $BGPAds$.

In the context of event composition systems, a distinction is made between *explicit events* and *inferred events*. The former type are events for which an event notification is signaled by an **external** event source. The latter type of events are inferred from other events using a set of rules.

Event Instance Data (*EID*) represents the information a composite event system holds about event instances. *EID* incorporates all relevant data about an event, including its type, time of occurrence, *etc.* In event composition systems with no uncertainty, each event can be represented by a single tuple of values $Val = \langle val_1, \ldots, val_n \rangle$ (one value for each type of data associated with the event). To capture the uncertainty associated with an event instance, the *EID* of each event instance is a Random Variable (*RV*). The possible values of *EID* are taken from the domain $V = \{NO\} \cup V'$, where NO stands for the non-occurrence of an event and V' is a set of tuples of the form $\langle val_1, \ldots, val_n \rangle$.

The semantics of a value of E (encoded as an *EID*), representing the information the system has about event e, are as follows: the probability that the value of E belongs to a subset $S \subseteq V \setminus \{NO\}$ is the probability that event e has occurred, and that the value of its attributes is some tuple of the form $\langle val_1, \ldots, val_n \rangle$, where $\{\langle val_1, \ldots, val_n \rangle\} \in S$. Similarly, the probability associated with the value $\{NO\}$ is the probability that the event did not occur. For example, assume that the system considers the following alternatives regarding a $BGPAds$ event: the event was not generated at all; the event was generated, stating that the number of BGP advertisements that occurred at 10:00AM is 20; or the event was generated, stating that 30 advertisements occurred at 10:00AM. In addition, assume that the system considers the probabilities of these possibilities to be 0.3, 0.3, and 0.4, respectively. Then, the event can be represented by an RV E whose possible values are $\{NO\}, \{10{:}00\text{AM}, 20\}$, and $\{10{:}00\text{AM}, 30\}$. Also, $\Pr(E = \{NO\}) = \Pr\{E = \{10:00AM, 20\}\} = 0.3$ and $\Pr\{E \in \{\{10{:}00\text{AM}, 20\}, \{10{:}00\text{AM}, 30\}\} = 0.7$.

Finally, we introduce an *Event History* $EH_{t_1}^{t_2}$ to be the set of all events (of interest to the system), as well as their associated data, whose occurrence time falls between t_1 and t_2.

B. Inference Language and Algorithms

A rule language defines a set of rules, such that these rules, together with the information regarding explicit events, define the probability space of interest at each point in time t. At any time point t, the probability that an event e with specific data occurred at some time $t' \leq t$ should be computed, taking into account only the evidence that is known to the system at time t. Therefore, a (possibly different) probability space is defined for each t. An intuitively appealing way to define this probability space involves possible world semantics [9], in which possible event histories correspond to possible worlds. Using such a definition, the *possible world representation* is a probability space $P_t = (W_t, F_t, \mu_t)$ at time t such that W_t is a set of possible worlds, with each possible world corresponding to a specific event history at time t, $F_t \subseteq 2^{|W_t|}$ is a σ-algebra over W_t, and $\mu_t : F_t \rightarrow [0, 1]$ is a probability measure over F_t.

A rule r has three main functions. It defines a *temporal context* (*e.g.*, an interval) during which inference is relevant. For example, some rules may be relevant only during regular working hours. It also defines events that are relevant to the inference of other events. Finally, it defines conditions, attribute values, and probabilities that support the occurrence of an event. A formal definition of the semantics of rules appears in the online supplement.

A rule r is of the form $\langle LS, \{RE\}, ET \rangle$, where LS defines the temporal context. $\{RE\}$ is a set of rule expressions, each defining a condition under which an event of class ET may be inferred. A restriction that we impose ensures that a single rule is defined for each event class.

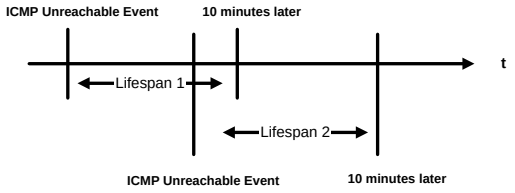

Fig. 1. Sliding Time Window Defined by Lifespan

The temporal context LS is defined by a start point and an end point, such that both may be either the occurrence time of an event, or an explicit time point. In addition, the end point may be determined as the length of time measured from the starting point. For example, consider a lifespan that defines an interval whose beginning is the occurrence time of an ICMP unreachable event, and the end point is 10 minutes afterwards. Such a lifespan defines a sliding time window as shown in Figure 1.

RE, which defines a condition under which an event of class ET may be inferred, is a tuple of the form $\langle Sel,\ Pat,\ Map,\ \Pr \rangle$. Sel is a function, that given an event history, selects a subset of that history on which a pattern Pat, a (possibly temporal) predicate, is tested. When Pat evaluates to $true$, an event of type ET may be inferred. Map is a set of functions that maps the attribute values of the events that triggered this rule to the attribute values of the inferred event. It may be any deterministic function from the attributes of the event classes of the selected events. Finally, $\Pr \in [0, 1]$ is the probability of inferring the event by the rule (for formal details, see the online supplement).

Security applications require that the inference process be done in real time. Inference algorithms should therefore be efficient, despite the non-monotonicity of the process (*e.g.*, new information may cause previously inferred events to no longer be considered possible). Our framework offers two inference algorithms. The first constructs a Bayesian network based on rules, ensuring that calculations are carried out on a valid probability space. Bayesian network construction is exponential in the state space of the events and we therefore offer an algorithm, based on Monte Carlo sampling, that can obtain approximated probabilities in polynomial time. This algorithm exhibits very good scaling characteristics, as described in Section IV-B.

IV. Case Study and Performance Results

A. Case Study

The occurrence of a potential security threat could be inferred from seemingly unrelated isolated events. Examples of such cases are:

Bio Hazard: Detecting the occurrence of a disease outbreak due to a terrorist attack could be based on recognition of the disease at several different hospitals in different geographical locations, but in very close temporal proximity. In addition, such outbreaks could be correlated with the presence of suspected terrorists at nearby airports. Uncertain rules calculate the probability of such a terrorist attack based on these basic events.

Computer Security: Login attempts to the same account with different passwords or to different accounts with different passwords in close temporal proximity. Rules may be defined to calculate the probabilities of an attempted break-in.
Recognition of DoS attacks: DoS could be inferred based on events indicating BGP routing instabilities.

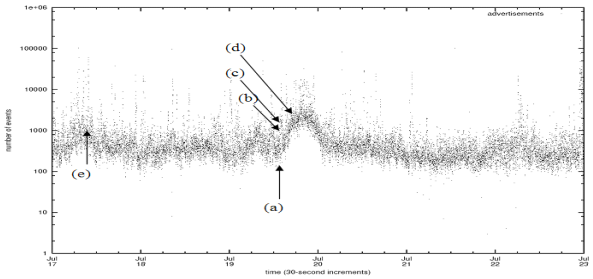

Fig. 2. July 19 Trend

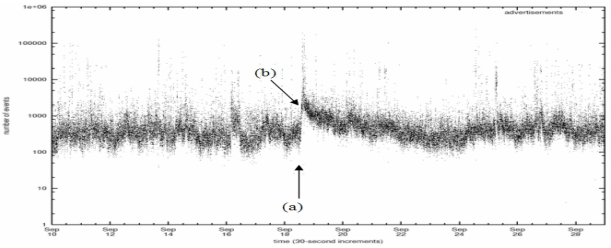

Fig. 3. September 18 Spike

We now demonstrate, based on Internet traffic monitoring events, how the language defined in Section III-B can be used for detecting DoS attacks caused by the propagation of worms. This is carried out based on the work in [5], in which it was shown that BGP routing instabilities can serve as an indicator of the spread of Internet worms. Specifically, in [5], the occurrence of two worms, the first on July 19, 2001, and the second on September 18, 2001, correspond to the patterns appearing in Figures 2 and 3, respectively. One pattern involves a gradual, yet constant and significant increase of BGP advertisements, corresponding to the occurrence of the Code Red II worm on July 19, 2001 (see Figure 2). The other involves a sudden large spike in the number of BGP routing advertisements. This can be seen in Figure 3(b), which shows a sudden spike on September 18, 2001. This spike corresponds to the propagation phase of the NIMDA worm.

The proposed framework defines worm detecting rules based on such patterns. Assume that the monitoring system generates events that provide the number of BGP advertisements in an hour. Such events are called $BGPAds$ events, and an attribute associated with each such event is the actual number of hourly BGP advertisements. These events can alternatively be inferred using additional rules based on individual BGP advertisement events.

To infer the occurrence of a worm based on the occurrence of $BGPAds$ events, two rules are defined. Rule r_1 is intended to detect a gradual and constant increase in the number of $BGPAds$ events. r_1 is defined with a sliding time window lifespan, similar to the one shown in Figure 1. This is a sliding window initiated by a $BGPAds$ event, for which the number of BGP advertisements is at least 1000. The lifespan closes when there is a decrease in the number of BGP advertisements. In addition, r_1 has the following two rule expressions:
RE_1: Exactly **three** consecutive $BGPAds$ events in with an increase of at least 400 advertisements between events. In such a case, the probability that a worm occurs is 0.8. Such probabilities can be inferred based on machine learning techniques, for example.
RE_2: At least **five** $BGPAds$ events that occur in consecutive hours with an increase of at least 400 advertisements between these events. Here, the probability that a worm occurs increases to 0.9.

In a similar manner, a rule r_2 can be defined, to recognize a sudden spike in advertisement numbers.

We next demonstrate how r_1 is used to infer the probability associated with a worm occurrence based on the trace in Figure 2:
On July 19 at midday, the number of BGP advertisements begins to rise. This is indicated by the arrow (a) in Figure 2. Note that this number rises in a linear fashion from the middle of the day until peaking 22 hours after the beginning of the day. Moreover, notice that the peak is around 5000, and at midday, the number of advertisements is 500. Therefore, in the space of 10 hours, the number of BGP events rises linearly by 4500, which means that the number rises by 450 per hour. As described above, this trend corresponds to the propagation phase of the Code Red II worm.
At around 15:00, there are 1000 advertisements ((b) in Figure 2) initiating the lifespan of rule r_1.
In the next three hours three additional events occur, with an increase of 450. Therefore, at 18:00 ((c) in Figure 2), it can be inferred from RE_1 that a worm

occurs with probability 0.8. This is six hours before 22:00, the peak of the BGP instability.
After two additional hours, the pattern defined by RE_2 evaluates to $true$. Therefore, according to RE_2, the probability of an attack increases to 0.9 by 20:00, two hours before the peak of the attack. This is denoted by arrow (d) in Figure 2.

In the above case, there are $BGPAds$ events that initiate the lifespan of r_2, yet r_2 is not evaluated to $true$ for the data shown in Figure 2. Therefore, inference is not carried out according to this rule. For the second trace (Figure 3), rule r_2 leads to a similar inference.

It is worth noting that for both rules, nowhere else in the traces does either of the two rules cause a new event to be considered possible. For example, even though at the point in time denoted by arrow (e) in Figure 2 there is a slight increasing trend in advertisements, rule r_1 is not triggered, as most of the time the number of advertisements is less than 1000, and even when the number of advertisements rises higher than 1000, there are no consecutive events in which there is an increase of at least 400 advertisements between events.

Our approach has significant benefits for security applications. First, uncertainty is explicitly taken into account, which means that the system becomes aware of a possible DoS attack much sooner than with a deterministic approach. Also, the quantification of the uncertainty enables decision-making based on the weight of this uncertainty. An additional advantage is that the rules enable expressing in an explicit and intuitive manner the relationship between the explicit events and the inferred events. Such explicit definition enables the system to explain the reason for the event inference. Such explanations increase user confidence and facilitate the acceptance of the system. Another benefit of the proposed framework is that rules can be dynamically added to enable the inference of the occurrence of worms based on other indications. Consider, for example, an increase in ICMP unreachable events, which may also serve as an indication for the occurrence of a worm (see [4]). A rule r_3 can be defined to infer the occurrence of a worm based on an increase in such messages. Moreover, an additional rule, r_4, can infer the possibility of a worm when it is considered possible either due to ICMP unreachable events, or to $BGPAds$ events. Now, if at some point in time, the system considers a worm possible due to $BGPAds$ events with probability 0.9 and, independently, considers a worm possible due to an increase in ICMP unreachable with probability 0.9, then the effect of rule r_4 will be to increase the overall probability of the occurrence of a worm to 0.99. Finally, other types of models for the inference of a DoS attack (*e.g.*, models based on statistical regression) can be incorporated into our framework. This can be done by ensuring that the results of such models are captured by explicit events, allowing the definition of rules which build on such events, and integrating these results with other information sources available in the system.

B. Empirical analysis

To test the scalability of the proposed solution framework we based the experiments on a rule set consisting of two types of rules: Level 1 rules, which materialize events based only on explicit events, and Level 2 rules, which materialize events based on events that were materialized by Level 1 rules. In all experiments, as each new explicit event appears, samples for the materialized events were generated until the probability of each event was approximated within the desired precision. The simulation environment and the inference algorithm were implemented in Java version 1.6.03, using JDK version 5.0, on the Windows XP operating system, and were carried out on a single computer with 1GB of memory and an AMD Athlon 64 3200 CPU.

We experimented with a variety of control parameters to measure performance. One such parameter is the *number of relevant events*. Active systems are expected to process a large number of explicit events. However, not all of these events may be relevant to the materialization of new events. Therefore, in the first set of experiments we tested how the performance of our system changes with the proportion of relevant explicit events. These experiments were modeled after the performance experiments carried out in [3], in which a similar test was carried out with regard to the performance of a deterministic event materialization system.

We have also quantified the *increase in the number of samples* required to obtain a good approximation, which is expected to increase with the number of possible event histories. Finally, the *precision of approximation* can be calculated using a statistical confidence interval, and is therefore measured with

two values: α, which is the confidence of the approximation interval, and β, which is the size of the interval. It is to be expected that obtaining higher precision requires more samples. The aim of this experiment was, again, to identify the manner in which the required number of samples increased as a result of the increase in required precision.

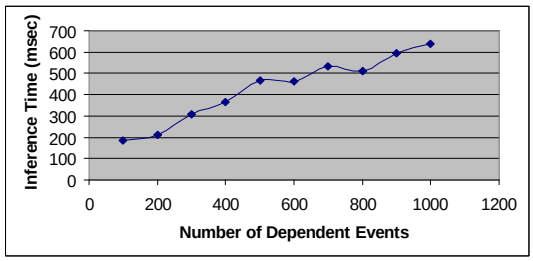

Fig. 4. Inference time with increasing number of dependent events

Due to lack of space, we refrain from presenting a detailed performance analysis. Instead, we show one example in which we measured the impact of the number of relevant events on inference performance. We define a single rule inferring event whenever a number of explicit events occur. This is similar to RE_1 of r_1 in Section IV, in which an inferred event occurs if three events of type $BGPAds$ occur within some time interval. The purpose of this experiment is to test the scalability of a single rule processing with increasing number of dependent events. The results are given in Figure 4. We observe that although the number of dependant events changes significantly, the inference time moderately increases, at times not even changing (*e.g.*, from 500 to 600 dependant events). In addition, in the tested range, while the overall number of events increases by a factor of 10, the computational effort increases only by a factor of 3.5 (from 183.25 msec to 636.75 msec).

Overall, our experiments show that the algorithm displays nice scalability characteristics as the number of possible event histories increases. The number of samples required for all tested precision grows sub-linearly in the number of possible worlds. In addition, our performance results are of the same order of magnitude as the deterministic event system analyzed in [3]. This result is encouraging because in our setting, repeated deterministic inference is carried out to enable the sampling. Finally, we observed that the percentage of relevant events, as well as the desired precision, are significant factors in the performance. We conclude that the initial performance experiments demonstrate good performance and scalability.

V. Discussion and Future Work

In this work, we demonstrated how a composite event system, augmented with uncertainty handling mechanisms, can discover events of interest in security related applications. While previous event-based applications in this area exist, our approach offers several major improvements, such as better adaptability, carrying out automatic actions using mechanisms such as utility theory, and the combination of existing analytical models.

Significant challenges still remain unresolved. One such challenge is how to obtain the information required to articulate the rules. In many cases, the type of input events based on which an inference rule should be defined can be obtained using expert knowledge, and machine learning can then be applied to exactly define the pattern and probabilities.

We shall also experiment more with the inference and sampling algorithms and seek ways to further optimize the sampling code for better results.

References

[1] S. Chakravarthy and D. Mishra, "Snoop: An expressive event specification language for active databases," *Data & Knowledge Engineering*, vol. 14, no. 1, pp. 1–26, 1994.

[2] N. H. Gehani, H. V. Jagadish, and O. Shmueli, "Composite event specification in active databases: Model and implementation," in *Proceedings of 18th International Conference on Very Large Data Bases*. Morgan Kaufman, 1992, pp. 23–27.

[3] A. Adi and O. Etzion, "AMIT - the situation manager," *VLDB Journal*, vol. 13, no. 2, pp. 177–203, 2004.

[4] G. Jiang and G. Cybenko, "Temporal and spatial distributed event correlation for network security," in *Proceedings of the 2004 American Control Conference (ACC)*, vol. 2. IEEE, June 2004, pp. 996–1001.

[5] J. Cowie, A. T. Ogielski, B. Premore, and Y. Yuanb, "Internet worms and global routing instabilities," in *SPIE*, vol. 4868, July/August 2002.

[6] C.-S. Li, C. Aggarwal, M. Campbell, Y.-C. Chang, G. Glass, V. Iyengar, M. Joshi, C.-Y. Lin, M. Naphade, and J. R. Smith, "Epi-spire: A system for environmental and public health activity monitoring," in *IEEE International Conference on Multimedia and Expo*, July 2003.

[7] M. Campbell, C.-S. Li, C. Aggarwal, M. Naphade, K.-L. Wu, and T. Zhang, "An evaluation of over-the-counter medication sales for syndromic surveillance," in *IEEE International Conference on Data Mining - Life Sciences Data Mining Workshop*, 2004.

[8] S. Wasserkrug, A. Gal, and O. Etzion, "A model for reasoning with uncertain rules in event composition systems," in *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*. AUAI Press, 2005, pp. 599–606.

[9] J. Y. Halpern, "An analysis of first-order logics of probability," *Artificial Intelligence*, vol. 46, no. 3, pp. 311–350, 1990.