# Improvements in Security Alert Analysis with a Truth Maintenance System

Albert Tang and Pradeep Ray
University of New South Wales
Sydney NSW 2052 Australia

Lundy Lewis
Southern New Hampshire University
Manchester, NH 03106 USA

## Abstract

*A high percentage of false positives remains a problem in current network security detection systems. With the growing reliance of industry on computer networks, and the growing variety of attacks that can be directed towards a computer network, it is clear that detection systems must be improved in order to tackle this growing problem. To help minimise the problem of false positives, this paper describes a method and apparatus for security alert analysis that is based on two technologies: (i) event correlation and (ii) a truth maintenance system. This work was undertaken in the context of practical network security management in a large outsourced management service provider in the Asia-Pacific region.*

## 1. Introduction

In recent years there have been a number of approaches to network security management. Software and hardware firewalls are used to block traffic and log the packets, web servers store logs of activity on their servers, network intrusion detection systems are implemented to log events when packet information indicates the possibility of an intrusion or a compromise to the system. In general, security analysts have a multitude of information to search through. Although there exist a variety of event correlation algorithms to assist the analyst in determining whether an attack is underway, there remains a need for improvement to find faster ways of determining whether an event or a datum represents a real threat or is a false positive.

Many security analysts deploy basic security management systems to consolidate security events and data from multiple sources. Commercial examples include products from NetForensics in the USA [1] and Intellitactics in Canada [2]. Unfortunately they are difficult to scale and are labor intensive since most of the intelligence for security analysis resides within the security staff's minds and less of it is within the system. Further, even if a network isn't expanding, the number and types of attacks are increasing rapidly, and thus the set of correlation rules must be updated accordingly [3]. It doesn't appear to be sufficient to keep a constant number of security staff unless the intelligence of the security system improves to match the growing variety of methods and techniques that attackers deploy. Clearly, the more security engineers are employed, or the more hours each one spends doing security analyses, the greater the cost of keeping a network secure.

The main motivations behind the development of better event correlation algorithms, better intrusion detection systems, and better methods of performing alert analyses are to decrease the number of false positives and to save time. The longer it takes to detect a successful intrusion or security breach, the greater the financial cost to respond to it and fix it.

This paper proposes a system for alert analysis with two primary components: (a) a prerequisite-based logic system that allows multi-step intrusions to be represented as a collection of required events with the capability of imposing ordering restrictions and (b) a truth-maintenance system (TMS) that allows administrators to manage/maintain alerts resulting from the multi-step intrusion specifications. In brief, the theory of TMSs and reasoning in AI provides a formal method by which to infer alerts based on given evidence, to retract prior inferences of alerts as the evidence changes or as the body of evidential knowledge is perturbed, and to maintain the evolution of knowledge of alerts over time.

The notion of multi-step intrusion specifications with ordering constraints has been discussed in prior work on Colored Petri Nets [4] and the State Transition Analysis Technique [5]. However, the idea of using a TMS to manage alerts is novel, and may be viewed primarily as a method and apparatus for administrators in maintaining the quality and consistency of alerts during alert analysis procedures.

The contribution of this paper is to describe an operational approach to the examination and management of security alerts, and thus may be considered as a back-end module attached to existing security management systems. In Section 2 we discuss the background for the work and related research. Section 3 provides a brief review of event correlations methods. Section 4 comprises the bulk of the paper, describing the basics of TMSs and their application to security alert management. Section 5 describes the design, development, and operation of a TMS-based alert management system. Section 6 provides a discussion of the dimensions in

1530-1605/08 $25.00 © 2008 IEEE

evaluating the system, and Section 7 follows with the conclusion and a discussion of areas for further research and experimentation.

## 2. Background and Related Work

### 2.1. Background

This work reported in this paper was undertaken in the context of practical network security analysis performed by large outsourced management service provider in the Asia-Pacific region (which must remain unnamed). The company is a provider of network security, data and voice services to its clients. The first author of the paper was fortunate to have access to the security analysts in the company before and during the work. The discussion below reflects the sentiments of the company's security analysts during initial interviews.

The most time-consuming and stressful part of the security analyst's job is to detect a real threat amidst a high rate of false positives. The higher the number of alerts to sift through, the longer it takes for a security engineer to detect a real threat. Not only does it affect the time to detect a real threat, it also lowers the attention spent on each alert due to the need to spend time on other alerts, which in turn calls for an increase in the number of security staff required to maintain the same amount of attention to an evolving alert list.

In sum, the increasing number of alerts logged and the high number of false positives from IDSs and firewalls result in an increased need for more security engineers and therefore significantly raises the cost of monitoring a client's network security. What is needed are better ways to perform alert analyses and to manage the overall alert list.

### 2.2. Related Work

Much work has been done since the original Anderson Report [6] in 1980 to advance the effectiveness of intrusion detection systems (IDSs). Lists of numerous intrusion detection systems, both commercial and research-oriented, are available [7]. IDSs are commonly implemented with packet sniffers such as Snort [8] which store a database of signatures. If a packet matches a signature, it will cause the production of an alert for a security engineer to investigate. This step of the procedure is invaluable because it creates large logs of potentially culprit alerts and it also stores a significant amount of information determining exactly what each matching packet contains.

A complementary approach to IDS effectiveness is to provide an event correlation mechanism that infers an alert or potential security threat, based on the occurrence of some collection of raw events. There are at least three approaches to event correlation. The first approach is based on simple thresholding in inference rules. However, thresholding is somewhat a double-edged sword: the fixed nature of a threshold means that at certain times in the day an alarm may not be triggered despite the increased security threat because it doesn't breach the threshold, a.k.a. a false negative. On the other hand, a fixed threshold means that without increasing the threshold, the increasing number of network security events would cause even more false positives to be added to the list of alerts for analysis.

A second approach, called the weighted sequence threshold-based approach, considers sequences of events. By assuming that attacks are based on a sequence of events, and that the later events are of greater significance than the earlier events, then a summary weighting is calculated. When a sequence of events' weighted sum crosses a certain threshold, an alert is raised. The goal of this approach is to raise alerts more appropriately based on the stage of the attack sequence and the weight of each event [9].

A third approach is to use policies to automatically acknowledge/group alerts and thereby reduce the number of security engineers needed to manage them. By implementing common alert characteristics into a policy for a customer, an alert that matches those characteristics would then be automatically acknowledged and grouped with its sibling alerts into one over-arching alert [10 - 13]. All of these approaches represent innovative ways to reduce the workload and the stress of security analysts, including methods to manage, cluster, merge, and correlate alerts.

The method described in this paper should be considered as complementary to these approaches. We assume that some correlation method exists whereby inferences are made from raw events or alerts into higher-order alerts, following which there is an alert analysis phase. Figure 1 shows our area of concentration as the alert analysis phase.

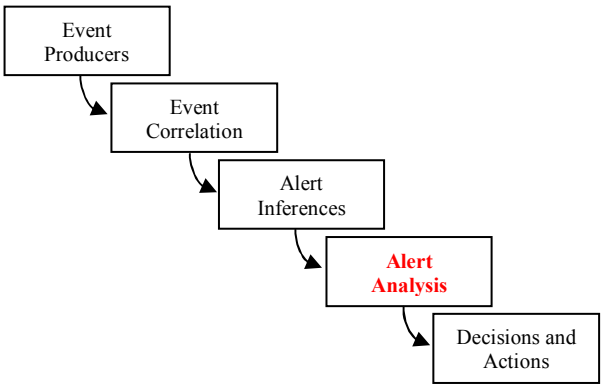


**Figure 1. A focus on alert analysis**

The novelty of our approach is to base the alert analysis phase on a TMS as a back-end to an event correlation system. The approach recognizes that as security analysts investigate alerts, it is also happening in the background that new events and alerts are being retracted, asserted, and sometimes re-retracted and re-asserted over time. The popping in and popping out of events should likewise cause similar popping in and out of alerts. In current systems, however, this doesn't happen. See the sample Snort alerts in Figure 2. It is the analyst's job to investigate each alert and determine whether it indicates a real threat, and delete or escalate it accordingly. Given the current method in the field, it is therefore difficult for analysts to keep track of the current state of things, and this contributes to their stress.

TMSs were designed for these sorts of situations, in particular for knowledge-base maintenance in large expert systems. They have been around in the artificial intelligence community for some twenty-five years (e.g. see the seminal reference [14]), but to the authors' knowledge they have not been applied to the problem of security alert analysis. In the literature TMSs have also been referred to as dependency-network maintenance systems and belief revision systems.

## 3. Rule-Based and Prerequisite-Based Event Correlation

A security threat is the result of a logical set of steps and strategies taken by the attacker. Attacks normally require certain conditions to be true and a particular strategy to exploit the conditions to be successful. Therefore, it would be logical to base an IDS on this model by creating prerequisites which an attacker would have to meet for an attack to be successful. Determining an attack to be genuine by prerequisites instead of threshold-based rules is a good way to model an attack for several reasons.

Prerequisites require less updating than thresholds, exact signatures, and rules. Thresholds, exact signatures, and rules typically require routine modification due to their lack of abstractness and they are more susceptible to environmental changes such as improving computer technology. For example, increased network capacity means that there will be more events, and if the event correlation algorithm were based purely on the number of like events for an alert to be raised, then thresholds based on a number of like events would need to be increased proportionally as network traffic increases.

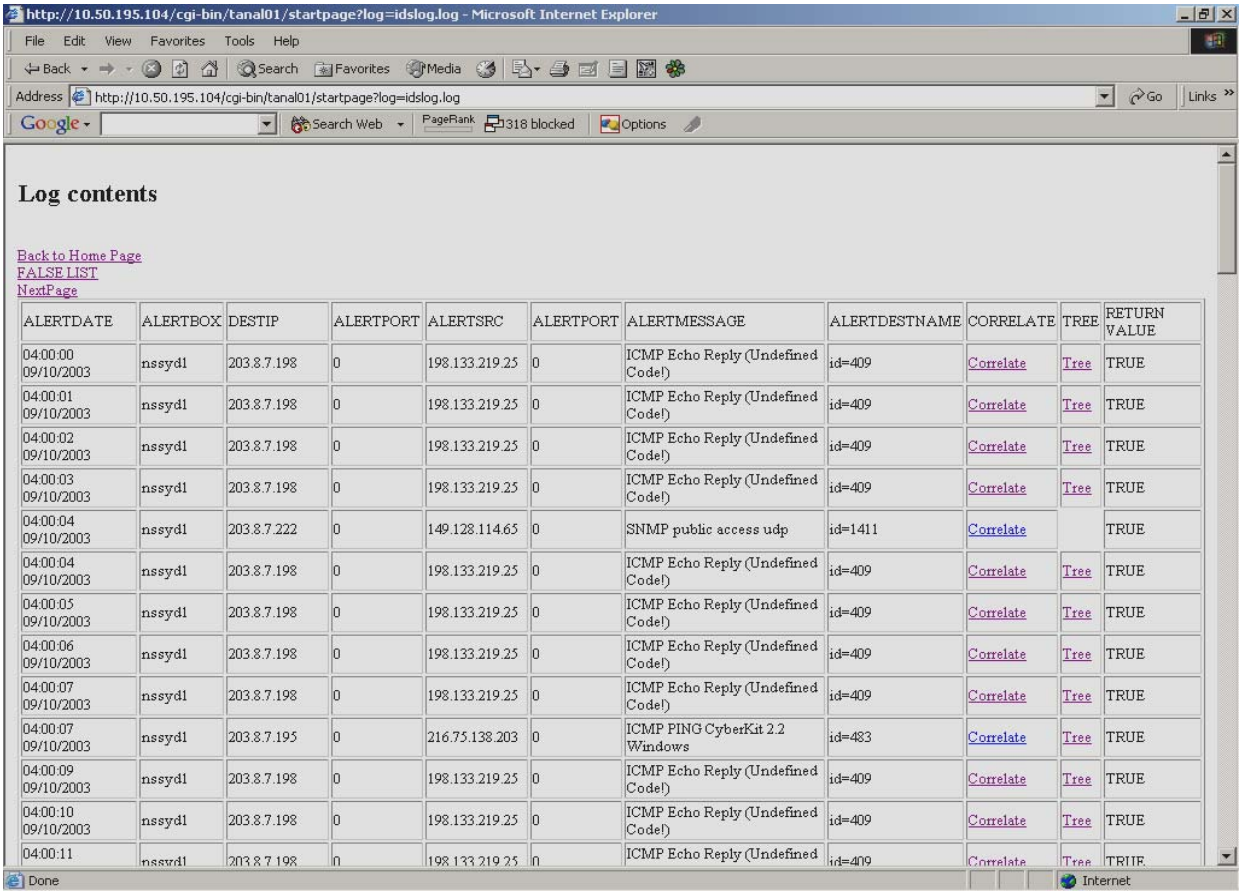


**Figure 2. A list of alerts obtained with Snort**

With the prerequisite-based approach, an attack is investigated if the required prerequisites in an attack have already been carried out, or at least partially carried out to indicate the attack [13].

## 4. Alert Analysis with a Truth Maintenance System

Consider the list of alerts as shown in Figure 2, obtained with Snort. Given such a list, it is the job of the security analyst to explore the grounds of each alert and to determine whether an alert is reason for taking action, declaring it a false positive, or setting it aside until new evidence presents itself. The purpose of a TMS is to assist the analyst in reasoning about the alert list.

### 4.1. The Problem of IDS Monotonicity

The use of event correlation, be it based on rules or prerequisites, is to make inferences from known raw events. Suppose that a collection of raw events *E* implies some set of alerts *A*, given the event correlation mechanism. The inferences are said to be monotonically increasing if when we add a new event to *E*, all the alerts inferred by the original *E* are still inferred by the new larger *E*.

Most IDS event correlation mechanisms are monotonic in this sense, and this contributes to the problem of stress in alert analysis. Often the analyst will determine that an alert is be acknowledged as a false positive because an event was presented to the system after the alert was inferred. What is required, then, is a system that exhibits non-monotonicity, i.e. a system that allows the possibility of an alert to be retracted given evidence of some new event.

### 4.2. The Problem of Batch Re-Computation

The simplest approach to the problem of IDS monotonicity is to re-compute all inferences in set A whenever a new event presents itself, is retracted, or is updated. For example, the system could number each event and alert chronologically in the order of occurrence. If an event at spot *n* is retracted, then the system could retract all alerts at spots greater than *n* and re-infer the alert list *A*. Clearly, however, this approach is computationally inefficient.

### 4.3. TMS Basics

The fundamental idea of TMS is to keep dependency records of an episode of reasoning. We let nodes in a dependency network represent either facts or justifications. A classic example is as follows. Suppose a program has the following entries in its knowledge base:

Node-1: The patient has a cold.
Node-2: If the patient has a cold, he sneezes.
Node-3: If A, and A implies B, then B.

Here, Node-1 and Node-2 are facts and Node-3 is an inference rule. The program might infer a new statement

Node-4: The patient sneezes.

and record (Node-1 Node-2 Node-3) as the justification for Node-4.

We say that a node is currently active if it is a fact in a line of reasoning or if it has a valid justification. So, Node-4 is active in our example, as are the nodes which justify it. When a node transitions from active to inactive, then the set of justifications is examined to determine the trickling effect. In our example, Node-1 becoming inactive would of course cause Node-4 to become inactive. And if Node-1 were to become active again, then of course Node-4 would become active.

This simple illustration provides the basic idea of truth maintenance. In the following sections we discuss TMS in somewhat more detail and show how we apply it to the alert analysis problem.

### 4.4. The Structure of a TMS

A TMS is a back-end mechanism that keeps track of the inferences made by an event correlation system. When used as a component module for security management, it can be loosely coupled to a collection of IDSs or other event producing systems such that a dependency record for each inferred datum is made available to the TMS. The dependency record consists of the information from which the datum was derived. The TMS couches all inferred data and their dependency records into a dependency network, which shows how a datum is related to the entire body of data. The purpose of the TMS is to provide bookkeeping when the body of data is perturbed. For example:

- The TMS updates the dependency network when a datum turns out to be incorrect or changes over time.
- When a current line of analysis is not leading to a determination of a real threat or a false positive, the TMS can return to some previous state of the dependency network and proceed with a different line of analysis.
- Since the TMS keeps a record of work performed during a particular line of analysis, the analysis can be re-used in other lines of analysis.

There exist two main varieties of TMSs: justification-based (JTMS) and assumption-based (ATMS) systems [15]. They differ regarding the structure of the dependency record for each derived datum. The data structure for the JTMS dependency record is:

[in_nodes; out_nodes] → inferred datum

The in_nodes is a list of data that is believed to be true. The out_nodes is a list of data that is believed to be not true. The inferred datum is explicitly marked "in" if all the elements in the in_nodes list are actually true and all members in the out_nodes are demonstrably false (as opposed to being false by default). These lists are also called belief lists and disbelief lists, or truth lists and false lists respectively in the literature.

A dependency network consists of a list such dependency records. The network is updated when an datum is changed from "in" to "out" or vice versa. Belief is suspended for those data that depend on the changed datum, i.e. they are marked "out."

Note that a contradiction or constraint violation occurs when an update forces one to believe and disbelieve an inferred datum. The utility of the TMS in these situations is that the dependency network provides an immediate indication of the assumptions on which the contradiction is based.

As a simple illustration, consider the dependency network in Figure 3. We'll let data A, B, C, … represent an assumption set, i.e. a set of given IDS alerts, and let $I_n$ indicate an inference based on either some rule or prerequisite graph.

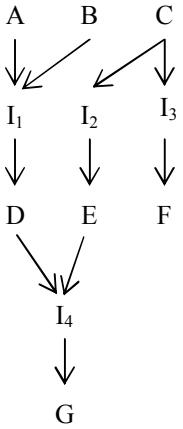


**Figure 3. A sample dependency network**

To simplify the example, assume that the dependency records include only in_nodes. The dependency network reflects the dependency relations of data provided by separate IDSs or other event producing systems. Node D is derived from nodes A and B via inference mechanism $I_1$, node E is derived from C via $I_2$, etc. In this manner, the JTMS records the following dependency records:

[A, B, $I_1$] → D
[C, $I_2$] → E
[C, $I_3$] → F
[D, E, $I_4$] → G

When nodes A, B, and C are in, the belief state is [A, B, C, D, E, F, G]. Suppose that the JTMS receives an instruction that node C is changed to out. The effects propagate through the network, and consequently the believe state is [A, B, D]. When only node C is in, the belief state becomes [C, E, F].

With the JTMS, only one assumption set can be explored at a time, whereas with an ATMS, several assumption sets can be explored simultaneously so that different lines of analysis can be performed simultaneously. The dependency information in ATMSs consists of a list of environments in which an inferred datum holds. Consider a set of possible assumptions, some of which are believed and others disbelieved, at given points in time. The set of possible assumptions can be pre-partitioned into an exhaustive finite number of environments which specify which assumptions can be believed at any point in time. For example, if the assumption set is [A, B, C] as in Figure 3, then the possible environments are [ ], [A], [B], [C], [A, B], [A, C], [B, C], and [A, B, C]. The data structure of an ATMS dependency record, then, is:

[ [environment1], [environment2], … ] → inferred datum

Each inferred datum in the ATMS dependency network is associated with a set of minimal environments from which it is derivable, i.e. those environments which include as few assumptions as possible. Following the example in Figure 3, the environments are recorded as follows:

[[A, B]] → D
[[C]] → E
[[C]] → F
[[A, B, C]] → G

The computation of belief states is the same as in the JTMS, e.g. when only C is out the belief state becomes [A, B, D].

JTMSs are suitable for single-context analyses, i.e. when alerts are investigated one at a time with a single line of reasoning. ATMSs, on the other hand, are suitable for multiple-context problems, when a single alert is investigated in the context of multiple possible environments. An advantage of a JTMS is that an explicit

explanation for an inferred datum is relatively easy, whereas an explanation with an ATMS consists only of the environments from which the datum is derived. The advantage of the ATMS, however, is that it is easy to change contexts and compare different sets of beliefs. In the work reported here, the JTMS was used as the foundation of the alert analysis system. The ATMS may be used for future experimentation.

### 4.5. Application of a JTMS to Alert Analysis

Most current IDSs rely on the security engineer discovering whether alerts are to be discarded as false positives or escalated to real security threats. When multiple alerts depend on a signal datum, usually multiple analyses are carried out, which is a duplication of work and time-consuming. Some IDSs may aggregate many of the same alerts into one, but nonetheless one prerequisite event may be the cause of many different types of alerts. If this one prerequisite is proven false, then all the other alerts which are brought up as a consequence of the prerequisite should likewise be removed. One of the advantages resulting from the use of a JTMS is the reduction in alerts for a security engineer to look into by removing the alerts which can be disproved.

How would a JTMS be used in practice? A JTMS uses a belief list and non-belief list (i.e. in_node and out_node lists). A belief list refers to a list of events or alerts that a security engineer believes to be true. Likewise, a non-belief list refers to a list of data or events that a security engineer believes to be false. This means that the security engineer must determine the lists' contents, and/or have an interface from an event producer such as an IDS that populates the lists. In the Figure 3 example, if one could disprove C and place it in the non-belief list, then both E and F would be automatically disproved and not be displayed.

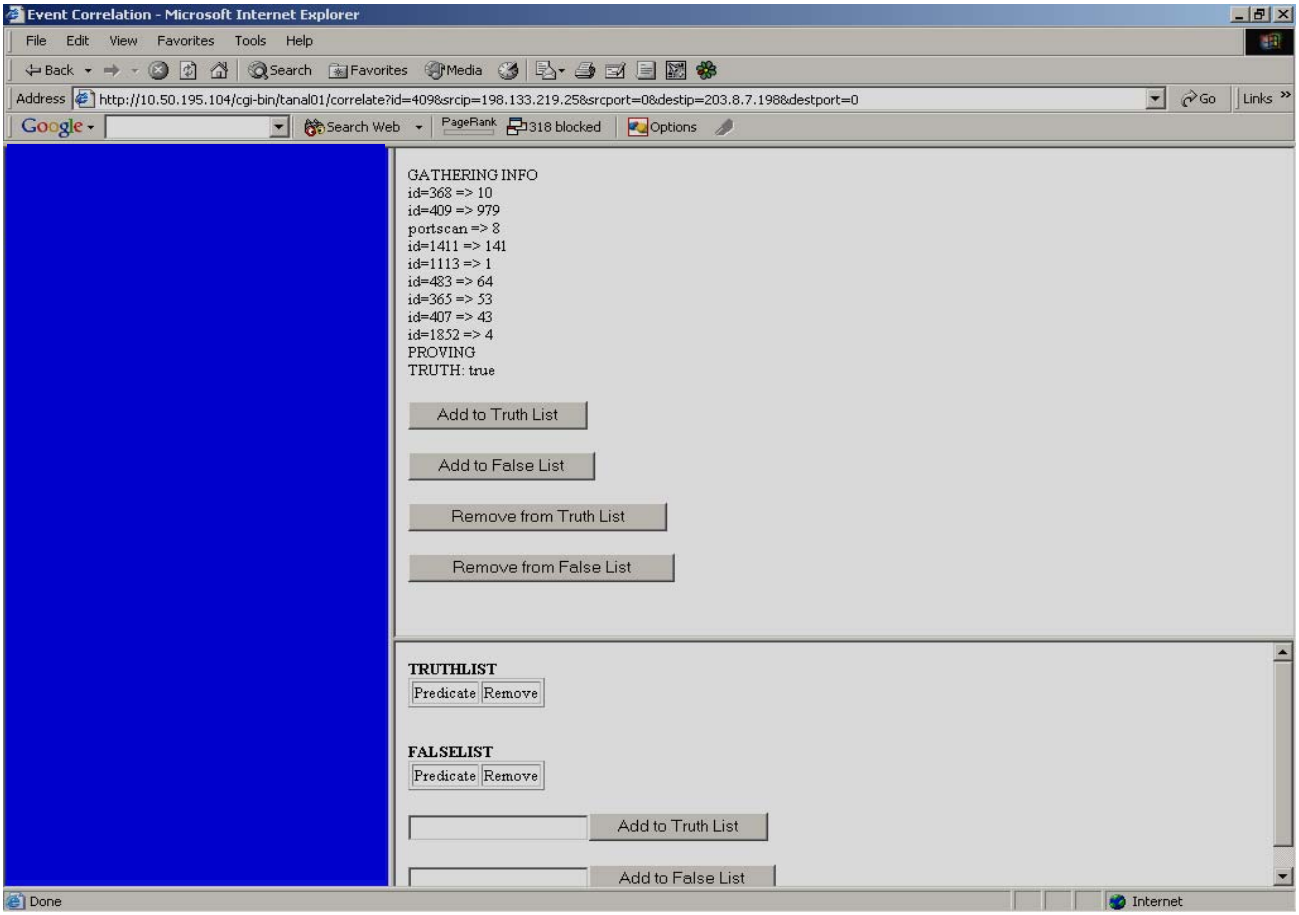


**Figure 4. Examining/manipulating belief and disbelief lists**

## 5. System Design and Operation

The JTMS alert analysis system was installed onto a web server which is accessible by web browsers from multiple analysts. The system consists of text files storing XML documents, text files to store the belief lists and disbelief lists, and Perl scripts to implement the functions for manipulating the dependency records as new items are asserted and retracted from the lists. The use cases of the system are: (i) add/remove data in the belief list, (ii) add/remove data in the disbelief list, (iii) view a resulting false positive list, (iv) view a resulting real threat list, and (v) view the analyses of peer analysts. The system inputs are events from IDS and other device logs and the data in the belief lists and disbelief lists provided by the analyst. The system outputs are a list of alerts which are likely to be false positives and a list of alerts that are possible threats, and explanations of each.

One begins using the system by clicking on an alert in the alert list (see the alert list in Figure 2), which brings up a screen whereby one can examine and manipulate the belief and disbelief lists for the alert, as shown Figure 4 (where the lists are labelled Truthlist and Falselist). If the lists are altered, then the flow returns to a modified Figure 4 type screen that shows the effect of such altering on other alerts – e.g. alerts other than the one in question may be declared as a false positive or a real threat if it's associated belief and disbelief lists have changed as a result of dealing with the original alert in question.

## 6. Test Results

The focus of the laboratory tests was to perform a subset of the JTMS operations under various conditions to get an idea of the speed of the operations vis-a-vis increasing complexity. Tests were performed on a Pentium III with a 797.433 MHz CPU. Other processes running on the machine used negligible CPU and pings to other machines on the network were less than 10ms. The experiments proceeded by gradually increasing complexity, first by gradually increasing the number of conjunctive events with 1, 3, 6, and 15 conjuncts, and secondly by increasing the number of events in IDS logs with 250, 500, 1000, 1500, and 2000 events. In each case, two tests were performed – one in which no events were in the belief list and one in which all events were in the belief list. Figure 5 shows the characteristics of the simplest test and the results. Figures 6 and 7 show the aggregated test results.

| Name | Conjuncts | XML Document |
|---|---|---|
| XML1 | 1 | <and><br><summaryidslog<br>summarytype="409"<br>count="100"><br></summaryidslog><br><srcipidslog type="409"><br></srcipidslog><br></and> |

| Condition Type | Condition | |
|---|---|---|
| XML file | XML1 | |
| Belief list: | Empty | |
| IDS log entries: | 250 entries | |
| | | |
| Start | Finish | Time |
| 13:53:01 EST | 13:53:07 EST | 6s |
| 13:53:23 EST | 13:53:29 EST | 6s |
| 13:54:07 EST | 13:54:14 EST | 7s |
| 13:54:36 EST | 13:54:42 EST | 6s |
| 13:55:36 EST | 13:55:43 EST | 7s |
| Average | | 6.4s |

**Figure 5. The simplest test and the results**

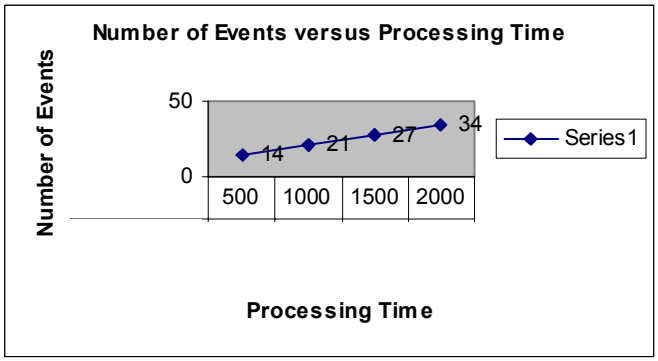


**Figure 6. Number of events vs. processing time**

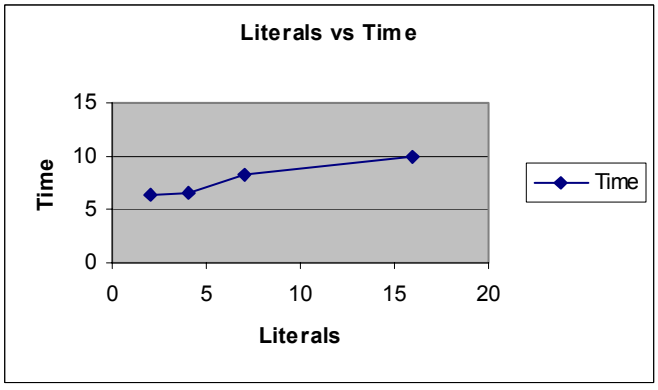


**Figure 7. Number of literals (conjuncts) vs. time**

## 7. Summary and Conclusion

This paper focused on the stage of alert analysis in security management. Rather than focus on better event correlation algorithms that produce fewer alerts and false positives, we took a large number of false positives as a given and concentrated on a method to manage and maintain the alert list.

Our approach builds upon a method that has issued from the AI community, called a truth maintenance system, that allows the construction and manipulation of dependency records. A dependency record reflects the beliefs and disbeliefs that should hold when a particular alert is deemed a false positive or real threat. Importantly, the system adjusts the detection of false positives and real threats when the belief lists and disbelief lists are perturbed. We described two types of TMSs, a JTMS and an ATMS. The JTMS was used as the foundation for the current system.

We believe that the JTMS approach could complement other methods of dealing with the overwhelming number of alerts presented to security analysts. For example, using the IDMEF alert format [16], the CorrelationAlert class could be used for tracking purposes based on the logic of the JTMS. Similarly, the 'partof' relation of the M2D2 model [17] could be used for the tracking purposes combined with JTMS logic, along with the additional information accounted for in the M2D2 model: information related to the characteristics of the monitored information system, information about the vulnerabilities, information about the security tools used for the monitoring, and information about the events observed. The method of alert verification [18], in which a process is launched in response to an alert that checks to see whether an accompanying attack has succeeded or not, and suppressing the alert if not, could be complemented by the JTMS logic. The chronicles formalism described in [19] could be used as a vehicle to express the JTMS logic. These are areas for further research and experimentation.

Finally, since the JTMS retracts and asserts alerts on an on-going dynamic basis, we are concerned with the effect of the scheme on false negatives. Intuitively, it is logically possible that a misjudgement on the part of a security analyst, e.g. in the case of a previously unknown threat/worm/virus, could result in an increase in false negatives and attacks going undetected. This is an outstanding problem and also an area for further research.

## References


[1] Netforensics, Inc. New Jersey, USA. www.netforensics.com

[2] Intellitactics, Inc. Toronto, Canada. www.intellitactics.com

[3] CERT/CC Statistics 1988-2005 www.cert.org/stats/cert_stats.html#incidents.

[4] S. Kumar and E. Spafford, "A pattern matching model for misuse intrusion detection," *Proceedings of the 17th National Computer Security Conference*, 1994

[5] K. Llgun, R. Kemmerer, and P. Porras, "State transition analysis: a rule-based intrusion detection approach," *IEEE Transactions on Software Engineering*, Volume 2, Issue 3 (March 1995), pp. 181 – 199.

[6] J. Anderson, "Computer security threat monitoring and surveillance," Technical report, James P. Anderson Co., Fort Washington, PA, 1980.

[7] Intrusion Detection Systems List and Bibliography, www-rnks.informatik.tu-cottbus.de/en/security/idsbody.html

[8] SNORT: www.snort.org

[9] P. Yarng, P. Ray, D. Maher, "Profiling Cyber Attacks using Alert Regression Profiles," *IEEE Globecom* 2003 Communications Security Symposium.

[10] A. Lam and P. Ray, "Security alert management in E-business networks", *International Conference on E-Business* (ICEB2004), Beijing China, December 2004.

[11] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework," *IEEE Symposium on Security and Privacy*, 2002, pp.202-215.

[12] S. Benferhat, F. Autrel, and F. Cuppens, "Enhanced correlation in an intrusion detection process," *Second International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security* (MMM-ACNS 2003), pp. 157-170.

[13] P. Ning, D. Reeves, and Y. Cui, "Correlating alerts using prerequisites of intrusions," Technical Report TR-2001-13, Department of Computer Science, North Carolina State University, 2001.

[14] J. Doyle, "A truth maintenance system," *Artificial Intelligence* 12(3), pp. 231-272, 1979.

[15] J. de Kleer, "Choices without backtracking," *Proceedings of the American Association of Artificial Intelligence Conference*, 1984.

[16] The Intrusion Detection Message Exchange Format, http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-14.txt, 2005.

[17] Morin et al., M2D2: A Formal Data Model for IDS Alert Correlation, RAID 2002.

[18] Kruegel and Robertson, Alert verification: Determining the success of intrusion attempts, DIMVA2004.

[19] Morin and Debar, Correlation of Intrusion Symptoms: an Application of Chronicles, RAID 2003.