

# TRINETR: Facilitating Alerts Analysis and Response Decision Making

Jinqiao Yu  
*Department of Mathematics and Computer  
Science  
Illinois Wesleyan University  
P.O.Box 2900  
Bloomington, IL 61701  
jyu@csee.wvu.edu  
Tel: 309-556-3535 Fax: 309-556-3864*

Ramana Reddy, Sumitra Reddy  
*SIPLab, CERC  
Department of Computer Science and  
Electronical Engineering  
West Virginia University  
Morgantown, WV 26505  
{ramana.reddy, sumitra.reddy}@mail.wvu.edu,  
Tel: 304-293-7225 Fax: 304-293-7541*

## Abstract

*Due to many inherent deficiencies and flaws, current intrusion detection systems (IDS) are plagued by numerous problems. Intrusion Detection Systems are often inefficient and ineffective when used alone. IDS products need to be fully integrated into the security defense line. Intrusion alert analysis and management are crucial in achieving this. In this paper, we describe an intrusion detection alert management and analysis system, called TRINETR, which can serve as a layer above IDS to make the use of IDS more efficient and intrusion alerts more accurate and meaningful as well as provide real time security decision making support. Implementation and Testing of a prototype system are also presented in this paper.*

## 1. Introduction

In response to the daunting threats of rapidly growing cyber attacks, a promising approach to deterring intruders is to detect, record and analyze attacks and finally prosecute malicious attackers using the evidence collected through the above process. Computer and network forensics is such a new emerging discipline. It involves capturing, recording, and analysis of network events in order to discover the source of security attacks or other problem incidents. It attempts to prevent hackers from attacking a system and searches for evidence after an attack has occurred.

Intrusion detection system (IDS) is an indispensable part of computer and network forensics. It is deployed to monitor network and host activities including data flows and information accesses. Its main purpose is to detect real-time, ongoing intrusions and alert system

administrators so that appropriate actions could be taken to stop the intrusions or discover the damages if the attacks had already succeeded. In recent years, intrusion detection has begun to gain wide acceptance in enterprises as a necessary and worthwhile investment in security. Despite the recent success of IDS, it is still presenting many weaknesses. Some of these weaknesses are so critical that they may hinder its future widespread acceptance. Some of the weaknesses are listed in the following:

- *Alert Flooding.* Current IDS systems are prone to alert flooding. System administrators can easily be overwhelmed by the vast amount of log information and alerts, especially false positive alerts.
- *Too Many False Positives Alerts.* Due to the quality of detection signatures and the difficulty to define precisely the boundary between normal and abnormal behaviors, both signature-based and anomaly detection tend to generate too many false positive alerts, i.e., they generate excessive false alerts upon normal network and host behaviors. Consequently, real malicious and important alerts are often buried under tons of false alarms.
- *Lack of Context Awareness.* Current IDS products are often unaware of the network and host system context they are monitoring. For instance, an IDS often does not know whether the critical host it is monitoring is running a Windows operating system or a Unix operating system. Therefore, without knowing who it is protecting, an IDS product cannot distinguish those real harmful attacks from random probes. A direct consequence

of this weakness is the generation of excessive meaningless alerts – alert flooding.

The last weakness cannot be solely blamed on IDS products. It is often due to the lack of a systematical integration of IDS products into a network's security defense. IDS products are isolated from the protected systems. One common solution to this weakness is alert filtering and signature tuning. However, alert filtering and signature tuning require the security administrators to understand the IDS signatures thoroughly. Filters are also difficult to maintain. For instance, different filters are needed for different IDS products since each IDS system has its unique signature sets.

Detection of ongoing intrusions is just the beginning of the battle against malicious attacks. Prompt and appropriate responses to thwart these attacks are more crucial in winning this battle and saving valuable assets. However, since IDS products are not fully integrated into the network and system security environments they are monitoring, alerts generated by them cannot provide effective security decision support against threats. They often provide only a short description of the generated alerts. No appropriate real-time security solutions or advices corresponding to the alerts emanated are provided. It is left to the expertise and experience of the system administrators to search for security solutions. Consequently, response to ongoing attacks could be delayed and the best time to stop attacks could be missed.

To address the above weaknesses challenging IDS products and facilitate more efficient and more effective detection and security response, we are developing an intrusion alert management and analysis system, called TRINETR, which can serve as a layer about intrusion detection. In TRINETR, a generic architecture is developed to manage, analyze, deal with alerts and coordinate multiple IDS products. The core of this architecture is a framework that incorporates dynamic network and host assets information and commonly known vulnerability knowledge as well as pre-specified security policies into the alert evaluation and decision making support process. No alert filtering, signature tuning and corresponding administration overhead are needed. This framework can not only be used to provide alert prioritization but also to provide appropriate real time security solution advices to support decision making for real harmful alerts.

The outline of this paper is as follows: Section 2 reviews the related work. Section 3 gives an overview of the generic architecture that is used to incorporate multiple heterogeneous IDS sensors. Section 4 details the design of the knowledge-based alert evaluation and security decision support framework. In Section 5, we describe the alert correlation. In Section 6, we discuss the implementation and testing of our prototype systems. Section 7 wraps up the paper with a brief conclusion.

## 2. Related Work

One of the active research areas in the intrusion detection community is the development of technologies to manage, analyze and interpret intrusion detection alerts produced by IDS products. One research study similar to our work is M-Correlator [1]. They used a mission-impact-based approach in the analysis of security alerts produced by heterogeneous information security devices. In M-Correlator, Nmap is used to generate a topology map of the protected network. After that, M-Correlator develops a relevant score that assesses per alert the likelihood of successful intrusion. The topology map is similar to the network agent in TRINETR. However, Nmap can only give a rough map of the protected network. For instance, the results obtained from Nmap are still guesses. It may not be able to tell whether it is exactly Windows Me or Windows 2000 Pro. Furthermore, M-Correlator does not provide decision support after alert ranking.

Lee and Kim [2] analyzed the requirements of the decision support system for network security management. A prototype is also implemented to provide the basic functions of a design of the described system, but the design is only used for intrusion prevention. No real time security decision assistance is provided.

In TRINETR, a detailed network topology is maintained so that it can provide accurate topology information instead of guessing. In addition, TRINETR also provides real time security decision support.

In the next section, we will describe the generic architecture presented in the TRINETR, which can incorporate multiple IDS sensors.

### 3. TRINETR Architecture

The TRINETR architecture (Figure 1) consists of four main components: (1) Alert Aggregator, (2) Knowledge-based Alert Evaluation and Security Decision Support Component, (3) Alert Correlator and (4) Synthetic Alert Report Generator. Each of these four components will be briefly described in the following:

**3.1 IDS Alert Aggregator** Alert Aggregator is responsible for collecting alerts from different IDS products, converting them into a standard format and clustering them based on defined similarities. The standard format we chose is the Intrusion Detection Message Exchange Format (IDMEF) [3]. After appropriate aggregation, the amount of alerts could be significantly reduced. Scattered, random alerts will be aggregated into related alert groups to be further analyzed by other components. More details of this component can be found in [4].

**3.2 Knowledge-based Alert Evaluation and Security Decision Support Component.** This component evaluates aggregated alerts using knowledge about known vulnerability requirements of the attack and targets configuration and system information. A priority is assigned to each alert. System immune alerts and false positive alerts will be either eliminated or marked as lower priority alerts. Security solutions to higher priority, real harmful alerts will be generated using known security solutions in accordance with security policy and contingency plans. The suggested security solutions then can be used to facilitate security decisions as well as automate contingency responses. This component will be described in detail in Section 4.

**3.3 Alert Correlator.** The main purpose of this component is to find logical relationships among the alerts. Clusters of alerts are correlated together in the hope of finding where the initial attacks come from and where they actually end up. This component can also be used to find patterns among series of attacks. Alert Correlator will be described in Section 5.

**3.4 Synthetic Alert Report Generator** Intrusion reports containing both synthetic alerts providing an overall picture of once isolated alerts and the security solutions based on the predefined enterprise security and contingency plans will be generated. Statistical information of intrusions based on the attack source, target, exploited vulnerabilities, attack classification

and other useful data will also be calculated and provided in graphical forms.

The core of TRINETR architecture is the knowledge-based alert evaluation and security decision support component. In the next section, we will describe this component in details.

### 4. Knowledge-Based Alert Evaluation and Security Decision Support Framework

This component is a framework that provides two main functions: (1) evaluate alerts against network and host

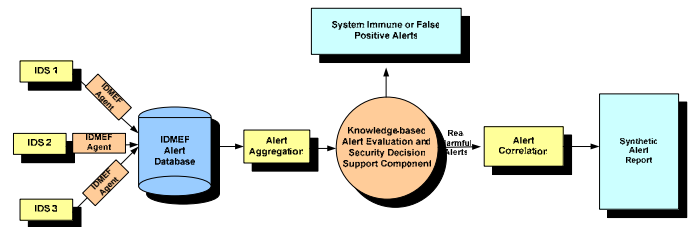


Figure 1 TRINETR Architecture

asset information, using known vulnerability requirements of the attacks, (2) provide appropriate security defense advices based on the target's current topology, configuration, and other related system information in accordance with pre-specified security policies for security decision support. This functionality can be further extended to provide automated intrusion response mechanisms. The framework with a typical usage scenario is depicted in Figure 2. The following sections illustrate each of the important components involved in the alert evaluation and security decision support process.

**4.1 Host Agents** Host Agents run on each of the critical hosts in the protected network domain. They collect low-level details of the host's system and configuration information. For instance, the information gleaned can include OS version, service pack version, installed hotfixes, running services and their versions, kernel version, installed modules etc.

**4.2 Coordinator Agent** The coordinator agent coordinates with host agents. Host information collected from host agents is first sent to the coordinator agent. The coordinator agent then stores the information in the network and host asset knowledge base. Meanwhile, if the information needed by an evaluation process is not available in the knowledge base, the coordinator agent then requests

the corresponding host agent to collect that information and send it back.

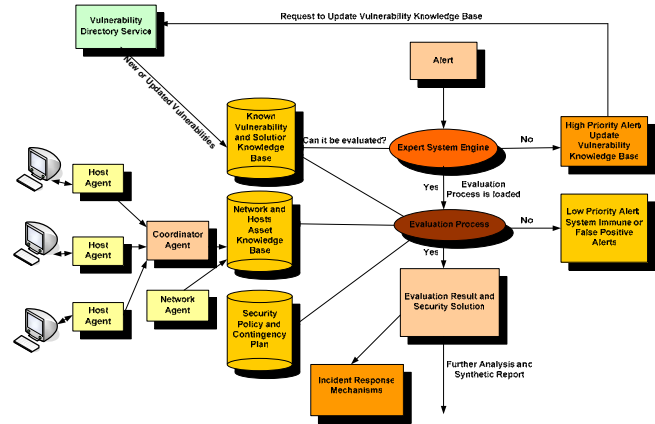
**4.3 Network Agent** The network agent maintains an internal topology map of the protected network. The network agent identifies the available assets on the network, their current state (up or down), IP address to hostname match, open ports and the applications behind those ports, active TCP and UDP network services and hardware information, etc. The information is then stored in the network and host assets knowledge base. The network agent runs at intervals to maintain an updated topology map of the protected network.

**4.4 Network and Host Assets Knowledge Base** This knowledge base contains dynamic network and host asset information. The network and host asset information collected by the above agents is stored in this knowledge base to be used in the evaluation process.

**4.5 Known Vulnerability and Solution Knowledge Base** Currently, there are a few of independent organizations or commercial companies monitoring the emergence of new vulnerabilities. New vulnerabilities with the infected system information are then released to the public along with corresponding security solutions. These vulnerability monitoring and reporting organizations, to name a few, include MITRE [5], SecurityFocus [6], CERT [7], and ISS [8]. While each of these organizations or companies maintain their own separate database, a common naming scheme to cross-link the vulnerabilities was lacking. The foundation of CVE (Common Vulnerabilities and Exposure) [5] solved the problem. CVE is a list or dictionary that provides common names for publicly known information security vulnerabilities and exposures. Although CVE provides a common name scheme, it does not detail the infected system requirements and the corresponding security solution information. On the other hand, some of the vulnerability monitoring organizations, such as SecurityFocus’s bugtraq, as well as the product vendors whose products are infected often provide detailed information about systems that could be infected by the vulnerabilities, and provide security solutions in the forms of patch download, configuration suggestions and other methods to plug the security holes. Furthermore, after the emergence of CVE naming scheme, many IDS products use CVE taxonomy to reference their alerts. Therefore, by cross linking CVE vulnerability naming with the infected system requirements as well as security solutions, we

can accurately evaluate IDS alerts as well as provide appropriate security decision support.

**4.6 Vulnerability Directory Service** New vulnerabilities are added in the knowledge base by a vulnerability directory service as they are announced. A complete and “stay-current” knowledge base makes the alert evaluation more accurate and complete.



**Figure 2. Knowledge-based Alert Evaluation and Security Decision Support Framework**

**4.7 Security Policy and Contingency Plan** A security policy is a plan outlining what the company's critical assets are and how they must (and can) be protected. A contingency plan clearly states what must be done in various emergency situations; the main idea is to minimize and limit damage. By incorporating security policies into the alert evaluation process, we can ensure that critical assets are better protected and the alert priority process is in concordance with the enterprise network’s interests. In addition, the contingency plan provides some of the guidance in face of malicious threats so that appropriate defense mechanisms could be taken to deter intrusion and minimize damages.

**4.8 Expert System Engine** In our alert evaluation and security decision support framework, the evaluation processes are executed by an expert system engine. After the aggregation process, IDS alerts are asserted as “facts” into the knowledge base. The evaluation process is then triggered by the assertion of new alerts. We bring an expert system into the framework for the following reasons: (1) IDS products should never be standalone. They should be used closely with security policies, contingency plans and other defense mechanisms. In reality, security policies and

contingency plans are often implemented as predefined rules. Therefore, after the evaluation process, certain defense actions triggered by these rules can be executed by the expert system to deter intrusions. (2) By representing alerts as facts in the knowledge base, we can query the knowledge base to find relationships between facts (alerts). This is the primary purpose of our third component, alert correlator, in the TRINETR architecture.

#### **4.9 Alert Evaluation and Security Decision Support Process in a Typical Scenario**

After alerts are clustered in the aggregation process, a representative alert is sent to the evaluation process. This meta alert contains the following information: a unique alert ID, source, target, time, classification and reference. Generated alerts are asserted as facts into the expert system's knowledge base. After a new alert is generated, an evaluation process is loaded to analyze the alert. In the following example, the alert refers to CVE-2001-0341[9], which is a remote root gain vulnerability infecting windows systems.

The evaluation process first queries the vulnerability knowledge base to see whether CVE-2001-0341 vulnerability information is available or not. If not, the alert is marked as a higher priority alert and the vulnerability directory service is notified to update its knowledge base. Otherwise, the vulnerability information concerning CVE-2001-0341 is retrieved from the knowledge base. The evaluation process identifies the target and uses network topology information to roughly evaluate alerts. If the target's OS is in the vulnerability list, the process then goes to next step. Otherwise, the process marks the alert as a lower priority alert. In this example, the evaluation process goes to the next step to retrieve target 192.168.1.19's host information. The next step is the actual evaluation process. The evaluation process compares the target information against the vulnerability requirement information. A relevance score is calculated based on matches of the information. The relevance score ranges from 0 to 1 with 1 meaning a perfect match and 0 meaning no match at all. After relevance score is calculated, the evaluation process then retrieves the security solution in the fact. The evaluation process now checks the security solution with the target configuration. If the security solution is already in the target configuration, the alert is then marked as system immune alerts. If not, the evaluation process starts the next step. The evaluation process then queries the security policy and contingent plan knowledge base. In the security

policy, target (192.168.1.19) is listed as a critical asset. Therefore, the evaluation process marks the alert as extremely high alert. The associated rule in the contingent plan then triggers response mechanisms, which may include paging the system administrator immediately, downloading the patch identified in the security solution or terminating connection to the source.

After alerts are evaluated, they are sent to the next component: alert correlator.

### **5. Alert Correlator**

This component is used to correlate scattered alerts in the hope of finding logical relations among them. More specifically, the objectives of correlating alerts include the following:

- By finding the logical relations among alerts the attacker's intrusion strategies could be found and his final objectives as well as each step carried out to achieve the final objectives could be disclosed.
- By finding how the intrusions are related, we can find different vulnerabilities in network systems that could be used collectively to breach the system. As a result, vulnerabilities could be made up as a whole, and better protection could be achieved.
- By finding the relations among intrusions, we can find the prerequisite of some intrusions. Therefore removing those prerequisites can minimize the chance of those intrusions to succeed.
- By discovering causal relations among intrusion events, we can predict the occurrence of certain intrusion given the observations of its precursor events.

In this component, we explored two approaches: a relatively simple attack pattern-based approach is used to correlate the aggregated alerts in real time; a time series analysis is used offline in order to find more obscure relations and extract correlation rules.

#### **5.1 An Attack Pattern-based Alert Correlation**

In this approach, isolated alerts are aggregated into different groups according to certain common characteristics. These aggregations are called "attack patterns". Each attack pattern is a clustering of alerts

having certain characteristics in common. Currently, the correlation process aggregates alerts based on three axes: source, target and classification.

This aggregation-based alert correlation is relatively rough. By aggregating alerts into different attack patterns based on the above three axes, we can find many loose relations among them. For instance, after aggregating alerts with the same target and belonging to same classification, we can find a distributed attack against a single target using the same attack such as a DDoS attack. We can also find an attacker tampering the same target with a series of attacks by aggregating alerts with the same source and the same target.

Based on the combination of the three axes: source, target and classification, we can correlate alerts into seven different attack patterns.

Firstly, we use all three axes: source, target and classification to group alerts.

Pattern 1: Alerts with the same source, the same target and belonging to the same classification. This allows us to detect, for instance, the same attacker who is launching a series of Buffer Overflow attacks against a single Web server.

Secondly, we eliminate the classification condition.

Pattern 2: Alerts with the same source and the same target. This pattern allows us to detect an attacker who is attacking a target, for instance, a FTP server with different methods. By survey past attacks, we can find the most attempted approach to breach an application.

Thirdly, we eliminate the target condition and only keep the source axis.

Pattern 3: Alerts with the same source. This pattern clusters all the attacks coming from the same attacker. It collectively displays all the targets the attacker attempts and all the attack methods the attack exploits. For instance, it can reveal the attack steps carried by the same hacker: he first tried a couple of probes to find active IP address, then he launched several different Buffer Overflow attacks against a single target etc.

The fourth attack pattern comes when only the source and classification conditions are used.

Pattern 4: Alerts with the same source and classification. This pattern allows one to detect the

same attacks launched from a single attacker. It shows, for instance, an attacker who is exploiting a RPC attack against multiple mail servers in a protected network.

If we eliminate the source condition and keep the target and classification, we get the fifth pattern.

Pattern 5: Alerts with the same target and classification. This pattern allows one to detect, for instance, a distributed attack from different sources against a single target and exploiting the same vulnerabilities. A distributed DNS attack from multiple sources against the same target can be caught by this scenario.

If only the target condition is used, we establish the sixth pattern.

Pattern 6: Alerts with the same target. This pattern allows one to find all attacks against the same target. It can show the different exploits that have been attempted against the same target and from which sources.

The seventh pattern only deals with attacks exploiting the same methods.

Pattern 7: Alerts with the same classification. This pattern reveals how many times the same exploiting methods have been used and against which targets and from which sources. This pattern allows one to survey the same exploiting methods, for instance a newly discovered vulnerability in IIS 5.0, having been attempted against the protected network.

This attack pattern based approach is used to correlate alerts in real time. However, it can only find shallow relations among attacks. More sophisticated and obscured logical relations among independent alerts cannot be extracted through this approach.

In many applications of interest one is faced with the problem of identifying precursor events for extraordinary phenomena. Well known examples include earthquakes and financial market crashes, where the benefits of determining reliable precursors is of great importance. Network security is another field in which the determination of reliable precursors can be very valuable in detection and even prevention of malicious attacks. For instance, if certain abnormal activities preceding a DNS attack could be identified as precursor events, system administrators could prevent or minimize the detrimental effects of the

attack by monitoring the occurrence of those precursor events and taking appropriate actions after that. Identifying precursor events from observations of past events is the study of time series analysis.

## 5.2 Time Series Analysis

Time series analysis [10] has two main goals: (1) identifying the nature of the phenomenon represented by the sequence of observations, and (2) forecasting (predicting future values of the time series variable). Both of these goals require that the pattern of observed time series data is identified and more or less formally described. Once the pattern is established, we can interpret and integrate it with other data. Regardless of the depth of our understanding and the validity of our interpretation (theory) of the phenomenon, we can extrapolate the identified pattern to predict future events.

In our second correlation approach, we explore the application of time series analysis to the offline alert correlation process. The study is adapted from [11] and [12]. In this approach, intrusion alerts are considered as events. The occurrence of these events corresponds to a series of time events. Granger Causality Test[12] is applied to extract precursor rule and test the causality relations among them. The key strength of the approach is that it does not depend on prior knowledge about alerts, for instance the pre-specified correlation rules. In this sense it is similar to the anomaly-based detection. Intelligent attackers do not follow “rules”. They often intentionally disguise their attack patterns to avoid detection. Therefore a rule-based correlation approach cannot find new correlation patterns just as signature-based detection cannot find new intrusion methods. In addition, it is difficult to define exhaustively all of the correlation patterns. A slight change in attack sequence may result in a new correlation pattern. The GCT test can also list a few correlation candidates based on GCI values. This screening can facilitate security analysts to narrow down their targets and therefore reduce their workload.

After exploring this approach, we find that although it can find causal relations and extract certain precursor rules among alerts, domain expertise is still needed to make decision. Furthermore, the frequencies of many attacks are often not significant enough to be used in the analysis. Therefore this approach, at best, is more suitable to be used in knowledge-training of a correlation system in defining correlation patterns or rules.

## 6. Implementation and Testing

We have implemented a prototype system to testify our design. We used two different network-based IDS: Snort and Prelude. These two IDSs were installed in a Linux system to monitor a subnet with heterogeneous operating systems and different configurations. The alert aggregation process was implemented as Perl scripts. For each IDS, a Perl script was written. This Perl script served as the IDMEF agent to that IDS sensor, converting its alerts into the standard IDMEF format and storing alerts into a relational database. Both host and network agents were also implemented in Perl. A Jess Expert Engine was deployed to be used in the evaluation process. Evaluation processes were implemented as Java classes and are loaded on demand by the Jess Engine to evaluate the corresponding alerts. The attack pattern-based alert correlator was also implemented as a java class. We also implemented the time series analysis algorithm to work on alerts stored in the database before being aggregated because the actual counts of attacks are crucial in the statistical analysis. Finally, a PHP front-end was provided as a management console to view the alerts and information asset information.

We then extensively tested the prototype system. We noticed that during our experiment, the aggregation process can significantly reduce the amount of alerts by roughly a factor of 10 by removing duplicates and obvious irrelevant alerts. To test the evaluation and security decision support performance, we deliberately designed our attacks in order to generate system immune alerts and false positives. For instance, we attacked one of our Linux hosts with several IIS buffer overflow attacks, and attacked one of our Window 2000 machines with Service Pack 3 installed containing the necessary hotfix with an attack exploiting certain vulnerability in Service Pack 2. However during the experiment, all of the attacks were detected by Snort and Prelude, and some of the generated alerts were marked with the highest severity, but the knowledge-based evaluation process successfully evaluated them as system immune alerts. Furthermore, we also launched some exploits which would be certainly vulnerable to the targets. This time the alert evaluation process not only marked them as high priority alerts, but also suggested to us the corresponding security solutions to help make security response decisions.

## 7. Conclusion and Future Work.

In this paper, we have presented the need for intrusion detection alert management, analysis and corresponding security decision support and have described our research in this area. Our proposed architecture consists of four main components: alert aggregator, alert evaluation and security decision support component, alert correlator and synthetic alert generator. The purpose of this work is to provide a layer above intrusion detection and to make intrusion detection more accurate, easier to use and provide appropriate security solutions to system administrators to facilitate decision making as well as to automate appropriate processes. We are planning to integrate some automatic security management processes including patch management and auto configuration into the defense system.

## 8. Reference:

- [1] P. A. Porras, M. W. Fong, and A. Valdes “A Mission-Impact-Based Approach to INFOSEC Alarm Correlation”, In Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID 2002) (October 2002)
- [2] J. S. Lee and S. C. Kim “Design of the Decision Support System for Network Security Management to Secure Enterprise Network”, In Proceedings of the 4th Information Security Conference (ISC 2001) (October 2001)
- [3] INTRUSION DETECTION MESSAGE EXCHANGE MESSAGE FORMAT (IDMEF), <http://search.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-01.txt>. (2004)
- [4] Yu, J., Reddy, R., Selliah, S., Kankanahalli, S., Reddy, S., and Bharadwaj, V., “TRINETR: An Intrusion Detection Alert Management System”, In Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE-2004) Enterprise Security Workshop, (June, 2004)
- [5] COMMON VULNERABILITIES AND EXPOSURES (CVE), The MITRE Corporation. <http://www.cve.mitre.org>, (2004).
- [6] SecurityFocus <http://www.securityfocus.com> (2004)
- [7] CERT, <http://www.cert.org> (2004)
- [8] ISS (Internet Security Systems) <http://www.iss.net> (2004)

[9] CVE-2001-0341

<http://www.securityfocus.com/bid/2906>, (2004)

[10] Fuller, Wayner A. *Introduction to Statistical Time Series*, Second Edition, Wiley Series in Probability and Statistics. (1996)

[11] Cabrera, J.B.D. and Mehra, R.K., “Extracting precursor rules from time series – a classical statistical viewpoint”, In *Proceedings of the Second SIAM International Conference on Data Mining*, pp. 213-228. (2002)

[12] Qin, X. and Lee W., “Statistical Causality Analysis of INFOSEC Alert Data”, In *Proceedings of RAND 2003*, pp. 73-93, (2003)