Alert Fusion for A Computer Host Based Intrusion Detection System

Chuan Feng, Jianfeng Peng, Haiyan Qiao, Jerzy W. Rozenblit
Electrical and Computer Engineering Department
The University of Arizona
Tucson, Arizona 85721
Email: {fengc, jeff, haiyanq, jr}@ece.arizona.edu

Abstract

Intrusions impose tremendous threats to today's computer hosts. Intrusions using security breaches to achieve unauthorized access or misuse of critical information can have catastrophic consequences. To protect computer hosts from the increasing threat of intrusion, various kinds of Intrusion Detection Systems (IDSs) have been developed. The main disadvantages of current IDSs are a high false detection rate and the lack of post-intrusion decision support capability. To minimize these drawbacks, we propose an event-driven intrusion detection architecture which integrates Subject-Verb-Object (SVO) multi-point monitors and an impact analysis engine. Alert fusion and verification models are implemented to provide more reasonable intrusion information from incomplete, inconsistent or imprecise alerts acquired by SVO monitors. DEVS formalism is used to describe the model based design approach. Finally we use the DEVS-JAVA simulation tool to show the feasibility of the proposed system.

1 Introduction

The scale and intensity of information attacks have risen over the years. These attacks usually cause severe loss to individuals and businesses, and threaten the security of a nation. Routine authentication and access control mechanisms have proven inadequate in preventing attacks. Information security infrastructure has relled more on Intrusion Detection Systems (IDSs) [1]. Generally, an IDS detects unwanted manipulations to a system, such as network attacks against vulnerable services, data driven attacks on applications, host based attacks such as privilege escalation, unauthorized access to sensitive files, and malwares.

Many techniques have been applied within IDSs to detect intrusions, such as expert systems, state transitions, probabilistic approaches, process profiling, etc. These various IDSs can be categorized into two classes: signature-based and anomaly-based detections [2]. Signature-based IDSs compare current events with known attacks and look for similarities. The major limitation of the signature based detection method is that it cannot detect novel attacks. Anomaly based IDSs do not share this limitation since they model normal behaviors and attempt to identify abnormal activities of computer performance metrics. Fore this reason, anomaly detection exceeds signature based detection. However, anomaly detection systems are seldom implemented due to their high false alert rate. Our intention is to build a host based intrusion detection system that can detect novel attacks with a low false alert rate.

All IDSs are based on the belief that an intruder's behavior has measurable differences from that of a legitimate user. Therefore how to manipulate the incoming event is a big issue. Traditionally, IDSs treat an incoming event as a single signal, which increase both false positive and negative alert rates. Our solution is to apply Subject-Verb-Object multi-point monitors, which can detect deviations from normal behavior and trigger different alerts.

The issue of the SVO model is that when an IDS runs, it will generate a tremendous number of events in real time. Hence, it is inefficient and impractical for the system administrator to react promptly. To minimize the cognitive overload on administrators, yet present them with the information they require, an alert fusion model is implemented to collate the alerts from different sensors which provide inconsistent information [6] [8] [11] [13] .

The proposed fusion model employs multi-level processing architecture, which identifies categories of techniques and algorithms for performing the specific functions [3] [4]. In addition to the source data and model interface, four processing levels are involved: source preprocessing, alert data normalization, spacial alert fusion, and temporal alert fusion. A knowledge base is applied to improve the performance of the fusion processes. Each processing level has knowledge of the nature of the problem.

This paper is organized as following: section 2 describes the SVO model and system architecture of the proposed

Proceedings of the 14th Annual IEEE International Conference and
Workshops on the Engineering of Computer-Based Systems (ECBS'07)
0-7695-2772-8/07 $20.00 © 2007 IEEE

IDS. Section 4 presents the multi-level alert fusion model. In section 3, we discuss the DEVS model design and simulation using DEVS-JAVA in detail.

2 Event-Driven Intrusion Detection System

2.1 SVO Model

Subject-Verb-Object (SVO) is a linguistic typology concept we use to describe the intrusion event. The Subject refers to the origin of the action; the Verb indicates the subject's action; and the Object denotes somebody or something involved in the subject's performance. Any event happening inside a computer has its subject, which most often is a running process. Here we explicitly differentiate the subject from the owner, who is the actual user that runs the process. Similarly, the verbs and objects can be identified. Putting the subject, verb and object into a triple allows us perform fine grain analysis of the events.

In order to reduce the triple events combination, a critical event set is defined so that events out of this set either pose no threat or are trivial threats that can be ignored. In the current stage of our research, we focus on the critical set that contains important files we want to protect.

2.2 System Architecture

With the SVO model and critical event set, the IDS architecture is shown in Fig 1. In the figure, there is an SVO anomaly detection engine, which consists of separate Subject, Verb and Object detection models; an alert fusion engine; and a decision support model. They address problems such as how to detect an abnormal event, what to do with an alert and how to assist the system administrator when a threat is identified.

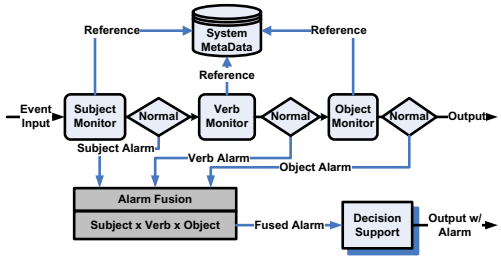

Figure 1. IDS system

3 Alert Fusion Model

The goal of the alert fusion model is to get more consistent intrusion information based on the alert data coming from the anomaly detection engine. The multi-level processing model of the fusion model is shown in Fig. 2.

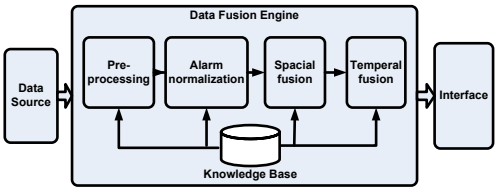

Figure 2. Alert fusion model

The Data Source at the left end of Fig. 2 indicates alerts from different anomaly monitors. The right end of the figure shows the system interface to the fusion system. Four processing levels are presented in the figure. The first one is Source Preprocessing, which synchronizes the information flow from different sensors and reduces data overhead for further levels. The second processing level is Alert Normalization, which transforms different alerts into a consistent set of scale. The third level is Spacial Alert Fusion, which fuses alerts from different anomaly detection monitors. The fourth level is Temporal Alert Fusion, which combines alerts within a time sequence and gives more useful intrusion information. The four levels are described below in greater detail.

3.1 Source Preprocessing level

Source Preprocessing, the first step of alert fusion, addresses reducing the overhead alert information and distributing data to appropriate processes.

Following the SVO model, an event, such as "process A read file B" will be divided into separate parts for analysis. Because each anomaly detection monitor has its own operational delay, the detected information (Alert/Normal) is sent into the fusion model asynchronously. Before any further operations, the preprocessing level synchronizes the information flow. To discern events, a unique EventID is assigned to each event before it is brought into various monitors. Our synchronization engine uses the EventID to combine alerts from different sources. The algorithm is presented as following:

```
if EventID is in the queue
then add the detecting info into the slot.
    if a slot is full
    then output data to next processing level.
    else continue.
else create a new slot.
```

Table 1 indicates one instance of the synchronization queue. In this scenario, event 2001 will soon be sent for further manipulation, so event 2002 will move into the first slot. If a new event comes in, it will be put after event 2003.

Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)
0-7695-2772-8/07 $20.00 © 2007 IEEE

Table 1. Synchronization queue

Slot	EventID	subject	Verb	Object
1	2001	abnormal	normal	abnormal
2	2002	normal	normal	waiting...
3	2003	waiting...	normal	waiting...

After synchronization, a knowledge based filter is implemented to reduce overhead information for the next processing level. The algorithm is shown below:

`if` $e \in C$
`then` output e to next level
`else` discard e

Where e is the event, and C expresses the critical event set. For instance, a critical event set can be defined in the form: "$*, write,$ `c:\Windows\*`", which indicates that any process that tries to write files in the folder "`c:\Windows`" will be considered as a critical event. The basic idea of the filter is to discard any event that is not in the critical event set so that it can minimize the load of next processing level.

3.2 Alert normalization level

The second processing level is alert normalization, which transforms different kinds of alert into one consistent set of scale. The STRIDE/DREAD threat model [7], developed by Michael Howard and David LeBlanc in Microsoft, is applied to achieve this.

STRIDE stands for:

- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation disclosure
- **D**enial of service
- **E**levation of privilege

And DREAD is an acronym for:

- **D**amage potential
- **R**eproducibility
- **E**xploitability
- **A**ffected Users
- **D**iscoverability

The first step of threat modeling is to create a model of applications, then categorize threats to each attack target node with STRIDE. After that, we can apply a DREAD category to evaluate the intrusion information.

In order to apply the STRIDE/DREAD model, a Categories Threat Level Function (CL Function) is implemented as shown in Fig. 3, where the input of the CL function is the alert and the output is the normalized score of the threat. Parameters of the function include the STRIDE/DREAD coefficients. Here we utilize a matrix with dimension $m \times n$ as the threat modeling coefficients. Normally, m is equal to 6 and n is equal to 5, according to the STRIDE/DREAD. Therefore up to 30 coefficients are involved. These coefficients can be set up in advance manually. A machine learning method such as the Backpropagation (BP) algorithm can be used to improve the CL function in the future.

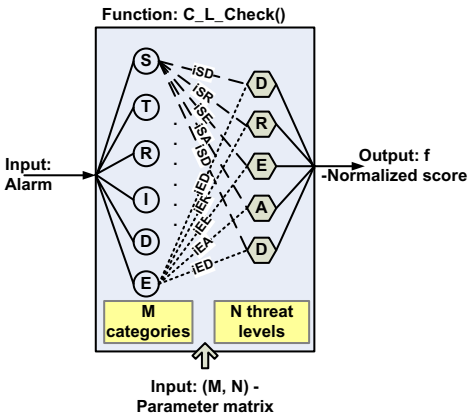

Figure 3. CL Function

3.3 Spacial Alert Fusion level

After the operations of the former processing levels, a threat chart can be obtained. An example is shown in Table 2, where each row indicates threat scores of different features within one event. We called it spacial information of the alert. In this processing level, the "spacial information" as we call it, will be fused into one unique threat information using a two-stage fusing algorithm. The first stage consolidates DREAD scores of different features within SVO models. For example, a common computer process, which is an instance of subject, has three features: user of the process, process identification, and process commonality. User of the process also has three features, which are user location (local or remote), user privilege, and user identification. The DREAD score of the process can be reached using Equation 1.

$$P_x = \sum_{1}^{6} F_x \tag{1}$$

Where P_x means one score within five DREAD categories of the process(subject), and F_x means the corresponding score of one feature. There are six features in our example, so the score of the process is the sum of the six features.

Table 2. Threat chart

Event	subject	Verb	Object
1	Feature 1 Feature 2 ... Feature n	Operation	Feature 1 Feature 2 ... Feature n
2	Feature 1 Feature 2 ... Feature n	Operation	Feature 1 Feature 2 ... Feature n
...	...	...	...

The second step is to combine the DREAD scores from the SVO monitors into one score of the event. Here a pre-set security index is utilized. For example, a ftp client is more dangerous than Windows NotePAD because it may transfer confidential files. Thus each process is assigned a danger level, which indicates the potential hazard. In the same way, files have their own importance level. So the final DREAD score will be calculated as in Equation 2.

$$E_x = S_x * I_s + O_x * I_o + I_v \quad (2)$$

where E_x is the final score within DREAD categories of the event, S_x is the corresponding score of the Subject, O_x is the DREAD score of the Object, I_s is the security index of Subject, I_o is the security index of the Object, and I_v is the security index of the Verb.

3.4 Temporal Alert Fusion level

Temporal Alert Fusion, the final processing level, is still under development. The basic idea of this processing level is fusing the alerts in a time sequence so that more intrusion information can be acquired. A scenario tree will be built to describe different intrusions. Figure 4 shows a scenario tree [5] [12]. In the figure, the nodes are called scenario nodes,

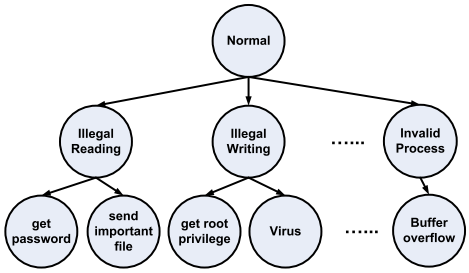

Figure 4. Scenario Tree

which indicate system states. Any path through the nodes is a sequence of intrusions. This processing level only sends output information when the states have changed. Using this tree, redundant alerts can be minimized. For instance, an intruder keeps trying to send a confidential document with high importance level to outside networks. Because

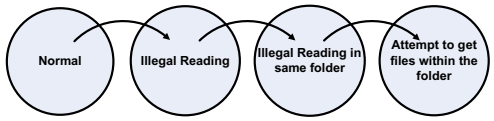

Figure 5. Scenario Path

of Temporal Alert Fusion, the next model of the IDS will receive one alert about this intrusion rather than thousands of duplicated warnings, as shown in Fig 5.

4 DEVS Model Framework

Discrete Event System Specification (DEVS) [14] is applied in this paper to describe the system above. DEVS is a state centered formalism approach which can model systems using an explicit timing specification. According to DEVS system definition, a basic model consists of the following information:

- The set of input ports that receive external events.
- The set of output ports that sent events.
- The set of state variables and parameters.
- The time advanced function which controls the timing of internal transitions.
- The internal transition function which specifies the state transition at the time given by time advanced function.
- The external transition function which specifies how the states changed when an input is received.
- The confluent transition function which decides the transition when conflict happens between internal and external transition functions.
- The output function which gives output before an internal transition.

Events determine values appearing on ports. When external events are received, the model must determine how to respond to them. Meanwhile, the model may change the state automatically. In the following, the DEVS model of our system is discussed in detail.

4.1 System Design

The DEVS system shown in Fig. 6 is designed according to the IDS system shown in Fig. 1. Borland JBuilder 2005 and DEVS-JAVA 3.0 were used for development. The monitor at the left side is applied to simulate the computer system sensor, which provides the host event to the IDS. Subject checking and object checking are represented as two coupling models of DEVS, which consist of atomic models. Verb checking has been integrated into the alert fusion model for simplicity. Alert fusion and verification models are placed at the right side as atomic models.

The Fig. 7 shows the details of the subject (process) and object (file) checking models. Different feature checking models are represented by DEVS atomic models. The basic model is an abstract model called CheckProcess, which is the parent class of all the feature checkers.

On the beginning, information sent from the lower processing levels will be analyzed by the knowledge based engine. If no data are available, the current fusion cycle is ignored, and nothing is generated. Otherwise, a formalism structure M is used to represent the system:

$$M =< X, S, Y, \delta_{ext}, \delta_{int}, \lambda, ta > \quad (3)$$

where X is the set of input data sources:

$$X = \{(p, e_v)|p \in InPorts, e_v \in Event\},$$

Where $InPorts$ is the set of input ports and $Event$ is the related system event acquired from the system monitor.

Y is the set of outputs:

$$Y = \{(p, e_v)|p \in OutPorts, e_v \in Event\},$$

Where $OutPorts$ is the set of output ports.

$Event = \{SystemInfo\} \times R_0^+$, which is the product of system information and DREAD score. So $e_v = \{(I, DREAD)|I \in \{SystemInfo\}, DREAD \in R_0^+\}$.

S is the set of states:

$$S = \{passive, normal, abnormal\} \times R_0^+,$$

$\delta_{ext}(phase, \sigma, e, X)$ is the external transition function. It will be triggered when some input is available:

```
if phase = passive and Check() = normal
then δext = (normal, Tcheck, ev)
else if phase = passive and Check()! = normal
       then δext = (abnormal, Tprocess + Tcheck, e'v)
```

Where $T_{process}$ is the processing time of STRIDE/DREAD evaluation; T_{check} is the feature checking time; σ is the time advance, which indicates that after σ time, the state of the system will transit automatically; $e'_v = (e_v.SystemInfo, e_v.DREAD + DREAD)$.

$\delta_{int}(phase, \sigma, S)$ is the internal transition function, which runs when the time elapsed is equal to σ:

$$\delta_{int} = (passive, \infty).$$

$\lambda(phase, \sigma, e_v)$ is the output function, which runs just before the internal transition function δ_{int},

$$\lambda = e_v.$$

Time advanced function ta is defined as:

$$ta(phase, \sigma, (u, v)) = \sigma.$$

The only difference between the various feature checkers is the function $Check()$ within the external transition function δ_{ext}.

4.2 Fusion Model Design

A multi-level alert fusion model is more complex than the checking models. To minimize programming, the first two processing levels, i.e., the Source Preprocessing and Alert Normalization, are put into atomic feature checkers. The function $Check()$ in the external transition function not only provides various feature checking mechanisms, but also does a critical event set check and calculates the related STRIDE/DREAD score. In this paper, the critical set and STRIDE/DREAD matrix are predefined.

Within the alert fusion DEVS model, Spacial Alert Fusion is achieved by equations 1 and 2. Verb checking is also embedded into this model, so that different operations from one process will relate to different security indexes of the subject. The exact index value is set up in advance manually within our simulation.

Various system states and transition links are also defined for temporal alert fusion. Thus the fusion model can eliminate duplicate alerts efficiently.

DEVS model of fusion model is indicated in the following:

$$F =< X, S, Y, \delta_{ext}, \delta_{int}, \lambda, ta > \quad (4)$$

The input set X and output set Y are the same as the ones in feature checking models. In the alert fusion model, there are two input ports, $Inports = \{"Subject", "Object"\}$; and two output ports, $Outports = \{"normal", "alert"\}$.

S is the set of states:

$$S = \{passive, subject, object, normal, alert\} \times R_0^+,$$

$\delta_{ext}(phase, \sigma, e, X)$ is the external transition function, which runs when some input is available:

```
if phase = passive and X = ("Subject", ev)
then δext = (subject, Tprocess, ev)
```

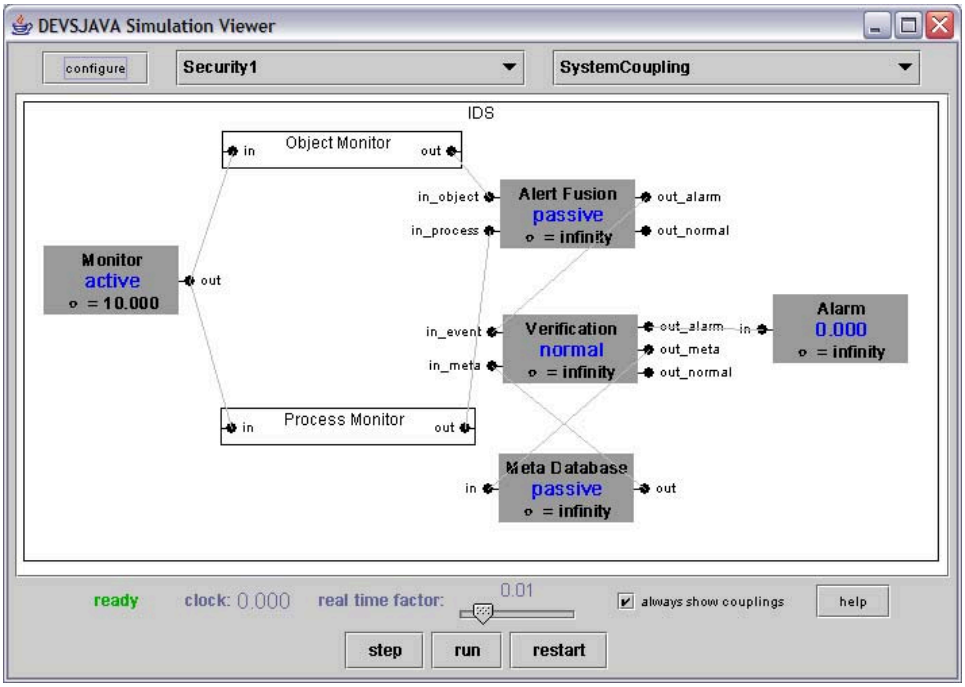

Figure 6. DEVS framework of the IDS

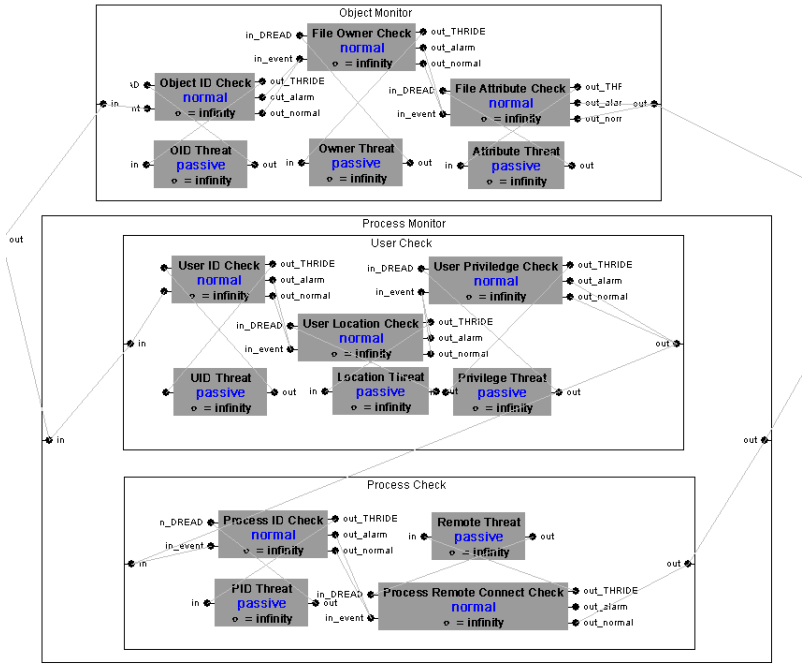

Figure 7. Detail of the coupling models

Proceedings of the 14th Annual IEEE International Conference and
Workshops on the Engineering of Computer-Based Systems (ECBS'07)
0-7695-2772-8/07 $20.00 © 2007 IEEE

if $phase = passive$ and $X = (\text{"}Object\text{"}, e_v)$
then $\delta_{ext} = (object, T_{process}, e_v)$
if $phase = subject$ and $X = (\text{"}Object\text{"}, e_v)$
then if $Fuse() = false$
then $\delta_{ext} = (normal, T_{process}, e_v)$
else $\delta_{ext} = (alert, T_{process}, e'_v)$
if $phase = object$ and $X = (\text{"}Sbject\text{"}, e_v)$
then if $Fuse() = false$
then $\delta_{ext} = (normal, T_{process}, e_v)$
else $\delta_{ext} = (alert, T_{process}, e'_v)$

$T_{process}$ is the processing time of calculation; σ is the time advance; e'_v is the fused DREAD score after spacial and temporal alert fusion process. The fusion function $Fuse()$ can achieve the calculation.

$\lambda(phase, \sigma, e_v)$ is the output function.

if $phase = normal$
then $\lambda = (\text{"}normal\text{"}, e_v)$
if $phase = alert$
then $\lambda = (\text{"}alert\text{"}, e_v)$

Time advanced function ta is shown as below:

$$ta(phase, \sigma, (u, v)) = \sigma$$

4.3 Simulation

A DEVS-JAVA simulation is described in this section. At first, imitation attack data are utilized to demonstrate the feasibility of the proposed methods. The scenario is shown as below.

First 100 system events are generated. Within the events, 3 intrusions are added. Event number 15 is an intrusion where a legal user tries to access a file which he has no privilege to open. Events number 35 to 45 indicate a scenario where a remote access user tries to access a serious important file that he has no right to access. Event 65 shows an unauthorized user trying to access a file.

The simulation result indicates that 13 alerts have been intercepted, the total number we inserted. The system output is printed as following:

```
Alert! Event ID: 15
Object ID Check
User Location Check
File Attribute Check

Alert! Event ID: 35
User Privilege Check
File Attribute Check
Process Remote Connect Check

Alert! Event ID: 65
User Location Check
User Privilege Check
Process ID Check
```

The result shows the duplicated alerts from event 35 to event 45 have been eliminated due to the temporal alert fusion. The alerts from different features have been integrated together due to spacial alert fusion. Therefore, only three alerts were displayed to the administrator while no useful information was discarded.

5 Conclusion

In this paper, an alert fusion model of the Subject-Verb-Object model based computer host intrusion detection system has been proposed. A multi-level alert fusion model is used to minimize the duplicate alert information from spacial and temporal aspects without losing information. DEVS formalism implementation and DEVS-JAVA simulation is applied to validate the model. Simulation results indicate that the alert fusion system works well.

In the current research, the feature checking threshold and STRIDE/DREAD matrix are predefined. In future research, a machine learning mechanism can be applied to help the system adapt complex intrusions.

References

[1] B. Mukherjee, L. Heberlein and K. Levitt, "Network Intrusion Detection," *IEEE Network*, 1994.

[2] K. Ilgun, "USTAT: A real-time intrusion detection system for UNIX," *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, 1993.

[3] D. L. Hall and J. Llinas, "An Introduction to Multisensor Data Fusion," *Procedings of the IEEE*, Vol. 85, No. 1, pp. 6-23, 1997.

[4] S. Eyles and R. Westgarth, "The Specification of a Submarine Data Fusion System,"*IEEE Colloquium on Principles and Applications of Data Fusion*, pp. 1-8, Fed 4 1991.

[5] S. Mathew, C. Shah and S. Upadhyaya, "An Alert Fusion Framework for Situation Awareness of Coordinated Multistage Attacks," *Processing of the Third IEEE International Workshop on Information Assurance*, 2005.

[6] A. Chan, W. Ng and et al., "Comparision of Different Fusion Approaches For Network Intrusion Detection Using Ensemble of RBFNN," *Proceeding of the Fourth International Conference on Machines Learning and Cybernetics*, pp. 3846-3851, 2005.

[7] M. Howard and D. LeBlanc, "Writing Secure Code," *Microsoft Press*, 2nd edition (December 4, 2002)

[8] N. Ye, J. Giordano and et al., "Information Techniques for Network Intrusion Detection," *IEEE Information Technology Conference, 1998*, pp 117-120, 1998.

[9] A. Siraj, R.B. Vaughn and S.M. Bridges, "Intrusion Sensor Data Fusion in an Intelligent Intrusion Detection System Architecture," *Proceddings of the 37th Hawaii International Conference on System Sciences*, pp 1-10, 2004.

[10] M. Shankar, N. Rao and S. Batsell, "Fusion Intrusion Data For Detection and Containment," *IEEE Military Communications Conference, 2003*, pp 741-746, 2003.

[11] Y. Wang, H, Yang and et al., "Distributed Intrusion Detection System Based on Data Fusion Method," *Proceedings of the 5th world Congres on Intelligent Control and Automation*, pp 4331-4334, 2004.

[12] A. Siraj and R.B. Vaughn, "Multi-Level Alert Clustering for Intrusion Detection Sensor Data," *2005 Annual Meeting of the North American Fuzzy Information Processing Society*, pp. 748-753, 2005.

[13] N. Ye and M. Xu, "Information Fusion for Intrusion Detection," *Proceedings of the Third International Conference on Information Fusion*, pp 17-20, 2000.

[14] B.P. Zeigler and H.S. Sarjoughian, "Introduction to DEVS Modeling and Simulation with JAVA: Developing Component-Based Simulation Models," *Draft Version 3*, 2005

Proceedings of the 14th Annual IEEE International Conference and
Workshops on the Engineering of Computer-Based Systems (ECBS'07)
0-7695-2772-8/07 $20.00 © 2007 IEEE