

## DECISION MAKING FOR NETWORK HEALTH ASSESSMENT IN AN INTELLIGENT INTRUSION DETECTION SYSTEM ARCHITECTURE

AMBAREEN SIRAJ<sup>†</sup>, RAYFORD B. VAUGHN<sup>\*:‡</sup> and SUSAN M. BRIDGES<sup>§</sup>

*Department of Computer Science & Engineering  
Butler Hall, Room 300, Corner of Barr and Perry  
Mississippi State University  
Mississippi State, MS 39762, USA*

<sup>†</sup>*ambareen@cse.msstate.edu*

<sup>‡</sup>*vaughn@cse.msstate.edu*

<sup>§</sup>*bridges@cse.msstate.edu*

*http://www.cse.msstate.edu/~security*

This paper describes the use of artificial intelligence techniques in the creation of a network-based decision engine for decision support in an Intelligent Intrusion Detection System (IIDS). In order to assess overall network health, the decision engine fuses outputs from different intrusion detection sensors serving as “experts” and then analyzes the integrated information to present an overall security view of the system for the security administrator. This paper reports on the workings of a decision engine that has been successfully embedded into the IIDS architecture being built at the Center for Computer Security Research, Mississippi State University. The decision engine uses Fuzzy Cognitive Maps (FCM)s and fuzzy rule-bases for causal knowledge acquisition and to support the causal knowledge reasoning process.

*Keywords:* Fuzzy Cognitive Map; intrusion detection; alert fusion; decision engine.

### 1. Introduction

Over the past few years, demands for “more secured” systems have increased heavily. For this reason, most modern intrusion detection systems employ multiple intrusion sensors to maximize their trustworthiness. The multiple intrusion sensors employ different strategies based on the model they use, the data source they monitor and the techniques they employ. With different types of sensors in place it is often difficult to obtain an overall picture of the security status of a complex system. This is especially true of systems, for example, that may be geographically distributed over a wide area or perhaps those that employ newer architectures like high performance clusters. It is, therefore, extremely important to correlate/merge/fuse

\*Corresponding author.

the different outputs of these sensors in an effective and intelligent manner in order to provide the security administrator with an “overall security view” describing the general health of the system by revealing its illnesses, if any. This overall security view of the multi-sensor intrusion detection system can serve as an aid to appraise the trustworthiness in the system.

This paper presents our research effort in that direction. We describe the use of artificial intelligence techniques in the creation of a network-based Decision Engine for decision support in an Intelligent Intrusion Detection System (IIDS) architecture involving multiple sensors. The Decision Engine fuses information from the different intrusion detection sensors using a causal knowledge-based inference technique. Fuzzy Cognitive Maps (FCMs) and fuzzy rule-bases are used for the causal knowledge acquisition and to support the causal knowledge reasoning process.

### **1.1. *Introduction to the IIDS***

The “health” of a computer network needs to be assessed and protected in much the same manner as the health of a person. When an individual becomes ill, for example, it is generally noticed in stages, as the illness occurs over time. A system under attack experiences a similar build up that can be characterized as increasing suspicion. The task of an intrusion detection system is to protect a computer system by detecting and diagnosing attempted breaches of integrity of the system. A robust intrusion detection system for a computer network will necessarily use multiple sensors, each providing different types of information about some aspect of the monitored system. In addition, the sensor data will often be analyzed using multiple techniques.

The Intelligent Intrusion Detection System (IIDS) is an active intrusion detection research effort within the Center for Computer Security Research (CCSR) at Mississippi State University. IIDS is characterized by the following unique features:<sup>2,25</sup>

- operation in near real time to detect intrusions early;
- adaptive in a computing system to changing network/user behavior;
- distributed to capture intrusion at various entry points in the network;
- network-based modular architecture to monitor activities across the whole network;
- incorporation of both anomaly and misuse detection (i.e. misuse detection looking for known patterns of attack and anomaly detection looking for deviations from “normal” patterns of behavior);<sup>34</sup>
- intelligent, by employing artificial intelligence techniques other than conventional methods;
- efficient, to minimize false negative rates and false positive rates;
- equipped with a graphical presentation tool to display security status; and
- extending in the high performance cluster environment.

Today, artificial intelligence has become an indispensable tool in designing effective intrusion detection systems. IIDS uses artificial intelligence techniques in the following ways:<sup>3</sup>

- In anomaly detection by using data mining algorithms that
  - are integrated with fuzzy logic for the generation of more abstract and flexible patterns;
  - use genetic algorithms to optimize membership functions for performance.
- In anomaly detection by using sequence matching, Hidden Markov models, and Neural networks that profile host-based process behavior.<sup>11,23</sup>
- In a Decision Engine by causal knowledge inferencing with Fuzzy Cognitive Maps.<sup>31</sup>

### 1.2. Introduction to the Decision Engine

In the IIDS architecture (Fig. 1), multiple sensors, both anomaly and misuse detection sensors serving as “experts”, monitor activities on individual workstations,

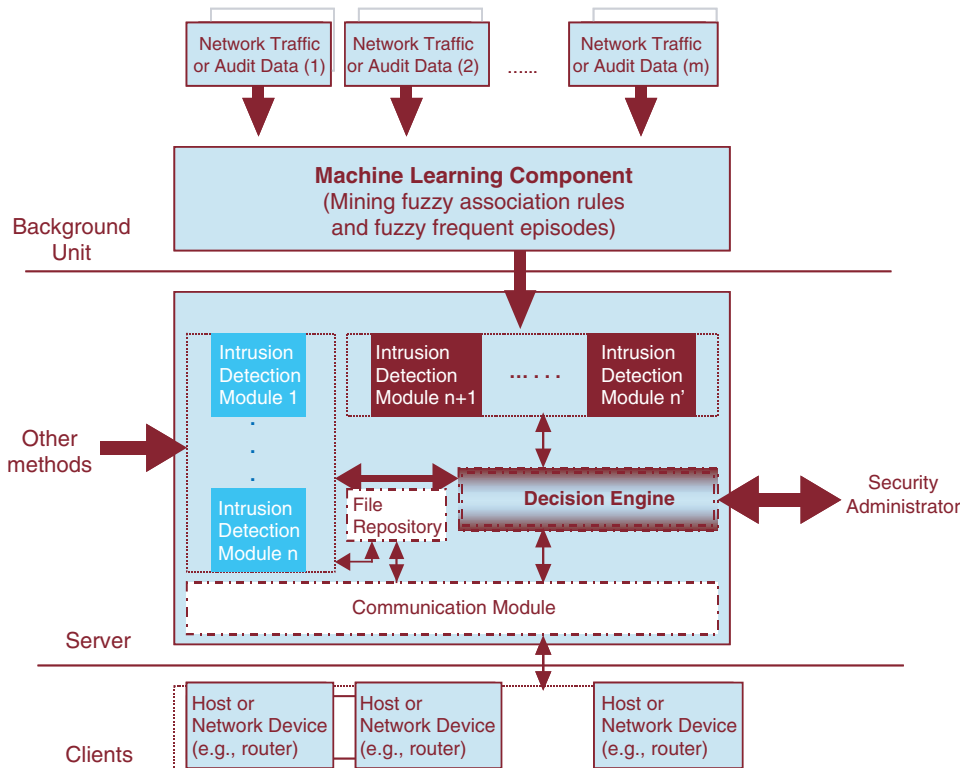


Fig. 1. IIDS architecture.

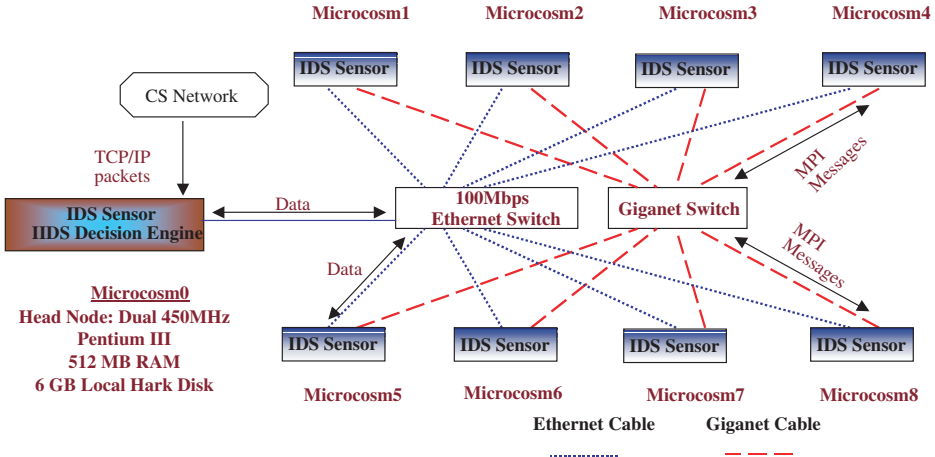


Fig. 2. Decision Engine in the cluster environment.<sup>22</sup>

activities of users, and network traffic by detecting intrusion from within the network traffic and at individual client levels. These components use different methods (i.e. various machine learning and expert systems techniques) to detect intrusion information and then, pass anomalous/malicious system behavior indications to the Decision Engine.

Currently, the IIDS architecture runs in a high-speed cluster environment. In the high performance cluster environment, the Decision Engine resides in the head node (the control point for access in cluster) and monitors intrusion activities across the cluster of eight internal nodes interconnected through both Ethernet and Gigabit switches (Fig. 2).

### 1.3. Introduction to Fuzzy Cognitive Maps

Since Professor Lotfi Zadeh introduced fuzzy logic in the mid-1960s, it has been extensively used in a wide range of application domains and problem solving. Only in recent years has it generated interest in the area of intrusion detection in network security. Pioneers in this direction are shown in Refs. 2, 9, 10, 14, 26.

Fuzzy Cognitive Maps (FCMs) originated from the combination and synergism of fuzzy logic and neural networks, combining the robust properties of both.<sup>33</sup> It is an efficient soft computing tool that supports adaptive behavior, based on empirical prior knowledge, and provides a graphical representation of that knowledge that can be used for the explanation of reasoning.<sup>33</sup> The idea of FCMs stemmed from cognitive maps, which were proposed by political scientist Robert Axelrod.<sup>1</sup> Professor Bart Kosko of the University of Southern California extended the cognitive maps (signed digraphs of nodes and edges), to fuzzy cognitive maps by considering the fuzzy aspects of causality and by accommodating the knowledge-base building property.<sup>18</sup>

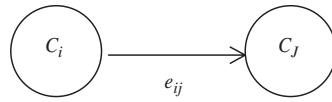


Fig. 3. Two FCM concepts and a connecting edge representing a causal link.

FCMs model the world as concepts and causal relations between concepts in a structured collection.<sup>18–20</sup> Concepts (nodes) in an FCM are events that originate in the system and whose values change over time. The causality links between concepts are represented by directed edges that measure how much one concept impacts the other(s).

As shown in Fig. 3, the edge  $e_{ij}$  can be used to define rules or causal flow between the concept nodes  $C_i$  and  $C_j$ .<sup>21</sup> An edge value of  $e_{ij} = 0$  indicates that there is no cause-effect relation between the concepts  $C_i$  and  $C_j$ . A value  $e_{ij} > 0$  denotes positive causality, i.e. whenever concept  $C_i$  increases,  $C_j$  increases by the degree  $e_{ij}$ . Conversely,  $e_{ij} < 0$  denotes negative causality, i.e. whenever concept  $C_i$  increases, there is a decrease in  $C_j$  by the degree  $e_{ij}$ . Therefore, a higher absolute value for  $e_{ij}$  indicates greater affect of the cause and a lower value indicates lesser affect of the cause.

Researchers have used FCMs for many tasks in several different domains.<sup>8,27,29,32,33</sup> As reported in Ref. 31, we have been utilizing FCMs for fusing intrusion alert information in a multi-sensor intrusion detection environment. Recently, in Ref. 37, the Fuzzy Intrusion Recognition Engine (FIRE), a research effort at Iowa State University, illustrated the use of FCMs in detecting complex attack scenarios for a network-based anomaly detection system. In this paper, we report on our use of FCMs in a Decision Engine that effectively fuses intrusion alert information to make decision about overall network health when multiple types of intrusion sensors (both misuse and anomaly) are at work.

## 2. The Decision Engine

### 2.1. The Decision Engine framework

In the IIDS architecture, the Decision Engine analyzes reported intrusion information from multiple possibly disparate sources. In general, responsibilities of the Decision Engine include the following tasks that are carried out by the different core components of the Decision Engine (Fig. 4):

- (1) **Host/User Misuse Alert Inference Module** generates the misuse alert status for each individual host and user in the network by analyzing information reported by multiple misuse detection components.
- (2) **Network Anomaly Alert Inference Module** generates the anomaly alert status for the network by analyzing information reported by multiple anomaly detection components.

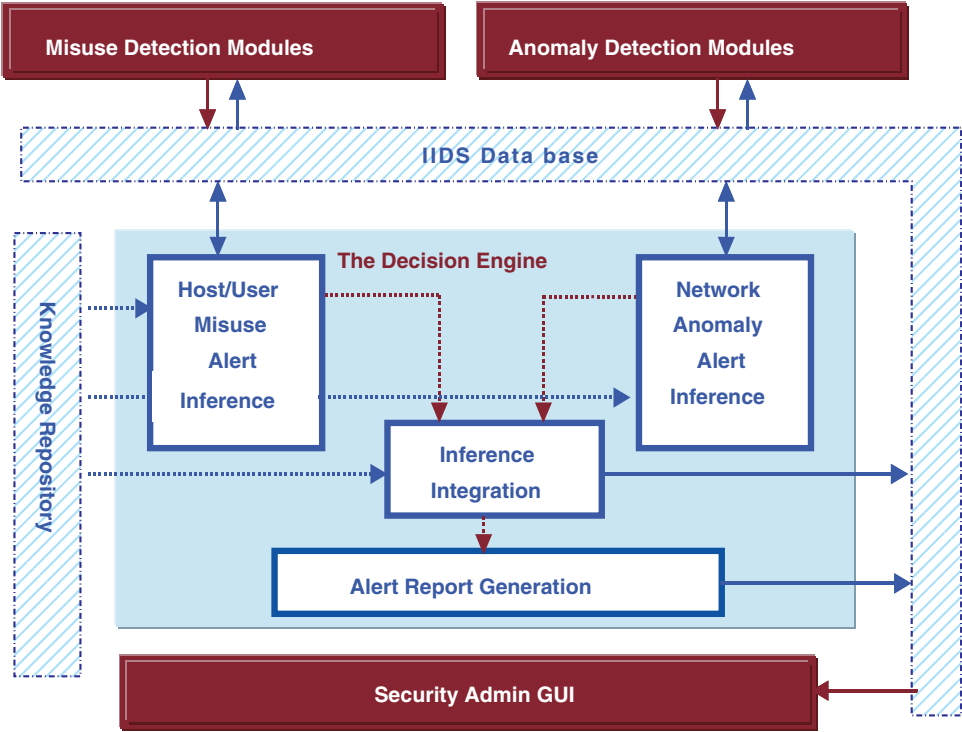


Fig. 4. Decision Engine framework.

- (3) *Inference Integration Module* combines misuse alert information for all hosts and all users in the network to generate the overall host and user misuse alert status.
- (4) *Alert Report Generation Module* prepares and transmits reports on the overall network alert status for the Graphical User Interface (GUI) monitored by the security administrator.

The scratchpad of the Decision Engine is essentially a central data repository, known as the IIDS database. The IIDS database is a collection of relations that holds network information and serves as the communication mechanism between the Decision Engine and different detection components/GUI. The relations in the database are mainly partitioned into two groups depending on the kind of information they contain and are as follows:

- (1) *Topology Database (TDB)*: Stores the network configuration and relationships between different hosts, switches, and network cards.
- (2) *Intrusion Database (IDB)*: Stores all information related to monitoring for network intrusion. It includes data reported by the various sensors and the inferences generated by the Decision Engine. The IIDS database is further described in Sec. 2.4.

The Decision Engine also maintains a knowledge repository that provides the Decision Engine with necessary knowledge for making appropriate fuzzy inferences. This knowledge is extracted from the security administrator as per the security policy of the network system.

## ***2.2. Workings of the Decision Engine***

The Decision Engine assesses the network health for the IIDS with the objective of providing the security administrator an overall security view in terms of possible intrusive behaviors as determined primarily by the following two factors:

- (1) Misuse alert level of all the hosts and users in the network.
- (2) Anomaly alert level of network traffic.

The Decision Engine makes an assessment about the former two factors, i.e. the overall hosts/users misuse alert level, by analyzing and investigating the results generated by the misuse detection components which look for signatures of known attacks. The analysis involves identifying suspicious activities across the network for each of the hosts and users of the network and then generating an alert status for each of them, as described in Sec. 2.3.1. To make decisions about the anomaly alert level of the network, the Decision Engine analyses data from multiple anomaly detection sensors, as described in Sec. 2.3.2.

## ***2.3. The Decision Engine model***

The model of the Decision Engine using FCMs for decision support in both misuse and anomaly alert inference is described below.

### *2.3.1. Misuse alert inference*

For misuse detection, the IIDS uses rule-based detection mechanisms that work on each of the hosts of the network.<sup>17</sup> Outputs from the misuse detection sensors working as sentries, may be crisp or fuzzy. For some attacks, the attack signature is either present or absent and the output of the module is binary (e.g. a SYN\_Flood attack is or is not occurring). For other types of attacks like the number of failed logins, the output of the misuse detection sensor is a fuzzy measure of the degree of suspicion. The Decision Engine must assess the results of the multiple misuse detection sensors in order to compute the alert status for each host and for each user account. This decision making process undergoes two levels of analysis as in Fig. 5:

- (1) At the initial level, Suspicious Events (SE) are identified for individual hosts and users across the network by combining reports from the Misuse Detection Modules (MDMs). These events are marked as “suspicious” because these are out of the ordinary and call for security administrator’s attention.
- (2) At the second level, the misuse alert level for each host and user (H/U) is computed by combining the results of the suspicious activities.

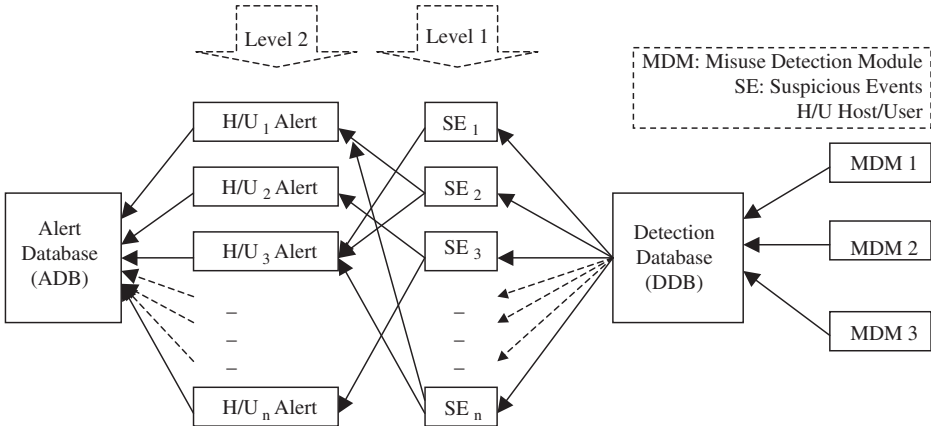


Fig. 5. Individual misuse alert generation for hosts/users.

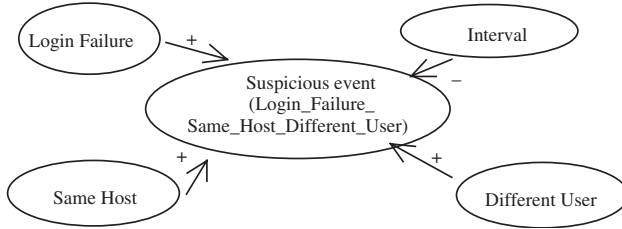


Fig. 6. Example FCM for capturing suspicious misuse scenarios.

For misuse alert inference, we have an FCM model implemented that incorporates several FCM concepts for each kind of misuse detection (e.g., failed login attempts, finger bombs, SYN\_Flood attacks). Suspicious Events can be of different types that affect the host and user alert level in specific ways. Each of these Suspicious Events can be triggered by analyzing the results of misuse detection sensors and are activated with the help of a fuzzy rule-base where fuzzy rules are used to map multiple inputs to outputs. We instrument the causes or concepts as fuzzy variables represented by multiple fuzzy sets and the causal links or edges as fuzzy relations represented by one or more fuzzy rules.

The following simple example illustrates how an FCM can be used to capture a Suspicious Event pattern for failed login attempts.<sup>31</sup> Consider the following scenario: *an intruder is trying to access the network by breaking into a particular host. The intruder tries to login with several users' passwords and fails.* One can identify such an attack scenario by observing the number of login failures, the user and the host involved, and the date and time of attack. Figure 6 shows an FCM that captures this scenario. We can see how the concepts Login\_Failure\_Same\_Host\_Different\_User positively impact the Suspicious Event and the concept Interval between failures negatively impacts it. This kind of attack would generate an alert for both the host and the user(s) concerned.

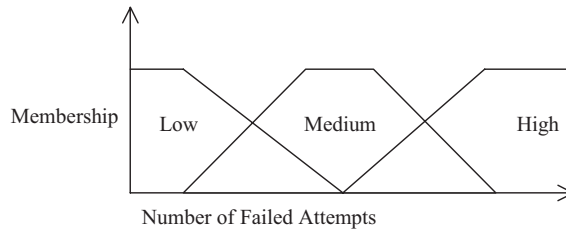


Fig. 7. A complete term set for the fuzzy variable login failure.

As stated earlier, in causal knowledge inferencing with fuzzy cognitive modeling, the concepts in FCM can be represented by fuzzy variables that define the constituting fuzzy sets. The fuzzy variables define the language for the concepts and the fuzzy sets describe the concept.<sup>24</sup> For example, the Decision Engine implements the concept *Login\_Failure* as a fuzzy variable, where the complete term set is composed of three fuzzy sets — *Low*, *Medium* and *High* (Fig. 7).

The activation of the output concept *Suspicious\_Event\_Login\_Failure\_Same\_Host\_Different\_User* by the input concepts can be implemented using a fuzzy rule-base where the fuzzy rules map the causes (input concepts) to the effects (output concepts). Multiple fuzzy rules may be used to correspond to the knowledge described in an individual FCM. For example, a fuzzy rule such as the following, can be used to implement the FCM in Fig. 6:

*If number of login failures is high and interval is short and this happened for same host and different users then, Login\_Failure\_Same\_Host\_Different\_User is activated highly.*

For such fuzzy reasoning, our Decision Engine uses Mamdani Fuzzy Inference. Mamdani Fuzzy Inference is characterized by ANDing of premises, ORing of consequents, implication, aggregation of multiple conclusion and defuzzification.<sup>28</sup> Therefore, when there are multiple antecedents in rules, the minimum of the maximum membership values of the intersections of the crisp inputs and the antecedent fuzzy set pairs is considered.<sup>24</sup> This strategy of selecting minimum common truth has the advantage that no condition can be stronger than its weakest link<sup>6</sup> and therefore, contributes to fewer false alarms in this case. With multiple rules fired, the consequent is truncated to the maximum of the fuzzy values (this ensures that the truth of the truest rules will dominate<sup>6</sup>). The aggregated output is finally decoded to extract a crisp value that “best represents the fuzzy set”<sup>16</sup> by “balancing the consequent fuzzy reason”.<sup>6</sup> We have found that using Mamdani Fuzzy Inference was very efficient and inexpensive computationally.<sup>24</sup>

Multiple FCMs capture different types of intrusive behavior as suspicious events in the Decision Engine.<sup>31</sup> At the next level of misuse inference the Decision Engine

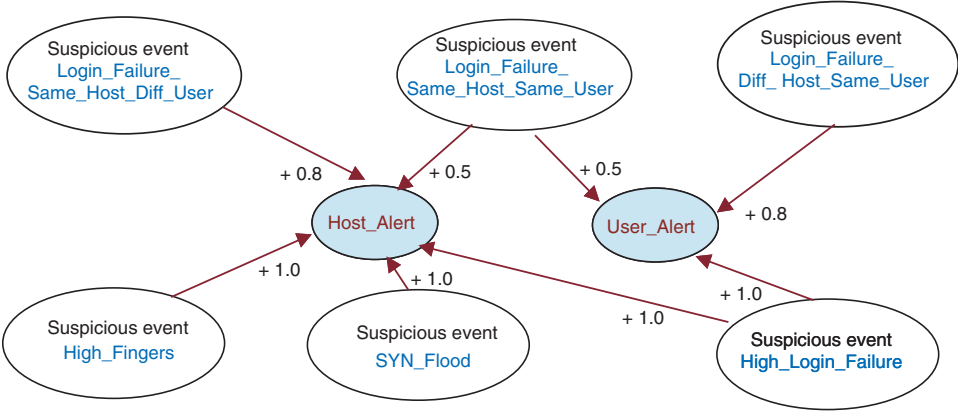


Fig. 8. Combining evidence of multiple suspicious events.

computes the level of alert for individual hosts and users by combining suspicious activities affecting each of them. All activated suspicious events impact the host and/or user alert levels in different ways because the degrees of impact vary depending on the nature of the Suspicious Event itself. For example, a suspicious event due to *Login\_Failure\_Same\_Host\_Same\_User* should not impact the user alert level as much as a Suspicious Event due to *Login\_Failure\_High\_Finger*. The alert value for a particular host or user reflects the degree to which it is operative in the system at a given time and is calculated as a function of all the activated suspicious events for that particular host or user at that time. Figure 8 shows an example of a subset of FCMs that our Decision Engine employs for evidence combination in the IIDS architecture.

To fuse the impacts of the different suspicious events activated for each host and user, we utilize the resemblance between FCMs and neural networks.<sup>4</sup> In the neural network approach, the concepts of the FCM are represented by neurons and the edges are represented by the weights of the connecting neurons. The suspicious events in the FCMs, treated as neurons, trigger activation of the alert levels with different weights depicting causal relations between them. An adjacency matrix is used to list these cause and effect relationships between the nodes. At any time, the alert level for any particular host/user collectively represents the effects of all the suspicious events activated for that host/user at that time. The runtime operation is observed by determining the next value of each concept from the current concept and connecting edge values.<sup>4</sup> This can be represented as the following for each concept  $C_i$  at  $t_{n+1}$  time where  $e_{ki}$  is the edge strength between  $C_i$  and each of its contributing concepts  $C_k$ ,<sup>21</sup>

$$C_i(t_{n+1}) = S \left[ \sum_{k=1}^n e_{ki}(t_n)C_k(t_n) \right].$$

Here  $S$  is the normalization function to force the aggregated alert value to be monotonically mapped into a normalized range, i.e. between 0 to 1. The Decision Engine uses weighted average normalization in case of multiple contributing evidences, as shown in the following function,

$$S(x) = x / \sum_{k=1}^n e_{ki}.$$

Generally, in FCMs, the edge values are determined from expert knowledge and experience. These are dependent on the expert's experience and engineering judgment.<sup>32</sup> Once the values are initially set, their performance can be observed over time and their values can be tuned for an optimal performance by the security administrator based on empirical performance in terms of the alerts generated. We have found that FCMs offer a highly flexible structure in this regard. A variety of both manual and automated techniques can potentially be used to fine-tune these parameters.

### 2.3.2. Anomaly alert inference

In our IIDS architecture, multiple anomaly detection sensors analyze network audit data. The current sensors we have constructed employ fuzzy data mining techniques with different but supporting approaches such as fuzzy association rules and fuzzy frequency episodes.<sup>2</sup> The outputs of these modules are fuzzy measures of the "normality" of the audited data. One of the main tasks of the Decision Engine is to analyze the combined effect of the outputs of these anomaly detection sensors in order to determine the network anomaly alert level. For this purpose, FCMs are used to relate the causal relationship between the fuzzy measures that influence the anomaly alert level of the network.<sup>31</sup>

The anomaly detection sensors measure the dissimilarities between normal and abnormal data and report anomalous behavior in terms of an alarm percentage and a similarity measurement. In order to assess the combined effect of the outputs of the anomaly detection sensors on the network's anomaly alert level, the Decision Engine uses FCMs such as the one in Fig. 9.

There are two causal relations in this FCM. One denotes that as the similarity measure increases, the anomaly alert level decreases with 0.8 causality. Intuitively this can mean that when the similarity measure is very high, the anomaly alert would be very low and, conversely, when the similarity measure is low, the anomaly

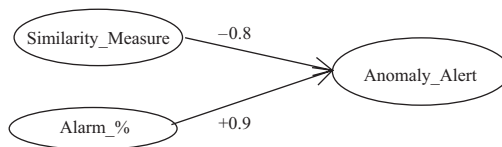


Fig. 9. Example FCM for fusing anomaly detection results.

alert would be high. Another FCM relation shows that alarm percentage increases the anomaly alert with 0.9 causality. Therefore, if alarm percentage is high, then the anomaly alert is high, and likewise, if the alarm percentage is very low, then anomaly alert is very low. FCMs can be used to replace many rules in a conventional rule-based system.

### 2.3.3. *Inference integration*

Inference integration involves integrating the affects of the anomaly and misuse inferences to determine the overall network alert situation considering combinations of seemingly harmless events. The overall network health is assessed by determining the overall misuse alert level for hosts and users and anomaly alert level for the network. Typically, the Decision Engine considers the maximum of the current alert values as the overall misuse host/user alert level. For atypical cases, for example, situations when there is a large number of Low level alerts for hosts/users signifying the possibility of a slow attack in progress, the Decision Engine forces the overall misuse host/user alert level to be High ensuring the security administrator's attention. The Decision Engine recognizes these kinds of situations by categorizing the received alert values as Low, Medium and High based on the alerts' strength and then observing the distribution of different host/user alerts by computing standard deviations. The Decision Engine follows the same principle for computing the overall anomaly alert level for the network.

### 2.3.4. *Report generation*

The Decision Engine reports its deduction to the security administrator by sending and recording the results in the IIDS Alert Database. The security administrator monitors the network health through a Graphical User Interface<sup>13</sup> that picks up and presents alert data in graphical form from the Alert Database.

## 2.4. *Design and implementation of the Intrusion Database (IDB)*

The implementation of the Decision Engine involved the design and implementation of the Intrusion Database in the IIDS Database that directly interacts with our Decision Engine, and the design and implementation of the core Decision Engine module. The relations in the IIDS database are implemented with an Oracle database management system. The rationale for such use includes easy but controlled access, minimization of redundancy, consistency of data, data integrity, organized presentation and sharing of data. In addition, the DBMS provides an extra level of security.<sup>30</sup> Depending on the nature of the intrusion information contained and their interaction with the Decision Engine, the IDB is partitioned into two sub-groups:

- (1) *Detection Database (DDB)*: The DDB consists of the relations that store output generated by the various misuse and anomaly detection sensors employed in

the IIDS architecture. Currently, the IIDS has three types of misuse detection sensors implemented and embedded into the system (for detecting failed login attempts, finger bombs, and SYN\_Flood attacks<sup>17</sup>) and one anomaly detection sensor designed for the system.<sup>25</sup>

- (2) *Alert Database (ADB)*: The ADB stores results of the Decision Engine's network health assessment. It consists of relations that store particulars of different types of suspicious events and host/user/network alert information.

## 2.5. Design and implementation of the Decision Engine

Knowledge in any system that uses fuzzy logic, is distributed in both fuzzy sets and fuzzy rules.<sup>6</sup> For causal knowledge inferencing with fuzzy cognitive modeling, the Decision Engine treats the concepts Login\_Failure, Fingers, and Suspicious Events as fuzzy variables. The complete term set for each of the three concepts essentially consists of three fuzzy sets that are Low, Medium and High. For Fingers and Suspicious Events, the fuzzy set Low is implemented with a  $Z$  membership function, Medium is implemented with Triangular membership function and High is implemented with  $S$  membership function. For Login Failures, Low is implemented with a  $Z$  membership function, Medium is implemented with a Trapezoidal membership function and High is implemented with an  $S$  membership function as shown in Fig. 7 in Sec. 2.3.1.

We carry out the fuzzy reasoning process implemented with the Mamdani principle in four basic steps:<sup>38</sup>

- (1) *Fuzzy\_Match*: This function takes a crisp input (e.g. the crisp number of fingers) and then matches it with the relevant fuzzy sets of the fuzzy variable (e.g. fuzzy sets Low, Medium and High of fuzzy variable Fingers) in order to find the degree of membership of the crisp input in each of the fuzzy sets.
- (2) *Fuzzy\_Inference*: This function uses the matched degree of membership for the fuzzy sets found in the "fuzzy\_match" function as antecedents to infer the degree of rule matching for all relevant rules in the knowledge base using the clipping method. It then generates the consequents of the matched rules as output fuzzy sets (e.g. in this case fuzzy sets Low, Medium and High of fuzzy variable Suspicious Events). For fuzzy inference, the clipping method was used to suppress the membership function of the consequents of the rules fired.
- (3) *Fuzzy\_Rule\_Combination*: This function takes the individual fuzzy outputs contributed from each of the rules matched in the "fuzzy\_inference" function and then aggregates them to generate the final fuzzy output by applying the max fuzzy disjunction operation.
- (4) *Defuzzification*: This function uses centroid defuzzification to convert the final fuzzy output from the "fuzzy\_rule\_combination" function to generate one crisp output (e.g. in this case, a crisp value for suspicious event). Our Decision Engine employed the centroid defuzzification method because this method ensured contribution of distributed data.<sup>6</sup>

## 2.6. Experimentation and results

We conducted a series of experiments to verify the Decision Engine's functionality. Preliminary experimentation was conducted with fabricated data in the misuse detection tables to observe whether or not the Decision Engine worked to generate appropriate inferences. Later in our research, as the misuse detection sensors became operational,<sup>17</sup> the operations of the Decision Engine were tested successively with individual misuse detection sensors. Although designed and implemented experimentally, the operational prototype of the anomaly detection sensors is not currently in place in the IIDS, so for experimentation purposes, our Decision Engine analyzed artificial data from the anomaly detection module database. Ultimately, the Decision Engine was tested with all the misuse detection sensors running concurrently in the IIDS architecture with artificial anomaly detection data in place.

The experimental setup consisted of a network of SUN workstations running the Solaris 5.6 operating system and was situated in the Software Engineering laboratory of the Department of Computer Science & Engineering (CSE), Mississippi State University. The final experiment included running a single thread of execution from the set of three misuse detection sensors (telnet misuse sentry, finger misuse sentry and SYN sentry) to the security admin GUI via the Decision Engine.

### 2.6.1. Testing

For demonstration of the Decision Engine's alert generation hosts (HostID: 1037, 1066, 1047) and users (UserID: 2046, 2053, 2064, 2065, 2067, 2093) of the CSE network were targeted. Each of the hosts had the three types of IDS sentries running on them reporting intrusion data to MDM1 (relation to capture failed login attempts), MDM2 (relation to capture finger attacks), and MDM3 (relation to capture SYN flood attacks) in the Detection database. The desired behavior was the following:

- (1) The Decision Engine checks for intrusion data in IIDS database periodically.
- (2) Once intrusion information arrives in the detection tables, the Decision Engine analyzes the data and generates alerts accordingly.
- (3) Finally, the alert status is reported in the Alert database to be displayed by the GUI.

For testing, different types of attack scenarios were simulated consisting of mainly three categories of tests with the following objectives and expectations:

#### TEST1

*Objective:* For misuse of telnet, the objective was to generate a high number of failed login attempts for specific hosts (HostID: 1037, 1066) and users (UserID: 2053, 2067) a moderate number of failed login attempts affecting specific hosts

(HostID: 1047, 1066), specific users (UserID: 2065, 2067, 2093) and specific host and user pairs.

*Expectation:* The expectation was that the Decision Engine would be able to correlate all these failed attempts to recognize suspicious activities in the network and reliably generate alerts for the hosts/users affected in such activities.

#### TEST2

*Objective:* For misuse of finger, the objective was to generate a high number of recursive fingers for specific host (HostID: 1047).

*Expectation:* The expectation was to see if the Decision Engine would be able to capture such a finger bomb targeted for a specific host and generate a high alert accordingly.

#### TEST3

*Objective:* For SYN flood, the objective was to flood a specific host with spoofed source IP addresses (HostID: 1066).

*Expectation:* The expectation was that the Decision Engine would be able to use knowledge of this situation to immediately generate the highest alert for the host involved.

The attack simulation for telnet and finger misuse was staged in real time where the misuse detection sensors and the Decision Engine operated almost simultaneously but for the SYN-flood attack, the data was pre-collected due to security concerns in the CCSR.<sup>17</sup> For testing anomaly alert inference, data was used for two different anomaly detection sensors to determine whether or not the Decision Engine would be able to fuse the detection data and generate network anomaly alert inference accordingly. The results of the Decision Engine were reviewed through the presentation tool designed for the security administrator,<sup>13</sup> i.e. the graphical user interface (GUI). Details of the experimental setup are described below.

#### 2.6.2. Results and analysis

The results of the experiments were encouraging. The Decision Engine fused and analyzed information in the detection database effectively. The decision making results were reported correctly in the Alert database. In this paper, due to space constraint, we report only a few examples of our results of the Decision Engine's network health assessment, followed by analysis.

First, we investigate an example from TEST1 and TEST3. We analyze the host with HostID 1066. As the attack simulations for TEST1 and TEST3 were carried out, the MDM1 relation for capturing failed login attempts recorded the following

data into MDM1 relation in the IIDS detection database:

ID	SOURCE HOST ID	USER ID	LOGIN NUMBER	OCCURANCE DATE	OCCURANCE TIME	RECENT
21106602322173447	1066	2067	7	18-NOV-02	173447	1
21106602322173539	1066	2046	1	18-NOV-02	173548	1
21106602322173528	1066	2093	4	18-NOV-02	173528	1
21106602322173532	1066	2065	2	18-NOV-02	173532	1

The Decision Engine analyzed this data and activated multiple Suspicious Events for host 1066 and recorded the following in the Suspicious Events relation:

SID	SEID	MID	USID	ANID	RECENT	EVENT VALUE
1002322173800	310	1066	2067	9999	1	0.499524
1102322173801	311	1066	9999	9999	1	0.85156

The data in this relation describes the activated Suspicious Events concerning host 1066:

- (1) The record in MDM1 with RID: 21106602322173447 activated the event (SID: 1002322173800) of type Suspicious\_Event\_High\_Login\_Failure (SEID: 310) with a value in the medium range (0.499524).
- (2) All of the above records contributed to activate the event (SID: 1102322173801) of type Suspicious\_Event\_Login\_Failure\_Same\_Host\_Different\_User (SEID: 311) with a value in the high range (0.85156).

In addition to the failed login information in the MDM1 relation (TEST1), for the same host 1066, the Decision Engine found the following relevant data in the MDM3 relation (TEST3) dedicated to SYN\_Flood Attacks:

RID	TARGET HOST ID	SYN REQ NUMBER	SYN ATTACK	OCCURANCE DATE	OCCURANCE TIME	RECENT
23106602322171718	1066	100	1	18-NOV-02	171718	1

In response to this, the following event (SID: 3002322173825) of type Suspicious\_Event\_SYN\_Flood\_Attack (SEID: 330) for host 1066 was activated by the Decision Engine with the maximum possible value of 1:

SID	SEID	MID	USID	ANID	RECENT	EVENT VALUE
3002322173825	330	1066	9999	9999	1	1

After all the suspicious activities affecting the host 1066 were activated, the Decision Engine combined this information in order to compute its alert level. Since, this host was under an obvious SYN\_Flood attack, the Decision Engine retained the alert level for the host at its peak, i.e. 1 and reported the following in the Host Alert relation.

MID	CID	MACHINE ALERT VALUE	RECENT	ALERT DATE	CHECKED	CHECKED DATE	CHECKED BY
1066	4102322173833	1	1	18-NOV-02	0		

However, had the host 1066 not been under a SYN\_Flood attack, the Decision Engine would have computed the alert level for the host by combining all the evidence of the activated events using FCMs implemented in the neural network approach. If that were the case, then its alert value would have been reported by combining Suspicious\_Event\_High\_Login\_Failure and Suspicious\_Event\_Login\_Failure\_Same\_Host\_Different\_User:

MID	CID	MACHINE ALERT VALUE	RECENT	ALERT DATE	CHECKED	CHECKED DATE	CHECKED BY
1066	4102322173833	0.655984	1	18-NOV-02	0		

Next we investigated another example involving a user with UserID 2093 from TEST1. When the Decision Engine found the following data in the MDM1 relation concerning this user,

RID	SOURCE HOST ID	USER ID	LOGIN NUMBER	OCCURANCE DATE	OCCURANCE TIME	RECENT
21106602322173528	1066	2093	4	18-NOV-02	173528	1
21104702322173553	1047	2093	5	18-NOV-02	173553	1

it activated the event (SID: 1202322173813) of type Suspicious\_Event\_Login\_Failure Same\_User\_Different\_Host (SEID: 312) with a value in the medium range (0.63058).

SID	SEID	MID	USID	ANID	RECENT	EVENT VALUE
1202322173813	312	9999	2093	9999	1	0.63058

Finally, the alert value for this user is calculated as:

USID	CID USER	ALERT VALUE	RECENT	ALERT DATE	CHECKED	CHECKED DATE	CHECKED BY
2093	4202322173838	0.504464	1	18-NOV-02	0		

It should be noted that this alert level is different, actually less than the value of the Suspicious Event generated for this user because according to the FCM modeling, the activated Suspicious.Event.Login.Failure.Same.User.Different.Host (edge weight 0.8) produces lesser impact on the alert level than Suspicious.Event.High.Login.Failure (edge weight 1.0).

We now investigate a host (HostID: 1047) targeted in TEST2 (finger misuse). As TEST2 was carried out, the misuse detection sensors for capturing finger misuse recorded the following data into MDM2 relation:

RID	SOURCE HOST ID	TARGET TYPE	TARGET HOST	TARGET USER	FINGER NUMBER	OCCURANCE DATE	OCCURANCE TIME	RECENT
22104702322173544	1047	1	1047	9999	9	18-NOV-02	173544	1

The Decision Engine analyzed this data and activated for the event (SID: 2002322173822) of type Suspicious.Events.High.Fingers (SEID: 320) with a value in the high range (0.836667) and recorded the following in the Suspicious Events relation:

SID	SEID	MID	USID	ANID	RECENT	EVENT VALUE
2002322173822	320	1047	9999	9999	1	0.836667

In addition to the information in the MDM2 relation (TEST2), for the same host, the Decision Engine found the following relevant data in the MDM1 relation (TEST1):

RID	SOURCE HOST ID	USER ID	LOGIN NUMBER	OCCURANCE DATE	OCCURANCE TIME	RECENT
21104702322173430	1047	2067	4	18-NOV-02	173430	1
21104702322173455	1047	2065	1	18-NOV-02	173455	1
21104702322173553	1047	2093	5	18-NOV-02	173553	1

and therefore activated the Suspicious.Event.Login.Failure.Same.Host.Different.User with an event value in the medium range (0.681062) from the above records.

SID	SEID	MID	USID	ANID	RECENT	EVENT VALUE
1102322173816	311	1047	9999	9999	1	0.681062

After all the suspicious activities affecting the particular host were activated, the Decision Engine combined the information in order to compute the alert value for the host and reported the following in the IIDS alert database:

MID	CID	MACHINE ALERT VALUE	RECENT	ALERT DATE	CHECKED	CHECKED DATE	CHECKED BY
1047	4102322173832	0.767509	1	18-NOV-02	0		

Once the Decision Engine was done computing the alert level for all the hosts and users in the network, the Host Alert and User Alert relations report:

MID	CID	MACHINE ALERT VALUE	RECENT	ALERT DATE	CHECKED	CHECKED DATE	CHECKED BY
1037	4102322173829	0.681062	1	18-NOV-02	0		
1047	4102322173832	0.767509	1	18-NOV-02	0		
1066	4102322173833	1	1	18-NOV-02	0		

USID	CID	USER ALERT VALUE	RECENT	ALERT DATE	CHECKED	CHECKED DATE	CHECKED BY
2053	4202322173834	0.681062	1	18-NOV-02	0		
2065	4202322173836	0.198052	1	18-NOV-02	0		
2067	4202322173837	0.609166	1	18-NOV-02	0		
2093	4202322173838	0.504464	1	18-NOV-02	0		

The overall alert misuse host and user alert level was computed and was found to be 1.0 and 0.681062 respectively. This data along with the anomaly alert level of the network was reported in the Alert database. The GUI in the IIDS, which continued to monitor the intrusion database for new information in real time, picked up the alert data generated by the Decision Engine and displayed it in graphical form as shown in Fig. 10. The bar graph in the middle of the figure shows the alerts generated by the Decision Engine. The three bars indicate the misuse alert level of users, misuse alert level of hosts and anomaly alert level of the network respectively. The percentages shown indicate the level of alert and the color codes used were Green for a low level of alert (<30%), Yellow for a Moderate level of alert (30% to 70%) and Red for a High level of alert (>70%).<sup>13</sup>

The IIDS intrusion database is designed to track alert generation from the overall alert level to the primary detection database level. In addition, our GUI is also equipped with the step-by-step drill down capability from this uppermost level of security status view to details of intrusion at the detection database level. Therefore, the security administrator is able to “click to view” the thread of alert generation starting from the bar graphs of overall alerts to the misuse/anomaly detection database level and trace back the Decision Engine’s reasoning process.

Below we include the GUI snapshots to demonstrate the operations of the Decision Engine. Following the example described below (for the host with

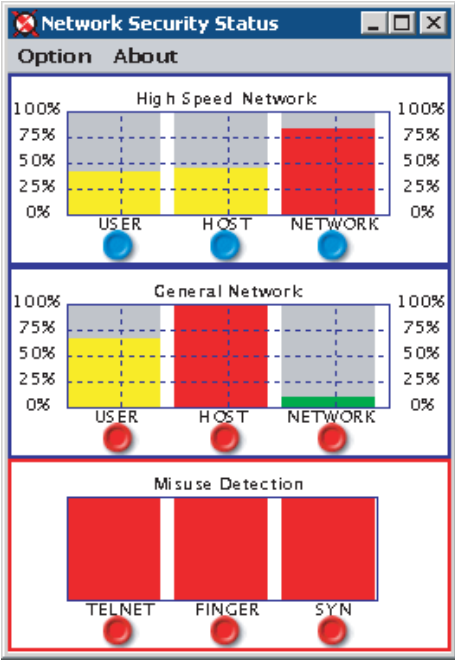


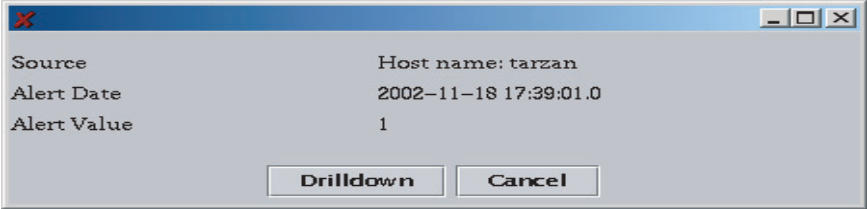
Fig. 10. Top level GUI display.

HostID: 1066 and Host Name: Tarzan), we show the capability to trace back leading to the following chain of evidence (upper to lower levels):

- (1) At level 1: From overall misuse alert level to all alerts for the involved hosts:



- (2) At level 2: From alert situation for all hosts to details of a particular host:



(3) At level 3: From alert information of particular host to the suspicious events information that contributed to the alert:

ID	INTRUSION_NAME	EVENT_VAL...	NODE_NAME	USER_NAME	SEID
1002322173800	High_Login_Failure	0.499524	tarzan	yun	310
1102322173801	Login_Failure_Same_Host_Different_User	0.851560	tarzan		311
3002322173825	SYN_ATTACK	1	tarzan		330

(4) At level 4: From particular Suspicious Event’s information to the detection data that contributed to the activation of the Suspicious Event:

SOURCE_HOST	USER_NAME	LOGIN_NUMBER	OCCURRENCE_DATE	OCCURRENCE_TIME
tarzan	kc8	1	2002-11-18 00:00:00	173548
tarzan	tomy	4	2002-11-18 00:00:00	173528
tarzan	wss2	1	2002-11-18 00:00:00	173539
tarzan	yan	2	2002-11-18 00:00:00	173532
tarzan	yun	7	2002-11-18 00:00:00	173447

For the example cited for the user with UserID 2093 (User Name: Tomy) the same type of drill down presented the following:

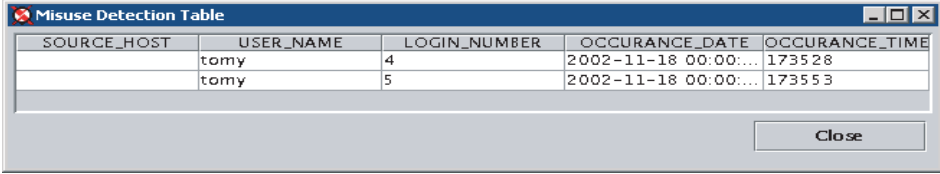
(1) Level 1

(2) Level 2

(3) Level 3

ID	INTRUSION_NAME	EVENT_VALUE	NODE_NAME	USER...	SEID
1202322173813	Login_Failure_Different_Host_Same_User	0.630580		tomy ...	312

(4) Level 4



SOURCE_HOST	USER_NAME	LOGIN_NUMBER	OCCURANCE_DATE	OCCURANCE_TIME
	tomy	4	2002-11-18 00:00:...	173528
	tomy	5	2002-11-18 00:00:...	173553

Another objective of our Decision Engine is to check misuse history for a specific host or user within a specified time frame. Currently, the Decision Engine implements this feature for a time frame of one hour and for non-obvious suspicious events. To test this feature, we performed more failed attempts for Host 1066 within one hour of the previously described experiment. In the case where the Decision Engine did not perform any misuse history check for the host, this scenario activated a very low Suspicious Event (0.126509), which in turn contributed to a low alert value of 0.126509. But, in the case where the Decision Engine did carry out the history check, i.e. looked into an hour of history of similar types of misuse for the host, it raised the value of the same event to 0.85156 and raised the Low alert level to be Moderate.

In this paper, due to space constraints, we analyzed the results for a few cases. Other cases presented us with similar data and analysis activities.

In an earlier paper,<sup>12</sup> we have reported our use of MIT Lincoln Lab’s data set in anomaly detection with fuzzy data mining. For our Decision Engine’s functionality testing, we did not use this data set because our work has been designed primarily with the high performance cluster environment in mind and we have already migrated to that environment. The MIT Lincoln Lab’s data set is not appropriate in such an environment, which is significantly different from traditional environment in terms of configuration, network failures and performance issues and therefore, required us to use a customized attack dataset suited for this environment. Torres *et al.* reports on this attack dataset for the cluster environment in Ref. 36.

### 3. Conclusions and Future Work

This paper described the workings of a Decision Engine for our Intelligent Intrusion Detection System that utilizes causal knowledge inference based on Fuzzy Cognitive Maps (FCM). We decided to use cognitive modeling with FCMs because FCMs are interesting yet simple decision support tools. Using FCMs for decision support is attractive for the following reasons:

- FCMs offer a practical yet natural knowledge acquisition scheme that represents expert’s knowledge in a structured way and works well with human expert’s thinking.
- FCMs are particularly suitable in a dynamic environment such as the network security domain because they are flexible enough to capture adaptive nature

of human knowledge and therefore one can easily add new concepts or delete idle/obsolete concepts as necessary without difficulty.

- FCMs are suitable for soft knowledge domains such as intrusion detection where systems concepts/ relationships and also the meta-system language are essentially fuzzy.
- FCMs accommodate disparate entities easily, which is particularly attractive in this context of decision making where multiple sensors with dissimilar causes and effects are at play.
- FCMs allow fast computation, which is a requirement for any real time intrusion detection architecture.
- FCMs help prevent certain kinds of knowledge extraction problems often encountered in traditional rule-based systems.<sup>5</sup>

The FCMs in the Decision Engine are simple in the sense that they do not contain any feedback loops. There are two reasons for this. One is that there was no need/demand of the feedback loops for the particular FCM models that we have been using for suspicious event generation and alert fusion. Second, we are hesitant to use feedback loops because such complex FCMs can often give rise to instability and chaotic behavior in the system,<sup>7</sup> which is naturally undesirable for a high-risk system such as a real-time intrusion detection system. We also understand that sometimes intrusive behavior can be complex and therefore, when needed, we will investigate complex FCMs with feedbacks.

In this project we have used FCMs whose structure has been defined by human experts, in this case, it will be the security administrator. In the future, we may explore how FCMs can self-learn and self-train like neural networks with minimal involvement of the human expert. Also, in order to deal with unseen/novel situations, we plan to explore how FCM structures can automatically be constructed from available data itself.

Currently, when evidence of intrusion is found for certain hosts/users, the Decision Engine looks back in history within a time window to check if there is more such suspicious activity that affected the host/user concerned. If so, the Decision Engine also takes that information into account. The time window is now limited because looking back further in the past would unacceptably slow down the Decision Engine. There are also data storage issues involved. In the future, we may extend the time window further and employ more detailed background checking.

At present the Decision Engine is working in the IIDS architecture running in a high performance cluster environment where the Decision Engine resides centrally and assesses the health of the network by analyzing the different hosts/users in the cluster. The cluster presently employs an open source misuse sensor, Snort to monitor network traffic and multiple anomaly sensors<sup>11,23</sup> to profile host/user behavior. Because the architecture and the technology used in the Decision Engine is scalable, the Decision Engine should be able to incorporate such responsibilities in the cluster environment.

Currently, we are in a process to tailor the IIDS database to conform to the Intrusion Detection Message Exchange Format (IDMEF), an effort to standardize alert and intrusion management from the Internet Engineering Task Force (IETF).<sup>15</sup> We are also customizing the Decision Engine's functionality to accommodate the use of such standard.

## Acknowledgements

The authors wish to express their sincere appreciation to all the reviewers, whose helpful comments made the paper better and more accurate. We would particularly like to thank one reviewer who provided additional references for us.

This work was partially sponsored by Grant#N00014-01-1-0678 from the office of the Naval Research, Grant#CCR-0098024 from the National Science Foundation, and Contract # DAAD1701C001101010015 from the Army Research Laboratory.

## References

1. R. Axelrod, *Structure of Decision: The Cognitive Maps of Political Elites* (Princeton University Press, Princeton, NJ, 1976).
2. S. M. Bridges and R. B. Vaughn, Intrusion detection via fuzzy data mining, in *Proc. 12th Annual Canadian Information Technology Security Symposium*, June, 2000, Ottawa, Canada, Communications Security Establishment, pp. 111–121.
3. S. M. Bridges, R. B. Vaughn and A. Siraj, AI techniques applied to high performance computing intrusion detection, in *Proc. 10th Int. Conf. Telecommunication Systems, Modeling and Analysis*, Monterey, CA **2** (3–6 October 2002) 100–114.
4. D. Brubaker, Fuzzy cognitive maps, *EDN Access*, April 1996.
5. M. Caudill, Using neural nets: Fuzzy cognitive maps, *AI Expert* **6** (1990) 49–53.
6. V. Dhar and R. Stein, *Seven Methods for Transforming Corporate Data into Business Intelligence* (Prentice Hall Inc., NJ, 1997).
7. J. A. Dickerson and B. Kosko, Virtual worlds as fuzzy dynamical systems, in *Technology for Multimedia*, ed. B. J. Sheu (IEEE Press, 1996).
8. J. A. Dickerson, Z. Cox, E. S. Wurtele and A. W. Fulmer, Creating metabolic and regulatory network models using fuzzy cognitive maps, in *Proc. IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference*, July 2001, Vancouver, British Columbia.
9. J. E. Dickerson and J. A. Dickerson, Fuzzy network profiling for intrusion detection, in *Proc. 19th Int. Conf. North American Fuzzy Information Processing Society*, July 2000, Atlanta, pp. 301–306.
10. J. E. Dickerson, J. Juslin, O. Koukousoula and J. A. Dickerson, Fuzzy intrusion detection, in *Proc. IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference*, Vancouver, British Columbia **3** (July 2001) 1506–1510.
11. G. Florez, A trusted environment for MPI programs, Masters thesis, Mississippi State University (2002).
12. G. Florez, S. M. Bridges and R. B. Vaughn, An improved algorithm for fuzzy data mining for intrusion detection, in *Proc. North American Fuzzy Information Processing Society Conference*, 27–29 June 2002, New Orleans, LA.

13. H. Gao, A presentation tool for the intelligent intrusion detection system, Masters Project Report, Mississippi State University, 2002.
14. J. Gomez and D. Dasgupta, Evolving fuzzy classifiers for intrusion detection, in *Proc. 3rd Annual IEEE Information Assurance Workshop*, 17–19 June 2000, West Point, New York.
15. Internet Engineering Task Force (IETF), Intrusion detection exchange format (idwg) (current 29 September 2003), <http://www.ietf.org/html.charters/idwg-charter.html>.
16. J. S. R Jang, C. T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing* (Prentice Hall, Upper Saddle River, NJ, 1997).
17. S. Kayapanda, Implementation of a prototype misuse detection component for an intelligent intrusion detection system, Masters Project Report, Mississippi State University, 2002.
18. B. Kosko, Fuzzy cognitive maps, *Int. J. Man-Machine Studies* **24** (1986) 65–75.
19. B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence* (Prentice Hall, Englewood Cliffs, NJ, 1992).
20. B. Kosko, *Fuzzy Thinking: The New Science of Fuzzy Logic* (Hymerion, New York, 1993).
21. B. Kosko, *Fuzzy Engineering* (Prentice Hall, Upper Saddle River, NJ, 1997).
22. W. Lei, The integration of security sensors into the intelligent intrusion detection system (IIDS) in a cluster environment, Masters Project Report, Mississippi State University, 2002.
23. Z. Liu, A lightweight intrusion detection system for the cluster environment, Masters Thesis Report, Mississippi State University (2003).
24. C. G. Looney, Fuzzy rules and approximate reasoning (current 20 October 2002). <http://ultima.cs.unr.edu/cs791j/unit2791j.htm>.
25. J. Luo, Integration fuzzy logic with data mining methods for intrusion detection, Masters thesis, Mississippi State University (1999).
26. J. Luo and S. M. Bridges, Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection, *Int. J. Intelligent Systems* **15**(8) (2000) 687–703.
27. T. D. Ndousse and T. Okuda, Computational intelligence for distributed fault management in networks using fuzzy cognitive maps, in *Proc. IEEE Int. Conf. Communications Converging Technologies for Tomorrow's Application* (IEEE, New York, 1996), pp. 1558–1562.
28. R. A. Orchard, NRC FuzzyJ Toolkit for the Java(tm) Platform User's Guide Version 1.3, (current 20 October 2002), [http://www.iit.nrc.ca/IR\\_public/fuzzy/fuzzyJDocs/](http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJDocs/).
29. C. F. Pelaez and J. B. Bowles, Applying fuzzy cognitive maps knowledge representation to failure modes effects analysis, in *Proc. IEEE Annual Symposium on Reliability and Maintainability*, 1995, pp. 450–456.
30. C. P. Pfleeger, *Security in Computing* (Prentice Hall, Inc., Upper saddle River, NJ, 1996).
31. A. Siraj, S. M. Bridges and R. B. Vaughn, Fuzzy cognitive maps for decision support in an intelligent intrusion detection system, in *Proc. Int. Fuzzy Systems Association/North American Fuzzy Information Processing Society (IFSA/NAFIPS) Conference on Soft Computing*, July 2001, Vancouver, Canada.
32. E. Smith and J. H. P. Eloff, Cognitive fuzzy modeling for enhanced risk assessment in health care institution, *IEEE Intelligent Systems and their Applications*, March/April 2000, pp. 69–75.
33. C. D. Stylios and P. P. Groumpos, A soft computing approach for modelling the supervisor of manufacturing systems, *Journal of Intelligent and Robotics Systems* **26**(3–4) (1999) 389–403.

34. A. Sundaram, An introduction to intrusion detection (current 20 January 2001), <http://www.cs.purdue.edu/coast/archive/data/categ24.html>.
35. R. Taber, Knowledge processing with fuzzy cognitive maps, *Expert Systems with Applications* **2** (1991) 83–87.
36. M. Torres, L. Zhen, G. Florez, R. B. Vaughn and S. M. Bridges, Attacking a high performance computer cluster, in *Proc. 15th Annual Canadian Information Technology Security Symposium*, 12–15 May 2003, Ottawa, Canada.
37. J. Q. Xin, J. E. Dickerson and J. A. Dickerson, Fuzzy feature extraction and visualization for intrusion detection, in *Proc. FUZZ-IEEE*, 2003, St. Louis, MO, USA.
38. J. Yen and R. Langari, *Fuzzy Logic: Intelligence, Control and Information* (Prentice Hall, Upper Saddle River, NJ, 1999).