Alert Prioritization in Intrusion Detection Systems

Khalid Alsubhi*, Ehab Al-Shaer*‡, and Raouf Boutaba*

(*)Davird R. Cheriton School of Computer Science, University of Waterloo, Canada
(‡)School of Computer Science, DePaul University, Chicago, USA
email: kaalsubh@cs.uwaterloo.ca; ehab@cs.depaul.edu; rboutaba@uwaterloo.ca

***Abstract*—Intrusion Detection Systems (IDSs) are designed to monitor user and/or network activity and generate alerts whenever abnormal activities are detected. The number of these alerts can be very large; making the task of security analysts difficult to manage. Furthermore, IDS alert management techniques, such as clustering and correlation, suffer from involving unrelated alerts in their processes and consequently provide imprecise results.**

In this paper, we propose a fuzzy-logic based technique for scoring and prioritizing alerts generated by an IDS[1]. In addition, we present an alert rescoring technique that leads to a further reduction of the number of alerts. The approach is validated using the 2000 DARPA intrusion detection scenario specific datasets and comparative results between the Snort IDS alert scoring and our scoring and prioritization scheme are presented.

***Index Terms*—Alert management, alert prioritization**

I. Introduction

Network attacks are growing more serious, forcing system defenders to deploy appropriate security devices such as firewalls, Information Protection Systems (IPSs), and Intrusion Detection Systems (IDSs). An IDS is aimed to inspect user and/or network activity looking for suspicious behavior that they report to security analysts in the form of alerts. There are two common types of IDSs depending on the method employed for traffic inspection: signature-based and anomaly-based. A *Signature-based* IDS generates an alert when the traffic contains a pattern that matches signatures of malicious or suspicious activities. An *anomaly-based* IDS examines ongoing activity and detects attacks based on the degree of variation from normal past behavior [4]. However, both of these mechanisms suffer from the problem of generating a large number of alerts. These alerts need to be evaluated by security analysts before any further investigation in order to take appropriate actions against the attacks.

IDSs usually generate a large number of alerts whenever abnormal activities are detected. Inspecting and investigating all reported alerts manually is a difficult, error-prone, and time-consuming task. In addition, ignoring alerts may result into successful attacks being missed. To tackle this problem, low-level and high-level alert evaluation operations have been introduced. Low-level alert operations deal with each alert individually to enrich its attributes or assign a score to it based on the potential risk. High-level alert management techniques, such as aggregation, clustering, correlation, and fusion, were proposed to deal with sets of alerts and provide an abstraction of them. However, the high-level techniques suffer from including alerts that are not significant, which leads to inappropriate results. Therefore, low-level evaluation techniques are needed to automatically (or semi-automatically) examine large numbers of alerts and prioritize them, leaving only important alerts for further inspection. Accordingly, the reduced set of alerts leads to more precise high-level alert analysis. From this work, the security administrator will be provided with an effective technique to evaluate and manage alerts, thereby saving his or her time and effort.

This paper describes a method for automatically evaluating IDS alerts based on metrics related to the applicability of the attack, the importance of victim, the relationship between the alert under evaluation and previous alerts, and the social activities between the attackers and the victims. These metrics are input of a Fuzzy logic system in order to investigate the seriousness of the generated alerts and assign a score to each of them. This evaluation process will prioritize alerts when presented to the security administrator for further investigation. Additionally, we propose a rescoring technique to dynamically rescore alerts based on the relationship between attacks or the degree of maliciousness of an attacker. Finally, we validate our proposal by two experiments. In the first experiment, we score alerts generated by the Snort IDS [28] using 2000 DARPA intrusion detection scenario specific datasets [15]. In the second experiment, we use the same dataset to validate our rescoring technique.

The paper is organized as follows. Section II discusses related works. Section III describes our proposed alert prioritization system. Section IV presents the identified alert prioritization metrics. Section V explains fuzzy logic inference and its use in this work. Section VI explains the technique for alert rescoring. Simulation results are presented and discussed in Section VII. Finally, Section VIII concludes the paper.

II. Related Work

Attacks are presented to a security administrator through alerts generated by the sensing devices, such as IDSs. It is common that an IDS generates a large number of alerts whenever the defined policies (rules) have been matched. With a large number of alerts, security administrators are overwhelmed and it becomes difficult to manually distinguish between the real attacks and the false ones. Two general

[1]This research has been supported under the Natural Science & Engineering Research Council of Canada (NSERC) grant STP-322235-05.

978-1-4244-2066-7/08/$25.00 ©2008 IEEE

solutions exist to deal with this problem. The first solution focuses on the monitoring device itself by enhancing its detection mechanism, eliminating its unnecessary rules, optimizing its signatures, and choosing the right location [22]. Although this solution promises to reduce the number of alerts, it requires prior knowledge by the security administrator of the detection mechanism. The second solution focuses on the sensor's outputs. Several IDS alert management techniques fall into this category and include aggregation [5], clustering [14], correlation [18], [26], and fusion [8]. Generally, reducing the number of alerts, prioritizing the most critical attacks, and discarding the false alerts are the main objectives of IDS alert management approaches. Furthermore, these techniques assist the security administrators in understanding the situation revealed by the IDS.

The techniques to construct attack scenarios fall into three classes. The first class includes correlating alerts based on the similarity between alerts attributes, such as IP addresses, ports etc. (e.g., probabilistic alert correlation [26]). The second class is based on specifying known attack sequences [2]. The third class is based on the dependencies between alerts by matching prerequisites (of an attack) with consequences (output and state after the attack succeeds) of attacks [18]. In this work, we used the first class by defining a relationship metric between alerts.

In [6], Jinqiao Yu et al. evaluate alerts based on two aspects. Firstly, alerts that do not correspond to any attack in the vulnerability knowledge base are prioritized for further investigation and secondly, the applicability of the attack against the protected network is examined. Similarly, Qin and Lee [21] compute the alert priority score based on the severity of the corresponding attack and the relevance of the alert to the protected networks and hosts. Porras et al. propose an alert ranking technique, known as the M-Correlator [19]. This approach ranks alerts based on the likelihood of the attack to succeed, the importance of the targeted asset, and the amount of interest in the type of attack. Although these techniques are promising in the evaluation of alerts generated by signature-based IDSs, they cannot evaluate alerts raised by anomaly-based IDSs, since they heavily rely on the vulnerability knowledge base. Our work extends on these works by offering a technique which works with both signature-based and anomaly-based IDSs and makes use of additional criteria, such as the sensor sensitivity, relationship between alerts, service stability, and social activity between source and target for more accurate evaluation of the alerts. Furthermore, our approach differs from previous ones by providing a rescoring function of early non-critical attacks that prepare for later critical ones. This way, the early steps of the attack will be prioritized for further analysis, such as correlation.

Several works have focused on devising a standard format for representing and exchanging IDS alerts. Examples include CIDF, [24], IDXP [9], and IDMEF [3]. The Intrusion Detection Message Exchange Format (IDMEF) [3] is a standard data format that reports alerts about events considered suspicious. IDMEF seems to be a promising solution as it provides flexibility for future extensions. As shown in figure 1, IDMEF uses an XML format to represent the detected event information. In our work, we made use IDMEF and this provided us with the ability to expand alert reports to include both the overall alert score and the alert rescoring values (figure1).

```
...
< IDMEF-Message version="1.0" xmlns="xml:idmef">
 <Alert messageid="abc123456789">
  <Analyzer analyzerid="bc-sensor01">
  ...
  <Source ident="a1a2" spoofed="yes">
   <Node ident="a1a2-1">
    <Address ident="a1a2-2" category="ipv4-addr">
     <address¿192.0.2.200¡/address>
    </Address>
   </Node>
  </Source>
  ...
 <AdditionalData type="integer" meaning="Score">
  10
 </AdditionalData>
 <AdditionalData type="integer" meaning="Rescore">
  10
 </AdditionalData>
 </Alert>
</IDMEF-Message>
```

Fig. 1. IDEMF Extended Format

III. General Architecture Overview

As shown in the doted areas of figure 2, our alert management architecture involves three main components: (a) data collection, (b) alert scoring metrics and inference, and (c) alert analysis.

Data collection includes the alert database, environment parameters, security administrator parameters and the vulnerability knowledge base. Alert attributes consist of several fields that provide information about the attack. This information varies from one IDS product to another. However, we assume that the alert structure is compatible with the IDMEF format. The security administrator can specify the environment parameters to involve in the evaluation process. These can include for instance information about running services, applications, operating systems (+versions), and existing (current) vulnerabilities. Furthermore, the security administrator can specify the importance of each host in the network including the IDSs. Public vulnerability knowledge bases, such as the National Vulnerability Database (NVD) [17] and Bugtraq [12], contain detailed information about known attacks. The availability of such data bases can help in the alert evaluation process. The data collection component makes the above resources available to the alert scoring metrics and inference component.

Alert Scoring Metrics and Inference component is a key element of our architecture. Based on the information received from the data collection component, several metrics are computed and used as indicators to accurately evaluate the alerts. In this perspective, the computed metric values are passed to the Fuzzy logic inference engine to calculate the overall alert score.

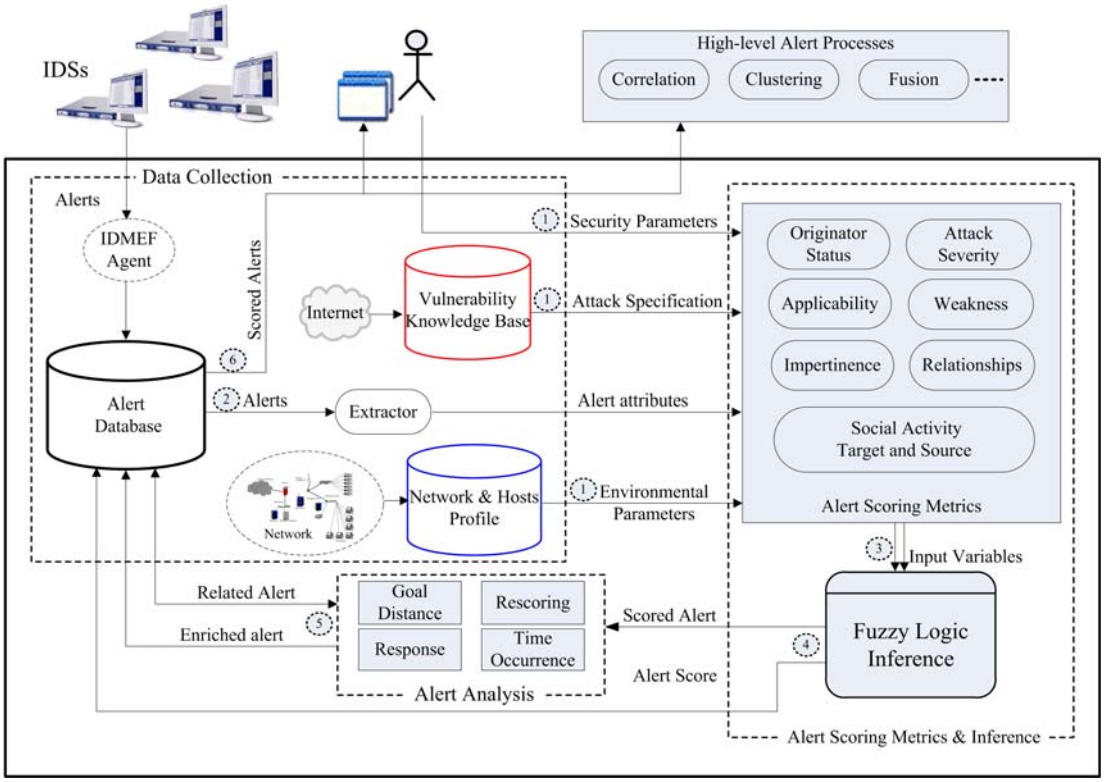

Fig. 2. General Evaluation Process

Alert analysis provides an additional evaluation of the IDS alert. This component hosts the functions of alert rescoring, measurement of distance to known attacks, time of occurrence, and response plans.

As can be deduced from figure 2 the alert evaluation process is carried out as follows. When a new alert is received, the associated scoring metrics are computed based on the available security parameters, environmental parameters and vulnerability knowledge base. Fuzzy logic inference is then employed in order to assign a score to the alert based on the metrics values. After that, the alert is stored in the alert database with its score. The alert is also passed to the alert analysis component for further investigation.The analysis measures the distance of the current attack from its possible goals, rescores the alerts that are suspicious to be a preparation step for later attacks, detects activities that violate the scheduled time of permissibility of certain activities, and provides a response plan. Finally, alerts which received high scores are presented to the security administrator for further investigation.

IV. Alert Prioritization Metrics and Reasoning Approach

The alert scoring metrics of figure 2 are used to evaluate the criticality of alerts Although some of these metrics have been individually used in previous works (i.g., [6], [16], [20], [21]), they have not been used altogether as proposed in this paper. Additionally, we define new metrics that help in accurately evaluating IDS alerts. Our scoring technique does not require all the metrics to be available during the evaluation. Intuitively, the presence of a large number of indicators will definitely increase the accuracy of the alert score. However, most of the metrics are easy to obtain, especially those that deal with protected environments and the vulnerability knowledge base. In the following, the alert scoring metrics presented in figure 2 are detailed.

A. Applicability metric

Applicability is a process that checks whether an attack that raises an alert is applicable in the current environment. This requires knowledge from the vulnerability knowledge base, the set of running services, applications, and operating systems. Alert attributes help in identifying the potential attack(s). Then, the vulnerability knowledge base is checked to determine whether the attack is applicable in the current environment. In general, figuring out the applicability of the attack on a given network is reduced to a search problem.

B. Importance of Victim metric

This metric is used to specify critical machines, services, applications, accounts, and directories in the environment. The goal of this metric is to increase the score of alerts related to suspicious activities that target critical system components, such as a main server. Equation 1 computes a weighted projection is used to calculate the value of the importance metric. A high weight is chosen if the target is critical and vice versa.

$$Importance(M) = \prod_{x \in \{serv,app,acct,dir\} \text{ running on } M} w(x) \quad (1)$$

TABLE I
SENSOR STATUS PARAMETERS

Variable	States
Placement	Critical Moderate Regular
Configuration	Configure Not Configure
Accuracy (BDR)	Probability (0-1)
Up_to_Date	Updated Not updated

$Importance$ describes the condition of the victim machine that is running in the protected environment. The value of the $Importance$ is calculated on a scale from 0 to 1. 0 indicates that the victim machine reported in the alert does not include any important host, service, application, account, or directory. Scores closer to 1 indicate that the attack is targeting an important system component.

C. Sensor Status metric

What part of the environment does the IDS monitoring device cover? Is it well configured? Is it up-to-date? What is its accuracy? Answering these questions for each sensor in the environment gives an evaluation of its status. Table I shows all possible values that can be entered by the security expert who manages the sensor. The accuracy value of the sensor can be calculated offline using the Bayesian Detection Rate (BDR) formula [1]. The Bayesian detection rate computes the true positive probability $P(A/I)$ that alert A was raised given the attack I has been detected. The BDR formula uses the past experience of the sensor activity as follows:

$$P(A|I) = \frac{P(I)P(A|I)}{P(I)P(A|I) + P(\neg I)P(A|\neg I)}. \quad (2)$$

Where:
- I: Attack has occurred.
- $\neg I$: No Attack has occurred.
- A: Alert has been raised.
- $\neg A$: No Alert has been raised.

The status of the sensor is computed based on a simple projection function. For instance, if the location of the sensor is *Critical*, its accuracy is *high*, it is well *Configured*, and it is *up to date*, then the status of the sensor it *high*. Accordingly, alerts generated by that sensor are given more confidence and are subject to higher scrutiny.

D. Attack Severity metric

The severity score metric measures the risk level posed by a particular vulnerability. There are several sources which provide severity scores for known attacks, such as MITRE Common Vulnerabilities and Exposures (CVE) [27], Secunia [23], and industrial products (e.g., Microsoft). We provide a measurement of the severity score which makes use of the the multiple scores provided by the above organizations. The severity score value may vary from one organization to another. For instance, the FileZilla unspecified format string vulnerability has been reported in NIST as a very severe vulnerability scored 10 out of 10, while the Secunia reported this vulnerability to be moderately critical.

$$SS(a) = \frac{\sum_{i=1}^{n} w(SS_i) \times SS_i(a)}{\sum_{i=1}^{n} w(SS_i)} \quad (3)$$

SS(a) represents the expected severity of the attack that triggered alert a. Severity scores that are available from security analysis databases come in numerical or categorical formats. There is a need to first normalize the severity score to a numerical value. Then, the weighted average of the available severity scores is computed based on equation 3.

E. Service vulnerability metric

We adopt the method proposed by Abedin et al [7] to analyze only the service that the attacker is targeting. This method is used to calculate a unified score representing the strength or weakness of the targeted service. The result is then used in the overall alert scoring.

Since the targeted service is listed in the alert's content, it is possible to check the set of current vulnerabilities of that service. A second source of information is to mine the vulnerability knowledge bases to check how vulnerable this service has been in the past. This is related to those vulnerabilities that have been removed through software patches. In addition, since newly released services tend to have more vulnerabilities than services that have been in use since long, using the service release time contributes too to the overall service vulnerability analysis. In summary, it is possible to measure the Vulnerability Score VS(s) of a service s which appeared in the alert a based on the current vulnerability score $VS_c(s)$, the past vulnerability score $VS_p(s)$, and the release time $RT(s)$ of the service.

In order to determine the set of services, applications, and operating systems that the network is running, and consequently find out the current and historical vulnerabilities associated with them, available network scanning software, such as Nmap [10], or Nessus [11] can be used. For both existing and historical vulnerabilities, we are interested in the severity score SS. The service release time $RT(s)$ serves an indicator of the stability of the service. As shown in figure 3, experience has shown that services in the early days of their release have more vulnerabilities than those released longer in the field.

1) Existing vulnerability score metric: A raised alert a explicitly mentions the targeted service s (including the application, OS, or service) that the attacker is trying to violate. In may cases, the targeted service can also be determined from the port number that is stated in the alert. For a specific or group of targeted services S_i(s), we can explore the existing vulnerabilities V_i(s) and calculate the value of the VS_e based on the severity score SS of these vulnerabilities. However, there is a difference between vulnerabilities having a published solution that has not yet been applied, and vulnerabilities that

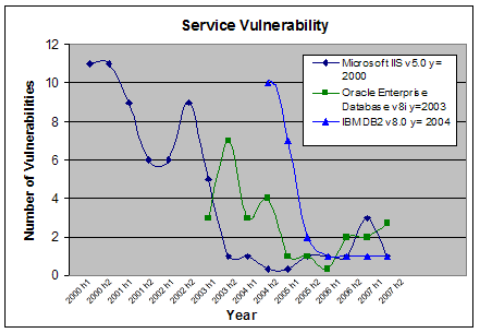

Fig. 3. Evolution of Service Vulnerabilities Over Time Generated from data provide in [13]

still wait for a solution. The VS_e score is more biased towards the highest severity score $SS(v)$ of the existing vulnerabilities. The following equation calculates the VS_e by the weighted arithmetic mean as follows:

$$VS_e(s) = \frac{\sum\limits_{v \in V_e(s)} w(v) \times SS(v)}{\sum\limits_{v \in V_e(s)} w(v)} \quad (4)$$

2) Historical vulnerability score metric: For the historical vulnerability score $VS_h(s)$ of service s, vulnerability knowledge bases, such as CVE, are consulted to measure the stability of the service in the past. The criticality, represented by the severity score SS, of the past vulnerabilities can be high, medium, or low. High risk vulnerabilities receive a high score while low risk vulnerabilities receive a low score. In addition, the severity of a vulnerability decreases as it gains in age, reflecting the fact that vulnerabilities known since long tend to be no more efficient. As a result, old vulnerabilities receive low scores. Equation 5 shows how $VS_h(s)$ is computed.

$$VS_h(s) = \frac{\sum\limits_{v \in V_h(s)} w(v) \times SS(v) \times \varepsilon^{-age(v)}}{\sum\limits_{v \in V_h(s)} w(v)} \quad (5)$$

Finally, the overall vulnerability score $VS(s)$ of a service s is computed as the weighted average of VS_e, VS_h, and RT:

$$VS(s) = \eta_1 \times VS_e(s) + \eta_2 \times VS_h(s) + \eta_3 \times RT(s) \quad (6)$$

F. Relationship between Alerts

Usually attackers use multiple steps in order to achieve their final goal. The early steps are a preparation for the later ones. Calculating the final score of an alert needs hence to involve the relationship it has with previous ones. The score of the currently evaluated alert will increase if it is found that it has a relationship with other stored alerts. Algorithm 1 illustrates the procedure we use to evaluate the relationship between alerts. We restrict the search to those alerts that occurred in a close period -say a couple of hours- from the current alert. This restriction discards very old alerts and focuses on recent ones since attackers typically try to achieve their goal as soon as possible before they can be identified.

Algorithm 1 Alerts Relationship Algorithm

Require: Alert ca, AlertLog ℓ
Ensure: Relationship Degree
1: A= $\{a \mid a \in \ell \wedge$ Timestamp(a) within time threshold$\}$, $n = |A|$.
2: **for** $i = 0$ to n **do**
3: foreach alert $a \in$ A
4: Calculate the Relationship Score between Current Alert and a
5: **if** Relationship Score $>$ Highest Relationship Score **then**
6: Highest Relationship Score= Relationship Score
7: **end if**
8: **end for**
9: **Return** Relationship Score

The algorithm starts by measuring the similarity between the source and target IP addresses and port numbers of the alert being evaluated and all previous alerts. The following simple matching coefficient is used for IP address matching and takes into consideration the subnet the IP addresses belong to:

$$IP_{ss}(a_1, a_2) = \frac{\sum similar\ bits(a_1, a_2)}{\sum all\ bits(a_1, a_2)}$$

If the source addresses of the current alert and the previous alert share the same subnet then the IP similarity score IP_{ss} will be high. Similarly, the port number similarity takes a Boolean scored (1 for a matching). In addition, the type of attack is taken into account when calculating the relationship score. For the similarity between attack types, we benefit from the statistical analysis proposed by Valdes et al [26].

G. Social activity metric

A social network is a social structure made of nodes that are tied by one or more specific types of relations. In this metric, we are trying to construct and analyze the social network for the source and target that are stated in the alerts' attribute. The node of the social network will be the source address, target address, attack ID, and the sites the user has visited. The relationship between the nodes differs according to the object of the node. For instance, the social relationship between an attacker and a victim raises an "$alerted$" situation, whereas if it were a worm and host, we would have an "$infected$" situation. The goal in this metric is to find a triangular relationship that involves a hidden participant. This hidden participant could be a previous activity of the recent attacker.

V. Fuzzy Logic Inference

A Fuzzy logic system reasons about the data by using a collection of fuzzy membership functions and rules. It makes clear conclusions possible to derive from imprecise information. In this regard, it resembles human decision making because of its ability to work with approximate data and find precise results. Fuzzy logic differs from classical logic

TABLE II
FUZZY LOGIC INFERENCE RULES

Criteria	Rule1	Rule2	Rule3	Rule4	Rule5	Rule6	Rule7	Rule8	Rule9	Rule10	Rule11	Rule12
Applicability	High	High	Avg	Avg	-	High	Avg	-	High	Avg	High	Avg
Importance	High	High	High	High	Avg	High	Avg	Low	Low	Avg	High	Mid
Sensor Status	High	High	Avg	Low	Avg	Avg	Avg	Low	High	Avg	High	High
Severity	High	High	High	Avg	Low	Avg	-	Low	Avg	-	Avg	-
Weaknesses	High	Avg	High	-	Avg	Low	-	Low	-	Low	-	Low
Relationship	High	High	Avg	Avg	Avg	-	Low	-	Low	Low	-	Mid
Social Activity	High	Avg	High	Avg	-	Low	-	Low	-	Low	Low	-
Alert Score	High	High	High	Avg	Low	High	Avg	Low	Avg	High	Low	Mid

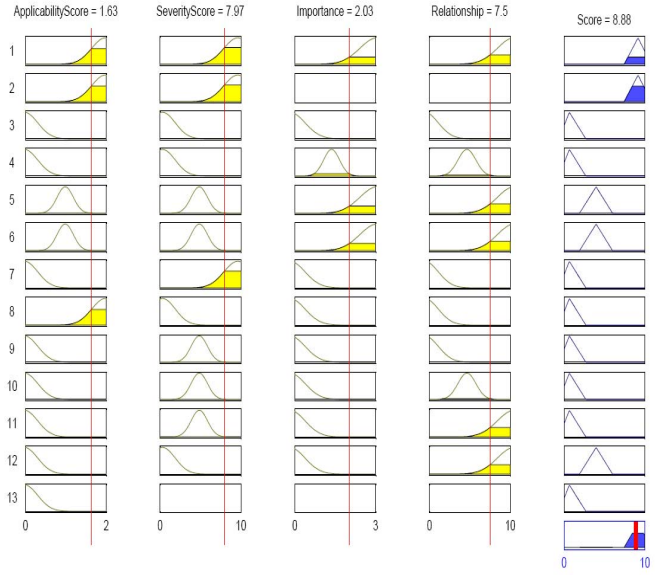

Fig. 4. Portion of Fuzzy Logic Inference System

in that it does not require a deep understanding of the system, exact equations, or precise numeric values. It incorporates an alternative way of thinking, which allows for complex modeling of systems using a high-level of abstraction of gained knowledge and experience. Fuzzy logic allows the expression of qualitative knowledge, including phrases such as "too hot" and "not bad", which are mapped to exact numeric ranges.

For the above reasons, we used a Fuzzy logic system to reason about IDS alerts. Results coming from the metrics presented in the previous section are used as input to Fuzzy logic Inference engine in order to investigate the seriousness of the generated alerts. The Fuzzy logic system requires a definition of the membership functions of all input metrics. In addition, fuzzy rules need to be defined in order to formulate the conditional statements that make the fuzzy inference. There are five parts of the fuzzy inference process: (1) fuzzification of the input variables, (2) application of the fuzzy operator (AND or OR) to the antecedent, (3) implication from the antecedent to the consequent, (4) aggregation of the consequents across the rules, and (5) defuzzification.

Membership Functions (MF) are curves that define how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. We use Gaussian distribution as the type of the membership functions.

Rules are defined by domain expert as shown in Table 4. Rules in a fuzzy expert system are usually of a form similar to:

$$\text{if } Applicability \text{ is } High \text{ and } Severity \text{ is } Avg \text{ then set } Alert\ Score \text{ to } High$$

where $Applicability$ and $Severity$ are input variables, $Score$ is an output variable, $High$ is a membership function (fuzzy subset) defined on $applicability$, Avg is a membership function defined on $severity$, and $High$ is a membership function defined on the alert $score$.

In our approach, we use Fuzzy logic to score the alerts generated by the different IDSs. As shown in figure 4, the Fuzzy logic Inference system first takes the input values from the metrics (e.g., applicability, severity, importance, and relationship metrics) and then fuzzifies them using the membership functions. Then, the rules will be evaluated to generate the output set for each active rule. All the outputs will be aggregated and a single fuzzy set will be provided. Then this fuzzy set will be defuzzified in order to give a numeric value that represents the seriousness of the alert.

VI. ALERT RESCORING

The goal of the alert rescoring technique is to score alerts that have already been scored based on their relationship with the current alert. One of the reasons for rescoring alerts is to notify security administrators of the early steps of the attack, which may have received low scores. Another reason is to emphasize preliminary activities of an attacker who is launching a critical attack. Scoring alerts only once limits the identification of the non-critical early-steps of an attack, especially if a filtering technique is used to discard low-score alerts. Also, high-level alert management techniques such as correlation or clustering can benefit from rescoring and provide more accurate results.

Alert management techniques such as aggregation, grouping, scoring, filtering, clustering, correlation, and fusion were proposed to deal with and abstract large numbers of alerts. First, alerts are aggregated from multiple IDSs then similar alerts are grouped together into hyper-alerts. Scoring function evaluate the hyper-alerts and assign a score to each one according to its importance. Low-score alerts are discarded and do not get involved in any further analysis. Correlation functions may then be applied to present the attack scenarios. Scoring

TABLE III
PRIORITIZED ALERT OF DARPA 2000 LLDOS 1.0 DATASET

Phase	Alert Name	Without grouping			With grouping		
		# Alert	# Prioritized Alert	Alert Score	# Alert	# Prioritized Alert	Alert Score
Phase 1	ICMP Echo Request	786	0	$\simeq$2	29	0	$\approx$2
	ICMP Echo Reply	30	0	$\simeq$1.5	12	0	$\simeq$1.5
Phase 2	RPC portmap sadmind request UDP	250	79	$\simeq$8	46	28	$\approx$8
	RPC sadmind UDP PING	9	6	$\simeq$5	4	3	$\simeq$5
Phase 3	RPC sadmind with root attempt UDP	46	28	$\simeq$9	40	29	$\simeq$9
	RPC sadmind UDP NETMGT_PROC_SERVICE	46	6	$\simeq$9	3	3	$\simeq$9
Phase 5	BAD-TRAFFIC loopback traffic	141	141	$\simeq$9	1	1	$\simeq$9
False Alert	NETBIOS NT NULL session	2	0	$\simeq$1	1	0	$\simeq$1
	ATTACK-RESPONSES Invalid URL	4	0	$\simeq$1	1	0	$\simeq$1
	SNMP request udp	12	0	$\simeq$6	9	0	$\simeq$6
	SNMP public access udp	6	0	$\simeq$5	6	0	$\simeq$5
	ATTACK-RESPONSES 403 Forbidden	10	0	$\simeq$1	3	0	$\simeq$1
	MS-SQL version overflow attempt	1	0	$\simeq$1	1	0	$\simeq$1
	ICMP redirect host	2159	0	$\simeq$1	26	0	$\simeq$1
Total		3502	260 (92.57% Reduction)		182	64 (64.83% Reduction)	

alerts once will discard early non-critical attacks that prepare for later critical attacks. Consequently, the early steps of the attacker are not involved in any further analysis such as attack scenario construction. For instance, if attackers first scan the victim machine by launching an $IPSweep$ attack. This probe will be scored low according to its seriousness and impact. Later, the attackers launch a $SadmindBufferOverflow$ attack based on the vulnerability findings of the scanned machine. This attack will be assigned a high score since it is a critical attack. The security administrator can not see the early steps of the attack if a filtering operation is applied. On one hand, involving only critical alerts in the high-level operations, such as correlation, prevents the non-critical early steps of the attack from being considered. On the other hand, involving all alerts in the high-level operation leads to an overwhelming number of correlated alerts that is difficult to manage. Hence it is important to highlight only the critical alerts and those related to them.

We perform a rescoring of alerts based on the $prepare\text{-}for$ relationship and the trustfulness of attacks. The $prepare\text{-}for$ checks if there is any relationship between the currently evaluated alert and the alerts in the alerts log. The trustfulness examines the previous activities of the current source of the attack if the launched attack is critical. Accordingly, we adjust the value of trustfulness for this source in our source evaluation metric.

VII. EXPERIMENTAL RESULTS

In order to to validate the effectiveness of our proposed approach, we used the DARPA 2000 specific intrusion detection scenario dataset (LLDOS 1.0 dataset) [15]. We also used Snort to scan and reply to the LLDOS 1.0 dataset file. Alerts generated by Snort were stored in a MySQL database. To examine each alert, we wrote a Java program that accesses the database through MySQL Connector/J. Finally, we used Matlab Fuzzy logic toolbox [25] to score each alert.

The DARPA LLDOS 1.0 dataset contains the traffic collected from the DMZ and the $inside$ part of the evaluation network. The series of attacks in the dataset are carried out over multiple sessions. These sessions start with scanning the network in order to launch a DDOS attack against an off-site server. The sessions can be grouped into the following five phases:

1. Scan the network (IPsweep)
2. Look for the sadmind daemon of live IP's
3. Exploit the sadmind vulnerability
4. Install an mstream trojan
5. Launch the DDoS attack

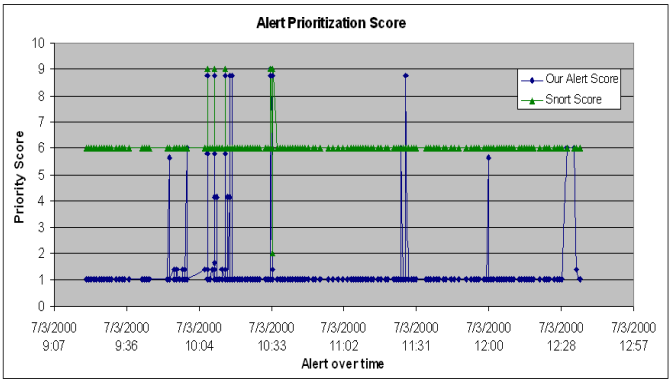

Fig. 5. Alert Prioritization Scores

With its maximum detection capability Snort was used to scan and detect intrusions within the binary tcpdump file of both of the $inside$ and DMZ traffics. Snort reported 3502 alerts (321 inside, 3181 DMZ). Since we are focusing on alert scoring and prioritization instead of clustering, we only grouped similar alerts that were very close in time in order to remove redundant alerts. This resulted in a new total of 141 alerts.

We applied our scoring technique to the alerts generated by Snort. For each alert, we compute the value of all the metrics we defined earlier, except for the sensor status, service vulnerability, and social activity metrics because the used dataset does not provide knowledge about the status of the targeted services and applications of the evaluation network. However, the other metrics were good enough to prioritize the most critical alerts. The attacker in the first phase tries to scan

the network by employing the ICMP echo-request, looking for "up" hosts. Snort generates 816 alerts as a response to attacker's ICMP requests and the hosts ICMP replies. Our technique evaluated these antecedents and scored them as low (1.2-2.3) as shown in Table III. In the second phase, we received 259 alerts from the traffic of both the DMZ and $inside$ parts, which represents the attacker's attempts to probe the discovered live hosts from the previous phase to determine which hosts are running the $sadmind$ remote administration tool. We scored these alerts differently based on the context in which they occurred. For instance, the "$RPC\ portmap\ sadmind\ request\ UDP$" alert that was triggered by the activity targeting the inside firewall interface is scored low. However, this alert is scored high when the target host is running a $sadmind$ service. The remote-to-root exploit has been tried several times in the third phase and Snort raised 92 alerts of which we prioritized 34. Since we focus on evaluating alerts generated by Network-IDSs, we did not involve the audit data from the hosts in the network and, therefore, phase 4 was not included. The DDOS attacks in phase five triggered 141 alerts which we prioritized as critical events.

Table III summarizes the results of our alert scoring (with and without the grouping function) technique on the DARPA 2000 dataset. Our IDS alert prioritization was effective in identifying the false positive alerts which Snort failed to detect. For example, Snort generates a "$MS\text{-}SQL\ version\ overflow\ attempt$" alert with the highest priority, but we scored this alert low based on our criteria since the target address is running a Mac operating system and this attack is impossible to succeed in this context. Figure 5 shows that after we score the alert, a security administrator can be provided with the most important alerts unlike the result of Snort which assign a level-two priority (out of 3) to most of the alerts. We also applied our rescoring technique to the alerts that are scored previously. Since the LLDOS 1.0 dataset consists of only one complete attack scenario; the first phase, which contains non-critical attacks (regular scanning), is targeted for rescoring. Our technique rescored the first phase of this scenario since it was a preparation step for the DDoS attack. As a result, all the critical alerts as well as the preparation steps have been prioritized and presented to security analyst.

VIII. Conclusion and Future Work

In this paper, we presented a method that evaluates IDS alerts based on a number of criteria. We used a Fuzzy logic Inference mechanism in order to score alerts. The score represents the seriousness of the alerts. Furthermore, we developed a rescoring technique that enabled us to rescore alerts to show the early steps of the attackers. We applied our approach to the alerts generated by scanning DARPA 2000 LLDOS 1.0 dataset and we successfully prioritized the most critical alerts along with their preparation steps.

There are a number of directions for future work. First, we plan to apply the proposed approach to heterogeneous IDS. Second, we intend to investigate anomaly-based IDS alerts. Using additional alert relationship criteria such as the use of prerequisite consequence and/or predefined scenarios techniques. Also, providing an analytical study of the alert prioritization, rescoring mechanism, and the fuzzy logic rules and inference engine is an appropriate extension to this research.

Acknowledgment

The authors are thankful to the ministry of higher education in Saudi Arabia (www.mohe.gov.sa/english) for providing a scholarship to the first author of this paper. The authors are also thankful to Dr. Issam Aib for his valuable comments that helped improving this paper.

References

[1] S. AXELSSON, *The Base-Rate Fallacy and the Difficulty of Intrusion Detection*, ACM Transactions on Information and System Security **3** (2000), no. 3, 186–205.
[2] F. Cuppens and R. Ortalo, *LAMBDA: A language to model a database for detection of attacks*, Proc. of Recent Advances in Intrusion Detection (RAID 2000) (2000), 197–216.
[3] D. Curry and H. Debar, *Intrusion detection message exchange format(idmef)*, IETF, 2007.
[4] H. Debar, M. Dacier, and A. Wespi, *Towards a taxonomy of intrusion-detection systems*, COMPUT. NETWORKS **31** (1999), no. 8, 805–822.
[5] Herv Debar and Andreas Wespi, *Aggregation and correlation of intrusion-detection alerts*, Recent Advances in Intrusion Detection, 2001.
[6] Jinqiao Yu et al, *Trinetr: An intrusion detection alert management system*, WETICE '04 (Washington, DC, USA), 2004.
[7] Muhammad Abedin et al, *Vulnerability analysis for evaluating quality of protection of security policies*, QoP '06: Proceedings of the 2nd ACM workshop on Quality of protection, 2006.
[8] Zhichun Li et al, *Towards scalable and robust distributed intrusion alert fusion with good load balancing*, LSAD '06', 2006.
[9] B. Feinstein and G. Matthews, *The intrusion detection exchange protocol (idxp)*, IETF RFC, 2007.
[10] http://insecure.org/nmap/.
[11] http://www.nessus.org/nessus/.
[12] http://www.securityfocus.com/archive/1, *Bugtraq*.
[13] http://www.securityfocus.com/vulnerabilities.
[14] K. Julisch, *Clustering intrusion detection alarms to support root cause analysis*, 2003.
[15] MIT Lincoln Lab, *2000 darpa intrusion detection scenario specific datasets*, MIT, 2000.
[16] Wang Li, Li Zhi-tang, Lei Jie, and Li Yao, *A novel algorithm sf for mining attack scenarios model*, icebe **0** (2006), 55–61.
[17] P. Meil, T. Grance, et al., *NVD national vulnerability database*.
[18] P. Ning, Y. Cui, and D. Reeves, *Constructing attack scenarios through correlation of intrusion alerts*, 2002.
[19] P.A. Porras, M.W. Fong, and A. Valdes, *A Mission-Impact-Based Approach to INFOSEC Alarm Correlation*, Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002) (2002), 95–114.
[20] Phillip A. Porras, Martin W. Fong, and Alfonso Valdes, *A mission-impact-based approach to infosec alarm correlation*, RAID, 2002, pp. 95–114.
[21] Xinzhou Qin and Wenke Lee, *Statistical causality analysis of infosec alert data*, RAID, 2003, pp. 73–93.
[22] M.J. Ranum, *False Positives: A Users Guide to Making Sense of IDS Alarms*, ICSA Labs IDSC, white paper, 2003.
[23] Secunia-Vulnerability and virus information, *http://secunia.com*.
[24] S. Staniford-Chen, B. Tung, and D. Schnackenberg, *The common intrusion detection framework (cidf)*, Information Survivability Workshop, Orlando FL, 1998.
[25] Fuzzy Logic Toolbox, *http://www.mathworks.com/products/fuzzylogic*.
[26] A. Valdes and K. Skinner, *Probabilistic Alert Correlation*, RAID 2001, Springer, 2001.
[27] MITRE Common Vulnerabilities and Exposures (CVE), *http://cve.mitre.org/*.
[28] www.snort.org.