A Fuzzy-logic based Alert Prioritization Engine for IDSs:

Architecture and Configuration

by

Khalid Ateatallah Alsubhi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2008

Abstract

Intrusion Detection Systems (IDSs) are designed to monitor a networked environment and generate alerts whenever abnormal activities are detected. The number of these alerts can be very large making their evaluation by security analysts a difficult task. The management is complicated by the need to configure the different components of alert evaluation systems. In addition, IDS alert management techniques, such as clustering and correlation, suffer from involving unrelated alerts in their processes and consequently provide results that are inaccurate and difficult to manage. Thus, the tuning of an IDS alert management system in order to provide optimal results remains a major challenge, which is further complicated by the large spectrum of potential attacks the system can be subject to.

This thesis considers the specification and configuration issues of *FuzMet*, a novel IDS alert management system which employs several metrics and a fuzzy-logic based approach for scoring and prioritizing alerts. In addition, it features an alert rescoring technique that leads to a further reduction of the number of alerts. We study the impact of different configurations of the proposed metrics on the accuracy and completeness of the alert scores generated by FuzMet. Our approach is validated using the 2000 DARPA intrusion detection scenario specific datasets and comparative results between the Snort IDS alert scoring and FuzMet alert prioritization scheme are presented. A considerable number of simulations were conducted in order to determine the optimal configuration of FuzMet with selected simulation results presented and analyzed.

Acknowledgements

All praise is to Allah for giving me the ability and knowledge to finish this thesis. I would like to express my deep gratitude to my supervisor Professor Raouf Boutaba for his guidance, support, patience, and encouragement. He was always available, professional, and extremely nice. The opportunity to work with Professor Raouf was as a valuable award from University of Waterloo to me.
Special thank is also due to Professor Ehab Al-Shaer for his guidance and comments throughout this research.
I would also like to thank Professor Ihab Ilyas and Professor Pin-Han Ho for serving as reviewers of this thesis and for their valuable comments.
I would also like to thank Issam Aib for his productive discussions, feedback, and reviews throughout this research. The continuous encouragements from Nizar Bouabdallah and Rami Langar are highly appreciated.
Special acknowledgment is to the Saudi Arabian Cultural Bureau in Canada for their financial support to complete my Master's degree. High appreciation is to Prof. Ahmed Mitwalli, and Dr. Yeihya Al-Khazraj for their support and encouragement.
Last but not least, I am forever indebted to never forgotten people **my parents**, and siblings for their encouragements, continuing support, prayers and love.

To my parents...

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Intrusion Detection Systems

Network attacks are growing more serious, forcing system defenders to deploy appropriate security devices such as firewalls, Information Protection Systems $IPSs$, and/or Intrusion Detection Systems ($IDSs$). $IDSs$ are either software or a hardware system, whose purpose is to inspect user and/or network activity by looking for suspicious activities that violate the system security policy. Suspicious activities could be caused by an attacker's attempt to access the systems from the Internet, authorized users trying to gain additional privileges for which they are not authorized, and authorized users who misuse the privileges given to them. In each case, IDS generates alerts notifying the security analysts about the anomalous incidents in order to take appropriate action.

IDSs can be categorized into two main classes based on what type of data they are inspecting namely, Network-based and Host-based IDS [9]. The Network-based IDS (NIDS) monitor the traffic transmitted from/to all devices on the network based on its place. This type of IDSs deals only with the packets data type to find intrusions. Several advantages are offered by NIDS. For example one NIDS can protect a large network base and deploying NIDS does not interfere with the normal operation of a network [3]. However, drawbacks of NIDS include dropping packets when monitoring/analyzing fast/busy links, inability to analyze encrypted information, and the inability to verify the success of the attack [33]. The second class of IDSs is the Host-based IDS (HIDS) which analyze the activities within the host such as operating system audit trails and system/application logs. Unlike the NIDS, HIDS can observe the outcome of an attempted attack. However, HIDS can be disabled easily by attacker who compromises the host. Generally, NIDS and HIDS are complimentary to each other, since they are detecting different type of attacks.

Nevertheless, there are two common types of IDSs based on their method of inspecting the traffic: signature-based and anomaly-based [9]. The signature-based IDS generates an alert when the traffic contains a pattern that matches signatures of malicious or suspicious activities. In turn, the anomaly-based IDS examines ongoing activity and detects the attack based on the variation from normal behavior. However, both types suffer from a common 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.

1.2 Alert Management in Intrusion Detection Systems

After deploying IDS, handling alerts that are generated is the first task that the security analyst should do. The challenge of IDS is not only by detecting intrusions but also by managing alerts. Since the number of these alerts can be very large and mixed with high false positive rate; making the management task of security analysts difficult to handle which accordingly he/she could not identify the real attacks from the false ones reported by the alerts. Several methods can be applied to manage alerts effectively. Reducing the large number of alerts is of alert management. Also, differentiating between legitimate alerts and false alerts is another task that security administrator should accomplish in order to determine whether the actual attack is occurring. Since attackers reach the final goal by launching their attacks in multiple steps, building the attacker's scenarios can be achieved by alerts management techniques. Generally, alert management techniques provide an effective ways to help the security administrator to evaluate and manage alerts, thereby saving his or her time and effort.

There are two general alert management classes namely low-level and high-level. Dealing with each alert individually to enrich its attributes or assign scores based on potential impact can fall into the low-level alert operations. high-level alert management techniques deal with a group of alerts and provide an general view of these alerts (i.e. such as aggregation, clustering, correlation, and fusion). Low-level and high-level alert management techniques are complementary to each other. First alerts need to be enriched by the low-level operation and then the high-level proceeds the enriched alert to provide more accurate results.

1.3 Motivation

IDSs usually generate a large number of alerts whenever abnormal activities are detected [14]. Inspecting and investigating all reported alerts manually is a difficult, error-prone, and time-consuming task. On the other hand, ignoring alerts might lead to successful attacks. Dealing with this problem is a challenging task which involves two alert evaluation phases: low-level and high-level. The low-level alert operations deal with each alert individually to enrich its attributes or assign a score to it based on potential risk. High-level alert management techniques, such as aggregation, clustering, correlation, and fusion, were proposed to deal with a set of alerts and provide an abstraction of these alerts. However, the high-level techniques suffer from including alerts that are not significant which consequently leads to inappropriate results. Therefore, low-level evaluation techniques are needed to examine large number of alerts automatically (or semi-automatically) and prioritize them, leaving only important alerts for further inspection and investigation. Accordingly, the reduced set of alerts leads to more precise high-level alert management results such as correlation and clustering. The goal of this work is that the security administrator will be provided with an effective technique to evaluate and manage alerts, thereby saving his or her time and effort.

In addition, it is not only sufficient to propose new IDS alert management systems, yet there is a need to study the degree of efficiency of these systems depending on different configuration sets. The tuning of an IDS alert management system in order to provide optimal results remains a major challenge. This hurdle is further complicated by the large spectrum of potential attacks the system can be subject to. There is also a need to consider whether there exist a unique optimal configuration which works all the time or whether this optimal configuration changes depending on system state and administrative policies.

1.4 Contributions

The contributions of this thesis can be summarized as follows:

1. We proposed FuzMet, an alert evaluation and prioritization framework, by addressing the limitations of previous works that deal with alert ranking [31,23,32,46]. Unlike the previous works, the use of new metrics such as, the sensor sensitivity, relationship between alerts, services stability, and social activity between source and target, allows us to accurately evaluate the alerts generated by either a signature-base or anomaly-based IDS. In addition, The automatic evaluation and prioritization processes make the management task for the security analysts easier and controllable.

2. A re-scoring technique that dynamically scores alerts based on the relationship between attacks or the level of maliciousness of attackers.

3. We applied Fuzzy-logic Inference approach as a reasoning technique to quantitatively score each alert based on the values of the metrics defined earlier. To the best of our knowledge, we are the first to use Fuzzy-logic for evaluating IDS alerts.

4. A comprehensive study of the impact of different configurations of the proposed metrics on the accuracy and completeness of the alert scores generated by FuzMet.

5. A survey and taxonomy of IDS alert management techniques.

1.5 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 reviews the background and provide literature survey of alert management techniques in IDS including alert prioritization approaches. Chapter 3 describes our proposed FuzMet alert prioritization scheme along with the metrics and the Fuzzy-logic approach that we used to score IDS alerts. Chapter 3 also investigates metrics configuration issues. Simulation results are presented and discussed in Chapter 4. Finally, Chapter 5 concludes this thesis.

Chapter 2

Background and Literature Survey

2.1 Why Alert Management

An alert in intrusion detection systems is typically a formatted message describing a circumstance relevant to network/host security that is derived from critical events [44]. Generated alerts are presented to the security administrators or some automatic control system. This allows to notify the security administrator about the abnormal activity that has been detected, and consequently take the proper action against the attack. Most of the time, security analysts are overwhelmed by the large number of alerts generated by IDS(s), as well as the high rate of false positives [29,19]. As a result, security administrators cannot easily distinguish between the alerts generated from legitimate traffic and the ones generated from suspicious traffic by the information provided by the alerts. In this content, alert management techniques were proposed to assist security administrators in better understanding the state of the network/host under attack. For instance, IDS alert prioritization techniques aim to score each alert based on its seriousness and impact and prioritize the high scored alert, which consequently reduce the overall number of alerts that are presented to security administrators. Alert correlation is another example of an IDS alert management technique, where the steps of the attack are linked together and the attack graph is constructed. Alert correlation helps the security administrators to understand the steps of an attack from its first step to its final goal. Generally, alert management techniques are essential steps after detecting the attack by the IDS.

2.2 IDS Alert Representation- IDEMF

Alert representation is the method that allows the IDS to provide the information about an attack to the security administrators. The alert representation format varies from one IDS product to another based on the strategy of each IDS to present the attack. Since each IDS has it own representation format, it becomes difficult to deploy different IDS products in the same network or to exchange data between heterogenous IDSs. Therefore, standard alert representation formats have been proposed to deal with the above problem.

The Intrusion Detection Message Exchange Format (IDMEF) is a standard format(RFC 4765) developed by Curry and Debar in the Intrusion Detection Working Group (IDWG) [1]. This standard defines a data format for describing the suspicious events that can be reported by the IDS [8]. IDMEF is an XML based specification for an intrusion alert format. IDMEF is used between an IDS and the manager to which it sends alerts. The IDMEF data model is an object-oriented representation of the alert.

Under the top-level class of the IDMEF-message, there are two types of messages; Alerts and Heartbeats. For the Alert message subclass, low-level subclasses are used to present detailed information about the event that has been detected by the IDS, as shown in fig 2.2. Whenever an IDS detects an intrusion, it generates an alert message class to its manager(s), which contains several subclasses. The *Analyzer* subclass carries unique information about the sensor that generated the alert. Three time subclasses are provided in the alert message. The time when the analyzer generated the alert is described in the *Createtime* subclass. Attack detection time is shown in the *Detecttime* subclass. The current time on the analyzer is describe in the *AnalyzerTime*. The source and the target of the event leading up to the alert are represented in the *Source* and *Target* subclasses, respectively. The *Classification*

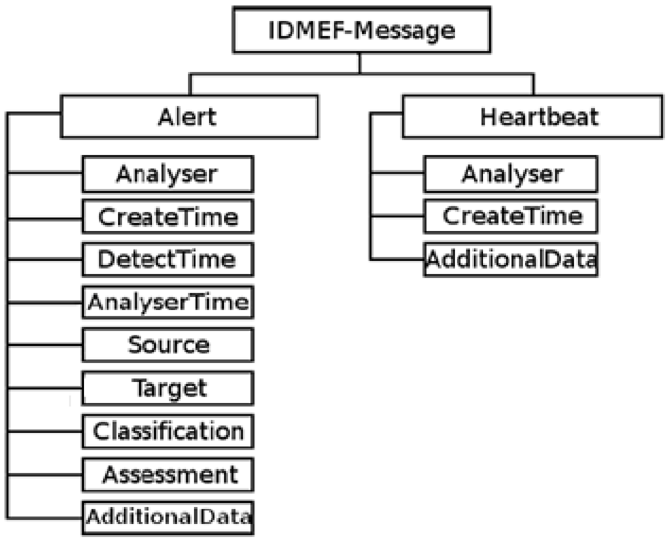


Figure 2.1: The IDMEF data model

subclass carries information about the name of the alert (attack) and some references that allow the IDS manager to obtain more information about the reported attack. The impact of the event, response actions taken by the analyzer, as well as the analyzer confidence can be described in the *Assessment* subclass. The last subclass in the alert message is the *AdditionalData* which can be used for presenting any information that does not fit into the data model.

For the second class of an IDMEF message, the *Heartbeat* messages class is triggered by the IDS to be sent periodically to show its current status. Several advantages are offered by the design of IDMEF format including:

- Flexibility to accommodate the needs of different IDSs, since one may wish to deliver more or less information about certain types of attacks.

- Capability of presenting messages generated by both network and host based sensors.

- The ease of extensions, either by using AdditionalData objects or by defining new object types.

However, IDMEF also has some disadvantages, such as:

- The format limits the semantic representation (i.e. two IDSs can name the attacks differently) [45].
- Difficulties in creating an equivalent objectoriented representation in a relational databases such as SQL.

2.3 IDS Alert Management Approaches

Alert management functions receive the alerts from IDS(s) and process them in order to make them suitable for human control. The general purpose of these techniques is to help the security administrator to fully understand what the IDS is addressing. In other words, all alert management techniques such as, alert fusion, alert aggregation, alert clustering, and alert correlation provide some form of high-level analysis and reasoning capabilities beyond low-level sensor abilities [34]. For instance, security analysts can be flooded by a large number of alerts in a short period of time, making the task of managing these number of alerts manually difficult; therefore, reduction techniques are needed. Another example for showing the need of alert management techniques can be seen in enterprise networks, where security administrators can deploy heterogeneous IDSs in different places; therefore, aggregating the alerts that have some common features together in a centralized place is the task of the aggregation function. In the following subsections we will briefly describe the most common IDS alert management techniques.

2.3.1 Alert Reduction

One of the biggest problems associated with IDS is the number of alerts. IDSs usually generate large number of alerts whenever abnormal activities are transmitted from/to the protected network and/or hosts. It is common that an IDS reports 10-200 alerts per day and this number increases when more than one IDS is deployed [24]. These number of alerts can easily overwhelm the security analysts who manage the IDSs in the network. Accordingly, the task of managing the alerts becomes very difficult; therefore, the security analysts will not fully understand the reported alerts, so that action against the detected attack will be taken. Therefore, techniques for reducing the number of alerts generated by IDS(s) are needed. Several alert management techniques were proposed to deal with this problem from different angles, but all of them fall into reducing the number of alerts. In this section, we first categorize the proposed approaches of decreasing the number of alerts by their procedure, and then describe each process individually providing the state-of-the-art approaches that exist in the field. However, reducing the number of alerts can be done either before the IDS raises the alerts or after the alerts are generated. Tuning the IDS, placing it correctly, or configuring it efficiently are examples of the techniques that deal with reducing the number of IDS alerts before they are generated, while, the techniques that treat the raised alerts include alert merging, and clustering. The reduction techniques can be described as follows:

Alert Aggregation: When IDS(s) generates the alerts, aggregation techniques try to group a set of alerts together that have some common characteristics, such as the source, the target, and the type of the attack. In the case of dealing with one IDS or multiple homogenous IDSs, these techniques can be applied easily sice the alerts format is the same. However, these methods become more complicated when heterogenous IDSs are deployed. Generally, these techniques relieve the security analysts from dealing with each alert individually. Debar and Wespi presented an aggregation algorithm that is used in the design and implementation of an intrusion-detection console built on top of the Tivoli Enterprise Console (TEC) [10]. In this approach, alerts are aggregated into so-called "situations", which are a set of alerts that share certain characteristics. Three alert attributes were involved to form the "situation", which are the source, the target, and the class of the attack.

Alert Merging: As we discussed earlier, the IDS generates too many elementary alerts that confuse the security administrators. Consequently, alerts need to be grouped together to form a global alert that gives the analyst a better understanding of the intrusion as described in the aggregation function. Alert merging techniques aim to combine a group of alerts into one hyper-alert that represents the abstract view of the underlaying alerts. For instance, an attacker performs a scan attack all hosts' ports in one subnet, seeking an open-vulnerable port for a further attack. As a result, IDS will generate too many alerts related to these scan attacks. However, generating one global alert informing the security administrators that the attacker is scanning all the sub-network will save the administrator effort, time, and most importantly he will take the appropriate action against the seriousness of the attack. Yu et al proposed a merging process that is applied to a group of alerts (cluster) that produce a synthesized alert [46]. Since alerts can be received from multiple IDSs, conflict might occur between alert attributes. They use a voting algorithm to solve the conflict by considering the dominant characteristic of the alerts participating in the conflict. Cuppens proposed another merging technique that uses expert rules to specify how the global alert is derived from a set of alerts [7]. In this method, four alert attributes are involved in the merging process namely, source, target, time, and classification. Conflict resolution is applied by specifying an integrity constraint that should not be violated. If so, the conflicting information will be provided in the additional data attribute of the global alert. Generally, merging techniques attempt to create a new alert as a representation of various of alerts that belong to same attack.

Alert Clustering: IDSs generate a number of similar alarms that represent one attack. Similar alarms can be grouped together to reduce the total number of IDS alerts. Alert clustering is a technique that groups alerts that share common features, such as source/target IP address or port number into one cluster. Each cluster contains alerts that share similar attributes. Several approaches have been proposed to cluster the alerts generated by IDSs. K. Julisch proposed a clustering technique based on forming a generalized view of false alarms [19]. The objective of this clustering technique is to discover root causes of the false positive alerts. In this research, Julisch discovers that 90% of the alerts correspond to a small number of root causes. Identifying and removing these root causes leads to a 82% reduction of the total number of alerts, making the IDS analysts focus on the alerts of the real attack. Another clustering technique was proposed by Cuppens [7] as part of the MIRADOR project [26]. First, alerts are transformed from XML format into rational database format. Then, an expert system is used to decide when a new alert can join (or form) the cluster. This decision is based on the similarity requirement defined by expert rules.

Network-based IDS Placement: Security administrators have full flexibility to deploy network-based IDSs in any place in their network where they can perform a better detection. However, the place of a network-based IDS can be one of the reasons that a large number of alerts are produced. For instance, one can deploy the IDS to monitor the traffic before it is filtered by the firewall to detect all attack trails, regardless of whether they are stopped by the firewall [5]. In this scenario, the IDS will generate more alerts than if it was deployed behind the firewall. In general, the place of the network-based IDS can be a trade-off between maximizing the protection and minimizing the number of alert generation.

Deactivating or Improving IDS Signatures: In signature-based IDSs, all known attacks are stored as signatures which have to be matched within the traffic. Disabling unnecessary signatures that detect normal traffic as abnormal can potentially reduce the number of alerts. For instance, if the system administrator allows remote login to the system while the IDS raises an alert whenever this service is used, then disabling the signature that is related to the remote access service will reduce the number of alerts generated by the IDS. Also, knowledge of the system environment allows the system administrator to disable any unrelated signatures from the database. For example, employing an IDS for Windows environments requires the system administrator to disable any signature related to the Unix environment. Nonetheless, in signature based IDSs, security experts write vulnerability signatures to accurately match the exact attack pattern without making any false positives. However, poorly written signatures can lead to false positives if legitimate traffic unexpectedly matches the attack signature [5]. Therefore, the quality of the attack signature has an effect on the number of generated alerts. Poor signatures need to be rewritten and improved to accurately represent the pattern of only the real attack. As a result, the number of alerts will be reduced to the detection of the actual attack.

Tuning the anomaly-based IDS: As we mentioned in section 1.1, the anomaly-based IDS monitors the traffic and raises alerts whenever a traffic varies from the norm by a ceratin threshold. The number of alerts in this type of IDS can be very large compared with the signature-based IDS. However, the number of alarms can be reduced by tuning the anomaly-based IDS. Tuning the anomaly-based IDS can be done by setting the threshold to a proper value so that the number of alarms will be reduced. However, adjusting the threshold value to reduce the number of alerts can either increase or decrease the detection. When the detection rate increases the number of false negative alerts increases [5]. But, when the detection rate decreases, then the rate of false positive alerts increases. Therefore, there is a trade-off between the number of alerts and the detection capability in the anomaly-based IDS.

2.3.2 Alert Correlation

Generally, correlation can be defined as the method of finding the relationship between two objects or series of objects. Specifically, the process of discovering the connection between different series of security events is defined as the alert correlation process. Attackers usually launch their attacks in multiple steps to achieve the final goal. Consequently, traditional IDSs focus on low level attacks and generate alerts for each one individually. In this case, security analysts face difficulties in manually checking the alert log to find the relationship between the attacks, since the number of alerts is large and the rate of false positives is high. Therefore, alert correlation techniques become essential techniques in order to uncover the relationship between alerts, and to construct the corresponding attack scenarios. Researchers proposed several techniques that deal with this problem which fall into three classes. The first class includes correlating alerts based on the similarity of the attributes, such as IP addresses, ports etc. The second class is based on specifying a known sequence of attacks. The third class is based on the dependencies between alerts by matching prerequisites with the consequences of attacks. The three alert correlation classes will be described in more detail as follows:

Correlating alerts based on similarity: An alert is usually divided into a number of attributes such as, source IP address, target IP address, port number for the source and the target, time, and attack type. In this class, alerts will be correlated together if their attributes are similar. Similarity functions are computed on the alerts attributes to make the correlation decision. In fact, these techniques promise to correlate some alerts that are sharing some features such as the source and target IPs. However, they fail to uncover the casual relationship between alerts. Valdes and Skinner developed a probabilistic alert correlation method aiming to correlate multiple attack steps and build the corresponding attacks scenarios [37]. Their probabilistic approach suggested a unified mathematical framework with appropriate similarity functions for each appropriate alert feature. Alerts will be correlated if the results of the similarity functions are closely matched, subject to meet a minimum degree of matching which is controlled by both a single configurable parameter and the weighted average of similarity values over the overlapping features. Any new alert will be merged with an existing meta alert as long as they result in the highest similarity values and pass the specified threshold.

Attack scenario predefinition: In this type of correlation method, alerts will be correlated based on the known attack scenarios. Attack scenarios can be defined by security experts or learned through training datasets. The alerts sequence will be compared with the known attack scenario in order to correlate them together and construct the detected attack steps. These methods can discover the causal relationship between attacks, but they are restricted to known attack strategies.

The dependencies between attacks: Most alerts are related to different alerts, since the early steps of attack prepare for later ones. Based on this observation, the connections (or relationship) between these alerts can be used to construct an attack scenario. Several techniques have been proposed to construct an attack plan by correlating alerts based on the prerequisites (what makes the attack successful) and the

consequences (what are the outputs of the attack) of each attack (corresponding alert). Two attacks will be correlated if any of the prerequisites of the later attack match any of the consequences of the early one. For these techniques to work, specific knowledge about the attacks are required to identify their prerequisites and consequences. These techniques promise to discover the casual relationship between alerts, and they are not restricted to known attack scenarios. However, they have a common weakness, that is, they fail to correlate unknown attacks (that they have no specific prerequisites and consequence). Ning et al. use logical predicates to model the alert as prerequisites and consequences of attacks [28]. The idea of the "hyper-alert" was introduced to represent every type of attack. A hyper-alert consists of facts, prerequisites, and consequences. A fact is a set of names (for attributes). Prerequisites and consequences form a logical combination of predicates where its variable can be found in fact. For instance, consider the buffer overflow attack against the remote administrator to "Sadmind." The type of hyper-alert will be:
$SadmindBufferOverflow = ((VictimIP, VictimPort), (ExistHost(VictimIP) \bigwedge VulnerableSadmind(VictimIP)), (GainRootAccess(VictimIP)))$

2.3.3 Alerts Visualization

IDSs report the alerts to the security analysts in row format stored in a database. Understanding the general view of what these alerts are addressing is difficult to achieve from looking at the alerts records. Therefore, alert visualization techniques were proposed to deal with this issue. These techniques will take the row alerts and represent all alerts visually. This visual representation can help the network administrator to have an overall picture of what is occurring in the network. For instance, visualizing all events targeting one specific host can cut down the time for the analyst to understand the reported situation. As a result, security analysts can efficiently analyze a graphical layout of alert logs easier and faster than analyzing alert textual logs. Several approaches have been proposed to visualize the alerts of an IDS. One example of the existing tools that visualize Snort IDS alarm log is SnortView [20].

2.3.4 Alert Scoring & Prioritization

Inspection devices, such as IDS, present attacks to a security administrator through alerts. The IDS often generates a large number of alerts. However, these alerts should not be treated equally since their importance and impact are different. Therefore, alert scoring and prioritizing techniques are needed to determine the important alerts to a security administrator for further action. Indeed, few works have been proposed to deal with this problem.

Porras et al [30] developed a prototype system called M-Correaltor, which ranked alerts based on the likelihood of success, the importance of victims, and attack type interest. The likelihood of success examines the matching between the vulnerability requirement and the target topology. Alerts are also prioritized based on the degree to which they targeted critical assets, and the amount of interest the user has towards the attack type. Mathematically, each incident will be ranked using an adoption of Bayesian belief network. As shown in figure 2.2, the Bayesian tree is made of three main branches: outcome, relevance, and priority. The outcome branch represents the information provided by the security devices. The priority subtree represents both the incident class importance and the severity of the attack. The last branch concerns the compression between the target environment and the vulnerability requirement of the corresponding attack. Bays net is used here because it is effective even with the lack of information in the network, such as the relevance subtree.

Jinqiao Yu et al [46] evaluated alerts based on two aspects: first, alerts that do not correspond to any attack in the vulnerability knowledgebase will be prioritized for further investigation; second, the applicability of the attack towards the protected network is checked. Similarly, Qin and Lee [32] prioritize alerts based on their relevance to the protected networks and hosts, as well as the severity level of the corresponding attacks assessed by security experts. As shown in figure 2.3, this technique employed the simple structure of Bayesian networks (one level) to compute the priority values for each alert. For each alert, Jinqiao Yu et al compared the relevance of the attack

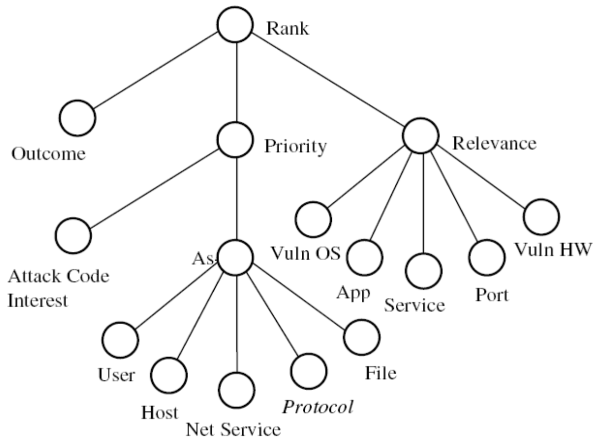


Figure 2.2: M-Correlator Incident Ranking

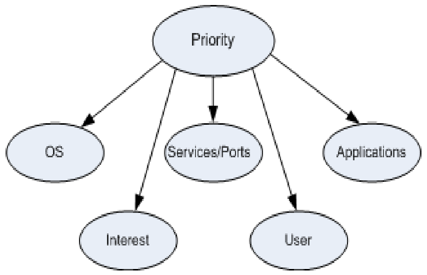


Figure 2.3: Alert Priority Computation Model

reported in the alert against the configuration of the target networks which include OS, Service/port, and Applications. The the existence of the target service/user in the hosts configuration was also analyzed. The interest of attacks can be specified by the security analysts based on the nature and mission of the attack. Finally, the compression results will be evaluated by the Bayesian networks.

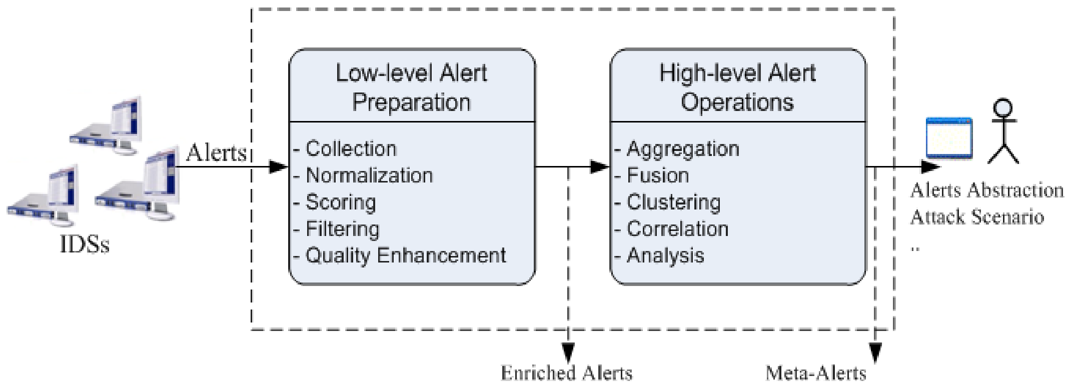


Figure 2.4: Alert Management Techniques Classification

2.4 Classification of Alert Management Techniques

Alert management techniques can be divided into two general classes: low-level and high-level. The low-level alert operations deal with each alert individually to enrich its attributes or assign scores based on potential risk. high-level alert management techniques, such as aggregation, clustering, correlation, and fusion, were proposed to deal with a set of alerts and provide an abstraction of these alerts. In fact, low-level alert preparation methods can help the high-level operations to provide more accurate results. For example, low-level evaluation techniques can examine a large number of alerts automatically (or semi-automatically) and prioritize them, leaving only important alerts for further inspection and investigation. As a result, high-level techniques will improve their outcomes due to the early low-level evaluation steps. Baker and Benaten described the alert management techniques separation issue with its benefit [4]. As shown in fig 2.4, low-level alert preparation techniques receive alerts from IDSs and enhance them before the high-level alert operations take place.

2.5 Comparison of Alert Prioritization techniques

In this section, we briefly revisit the alert ranking techniques before comparing them. Porras et al. ranked alerts based on their applicability, target importance, and attack

Table 2.1: Alert Prioritization techniques Comparison

	M-correlator	TRINETR	Qin's Approach
Signature-based alert	Yes	Yes	Yes
Anomaly-based alert	some	All	No
Zero-day attack	No	No	No
Training (Bays net)	Yes	No	Yes

severity [30]. Jinqiao Yu et al. calculated the alert priority based on its applicability and its existence in the vulnerability knowledgebase [46]. Qin and Lee evaluated the alert based on attack severity and its applicability.

As shown in table 2.1, All techniques can successfully evaluate and prioritize alerts generated by signature-based IDSs. This capability is due to that the signatures provide more information about the corresponding attacks. However, in the case of anomaly-based IDS alerts, M-correlater will prioritize all the alerts that target important assets. In this context, non-critical alerts that target critical machines will be prioritized even if these alerts should not be prioritized. TRINETR will prioritize all alerts generated by the anomaly-based IDS since this approach prioritizes any alerts that have no reference. Finally, Qin's approach completely fails to evaluate these types of alerts. M-correlater and Qin's approach need to train the Bayesian network in order to provide accurate results whereas TRINETR is required to encode too much expert rules. Generally, these techniques are promising to evaluate alerts generated by signature-based IDSs, but they cannot correctly evaluate alerts raised by anomaly-based IDSs, since they heavily rely on the vulnerability knowledge base of the known attacks.

2.6 Conclusion

In this chapter, we explained the need of alert management in intrusion detection systems. The standard alert representation format, IDMEF, was described in more detail due to its wide acceptance in the IDS research community. We tried to briefly cover all of the state-of-the-art management techniques that deal with IDS alert. Alert scoring and prioritization approaches were studied extensively. Generally, All alert management techniques aim to help the security analysts to understand what the IDS is stating and accordingly appropriate actions will be taken

Chapter 3

Alert Evaluation Architecture and Priority Scheme

In this chapter, we present FuzMet, an approach for prioritizing alerts generated by IDS, aiming to make the management process of the security analysts effortless and accurate. First, we generally explain FuzMet architecture and how it works. As we discussed in 2.3.4, researchers have proposed number of criteria to prioritized alerts which are not excluded from the following ones: the applicability of the attack, the severity of attack, and importance of the asset. However, in section 3.2, we extensively describe FuzMet metrics that are used to prioritize alerts and how they differ from the existing ones. The Fuzzy-logic reasoning approach and the advantages of using it in FuzMet scheme are shown in section 3.3. A re-scoring technique is presented in section 3.4. In Section 3.5, we present FuzMet configuration related issues. Finally, conclusion of the scoring and re-scoring approaches as well as the configuration issues are presented in section 3.6.

3.1 FuzMet Architecture Overview

This section describes FuzMet- an automatic IDS alert evaluation approach. The latter assists security analysts in automatically measuring the seriousness of each alert based

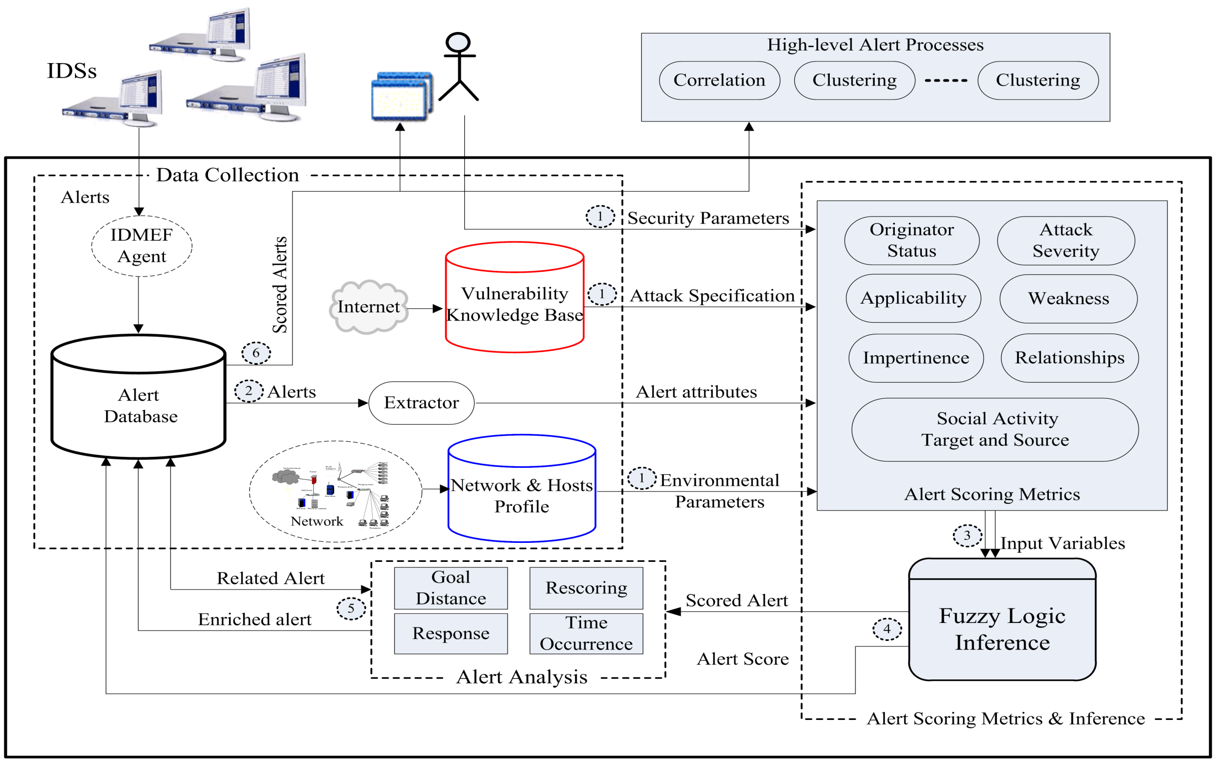


Figure 3.1: General Evaluation Process

on some criteria. We logically assign a numerical score to the alert using fuzzy logic. Typically, IDS alert management techniques, such as clustering and correlation, suffer from involving unrelated alerts in their processes. This consequently lead to imprecise results. Therefore, we introduce an alert rescoring technique that allows to a further reduction of the number of alerts.

As shown in figure 3.1 in the doted areas, FuzMet alert management architecture involves three main components: (a) data collection, (b) alert scoring metrics and inference, and (c) alert analysis.

3.1.1 Data Collection

Data collection includes four main resources: alert attributes, monitored environment, security administrator parameters, and vulnerability knowledge base. Alert attributes consist of several fields that provide information about the attack. The information provided varies from one IDS product to another. However, we assume that the alert is compatible with IDMEF format [8], which has recently became an industry standard. In the protected environment, we need some information to help us evaluate IDS alerts. Different environments contain different parameters based on the running services, applications, or operating systems. The security administrator of the protected network can specify the parameters that are involved in the evaluation process. For instance, we need information about all running services, applications, and operating systems, such as version, release time and existing vulnerabilities. Furthermore, the security administrator can specify the importance of each host in the network including the monitoring devices, the IDS. The public vulnerability knowledge bases, such as the National Vulnerability Database (NVD) [25] and Bugtraq [17], 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 & inference component.

3.1.2 Priority Metrics & Inference

Priority metrics and inference constitute the core components of FuzMet 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. The scoring metrics and reasoning approach will be discuss in more details in the later sections.

3.1.3 Alert analysis

This component provides an additional evaluation of the IDS alert. This component includes four main functions namely, rescoring, attack distance, occurrence time, and response plan. In this thesis, we investigate the rescoring function whereas the other functions have been described briefly and included in the our future works.

3.1.4 Flow Illustration

The flow of the evaluation diagram shown in figure 3.1 illustrates the alert evaluation process. First, the monitored environment information, the security administrator parameters, and the vulnerability knowledge base should be available before an alert is generated by the IDS. Second, when the alert is raised for evaluation, we compute the value of each scoring metric (which we will describe in detail later) based on the input data from the first step. Then, a fuzzy logic inference is used as a reasoning with the large number of alerts and present an abstraction of these technique to score each alert based on the metric values. The alert is then stored in the alert database with its score. The alert is also passed to the alert analysis component for further investigation. The analysis component measures the distance of the current attack from its possible goals. It rescores the alerts that are suspicious to be a preparation step for later attacks. It also detects suspicious activities which violate predefined system usage (such as using a port number which is only allowed during working hours). Finally, it provides a response plan to the intercepted attacks and makes it available to the system administrator for further investigation. As result, high scored alerts stored in the database can be presented to the security administrator for further investigation where appropriate actions will be taken. Furthermore, High-level alert management functions can benefit from the results offer by both the scoring and rescoring techniques to provide accurate outcomes.

3.2 Alert Scoring Metrics

The alert scoring metrics shown in Figure 3.1 are used to evaluate the criticality of alerts and to calculate a value for each one. Some of these metrics have been used in previous works (i.g., [31, 32, 46, 23]) but none associated all of them together. Additionally, we define new metrics that help us to accurately evaluate IDS alerts. FuzMet 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

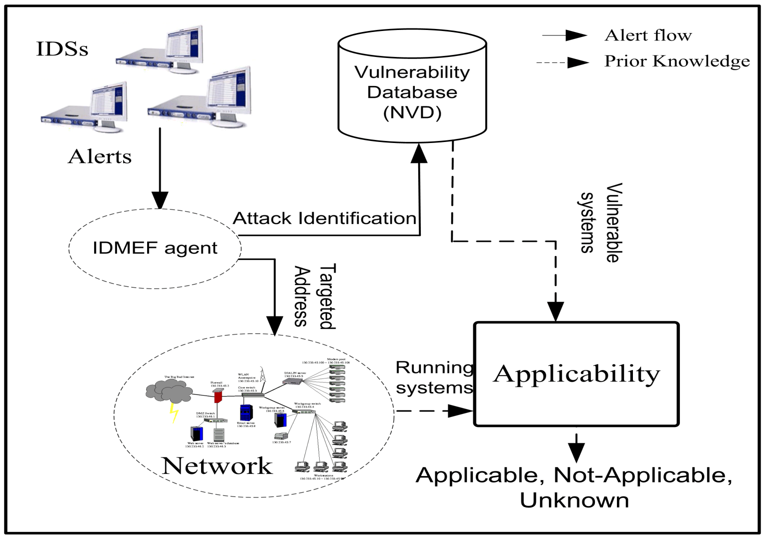


Figure 3.2: Attack Applicability Decision Process

those that deal with protected environments and the vulnerability knowledge base. We will describe the alert scoring metrics in greater detail in the following subsections.

3.2.1 Applicability

Applicability is a process that checks whether an attack that raises an alert is applicable in our environment or not. As shown in figure 3.2, in order to make a decision, this process requires checking with one of the vulnerability knowledge bases and knowing all the running services, applications, and operation systems. From alert's attributes, we extract the attack's specification which uniquely identifies the attack. Then, we can check with vulnerability knowledge bases to see whether the attack is applicable in our network or not. To illustrate the process, first, information about the protected environment and the attack should be available before the alert is generated. Whenever alert is raised, we select the target address and the attack identification from its attributes. From the environment knowledgebase, we can check what are the services (also version) are running in the target address reported in the alert. Form the vulnerability knowledgebase, we can check the infected services by the reported attack. As result, if there is match between the running systems and any of the infected infected system, attack will be considered applicable at this point. Otherwise, attack is mistakenly reported and the alert is treated as a false positive alert.

Furthermore, we further investigate the vulnerability that the attacker is trying to exploit to check whether it is patched or not. If it is patched, then the attack will be considered not applicable since there are no possibilities for that attack to success. But, the attack will be considered applicable if the corresponding vulnerability has not been patched. In general, figuring out the applicability of the attack on a given network is reduced to a search problem.

3.2.2 Importance of Victim metric

This metric is used to specify the criticality of the target machine reported in the alert. Several elements will participate in deciding the importance of the system in the environment including services, applications, and accounts. 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. Before introducing the function that calculates the criticality of the target machine, we will present a general weighted equation that is used in this metric and the rest of the thesis.

$$w'(a) = w(a) \times a \tag{3.1}$$

$$w(a) = \begin{cases} low & \text{if } 0 \leq a < th_l, \\ med & \text{if } th_l \leq a < th_h, \\ high & \text{if } th_h \leq a < 1. \end{cases} \tag{3.2}$$

Equation 3.1 aims to compute the value of any element based on its weight. A high weight is chosen if the object is critical and vice versa. In this metric, the criticality of a machine is calculated based on the running services/applications and the account associated with them. Different services have different weights according to their importance to the environment as well as the accounts. Equation 3.3 gives the formula

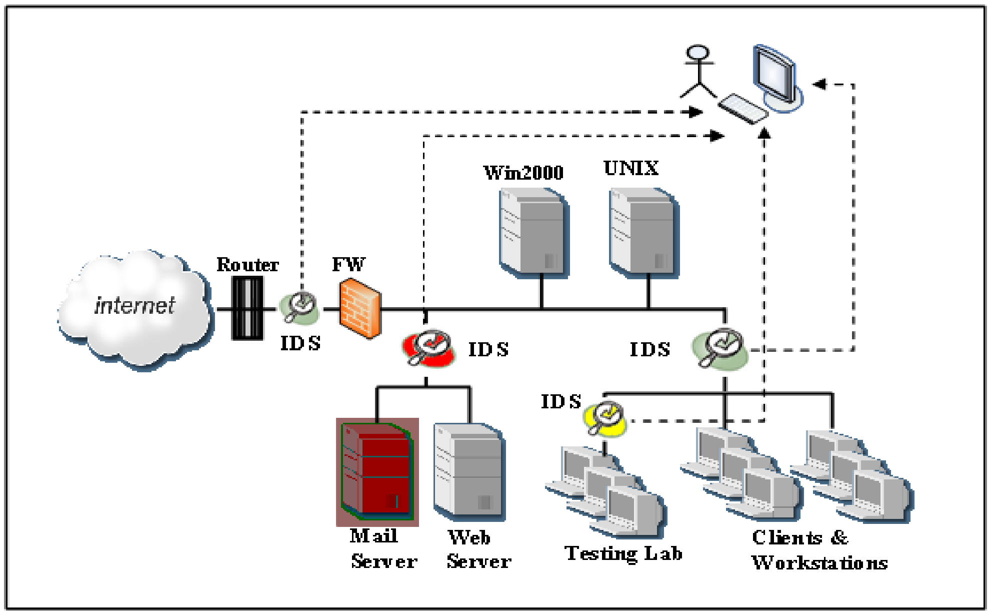


Figure 3.3: Different IDSs Placement Values

used for the importance metric.

$$I(m) = \frac{\sum\limits_{s \text{ runs on } m} w'(I(s) \times I(Ac(s)))}{\sum\limits_{s \text{ runs on } m} w(I(s) \times I(Ac(s)))} \tag{3.3}$$

Importance describes the significance of the victim machine that is running in the protected environment. The value of the importance is calculated on a scale from zero to one. A zero indicates that the victim machine reported in the alert does not include any important host, service, application, account, or directory. Scores closer to one indicate that the attack is targeting critical system component.

3.2.3 Sensor Status

What part of the environment does the monitoring device cover? Is it configured? Is it uptodate? What is its accuracy? Answering these questions for each sensor in the environment will describe its status. Let *Sensor_Status* denote the status of the IDS

Table 3.1: Sensor Status Parameters

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

that generates the alerts. The *Sensor_Status* is affected by its *placement*, *configuration*, *uptodate* and *accuracy* status. The security administrator has a full flexibility to deploy multiple IDSs in different places in the environment. However, not all places have the same value of sensitivity and criticality. The illustration example shown in figure 3.3 declares that several IDSs can be installed in a simple network to fully fulfill the security requirements [14]. The IDS protecting the DeMilitarized Zone (DMZ), which includes the Web and Mail servers, is considered to be in a critical place situation; therefore, a high value will be chosen for this IDS placement parameter. In contradict, The IDS running on monitoring the testing LAB segment has a low IDS placement value. For the *configuration* parameter, security administrator is usually aware of the configuration status of each IDS in the environment. This awareness is based on several constraints such as, deactivating unnecessary signatures, tuning threshold value, and service and rules compatibility. The *uptodate* status is determined based on the procedure of applying all the relevant signatures including the most recent ones. Like the antivirus tools, the IDSs vendors generate signature whenever a new attack is globally identified. If the security administrator is perfectly tracking the vendors' signatures generation then the value of *uptodate* status will be high and vice versa. Finally, the *accuracy* value of the sensor can be calculated offline using Bayesian Detection Rate (BDR) proposed in literature by Stefan Axelsson [2].

Formally, let I and $\neg I$ denote intrusive and nonintrusive behavior, respectively, and A and $\neg A$ denote the presence or absence of an intrusion alert. The Bayesian detection rate is the is simply computing the probability of the true positive $P(A/I)$ (alert is raised given attack is present). In order to calculate the BDR formula, one uses the past experience of the sensor inspection as follow:

$$P(A|I) = \frac{P(I)P(A|I)}{P(I)P(A|I) + P(\neg I)P(A|\neg I)}. \tag{3.4}$$

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

Table 3.1 shows all possible values that can be entered by the security expert who manages the sensors. For instance, if the place of the sensor is "*Critical*", its accuracy is *high*, it is well $Configured$, and it is *uptodate*. Then we expect that the value of the *Sensor_Status* to be high. Accordingly, the final alert's score will be increased. Basically, this indicator helps us to treat the alert based on the confidence that we have towards the sensor that generates the alert. The value of the *Sensor_Status* is computed based on the values of the four indicators following the formula:

$$Sensor\_status = \gamma_1 \times Place + \gamma_2 \times Conf. + \gamma_3 \times Update + \gamma_4 \times BDR \tag{3.5}$$

3.2.4 Attack Severity

Severity score measures the risk levels posed by a particular vulnerability. Security analysis sources, such as MITRE Common Vulnerabilities and Exposures (CVE) [38], Secunia [39],Internet Security Systems [40] and product's corporation (e.g., Microsoft) provide a severity score for all known attacks. For any raised alert, we will use multiple scores provided by the above organizations and take the weighted average to represent the severity score of the reported alert. The reasons behind involving multiple organizations to find the overall alert severity score are that each organization has its own metrics to calculate the severity score value and 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 [41] unlike the Secunia who reported this vulnerability as a moderately critical [42]. The goal of this metric is to present an accurate attack severity score which accordingly affect the overall alert score. Each attack severity score can fall into one of the three categorized classes namely high, medium, and low and a corresponding weight is chosen. For instance, attack severity score with value 9 is considered to be in the high class and the weight for this score will be high too. Doing this, allow us to be more biased to the high severity score reported in several attack analysis sources. Furthermore, we include the confidence value of each source that provides a severity score for known attacks. The goal of this constraint is to increase the trustfulness of the source which its services has been targeted. For example, if the attacker targeted a Microsoft's component (i.e. SQL server) then the attack severity score provided by Microsoft will have more trust values among the other participant scores. However, this parameter will make our overall severity score equation more dynamic.

$$SS(a) = \frac{\sum_{i=1}^{n} w'(SS_i(a)) \times \delta(i, v(a))}{\sum_{i=1}^{n} w(SS_i(a))} \tag{3.6}$$

$SS(a)$ represents the severity of the attack that triggers the alert. Security analysis experts publish severity scores (SS) for all discovered vulnerabilities. Several opinions from multiple vulnerability analysis databases will be collected as well as the opinion of the vendor whose service is being targeted. However, severity score representations come in different formats: numeric or categorical. Therefore, we would first normalize the severity score to a value between zero and one. Then, we can calculate the weighted average of the participants' severity scores as shown in equation 3.6. Obviously, the high severity score opinion will be given a high weight and vice versa.

3.2.5 Service vulnerability metric

We adopt the method proposed by Abedin et al [13] 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

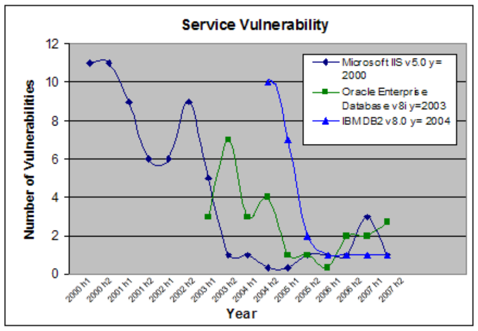


Figure 3.4: Evolution of Service Vulnerabilities Over Time
Generated from data provide in [18]

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 discovered through scanning softwares. 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 [16], 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 3.4, experience has shown that services in the early days of their release have more vulnerabilities than those released longer in the field.

Existing vulnerability 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 EVS 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 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 EVS by the weighted arithmetic mean as follows:

$$VS_e(s) = \frac{\sum\limits_{v \in V_e(s)} w(v) \times SS(v)}{\sum\limits_{v \in V_e(s)} w(v)} \tag{3.7}$$

Historical vulnerability 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 3.8 shows how $VS_h(s)$ is computed.

$$VS_h(s) = \frac{\sum\limits_{v \in V_h(s)} w(v) \times SS(v) \times \lambda^{-age(v)}}{\sum\limits_{v \in V_h(s)} w(v)} \tag{3.8}$$

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

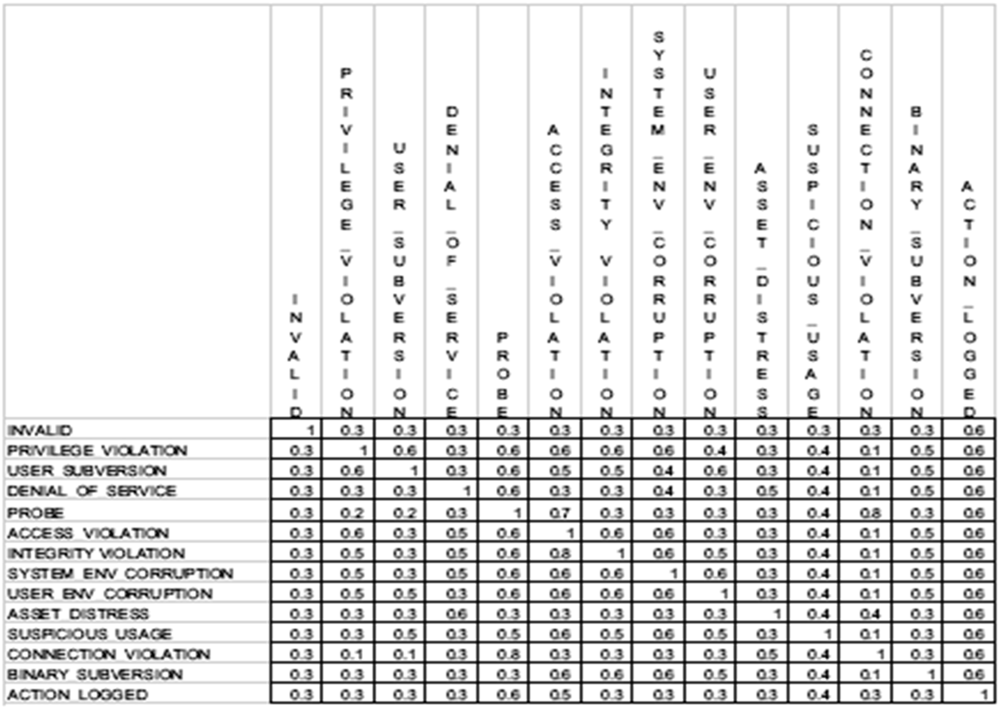
$$VS(s) = \eta_1 \times VS_e(s) + \eta_2 \times VS_h(s) + \eta_3 \times RT(s) \tag{3.9}$$

3.2.6 Relationship Between Alerts

Usually attackers launch their attacks in multiple steps in order to achieve the final goal. The early steps are a preparation for the later ones. Calculating the final score of the alert will involve evaluation of the relationship between the current alert and the previous ones. The score of the currently evaluated alert will increase if there is a relationship with any of the stored alerts. In this metric, we will restrict our investigation to the alerts that happened in a short period -say a couple hours- from the current alert. This restriction helps discarding very old alerts and focusing on recent ones since attackers typically try to achieve their goal as soon as possible before they can be identified.

The relationship between two alerts a_i and a_j is based on computing the similarity between their respective source IP addresses (Sim_{sip}), destination IP addresses (Sim_{dip}), source ports (Sim_{spt}), and destination ports (Sim_{dpt}). The source IP addresses similarity Sim_{sip} is computed by using the simple matching coefficient equation as shown in e eq.3.10. A similar formula exists for Sim_{dip}. However, calculating the similarity between two sources IP addresses is subject to two constraints. First, taking into consideration the subnet that the IP addresses are belong to. If the source addresses of the current alert and the previous alert share the same subnet then the (Sim_{sip}) will be high. Secondly, any difference in the $Most\ Significant\ bits$ of the IP addresses will make the (Sim_{sip}) low, unlike the changed of the $last\ significant\ bits$. For the ports number, Sim_{spt} and Sim_{dpt} are computed booleanly which produces one if the port numbers match and zero otherwise.

$$Sim_{sip}(a_1, a_2) = \frac{\sum similar\ bits(S_{IP}(a_1), S_{IP}(a_2))}{\sum all\ bits(S_{IP}(a_1), S_{IP}(a_2))} \tag{3.10}$$



	INVALID	PRIVILEGE_VIOLATION	USER_SUBVERSION	DENIAL_OF_SERVICE	PROBE	ACCESS_VIOLATION	INTEGRITY_VIOLATION	SYSTEM_ENV_CORRUPTION	USER_ENV_CORRUPTION	ASSET_DISTRESS	SUSPICIOUS_USAGE	CONNECTION_VIOLATION	BINARY_SUBVERSION	ACTION_LOGGED
INVALID	1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.6
PRIVILEGE VIOLATION	0.3	1	0.6	0.3	0.6	0.6	0.6	0.6	0.4	0.3	0.4	0.1	0.5	0.6
USER SUBVERSION	0.3	0.6	1	0.3	0.6	0.5	0.5	0.4	0.6	0.3	0.4	0.1	0.5	0.6
DENIAL OF SERVICE	0.3	0.3	0.3	1	0.6	0.3	0.3	0.4	0.3	0.5	0.4	0.1	0.5	0.6
PROBE	0.3	0.2	0.2	0.3	1	0.7	0.3	0.3	0.3	0.3	0.4	0.8	0.3	0.6
ACCESS VIOLATION	0.3	0.6	0.3	0.5	0.6	1	0.6	0.6	0.3	0.3	0.4	0.1	0.5	0.6
INTEGRITY VIOLATION	0.3	0.5	0.3	0.5	0.6	0.8	1	0.6	0.5	0.3	0.4	0.1	0.5	0.6
SYSTEM ENV CORRUPTION	0.3	0.5	0.3	0.5	0.6	0.6	0.6	1	0.6	0.3	0.4	0.1	0.5	0.6
USER ENV CORRUPTION	0.3	0.5	0.5	0.3	0.6	0.6	0.6	0.6	1	0.3	0.4	0.1	0.5	0.6
ASSET DISTRESS	0.3	0.3	0.3	0.6	0.3	0.3	0.3	0.3	0.3	1	0.4	0.4	0.3	0.6
SUSPICIOUS USAGE	0.3	0.3	0.5	0.3	0.5	0.6	0.5	0.6	0.5	0.3	1	0.1	0.3	0.6
CONNECTION VIOLATION	0.3	0.1	0.1	0.3	0.8	0.3	0.3	0.3	0.3	0.5	0.4	1	0.3	0.6
BINARY SUBVERSION	0.3	0.3	0.3	0.3	0.3	0.6	0.6	0.6	0.5	0.3	0.4	0.1	1	0.6
ACTION LOGGED	0.3	0.3	0.3	0.3	0.6	0.5	0.3	0.3	0.3	0.3	0.4	0.3	0.3	1

Figure 3.5: Attack Type Similarity provided in [37]

The overall similarity is a weighted sum of the four similarities mentioned above.

$$Sim(a_i, a_j) = \frac{w_{sip}Sim_{sip} + w_{dip}Sim_{dip} + w_{spt}Sim_{spt} + w_{dpt}Sim_{dpt}}{w_{sip} + w_{dip} + w_{spt} + w_{dpt}} \tag{3.11}$$

The relationship between two alerts (eq.3.12) is then computed by taking into account the follow up probability record between the corresponding type of attack of the alerts. This is a measure of the probability that an attack of type $T(a_j)$ is followed by an attack of type $T(a_i)$. The values of these are taken from the statistical analysis proposed by Valdes et al in [37]. As shown in figure 3.5, attacks are categorized into several incident classes based on their type. The probability that one incident class can be followed by another one is also provided in the figure. For instance, a probe incident class can be followed by access violation incident class with high probability, but this is not the case when it is the other way around since this it is not symmetric.

Algorithm 1 Alerts Relationship Algorithm

```
Require: Current Alert, Alert Log $\ell$
Ensure: Relationship Degree
 1: $A = \{a_i \mid a_i \in \ell$ & Timestamp$(a_i)$ within time window $RW\}$, $n = |A|$.
 2: for $i = 0$ to $n$ do
 3:   Calculate the Relationship Score between Current Alert and $a_i$
 4:   if Relationship Score > Highest Relationship Score then
 5:     Highest Relationship Score= Relationship Score
 6:   else
 7:     Keep the Highest Relationship Score
 8:   end if
 9: end for
10: Return Relationship Score
```

$$R(a_i, a_j) = P\left(T(a_j), T(a_i)\right) \times Sim(a_i, a_j) \tag{3.12}$$

Finally, the relationship score of an alert a_i (eq.3.13) is equal to the maximum relationship score it has with the alerts that occurred at most RW time units before a_i (i.e. $0 < ts(a_i) - ts(a_j) < RW$).

$$R(a) = \max_{0<ts(a_i)-ts(a_j)<RW} R(a_i, a_j) \tag{3.13}$$

Algorithm 1 illustrates the procedure of evaluating the relationship between alerts. The algorithm starts by specifying the time window *RW* that the relationship scores will be calculated between the current alert and the previous ones. We will consider the highest relationship score between two alerts as the overall alert relationship score.

3.2.7 Social Activity- Target and Source

A social network is a social structure made of nodes that are tied by one or more specific types of relations [15]. 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,

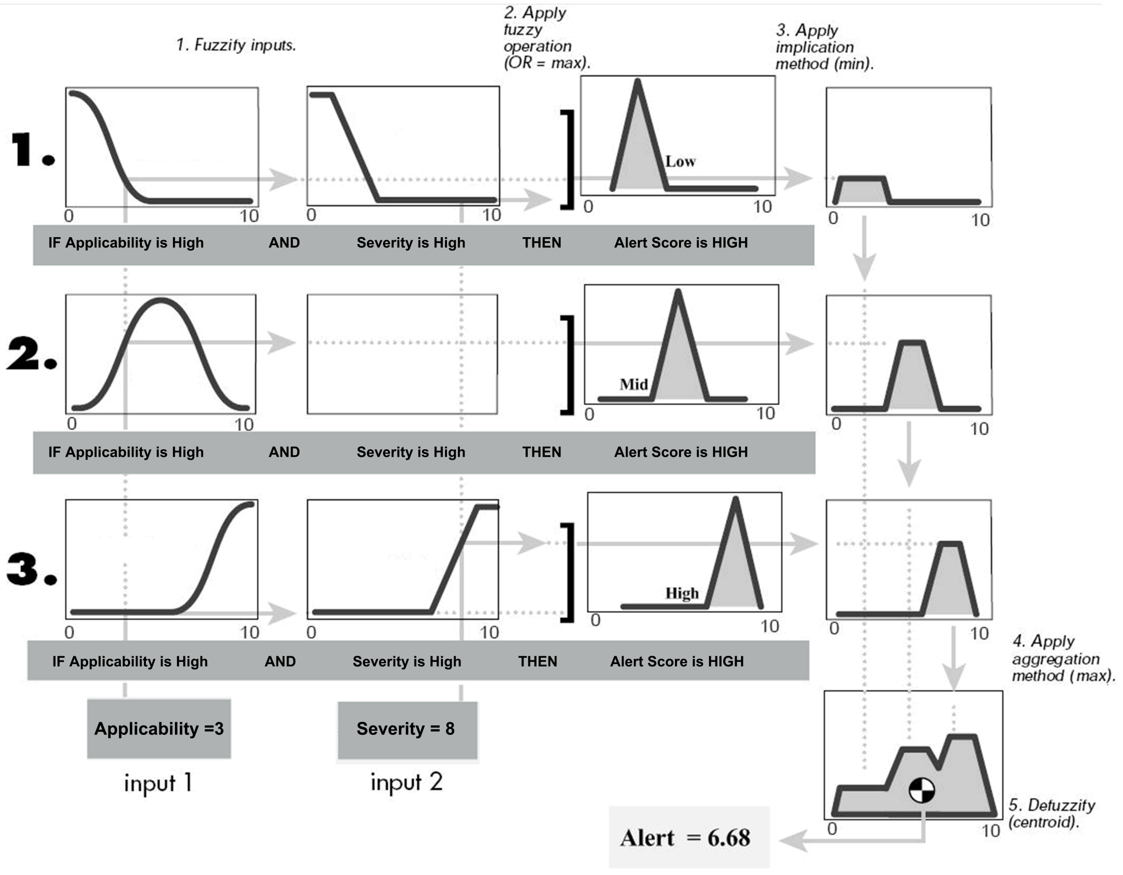


Figure 3.6: Portion of Fuzzy Logic Inference System

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. Our main 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.

Table 3.2: Fuzzy Logic Inference Rules

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

3.3 Fuzzy Logic Inference Approach

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 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 [27,36].

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. To achieve the smoothness and concise notation [43], we use Gaussian distribution curve as the type of the membership functions.

Rules are defined from domain expert as shown in Table 3.2. Rules in a fuzzy expert system are usually of a form similar to the following:

if *applicability* is *High* and *severity* is *Avg* then set *alert score* 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 FuzMet approach, we use Fuzzy Logic to score the alerts generated by any IDS, given some evidence from several sources namely, the alert itself, the environment where the alert is raised, and the vulnerability knowledge base. As shown in figure 3.6, Fuzzy Logic Inference first takes the input values from the metrics (e.g., applicability, severity, importance, and relationship metrics) and then fuzzify these inputs by 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.

3.4 Alert Rescoring Approach

The main goal of the alert re-scoring technique is to score alerts that are already scored based on their relationship with the current alert. One of the reasons for re-scoring alerts is to notify the security administrators with the early steps of the attack, that may be scored low. Another reason is to emphasize the previous activities of an attacker who is launching a very critical attack. However, the method of scoring alerts only once will limit the security administrators in identifying the non-critical early-steps of the attack, especially if a filtering technique is used discarding those low scored alerts. Also, high-level alert management techniques such as correlation or clustering can benefit from re-scoring and provide more accurate results.

Alert management techniques such as aggregation, grouping, scoring, filtering, clustering, correlation, and fusion were proposed to deal with the large number of alerts and present an abstraction of these alerts. First, alerts are aggregated from multiple IDSs then similar alerts will be grouped together into hyper-alerts. Scoring function will evaluate the hyper-alerts and assign a score for each one according to its importance or seriousness. Low-score alerts will be discarded and they will not be involved in any further analysis. Then, correlation functions may be applied to present the attack scenarios. Clearly, scoring alerts once will discard the early non-critical attack that prepare for a later critical attack. Consequently, the early steps of the attackers will not be involved in any further analysis such as attack scenario construction. For instance, 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 the critical alerts in the high-level operations, such as correlation, will prevent the non-critical early steps of the attack to be considered. On the other hand, involving all alerts in the high-level operation will make the total number of correlated alerts unmanageable manually. Hence it is important to highlight only the critical alerts and their related alerts. Therefore, we perform a re-scoring of alerts based on the $prepare-for$ relationship and the trustfulness of attacks. The $prepare-for$ checks if there is any relationship between the currently evaluated alert and the entire alerts log. The trustfulness examines the previous activities of the current source of the attack if the launched attack is critical. Then we can adjust the value of trustfulness for this source in our source evaluation metric.

Security analysts can apply the rescoring function periodically (at the end of the day) or before applying the high-level management techniques. Algorithm 2 illustrates the rescoring process in FuzMet. The algorithm starts by taking the alert log, the

Algorithm 2 Alerts Rescoring Algorithm

```
Require: Alert Log $\ell$, Score threshold $ST$, Relationship threshold $WT$
Ensure: Rescored Alert log
 1: A= {a(i) | a(i) $\in \ell$ }, n=|A|.
 2: B= {a(j) | a(j) $\in \ell \bigwedge 0 <$ Timestamp(a(i))- Timestamp(a(j)) $<$ WT }, m=|B|.
 3: for i=0 to n do
 4:    for i=0 to m do
 5:       Calculate Relationship: R = Relationship(a(i), a(j))
 6:       if R $> \nu \bigwedge$ AlertScore(a(i)) $>$ ST then
 7:          Fetch the Applicability of a(j)
 8:          if a(j) is Applicable $\bigwedge$ AlertScore(a(j)) $< \upsilon$ then
 9:             Rescore a(j) where AlertRescore(a(j)) $>$ ST
10:          else
11:             Keep the Alert Score
12:          end if
13:       else
14:          Keep the Alert Score
15:       end if
16:    end for
17: end for
18: Return Alert with Rescore Value
```

score threshold for prioritizing alerts, and the window time WT that limits the alerts involved in calculating the relationship degree. For each alert in the log, we calculate its relationship with the previous alerts within the window time WT. If the score of the current alert is high and the relationship degree is strong, then the involved previous alert is a candidate for rescoring. In order to rescore this alert, we further investigate its applicability against the protected network as well as its score should be low. If the conditions have been satisfied, then the alert will be rescored and prioritized for further investigation.

3.5 FuzMet configuration issues

The FuzMet IDS uses a number of metrics, fuzzy inference rules, and alert rescoring mechanisms all of which have several configurable parameters which influence the precision of intrusion detection and alert prioritization. The security administrator has a number of parameters that can be configured while others are not. Non-configurable parameters include the severity score values $SS_i(a)$ gathered from security expert organizations, sensor update status and accuracy, and the attack follow up probability matrix P (eq.3.12). The set of configurable parameters includes the five parameters of the weight function w (sec.3.2) which in turn influence the machine importance metric $I(m)$ (eq.3.3), the severity score metric SS (eq.3.6), the existing vulnerability metric VS_e (eq.3.7), and the historical vulnerability metric VS_h (eq.3.8). The importance $I(s)$ of each service s and the importance of each user account $I(a)$ need also to be configured in order to reflect the criticality of each service, user account, and machine. The sensor status metric has four configurable weight parameters. The severity score metric (eq.3.6) requires the definition of $\delta(i, v(a))$ for each severity score provider i and targeted victim $v(a)$ tuple. This parameter can be made into a more static form δ_i if only a single trust value is given to a severity score provider independent of the target victim. The service vulnerability metric (eq.3.9) has an additional four parameters including the decay coefficient λ as well as the three η_i weights. The relationship metric (eq.3.13) has also four additional parameters. This makes the overall number of configurable parameters equal to:

$$N_{cfgp} = 17 + |services| + |accounts| + |scoresources| \times |victims| \tag{3.14}$$

Besides metric configuration, there is a need to define the appropriate fuzzy inference rules which help in capturing the severity of each attack. As can be noticed, providing an optimal configuration of the FuzMet system is not trivial to say the least.

3.6 Conclusion

In this chapter, we presented FuzMet automatic method of evaluating alerts generated by IDS. Our main gaol was prioritizing the critical alerts to the security analysts among large number of alerts mixed with high rate of false positive. In order to achieve this, number of criteria were proposed. Applicability of the attack, the importance of victim, the relationship between the alert under evaluation and the previous alerts, and the social activities between the attackers and the victims are examples of the metrics we use to prioritize IDS alerts. These metrics are used as inputs of a Fuzzy-logic system in order to investigate the seriousness of the generated alerts and quantitatively calculate a score for each alert. This evaluation process will prioritize alerts when presented to the security administrator for further investigation. Additionally, we propose a rescoring technique to dynamically score alerts based on the relationship between attacks or the trustworthiness of the attackers.

Chapter 4

Experimental Results and Evaluation

This Chapter reports our evaluation methods conducted to validate the effectiveness of our proposed approach. First, the most popular datasets used in our experiments are described in section 4.2, namely, DARPA 2000 specific intrusion detection scenario dataset. [21]. In Section 4.3, we present the accomplished results of alert prioritization approach, FuzMet, which are divided into two parts. The first part shows the results of FuzMet carries out with only one configuration parameter set while the second present the experiments conducted with different configuration parameters sets and both are compared with snort results. Also, the consequences of the rescoring technique are shown in section 4.4. Finally, we conclude this chapter by describing the lessons gained by the conducted experiments in Section 4.5.

4.1 Overview of Alert Evaluation Experiments

One of the first concerns faced by those who are working in IDS alert management is how to find data sets that are appropriate for testing, evaluating, and validating their proposed algorithms. Generally, there are two methods used in this context. These methods are not only for evaluating alert management techniques but also for testing new intrusion detection algorithms. The first method is building a simulated network and collecting the relevant data for further investigations. However, creating a simulated network environment with hundreds of computers is often difficult and expensive. Also, simulated network environments may yield traffic that does not mimic a real network due to the inherent nature of simulated traffic [22]. Most importantly, comparing the results obtained through a simulated network environment with other results is not possible since both are using different environments. The second possible method is using the previously collected data sets. These data sets contain real attack instances on real networks which are publicly available for research purposes. Using the available data sets saves the development time and efforts. It is possible to compare results conducted by applying different approaches to the same data set. Several popular data sets are available such as, the DARPA Evaluation data sets cite[DARPA], the Knowledge Discovery and Data Mining (KDD) CUP 1999 competition data set [6], and DEFCON [35]. In our experiments, we used the second method for testing and validating the FuzMet approach. The characteristics of the data sets were suitable for our requirements. Particularly, we used the 2000 DARPA Intrusion Detection Scenario Specific Data Sets.

4.2 2000 DARPA Intrusion Detection Scenario Specific Data Sets

For the sake of testing and evaluating the efficiency of the IDS, the Defense Advanced Research Projects Agency (DARPA) has provided a number of Intrusion Detection Evaluation Data Sets including the 1998, 1999, 2000 Data sets. In this thesis, we used the latter one. The 2000 data set contains attack scenario that includes a distributed denial of service attack "DDoS". This attack scenario is carried out over multiple network and audit sessions in which an attacker probes the network, breaks into a host by exploiting existing vulnerability, installs the software required to launch a DDoS attack, and finally launches the DDoS attack against an off-site server. The 2000 dataset is divided into two data sets namely, LLDOS 1.0 and LLDOS 2.0.

4.2.1 LLDOS 1.0 Data sets

In general, Distributed Denial of Service (DDoS) is the type of attack conducted in this data set. in order for the attacker to achieve this gaol, number of different attacks are used as preparation steps. The attacker starts by using a scripted attack to break into a variety of hosts. This step is conducted by scanning all subnets in the target network looking for alive hosts, and identifying which alive hosts are running Solaris sadmind service by using rpc port scan. Then, the attacker uses the Solaris sadmind exploit, a well-known Remote-To-Root attack, to successfully gain root privileges in Solaris hosts in the targeted network. With the root access in the compromised machine, the attacker install the Mstream DDOS tool, that is capable of flooding target systems with high volumes of TCP packets [12], as the final preparation steps for lunching DDoS. Finally, the attacker launch a DDOS from the compromised network (by the installed DDoS tool) to an off-site server.

During the time of launching the attack, number of sensors were install in the network to record all the traffic including all attack instances. The DARPA LLDOS 1.0 dataset contains traffic collected from two network zones: “DMZ” and “inside”. The series of attacks in the dataset are carried over multiple sessions or phases, the interval times of which are shown in Table 4.1. A summary of the five phases of the attack scenario are:

1. Scans the network in order to launch a DDOS attack against an off-site server.
2. Looks for the *sadmind* daemon of live IP.
3. Break in hosts by exploiting a sadmind vulnerability.
4. Installing a mstream Trojan on the compromised machine.
5. Launches the DDoS attack against the remote site.

4.2.2 LLDOS 2.0 Data sets

The second attack scenario data set includes a similar sequence of attacks run by an attacker who is a bit more complicated than the one in version LLDOS 1.0. The main difference between LLDOS 2.0. and LLDOS 1.0 is that in LLDOS 2.0.2 the attacker uses DNS HINFO queries for probing the host, platform, and operating system rather than Sweeping IP's and using RPC port scaning. Also, the attacker tries to break in one host by exploiting the sadmind vulnerability and gain root privileges where he continues to compromise other hosts unlike the LLDOS 1.0 where the attack compromise each vulnerable host individually. From this point, the attacker proceeds to install the DDoS componenet and launches the DDoS attack. The five phases of the attack scenario are summarized as follows:

1. Probe of one host, Eyrie's public DNS server, via the HINFO query
2. Break in the host via the sadmind exploit.
3. Upload of mstream DDoS software and attack script (to break-into more hosts).
4. Initiate attack on other hosts
5. Launching the DDoS

4.3 Prioritizing Attacks Observations

In order to validate the effectiveness of the FuzMet approach for prioritizing IDS alerts. The DARPA 2000 intrusion detection LLDOS dataset 1.0 [21] was used and the generated alerts were stored in a MySQL database. Java was used to compute the different metrics related to the alerts. These metrics were then input to the fuzzy rule set of in order to generate FuzMet alert scores. Matlab Fuzzy Logic toolbox [36] was used for the fuzzy rules specification. Finally, we compared FuzMet alert scores with the ones generated by Snort. This results section is divided into two parts. The first part shows the output of Snort and FuzMet for one configuration set and details the comparison between them. The second part focuses on the optimal configuration problem and presents the result of a selected set of results from among a set of 200 conducted simulations.

Table 4.1: Prioritized alerts of the DARPA 2000 LLDOS 1.0 Dataset using Snort

Phase	From	To	Length	Alert Name	Non grouped			Grouped		
					#Alert	#high scores	Avg	#Alert	#high scores	Avg
Phase1	9:51:36	9:52:00	0:00:24	ICMP Echo Request	786	0	≈2	15	0	≈2
				ICMP Echo Reply	30	0	≈1.5	6	0	≈1.5
Phase2	10:08:07	10:17:10	0:09:03	RPC portmap sadmind request UDP	250	79	≈8	43	28	≈8
				RPC sadmind UDP PING	9	6	≈5	4	3	≈5
Phase3	10:33:10	10:35:01	0:01:51	RPC sadmind with root attempt UDP	46	28	≈9	37	29	≈9
				RPC sadmind UDP NETMGT_PROC_SERVICE	46	6	≈9	3	3	≈9
phase 4	10:50:01	10:50:54			0	0	0	0	0	0
Phase5	0.4765625	11:41:19	0:15:04	BAD-TRAFFIC loopback traffic	141	141	≈9	1	1	≈9
False alerts				NETBIOS NT NULL session	2	0	≈1	1	0	≈1
				ATTACK-RESPONSES Invalid URL	4	0	≈1	1	0	≈1
				SNMP request udp	12	0	≈6	9	0	≈6
				SNMP public access udp	6	0	≈5	6	0	≈5
				ATTACK-RESPONSES 403 Forbidden	10	0	≈1	3	0	≈1
				MS-SQL version overflow attempt	1	0	≈1	1	0	≈1
				ICMP redirect host	2159	0	≈1	26	0	≈1
Total					3502	260		156	64	

4.3.1 Alert scoring

Snort was used with the maximum detection capability to scan and detect intrusions within the binary tcpdump file of both of the "inside" and "DMZ" traffics. Snort reported 3502 alerts (321 inside, and 3181 DMZ). In order to filter out redundant alerts, we employed a grouping of alerts based on exact similarity within a specific window of time. This resulted in a new total of 156 alerts (Table 4.1), thus a gain of 95.5%.

The FuzMet scoring technique was applied 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. FuzMet evaluated these antecedents and scored them as low (1.2-2.3) as shown in Table 4.1. In the second phase, we received 259 alerts from the traffic of both the *DMZ* and *inside* parts, which

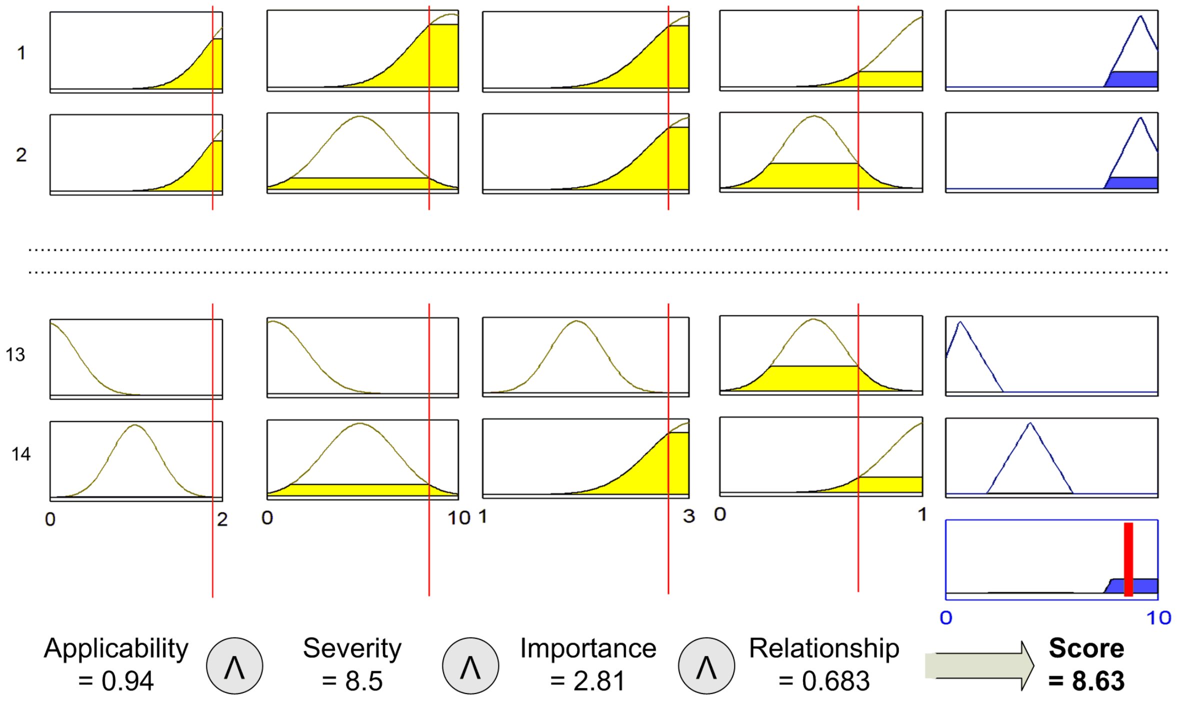


Figure 4.1: Fuzzy logic inference engine

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 4.1 summarizes the results of the FuzMet alert scoring (with and without the grouping function) technique on the DARPA 2000 dataset. FuzMet alert prioritization

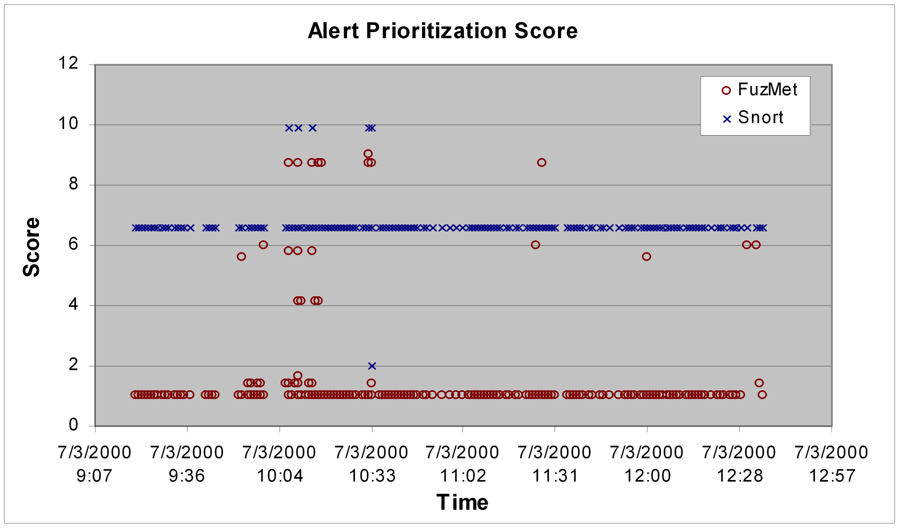


Figure 4.2: FuzzMet vs. Snort scores (Over Time)

was effective in identifying the false positive alerts which Snort failed to detect. For example, Snort generates a "MS-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 4.2 and 4.3 show that after we score the alert, a security administrator can be provided with the most important alerts unlike the result of Snort which assigns a level-two priority (out of 3) to most of the alerts.

4.3.2 FuzMet optimal configuration

As discussed in section 3.5, the parametrization of the FuzMet alert scoring system determines its effectiveness. Mis-configuration can easily lead to imprecise outcomes, which consequently results into missed attacks. The objective is hence to determine the optimal configuration of the FuzMet set of parameters. Because the used dataset does not come with information about network topology, set of running services, and the different user accounts, a number of FuzMet metrics couldn't be involved in the experiments. These include the sensor status, service vulnerability, and social relation-

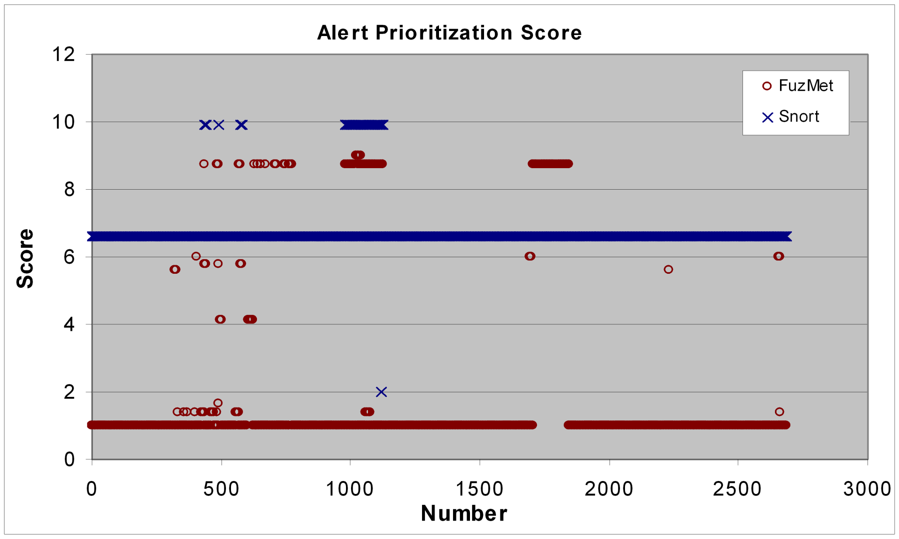


Figure 4.3: FuzzMet vs. Snort scores (By Number)

ship metrics. The metrics which were used are the applicability, importance, severity, and relationship metrics. For simplicity, all severity score sources were given equal confidence ($(\delta(i, v(a)) = 1)$.

200 simulations have been conducted with different configuration parameters in order to determine an optimal configuration set. An optimal configuration set is the one which makes FuzMet prioritize only those alerts which actually belong to the DDOS attack phases. Due to space limitation, only four representative simulations will be shown in this section. Table 4.4 shows the parameters of the severity and relationship metrics for the selected simulations. The weight group contains the configuration parameters for the w function which directly affects the severity score metric (eq.3.6). Table 4.2 shows the configuration parameters related to the importance metric. A value of 1 indicates of low importance while a value of 3 indicates a highest importance. A value of 0 indicates the case where the target machine is unknown or that the value has not yet been configured. Simulation S_1 showed the worst result among the 200 simulations. Only one single alert was given high priority (figure 4.4(a)). The result can be explained by the fact that the machines running the sadmind service, core to

Table 4.2: Importance configurations

Dest IP	Target	S1	S2	S3	S4
131.84.1.31	unknown	2	2	1	1
172.16.112.1	interface	3	1	2	2
172.16.112.10	Sun Solaris 2.6	0	1	3	3
172.16.112.100	Windows NT 4.0	3	0	2	2
172.16.112.105	unknown	1	3	1	1
172.16.112.194	Sun Solaris 2.5.1	1	1	2	2
172.16.112.50	Sun Solaris 2.5.1	0	3	3	3
172.16.113.148	Linux Redhat 5.0	1	3	2	2
172.16.113.204	Sun Solaris 2.5.1	0	0	2	2
172.16.113.84	SunOS 4.1.4	0	1	2	2
172.16.114.10	Sun Solaris 2.6	3	1	3	3
172.16.114.2	Sidewinder	0	1	1	1
172.16.114.20	Sun Solaris 2.7	1	2	3	3
172.16.114.30	Sun Solaris 2.7	2	0	1	1
172.16.114.50	Linux Redhat 4.2	1	0	2	2
172.16.115.1	firewall interface	2	1	1	1
172.16.115.20	Sun Solaris 2.7	0	2	3	3
172.16.115.87	Windows 95	0	1	2	2
172.16.116.44	Windows 3.1	3	3	2	2
172.16.117.103	MacOS	2	3	2	2
172.16.117.111	MacOS	1	1	2	2

Table 4.3: Fuzzy rules set

Rule	Applicability	Severity	Importance	Relationship	Output
1	High	High	High	High	**V.High**
2	High	Med	High	Med	**V.High**
3	Low	Low	Low	Low	**Low**
4	Low	Low	Med	Med	**Low**
5	Med	Med	High	High	**Med**
6	Med	Med	High	High	**Med**
7	Low	High	Low	Low	**Low**
8	High	Low	Low	Low	**Low**
9	Low	Med	Low	Low	**Low**
10	Low	Med	Low	Med	**Low**
11	Low	Med	Low	High	**Low**
12	Low	Low	Low	High	**Med**
13	Low	None	None	None	**Low**
14	High	Med	High	Med	**Low**

Table 4.4: Metric Parameters

Parameters		***S1***	***S2***	***S3***	***S4***
weight	low	0.10	0.34	0.35	0.16
	med	0.49	0.53	0.40	0.45
	high	0.83	0.56	0.45	0.60
	thl	0.30	0.30	0.30	0.30
	thh	0.60	0.60	0.60	0.60
relationship	sip	0.92	0.73	0.25	0.84
	dip	0.65	0.59	0.25	0.63
	spt	0.10	0.29	0.10	0.19
	dpt	0.22	0.29	0.10	0.11
	RW	10	10	5	20

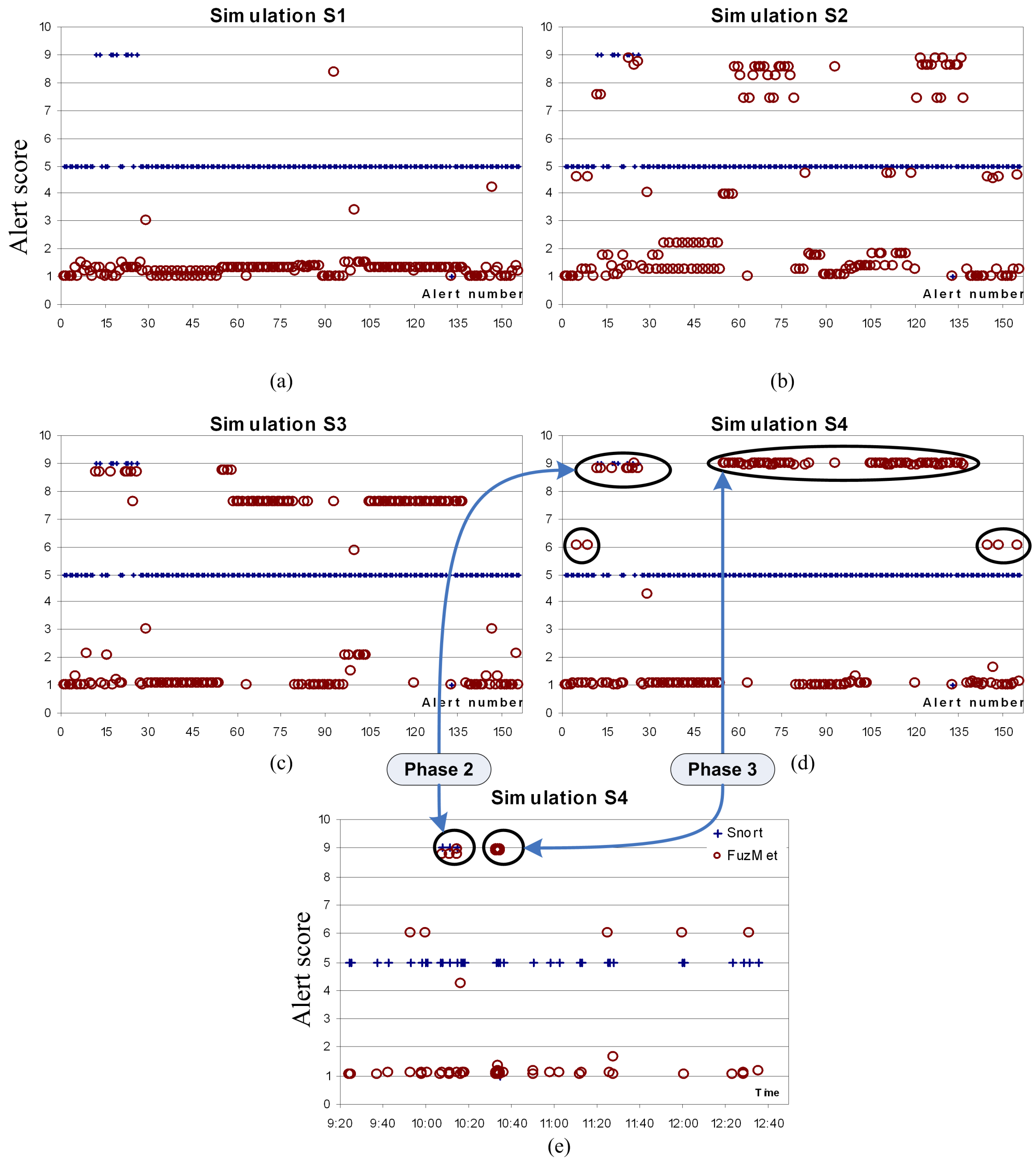


Figure 4.4: Configuration results

the DDOS attack, have been assigned a null importance (172.16.112.10, 172.16.112.50, and 172.16.115.20). In S_2, 31 alerts have been assigned high priority (figure 4.4(b)). S_2 differs from S_1 mainly in the importance metrics where the machines that are running Sadmind have now a non zero importance. S_3 gives a high priority to 11 alerts only (figure 4.4(c)). However, if the high priority threshold is lowered from 8 down to 7, S_3 records a number of 66 high priority alerts, hence equaling those generated by S_4 (figure 4.4(d)). S_4 manages to generate the best result among the 200 simulations with a number of 66 high score alerts and the highest average score of 4.02. S_3 and S_4 have the same importance configuration and differ in the weight and relationship parameters. S_4 is considered the best configuration not only based on the number of high alert scores. In fact, out of the four identifiable phases of the DDOS attack, only S_4 manages to detect all of them with a strong precision for phases 2 and 3 and a high-medium precision for phases 1 and 5 (phase 4 is only detectable by a host IDS and as such is excluded from the evaluation). In contrast, Snort manages to identify only phase 2. Figure 4.4(e) plots the results of S_4 based on time rather than on the number of alerts and shows the concentration of high score alerts for phases 2 and 3 which last 09:03 and 01:51 mins respectively. S_4 records 58 high score alerts for phase 3, with a duration of 01:49 mins. This interval is almost equal to the exact duration of phase 3 of the attack which corresponds to the exploitation of the sadmind vulnerability. It is to note also that even S_1 manages to generate one high score alert for phase 3 while Snort misses that phase completely and only detects phase 2 out of the four detectable phases.

4.4 Rescoring Alert Results and Advantages

We applied FuzMet rescoring technique to the alerts that are scored previously. As we discussed earlier in section 4.3.1, a simple alert grouping technique was applied to the alert log to remove the alerts redundancy (similar alerts occur in close time). This technique groups together the alerts that are similar in their IP addresses, port numbers, attack type. The grouped alerts represent all the attacker's steps used to launch a DDOS attack which are considered for rescoring. Since LLDOS 1.0 dataset

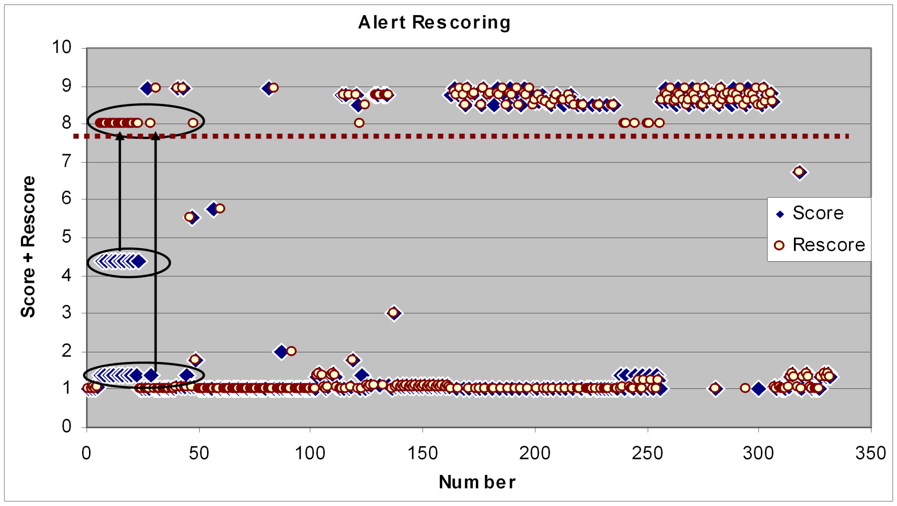


Figure 4.5: Alert Rescoring approach (By Number)

consists of only one complete attack scenario, the first phase, which contains non-critical attacks (regular scanning), is good candidate for rescoring for two reasons. First, it is considered to be a preparation step for later attacks. Second, FuzMet scores the alerts generated by this phase as low. Therefore, we focus our analysis on the alerts related to the first phase to check the usefulness of our rescoring approach.

In phase one, the attacker starts to scan the network at 09:51:36 until 09:52:02 by performing a scripted IPsweep of multiple class C subnets on the victim network. Previously, this phase was scored low by FuzMet according to its seriousness and impact. FuzMet successfully rescored this phase to be attached with the other phases of DDoS attack scenario. As shown in figure 4.5, the alerts related to phase one were scored between (1.35 to 4.38) but after applying the rescoring function their score increased to be 8. The rescoring value were chosen to be 8 to insure that the rescored alerts will be included in the prioritized alert set. As a result, all the critical alerts as well as the preparation steps have been prioritized and presented to the security analyst.

4.5 Conclusion

This chapter presented the experimentation results of FuzMet; a system that uses fuzzy logic inference to evaluate and prioritize IDS alerts based on several metrics. We defined seven metrics related to the applicability of the alert, importance of the target, sensor status, alert severity score, service vulnerability, alerts static relationships, and the social relationships between system users. In the experiments, we use Snort with its maximum detection capability to scan the DARPA 2000 LLDOS 1.0 dataset. We applied our scoring algorithm to the alerts generated by snort and the results showed that the security analyst can recognize the critical alerts among the others. A number of 200 simulations were conducted in order to determine optimal configuration for FuzMet since it has high number of configuration parameters required by the different metrics and fuzzy logic engine.

While Snort detected only one phase of the four detectable phases of the attack the best configuration of FuzMet managed to detect all of those phases with a good precision for two of the phases and a very good precision for the others. Unexpectedly, even worst case configuration of FuzMet did also good in the sense that it managed to raise one single high score alert for a phase which Snort did not detect at all.

The conducted simulations showed the viability of the FuzMet approach at least for the selected intrusion scenario. In addition, the divergent behavior of FuzMet depending on how it is configured helped in identifying the configuration which leads to best performance wherein all attack phases are identified.

Chapter 5

Conclusion And Future Work

After implementing an intrusion detection system, one realizes that detecting intrusions is not the only challenge to solve. In fact, managing the large number of generated alerts is also a demanding task for the security analysts. Evaluating these alerts manually can be difficult, error-prone, and time-consuming. Therefore, automatic alert evaluation and prioritization are needed to keep the number of alerts in a human control. We argue that with the presence of the attack knowledge base, network information, basic alert attributes, and some security administrator we can judge about the seriousness of each alerts generated by an IDS. As a result, only critical alerts (which are reasonably in small number) will be prioritized for further investigation by security administrators. Involving unrelated alerts at the high-level management techniques such as clustering and correlation is another problem we are trying to address in this thesis. With these unrelated alerts, the results of the high-level management approaches will be either inaccurate (by involving only critical alerts) or difficult to manage (by involving all reported alerts). For instance, one of the correlation graph produced by correlating a large number of alerts, consist of 2,940 nodes and 25,321 edges as reported in [29]. A human user will have difficulties in analyzing such a graph in a short period of time. Therefore, we proposed a rescoring approach aiming to involve into the prioritized alert set those non critical alerts that prepare for critical ones. As a result, the critical alerts as well as the preparation steps will be prioritized and presented to security analyst.

5.1 Summary of Contributions

The contributions of this thesis can be summarized as follows:

- An automatic alert evaluation and prioritization framework that make the management task for the security analysts easier and controllable.
- A re-scoring technique that dynamically score alerts based on relationship between attacks.
- A comprehensive study of the impact of different configurations of the proposed metrics on the accuracy and the completeness of the alert scores generated by FuzMet.
- A survey and taxonomy of IDS alert management techniques.

5.2 Thesis Summary and Concluding Remarks

In this thesis, we presented FuzMet; an approach that automatically evaluates IDS alerts band prioritizes the critical ones based on a number of criteria. These include the applicability of the attack, the importance of victim, status of the sensor, the severity of the attack, the relationship between the alert under evaluation and the previous alerts, and the social relationships between system users. We used a Fuzzy-logic inference mechanism in order to score alerts. This score represents the seriousness of the alerts. The Fuzzy-logic approach receives the values calculated by the metrics for each alert as inputs and compute the overall score for that alert. Furthermore, we developed a rescoring technique that enabled us to rescore alerts to show the early steps of the attackers. This technique tries to increase the score of the alerts that were already scored low but which participated in preparing for later attacks. Additionally, this thesis presents the specification and configuration issues of FuzMet. Because of the high number of configuration parameters required by the different metrics and fuzzy logic engine, this paper particularly emphasized the problem of optimal configuration for FuzMet. The simulations were specifically conducted in order to determine such a configuration.

In the experimentation, the FuzMet approach has been applied onto the alerts generated by Snort with its maximum detection capability and using the DARPA 2000 LLDOS 1.0 dataset. The dataset features a distributed denial of service attack on a remote site. For the scoring approach, we successfully prioritized the most critical alerts which are relatively small comparing with the number of generated alerts. FuzMet prioritized the critical alerts that are related to the launched attacks, unlike Snort which scores most of the alerts as medium. For rescoring approach, the results showed that all the non critical alerts that prepared for any critical ones were rescored and included with the set of the prioritized alerts. For finding the best configuration parameters of FuzMet metrics, a number of simulations were conducted and selected results were presented. The conducted simulations showed the viability of the FuzMet approach at least for the selected intrusion scenario. In addition, the divergent behavior of FuzMet depending on how it is configured helped in identifying the configuration which leads to best performance wherein all attack phases are identified. However, even with the obtained results, we did not completely solve the configuration problem of FuzMet. In fact, many of the metrics were not configurable due to the absence of their related data from the chosen scenario dataset. In addition, it is not possible to prove that the identified best simulation is the actual optimal configuration; or even whether it is the best just for that particular data set. The search space involves a considerable number of parameters and further analytical analysis is required.

Although, our objective is to make the evaluation process more manageable for the security analysts, still we are concerned about the detection of the real attack in our evaluation process. Since our mechanism of prioritizing the critical alerts depends on the successful attack detection by the deployed IDS, the attacks will not be prioritized if the IDS missed them. Additionally, the computing time for calculating the values of each metrics and the application of the Fuzzy-logic reasoning eliminates the real-time alert prioritization approach. However, real-time alert ranking is not essential in an Intrusion Detection Systems, unlike intrusion Prevention Systems (IPS) which require real-time alert detection and prioritization mechanism.

5.3 Future Directions

In the state of our work, several future research avenues can be followed:

- The used dataset was conducted in 2000 by DARPA. It is interesting to use more recent data sets that contain new attack instances to test FuzMet with. Also validating the capability of FuzMet by using real attack scenarios designed particulary to prove its effectiveness is required.

- The alert relationship metrics defined in section 3.3.6 is based on the similarity between alert. It is interesting to test different relationship strategies such as the one that is based on the prerequisites and consequences of the attacks.(these were discussed in section 2.3.4).

- As discussed in section 2.2.1, anomaly-based IDSs generate alerts whenever the traffic differs from the norm by a certain threshold. Investigating the usefulness of FuzMet as an anomaly-based IDS is an interesting direction since no work has been done in this regard.

- Providing an analytical study of the alert prioritization, rescoring mechanism, the configuration problem of the identified metrics, and the Fuzzy-logic rules and inference engine is an appropriate extension to this research. By doing so, we can validate the conducted results by comparing the results carried out by the simulations with the results carried out by the analytical models.

- Investigating the alert analysis component that includes attack distance, occurrence time, and response plan is considered for our future works.

Bibliography

[1] http://www.izerv.net/idwg-public/.

[2] S. AXELSSON. The Base-Rate Fallacy and the Difficulty of Intrusion Detection. *ACM Transactions on Information and System Security*, 3(3):186–205, 2000.

[3] R.G. Bace and P. Mell. *Intrusion Detection Systems*. US Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 2001.

[4] N.A. Bakar and B. Belaton. Towards Implementing Intrusion Alert Quality Framework. *Proceedings of the First International Conference on Distributed Frameworks for Multimedia Applications (DFMA'05)-Volume 00*, pages 198–205, 2005.

[5] T. Crothers. *Implementing Intrusion Detection Systems: A Hands-on Guide for Securing the Network*. Wiley, 2003.

[6] KDD Cup. Data. *Available on: http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html, August*, 2003.

[7] F. Cuppens. Managing alerts in a multi-intrusion detection environment. *Proceedings of the 17th Annual Computer Security Applications Conference*, 32, 2001.

[8] D. Curry and H. Debar. Intrusion detection message exchange format(idmef). www.rfc-editor.org/rfc/rfc4765.txt, 2007. RFC.

[9] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. *COMPUT. NETWORKS*, 31(8):805–822, 1999.

[10] H. Debar, A. Wespi, and P.T. R&D. Aggregation and Correlation of Intrusion-Detection Alerts. Springer, 2001.

[11] R. Deraison et al. The nessus project. *Web page at http://www.nessus.org*, 2003.

[12] D. Dittrich. The stacheldrahtdistributed denial of service attack tool. *The DoS Project's" trinoo" distributed denial of service attack tool. The" Tribe Flood Network" distributed denial of service attack tool The" mstream" distributed denial of service attack tool http://www. washington. edu/People/dad.*

[13] Muhammad Abedin et al. Vulnerability analysis for evaluating quality of protection of security policies. In *QoP '06: Proceedings of the 2nd ACM workshop on Quality of protection*, 2006.

[14] K.K. Frederick. Ronald W. Ritchey Stephen Northcutt. Lenny Zeltser. Scott Winters. *INSIDE-Network Perimeter Security.*

[15] R.A. Hanneman. Introduction to Social Network Methods. *University of California*, 2001.

[16] http://insecure.org/nmap/.

[17] http://www.securityfocus.com/archive/1. Bugtraq.

[18] http://www.securityfocus.com/vulnerabilities.

[19] K. Julisch. Clustering intrusion detection alarms to support root cause analysis, 2003.

[20] H. Koike and K. Ohno. SnortView: visualization system of snort logs. *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 143–147, 2004.

[21] MIT Lincoln Lab. 2000 darpa intrusion detection scenario specific datasets. 2000.

[22] K. Labib. Computer Security and Intrusion Detection. *Crossroads*, 11(1):2–2, 2004.

[23] W. Li, L. Zhi-tang, L. Jie, and L. Yao. A novel algorithm SF for mining attack scenarios model. *Proceedings of the IEEE International Conference on e-Business Engineering*, pages 55–61, 2006.

[24] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. A data mining analysis of RTID alarms. *Computer Networks*, 34(4):571–577, 2000.

[25] P. Meil, T. Grance, et al. NVD national vulnerability database.

[26] E. MIRADOR. Mirador: a cooperative approach of IDS. *Poster prØsentØ au 6Øme European Symposium on Research in Computer Security (ESORICS). Toulouse, France, octobre*, 2000.

[27] H.T. Nguyêñ and E.E. Walker. *A First Course in Fuzzy Logic*. Chapman & Hall/CRC, 2006.

[28] P. Ning, Y. Cui, and D.S. Reeves. Constructing attack scenarios through correlation of intrusion alerts, 2002.

[29] P. Ning, Y. Cui, D.S. Reeves, and D. Xu. Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):274–318, 2004.

[30] 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)*, pages 95–114, 2002.

[31] P.A. Porras, M.W. Fong, and A. Valdes. A mission-impact-based approach to infosec alarm correlation. *Recent Advances in Intrusion Detection: 5th Internatonal Symposium, RAID 2002, Zurich, Switzerland, October 16-18, 2002: Proceedings*, 2002.

[32] X. Qin and W. Lee. Statistical Causality Analysis of INFOSEC Alert Data. Springer, 2003.

[33] A. Singhal. Chapter 3 INTRUSION DETECTION SYSTEMS.

[34] A. Siraj and RB Vaughn. Multi-level alert clustering for intrusion detection sensor data. *Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American*, pages 748–753, 2005.

[35] DEFCON Capture the Flag (CTF) contest. DEFCON. 2000. *Available at http://www.defcon. org/html/defcon-8-post.html. Archive accessible at http://wi2600.org/mediawhore/mirrors/ shmoo/.*

[36] Fuzzy Logic Toolbox. http://www.mathworks.com/products/fuzzylogic.

[37] A. Valdes and K. Skinner. Probabilistic Alert Correlation. *Recent Advances in Intrusion Detection: 4th International Symposium, RAID 2001, Davis, CA, USA, October 10-12, 2001: Proceedings*, 2001.

[38] C. Vulnerabilities. MITRE Common Vulnerabilities and Exposures (CVE). *http://cve.mitre.org/.*

[39] C. Vulnerabilities. Vulnerability and virus information- Secunia). *http://secunia.com/.*

[40] C. Vulnerabilities. Internet Security Systems- IBM). *http://www.iss.net/.*

[41] FileZilla Unspecified Buffer Overflow Vulnerability. accessed on July 2007. *http://nvd.nist.gov/nvd.cfm?cvename=CVE-2007-0315.*

[42] FileZilla Unspecified Buffer Overflow Vulnerability. accessed on July 2007 . *http://secunia.com/advisories/20086.*

[43] W. Wang. *Fuzzy Logic Toolbox for Use with MATLAB.* MathWorks, Inc Natick, Mass, 1998.

[44] M. Wood and M. Erlinger. Intrusion Detection Message Exchange Requirements. draft-ietf-idwg-requirements-05. txt, February, March 2007.

[45] W. Yan, E. Hou, and N. Ansari. Extracting Attack Knowledge Using Principal-subordinate Consequence Tagging Case Grammar and Alerts Semantic Networks.

Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)-Volume 00, pages 110–117, 2004.

[46] J. Yu, Y.V.R. Reddy, S. Selliah, S. Kankanahalli, S. Reddy, and V. Bharadwaj. TRINETR: An Intrusion Detection Alert Management System. pages 235–240. IEEE Computer Society Washington, DC, USA, 2004.