Received: June 12, 2019

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Quality Framework to Improve IDS Performance Through Alert Post-Processing

Ali Mohamed Riyad[1]* **Mohammed Saleem Irfan Ahmed[2]** **Husni Hamad Almistarihi[2]**

[1]*Departement of Computer Science, EMEA College of Arts and Science, Kondotty, Kerala, India*
[2]*Department of Computer and Information Sciences, College of Science and Arts, Al Ula, Madhina, Saudi Arabia*
*Corresponding author's Email: amriyad@gmail.com

Abstract: An intrusion detection system is one of the network security tools installed to monitor suspicious activity in the network and act as a last line of defense. It normally notifies about the skeptical activity occurred in the network using sensors by sending alarms to the administrator. However, the IDS present in the large network generates not only a large number of alerts but also abundant false alerts. These generated alerts are very difficult to handle as it increases the burden for the network administrator and also pulls down the performance of the defense system. In order to overcome the issue, various countermeasures have been proposed. Commonly, to increase the quality of alerts, the alerts are post-processed in such a way that the false alerts are filtered out thereby refining the performance of the IDS defense. In this paper, we propose an IDS quality framework using alert post-processing techniques to separate out the false alerts generated by various sensors in the network. At low level alert post-processing, the priority scores are assigned based on the quality measures to filter the irrelevant alerts having less significance. At high level alert post-processing, higher level operations such as alert aggregation, clustering, and hyper alert correlation have been carried out to minimize the number of alerts and the high level report consisting of significant alerts is presented to the administrator. Experiments have been conducted using DARPA 2000 dataset to assess the performance of the proposed system. The system has produced pleasing results than many of the existing methods with 95% of alert reduction rate, 99% of completeness and 100% of soundness towards enlightening the quality of the alerts generated by the IDS.

Keywords: Intrusion detection system, False alert, Alert prioritization, Filtering, Alert aggregation, Alert clustering, Alert correlation.

1. Introduction

Computer networks and devices are considered to be an important asset for sharing files and other information. Due to the increase in the usage of the network, the security of these resources has become a major concern in this digital era. Tremendously, several critical activities are carried out through computer networks. Abundant security tools exist in protecting the computer networks and its resources from undesirable harmful traffic.

Among these tools, intrusion detection systems (IDS) play a significant role in protecting the network. The ability of the IDS is to warn the network administrator by generating alerts on identifying suspicious activities in the network. These malicious activities can be generally categorized as internal (caused by authorized user intentionally or unintentionally) and external (real alerts caused by unauthorized users) [1]. As it is difficult for IDS to differentiate the unintentional and intentional attacks, it largely generates false alerts. Meanwhile, in some cases, false alerts are also produced by misconfiguration of network devices and duplicate alerts generated by more than one sensors form the underlying network [2].

As the alerts are massive in numbers, analyzing and identifying the true alerts manually is certainly a time-consuming process. Though several measures have been taken in reducing false alerts, the most effective way is to post-processing the generated alerts [3]. Typically, IDS obtains input from various sensors and hosts in the network. This generates a

huge amount of alerts with low quality [4]. A preliminary low level operation includes alert filtering that filters out the enormous duplicate alerts for the same event that are produced by various numbers of sensors deployed in the network in a distributed fashion.

After eliminating the low priority alerts, the alerts are passed over to the higher level operations that include various processes such as aggregation and clustering to reduce the number of alerts by merging a group of similar alerts, alert correlation to find real attacks and to eliminate other alerts from further processing. Several methods and tools have been proposed by the researchers with the aim of improving the quality of the alerts produced by the IDS. However, the methods focus on specific needs and constraints that drive on specific tasks such as prioritization, fusion, and correlation. Also, the performance of the existing systems highly depends on the quality of the input given to the system.

In this paper, we proposed a comprehensive IDS quality framework using two level post-processing for improving the quality of the alerts produced by the IDS. At a lower level, the alerts are normalized and the priority score for the alerts are computed based on which the low quality alerts are filtered out. Higher level post-processing includes three components. Aggregation and merging of alerts are carried out in which similar alerts are aggregated and the alerts having similar characteristics are clustered together in the first and second component. The third component correlates the alerts by computing the correlation score between the alerts to find the dependency between them. Finally, significant alerts are presented to the administrator.

The paper is organized as follows. Section 2 discusses the solution from the literature related to the problem specified. Section 3 explains the proposed IDS quality framework along with all its components. Experimental results are discussed in section 4 and the paper is concluded in the conclusion section.

2. Related works

The idea of intrusion detection was first introduced by James P. Anderson in his technical report on intrusion detection systems [5]. From the last decade, research in the field of IDS and its security has attracted a number of researchers. Post-processing of intrusion alerts is comparatively a new field. There are various techniques used for eliminating false positives during post-processing of alerts. The most prominent solution is to apply a filter to reduce the number of false alerts. A filter with three components was proposed where each one computes the score for the alerts, and based on the combined score, the false alerts are filtered out [6-7]. However, the results depend on the quality of input data. Neural networks and the fuzzy logic based method was introduced in filtering low quality alerts but the method requires necessary training [8].

Another idea was proposed wherein the quality of the alerts are measured using network topology [9-10]. The methods analyze the alerts using quality parameters that include correctness, accuracy, reliability, and sensitivity. The main demerit of this method was that it employs inflexible parameter. Aggregation and correlation operations had been used as a component for handling IDS alerts [11]. Regrettably, the method is not tested with any datasets. An alert management approach with enhanced alert verification and alert aggregator modules were proposed to reduce the alerts quantity wherein the method produces inefficient clustering [12]. A correlation based alert processing model was introduced with an ample set of components. This not only produces better results but also increases the complexity and processing time of the system [13].

Complex theory based approach for hypothesizing missed security events was suggested [14]. Two parameters such as prerequisites and consequences of different types of attacks were utilized to correlate alerts. This method works on the predefined correlations between alerts for which prior analysis must be carried out. An attribute similarity clustering for reducing the alarms and reverse causation algorithm for creating a complete attack path centered on the attack association method was anticipated that depends on the professional knowledge base which increases the complexity [15]. Graph based IDS alert correlation method was proposed by correlating the subattacks, unfortunately, the method suffers from false alerts and also missed several attack scenarios [16-17].

An ontology based framework with inference language XSWRL was developed to correlate the attack [18]. The method utilizes multiple agents and sensors to convert the information to ontology. The method is very difficult to deal with new attack types. A similar hybrid approach was proposed using semantic analysis and ontology [19]. While the approach was considered to be novel, the work undergoes a major inadequacy during online correlation. A new alert correlation method based on complex Bayesian network was introduced which works without experts' knowledge [20]. Conversely, the method was computationally complex to implement. Another post-processing method comprising of prioritization and scalable distance-based clustering steps was introduced. The method

lacks in performance compared with other significant existing algorithms and is compared only with the DBSCAN clustering [21]. A multi-step attack scenario reconstruction based method was offered but the method runs on a predefined attack paradigm [22].

Similar methods were proposed in detecting false alerts using correlation wherein the methods execute only with prior knowledge about the attacks [23, 24]. These methods fail to handle novel attacks as they do not have any past experience. A new idea based on attack scenario reconstruction using attack semantics was suggested [25] with a major requirement of mature ontology related to the intrusion detection domain in which updating the ontology is a major concern.

3. Proposed IDS quality framework

The proposed IDS quality framework is based on applying post-processing steps on the alerts generated from various IDS of the network. The main aim of the post-processing is to reduce the false positive of the underlying intrusion detection system. This highly improves the quality of the results produced, as the increase in the false positives is the main issue that degrades the performance of an intrusion detection system. The overall architecture of the proposed work is divided into two broad categories in which the first one is the low level alert post-processing and the second is the high level alert post-processing. Maximum elimination of false positive alerts is the most challenging issue in intrusion detection research as it contributes to the quality alarms. This dual level mechanism has been introduced for processing the received alerts further collected from various IDS of the network and thereby eliminating the false positive alerts in an extensive way.

Instead of accepting the generated alerts as such, the alerts are processed to analyze the trustworthiness of the generated alerts and finally converts them into valuable intrusion reports. Initially, the alerts are normalized and duplicate alerts are filtered out based on which the priorities are issued to the alerts in the low level alert post-processing. In the high level alert post-processing, similar alerts are aggregated based on the alert attributes, then the similar aggregated alerts are clustered and finally, the hyper alert correlation carried out for extracting the useful alerts and based on which the reports are given to the user. The framework comprises of several components and the detailed framework is depicted in Fig. 1.

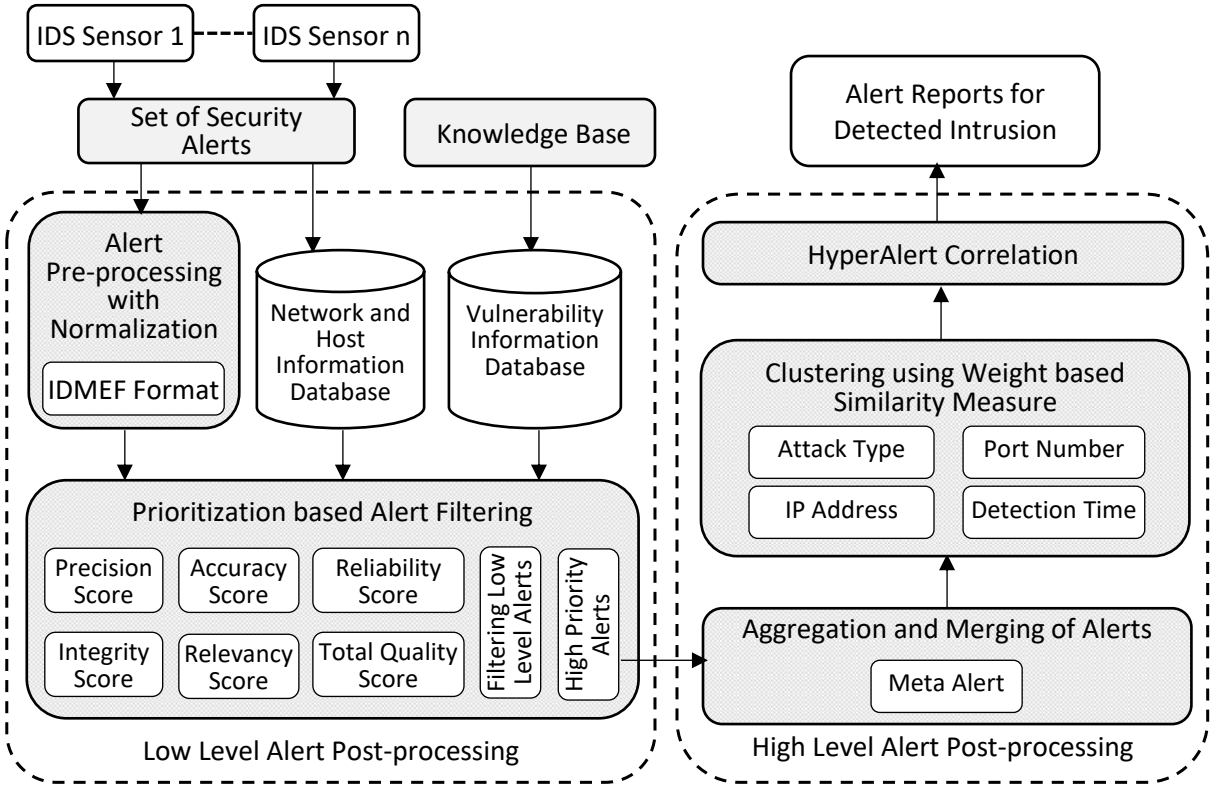

Figure. 1 Detailed structure of IDS quality framework with alert post-processing

Vulnerability information database: The vulnerability information database is formed mainly using the National Vulnerability Database (NVD) [26] which is primarily based on CVE (Common Vulnerabilities and Exposures) [27]. These databases are up to date. Vulnerability databases are freely available for download. The databases are updated regularly for the best performance of the relying processes.

Network and host information database*:* Network and host configuration collector gathers all the details of the network and the devices installed in the network and stores in the information in this database. It contains the information about the devices, the ports they listen and details of various software installed including the operating systems. Details of the vulnerabilities and exploits in the network and hosts are stored in this database. It also monitors the live hosts. Here, the destination IP, destination port and the vulnerabilities in the target device software and services are checked from the respective database to identify whether it is an attack or not.

3.1 Alert pre-processing with normalization

The intrusion detection system implemented on the network comprises of several sensors and obviously, these sensors generate different alerts for the same intrusion identified on the network or they may generate different alerts for different intrusions. These alerts collected from various IDSs in the network may have different formats. However, for further processing, the alerts are to be converted to a common format for applying prioritization. Thus, the primary goal of this step is to transform the collected alerts into a common form which makes further processing easy. The common format to be employed in the system must be a standardized format. From the literature, Intrusion Detection Message Exchange Format (IDMEF) [28] is the most widely used data format which is mainly used for exchanging the information and for successful interaction between the intrusion detection systems and the management systems. It is designed in such a way that the alert class sends the alerts from the analyzer to the manager. Some of the significant attributes that signify the alert information communicated between the analyzer and the manager is listed in Table 1. Based on the alert information, the alerts are encoded with a common IDMEF format [29]. The main parameters used to store the information about alerts are the alert number, sensor number, time stamp, attack type, IP addresses and port numbers of source and destination machines.

Table 1. Significant alert information

Alert attribute	**Information**
Analyzer (AID)	Alert originated Analyzer ID
CreateTime (CT)	Alert generated time
DetectTime (DT)	Intrusion detected time
AnalyzerTime	Alert sent time
Source	Details about attacks' origin
Target	Details about attacks' target.
Classification	Name of the attack
Assessment	Evaluation of attack severity
Additional Data	Other information of the attack

However, the common names for the alerts are taken from the vulnerability information database that is maintained by the framework. Though the alerts are converted to the common format, there is a possibility of incomplete information on the significant attributes. Basically, time related details along with the source and target are most important for processing the alerts. Thus the main goal is to pre-process the alerts by cleaning the alert database in such a way that the incomplete details or missing values are filled with appropriate values [13]. The missed alert created can be filled with attack detection time if it is not null, else filled with analyzer time at which the alert is sent. Similarly, the detection time can be filled with the creation time, a source and a target can be filled with analyzer ID.

3.2 Prioritization based alert filtering

The alerts are analyzed based on the quality measures. The score is computed for each alert and based on the scores, the low level alerts are filtered out. The significant alerts are prioritized based on the scores computed through quality measures [9]. Some of the quality measures employed in the proposed work are precision, accuracy, reliability, integrity, and relevance of the alerts.

Precision (P): It verifies the status of the destination host at the time of alert creation and assigns the score *0* or *1*.

Accuracy (A): It validates whether the destination is vulnerable to attack specified by the alert and assigns the score *0* or *1* accordingly.

Reliability (R): It verifies whether the rules and the attack signatures specified in the sensor are frequently updated and assigns the score *0* or *1*.

Integrity (I): It identifies the past history of the alerts generated from the particular sensor and assigns the score *0* or *1*.

Relevancy (Re): It is verified by identifying the evidence of the attack (particular attack pattern) in the particular destination host [30] and assigns the score *0* or *1*.

Once the scores are assigned for all the criteria, the weights are assigned for each criterion and based on which the total quality scores are computed for the alerts. Among these criteria, the precision and relevancy are most significant as it specifies the generated alert is true. So the weights can be assigned equally as *1*. The accuracy plays second significant as it partially signifies the created alert is true as it depicts whether the destination host is vulnerable to attack and the score can be assigned as *0.6*. Finally, reliability and integrity are less significant as the value *1* may also lead to false alarm rate and thus the weights are assigned as *0.2* each. Thus the final score can be computed using the formula given in Eq.(1) in which the total score *(TS)* is scaled between *0* and *1*.

$$TS = \frac{P + (A \times 0.6) + (R \times 0.2) + (I \times 0.2) + Re}{3} \quad (1)$$

where, *P*, *A*, *R*, *I* and *Re* represent the scores of precision, accuracy, reliability, integrity, and relevancy respectively.

The pseudocode for computing the final score using prioritization based filtering of the alerts are given in Fig. 2. The input parameters are accessed form vulnerability information database and network and host information database. Based on the total quality score, the priority can be assigned for the alerts. If the total alert score is *0* which implies the alert is completely false positive and if the total alert score is *1* which implies that the alert is completely true positive. However, if the value lies above *0* and *0.3* then it is almost false alert whereas if the score lies above *0.3* and *0.6* then the alert is feasibly true positive and if the score lies above *0.6* and below *1*, then the alert is almost true positive. Thus the true and almost true alerts are given as an input for the high level alert post-processing steps.

3.3 Aggregation and merging of identical alerts

One of the major issues is that the occurrence of the same attacks may be identified by several analyzers in which they produce similar identical alerts. These similar alerts are grouped into representative alerts and are also known as meta alerts. Similar alerts are aggregated only if they have same Source_IP (*SIP*), Target_IP (*TIP*) and Attack_type (*AID*) with the closer time period as it belongs to the same event. Thus, the temporal data and other information pertaining to the alerts are compared for aggregation. The maximum closer period can be set with a threshold based on which the alerts can be aggregated.

```
Global Vul_Info_db, NW_Host_Info_db
Function Score_Computation (alert parameters)
//Precision Score
  If a destination host at the time of alert is 'online'
      Assign a precision score as 1 else 0.
//Accuracy Score
  If a destination host network is vulnerable
      Assign accuracy score as 0.25.
  If a destination host running OS is vulnerable
      Assign accuracy score as 0.25.
  If a destination host opened port is vulnerable
      Assign accuracy score as 0.25.
  If destination host running application is vulnerable
      Assign accuracy score as 0.25.
//Reliability Score
  If the attack signature is updated before 12hrs of alert
      Assign reliability score as 1.
  Elseif attack signature is updated before 24hrs of alert
      Assign reliability score as 0.75.
  Elseif attack signature is updated before 7days of alert
      Assign reliability score as 0.5.
  Else Assign reliability score as 1.
//Integrity Score
  If the past history contains true alert for the attack
      Assign integrity score as 1.
  Elseif past history contains more no. of true attacks
      Assign integrity score as 0.5.
  Else Assign integrity score as 1.
//Relevancy Score
  If an attack pattern is found in a destination host
      Assign relevancy score as 1.
  Else Assign integrity score as 0.
Compute total score as in Eq. (1)
End Function
```

Figure. 2 Prioritization based alert filtering pseudocode

However, if the difference between the detection time of alert crosses the given threshold, then the alerts are considered as different.

The aggregation can be done by replacing the similar alerts with an aggregated alert that contains *SIP*, *TIP*, *AID* and the earliest time of the detection time along with the number of alerts (*NOA*) in the aggregated alert. This is because, as the alert belongs to the same event, the later time indicates that the event is identified with a time delay. In the proposed work, the threshold value for the detection time is set as 3 seconds. The pseudocode for the alert aggregation is given in Fig. 3. Each alert *a* in the input alert set *A* is compared with all the other alerts. A new single alert is created if they have same Source_IP (*SIP*), Target_IP (*TIP*) and Attack_type (*AID*) with the closer time period of 3 seconds. Here *NOA* represents the number of alerts in the aggregated alert.

```
Function Alert_Aggregation (alert set A)
For each alert a in the input alert_set A
//compare the alert attributes of a and other alerts A[i]
 If (a.SIP=A[i].SIP ∧ a.TIP=A[i].TIP ∧ a.AID=A[i].AID)
  If (Time_diff(a, A[i])< threshold) then
   New_alert r; r.NOA=1;
   For each attribute attr
    If attr=detect_time then
     r.detect_time = min(a.detect_time, A[i].detect_time)
    r.attr = a.attr;
    r.NOA = r.NOA+1;
   End For
End For
End Function
```

Figure. 3 Pseudocode for alert aggregation

After Aggregation, the process of merging identical alerts from different analyzers at various point of time is carried out. The alerts having similar attributes but different detection time with the threshold of 3 minutes which is relaxed when compared to aggregation are merged together to form hyper alerts. Thus the merging of two alerts is possible only if the following condition is met.

If (a1.SIP==a2.SIP ∧ a1.TIP==a2.TIP)
If (a1.DT ≠ a2.DT ∧ a1.AID ≠ a2.AID)
Merge(a1, a2);

where *a1* and *a2* are alerts, *SIP* and *TIP* represent the source and the target machines, *AID* represents the attack *ID* and *DT* represents the attack detection time.

3.4 Clustering using weighted similarity measure

The clustering component is employed to group similar alerts into groups. The feature based similarity is used to group similar alerts. In the proposed method, the features or alert attributes used for computing the similarity value are Attack_type (*AID*), Source_IP (*SIP*), Target_IP (*TIP*), Source_port (*SP*) and Target_port (*TP*). The agglomerative clustering is applied to group similar alerts. Each alert is considered as the singleton cluster and based on which the similarity between the clusters is computed. The pseudocode for clustering is given in Fig. 4.

The similarity score is computed by aggregating the similarity score of alert attributes where the final score always lies between *0* and *1*. Only the similarities that are greater than the given threshold is considered in which the cluster having a maximum similarity score is merged. The process continues until the similarity scores between all the clusters are less than the threshold value. The similarity between

```
Function Clustering (aggregated alert set A)
Let each aggregated alert as a Singleton Cluster
While True do
  For each pair of clusters
    Compute the similarity value
  End For
  Find the cluster pair having max. similarity score
  If sim(Ci, Cj)> threshold then
     Merge Ci and Cj
  Else Terminate loop
End While
End Function
```

Figure. 4 Pseudocode for clustering

the cluster having more than one alert is computed by taking the mean value.

The similarity measures between the alert attributes are computed based on the similarity between the attack type, IP address, and port used by the attack in of the alerts.

Similarity between attack: The similarity between the attack_type or *AID* of the alerts is computed by comparing the attack classes. This can be represented as in Eq. (2).

$$sim_{AID}(A_1, A_2) = \begin{cases} 1, A_1(AID) \equiv A_2(AID) \\ 0, Otherwise \end{cases} \quad (2)$$

A_1 and A_2 represent the two alerts or singleton clusters. If the clusters have more than one alert, then each alert in one cluster is compared with all the alert in the other cluster under comparison and the final score is the average of all the scores. Here for simplicity, the formula is given for the singleton clusters having single alerts.

Similarity between port number: For computing the similarity measure between the port number, they are grouped according to Internet Assigned Numbers Authority (IANA) [31]. The listing has three categories such as well-known ports (0-1023), registered ports (1024-49151), and dynamic or private ports (49151-65535). Port numbers within the same group are considered to be closer than the port numbers of a different group. Thus, well-known ports are closer to registered ports than private ports as the service provided by them are similar. The formula to compute the similarity score between the source ports $sim_{SP}(A, B)$ is given in Eqs. (3) and (4).

$$sim_{SP}(A,B) = \begin{cases} 1, A(SP) = B(SP) \\ 0.75, \delta p(A(SP)) = \delta p(B(SP)) \\ 0.5, \delta p(A(SP) \wedge \delta p(B(SP)) \in \{0,1\} \\ 0, Otherwise \end{cases} \quad (3)$$

$$\delta p(SP) = \begin{cases} 0, \ SP \in [0,1023] \\ 1, \ SP \in [1024,49151] \\ 2, \ SP \in [49152,65535] \end{cases} \quad (4)$$

Here, *A* and *B* are two alerts, *SP* represents the source port ID. $\delta p(SP)$ represents the port numbers of different groups. Similarly, the similarity score can be computed for the target port (*TP*) as $sim_{TP}(A,B)$.

Similarity between IP address: As port numbers, the IP address has several hierarchies of categories such as unicast that includes public and private, multicast, broadcast, etc. Accordingly, assigning the similarity scores based on these categories are very difficult. Thus, the hamming distance between the IP address is computed for the measuring the similarity. As a scaling factor, the distance is divided by the total number of bits compared. The formula to compute the similarity score between the source IP address of two alerts A_1 and A_2 $sim_{SIP}(A_1, A_2)$ is in Eq. (5).

$$sim_{SIP}(A_1, A_2) = \frac{d_H(A_1(SIP), A_2(SIP))}{Total\ No.of\ bits\ compared} \quad (5)$$

here, SIP represents source IP addresses. The same formula can be applied for the target IP address (TIP) if two alerts A_1 and A_2 $sim_{TIP}(A_1, A_2)$ for computing similarity.

Based on the attribute significance, the weights are assigned to them. The attribute weights and attribute similarity scores are used to compute the total similarity between the alerts. The weights are assigned in such a way that the values lie between *0* and *1* with a constraint that the sum of weights must be *1*. As the attack type is most significant, the weight assigned for the attribute w_{AID} is *0.3*. The next important attributes are *SIP* and *TIP* and thus the weights assigned for the attributes w_{SIP} and w_{TIP} are *0.2* each. Finally, the least important attributes *SP* and *TP* are assigned a weight w_{SP} and w_{TP} as *0.15* each. The final similarity score based on *AIP, SIP, TIP, SP* and *TP* for the alerts A_1 and A_2 is given in Eq. (6).

$$sim(A_1, A_2) = \sum_{i=1}^{5} sim_i \times w_i \quad (6)$$

where *i* value represents the alert attributes *AID, SIP, TIP, SP, TP*. Thus the alerts are clustered based on the computed similarity score and in the proposed method the threshold value for clustering is set as *0.4* as it provides optimal results.

3.3 Hyper alert correlation

Generally, hyper alerts provide high level abstraction or patterns about attacks identified by the generated alerts. Hyper alert correlation tries to identify the relationship between the generated hyper alerts in the form of cluster. The clustered alerts are correlated to gather knowledge about the alerts. To compute the correlation between the alerts four parameters such as the number of common SIP pairs (*A*), common TIP pairs (*B*), common SIP & TIP pairs (*C*) and common TIP & SIP pairs (*D*) are employed. The correlation represented as *Corr* between any two clusters containing hyper alerts A_1 and A_2 as given by [32] is given in Eq. (7).

$$Corr(A_1, A_2) = \frac{A + B + C + D}{2(m+n)} \quad (7)$$

Here, $A = UIPair(A_1(SIP), A_2(SIP))$ implies the unique identical pairs of source IP address that are common for the alerts A_1 and A_2. $B = UIPair(A_1(TIP), A_2(TIP))$ implies the unique identical pairs of the target IP address that are common for A_1 and A_2. $C = \mathrm{UIPair}(\mathrm{A_1(SIP)}, \mathrm{A_2(TIP)})$ implies the unique identical pairs where the source IP address in A_1 and target IP address in A_2 are same. $D = \mathrm{UIPair}(\mathrm{A_1(TIP)}, \mathrm{A_2(TIP)})$ implies the unique identical pairs in alerts A_1 and A_2 where the target IP address in A_1 is the same as the source IP address in A_2. The variable *m* and *n* is the total number of alerts in the clusters. Each pair of clusters are compared to compute the correlation score and finally, the total correlation score is computed by summing the correlation score of the cluster with all the other cluster. Obviously, the correlation between the same cluster can be set as *0*. The pseudocode for hyper alert correlation is shown in Fig. 5 where CM represents the correlation matrix and C represents the set of clusters.

Based on the total correlation value and the individual correlation score, the score of the cluster that deviates more than other clusters can be neglected and the other cluster having higher correlation value can be presented to the administrator. Based on the correlation value, the correlation graph can also be generated. The clusters can be arranged in such a way that their scores are arranged in descending order. Each cluster is

```
Function Correlation (C number of clusters A[])
   Create 2D correlation matrix CM[][]
   //For each pair of clusters
   For i=1 to C do
       For j=1 to C do
           If i != j then
               CM[i][i] = Corr(Ai,Aj)
           End IF
        End For
    End For
    Compute Total Correlation Values for the cluster
    Present Top most cluster to the administrator
    Create correlation graph based on correlation score
End Function
```

Figure. 5 Pseudocode for hyper alert correlation

removed and the correlation with other cluster are compared in which the cluster having maximum correlation value is related to the first cluster. This process continues until all the clusters are mapped.

4. Experimental analysis

Experimental analysis has been carried out to justify the improved performance of the proposed system.

4.1 Experimental setup

The hosts in the network are configured with core i7 3.4Ghz with 8GB RAM. Operating system used is 64 bit Windows 8.1. The network is connected to the outside network through the router and firewall. We use IDS in the form of mobile agents which employs the ensemble classification technique [33]. The multiple sensors collect the data and store it in the databases where mobile analysis agents in JADE environment [34] analyses the events. The generated alerts are normalized and prioritization based filtering is applied to the collected alerts. The quality alerts with higher priorities are aggregated using the alert aggregation algorithm. These hyper alerts were served to the correlation process for generating correlated attack graphs. Alert post-processing programs are written in Java and SQL Server database is used for storage purpose.

4.2 Dataset used

The DARPA 2000 1.0 dataset is used for the evaluation process which contains network traffic data from two sensors (inside and DMZ) with two LLDOS 1.0 and LLDOS 2.0.2 scenario specific datasets which were created by MIT Lincoln Labs [35]. In order to experiment the two scenarios, Tcpreplay 2.3.2 [36] is used as the replay tool to feed the sensors in the network which are eventually processed by various phases ending with identification of significant alert and alert correlation graphs. The dataset contains attack with multiple steps. The steps are as follows.

1. In the initial stage, the attacker scans the whole network to obtain the active hosts.
2. The attacker use ping to exploit sadmint on the hosts found in the previous step.
3. Now, the attacker uses the sadmint vulnerability to attain the root privilege.
4. After getting the root access, the attacker installs the DDOS malware in the compromised hosts and makes the machine DDOS master.
5. The master machine launches the DDOS attack.

4.3 Experimental Results

The experiments are performed using inside and DMZ traffic of LLDOS 1.0 and LLDOS 2.0.2 datasets. The prioritization based filtering of alerts has been performed and are analyzed by varying the threshold value from *0.5* to *0.8*. The number of alerts filtered (false positive) and the number of high priority alerts (true positive) selected for the next phase along with their rates are shown in Table 2.

The rates of reduction for the proposed method with the threshold value as *0.5, 06, 0.7* and *0.8* are nearly 70%, 80%, 87%, and 93%. Thus, if the threshold is set with a minimum value as 0.5, the high priority alerts include false positive.

At the same time, if the threshold is set with maximum value as *0.7* and *0.8*, then higher priority alerts (true positive) will get filtered out. Thus for implementation, the threshold is set as *0.6* which provide optimum reduction rate. The detailed report on a varying range of alert scores (AS) for the four datasets LLDOS 1.0 Inside, LLDOS 1.0 DMZ, LLDOS 2.0.2 Inside, LLDOS 1.0 DMZ with 0.6 as threshold is given in Table 3. The number of alerts filtered at each step such as priority based filtering during low level alert post-processing and aggregation & merging, clustering and hyper alert correlation during high level alert post-processing are analyzed. A detailed picture is given in Table 4. The number of input and output alert with detection rate at each stage for the four datasets LLDOS 1.0 Inside, LLDOS 1.0 DMZ, LLDOS 2.0.2 Inside, LLDOS 1.0 DMZ are given in Table 4. Thus, the total alert reduction rates for the four datasets such as LLDOS 1.0 Inside, LLDOS 1.0 DMZ, LLDOS 2.0.2 Inside, LLDOS 1.0 DMZ are 94.98%, 94.05%, 93.48%, and 94.60% respectively.

Table 2. Separation of low and high priority alerts by varying threshold value

Dataset	No. of Input Alerts	No. of Low priority Alerts	No. of High priority Alerts	Reduction Rate	Accepted Rate
Threshold value for High Priority Alerts = 0.5					
LLDOS 1.0 Inside	578	405	173	70.07%	29.93%
LLDOS 1.0 DMZ	941	654	287	69.50%	30.50%
LLDOS 2.0.2 Inside	276	188	88	68.12%	31.88%
LLDOS 2.0.2 DMZ	389	271	118	69.67%	30.33%
Threshold value for High Priority Alerts = 0.6					
LLDOS 1.0 Inside	578	466	112	80.62%	19.38%
LLDOS 1.0 DMZ	941	743	198	78.96%	21.04%
LLDOS 2.0.2 Inside	276	221	55	80.07%	19.93%
LLDOS 2.0.2 DMZ	389	313	76	80.46%	19.54%
Threshold value for High Priority Alerts = 0.7					
LLDOS 1.0 Inside	578	511	67	88.41%	11.59%
LLDOS 1.0 DMZ	941	819	122	87.04%	12.96%
LLDOS 2.0.2 Inside	276	242	34	87.68%	12.32%
LLDOS 2.0.2 DMZ	389	342	47	87.92%	12.08%
Threshold value for High Priority Alerts = 0.8					
LLDOS 1.0 Inside	578	548	30	94.81%	5.19%
LLDOS 1.0 DMZ	941	882	59	93.73%	6.27%
LLDOS 2.0.2 Inside	276	258	18	93.48%	6.52%
LLDOS 2.0.2 DMZ	389	361	28	92.80%	7.20%

Table 3. A detailed report on a varying range of alert scores

Dataset	No. of Input Alerts	No. of Low Priority Alerts			No. of High Priority Alerts	
		False Alert AS = 0	Almost False Alert 0<AS<0.3	Possibly False Alert 0.3≤AS<0.6	Almost True Alert 0.6≤AS<1	True Alert AS = 1
LLDOS 1.0 Inside	578	21	211	234	100	12
LLDOS 1.0 DMZ	941	11	333	399	190	8
LLDOS 2.0.2 Inside	276	8	96	117	46	9
LLDOS 2.0.2 DMZ	389	9	128	176	59	17

Table 4. Alert reduction rate at each stage for DARPA datasets

Stages	Dataset	Input Alerts	Output Alerts	Reduction Rate	Dataset	Input Alerts	Output Alerts	Reduction Rate
Prioritization	LLDOS 1.0 Inside	578	112	80.62%	LLDOS 2.0.2 Inside	276	55	80.07%
Aggregation		112	76	32.14%		55	37	32.73%
Clustering		76	48	36.84%		37	24	35.14%
Alert correlation		48	29	39.58%		24	18	25.00%
Total Alert Reduction Rate				*94.98%*	*Total Alert Reduction Rate*			*93.48%*
Prioritization	LLDOS 1.0 DMZ	941	198	78.96%	LLDOS 2.0.2 DMZ	389	76	80.46%
Aggregation		198	127	35.86%		76	52	31.58%
Clustering		127	87	31.50%		52	38	26.92%
Alert correlation		87	56	35.63%		38	21	44.74%
Total Alert Reduction Rate				*94.05%*	*Total Alert Reduction Rate*			*94.60%*

The clustering of meta alerts are carried out and the threshold value is fixed as *0.4* which has effective results when compared with other values. The clustered hyper alerts are correlated and the correlation matrix is created in which the correlation value between the clusters will be the elements of the matrix. Based on the correlation values, the final correlation is computed by summing the values and the value that deviates from other values will be removed. Thus the performance of the proposed method is evaluated using completeness (C_m) and soundness (S_m) measures [16]. The performance of the alert correlation is analyzed for the DARPA datasets and are presented in Table 5.

Table 5. Performance analysis for alert correlation for DARPA datasets

Details	LLDOS 1.0		LLDOS 2.0.2	
	Inside	DMZ	Inside	DMZ
Correlated alerts	29	56	18	21
Correctly correlated alerts	29	56	18	20
Incorrectly correlated alerts	0	0	0	1
Related alerts	29	57	18	22
Missed alerts	0	1	0	1
Completeness	100%	99.11%	100%	90.94%
Soundness	100%	100%	100%	95.24%

Table 6. Alert reduction rate comparison using DARPA datasets

Methods	Alert reduction
Intrusion alert quality framework [10]	98.03%
Intrusion detection alert correlation [13]	53.00%
Correlation based alert detection [14]	81.21%
Inverse causal correlation [15]	86.77%
Correlation, prioritization clustering [21]	89.50%
Proposed work	94.28%

The proposed method provides better results with 100% completeness for LLDOS 1.0 Inside, LLDOS 2.0.2 Inside and produces 99.11% completeness for LLDOS 1.0 DMZ and 90.94 % completeness for LLDOS 2.0.2 DMZ. Also, the method produces 100% soundness for LLDOS 1.0 Inside and DMZ and LLDOZ 2.0.2 Inside and 95.24% soundness for LLDOS2.0.2 DMZ.

The number of alerts used for the study differs from one researcher to another and therefore it is not possible to compare the proposed method with other methods. However, the existing low level alert post-processing methods and high level post-processing methods are compared individually with the proposed method based on the percentage of values.

The final rate of high priority alerts filtered by the proposed method is compared with the final rate of high priority alerts filtered by the various existing methods. The alert reduction rate for the proposed method and existing methods are listed in Table 6.

From the analysis, the proposed method has better alert reduction rate of 94.28% than all the existing methods under comparison except Intrusion alert quality framework having the false reduction rate of 98.03%. This is because the several true alerts are get filtered out in Intrusion alert quality framework. Though the proposed method has better performance, few of the true alerts having alert scores less than *0.5* is also considered as false alert with the error rate of 0.4%. Thus additional quality measures have to be incorporated to increase the accuracy. Similarly, the correlated alerts of the proposed method are compared with existing methods based on the completeness and soundness measures using DARPA LLDOS 1.0 dataset. The comparison is presented in Table 7.

Table 7. Performance comparison

Techniques	C_m (%)	S_m (%)
Correlation based [14]	93.96	95.06
Abstracted correlation graph [16]	86.5	100
Grammar-based Approach [17]	96.41	100
Ontology based method [18]	92.2	NA
Ontology based method [19]	100	99.7
Iterative alert correlation [20]	96.72	100
Attack pattern modelling [22]	87.1	86.27
Ontology based method [25]	100	97.22
Proposed Technique	99.11	100

From the analysis, the soundness of 100% is produced by the proposed method which is highly promising than the existing techniques. Meanwhile, the 99.11% completeness of the proposed method is better than many of the existing methods except ontology based methods.

Thus, the method has to be analysed in such a way to improve its performance towards 100% completeness. Though the proposed method provides better theoretical results for the alerts generated by the IDS, the method lacks in providing visual representation about the attack scenario and correlation graph has to be implemented. Also, the method does not handle the attacks missed by the IDS and the system must be implemented in the real environment. These limitations provide a room for enhancement of the work in the future.

5. Conclusion

Due to the large size of networks, the IDS has been installed with several sensors and many of them will generate alerts for the same suspicious activity which ends up with the low quality results. Post-processing of alerts is thus become more substantial as it improves the quality of results produced by the IDS. In this paper, an IDS quality framework using alert post-processing techniques to filter out the false alerts generated by various sensors have been proposed. The post-processing operations such as priority based filtering, aggregation, clustering, and correlation have been employed to analyze the quality of alerts and to remove the trivial alerts. Our implementation converts the large set of alerts with false positive into a reasonable amount of quality alerts. Experimental analysis has been made with

DARPA datasets in which the proposed system produced 94.28% of alert reduction thereby increasing the completeness and soundness to 99% and 100% respectively. From the results, it is found that the proposed approach is highly promising and capable of producing good results. The future study aims at improving the proposed method in achieving the 100% completeness.

References

[1] D. Nandasana and V. Barot, "A Framework for Database Intrusion Detection System", In: *Proc. of International Conf. on Global Trends in Signal Processing, Information Computing and Communication*, pp.74-78, 2016.

[2] X. Lu, X. Du, and W. Wang, "Network IDS Duplicate Alarm Reduction using Improved SNM Algorithm", In: *Proc. of International Conf. on Image, Vision and Computing*, pp.767-774, 2018.

[3] G.P. Spathoulas and S.K. Katsikas, "Enhancing IDS Performance through Comprehensive Alert Post-processing", *Computers & Security*, Vol. 37, pp.176-196, 2013.

[4] K. Alsubhi, E. Al-Shaer, and R. Boutaba, "Alert Prioritization in Intrusion Detection Systems", In: *Proc. of Network Operations and Management Symposium*, pp.33-40, 2008.

[5] J.P. Anderson, *Computer Security Threat Monitoring and Surveillance*, Technical report, James P.Anderson Company, Fort Washington, Pennsylvania, 1980.

[6] G.P. Spathoulas and S.K. Katsikas, "Reducing False Positives in Intrusion Detection Systems", *Computers & Securi*ty, Vol.29, No.1, pp.35-44, 2010.

[7] T. Chyssler, S. Nadjm-Tehrani, S. Burschka, K. Burbeck, "Alarm Reduction and Correlation in Defence of IP Networks", In: *Proc. of International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp.229–34, 2004.

[8] R. Alshammari, S. Sonamthiang, M. Teimouri, and D. Riordan, "Using Neuro-Fuzzy Approach to Reduce False Positive Alerts", In: *Proc. of Conf. on Communication Networks and Services Research*, pp. 345–349, 2007.

[9] N.A. Bakar and B. Belaton, "Towards Implementing Intrusion Alert Quality Framework", In: *Proc. of International Conf. on Distributed Frameworks for Multimedia Applications*, pp.198-205, 2005.

[10] N.A. Bakar, B. Belaton, and A. Samsudin, "False Positives Reduction Via Intrusion Alert Quality Framework", In: *Proc. of International Conf. on Networks Jointly held with International Conf on Communication*, Vol.1, pp.547-552, 2005.

[11] H. Debar and A. Wespi, "Aggregation and Correlation of Intrusion-Detection Alerts", In: *Proc. of International Sym. on Recent Advances in Intrusion Detection*, 2001.

[12] T.H. Nguyen, J. Luo, and H.W. Njogu, "An Efficient Approach to Reduce Alerts Generated by Multiple IDS Products", *International Journal of Network Management*, Vol.24, No.3, pp.153-180, 2014.

[13] F. Valeur, G. Vigna, C. Kruegel, and R.A. Kemmerer, "Comprehensive Approach to Intrusion Detection Alert Correlation", *IEEE Transactions on Dependable and Secure Computing*, Vol.1, No.3, pp.146-169, 2004.

[14] P. Ning, Y. Cui, D.S. Reeves, and D. Xu, "Techniques and Tools for Analyzing Intrusion Alerts", *ACM Transactions on Information and System Security*, Vol.7, No.2, pp.274-318, 2004.

[15] J. Liu, S. Li, and R. Zhang, "Algorithm of Reducing the False Positives in IDS based on Correlation Analysis", In: *IOP Conference Series: Materials Science and Engineering*, Vol.322, No.6, pp.1-5, 2018.

[16] S.O. Al-Mamory and H.L. Zhang, "Scenario Discovery using Abstracted Correlation Graph", In: *Proc. of International Conf. on Computational Intelligence and Security*, pp.702-706, 2007.

[17] S.O. Al-Mamory and H. Zhang, "IDS Alerts Correlation using Grammar-Based Approach", *Journal in Computer Virology*, Vol.5, no.4, p.271, 2009.

[18] W. Li, S. Tian, "An Ontology based Intrusion Alerts Correlation System", *Expert Systems with Applications*, Vol.37, pp.7138–7146, 2010.

[19] S. Saad and I. Traore, "Semantic Aware Attack Scenarios Reconstruction", *Journal of Information Security and Applications*, Vol.18, No.1, pp.53-67, 2013.

[20] R. Anbarestani, B. Akbari, and F. Fathi, "An Iterative Alert Correlation Method for Extracting Network Intrusion Scenarios", In: *Iranian Conf. on Electrical Engineering*, pp.684-9, 2012.

[21] J. Andrew and G.J.W. Kathrine, "An Intrusion Detection System using Correlation, Prioritization and Clustering Techniques to Mitigate False Alerts", *Advances in Big Data and Cloud Computing*, pp.257-268, 2018.

[22] Z. Liu, C. Wang, and S. Chen, "Correlating Multi-Step Attack and Constructing Attack

Scenarios based on Attack Pattern Modelling", In: *Proc. of International Conf. on Information Security and Assurance*, pp.214-219, 2008.
[23] Z. Zali, M.R. Hashemi, and H. Saidi, "Real-time Intrusion Detection Alert Correlation and Attack Scenario Extraction based on the Prerequisite-Consequence Approach", *International Journal of Information Security*, Vol. 4, No. 2, pp.125-36, 2013.
[24] A. A. Ramaki and A. Rasoolzadegan, "Causal Knowledge Analysis for Detecting and Modeling Multi-Step Attacks", *Security Communications Network*s, Vol.9, No.18, pp. 6042-65, 2016.
[25] M. Barzegar and M. Shajari, "Attack Scenario Reconstruction using Intrusion Semantics", *Expert Systems with Applications*, Vol. 108, pp.119-133, 2018.
[26] https://nvd.nist.gov/vuln/data-feeds
[27] Common Vulnerabilities and Exposures, http://www.cve.mitre.org/, 2003.
[28] D.Curry and H. Debar, *Intrusion Detection Message Exchange Format Data Model and extensible Markup Language (XML) Document Type Definition*, Internet draft, 2002.
[29] M.H. Bhuyan, D.K. Bhattacharyya, and J.K. Kalita, *Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools*, Springer, 2017.
[30] C. Kruegel, W. Robertson, and G. Vigna, "Using Alert Verification to Identify Successful Intrusion Attempts", Praxis der Informationsverarbeitung und Kommunikation, Vol.27, No.4, pp.219-227, 2004.
[31] S.E. Coull, F. Monrose, and M. Bailey, "On Measuring the Similarity of Network Hosts: Pitfalls, New Metrics, and Empirical Analyses", *In NDSS*, 2011.
[32] N. Hoque, D.K. Bhattacharyya, and J.K. Kalita, "An Alert Analysis Approach to DDoS Attack Detection", In: *Proc. of International Conf on Accessibility to Digital World*, pp.33-38, 2016
[33] A.M. Riyad and M.S. Irfan Ahmed, "An Ensemble Classification Approach for Intrusion Detection", *International Journal of Computer Applications*, Vol. 80, pp. 37-42, 2013.
[34] JADE Board, *JADE Security Add-On Guide*, Administrator's Guide of the Security Add-On, Version 28-February-2005, sJADE 3.3, TILAB. http://jade.tilab.com/download/jade/license/jade-download/
[35] MIT Lincoln Laboratory, DARPA 2000 Intrusion Detection Scenario Specific Data Sets, 2000, https://www.ll.mit.edu/r-d/datasets.
[36] A Turner, "Tcpreplay: Pcap Editing and Replay Tools for *nix 2010", http://tcpreplay.appneta.com/wiki/tcpreplay-man.html