An Improved Framework for Intrusion Alert Correlation

Huwaida Tagelsir Elshoush and Izzeldin Mohamed Osman

***Abstract*—Alert correlation analyzes the alerts from one or more collaborative Intrusion Detection Systems (IDSs) to produce a concise overview of security-related activity on the network. The process consists of multiple components, each responsible for a different aspect of the overall correlation goal. The sequence order of the correlation components affects the correlation process performance. The total time needed for the whole process depends on the number of processed alerts in each component. This paper proposes a new correlation framework based on a model that reduces the number of processed alerts as early as possible by discarding the irrelevant and false alerts in the first phases. A new component is added to deal with the unrelated alerts. A modified algorithm for fusing the alerts is also proposed. The intruders' intention is grouped into attack scenarios and thus used to detect future attacks. The contribution of this paper includes an enhanced new framework for alert correlation, the implementation of the alert correlator model based on the framework, and the evaluation of the model using the DARPA 2000 intrusion detection scenario specific datasets. The experimental results show that the correlation model is effective in achieving alert reduction and abstraction. The performance is improved after the attention is focused on correlating higher severity alerts.**

***Index Terms*—Intrusion detection, Alert correlation, Alert reduction, Alert correlation datasets.**

I. Introduction

IDSs may cooperate to complement each other's coverage. Even when different detection techniques are used, they analyze each other's alerts and reduce false positive alerts [13][16][19][22].

Deploying multiple IDSs might generate a huge number of alerts, where many are redundant, irrelevant and false positive alerts. Hence, data reduction, such as alert aggregation, alert filtering and reducing false alerts, without losing valuable information is essential [4][14][15].

Alert correlation is defined as a process that contains multiple components with the purpose of analyzing alerts and providing high-level insight view on the security state of the network surveillance [4][22]. Thus correlation aims to relate a group of alerts to build a big picture of the attacks, and hence can be used to trace each attack to its source.

The core of this process consists of components that implement specific function, which operate on different spatial and temporal properties [2].

The correlation components are effective in achieving alert reduction and abstraction. Research show that the effectiveness of each component depends heavily on the nature of the data set analyzed [2]. Moreover, the performance of the correlation process is significantly influenced by the topology of the network, the characteristics of the attack, and the available meta-data.

Since alerts can refer to different kinds of attacks at different levels of granularity, the correlation process cannot treat all alerts equally. Instead, it is necessary to provide a set of components that focus on different aspects of the overall correlation task. Some components, see Fig.1, e.g. those at the initial and second units, implement general functionality that is applicable to all alerts, independent of their type. Other components (e.g. in the third unit) are responsible for performing specific correlation tasks that cannot be generalized for arbitrary alerts, but for certain class of alerts.

Thus, one cannot, in general, determine a ranking among components with respect to their effectiveness. Each component can contribute to the overall analysis. Therefore, the most complete set of components should be used [2].

This paper focuses on reordering the correlation components such that redundant, irrelevant and false alerts are reduced as early as possible for the purpose of reducing the number of processed alerts to enhance the performance. The unrelated alerts that are not correlated are dealt with in a separate component. Hence, the overall effectiveness of the correlation process is improved in the proposed model.

The rest of this paper is organized as follows: The proposed correlation model is overviewed in section 2. In Section 3, the evaluation of alert correlators is discussed together with the factors affecting the alert reduction rates. Section 4 discusses the implementation of our model. A detailed description of the components of the correlation process and the results of applying each component to the sample datasets, DARPA 2000, is explained. Our model is also compared with previous correlation systems. Section 5 reviews some of the related work. Section 6 concludes the paper and recommends some future work.

II. The Proposed Correlation Model Overview

Our system architecture, see Fig. 1, is composed of ten main components, namely *normalization*, *preprocessing*, *prioritization*, *alert verification*, *alert fusion*, *focus recognition*, *uncorrelated removal*, *multi-step correlation*, *intention recognition*, and *impact analysis*.

The normalization and preprocessing components are contained in the *data normalization unit*. The *filter-based correlation unit* contains the prioritization and the verification components. The alert fusion, focus recognition, uncorrelated removal and multi-step correlation components are in the *data reduction unit*.

In the *normalization* component, alerts that are generated by multiples IDSs are collected and stored in a database

Manuscript received March 23, 2012; revised April 15, 2012.

H. T. Elshoush is with the Department of Computer Science, School of Mathematical Sciences, University of Khartoum, Sudan, e-mail: htelshoush@uofk.edu.

I. M. Osman is with Sudan University of Science and Technology Khartoum, Sudan, email: izzeldin@acm.org.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

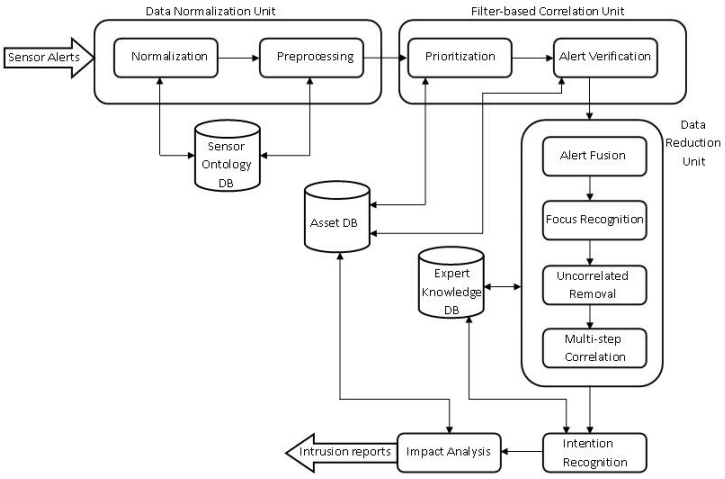

Fig. 1. The Proposed Correlation Model.

before they are modeled and converted into a standard format called Intrusion Detection Message Exchange Format (IDMEF). Then data *preprocessing* is required in order to clean the data, do feature extraction and selection, and finally deal with any incomplete or missing data.

The *filter-based correlation* unit either assigns a priority to each alert or identifies irrelevant alerts. Thus, alerts are ranked based on their severity level in order to discover the high and low risks alerts depending on information in the asset DB. In the *alert verification* component, alerts are verified to determine the false positives and invalid alerts.

Redundant alerts are fused based on similarity functions [22] in the *alert fusion* component in the *data reduction unit*. In the *focus recognition* component, alerts are aggregated then classified using feature similarity. Unrelated and false alerts tend to be random and will not correlate, hence uncorrelated alerts are removed by *uncorrelated removal* component. The last component of the data reduction unit, *multi-step correlation*, is expected to achieve substantial improvement in the abstraction level and data reduction. In this component, priori information of the network topology, known scenarios, etc are provided by the expert knowledge DB, and hence high level patterns are specified .

In *intention recognition* component, relevant behavior is grouped into attack scenarios to extract attack strategy and plan recognition. In the final component, *impact analysis*, the asset DB is consulted to determine all other services that are dependent on a specific target and then this information is added to the alert as a likely consequence of the attack.

III. Evaluation of Alert Correlators

A. Evaluating Alert Correlators

Correlation tools are evaluated "as a whole", without an assessment of the effectiveness of each component of the analysis process. As a result, it is not clear if and how the different parts of the correlation process contribute to the overall goals of correlation [4].

According to [4], the evaluation of the correctness of the correlation process has to be performed manually as:

- there exist very few shared datasets for correlation evaluation, and
- generating alerts from raw data might be risky and may bias the correlation evaluation process,
- no truth files are associated with datasets, which makes it difficult to know if the correlated alerts are representative of a meaningful grouping of detected attacks.

The effectiveness of IDS sensors are evaluated using the detection rate and the false positive rate. These values cannot be easily calculated for an alert correlator due [4]:

- The false positive is not clearly defined for a correlator, as it may receive a false positive (which is an IDS sensors output) and therefore draws a wrong assumption that an attack took place. Thus, validation of the input alerts is necessary.
- If there exists an attack creating a single low-level alert, no correlation will be performed and thus this may be considered a missed attack and hence results in a reduction in the detection rate.

B. Factors Affecting the Alert Reduction Rate

- Experiments showed that the achieved alert reduction rate (RR) is highly dependent on the features of the dataset being processed [4].
- A particular dataset that experience a high reduction rate during one correlation step might achieve poor reduction rates during other steps [4].

Hence, when testing correlation systems, it is important to calculate reduction rates for different datasets. Furthermore, it is useful to calculate the reduction rate during each correlation step in addition to the total reduction rate [4].

IV. Implementation and Experimental Results

In this section, the implementation of an intrusion alert correlator based on the proposed framework is explained.

In our implementation, we used Microsoft SQL Server 2005 as the relational database to store the alert datasets, the intermediate data, and the analysis results of each component as well as the correlated alerts.

The alert log files generated by RealSecure IDS of the DARPA simulation network is used [9].

A. Experiments on DARPA 2000 Datasets

DARPA 2000 [8] is a well-known IDS evaluation dataset created by the MIT Lincoln Laboratory. It consists of two multistage attack scenarios, namely LLDOS 1.0 and LLDOS 2.0.2. Both attack scenarios contain a series of attacks in which an attacker probes, breaks-in, installs the components necessary to launch a Distributed Denial of Service (DDoS) attack against an off-site server, with different stealth levels. LLDOS 2.0.2 is a bit more sophisticated than LLDOS 1.0.

Each scenario includes the network traffic collected from both the demilitarized zone (DMZ) and the inside part of the evaluation network. We have performed six experiments, four on the proposed model and two on the comprehensive approach model [2].

In both scenarios, the attacker tries to use the vulnerability of Sadmind RPC service and launches buffer overflow attacks against the vulnerable hosts. The attacker installs the mstream distributed DOS software after he breaks into the hosts successfully. Finally, the attacker launches DDOS

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

attacks from the victims. The differences between these two scenarios lay in two aspects: First, the attacker uses IPSweep and Sadmind Ping to find out the vulnerable hosts in LLDOS 1.0 while DNS HInfo is used in LLDOS 2.0.2. Second, the attacker attacks each host individually in LLDOS1.0, while in LLDOS2.0.2, the attacker breaks into one host first and then fans out from it [10][21].

B. Analysis and Performance Evaluation of Proposed Correlation Model

1) Data Normalization Unit:

- **Normalization**
  The interaction of collaborators from distinct IDS poses various requirements. Dependent on the level of collaboration, these include a common language, protocol or even a complete framework. Hence, in a distributed environment with heterogeneous IDSs, specifying a common alert format is fundamental for providing interpretability among IDSs. Moreover, high level analysis such as alert correlation also requires that the alerts that are processed to be in a generic format. There exists a variety of exchange formats; prominent examples include IDMEF and IODEF. Therefore, in this component, all attributes of each sensor alert will be translated into a common data format, IDMEF in particular [3][8]. The IDMEF data model is implemented using a Document Type Definition (DTD) to describe Extensible Markup Language (XML) documents. IDMEF is also an object-oriented representation and a Unified Modeling Language (UML) model. If only one type of sensor was used, then that sensor's format could be used as the standard format.
- **Preprocessing**
  The goal of the preprocessing component is to supply, as accurately as possible, missing alert attributes that are important for later correlation components. Reducing the dimensionality of the data improves the correlation process considerably and detection accuracy is improved as a result of removing irrelevant and redundant features. Hence, this component handles null values, missing and incomplete data [26]. Feature selection is used to reduce the alert attributes, as it is revealed from recent research [23][24][25] that feature selection improves detection accuracy and performance. The type information is useful because it allows one to group attacks with a similar impact together.
  In both scenarios, there are 45 features, of which only 7 features were extracted, namely EventID, timesec, SrcIPAddress, DestPort, DestIPAddress, OrigEventName, and SrcPort. The date attribute was represented in date/time format, and we converted it to time in seconds (represented as timesec). 5 alerts, representing incomplete data, were removed in all datasets, except for the inside segment of scenario 1.0., see Table I.

2) Filter-based Correlation Unit: The primary goal is to reduce the number of alerts to be correlated by eliminating false, irrelevant and low risk alerts. False alerts need to be handled at an early stage as they will have negative impact on the correlation result, and moreover the number of processed alerts will be greatly reduced.

TABLE I
IMPACT OF PREPROCESSING COMPONENT ON LLDOS SCENARIOS

	DMZ 1.0	Inside 1.0	DMZ 2.0.2	Inside 2.0.2
Input alerts	891	922	430	494
Output alerts	886	922	425	489

TABLE II
IMPACT OF PRIORITIZATION COMPONENT ON LLDOS SCENARIOS

	DMZ 1.0	Inside 1.0	DMZ 2.0.2	Inside 2.0.2
Input alerts	886	922	425	489
Output alerts	188	167	54	71
Reduction Rate	78.78%	81.89%	87.29%	85.48%

- **Prioritization**
  In this component, depending on the information contained in the asset DB, high and low risks alerts are identified. The low risks that do not have significant effect on the protected system are discarded. By alert ranking, the data fed into the remaining components is reduced as only the high and medium risks which are of higher importance or relevance is considered in the later components. That is, based on asset information, only relevant risky alerts are processed. Thus as acknowledged in [1], when the number of processed alerts is reduced, the performance is improved as the total time needed for the whole process depends on the number of processed alerts in each component.
  Based on asset information, groups may collaborate in sending only valuable information or information related to their protected system, so filtering might be done to irrelevant information.
  The ranking/priority of alerts of LLDOS scenarios from [7] is used and thus low risk alerts are discarded, and only the medium and high risk alerts are sent to the next component. After implementing the proposed model, the resulting alerts are shown in Table II.
- **Alert Verification**
  It distinguishes between successful and failed intrusion attempts. The verification of an attack can be accomplished by extending ID rules with an expected "outcome" of the attack that describes the visible and verifiable traces [2][6].
  Verification can be performed using both *passive* and *active* techniques, and each approach requires different support from the ID infrastructure [2][6][12].
  Because some sites might be interested in failed attack attempts, an alert should be differentiated from a successful instance. This alert reflects the missing contextual information that the IDS would require to determine a failed attack. Thus, it would be better if IDSs rules can be modified to include such information. As no sufficient information was found about the asset DB, this component could not be implemented.

3) Data Reduction Unit: Similar alerts are fused and thus data is reduced by eliminating data redundancies, and irrelevant, false and unreal alarms using alert correlation, as false alerts are less likely to be correlated.

- **Alert Fusion**
  Algorithm 1 is the alert fusion component method.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Algorithm 1: Alert Fusion Algorithm

```
Parameter window-size, fuse-window, thread-window
Global alert-queue, fuse, thread

fuse(alert)
al ← get a:alert with lowest start-time from alert-queue where
if alert.analyzer ∩ a.analyzer is empty and all overlapping attributes
except start-time, end-time, analyzer, alertid are equal then
    fuse
    window-size = fuse-window
else
    if alert.victimhosts = a.victimhosts and alert.attackerhosts =
    a.attackerhosts then
        thread
        window-size = thread-window
    end if
end if
if al¬null then
    replace al in alert-queue with fuse-merge(alert, al)
else
    add alert to alert-queue
    remove all a:alert from alert-queue where
    a.start-time < (alert.start-time - window-size)
    pass removed alerts to next correlation component
end if

fuse-merge(alert1, alert2)
r ← new alert
r.alertid ← get unique-id()
r.start-time ← min(alert1.start-time, alert2.start-time)
r.reference ← (alert1.alertid ∪ alert2.alertid)
if fuse then
    r.end-time ← min(alert1.end-time, alert2.end-time)
    for each attr:attribute except start-time, end-time, reference, alertid
    do
        r.attr ← alert1.attr ∪ alert2.attr
    end for
    fuse ← false
else
    if thread then
        r.end-time ← max(alert1.end-time, alert2.end-time)
        r.analyzer = alert1.analyzer ∪ alert2.analyzer
        thread ← false
    end if
end if
if alert1.name = alert2.name then
    r.name ← alert1.name
else
    r.name ← "Attack Thread"
end if
for each attr:attribute except start-time, end-time, reference, analyzer,
alertid do
    if alert1.attr = alert2.attr then
        r.attr ← alert1.attr
    else
        r.attr ← null
    end if
end for
return r
```

end

It combines a series of alerts that refer to attacks launched by one attacker against a single target. This component removes duplicates created by the independent detection of the same attack by different sensors, and also correlates alerts that are caused by an attacker who tests different exploits against a certain program or that runs the same exploit multiple times to guess correct values for certain parameters (e.g., the offsets and memory addresses for a buffer overflow) [2][6][12]. The alert fusion component keeps a sliding timewindow of alerts. The alerts within the timewindow are stored in a time-ordered queue. When a new alert arrives, it is compared to the alerts in the queue, starting with the alert with the earliest timestamp. A fusion match is found if all overlapping attributes are equal and the new alert is produced by a different sensor. The timestamp of the meta-alert is assigned the earlier of the sub-alerts times. On the other hand, attack threads are constructed by merging alerts with equivalent source and target attributes that occur in a certain temporal proximity but the alerts need not be produced by different sensors. The timestamp of the meta-alert is assigned the earlier of the two start-times and the later of the two end-times. The value of the time window should be a good trade-off between a small value, which would cause several attack threads to go undetected, and a larger value, which would slow down the system by requiring the component to keep a large number of alerts in the queue. Table III shows the results of our implementation.

TABLE III
IMPACT OF ALERT FUSION COMPONENT ON LLDOS SCENARIOS

	DMZ 1.0	Inside 1.0	DMZ 2.0.2	Inside 2.0.2
Input alerts	188	167	54	71
Output alerts	92	110	34	45
Reduction Rate	51.06%	34.13%	37.04%	36.62%

TABLE IV
IMPACT OF FOCUS RECOGNITION COMPONENT(ONE-TO-MANY)

	DMZ 1.0	Inside 1.0	DMZ 2.0.2	Inside 2.0.2
Input alerts	92	110	34	45
Output alerts	42	57	23	27
Reduction Rate	56.52%	48.18%	32.35%	40%

TABLE V
IMPACT OF FOCUS RECOGNITION COMPONENT(MANY-TO-ONE)

	DMZ 1.0	Inside 1.0	DMZ 2.0.2	Inside 2.0.2
Input alerts	42	57	23	27
Output alerts	31	28	5	24
Reduction Rate	26.19%	50.88%	78.26%	11.11%

- **Focus Recognition**
  The focus recognition component has the task of identifying hosts that are either the source or the target of a substantial number of attacks. This is used to identify denial-of-service (DoS) attacks or port scanning attempts. More specifically, this component aggregates the alerts associated with single hosts attacking multiple victims (called a one-to-many scenario) and single victims that are targeted by multiple attackers (called a many-to-one scenario) [2][6][12].
  The one-to-many scenario has two tunable parameters: the size of the timeout, which is used for the initial window size, and the minimum number of alerts for a meta-alert to be generated. In our experiments, the minimum number of alerts in a meta-alert was two.
  We first applied one-to-many focus recognition on DARPA datasets, then followed by many-to-one focus recognition. Some horizontal scan and multi-scan attacks were observed. Tables IV and V show the reduction rates of the focus recognition component. DMZ in scenario 2.0.2. shows a great reduction rate as is expected being a multistage attack scenario.
- **Uncorrelated Removal**
  As shown in [10], alert correlation can be used to differentiate between false and true alerts. False alerts and unreal alarms tend to be more random than actual alerts, and are less likely to be correlated. Thus, based on this founding, we intentionally removed the uncorrelated alerts, resulting in Table VI, which shows great RR.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE VI
IMPACT OF UNCORRELATED REMOVAL COMPONENT ON LLDOS SCENARIOS

	DMZ 1.0	Inside 1.0	DMZ 2.0.2	Inside 2.0.2
Input alerts	31	28	5	24
Output alerts	6	7	3	5
Reduction Rate	80.65%	75%	40%	79.17%

TABLE VII
IMPACT OF MULTI-STEP CORRELATION COMPONENT ON LLDOS SCENARIOS

	DMZ 1.0	Inside 1.0	DMZ 2.0.2	Inside 2.0.2
Input alerts	6	7	3	5
Output alerts	6	5	2	4
Reduction Rate	0%	28.57%	33.33%	20%

- **Multi-step Correlation**
  The goal of this component is to identify high-level attack patterns that are composed of several individual attacks. The high-level patterns are usually specified by using some form of expert knowledge [2][11][12][22]. Specifically, it may also associate network-based alerts with host-based alerts that are related to the same attack, called *attack session reconstruction*. This requires either real-time access to the systems being protected or very detailed auditing information in order to map network traffic to host activity. Identifying relations between these two types of alerts is difficult because the information that is present in the alerts differs significantly. While multistep attack analysis may not generate the same level of alert reduction achieved by other components, it often provides a substantial improvement in the abstraction level of the produced meta-alerts. Newly detected attack strategies are fed back to the expert knowledge DB to keep it updated.
  Results of implementation is shown in Table VII.

4) Intention Recognition: Intention or plan recognition is the process of inferring the goals of an intruder by observing his/her actions [4]. It deduces strategies and objectives of attackers based on attack scenarios that are output by correlation systems. Failed attacks can be useful to know so that they can be avoided in the future.

Using alert correlation, the intruders' relevant behavior can be grouped into attack scenarios, and later on, their attack strategy or plan can be extracted and fed back to update the expert knowledge DB.

5) Impact Analysis: The final component contextualizes the alerts with respect to a specific target network. It determines the impact of the detected attacks on the operation of the network being monitored and on the assets that are targeted by the malicious activity.

Impact analysis requires a precise modeling of the relationships among assets in a protected network and requires constant monitoring of the health of those assets. Insufficient information deters the implementation of this component.

C. Summary of Experimental Results

Fig. 2 shows the effect of our correlation model on LLDOS 1.0 and 2.0.2 scenarios. There is a substantial drop in the number of alerts in the priority component for all datasets. This reduces the number of processed alerts considerably and thus improves the correlation process performance.

TABLE VIII
TOTAL ALERT REDUCTION FOR THE PROPOSED MODEL

	DMZ 1.0	Inside 1.0	DMZ 2.0.2	Inside 2.0.2
Input alerts	891	922	430	494
Output alerts	6	5	2	4
Reduction Rate	99.33.%	99.46%	99.53%	99.19%

TABLE IX
NO. OF PROCESSED ALERTS USING PROPOSED MODEL FOR SCENARIO 2.0.2

	Prepr.	Prio.	Fus.	1:M	M:1	U.Rem.	Multi	total
DMZ	425	54	34	23	5	3	2	546
Inside	489	71	45	27	24	5	4	665

TABLE X
NO. OF PROCESSED ALERTS USING COMPREHENSIVE APPROACH FOR SCENARIO 2.0.2

	Prepr.	Fus.	1:M	M:1	Multi.	Prio.	total
DMZ	425	241	63	46	44	5	824
Inside	489	276	71	44	33	7	920

Table VIII shows the total alert reduction for each dataset. Tables IX and X show the number of processed alerts in each component for our proposed model compared to the Comprehensive approach discussed in [2]. Since the processing time is proportional to the number of processed alerts, hence Fig. 3 shows that our model gives better results.

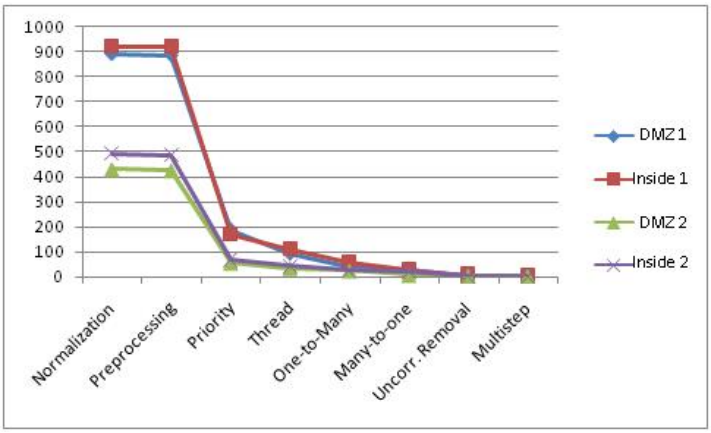

Fig. 2. Effect of Correlation Model on LLDOS 1.0 and 2.0.2 Scenarios.

V. RELATED WORK

In [1], Taha et al presented an agent-based alert correlation model. A learning agent learns the nature of dataset to select which components to be used and in which order. They proved that their method achieved minimum alerts to be processed on each component, depending on the dataset, and minimum time for correlation process. Their method differs from ours, in that they have learning agent, and we specify an order of the components which gives better performance by processing less number of alerts, hence minimum correlation time as only the high risk alerts are processed.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

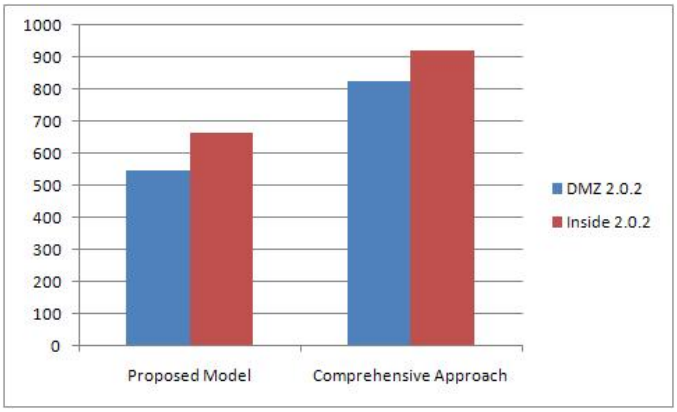

Fig. 3. Comparison of Processing Time of Proposed Correlation Model and Comprehensive Approach on LLDOS Scenario 2.0.2.

Valeur et al in [2] presented a complete comprehensive set of components. Their experiments demonstrated that the effectiveness of each component is dependent on the data sets being analyzed, and each component can contribute to the overall performance.

From the analysis in [5], researchers propose an improved solution for an alert correlation technique based on six capabilities criteria identified which are capabilities to perform alert reduction, alert clustering, identify multi-step attacks, reduce false alert, and to detect known and unknown attacks.

In [17], a decentralized, multi-dimensional alert correlation algorithm for CIDSs is proposed. A two-stage algorithm, implemented in a fully distributed CIDS, first clusters alerts locally at each IDS, before reporting significant alert patterns to a global correlation stage.

Ghorbani et al in [18] showed an overall view of the applied techniques which have been used for different components of an alert correlation framework.

Meinel et al in [20] identified the data storage and processing algorithms to be the most important factors influencing the performance of clustering and correlation. They proposed and implemented the utilization of memory-supported algorithms and a column-oriented DB for correlation and clustering in an extensible IDS correlation platform.

VI. Conclusion and Future Work

Automation of alert management and analysis is crucial because of the large number of alerts. Alert correlation analyzes the alerts and aims to relate different alerts to build a big picture of the attack, thus giving a high-level view of the security status. The proposed model attempts to minimize the number of processed alerts on each component and thus minimizing the correlation processing time. It removes irrelevant, unreal and false alerts in the early phases of the correlation by reordering the components. Uncorrelated alerts are also dealt with in order to discard irrelevant and false positives. Thus by diverting more resources to deal with high risk/priority alerts to be correlated, the effectiveness of alert correlation is improved.

The experimental evaluation reported in this paper is still preliminary, though it has demonstrated the potential of the proposed model. Further experiments and comparisons with different datasets and a real network dataset will be investigated.

References

[1] A. E. Taha, I. Abdel Ghaffar, A. M. Bahaa Eldin and H. M. K. Mahdi, "Agent Based Correlation Model For Intrusion Detection Alerts," *Published by the IEEE Computer Society*, May 2010.

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

[3] H. Debar, D. Curry and B. Feinstein, "The Intrusion Detection Message Exchange Format (IDMEF)". Available: *http://www.ietf.org/rfc/rfc4765.txt*, 2007.

[4] A. A. Ghorbani, W. Lu and M. Tavallaee, "Network Intrusion Detection and Prevention: Concepts and Techniques," *Published by Springer*, a textbook, 2010.

[5] R. Yusof, S. R. Selamat and S. Sahib, "Intrusion Alert Correlation Technique Analysis for Heterogeneous Log," *IJCSNS International Journal of Computer Science and Network Security*, VOL.8 No.9, pp. 132-138, Sept. 2008.

[6] F. Valeur, "Real-time ID Alert Correlation," *PhD Thesis*, June 2006.

[7] M. M. Siraj, M. A. Maarof and S. Z. M. Hashim, "Intelligent Alert Clustering Model for Network Intrusion Analysis," *Published by ICSRS Publication, Int. J. Advance. Soft Comput. Appl.*, Vol. 1, No. 1, July 2009, ISSN 2074-8523.

[8] MIT Lincoln Laboratory 2000 DARPA Intrusion Detection Scenario Specific Datasets. *http://www.ll.mit.edu/index.html.*

[9] Ning P. TIAA: A Toolkit for Intrusion Alert Analysis. Available from: http://discovery.csc.ncsu.edu/software/correlator/; 2007.

[10] P. Ning, Y. Cui and D. S. Reeves, "Constructing Attack Scenarios through Correlation of Intrusion Alerts," *in Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington D.C.*, pp. 245-254, November 2002.

[11] P. Ning, Y. Cui and D. S. Reeves, "Analyzing Intensive Intrusion Alerts Via Correlation," *in Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002), LNCS 2516, Zurich, Switzerland*, pp. 74-94, October 2002.

[12] C. Kruegel, F. Valeur and G. Vigna, "Intrusion Detection and Correlation - Challenges and Solutions," *Published by Springer*, a textbook, 2005.

[13] P. Ning, S. Jajodia and X. S. Wang, "Intrusion Detection in Distributed Systems - An Abstraction-Based Approach," *Published by Kluwer Academic Publishers*, a textbook, 2004.

[14] G. P. Spathoulas and S. K. Katsikas, "Reducing false positives in intrusion detection systems," *Published by Elsevier Ltd. Computer Security*, 2009.

[15] S. X. Wu and W. Banzhaf, "The Use of Computational Intelligence in Intrusion Detection Systems: A Review," *Published by Elsevier Ltd. Applied Soft Computing Journal*, Vol. 10, pp. 1-35, Jan. 2010.

[16] C. V. Zhou, C. Leckie and S. Karunasekera, "Decentralized multidimensional alert correlation for collaborative intrusion detection," *Published by Elsevier Ltd. Journal of Network and Computer Applications*, Vol. 32, pp. 1106 -1123, Sept. 2009.

[17] C. V. Zhou, C. Leckie and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Published by Elsevier Ltd. Computer Security*, pp. 1-17, June 2009.

[18] R. Sadoddin and A. Ghorbani, "Alert Correlation Survey: Framework and Techniques," Oct-Nov. 2006.

[19] R. Bye, S. A. Camtepe and S. Albayrak, "Collaborative Intrusion Detection Framework: Characteristics, Adversarial Opportunities and Countermeasures," August 2010.

[20] S. Roschke, F. Cheng and C. Meinel, "A Flexible and Efficient Alert Correlation Platform for Distributed IDS," *Fourth International Conference on Network and System Security*, 2010.

[21] Y. Cui, "A Toolkit for Intrusion Alerts Correlation Based on Prerequistes and Consequences of Attacks," *MSc. Thesis*, Dec. 2002.

[22] H. T. Elshoush and I. M. Osman, "Alert Correlation in Collaborative Intelligent Intrusion Detection Systems - A Survey," *Published by Elsevier Ltd. Journal of Applied Soft Computing*, Volume 11, Issue 7, pp. 4349-4365, October 2011.

[23] A. Zainal1, M. A. Maarof1 and S. M. Shamsuddin, "Features Selection Using Rough-PSO in Anomaly Intrusion Detection," 2007.

[24] A. Zainal, M. A. Maarof and S. M. Shamsuddin, "Feature Selection Using Rough Set in Intrusion Detection".

[25] F. Amiri, M. M. R. Yousefi, C. Lucas and A. Shakery "Improved Feature Selection for Intrusion Detection System," *Published by Elsevier Ltd. Journal of Network and Computer Applications*, Jan. 2011.

[26] J. J. Davis and A. J. Clark, "Data Preprocessing for Anomaly-based Network Intrusion Detection: A Review," *Published by Elsevier Ltd. Journal of Computer and Security*, pp. 353-375, October 2011.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)