Alert correlation and aggregation techniques for reduction of security alerts and detection of multistage attack

Faeiz M. Alserhani
College of Computer & Information Sciences, Dep. of Computer Engineering & Networks
Aljouf University
Sakaka Aljouf, Saudi Arabia
fmserhani@ju.edu.sa

***Abstract*— Malicious attacks by intruders and hackers exploit flaws and weakness points in deployed systems through several sophisticated techniques. Consequently, automated detection and timely response systems such as Network Intrusion Detection Systems (NIDS) are urgently needed to detect abnormal activities by monitoring network traffic and system events. The current implementation of NIDS generates huge volumes of alerts overwhelming the security analyst which makes event observation tedious. Hence, an alert correlation and aggregation technique is proposed to provide a complementary analysis to link elementary alerts and provide a more global intrusion view. We have proposed a framework for alert correlation to discover the logical relationships between atomic alerts potentially incorporated in multi-stage attacks and to remove data redundancy. The correlation process is essentially modularized based on an extension of the properties and characteristics of the *requires/provides* model. The aggregation of alerts is based on graph reduction techniques that remove duplication in vertex set and migrating connecting edges to a nominated node. The resulting attack graph consists of nodes representing aggregated alerts and edges representing the casual relationships. The experimental results have showed an efficient capability to detect attack scenarios and to reduce generated security alerts.**

Keywords—Intrusion detection systems; Alert correlation; Alert aggregation; Multi-stage attack

I. INTRODUCTION

Malicious attacks by intruders and hackers exploit flaws and weaknesses in the deployed systems. This is done by several sophisticated techniques cannot be prevented by traditional measures. Hackers are shifting their focus from looking for fame and advertised attacks to profit-oriented activities. The current trends in cyber attacks are hidden, slow-and-low, and coordinated. NIDS are considered to be important security tools to defend against such threats. The effectiveness of any NIDS depends on its ability to recognize different variations of cyber attacks. The current implementation of intrusion detection systems (commercial and open-source) is employing signature-based detection in addition to few simple techniques for statistical analysis. The main task of signature-based systems is to inspect the network traffic and perform pattern matching to detect attacks and generate alerts. A huge number of alerts are generated every day stressing the administrator; this may oversight an actual threat. Quality of these alerts is debatable particularly if the majority is false positives. For this reason, high-level and real-time analysis techniques are needed. This can be achieved by discovering the logical connections between the isolated alerts. It has been practically identified that most of attacker activities consists of multiple steps (attack scenario) and occur in a certain time (attack window). Identification of such strategy can lead to the recognition of attack intensions and also prediction of unknown attacks. Some simple analysis tools have been developed to generalize these alerts based on attack classes [1].

In recent years, alerts clustering and correlation techniques have been employed to provide a global view of attacker's behavior by analyzing low-level alerts produced by the IDS sensors. The main objective of alerts correlation is to build an abstract modeling of alerts by generalizing the detected events instead of the current specific modeling. The constructed inference will progress even in case of unforeseen attacks. Different approaches have been utilized to build the correlation models, and can be categorized into three main disciplines: probabilistic approaches, scenario-based approaches and pre/post conditions approaches. The probabilistic approaches are inspired from anomaly-based intrusion detection systems where prior knowledge is not required. In this category, relations between incurred events are computed statistically providing automatic knowledge acquisition. Data mining, clustering, association rules techniques are examples of these approaches. [2] presented a probabilistic approach to provide unified mathematical framework that perform a partial matching of features. Features are extracted and minimum similarities are computed and weighted. [3] proposed alarm clustering to discover the root causes of different alarms. The aim was to reduce the volume of alarms to manageable size. Even though, these methods are useful for alert fusion and statistical purposes but they fail to discover the causal connection between alerts.

Recently, [4] and [5] employed different data mining algorithms for real-time correlation to discover multi-stage attacks. Off-line attack graph is constructed using manual or automatic knowledge acquisition and then attack scenarios are recognized by correlating the collected alerts in real-time. The incoming step of an attack can be predicted after detection of few steps of attack in progress. In [4] association rule mining algorithm is used to generate the attack graph from different attack classes based on historical data. "Candidate attack sequences" are determined using a sliding window. In [6] AprioriAll algorithm which is a sequential pattern matching technique is used to generate correlation rules based on temporal and content constraints. [6] adopted a classical sequential mining method GSP[7] to find the maximal alerts sequence and then to discover the attack strategy. The limitation of their work is the use of only attack class and temporal as features.

On the other hand, scenario-based modeling is based on manual knowledge acquisition that specifies intrusion steps by experts. Scenario libraries are used to build the model and to discover the logical connections between alerts. LAMBDA [8] is an intrusion specification language to describe the conditions and effects of an intrusion in connection to the variable state of the victim system. Similarly, in STATL [9] language, sequence of events conducted by the attacker can described to express multi-stage attack. However, these approaches need a manual description of possible attacker's behavior and if a single step is missed the whole behavior go undetected.

The third category is the pre/post conditions techniques which are based on the notion that the older alerts prepare for the later ones. These approaches require specifying the criterion used to discover the relations between alerts and the weights of such relations. Early, [10] proposed a require/ provide capabilities model using attack specification language "JIGSAW". However, the exact matching between require and provide conditions is employed causing different variation of the same behavior is not detected. [11] proposed MIRADOR correlation approach for alert clustering, merging and then correlation. Explicit correlation of events based on security experts is used to express the logical or topologic links between events. Attack is specified using five fields and based on the language of LAMBDA [8]. Partial matching techniques are adopted to build the model. In addition to explicit correlation, implicit correlation is used to overcome possibly missing events.

Authors in [12,13] proposed alert correlation model based on prerequisites and consequences of individual detected alerts. A knowledge database "Hyper-alert Type Dictionary" contains rules that describe the conditions where prior behaviors prepare for later ones. Attack strategy is represented as a Directed Attack Graph(DAG) with constraints on the attack attributes considering the temporal order of the occurring alerts. The nodes of the DAG represent attacks and the edges represent causal and temporal relations. Similarities between these strategies are measured to reduce the redundancy. A technique of hypothesizing and reasoning about missing attacks by IDS is presented to predict attribute values of such attacks. The significance of their work is the reduction of the huge number of security incidents and to report a high-level view for the administrator. However, the proposed system is useful as a forensic tool where it perform offline analysis. In addition, building the knowledge database containing rules of the applied conditions is a burdensome. However, authors have not provided a mechanism to build the Hyper Alert dictionary. Also, the generated graph is huge even with medium size datasets.

In other respect, [14,15] proposes a combination of statistical and knowledgebase correlation techniques. Three algorithms are integrated based on assumption that some attack stages have statistical and temporal relations even though direct reasoning link is not existent. Bayesian-based correlation engine is used to identify the direct relations among alerts based on prior knowledge. In contrast to previous approaches, knowledge of attack steps incorporates as a constraint to probabilistic inference to avoid the exact matching of pre and post conditions. Causal Discovery Theory-based engine is developed to discover the statistical of one-way dependence among alerts. In addition, Granger-Causality-based algorithm is used by applying statistical and temporal correlation, to identify mutual dependency. However, the problem of selection time window for temporal correlation is still an open problem. Attackers can exploit the slow-and-low attack to avoid detection. Attack prediction also relies on prior knowledge where zero-day attack is not detected.

Although the past techniques dealt with reducing the massive number of collected data by NIDS, however there are many limitations. First, the analysis of attack strategy recognition is too complex especially if the task broadens to predict the unknown steps. Knowledge-based approaches are more accurate due to rules matching mechanism which are built based on experts' knowledge, but it needs more efforts to provide precise rules. Statistical and temporal analysis techniques are unable to detect causal relations among events, but they don't require prior defined rules. Adoption of such systems in real-time is still an open question, where most proposed systems have been tested in offline fashion or in a low volume traffic environment. The huge number of detected events leads to graph explosion as in [12,13]. Moreover, missing attacks by the IDS can result in separate scenarios related to the same attack. Attackers also exploit the attack sliding window used in most approaches by performing slow-and-low attack.

In this paper we have extended our previous work in [16][17] to describe the details of the proposed model design. The underlying principle of the model based on provides/requires model is defined precisely giving some clarification examples. The discussion has been supported by the evaluation using different metrics. The rest of this paper is organized as follows: section 2 explains the concepts of the proposed model. In Section 3, we present a description of the knowledge-based modeling and its related components. Section 4 gives the experimental results and then we conclude in section 5.

II. MULTI-STAGE ATTACK RECOGNITION SYSTEM (MARS) FRAMEWORK

The MARS framework [16][17] is a logical framework supported by various components for alert correlation, aggregation, reduction and multi-stage attack recognition, as shown in Fig.1. Despite the differences between alert correlation approaches, they require some common modelling. A knowledge-base that contains attack characteristics is either abstracted or using actual attack details. Information acquisition for a knowledge base is based on the model employed (e.g. expert systems, artificial intelligence). The main drawback of the previous approaches is that they do not provide knowledge representation in a systematic way. For instance, *requires/provides* is a general alarm management model that has been used widely in the alert correlation field, but most of the proposed paradigms are based on ad hoc methods of knowledge representation. In our framework, knowledge elements are designed using a formal knowledge formalization exploiting available information provided by IDSs, vulnerability scanners and environment configurations. It also allows interactive communication between the administrator and the core system engine. Generated events reflecting the detected security situation are produced after a series of processing functions to reduce the data size. The implementation of the MARS framework will be discussed in Chapter 5. In this chapter, the underlying principles of the proposed framework are introduced.

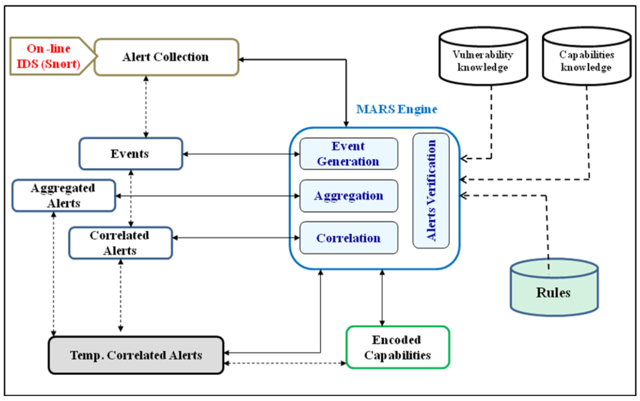

Fig. 1. Multi-stage Attack Recognition System (MARS) framework.

Fig.1 gives a graphical representation of the framework components that implemented in MARS system. The first task is performed on all received alerts from the IDS sensor e.g. Snort. *Alert Collection* contains normalized alerts presented in a standardized format that are understood by all correlation components. Also, a pre-processing function is carried out to normalize all required alert attributes such as time stamp, source, and destination addresses. The final results of this process are stored in *Alert Collection* which represents the main data input for the MARS engine. MARS engine consists of four components: 1) *Alert Verification* 2) *Correlation* 3) *Aggregation* and 4) *Event generation.* The task of *Alert Verification* component is to take a single alert and determine the success of the attack that corresponds to this alert. Failed attack should be assigned as a low level of importance. However, these failed attacks are not ignored and saved in the database which can be used as evidence to support other correlation instances. The *Aggregation* component is responsible for combining a series of alerts that refer to attacks related to the same activity. IDS sensor produces number of alerts corresponding to the same attack which are conducted at the same time. Similar alerts are aggregated and a representative alert is assigned based on a temporal relationship. These aggregated alerts are saved in the aggregation collection and are used to generate multi-stage attack events. The main task of the *Correlation* component is identifying the logical connection between received alerts based on the used correlation algorithm. If any link between two alerts is recognized, they are correlated and stored in a temporary collection and then transferred to the correlation collection after performing the aggregation process. The task of the *Event Generation* component is identifying and constructing multi-stage attack patterns which are composed of a sequence of individual alerts. A new event is generated if at least two alerts are correlated and then the generated events are stored in the *Events* collection.

Two knowledge bases are used by MARS engine to support the correlation process: 1) Capabilities Knowledge base and 2) Vulnerabilities knowledge base. The capabilities database contains modelled attacks and the relationships between different attacks based on pre and post conditions of each modelled attack. Snort signatures are used in the current implementation and this can be extended to include attack definitions from other sources. Vulnerabilities database contains network and host configuration of the protected system in addition to the detected vulnerability information by the available scanner.

The initial task executed by the MARS engine is obtaining alerts from the alert collection and then creating encoded capabilities corresponding to each alert. Alerts attributes and the information supplied by the used capabilities knowledge base are used to build the encoded capabilities collection. Thus, the encoded data is utilized to produce the initial correlation information and then it is stored in the *Temporary Correlated Alerts* collection. This collection contains atomic logical connections between alerts which are consequently aggregated to obtain the aggregated collection. The generated events (Multi-stage attack instances) are constructed based on the aggregated alerts in order to minimize the resulting graph.

III. ALERT CORRELATION ALGORITH

The principle objective of the proposed framework is to identify the causal relationships between a series of attacker actions that are temporally ordered. The concept of alert correlation should not be confused with alert aggregation or alert fusion, as the latter group alerts based on clustering regardless of their temporal relations in some approaches. Alert correlation is the process of identifying a sequence of distinguished alerts that fall in the same generalized attack pattern. Fig. 2 shows the relationship between alert correlation and alert aggregation. Correlation functions are performed across the x-axis and aggregation functions along the y-axis. In this regard, we do not need to define explicitly the attack scenario, and instead the logical rules are generated using the pre- and post-conditions of each activity. Attributes provided by elementary alerts are used to define instances of alerts. Instances of system conditions are instantiated with time constraints, and correlation rules are created.

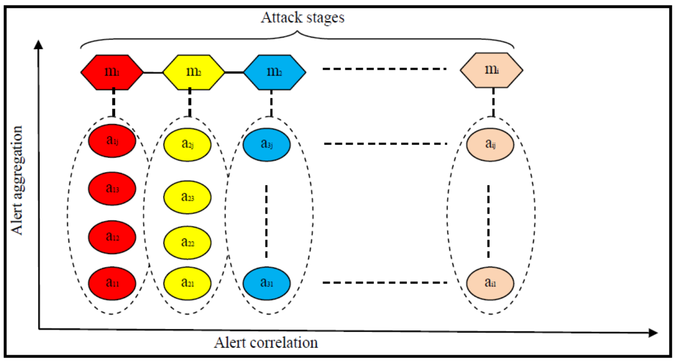

Fig. 2. Relationships between alert correlation and aggregation.

Definition 1 Given a pair of attack instances $a : a_1, a_2$ ordered temporally in the following time slots respectively:

a_1: t_{s1} and t_{e1}

a_2: t_{s2} and t_{e2}

where t_s is the start time, and t_e is the end time.

a_1 is correlated with for a_2 if:

1- There exists at least one common capability C in $R(a_2)$, and $P(a_1)$.
2- Satisfaction of $V(a_2)$ constraints.
3- $P(a_1).t_{e1} \leq R(a_2).t_{s2}$

The proposed correlation approach consists of a series of complementary phases discussed in the following sections. A complete description of the related algorithms is given to show the system's functioning.

A. *Initialization of instances of pre- and post-conditions*

The objective of this procedure is to create instances of pre- and post-conditions for each alert received. Encoded conditions are in the form of corresponding capabilities based on the arguments obtained from the in-memory knowledge dictionary. Pre-condition details of previous processed alerts are deleted because they are no longer used. In other words, the remaining possible causal links of any alert are ignored as the time constraints are not satisfied.

Consider alert a_1 detected between the times t_1 and t_2, and another alert a_2 observed between t_3 and t_4, where $t_1 \leq t_2 \leq t_3 \leq t_4$. Even though a_2 has some post-conditions that match a_1 pre-conditions, they will not be correlated as a_1 is detected before a_2.

A matching between the signatures IDs in the knowledge library and those of the sequence of the received raw alert is performed. Therefore, lists of pre- and post-condition identifiers are obtained. The argument of each condition is identified and the encoded capabilities information is stored in corresponding collections in the database.

B. *Knowledge initialization*

A complete knowledge is initialized in memory when the MARS server starts. The total memory space of a knowledge base of 15,000 signatures does not exceed a few kilobytes. The initialization process incorporates parsing of the knowledge text file (instead of a text file, an XML representation can be used for faster processing). A dictionary data structure is created to store knowledge details.

C. *Correlation algorithm*

The encoded capabilities stored in a collection of pre- and post-conditions are used to create the initial correlation graph, called a *temporary correlated collection*. In this collection, all correlated elementary alerts are stored for further processing, reflecting atomic correlations. The size of the information in temporary collections may be huge, and hence graph reduction and alert aggregation functionality are performed to obtain the final graph. The correlation process is based on the satisfaction of:

- Causal relationship based on pre- and post-conditions of each detected alert.
- Temporal and spatial constrains such as IP address, port and detected time.
- Service configuration and vulnerability details.

Each correlated alert must belong to what we have called in this research *generated events*. Complete details of events are stored in a separate collection designated *InfallEventsC*. Initially, an in-memory hash table called a *correlated map* is created, and then the details are transferred to a *temporary correlated collection*. The detected event takes the earliest start time and the latest end time among the start and end times of all corresponding alerts. An event is detected if at least two correlated alerts are detected. However, every new event is evaluated if it can be combined with other detected events on the basis of common characteristics. If there is a casual link between previous aggregated alerts and one of the detected alerts associated with the new event, the two events can be combined. In case of a connection between two events, the original event will become a master event and the new one will be considered a slave event during the process until they become a single accumulated event. The resulting event title is a concatenation of the intrusion category names of each group of events, as shown in Fig. 3, where Attack A, B, and C are general descriptions of the attack.

Once all received alerts are processed and each alert is assigned to a specific intrusion event, the original alert collection is updated in order to perform alert aggregation.

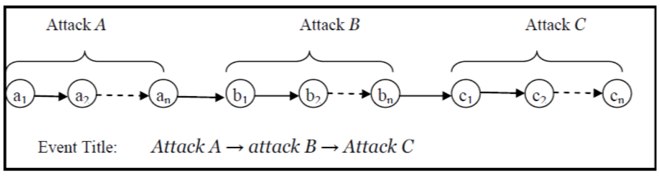

Fig. 3. Construction of an event title.

IV. Alert aggregation

A common problem among alert correlation systems is the huge amount of atomic alerts generated by an IDS and possibly by several IDSs. An IDS may trigger a large quantity of the same alerts at close time intervals that are related to the same security violation. Alert aggregation is proposed to remove duplicated alerts, e.g. the same alerts corresponding to the same signature description or attack class. A pre-defined window is used to determine whether two alerts are close enough to be aggregated into a single alert. In addition, our aggregation approach is based on graph reduction techniques that remove duplication in vertex set and migrating connecting edges to the nominated node. The resulting graph will only contain alerts that are in fact representing different security events.

Definition 2: Given a cyclic directed graph $G(V,E)$ where V is the vertex set and E is the edges set, the in-degree of a vertex is the number of edges entering it. A vertex with zero in-degree values indicates a vertex with no edges entering it (e.g. root nodes).

In the attack graph, the node's in-degree is the number of how many times the alert appears in *caused* alert group. The aggregation algorithm begins with defining the in-degree value of each node which is not aggregated in the graph. A list of zero in-degree values are identified to represent the first layer of the graph nodes; in other words, the alerts that are not caused by others. The zero in-degree list will contain groups of similar alerts occur at different times. Each group is treated as follows:

1) Nominate a master alert, which is the first alert in the temporally sorted list.

2) The aggregation process for the other alerts in the same group is based on:

- Similarity of signature IDs; this can be generalized to consider attack classes for a coarse granularity.
- Equality of source and destination IP addresses of the parties involved.
- The time difference between the detection of the two alerts does not exceed a defined value, e.g. 1 second.

3) If the above conditions are satisfied, the processed alert is added to the aggregated alerts corresponding to its master alert.

4) Change all the relationships between the aggregated alert and other alerts in the whole graph by replacing it with its master alert. Hence, the master alert will represent the aggregated alerts without losing the causal connections in the primary correlated collection.

5) Since all aggregated alerts are represented by a single alert, the corresponding time should cover the actual detection time for any further correlation. Thus, the start time of the master alert is the earliest time among the aggregated start times, and the end time is the latest one.

6) Remove the aggregated alert from the graph; however, the original information is not ignored as the graph can be disaggregated when required. Each master alert has its own counter of related aggregated alerts and graph layer.

After aggregating each group, the first graph layer, zero in-degree of all aggregated groups, is decremented by 1 to obtain the next layer. This is an opposite method to creating zero in-degree values. The second level will also have zero in-degree nodes and the same procedure is executed in an iterated fashion until all the graph nodes are treated.

V. Graph reduction

In order to reduce the complexity of the resulting graph, data redundancy should be eliminated. The graph consists of nodes representing aggregated alerts and edges representing the casual relationships. The number of nodes is not affected while the number of edges is minimised without affecting reachability. Hence, the target is to find a minimal DAG with the least number of arcs and which is equivalent to the original DAG. Consider the case shown in Fig. 4, with four alerts: *a*, *b*, *c*, and *d*. If Alert *a* is causing Alert *b* and *b* is causing *c*, there is no need for the transitive edge between *a* and *c*, and similarly the edges between *a-d* and *b-d*. The removal of the transitive optional edges does not have any effect on connectivity between the original nodes.

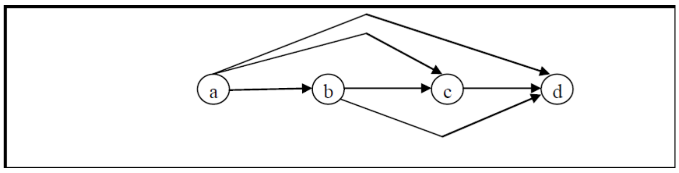

Fig. 4. Transitive edges in graph.

This is based on the assumption that the relationships between nodes can propagate and the removed edges are considered optional.

Definition 3: given a *DAG* $G=(V,E)$, $V=X$ is the vertex set, $E=R$ is the set of arcs of the graph, let $n=\#V$, $V=\{1,\ldots,n\}$, the reduced graph $G'(V,E')$ is a *DAG* with the following properties:

1) The vertex set ($\#V$) of $G(V,E)$ is equal to the vertex set ($\#V$) of $G'(V,E')$.

2) The directed paths between the vertex in $G(V,E)$ and $G'(V,E')$ are similar.

3) $G'(V,E')$ has the smallest number of edges $E'=R'$ between vertex sets without affecting the connectivity, $R'<=R$.

Two algorithms have been developed: online graph reduction for edge deletion on the left side of the graph, and offline graph reduction for edge deletion on the right side of the graph. The online algorithm removes the transitive edges at the real-time when every node joins the graph. This procedure is performed at the first stage of correlation and before alert aggregation in order to minimise the system's processing time. The offline algorithm results in a further graph reduction if any redundant connection exists after the graph is built, starting from the leaf nodes to the root nodes.

To clarify the idea, consider the alerts correlated by the system in the initial stage shown in Fig.5. There are five nodes and eight edges connecting these nodes to represent the causal relationship. In Fig. 5 (a), the number of the representing nodes n is half the number connecting arcs $\#V$. The edges $1\rightarrow5$ and $2\rightarrow5$ can be deleted because they are redundant and the description of the intrusion sequence will not be affected.

In the proposed reduction algorithm, each node has two lists of children and parents, and the aim is to remove the duplicates in these.

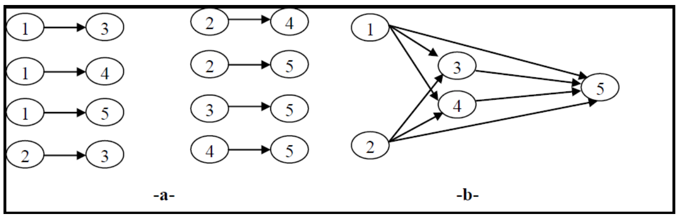

Fig. 5. Example of graph reduction.

VI. Evaluation

DARPA 2000 datasets, including LLDDOS 1.0 and LLDDOS 2.0 [18], are often used to evaluate IDSs and alert correlation systems. They consist of two multi-stage attack scenarios to launch Distributed Denial of Service attacks (DDoS). The evaluation goal is to test the effectiveness of our approach to recognize attack scenarios, to correctly correlate the alerts, and to minimize the alerts volume. We have used these datasets for their available ground truth to assess our correlation approach and to compare our results with those of other researchers. These datasets do not contain the actual alerts from the IDS sensors, and hence we have generated them using a Snort sensor. The detected events evolve over time instead of by batch analysis.

Our proposed system has achieved high levels of accuracy among the datasets in LLDDOS1.0, and acceptable levels in LLDDOS2.0 as shown in Table. 1. The only low accuracy rate recorded is from the analysis of the DMZ2.0 dataset, and of which we are aware because the actual attack was performed inside the network. In addition, the volume of alert information has been significantly reduced, achieving more than a 90% reduction rate in most test cases.

VII. Conclusion

We have proposed a correlation framework to achieve high quality multistage attack recognition and to provide the security operator with a global view of the security perspective. The pre- and post- condition approach is used to identify the logical relations among low level alerts. The alerts are aggregated, verified using vulnerability modelling, and correlated to construct multi-stage attacks. The results show that our approach can effectively detect multi-stage attacks. The resulting attack graph is reduced due to implementation of the graph reduction algorithms.

The proposed approach can build an overall view of the system's security status even with incomplete alert information. The outcome of the proposed framework is the minimisation of the effects of missing audit data, the reduction of the large volume of redundant alert which are mostly false positives, and the extraction of an attack behaviour summary in the form of a multi-stage attack scenario

TABLE I. System Evaluation

		LLDOS1.0		LLDOS2.0	
		DMZ	Inside	DMZ	Inside
# elementary alerts		3684	720	1214	199
# related alerts		1262	369	12	25
Correlation rate	# relevant correlations	1849	2915	61	91
	# detected correlations	1788	2959	69	96
	Recall rate (%)	88.4%	93.7%	60.9%	73.6%
	Precision rate (%)	91.5%	92.3%	68.9%	82.7%
Correlations with aggregation		177	156	22	65
# detected events		25	17	3	6
# aggregated alerts		135	114	17	37
Reduction rate		96.3%	84.2%	98.6%	81.4%

References

[1] "Basic Analysis and Security Engine"; http://base.secureideas.net/

[2] A. Valdes and K. Skinner. Probabilistic alert correlation. Lecture Notes in Computer Science, 2212:54-68, 2001

[3] K. Julisch. Clustering intrusion detection alarms to support root cause analysis. ACM Trans. Inf. Syst.Secur., 6(4):443-471, 2003.

[4] Zhi-tang Li, Jie Lei, Li Wang, Dong Li, "A Data Mining Approach to Generating Network Attack Graph for Intrusion Prediction," Fuzzy Systems and Knowledge Discovery, Fourth International Conference on, vol. 4, pp. 307-311, Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007) Vol.4, 2007.

[5] Jie Ma, Zhi-tang Li, Wei-ming Li, "Real-Time Alert Stream Clustering and Correlation for Discovering Attack Strategies," Fuzzy Systems and Knowledge Discovery, Fourth International Conference on, vol. 4, pp. 379-384, 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, 2008.

[6] Li, Z., A. Zhang, et al. Real-Time Correlation of Network Security Alerts. Proceedings of the IEEE International Conference on e-Business Engineering, IEEE Computer Society, 2007

[7] R. Agrawal and R. Srikant: Mining sequential patterns. In: Research Report RJ 9910, IBM Almaden Research Center, San Jose, California, October 1994.

[8] F. Cuppens and R. Ortalo. Lambda: A language to model a database for detection of attacks. In RAID '00: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection, pages197-216, London, UK, 2000. Springer-Verlag.

[9] S. Eckmann, G. Vigna, and R. Kemmerer. Statl: An attack language for state-based intrusion detection, 2002.

[10] S. J. Templeton and K. Levitt. A requires/provides model for computer attacks. In NSPW '00: Proceedings of the 2000 workshop on New security paradigms, pages 31-38, New York, NY, USA, 2000. ACM Press.

[11] F. Cuppens. Managing alerts in a multi-intrusion detection environment. In 17th Annual Computer Security Applications Conference New-Orleans, New-Orleans, USA, December 2001.

[12] Peng Ning, Yun Cui, Douglas Reeves, and Dingbang Xu, "Tools and Techniques for Analyzing Intrusion Alerts," in *ACM Transactions on nformation and System Security*, Vol. 7, No. 2, pages 273--318, May 2004.

[13] Peng Ning, Yun Cui, Douglas S. Reeves, "Constructing Attack Scenarios through Correlation of Intrusion Alerts," in Proceedings of the 9th ACM Conference on Computer & Communications Security, pages 245--254, Washington D.C., November 2002.

[14] X. Qin. A Probabilistic-Based Framework for INFOSEC Alert Correlation. PhD thesis, Georgia Institute of Technology, 2005.

[15] X. Qin and W. Lee. Attack plan recognition and prediction using causal networks. In ACSAC '04: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), pages 370-379, Washington, DC, USA, 2004. IEEE Computer Society.

[16] Faeiz Alserhani and Monis Akhlaq et al, "MARS: Multi Stage Attack Recognition System, In Proc. of the International Conf. on Advanced Information Networking and Applications (AINA), Perth, 2010, pp. 753-759.

[17] Faeiz Alserhnai and Monis Akhlaq et al, Event-based Correlation Systems To Detect SQLI Activities, In Proc. Of the International Conference on Advanced Information Networking and Applications (AINA), Bioplois, Singapore, 2011.

[18] "MIT Lincoln Laboratory "; http://www.ll.mit.edu/.