Military Technical College
Kobry El-Kobbah,
Cairo, Egypt

7th International Conference on Electrical Engineering

ICEENG 2010

Towards Implementing Agent Based Correlation Model For Real-Time Intrusion Detection Alerts

By

Ismail Abdel Ghafar, Ayman E. Taha* *Ayman M. Bahaa Eldin , Hani M. K. Mahdi* **

Abstract:

Alert correlation is a promising technique in intrusion detection. It analyzes the alerts from one or more intrusion detection system and provides a compact summarized report and high-level view of attempted intrusions which highly improves security effectiveness. Correlation component is a procedure which aggregates alerts according to certain criteria. The aggregated alerts could have common features or represent steps of pre-defined scenario attacks. Correlation approaches composed of a single component or a comprehensive set of components. The effectiveness of a component depends heavily on the nature of the real alerts or the dataset analyzed. The order of correlation components affects the correlation process performance. Moreover not all components should be used for different dataset. This paper presents implementation of an Agent Based Correlation Model for real-time intrusion detection alerts. Learning agent learns the nature of alerts within a network then guides the whole correlation process and components in such a suitable way of which components could be used and in which order. The model improves the performance of correlation process by selecting the proper components to be used. The simulation results showed that ABCM model assures minimum alerts to be processed on each component depending on the dataset and minimum time for correlation process.

Keywords:

Alert Correlation; Intrusion Detection; Learning Agent; Agent-Based Systems

---

* *Egyptian Armed Forces*

** *Computer and Systems Engineering Department, College of Engineering, Ain Shams University, Abasia, Cairo, Egypt.*

1. Introduction:

Intrusion detection is an essential technique which provides an extra layer of defense when security mechanisms (authentication, authorization, and auditing) fail. Intrusion Detection System (IDS) can detect either outside intrusions or monitors unauthorized activities inside the network. However IDS have some limitations which affect its performance. First, IDS is prone to producing a large number of alerts, which is difficult for experts to analyze and discover causal relationships in alert streams. Second, false positives and false negative of IDS are inevitable. Third, IDS can only detect single attack but not multi-step attacks, to detect which network security experts need to analyze manually. Finally, it is hard to deploy IDS in large scale network.

To tackle this issue, researchers and vendors have proposed alert correlation, an analysis process that aggregates and correlates the alerts. We can strikingly refine information quality of the alerts by this technique. Alert correlation gives network security administrator compact reports which provide a high-level view of intrusions and has drastically reduced the security experts' task. Analyzing the alert correlation, the security expert can evaluate overall network security incident and take counter measures instantly.

2. Alert Correlation Process:

There are three famous techniques [1, 2] for alert correlating which are Similarity-based, Pre-defined attack scenarios and Pre-requisites and consequences of individual attack. There are two architectures for alert correlation system: centralized architecture [3] and distributed architecture [4, 5]. The key process unit of centralized architecture is Central IDS Correlation Node, which directly processes alerts from multiple IDS sensors. The correlation algorithm of this architecture is simple and can correlate overall alerts quickly. Distributed architecture composed of a set of correlation nodes and categorized as complete distributed architecture or hierarchical distributed architecture.

Many tools and techniques have been implemented for alert correlation [6-8]. This paper will focus on an alternative approach model for real-time alert correlation [9, 10] which has been produced as integrated solution. It consists of a set of correlation components which cover different correlation techniques. As shown in Figure 1, the alert correlation module is composed of a set of procedures which can be arranged in different ways. The module input alerts in the Intrusion Detection Message Exchange Format (IDMEF) [11]. Some procedures process data of an alert and the others implement correlation methods by combining alerts using individual filters.

Six main components have been implemented depending on five types of filters: Fusion, One2One, Network-Host, One2Many, and Many2One. The correlation components which effectively reduce alerts are: Alert Fusion (AF) which combines duplicate alerts that represent the independent detection of the same attack by

different IDS. Alert Verification (AV) which takes a single alert and determines the success of the attack corresponding to that alert. Thread Reconstruction (TR) which combines a series of alerts that refer to attacks launched by a single attacker against a single target. Attack Session Reconstruction (ASR) which associates network-based alerts with host-based alerts that are related to the same attack, the goal of ASR component is to link network-based alerts to related host-based alerts.

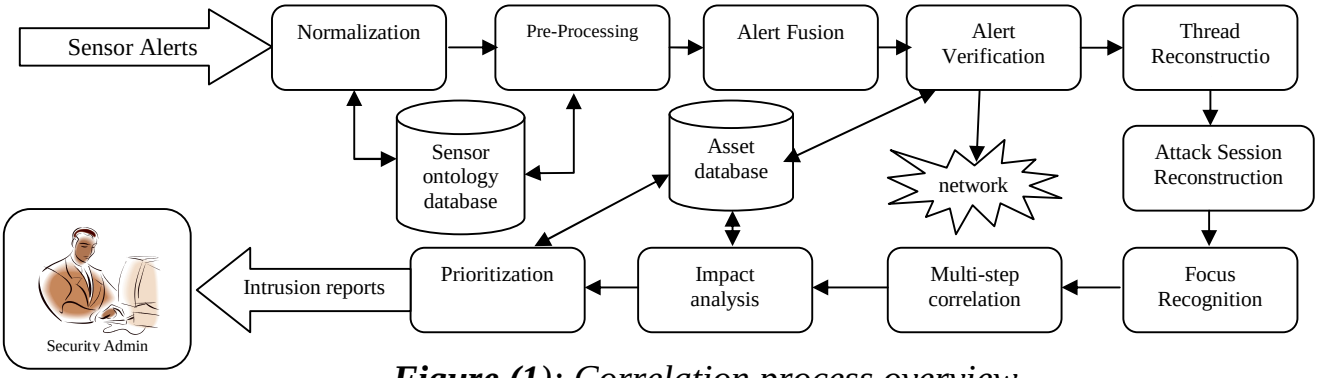

***Figure (1)**: Correlation process overview*

Focus Recognition (FR) which identifies the hosts that could be the source or the target of a substantial number of attacks. More specifically, this component aggregates the alerts associated with a single host attacking multiple victims (called a one2many scenario), and a single victim that is targeted by multiple attackers (called a many2one scenario). Multi-Step Attack (MSA) which identifies common attack patterns such as recon-breakin-escalate or island-hopping attacks {attacker breaks into a host and uses it as a launch for more attacks}. The victim in one alert becomes the attacker in the following one

There are more additional two components: impact analysis, and prioritization, that depend on the nature and the policy of the protected network. However, both of them are not evaluated in this approach. The study of the model (Table 1) shows that normalization and pre-processing are necessary for alert attributes setting. TR and FR, that have the highest Reduction Rate (RR) percentage {RR is the rate of reduction of input alerts to produce output alerts}, are considered the most effective components used for all datasets. Both AF and MSA have lower RR values, yet they are still used for the most of datasets. Each of AV and ASR does not have any effect except on one dataset only. It is concluded that the affected correlation components are six (AF, AV, TR, ASR, FR, MSA), but not all of such components are used for all different dataset (The average is 3.7 component).

The sequence order of 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. Table 1 shows analysis result of the effectiveness of each component on the different analyzed datasets. The last row shows the total count of effective components whose reduction rate is more than zero value. Such count differs according to the dataset, and varies from minimum two components in the case of " Rome AFRL" dataset to five in the case of" Treasure hunt". The RR for each component varies from 0 to 99.91 % depending on the component algorithm and selected dataset. Different reduction rates of each

component simply affect the following component input of alert stream, i.e. the arrangement of components is a primary concern for each dataset to obtain faster correlation process.

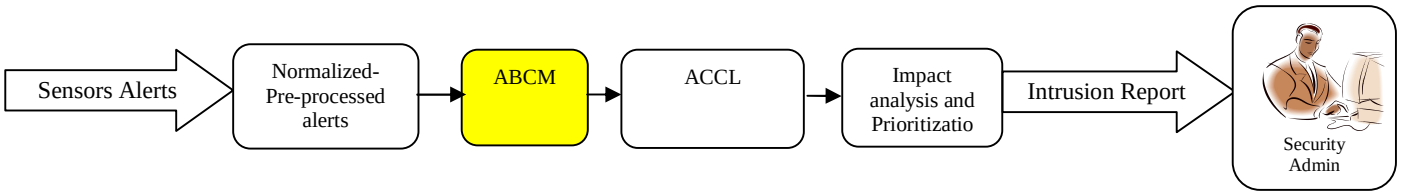

Figure (2): *ABCM correlation process*

Table (1): *Reduction Rate matrix for components/datasets*

	MIT/LL1999	MIT/LL2000	CTV	Defcon 9	Rome AFRL	Honeypot	Treasure Hunt	Average
AF	6.38	0.01	0.04	28.43	0	0	0.09	4.99
AV	0	0	0	0	0	97.	0	13.9
TR	77.1	6.61	31.5	60.25	69.8	71.	99.9	59.5
ASR	0	0	0	0	0	0	2.27	0.32
FR	10.9	49.6	89.9	88.65	70.8	2.2	50.6	51.8
MSA	0	0.16	0.63	1.24	0	1.0	2.2	0.7
Count	3	4	4	4	2	4	5	3.7

3. AGENT BASED CORRELATION MODEL:

Figure 2 shows the proposed model which presents an Agent Based Correlation Model (ABCM) for real-time Intrusion Detection Alerts. In this model Smart Learning Agent (SLA) learns the nature and characteristics of normalized alerts produced by different IDSs within a network, and then it selects the suitable correlation components that can be used and their proper order.

The model provides minimum correlation time for all datasets, whatever their nature. ABCM consists of two phases, learning phase and correlation phase. The input of ABCM is normalized and pre processed alerts while the output goes to a set of selected correlation components called Active Correlation Components List (ACCL). The selection of added components in ACCL depends on agent learning, the output alerts correlated by ACCL is directed to the last two components of correlation process. This model is based on the real-time correlation model [9, 10]. However, instead of using sequence of all correlation components, it uses a set of specific effective correlation components depending on agent learning.

3.1. Learning Phase

During the learning phase, SLA learns the output of each component and the dataset nature. Based on this learning beside a set of rules and knowledge base as well, SLA can determine the active correlation components and their proper order. Each correlation component has specific criteria to aggregate and correlate alerts. The knowledge base for learning is formed by the criteria for each component in addition to the RR obtained by each component.

Algorithm 3.1 Learning Phase

Inputs: (IS) normalized and preprocessed stream of alerts, IP: number of input alerts, learning parameter (Alerts number N)

Output: (ACCL), set of active correlation components (RRc > 0)

Initialization: Empty ACCL (Ser, CC, RR) ACCL (0, ,0), k=6 (maximum number of correlation components), m=0

For alerts in N

While k > 1 do

// For each component do

OSc ⟵ CORRc(IS);
OPc ⟵ no of alerts in OSc
RRc ⟵ (1-OPc/IP)*100
if RRc > 0 then
begin
ACCl(Ser) ⟵ ser+1;
ACCL(RR) ⟵ RRc;
ACCL(CC) ⟵ CC;
end
Else
// For each component with RRc=0
begin
Disable component;
m ⟵ m+1;
end
end if
/ / end if RR>0
k ⟵ k-1;
loop
end while
sort ACCL(RR, descending);
end for
return ACCL;

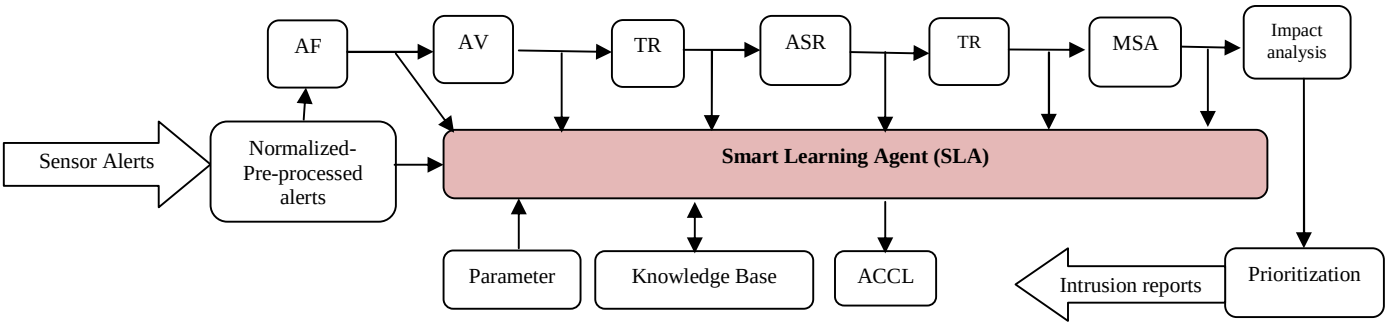

Figure (3): *Learning Phase*

Figure 3 shows that SLA learning depends on initial inputs. These inputs are: the learning parameter which could be a period of time (t) or specific number of alerts (N); the normalized pre-processed alerts; and a pre generated knowledge base. The learning phase starts through the execution of initial correlation process. The initial components order could be as described in model [9, 10], or it could be random. The implemented model support parallel learning for all correlation components. Each component aggregates and merges its input alerts according to component algorithm and criteria. Alerts attributes (source, attack type, destination) are used as the basis of merging alerts. RR can be calculated through comparing output alerts with input alerts. Depending on the value of RR for each component, SLA builds ACCL which contains the components with RR higher than zero value. Algorithm 3.1 describes the learning phase process; SLA builds ACCL in descending order of the components reduction rate. The normalized preprocessed alerts go through their basic correlation path. By the end of the correlation process of each component, SLA reads RR of each component. If the RR is higher than zero, the component data will be added to the ACCL which includes serial, component name, and RR value.

By the end of correlation of the last component, ACCL will contain a specific set of components with different RR values. The agent sorts these components in descending order of their RR. Learning phase should be enough for studying the nature of alerts in the network. Such phase continues depending on the learning parameter (t or N) and/or assuring no changes of the alerts nature. SLA could be a part of correlation process by eliminating some alerts depending on the network nature (alerts against windows server while maintaining UNIX server).

3.2. Correlation Phase

By the end of learning phase, ACCL will contain only effective correlation components in descending order of their RR. In the correlation phase, the flow of normalized alerts stream will be controlled by the agent. Alerts are directed to the first component in ACCL which has higher RR during the learning phase. The output of the first component will be the input of the second one which has second higher RR, and so on till they reach the last component in ACCL.

Figure 4 describes correlation phase of the normalized preprocessed alerts. Alerts are directed through SLA to one path of many alternative paths. These alternative paths represent different suggested ACCLs which have been implemented previously during the learning phase.

For example, the analysis of RomeAFRL dataset correlation shows that ACCL has only FR and TR (Highlighted Boxes in figure 4) with RR values (FR=70.87% and TR=69.82%), While other four components AF, AV, ASR, and MSA have no effect on that dataset.

Algorithm 3.2 Correlation Phase
Inputs: (IS) normalized and preprocessed stream of alerts, IP: number of input alerts, ACCL (Ser, CC, RR)
Output: (OS) correlated stream of alerts,
Begin
While ACCL (ser) > 0 (is not empty) do
// loop for all components in ACCL
begin
CORRc(IS) using ACCL(CC);
OSc ⟵ CORRc(IS);
OPc ⟵ no of alerts in OSc
RRc ⟵ (1-IP/OPc)*100
ACCL(CC) ⟵ next ACCL(CC);
// next lowest RR component in ACCL
loop
// all components in ACCL have been used
end while
OS ⟵ OSc of last component in ACCL
end
return OS;

Algorithm 3.2 shows the correlation phase; ACCL and normalized alerts stream are both inputs of the algorithm, while the correlated alerts OCc are considered to be the output. The agent uses the components in ACCL to correlate the input alerts, and then it moves the pointer of ACCL to the next component, the loop continues till using all components in ACCL.

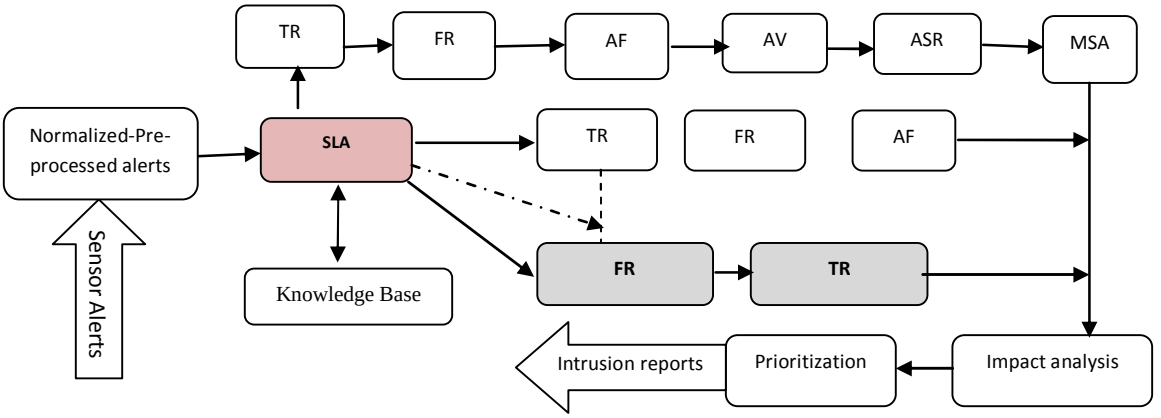

Figure (4): *Correlation Phase*

3.3. ABCM Implementation

ABCM model was analytically compared with different correlation approaches in [12]. In this paper a model has been implemented to compare between the proposed ABCM correlation model and comprehensive Approach Model (CAM) [9, 10]. Random alerts generator component has been used to generate alerts to be used for correlation either by CAM or ABCM. Figure 5 shows the main interface of the model. It contains button for alert generator module, these generated alerts could be saved and loaded again for further process.

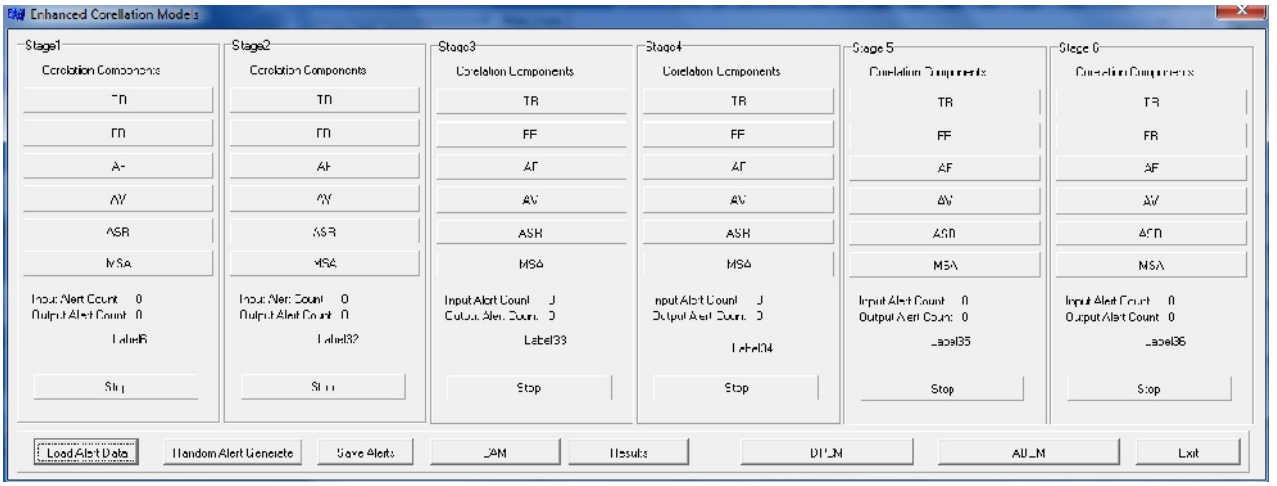

Figure (5): *Correlation Model Main interface*

The main interface contains buttons to correlate the generated alerts using either CAM or ABCM approaches and showing results for each of them.

Generation of random alerts depends on several parameters as shown in Figure 6. These parameters include windows size, start time, attack types count, victim count, and finally attacker count. These parameters could be changed to form a specific alerts nature. By the end of entering these parameters we choose how many alerts we want to generate, the alerts counts could be from minimum 100 alerts to 100,000 alerts. After generating the alerts set we can save it using save alerts button for later use.

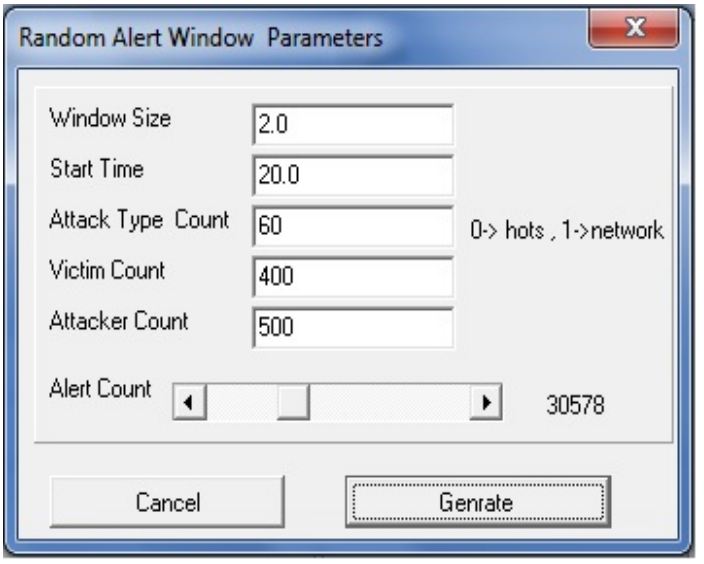

Figure (6): *Random Alert Generator Module.*

Either by using the current generated alerts or loading of previously saved alerts we can start correlation of these alerts. CAM button correlates the alerts using CAM model, it perform the whole correlation process as described in [9, 10]. By the end of the process we can show CAM results using the button results which display the CAM correlation results as shown in Figure 7. These results contains input alerts

count, output alerts count, the reduction rate, and the processing time (msec) for each component and for the entire correlation process.

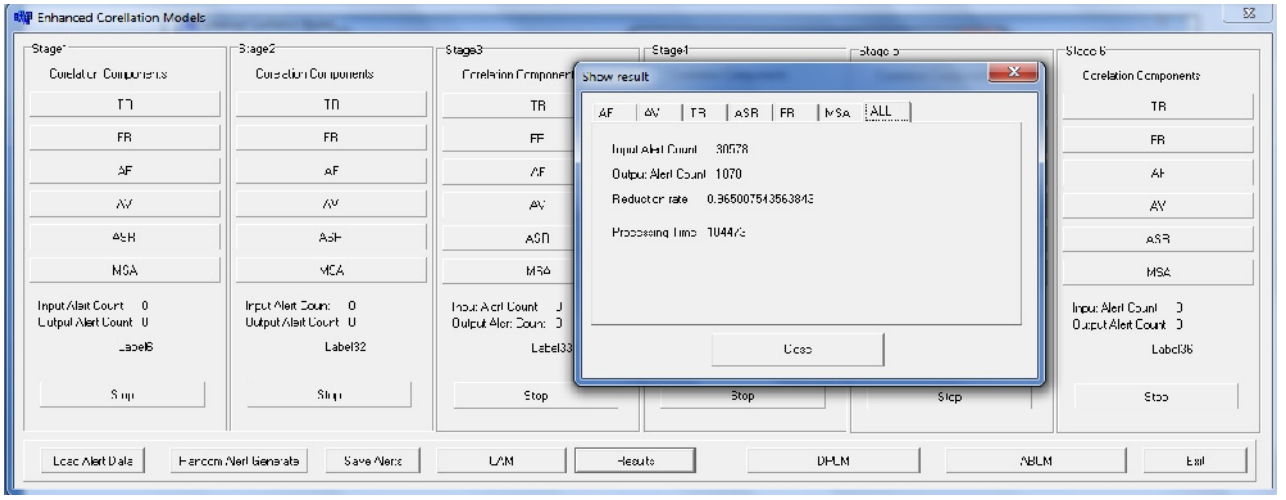

Figure (7): *CAM model correlation Results*

ABCM model correlates same set of generated alerts; it works in three processes as shown in Figure 8. First step determine the number of alerts to be learned for generating the ACCL.

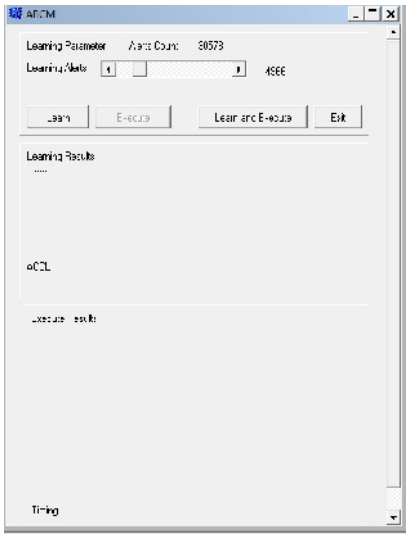

Figure (8-a): *ABCM Correlation Process*

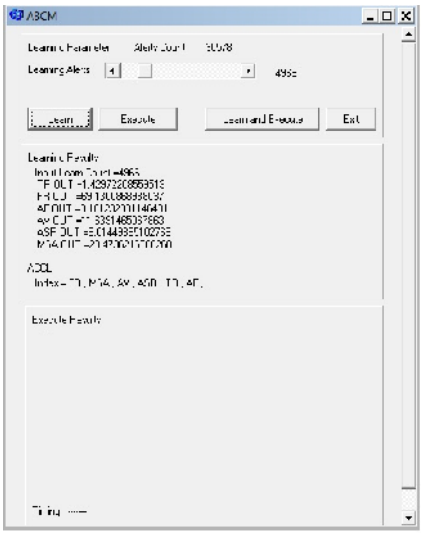

Figure (8-b): *ABCM Learning Results*

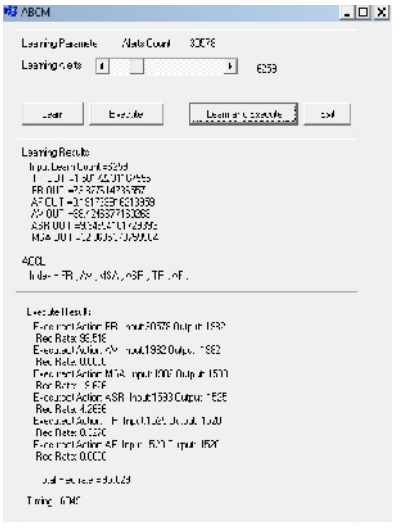

Figure (8-c): *ABCM Correlation Results*

The generated ACCL may contain all correlation components or a specific set of them depending on the learned alerts. After determining of the learned alerts count, learn button could be used to start learning process which finally produces ACCL and indicates which components used and the reduction rate obtained by each component as shown in Figure 8-b.

After generating of ACCL in learning phase, the execute button will be used to perform the correlation for all generated alerts using the sequence in ACCL which finally produce the ABCM correlation results as shown in Figure 8-c. These results include input alerts count, output alerts count, and reduction rate for each

components, then finally the total reduction rate by ABCM and the total correlation time (milliseconds).

4. Experimental Results:

Performance of each correlation approach depends on the generated alerts; results of applying both correlation approaches on the generated alerts are shown in Table 2. Table 2 shows that ABCM has the lower correlation time compared with CAM for different sets of generated alerts. ABCM correlation has very short correlation time compared with CAM , minimum ABCM correlation time is 4.07 sec compared with minimum CAM correlation time 29.89 second in case of 11953 alerts. Where maximum correlation time using ABCM is 53 second compared with 1082 second obtained by CAM in case of 100000 alerts.

Table (2): *Correlation Times of CAM and ABCM*

No Of Alerts	Correlation Time (Sec)		Time Reduction %
	CAM	ABCM	
11953	29.89	4.074	86.37
22230	57.206	14.731	74.25
30578	126.954	14.849	88.30
40000	143.677	15.304	89.35
50050	278.696	21.279	92.37
60209	389.878	23.822	93.89
70350	539.748	28.179	94.78
80528	704.469	31.021	95.60
90400	1006.737	41.926	95.84
999999	1082.055	53.579	95.05

Last column in table 2 shows the Reduction of Time (RT) percentage obtained by ABCM compared with CAM [9, 10]. ABCM time reduction rate varies from value of 74 % in case of 22230 alerts count to value of 95 % in case of alerts count more than 80000. Figure 9-a shows graph chart representation of correlation time in case of CAM and ABCM models for different alerts count which varied from 10000 alerts till 100,000 alerts. The graph shows the big difference in time and the time saved using ABCM instead of CAM approach.

Table 3 shows reduction rate obtained by CAM and ABCM for different sets of generated alerts. The results show that ABCM almost has similar accuracy of reduction rate obtained by CAM. Last column in Table 3 indicates the difference in RR for both approaches. It shows that the RR obtained by ABCM is fewer than RR obtained by CAM with value less than one in most cases, while both RR almost equal in case of correlation of very large number of alerts.

Figure 9-b shows chart representation of CAM and ABCM reduction rate values, the graph shows the RR values for both correlation approaches very similar for different sets of generated alerts.

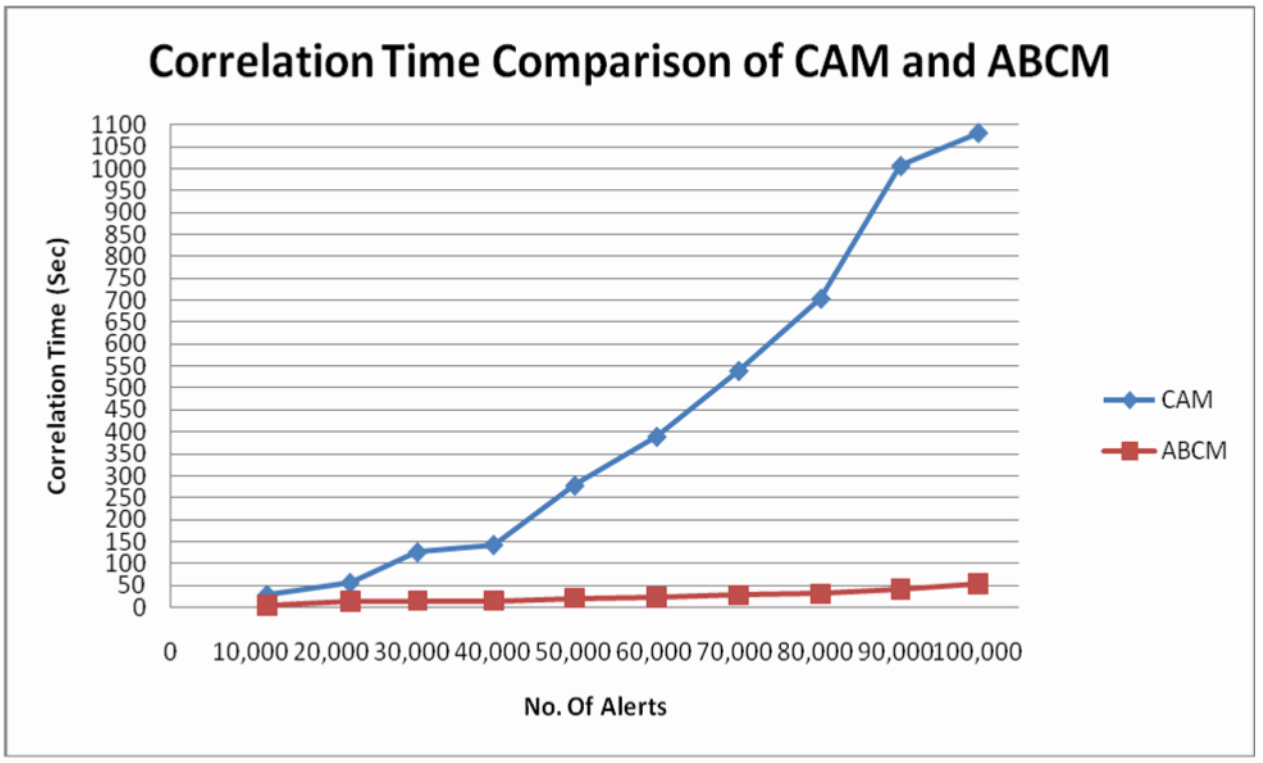

Figure (9-a): Correlation time comparison for CAM and ABCM

Table (3): *Reduction Rates of CAM and ABCM*

No Of Alerts	Reduction Rate %		Difference in RR
	CAM	ABCM	
11953	89.40	87.62	1.78
22230	94.99	94.06	0.94
30578	95.66	95.03	0.63
40000	96.22	96.04	0.18
50050	97.12	96.36	0.76
60209	98.08	96.98	1.10
70350	98.06	97.43	0.63
80528	98.30	98.08	0.23
90400	98.44	98.29	0.15
999999	98.50	98.64	-0.01

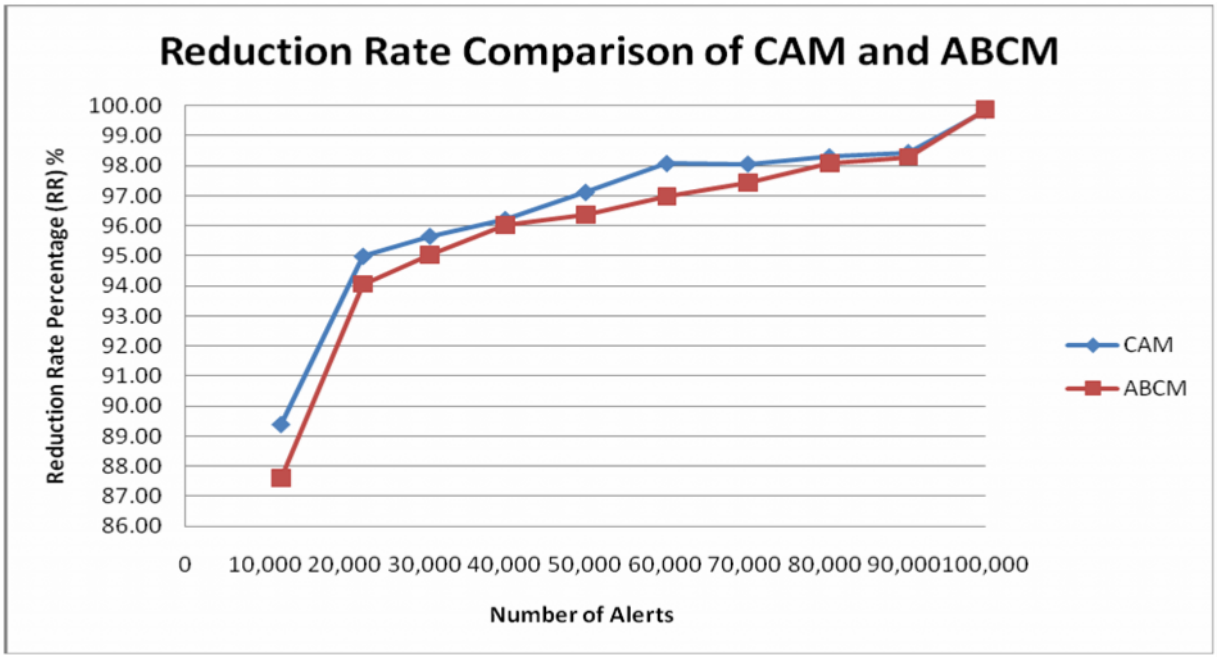

Figure (9-b): *Reduction Rate comparison of CAM and ABCM*

5. Conclusions and future work:

In this paper, an Agent Based Correlation Model (ABCM) for real time intrusion detection alerts has been implemented. The model works through learning phase and correlation phase. In the learning phase SLA learns the nature of alerts datasets and effective correlation components and their RR and builds ACCL. The ACCL contains the effective correlation components in descending order of their RR. Depending the learning phase, the agent controls the correlation process during the correlation phase using the implemented ACCL. The order of correlation starts with component with higher RR in ACCL then followed by lower RR until correlation by the last component in ACCL.

The proposed model improves the correlation process performance by decreasing the total correlation time. The results showed that ABCM has better performance compared with CAM by average percentage 90% of time reduction for all generated alerts. ABCM has almost same reduction rate compared with CAM. That means the proposed model maintains the same correlation accuracy provided by CAM in less time and less number of components.

The proposed model is scalable regarding the number of correlation components in ACCL. The hardware improvement needed for agent process in each phase is not significant and causes no problem in recent technology.

The implemented ABCM model could be used for real-time intrusion detection alerts. Future work will include investigation the optimal learning parameters and combining both phases to support continuous adaptive learning during correlation process. Also distributed correlation agents would be investigated to assure scalable alert correlation for large scale network.

References

[1] Robiah Yusof, Siti Rahayu Selamat, and Shahrin Sahib "Intrusion alert correlation technique analysis for heterogeneous log," IJCSNS International Journal of Computer Science and Network Security, vol.8, No.9, September 2008.

[2] Zhai, Y., Ning, P., & Xu, J. "Integrating IDS alert correlation and OS-level dependency tracking." North Carolina State University, North Carolina, 2005.

[3] C. Mu, H. Huang, and S. Tian, "A survey of intrusion-detection alert aggregation and correlation techniques," Journal of Computer Research and Development, vol. 43, pp. 1-8, 2006.

[4] Donghai Tian, Hu Changzhen, Yang Qi, and Wang Jianqiao "Hierarchical Distributed alert correlation model," 2009 Fifth International Conference on Information Assurance and Security, pp. 765-768, Aug, 2009.

[5] Chenfeng VincentZhou, ChristopherLeckie, Shanika Karunasekera, "Decentralized multi-dimensional alert correlation for collaborative intrusion detection," scienceDirect, Journal of Network and Computer Applications, 32 (2009) 1106–1123, Feb 2009.

[6] C. Mu, H. Huang, and S. Tian, "Adaptive alert aggregation in intrusion detection alert management & Intrusion response system," Journal of Computer Science, vol. 34, No. 12, pp. 73-77, 2007.

[7] P. Ning, Y. Cui, D. S. Reeves, and D. Xu, "Techniques and tools for analyzing intrusion alerts," ACM Transactions on Information and Systems Security, vol. 7, pp. 274-318, 2004 .

[8] H. Debar and A. Wespi, "Aggregation and correlation of intrusion detection alerts," Proc. Int'l Symp. Recent Advances in Intrusion Detection, pp. 85-103, Oct. 2001.

[9] 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, pp. 146-69, 2004.

[10] F. Valeur, "Real-time Intrusion Detection Alert Correlation," Ph.D. Thesis, University of California Santa Barbara, Santa Barbara, California, USA, 2006.

[11] D. Curry and H. Debar, "Intrusion Detection Message Exchange Format: Extensible Markup Language (XML) Document Type Definition," draft-ietf-idwg-idmef-xml-10.txt+, Jan. 2003.

[12] A. Taha, A. Bahaa, I. Abdel Ghafar, and H. K. Mahdi, "Agent Based Correlation Model For Intrusion Detection Alerts," IEEE International Conference on Intelligence and Security Informatics (ISI 2010), Vancouver, British Columbia, Canada, May 23-26, 2010.