International Journal of Computer & Information Technologies (IJOCIT)
www.ijocit.ir & www.ijocit.org
ISSN = 2345-3877

OMADM: Online Multi-step Attack Detection Method

Ali Amiri[1], Alireza Nowroozi[2]
Security Evaluation Lab for ICT Appliances, IT Security Institute, ICT Department, Malekashtar University of Technology, Tehran, Iran[1,2]
Email: ali.amiri@chmail.ir
Security Evaluation Lab for ICT Appliances, IT Security Institute, ICT Department, Malekashtar University of Technology, Tehran, Iran
Email: nowroozi@mut.ac.ir)
ali.amiri@chmail.ir[1], nowroozi@mut.ac.ir[2]

Abstract: - Network Intrusion detection systems (NIDS) have become an important and essential part of computer networks, and increase the security of them. Traditional NIDS, despite their advantages, have some disadvantages such as: producing high amounts of alerts that are low-level, mixing true alerts with false alerts, inability to find a logical connection between alerts for detecting novel and multi-step attacks, and Managing and detecting alerts in an offline mode. As a result, it is difficult for human users and intrusion response systems to understand the alerts and takes proper actions on time. A new kind of attacks that NIDS has some weaknesses for detecting them, are multi-step attacks. In this kind of attacks, the attacker runs the attack based on a pre-designed scenario and in separate steps; each of these steps has a logical connection with other steps. In this paper, we propose an online multi-step attack detection method (OMADM) based on prerequisites and consequences of the attacks. In OMADM method, the alerts are processed in an online mode, and the attack scenarios will be generated in an online mode. To evaluate and make sure the accuracy for this method and validating OMADM, we implement an online multi-step attack detection tool (OMADT), a prototype of OMADM, and evaluate OMADM with DARPA 2000 and a collected dataset that includes some attack scenarios. Each attack scenario in our dataset has different models. Our experiment demonstrates the accuracy, speed, and the high ability of this method in alert correlation and detecting online multi-step attacks and generating online attack scenarios.

Keywords: Network Intrusion detection system, multi-step attacks, attack scenarios, alert correlation

1. Introduction

In order to detect an intruder who is trying to penetrate into our network, we use NIDS. NIDS receives security alerts and analysis this alerts automatically. The result from this analysis help us detect and prevent the same attack in the future. Intrusion detection techniques can be categorized as anomaly detection and misuse detection. Anomaly detection is based on the normal behavior of a subject (e.g., a user or a system). We detect a normal behavior of a subject and model it. With this assumption that any action that significantly deviates from the normal behavior is considered intrusive, we detect the attack.

Misuse detection detects attacks based on the characteristics of known attacks or system vulnerabilities; any action that conforms to the pattern of a known attack or vulnerability is considered intrusive. The Traditional intrusion detection systems use this method for detecting intrusive activity. Traditional IDSs fail to detect new attacks because of complexity and sophistication of these attacks. This IDSs also generate a large amount of alerts that are mixed with false alerts. This situation makes it difficult for the network managers to have a clear view of the security status within the network and do some effective measures in time. Alert correlation can solve these problems. Alert correlation reduces the large amount of alerts and reduces false alerts. One of the alert correlation methods is alert correlation based on prerequisites and consequences of attacks. In this method, process of correlation tries to find the Causal relationship between alerts based on their prerequisites and consequences. The prerequisite of an attack is the necessary conditions for the attack to be successful, while the consequence of an attack is the possible outcome of the attack. The main idea for this method is: correlate alerts if the prerequisites of some later alerts are satisfied with the consequences of some earlier alerts. One problem that is not completely solved with alert correlation methods is multi-step attacks. In a multi-step attack, the intruder breaks the attack into many steps that each step is done separately and there is a logical connection between these steps.

In this article, we propose a method for solving the mentioned problems. We called our method "Online Multi-step Attack Detection Method" or OMADM. In OMADM, alerts are correlated based on their prerequisites and consequences of attacks.

The rest of this paper is organized as follows: section 2 discusses related work. Section 3 presents our method, OMADM. Section 4 presents our tool, OMADT and reports our experiment. Section 5 presents a comparison between OMADT with a tool called TIAA.

2. Related work

Intrusion detection has been studied for more than 34 years since Anderson's report. A survey of the early work on intrusion detection is given in [1] by Stefan et al. and [2] by Peyman Kabiri et al. All these IDSs are aimed at detecting low-level attacks or anomalies, and none can capture the logical steps or attack strategies behind these attacks. It is usually up to human users to discover the connections between alerts. However, in the large-scale network situations, IDSs may generate large numbers of alerts, and it seems impossible to correlate alerts by hands. A lot of methods have been proposed for alert correlation and for solving the mention problems that you can see a comprehensive survey on them in [3] by Saeed Salah et al. One of this

method is alert correlation based on prerequisites and consequences. A lot of work has been done with this method of correlation, but we will introduce some of them that are more important.

Zhaowen et al. [4] proposed RIAC, a real time alert correlation model to analyze and discover attack scenarios behind alerts. The assumption here states that the component attacks are usually not isolated, but related to different stages of the attacks, with the early ones preparing for the later ones. They introduce the notion of hyper-alerts to represent the prerequisite and the consequence of each type of alert by using logical predicates. Each hyper-alert is a tuple (fact, prerequisite, consequence), where fact is the set of alerts attribute's names, and prerequisite and consequence are two different sets, each one consisting of a logical combination of predicates expressed as mathematical conditions on the variables contained in the set fact.

Ning et al. [5] also published a similar work. They presented TIAA, a toolkit for constructing attack scenarios by using predicates as the basic constructs to represent the prerequisites and (possible) consequences of attacks. Based on the prerequisites and consequences of different types of attacks, the proposed method correlates alert by partially matching the consequences of some prior alerts with the prerequisites of some later ones.

Whereas TIAA allows partial satisfaction of prerequisites, JIGSAW [6] requires that all capabilities be satisfied. JIGSAW is a multistage correlation system. It uses capabilities and concepts to formulate the attack conditions. Capabilities are used to describe the information that the attacker must know to perform a certain attack, while concepts are used to model fragments of complex attacks.

MIRADOR was developed independently and in parallel to TIAA. MIRADOR correlation method proposed by Cuppens and Miege in [7]. The MIRADOR approach also correlates alerts using partial match of prerequisites (preconditions) and consequences (post conditions) of attacks. However, the MIRADOR approach uses a different formalism than TIAA. In particular, the MIRADOR approach treats alert aggregation as an individual stage before alert correlation, while TIAA allows alert aggregation during and after correlation.

Xiao et al. [8] proposed an alert correlation approach for alert fusion. It has two phases. First, using a fuzzy clustering algorithm, some alert subsets are created. Second, the method of correlating alerts based on prerequisites and consequences of attacks is adapted to be applied to these subsets.

Finally, Alserhani et al. [9] developed a rule based correlation language MARS, a Multi-stage Attack Recognition System. Unlike others, they add another two parameters for modeling attack consequences, i.e., vulnerability and extensional consequences. MARS is mainly based on the phenomena of ‘‘cause and effect’’. It has two main components: online and offline. The main purpose of the online component is to receive raw alerts and generates hyper-alerts. Then, multi-stage attack recognition is applied to correlate hyper-alerts based on rules provided by the offline component.

The method that we use in this article is close to the methods that have been used in TIAA, JIGSAW, and MIRADOR. A problem of these methods [10-12] is that they are offline. In addition, the solutions such as [13–15] that do operate in online mode have problems in performance and are only able to operate in real-time on datasets with a low alert-rate.

3. OMADM, correlation and multi-step attack detection system

For solving problems that we mention in the previous section, we propose an online multi-step attack detection method (OMADM) for detecting the logical connection between alerts and extracting attack scenario. For detecting the connection between alerts and extracting attack scenario OMADM uses alert correlation method based on the prerequisite and consequence. The first comprehensive architecture is presented in [16]. Fig.1 show the architecture of OMADM.

For correlation between alerts we need a knowledge base that stores the hyper-alerts that each of these hyper-alerts is equivalent to an intrusion detection system alert or an attack. A hyper-alert type T is a triple (fact, prerequisite, and consequence), where (1) fact is a set of attribute names, each with an associated domain of values such as source and destination IP addresses, (2) prerequisite is a logical combination of predicates whose free variables are all in fact, and (3) consequence is a set of predicates such that all the free variables in consequence are in fact. The knowledge base consists of three sections:

- Predicates: which specify the prerequisites, and the consequences of a given alert.
- Hyper-alert types: which specify the related predicates of a given alert such as fact, prerequisite that are needed to this hyper alert and consequence that happen in the Occurrence of this hyper-alert.
- Implications: which specify the relations between different predicates.

In addition to the knowledge base for alert correlation, we need a database. Fig.2 show the database of OMADM.

The main idea of OMADM is: each attack in addition to its own consequences may be, in consequence, of earlier attacks. It means that some attack happened before and now there is a new attack. This attack has some consequences, but this attack may be, in consequences of the earlier attack. The main idea of most of the alert correlation methods that are using pre/post condition is: correlate alerts if the prerequisites of some later alerts are satisfied with the consequences of some earlier alerts. The difference between our idea and this idea is: When an attack occurs, we create a list of all the attacks that can occur as a result of the attack base on this attack and our knowledge base. When a new attack occurs, we match the attack with the previous list of attacks. If some important variable matches, these attacks are correlated with each other. But in other methods when a new attack happens, its prerequisites are checked with the consequences of previous attacks. Our matching is based on attack not based on prerequisites and consequences of attacks. This is why our method is online. With regard to this idea, we explain other parts of OMADM architecture.

Alert receiver gets the different alert from different IDSs that are distributed over the network. In fact, all the generated alert in the network will lead to the alert receiver. Each of these alerts has a standard structure like IDMEF format and other vendor specific formats. Alert receiver gives this row alert to the OMADM database and alert normalization section.

International Journal of Computer & Information Technologies (IJOCIT)
www.ijocit.ir & www.ijocit.org
ISSN = 2345-3877

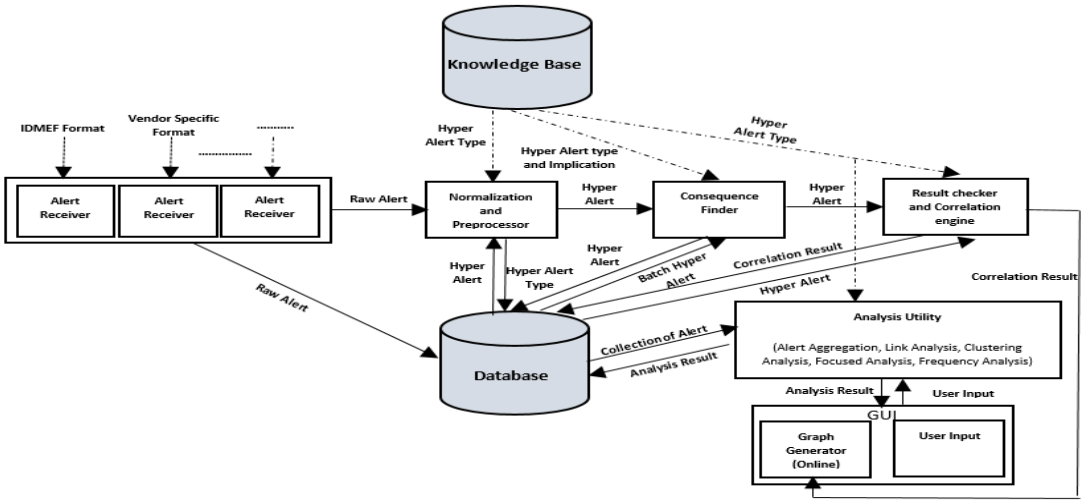


Fig1: OMADM Architecture

Fact_ID	Fact_name	Type
1	DestIPAddress	Varchar(15)
2	DestPort	int
3	SrcIpAddress	Varchar(15)
4	SrcPort	int

1) Fact

HyperAlert_ID	HyperAlertName	Protocol
20	Sadmind_Amslverify_Overflow	SADMIND
21	Sadmind_Ping	SADMIND

2) KB-HyperAlert

HyperAlert_ID	Fact_ID
20	1
20	3
21	1

3) HyperAlert-Fact

Predicate_ID	Name	Arg_ID	Arg_Pos
4	GainAccess	4	1
17	SystemAttacked	17	1
18	SystemCompromised	18	1
21	VulnerableSadmind	21	1
25	OSSolaris	25	1

4) Predicate

Predicate_ID	Implied_Predicate	Implying_Arg	Implied_Arg
4	18	4	18
18	17	18	17

5) Implication

HyperAlert_ID	Predicate_ID	Arg_ID	Fact_ID
20	21	21	1
20	25	25	1
21	25	25	1

6) Prerequisite

HyperAlert_ID	Predicate_ID	Arg_ID	Fact_ID
20	4	4	1
21	21	21	1

7) Consequence

ID	HyperAlertName	Imp_Value	Imp_Parameters	Result
1	Sadmind_Ping	0,0,172.16.115.20,0	DestIPAddress	Sadmind_Amslverify_Overflow
2	Sadmind_Amslverify_Overflow	0,0,172.16.115.20,0	DestIPAddress	FTP_Put,Mstream_Zombie,Rsh

8) BaseInfo

ID	HyperAlertName	ScrIP	ScrPort	DestIP	DestPort	Date	Previous	Is Connected
1	Sadmin_Ping	202.10.20.12	647	172.16.115.20	54791	2001-11-09 23:53:20		
1	Sadmind_Amslverify_ Overflow	202.10.20.12	647	172.16.115.20	54791	2001-11-09 23:53:47	1	1

9) OtherInfo

Fig2: OMADM Database

In the alert normalization and preprocessor, the alerts that are received from alert receivers, first go in a new and necessary template that is needed for OMADM and then they are delivered to preprocessor. OMADM needs only some parameters of received alerts that are from different sensors with different templates.

The preprocessor, generate the hyper-alert equivalent to each alert and give this hyper-alert to the next section, consequence finder. You can

see some hyper-alert in Fig.2 in KB-HyperAlert table.

With regard to the hyper-alert that is received from the previous section, consequence finder by using the knowledge base will find all the consequences that are resulting from this hyper-alert. It means that we find all the attacks that may happen because of this attack. Then, because may exist an implicit consequence for some hyper-alert, the result finder, by using the implication table, for each funded hyper-alert finds the implicit consequences. The result field in the BaseInfo table shows all the hyper-alerts that the consequence finder finds. The goal of using the OtherInfo table is that it is possible that similar hyper-alerts happen and the only difference between these hyper-alerts is the time of occurrences or not important parameters. In order not to make difficult for the searching query in the BaseInfo, we insert these similar hyper-alerts in the Otherinfo. We also insert the first alert of these similar alerts in the Otherinfo table. After doing these works we have a set of consequences for each hyper-alert. Consequence finder will pass the set of consequence that may happen because of the occurrence of this hyper-alert to the next section, result checker and correlation engine.

For now, we have a set of consequences or hyper-alerts that may happen because of this hyper-alert. However, this hyper-alert may be the consequence of another hyper-alert or alerts. In fact, it is possible that this hyper-alert has a connection with earlier hyper-alerts. In order to address this issue, we will search this hyper-alert in all the rows of the result field in the BaseInfo table, and if we find a hyper-alert or more, we will check the important parameters of each founded hyper-alert with the original hyper-alert. If these parameters are the same, we correlate these two hyper-alerts. We extract the important parameter from the knowledge base. In order to correlate two hyper-alerts the date and time of them must be checked. Time and date of second hyper-alert must be greater than the first hyper-alert. Correlating hyper-alerts that are related to an attack that the second step of this attack happens before the first step is a mistake. The previous field in the OtherInfo table shows the ID of the previous hyper-alert that this hyper-alert correlated with and if the isconnected field in the OtherInfo is set to 1 it means that this hyper-alert is correlated with another hyper-alert. As soon as two hyper-alerts are correlated, the graph generator produces the equivalent of them and shows it in the output. The resulted graph is dynamic and will be updated with the new step of the attack. The hyper-alerts are the node of this graph, and the edge of this graph shows the correlation relation.

The goal of alert aggregation is to reduce the complexity of hyper-alert correlation graphs without sacrificing the structures of the attack scenarios; it allows analysts to get concise views of correlated alerts.

The difficulty of understanding a large hyper-alert correlation graph is mainly due to the large number of nodes and edges in the graph. Thus, a natural way to reduce the complexity of a large hyper-alert correlation graph is to reduce the number of nodes and edges. However, to make the reduced graph useful, any reasonable reduction should maintain the structure of the corresponding attacks. We propose to aggregate hyper-alerts of the same type to reduce the number of nodes in a hyper-alert correlation graph.

Link analysis is intended to analyze the connection between entities represented by categorical attribute values. Examples include

how two IP addresses are related to each other in a collection of alerts, and how IP addresses are connected to the alert types. Though link analysis takes a collection of hyper-alerts as input, it indeed analyzes the raw intrusion alerts corresponding to these hyper-alerts.

Clustering analysis is to partition a set of hyper-alerts into different groups so that the hyper-alerts in each group share certain common features.

Focused analysis is to help an analyst focus on the hyper-alerts in which he/she is interested. In particular, this may generate hyper-alert correlation graphs much smaller and more comprehensible than the original ones.

Frequency analysis is developed to help an analyst identify patterns in a collection of alerts by counting the number of raw alerts that share some common features.

4. OMADM, correlation and multi-step attack detection system

By using the method that we mention in section 3, OMADM, and to evaluate the proposed method, we implement a tool by using Java and MySQL. We call this tool OMADT[1].

For evaluating the proposed method, we use two knowledge bases. The first knowledge base was used in Ning et al. [5]. They produce this knowledge base for DARPA 2000. We create the second knowledge base for evaluating the correctness of the proposed method. We take two multi-step attack scenarios from dataset in [17]. This dataset is not available for public. We also take three multi-step attack scenarios from CTF 2010. For these five multi-step attack scenarios we make a knowledge base and for each of their steps, we generate some true and false alerts. You can see this five scenario in Fig.3.

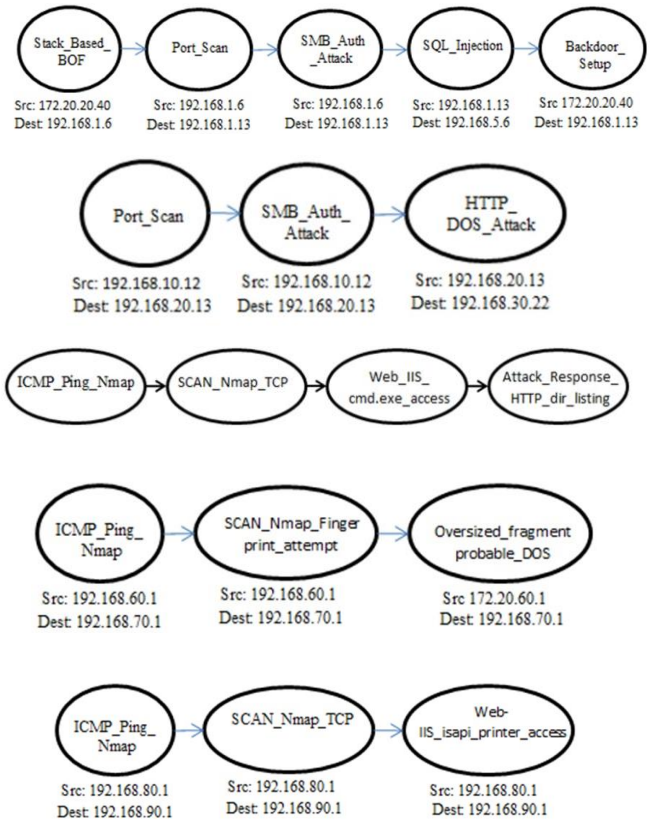


Fig3: Five attack scenarios

We generate alerts for these scenarios in full mode and partial mode (ignore some steps of each scenario), and we give these alerts in an online mode to OMADT. We inject the alerts completely, and OMADT detects the full scenario correctly. We remove some steps of the attack scenario and make it a partial scenario. If the steps of the attack are related to each other, OMADT correlates these steps correctly but if the steps of the attack aren't related to each other, OMADT can't correlate these steps. By using DARPA 2000, we evaluate OMADT. You can see the result in Table 1.

5. Comparing OMADT and TIAA

[1] Online Multi-step Attack Detection Tool

The completeness and soundness of TIAA and OMADT on DARPA 2000 are like each other with this difference; that OMADT is online and TIAA is offline. In the following, we will evaluate the performance of TIAA and OMADM.

For evaluating the performance, we use a computer with Microsoft windows 7 operating system and 8 gigabyte RAM and an Intel core i5 3.2 GHz CPU. The first performance test is the amount of time consumes for correlating the alerts in the DARPA 2000 dataset. You can see the result of this evaluating in table 2. We repeat this test five times for more accuracy, and the amount of consumed time was the same almost.

Table 1.Completeness and Soundness of Alert Correlation

	LLDOS 1.0		LLDOS 2.0.2	
	DMZ	**Inside**	**DMZ**	**Inside**
correctly correlated alerts (OMADT)	54	41	5	12
related alerts (OMADT)	57	44	8	18
correlated alerts (OMADT)	57	44	5	13
completeness measure Rc (OMADT)	94.74%	93.18%	62.5%	66.7%
soundness measure Rs (OMADT)	94.73%	93.18%	100%	92.3%

$$Rs=\frac{Correctly\ Correlated\ Alert}{Correlated\ Alert}\ ,\ Rc=\frac{Correctly\ Correlated\ Alerts}{Related\ Alerts}$$

Table 2. Time consumption

execute time(s)	**Inside1**	**DMZ1**	**Inside2**	**DMZ2**
TIAA	210 sec	200 sec	112 sec	106 sec
OMADT	51 sec	50 sec	31 sec	27 sec
Improvement=Im (%)	75.7%	75%	72.3%	74.5%

As you can see in table 2, OMADT is four times faster than TIAA. We calculate the improvement on percent as Im=(1- $\frac{ours}{TIAA}$)*100.

You can see the average and maximum use of CPU by these tools in table 3.

For getting the average of CPU consumes we perform each test five times and calculate the average of this five test and import this number as the average. For getting the maximum usage of CPU, we perform this test five times and with this regard that the output number of this five test is almost the same we choose a number that is almost in the middle of these five numbers and import this number in the table. As you see in table 3, the maximum of CPU usage by OMADM is lower than TIAA, but the average CPU usage by OMADM is bigger than TIAA. If you pay attention more and compare this table with table 2 you will understand that although the average usage of CPU by OMADM is bigger than TIAA, but if we multiply this usage into the time of usage, the resulted number is much less.

You can see the result of this work in row four and row five in table 3.
You can see the memory usage by OMADM and TIAA in table 4. As you see in this table, the amount of RAM usage by OMADM is constantly 251 MB, but this amount for TIAA is a little more.

Table 3. CPU utilization

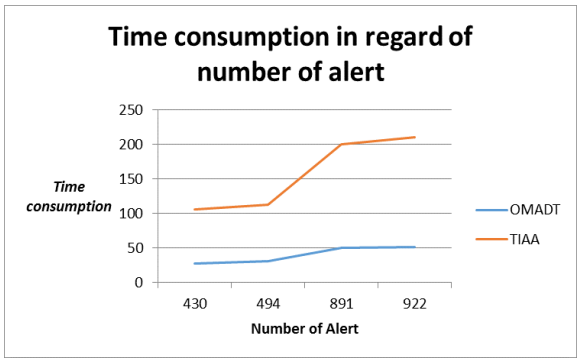
CPU utilization(%)	Inside1		DMZ1		Inside2		DMZ2	
	Avg*	Max	Avg*	Max	Avg*	Max	Avg*	Max
TIAA	0.84	24.94	0.8	29.81	0.9	25.23	0.9	19.19
OMADT	2	17.15	2.27	25.47	2.4	17.15	2.4	14
Change rate	2.38	0.68	2.83	0.85	2.66	0.67	2.66	0.72
Time avg for TIAA	0.84*210=176.4		0.8*200=160		0.9*112=100.8		0.9*106=95.4	
Time avg for OMADT	2.38*51=121.38		2.27*50=112.5		2.4*31=74.4		2.4*27=64.8	
Change rate for Time avg	0.68		0.7		0.73		0.67	
Improvment1	31.1%		29.68%		26.19&		32.07%	
Improvment 2	32%		15%		33%		28%	

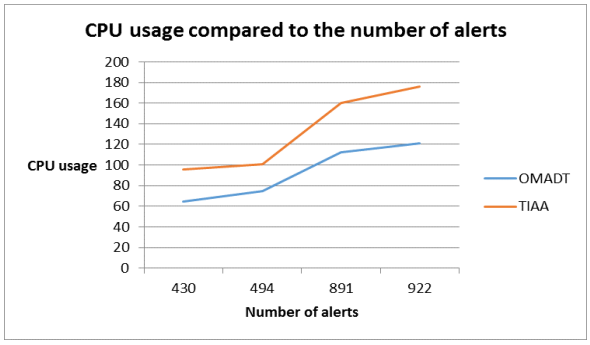
Change Rate= $\frac{OMADT}{TIAA}$ Improvment1= $(1-\frac{\text{Time avg for OMADT}}{\text{Time avg for TIAA}})*100$ Improvment2= $(1-\frac{\text{Max (OMADT)}}{\text{Max (TIAA)}})*100$

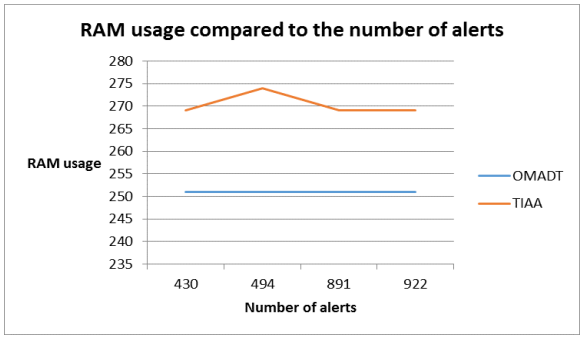
Table 4.Memory usage

RAM utilization(MB)	Inside1		DMZ1		Inside2		DMZ2	
	Avg*	Max	Avg*	Max	Avg*	Max	Avg*	Max
TIAA	269	272	269	270	274	275	269	273
OMADT	251	251	251	251	251	251	251	251
Improvment	6.69%	7.72%	6.69%	7.04%	8.39%	8.72%	6.69%	8.05%

The following charts are a comprehensive compression about the performance of TIAA and OMADM.







Furthermore, in the OMADM output graph because of the changes that we have done, we have some improvement and these changes make the graph briefer and more exact. For example, you can see two graphs for the LLDOS1.0 inside traffic that is generated by OMADM and TIAA in the Fig.4 and fig.5.

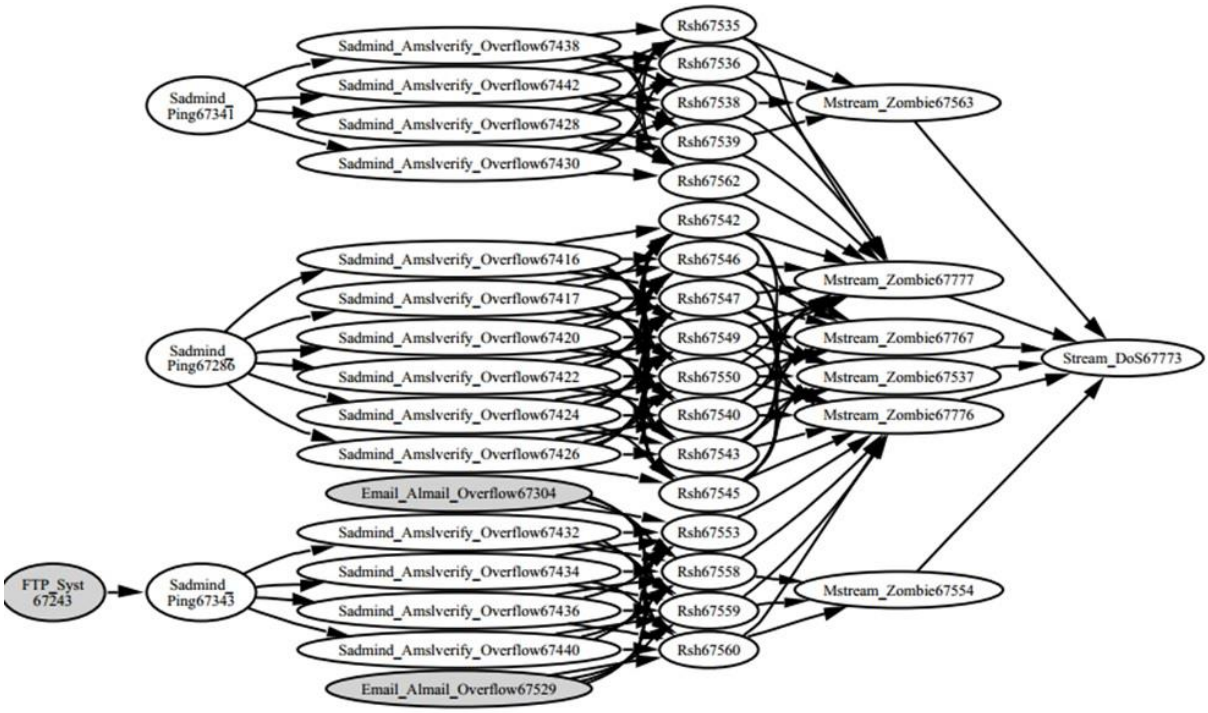


Fig4: TIAA Graph for LLDOS1.0 inside traffic

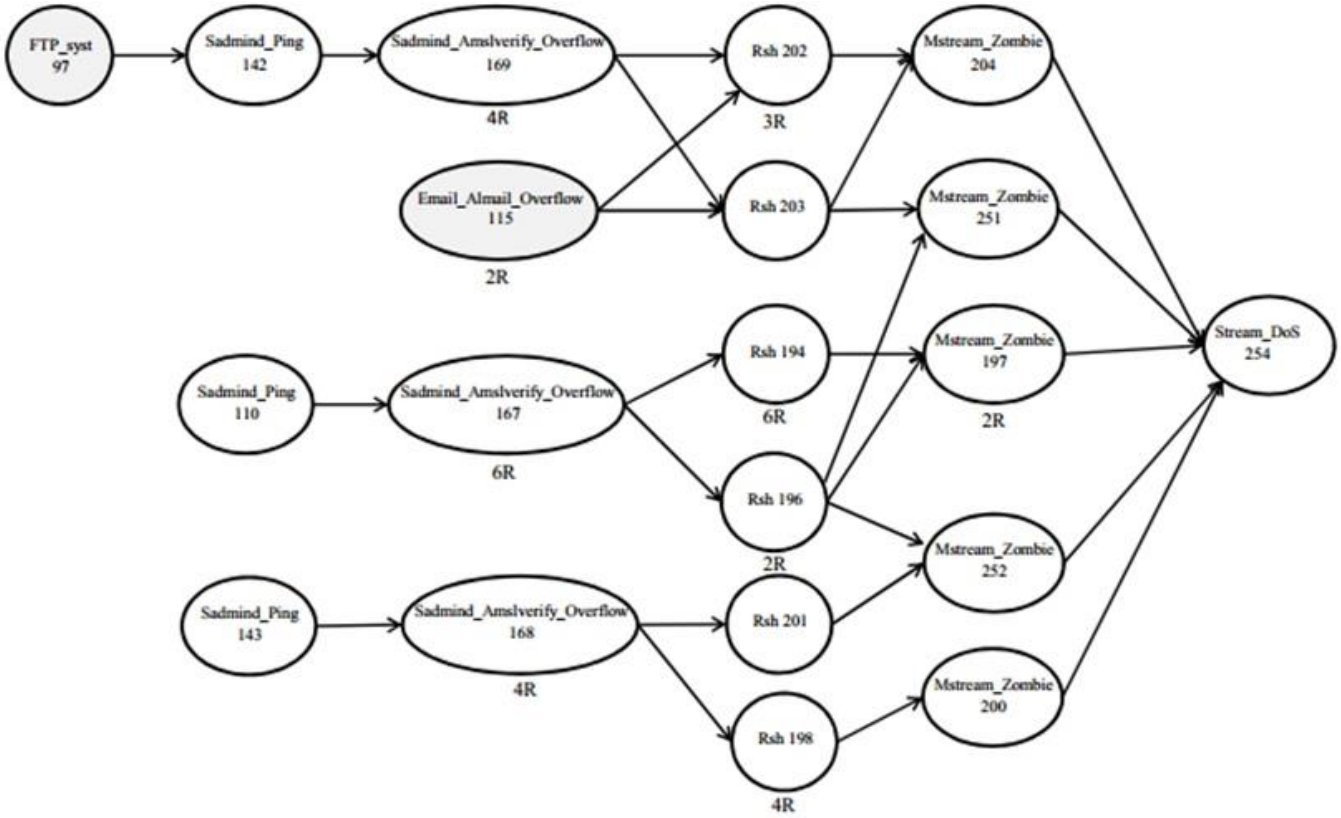


Fig5: OMADM Graph for LLDOS1.0 inside traffic

6. References

[1] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," *Tech. Rep.*, vol. 99, 2000.

[2] P. Kabiri and A. Ghorbani, "Research on Intrusion Detection and Response: A Survey.," *IJ Netw. Secur.*, vol. 1, no. 2, pp. 84–102, 2005.

[3] S. Salah, G. Maciá-Fernández, and J. E. Díaz-Verdejo, "A model-based survey of alert correlation techniques," *Comput. Networks*, vol. 57, no. 5, pp. 1289–1317, Apr. 2013.

[4] Z. Lin, S. Li, and Y. Ma, "Real-Time Intrusion Alert Correlation System Based on Prerequisites and Consequence," *IEEE Wirel. Commun. Netw. Mob. Comput.*, pp. 1–5, Sep. 2010.

[5] P. Ning, Y. Cui, D. S. Reeves, and D. Xu, "Techniques and tools for analyzing intrusion alerts," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 2, pp. 274–318, May 2004.

[6] S. J. Templeton and K. Levitt, "A requires/provides model for computer attacks," *Proc. 2000 Work. New Secur. Paradig. - NSPW '00*, pp. 31–38, 2000.

[7] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework," *IEEE Secur. Privacy, 2002. Proceedings. ...*, pp. 202–215, 2002.

[8] S. Xiao, Y. Zhang, X. Liu, and J. Gao, "Alert fusion based on cluster and correlation analysis," *Converg. Hybrid ...*, pp. 163–168, 2008.

[9] F. Alserhani, M. Akhlaq, I. U. Awan, A. J. Cullen, and P. Mirchandani, "MARS: Multi-stage Attack Recognition System," *2010 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, pp. 753–759, 2010.

[10] A. Valdes and K. Skinner, "Probabilistic Alert Correlation," SRI International, pp. 54–68, 2001.

[11] D. Li, Z. Li, L. Wang, and M. Roesch, "Reducing false positives based on time sequence analysis," Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), 2007.

[12] H. T. Elshoush and I. M. Osman, "Reducing false positives through fuzzy alert correlation in collaborative intelligent intrusion detection systems," International Conference on Fuzzy Systems, Jul. 2010.

[13] 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.

[14] G. Tedesco and U. Aickel in, "Real-Time Alert Correlation with Type Graphs," in Proceedings of the 4th International Conference on Information Systems Security, 2008.

[15] F. Valeur, "Real-time intrusion detection alert correlation," Ph.D. dissertation, Univ. Santa Barbaration, 2006.

[16] F. Valeur, G. Vigna, C. Kruegel, and R. a. Kemmerer, "Comprehensive approach to intrusion detection alert correlation," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 3, Jul. 2004.

[17] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Towards Developing a Systematic Approach To Generate Benchmark Datasets for Intrusion Detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

Authors Profile

Ali Amiri received the B.S. degree in Software engineering from ACECR ISFAHAN UNIVERSITY OF TECHNOLOGY (I.U.T) in 2009 and M.S. degree in Information Technology Security from Malekesahtar University in 2013. He then worked for Security Evaluation Lab for ICT Appliances in Malekesahtar UNIVERSITY. He is currently working in network and security

section in the Iranian public email service (chmail.ir). His research interests include information security, penetration test methodologies and methods, network management and configurations, routing protocols.

Dr. Alireza Nowroozi is assistant professor in Security department of the Malekesahtar University of Technology. His research studies are mainly focused on IT Security, Crisis Management and Decision Making. He earned his BS in Software Engineering from the Ferdowsi University of Mashhad and his MS in Computer Science from Sharif University of Technology. He holds his PhD in Computer Science in Amirkabir University of Technology. He earned the highest GPA during MS and PhD education. He stood first in Azad University's MS entrance exam in AI and ranked second in the state universities' MS entrance exam in Sharif University of Technology in CS. His PhD and MS students are now active in security evaluation, malware analysis, network security and penetration testing. He manages the Security Evaluation Lab of Network Appliances.

