False Alert Reduction and Correlation for Attack Scenarios with Automatic Time Window

M. Logaprakash
Department of CSE (PG)
Sri Ramakrishna Engineering College
Coimbatore, India

Abstract - The Intrusion Detection system (IDS) will provide alerts for the attacks happened in the network. Managing and analyzing vast amount of the low level alerts are very difficult for network administrator. And also false alerts are raised from IDS. False alert reduction method has been proposed to reduce the number of false alerts raised by IDS. And also alert correlation method has been implemented to correlate the relation between the alerts. The model presented in this paper consists of two methods they are: (1) Static filter and (2) adaptive filter. The alert correlation method contains two parts they are: (1) Attack graph based method and (2) similarity based method. Both the methods are work co-operatively. In order to update the attack graph, a technique which is actually the most salient feature of this model capability of hypothesizing missed exploits and discovering defects in pre and post conditions of known exploits in attack graphs has been presented. Additional method named alerts bisimulation for compressing graphs of correlated alerts has been proposed.

Key Terms – Intrusion Detection System, Network security, False alert reduction, Alert correlation.

1 Introduction

In recent years the race for network security is never ending along with the hidden players. Security has always been a great concern about networks because without security, networks lose their advantages such that people and companies prefer to use old-fashioned ways of communication which at least are not as vulnerable as networks to security. The intrusion detection system (IDS) is part of the race. IDS monitor the entire network and produce alerts when any exploit happen in the network. IDS report the attack to the network administrator as an alert. There are two types of IDSs: (1) signature-based and (2) anomaly-based IDSs. Signature-based IDSs have a huge database

Mr. J. Selvakumar
Assistant Professor, Department of CSE (PG)
Sri Ramakrishna Engineering College
Coimbatore, India

about known attacks and if they can match any behavior in a network with the signature of a known attack in their signature database, they recognize the behavior as an adverse action and raise some alerts.

On the contrary, anomaly-based IDSs learn about any normal behavior in networks and then, they raise alerts when they detect the network deviates from the normal behavior. Something that is common between these two types of IDSs is their outcome: alerts. The number of alerts raised by IDSs is too many and also they are too low-level to be properly analyzed by administrators. Furthermore, a noticeable part of these alerts are false positive that can easily report false attacks to administrators. To address these problems, alert aggregation and clustering, alert correlation, false alert reduction, alert verification, alert prioritization and other alert management methods have been proposed. The difference between alert correlation techniques and other alert management techniques is that they try to find such relations between alerts that show the progress of attacks occurring in the network.

Alert clustering and aggregation methods try to cluster alerts based on different factors. For example, Qin and Lee [2] use alerts attributes such as source and destination IPs, timestamp etc. to find similar alerts. In [3,4], if two alerts have the same root causes, they are placed in the same group. Similarly, Xu and Ning [5] aggregate alerts using triggering events. There are also methods that employ machine learning techniques for alert clustering. Dain and Cunningham [6] presented a data mining method to find similar alerts. False alert reduction methods reduce the number of false positive alerts. Julisch [4] believes that 90 percent of alerts are false positive and these false alerts usually have a limited number of root causes. Hence, removing the root causes leads to a significant decrease in the number of false positive alerts. Chyssler et al. [7] proposed a technique to filter false alerts.

Alert prioritization methods [8,9] assign a priority to each alert showing the severity of the attack which IDS raised alert for. Porras et al. [9] proposed a tool called MCorrelator that employs topological information of the network and a fact base to estimate a relevance score for each alert, showing probability of success attack for that alert. Then, the priority of alerts are assigned based on relevance scores and some other factors.

Finally, alert verification methods [10] try to find whether corresponding attack of an alert was successful or not.

When studying the range of problems in dealing with security issues in the management network of telecom service providers, the following needs are identified: (1) Reduction of the message counts such that the operator can cope with them. (2) A lower rate of false alarms. (3) Information collection and correlation from various sources to identify indices of attacks. (E.g. The combining of information about the network topology with IDS alarms). (4) Indication of general network "health" with predictive elements so that total service collapse is avoided. The work in this paper addresses the first two issues and to some extent the third issue which is a prerequisite for future sophisticated analysis of alarm data.

2 Alert Correlation

Alert correlation methods aim at finding causal relations between alerts in order to generate high level alerts. An important class of these methods [5, 11] use domain knowledge about relations between alerts like attack graphs. An attack graph is a graph where each node is either an exploit or a security condition. Moreover, if a security condition is a prerequisite for an exploit, there is an edge from the condition to the exploit. Similarly, an edge from an exploit to a condition illustrates that executing the exploit satisfies the condition. We define attack graphs formally in Section 2. An attack-graph-based method correlates alerts using relation between their corresponding exploits in the attack graph. Some methods that are similar to attack-graph-based methods in essence use predefined rules to correlate alerts. Another group of correlation methods [6] try to estimate correlation between alerts using similarity of their attributes. These similarity-based methods usually employ machine learning techniques, too.

2.1 Attack Graph Based Method

Based on the knowledge encoded in an attack graph, two alerts are correlated if the corresponding exploit of one of them provides a condition which is required by the corresponding exploit of the other one. In each exploit of the attack graph is converted to a queue and each condition is converted to a variable. Since the length of queues is one, only the last alert mapped to each exploit is saved in the memory. The reason for this is that two alerts mapping to the same exploit shows the attacker repeated an attack step.

In [17], for each exploit a Breadth-First Search (BFS) is performed by following directed edges. For each edge encountered during the search, a forward pointer is created. Similarly, another search is performed by following directed edges in their reversed direction and a backward pointer is created for each encountered edge. In other words, for every exploit, they store two BFS trees (back-ward and forward). Set of queues, variables and backward and forward pointers of queues is called queue graph which is created offline. At run-time, they first find corresponding exploit of the new alert by mapping it to an exploit in the attack graph and then insert it in its corresponding queue. Then, variables provided by the queue of the new alert are marked to show that they are satisfied. After that, they conduct a BFS by following backward pointers of that queue. By doing this BFS, they find those alerts that are correlated with the new alert because backward pointers include queues whose exploits provide preconditions of the exploit of the new alert. Although BFS is a quadratic time algorithm, this search takes linear time because it is carried out on a tree created offline not on a graph.

Every time we hypothesize an exploit in the attack graph, we should insert its post-conditions in the attack graph. Moreover, each of its pre-conditions which could not be satisfied gets *HYP* value because when an exploit is executed, all of its pre-conditions should be satisfied in advance. There is an important point we did not discuss so far in our algorithms in order to avoid complexity. When we have to use the function *corAlerts* to correlate an alert, it indicates there is a problem in modeling exploit of that alert. This problem can cause more problems in the future because correlation of the next alerts with this alert can be missed due to this weakness in modeling. Hence, correlation of this type of alerts is required to be investigated using our similarity-based method.

2.2 Similarity Based Method

This module does the task of helping previous method in two states: (1) corresponding exploit of the new alert does not exist in the attack graph; (2) Attack graph method cannot find those alerts in the result graph that are correlated with the new alert. In this method, we extend the method presented in the paper. Central to this module is the fact that correlated alerts are always similar in terms of their attributes. This similarity could be in their IPs, Ports etc. To recognize whether two alerts are correlated, a similarity vector is created for them. Then, the similarity vector is fed into a trained classifier and the classifier returns a probability which shows how strong these two alerts are correlated. If the probability is less than a threshold, alerts are not correlated.

3 Alert Reduction

There are many triggers for alarms that should be investigated, for example: alarms with high severity, hosts with a lot of alarms, hosts with a lot of different alarms, unusual events, high rate of alarms and strange payload in "normal" alarms. Reviewing how a security expert works, some features can be noted that will be reflected in our agents that mimic the behavior: (1) Port scans are aggregated so that each port scan only generates one alarm. (2) When analyzing the alarms, all three sources are used around the time of interest. (3) Knowledge about

the topology of the network is captured to assist the decisions.

3.1 Static Filter

Studying the output of the IDSs when no attacks are launched shows that there are a lot of uninteresting alarms coming from Snort. Even though each of them has a low severity, their severity sum is high depending on their vast numbers. For example, each time a file is checked for changes, an alarm is produced. Based on this study filters were inserted to remove the uninteresting alarms, such as messages that Snort checks a file, filters for messages that arose due to current misconfigurations, and a filter that removed port scan alarms in Snort. Therefore static filters exclude data from Snort that does not carry any valuable information.

Filters are implemented either as an *ignore* filter or as a *delete* filter. The ignore filters keep the alarms in the database but they are not forwarded to the next agent. But they are saved for forensic investigation. The delete filters removes alarms permanently from the database. All static filters have a limited duty cycle defined by the operator and have to be reactivated afterwards.

3.2 Adaptive Filter

Manually studying the output of the IDSs and adding filters can solve the problem with uninteresting and misconfiguration messages for the current set-up. However, since it is not possible to foresee all future misconfigurations, adaptive filtering algorithms were implemented to update the filters. An automatic filter detector based on Naïve Bayesian (NB) learning was designed for Snort. The idea is to train a NB text classifier to classify messages by looking at the words that appear in messages.

The following diagram shows the overview of proposed system,

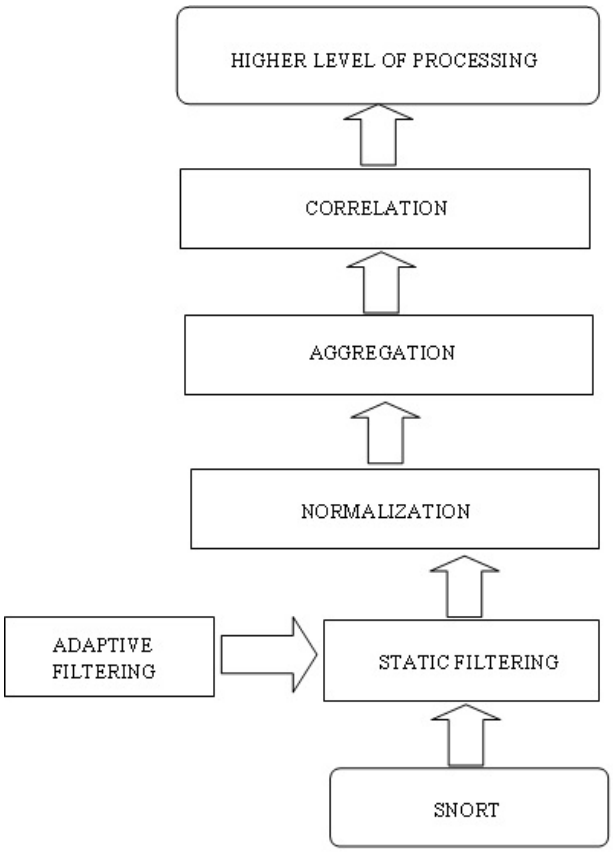

Fig.1 - Overview of Proposed System.

The NB-classifier is first trained by presenting a number of messages labeled as interesting or not. The meaning of interesting is a message that can be useful when analyzing it for signs of an attack. During training, a knowledge base is built up by counting the occurrences of the words in the message. After training, the on-line classification is based on computing the probability that an object belongs to a class based on how common the values of the attributes (words) are in that class within the training data. During training period the NB classifier is to assess whether the unknown messages are interesting. For performance reasons, the algorithm for adaptive filtering is launched on a periodic basis. The algorithm suggests filter rules for top scoring events to a human expert via HMI. The reply is also used to optimize the training corpus, achieving on-line retraining. In order to avoid over learning effects, features, hardly ever occurring in time, are reduced in their significance and are finally forgotten.

3.3 Aggregation

Repeated, identical alarms do not provide any additional information. It would reduce the information overload if each all these alarms were represented in only one alarm including the number of its frequency. The relevant fields for aggregation have to be defined for each source individually. For Snort the source and destination IP, as well the server ports and the messages are relevant. Ephemeral ports can be ignored. For Snort, the PID number is unimportant but the program and the message field is relevant. All data is aggregated within a given time window.

This method does not reduce the false alarm rates, but can help to keep the alarm rates down in order not flood the receiver. Obviously, the larger the time window the fewer alarms are left after aggregation. However, higher level network correlation agents are dependent on recognition of anomalous situations within short enough times for reaction.

3.3 Hyper Alert

As discussed, our model creates a few hyper alerts in addition to the result graph. A hyper alert contains those alerts that are correlated based on only similarity of their features. However, a network administrator may want to analyze these hyper alerts besides the result graph. By investigating created hyper alerts, he may discover root causes of false alerts or even new attack patterns.

However, hyper alerts still contain many alerts with different relationships with each other. A noticeable number of alerts in each hyper alert are redundant and does not have any added value for the network administrator. For example, IDS might raise some alerts with the same type whose correlations with other alerts are very similar. We can eliminate some of these similar alerts to compress hyper alerts.

To compress a hyper alert ha_i, merge every two alerts in ha_i that are bisimilar. Merge two alerts by replacing them with an alert set containing merged alerts. When it wants to investigate the correlation of a new alert and an alert set using the similarity-based method, it is enough to compare the new alert with the last alert in the alert set. Hence, compressing the hyper alerts decreases the number of comparisons as well. By merging bisimilar alerts, number of nodes and edges in hyper alerts reduces considerably without losing important information. Although, it is possible that we merge two alerts having different sources and destinations, they probably are not distinguishable for the network administrator due the essence of bisimulation relation. We can say we just remove repetitive information.

4. Experimental Results

In this section, we first test our model using DARPA2000 dataset to demonstrate how it works. Despite the fact that DARPA2000 is a 12-year-old dataset, it has been used in many papers since it was introduced and it is the only choice to compare alert reduction and correlation methods. Also, we show the ability of the model to complete a partial attack graph. Moreover, we evaluate the two modules when they work alone and compare them with the proposed model as a whole where the two modules cooperate with each other. Finally, we test the model on 4 intensive datasets to show its performance.

4.1 DARPA2000 Data set

In order to evaluate the model, we use DARPA2000 dataset. Unfortunately, DARPA has not proposed the attack graph of the network which was used for creating their dataset. However, we created its attack graph by looking through the labeled alerts produced by DARPA and rules between pre and post conditions of attack steps proposed by Ning et al. [21] . To generate alerts, used Snort because it is a well-documented IDS with well defined signatures.

There are two attack scenarios in DARPA2000 dataset, LLDOS1.0 and LLDOS2.0.2. In the both scenarios, a novice attacker tries to install components necessary to run a Distributed Denial of Service, and then launch a DDOS at a US government site. The main difference between 2.0.2 and 1.0 is that in 2.0.2 the attacker probes for host operating system by doing DNS HINFO queries, rather than sweeping IP's and rpc ports, and that they break-into one host first, then fan out from there, rather than attacking each host individually.

The followings are some of the exploits and security conditions identified from the DARPA2000 dataset,

4.1.1 Exploits

- **Sadmind-Ping (machine1, machine2)**: using the ''ping'' option of the sadmind exploit program on machine2 from machine1.
- **Remote2Root (machine1, machine2)**: execution of sadmind Remote-To-Root exploit on machine2 from machine1.
- **RemoteShell (machine1, machine2)**: execution of a shell command from machine1 using a trust relationship between the user on machine1 and the user account on machine2.
- **MStreamZombie (machine1)**: communications between an mstream master and zombie (machine1).
- **Stream-DoS**: sending a high volume of TCP packets with ACK flag set to a host.

1. Security Conditions

- **vul (v1, machine1)**: vulnerability v1 exists on machine1.
- **Access (machine1)**: root access on machine1.
- **ReadyToLunchDDoS (machine1)**: machine1 is ready to a lunch a DDoS attack.

Topic	Classification Algorithm		
	Naïve Bayes (%)	LogitBoost (%)	Decision Stump(%)
Alerts	98.50	93.00	27.50

Table I. Result of Alert Reduction (weka)

5. Conclusion

In this project an alert reduction method has been proposed to reduce the amount of false alerts arise in IDS. The reduced alerts are correlated by Alert correlation method to minimize the number of alerts given to network administrator by the IDS. Static filter and adaptive filter are the two sensors used to identify the false alerts generated. The attack graph based method and similarity based method is proposed to correlate the alerts. Moreover, defects of the attack graph have been detected

and updated to always have up-to-date knowledge about attacks. The new concept of *alerts bisimulation* to compress groups of correlated alerts called hyper alerts has been presented.

REFERENCES

1) Seyed Hossein .A, Saeed .J, Mahdi .A, "*A hybrid model for correlating alerts of known and unknown attack scenarios and updating attack graphs*", Computer Networks, vol: 55,Elsvier,2011,pp: 2221–2240.

2) X. Qin, W. lee, Statistical causality analysis of infosec alert data, in: 6th International Symposium on Recent Advances in Intrusion Detection (RAID), Pittsburgh PA, vol. 2820, 2003, pp. 73–93.

3) K. Julisch, Clustering intrusion detection alarms to support root cause analysis, ACM Transactions on Information and System Security(TISSEC) 6 (4) (2003) 443–471.

4) K. Julisch, Mining alarm clusters to improve alarm handling efficiency, in: 7th Annual Computer Security Applications Conference (ACSAC), IEEE Computer Society Washington, DC, USA, New Orleans, LA, 2001, pp. 12–21.

5) D. Xu, P. Ning, Alert correlation through triggering events and common resources, in: 20th Annual Computer Security Applications Conference(ACSAC), IEEE Computer Society Washington, DC, USA, 2004, pp. 360–369.

6) O. Dain, R.K. Cunningham, Fusing a heterogeneous alert stream into scenarios, in: ACM workshop on Data Mining for Security Applications, 2001, pp. 1–13.

7) T. Chyssler, S. Nadjm-Tehrani, S. Burschka, K. Burbeck, Alarm reduction and correlation in defence of ip networks, in: 13th IEEE International Workshops on Enabling Technologies, 2004, pp. 229– 234.

8) F. Valeur, G. Vigna, C. Kruegel, R.A. Kemmerer, A comprehensive approach to intrusion detection alert correlation, IEEE Transaction On Dependable and Secure Computing 1 (3) (2004) 146–169.

9) P.A. Porras, M.W. Fong, A. Valdes, A mission-impact-based approach to infosec alarm correlation, in: Recent Advances in Intrusion Detection (RAID), LNCS, vol. 2516, Springer, Berlin/Heidelberg, 2002, pp. 95–114.

10) C. Kruegel, W. Robertson, Alert verification: determining the success of intrusion attempts, in: 1st Workshop on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), Germany, 2004, pp. 1–14.

11) H. Debar, A. Wespi, Aggregation and correlation of intrusion-detection alerts, 4th International Symposium on Recent Advances in Intrusion Detection(RAID), vol. 2212, Springer-Verlag, London, UK, 2001, pp. 85–103.

12) P. Ning, Y. Cui, An Intrusion Alert Correlator Based on Prerequisites of Intrusions, Technical report, Department of Computer Science, North Carolina State University, 2002.

13) P. Ning, D. Xu, Learning attack strategies from intrusion alerts, in: 10th ACM Conference on Computer and Communications Security, ACM, New York, USA, Washington D.C., USA, 2003, pp. 200–209.

14) F. Cuppens, A. Mige, Alert correlation in a cooperative intrusion detection framework, in: 23rd IEEE Symposium on Security and Privacy, IEEE Computer Society, Washington, DC, USA, 2002, pp. 202–215.

15) S.J. Templeton, K. Levitt, A requires/provides model for computer attacks, in: 3rd Workshop on New Security Paradigms, ACM New York, USA, Ballycotton, County Cork, Ireland, 2000, pp. 31–38.

16) Z. Jimmy, M. Heckman, B. Reynolds, A. Carlson, B.M., Modeling network intrusion detection alerts for correlation, in: ACM Transactions on Information and System Security, 10(1), 2007, pp. 1–31.

17) A. Siraj, R.B. Vaughn, A cognitive model for alert correlation in a distributed environment, in: Intelligence and Security Informatics, LNCS, vol. 3495, Springer, Berlin/ Heidelberg, 2005, pp. 218–230.

18) O.M. Dain, R.K. Cunningham, Building scenarios from a heterogeneous alert stream, in: IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, vol.6, 2002, pp. 231–235.

19) A.M. Makhlouf, N. Boudriga, Multi-violation detectors-an algebraic tool for alert correlation and intrusion detection, in: 2nd Information and Communication Technologies Conference(ICTTA), vol. 2, 2006, pp. 3181–3186.

20) B. Zhu, A. Ghorbani, Alert correlation for extracting attack strategies, International Journal of Network Security 3 (3) (2006) 244–258.

21) E. Zambon, D. Bolzoni, Network Intrusion Detection Systems. False Positive Reduction Through Anomaly Detection. <http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zambon.pdf>.

AUTHORS PROFILE

M. Logaprakash received the B.Tech degree in Information Technology from the Anna University, Chennai. He is currently a PG scholar in Department of CSE (PG) at Sri Ramakrishna Engineering College, Coimbatore.

J. Selvakumar received the B.E degree in computer Science & Engineering from the Madras University in May 2001 and the M.E degree in Computer Science & Engineering from the Bharathiyar University in December 2002. He is currently working towards the Ph.D degree at the Anna University, Chennai. He is currently an Assistant Professor. His research interests in Requirements Engineering.