Dynamic Intrusion alerts generation and Aggregation using Intelligent IDS

Mrs.Sudha Singaraju*1, G.Srikanth*2

Assistant Professor, Dept of Computer Applications, SNIST, Ghatkesar, Hyderabad, AP, India

M.C.A Student, Dept of Computer Applications, SNIST, Ghatkesar, Hyderabad, AP, India

ABSTRACT:

The essential subtask of intrusion detection is Alert aggregation. Protecting our data in the internet is a great risk. Intruders and hackers are always ready grab our data. To identify unauthorized users and to cluster different alerts produced by low-level intrusion detection systems firewalls, Intrusion detection system has been introduced. The relevant information whereas the amount of data can be reduced substantially by Meta-alters which will be generated for the clusters. At a certain point in time which has been initiated by an attacker is belonging to a specific hacking. For communication within a distributed intrusion detection system the meta-alerts may be the basis for reporting to security experts. In this paper, for online alert aggregation we propose a novel technique which is based on a dynamic and probabilistic model of current attack situation. For the estimation of the model parameters, it can be regarded as a data stream version of a maximum likelihood approach. The first alerts, which are belonging to a new attack instance, are generated with meta-alerts with a delay of typically only a few seconds. To achieve
Reduction rates while the number of missing meta-alerts is extremely low can be possible with the three benchmark data sets are demonstrated.

KEYWORDS: Intrusion Detection System, Alert Aggregation, different layers, Meta alerts.

I.INTRODUCTION:

Intrusion detection is the problem of identifying unauthorized use, misuse, and abuse of computer system. The intruders are people with malicious intensions. Their aim is to break security of IT systems for monetary and other gains. At present, most IDS are quite reliable in detecting suspicious actions by evaluating TCP/IP connections or log files, for instance. Once AN ID finds a suspicious action, it immediately creates an alert which contains information about the source, target, and estimated type of the attack (e.g., SQL injection, buffer overflow, or denial of service). As the intrusive actions caused by a single attack instance which is the occurrence of an attack of a particular type that has been launched by a specific attacker at a certain point in time are often spread over many network connections or log file entries, a single attack instance often results in hundreds or even thousands of alerts.

IDS generally detect attack types and takes appropriate actions. In the process of detection the IDSs provide many alerts including false alerts. The alerts might have different features such as false positives and true positives. When flood of alerts are created by IDSs in order to prompt security administrators the happening in the network, it is not easy to interpret each and every alert and come to a conclusion about the risk, severity of risk and the protection measures. Moreover security personnel may take wrong decisions due to false positives in the alerts and their inability to correctly interpret the bulk of alerts raised by the system. This is the motivation behind this paper. This paper aims at aggregating the flood of security alerts and provides concise feedback to security personnel so as to enable them to take actions quickly. Our approach has the following distinct properties. It is a generative modeling approach using probabilistic methods. Assuming that attack instances can be regarded as random processes “producing” alerts, we aim at modeling these processes using approximate maximum likelihood parameter estimation techniques. Thus, the beginning as well as the completion of attack instances can be detected. . It is a data stream approach, i.e., each observed alert is processed only a few times. Thus, it can be applied online and under harsh timing constraints.

II.RELATED WORK:

In the existing system, one of the main drawbacks is large amount of alerts produced. To detect attacks with high accuracy the existing IDS are optimized. However, the IDS have been outlined in a number of publications as it has various disadvantages and a lot of work has been done to analyze IDS in order to direct future research. The correlation of alerts from IDS focuses on the recent research. All approaches outlined in the following present either online algorithms or as we see it if not stated otherwise, can easily be extended to an online version. The attack thread reconstruction is the one step presented correlation approach, which can be seen as a kind of attack instance recognition.

A strict sorting of alerts within a temporal window of fixed length according to the source, destination, and attack classification is used but no clustering algorithm is used. Alerts that share the same quadruple of source and destination address as well as source and destination port is used to eliminate duplicates. In order to provide a more condensed view of the current attack situation the definition of such situations is also used in to cluster alerts. A group of alerts belong to alert clustering or even though the same attack occurrence is used called clustering and there is no clustering algorithm in a classic sense. A similarity relation which is based on expert rules is used to group similar alerts together as the alerts from one IDS are stored in a relational database. For instance, with imperfect classifiers as two alerts are defined to be similar as these approaches are likely to fail under real-life conditions, with false alerts or wrongly adjusted time windows.. A weighted, attribute wise similarity operator is used to decide whether to fuse two alerts or not. This approach suffers from the high number of parameters that need to be set.

The similarity operator presented in has the same disadvantage there are lots of parameters that must be set by the user and there is no or only little guidance in order to find good values. In, another clustering algorithm that is based on attribute-wise similarity measures with user defined parameters is presented. However, a closer look at the parameter setting reveals that the similarity measure, in fact, degenerates to a strict sorting according to the source and destination IP addresses and ports of the alerts. The drawbacks that arise thereof are the same as those mentioned above. In, three different approaches are presented to fuse alerts. The first, quite simple one groups alerts according to their source IP address only. The other two approaches are based on different supervised learning techniques. Besides a basic least-squares error approach, multilayer perceptions, radial basis function networks, and decision trees are used to decide whether to fuse a new alert with an already existing meta-alert (called scenario) or not. Due to the supervised nature, labelled training data need to be generated which could be quite difficult in case of various attack instances.

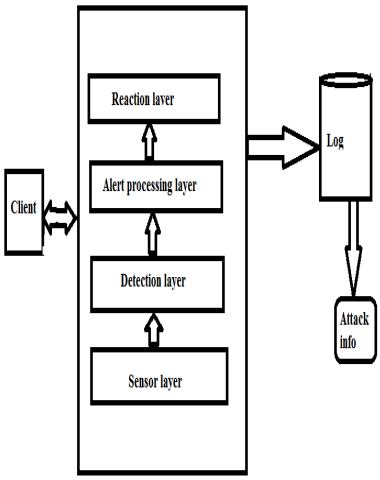
In, an offline clustering solution based on the CURE algorithm is presented. The solution is restricted to numerical attributes. In addition, the number of clusters must be set manually. The alert clustering solution described in is more related to ours. A link-based clustering approach is used to repeatedly fuse alerts into more generalized ones. The intention is to discover the reasons for the existence of the majority of alerts, the so called root causes, and to eliminate them subsequently. Attack instances that result in a small number of alerts (such as PHF or FFB) are likely to be ignored completely. The main difference to our approach is that the algorithm can only be used in an offline setting and is intended to analyze historical alert logs. In contrast, we use an online approach to model the current attack situation. The alert clustering approach described in is based on but aims at reducing the false positive rate. The created cluster structure is used as a filter to reduce the amount of created alerts. Those alerts that are similar to already known false positives are kept back, whereas alerts that are considered to be legitimate (i.e., dissimilar to all known false positives) are reported and not further aggregated. The same idea—but based on a different offline clustering algorithm—is presented in [20]. An offline clustering solution based on the CURE algorithm is presented. The solution is restricted to numerical attributes. In addition, the number of clusters must be set manually. This is problematic, as in fact it assumes that the security expert has knowledge about the actual number of ongoing attack instances. The alert clustering solution is more related to ours. A link-based clustering approach is used to repeatedly fuse alerts into more generalized ones. The intention is to discover the reasons for the existence of the majority of alerts, the so-called root causes, and to eliminate them subsequently. An attack instance in our sense can also be seen as a kind of root cause, but in root causes are regarded as “generally persistent” which does not hold for attack instances that occur only within a limited time window.

III.PROBLEM DEFINITION:

Network Security is an important issue in our current Task. Providing certain firewall, antivirus, security for password become hectic even though we cant control the network type of anomalies or misuse activities. To maintain the security issue at a time we introduce a novel method call online alert aggregation which is based on dynamic or probabilistic model of the current attack situation.

IV.Application Specific for Alert Aggregation:

Server: Server module is the main module for this project. This module acts as the Intrusion Detection System. This module consists of four layers sensor layer (which detects the user/client etc.), Detection layer, alert processing layer and reaction layer. In addition there is also Message Log, where all the alerts and messages are stored for the references. This Message Log can also be saved as Log file for future references for any network environment.



intrusion detection system

4.1Client: Client module is developed for testing the Intrusion Detection System. In this module the client can enter only with a valid user name and password. If an intruder enters with any guessing passwords then the alert is given to the Server and the intruder is also blocked. Even if the valid user enters the correct user name and password, the user can use only for minimum number of times. For example even if the valid user makes the login for repeated number of times, the client will be blocked and the alert is sent to the admin. In the process level intrusion, each client would have given a specific process only. For example, a client may have given permission only for P1process. If the client tries to make more than these processes the client will be blocked and the alert is given by the Intrusion Detection System. In this client module the client can be able to send data. Here, when ever data is sent Intrusion Detection System checks for the file. If the size of the file is large then it is restricted or else the data is sent.

4.2 DARPA Dataset: This module is integrated in the Server module. This is an offline type of testing the intrusions. In this module, the DARPA Data Set is used to check the technique of the Online Intrusion Alert Aggregation with Generative Data Stream Modeling. The DARPA data set is downloaded and separated according to each layers. So we test the instance of DARPA Dataset using the open file dialog box. Whenever the dataset is chosen based on the conditions specified the Intrusion Detection System works.

4.3Attack Simulation: In this module, the attack simulation is made for ours elf to test the system. Attacks are classified and made to simulate here. Whenever an attack is launched the Intrusion Detection System must be capable of detecting it. So our system will also be capable of detecting such attacks. For example if an IP trace attack is launched, the Intrusion Detection System must detect it and must kill or block the process.

V.ALERT AGGREGATION ALGORITHM:

Step 1: Select the ‘n’ layers needed for the whole IDS.
Step 2: Build Sensor Layer to detect Network and Host Systems.
Step 3: Build Detection Layer based on Misuse and Anomaly detection technique.
Step 4: Classify various types of alerts. (For example alert for System level intrusion or process level intrusion)
Step 5: Code the system for detecting various types of attacks and alerts for respective attacks.

Step 6: Integrate the system with Mobile device to get alerts from the proposed IDS.
Step 7: Specify each type of alert on which category it falls, so that user can easily recognize the attack type.
Step 8: Build Reaction layer with various options so that administrator/user can have various options to select or react on any type of intrusion.

Step 9: Test the system using Attack Simulation module, by sending different attacks to the proposed IDS.
Step 10: Build a log file, so that all the reports generated can be saved for future references.

VI.CONCLUSION:

In this paper we demonstrated online alert aggregation approach. **B**y using three benchmark data sets the reduction rates are extremely low with the number of missing meta-alerts which are demonstrated here. The reduction rate with respect to the number of alerts was up to 99.96 percent in our experiments. The instance detection rate is very high although there are situations that described are especially clusters that are wrongly split. Only very few attack instances were missed. In the future we will also apply our techniques to benchmark data that fuse information from heterogeneous sources

VII. REFERENCES

[1] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Technical Report 99-15, Dept. of Computer Eng., Chalmers Univ. of Technology, 2000.
[2] M.R. Endsley, "Theoretical Underpinnings of Situation Awareness: A Critical Review," Situation Awareness Analysis and Measurement, M.R. Endsley and D.J. Garland, eds., chapter 1, pp. 3-32, Lawrence Erlbaum Assoc., 2000.
[3] C.M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.
[4] M.R. Henzinger, P. Raghavan, and S. Rajagopalan, Computing on Data Streams. Am. Math. Soc., 1999.
[5] A. Allen, "Intrusion Detection Systems: Perspective," Technical Report DPRO-95367, Gartner, Inc., 2003.
[6] F. Valeur, G. Vigna, C. Kru¨ gel, and R.A.Kemmerer, "A Comprehensive Approach to Intrusion Detection Alert Correlation," IEEE Trans. Dependable and Secure Computing, vol. 1, no. 3, pp. 146-169, July-Sept. 2004.
[7] H. Debar and A. Wespi, "Aggregation and Correlation of Intrusion-Detection Alerts," Recent Advances in Intrusion Detection, W. Lee, L. Me, and A. Wespi, eds., pp. 85-103, Springer, 2001.
[8] D. Li, Z. Li, and J. Ma, "Processing Intrusion Detection Alerts in Large-Scale Network," Proc. Int'l Symp. Electronic Commerce and Security, pp. 545- 548, 2008.
[9] F. Cuppens, "Managing Alerts in a Multi-Intrusion Detection Environment," Proc. 17th Ann. Computer Security Applications Conf. (ACSAC '01), pp. 22-31, 2001.
[10] A. Valdes and K. Skinner, "Probabilistic Alert Correlation," Recent Advances in Intrusion Detection, W. Lee, L. Me, and A. Wespi, eds. pp. 54-68, Springer, 2001.
[11] K. Julisch, "Using Root Cause Analysis to Handle Intrusion Detection Alarms," PhD dissertation, Universita¨ t Dortmund, 2003.
[12] T. Pietraszek, "Alert Classification to Reduce False Positives in Intrusion Detection," PhD dissertation, Universita¨ t Freiburg, 2006.
[13] F. Autrel and F. Cuppens, "Using an Intrusion Detection Alert Similarity Operator to Aggregate and Fuse Alerts," Proc. Fourth Conf. Security and Network Architectures, pp. 312-322, 2005.
[14] G. Giacinto, R. Perdisci, and F. Roli, "Alarm Clustering for Intrusion Detection Systems in Computer Networks," Machine Learning and Data Mining in Pattern Recognition, P. Perner and A. Imiya, eds. pp. 184-193, Springer, 2005.
[15] O. Dain and R. Cunningham, "Fusing a Heterogeneous Alert Stream into Scenarios," Proc.2001 ACM Workshop Data Mining for Security Applications, pp. 1-13, 2001.
[16] P. Ning, Y. Cui, D.S. Reeves, and D. Xu, "Techniques and Tools for Analyzing Intrusion Alerts," ACM Trans. Information Systems Security,vol. 7, no. 2, pp. 274-318, 2004.
[18] S.T. Eckmann, G. Vigna, and R.A. Kemmerer,"STATL: An Attack Language for State- Based Intrusion Detection," J. Computer Security, vol. 10, nos. 1/2, pp. 71-103, 2002.
[19] M.S. Shin, H. Moon, K.H. Ryu, K. Kim, and J.Kim, "Applying Data Mining Techniques to Analyze Alert Data," Web Technologies and Applications, X
[20] J. Song, H. Ohba, H. Takakura, Y. Okabe, K.Ohira, and Y. Kwon, "A Comprehensive Approach to Detect Unknown Attacks via Intrusion Detection Alerts," Advances in Computer Science—ASIAN 2007, Computer and Network Security, I. Cervesato, ed., pp. 247-253, Springer, 2008.