*International Journal of Research in Advent Technology (IJRAT) Special Issue*
*E-ISSN: 2321-9637*
*National Conference "MOMENTUM-17", 14th & 15th February 2017*
*Available online at www.ijrat.org*

# Data Streaming Online Intrusion Alert System

Mrs. Shelake R.M.[1], Prof. Sushinder Reddy [2]
*P.G. Student,ComputerDept, MIT Ishnapure, Patancheru, Hyderabad(India),*
*Computer Dept, ,MIT Ishnapure, Patancheru, Hyderabad(India)[2]*
*shelakerekha@gmail.com[1], sushinderreddy1@gmail.com[2]*

**Abstract-**
Intrusion detection systems (IDS) are besides other protective measures such as virtual private networks, authentication mechanisms, or encryption techniques very important to guarantee information security. They help to defend against the various threats to which networks and hosts are exposed to by detecting the actions of attackers or attack tools in a network or host-based manner with misuse or anomaly detection techniques Basically, it can be regarded as a datastream version of a maximum likelihood approach for the estimation of the model parameters. With three benchmark data sets, we demonstrate that it is possible to achieve reduction rates of up to 99.96 percent while the number of missing meta-alerts is extremely low. In addition, meta-alerts are generated with a delay of typically only a few seconds after observing the first alert belonging to a new attack instance..

***Index Terms—Intrusion detection, alert aggregation, generative modeling, data stream algorithm.***

## 1. INTRODUCTION

INTRUSION protective measures such as virtual private networks, detection systems (IDS) are besides other authentication mechanisms, or encryption techniques very important to guarantee information security. They help to defend against the various threats to which networks and hosts are exposed to by detecting the actions of attackers or attack tools in a network or host-based manner with misuse or anomaly detection techniques [1]. At present, most IDS are quite reliable in detecting suspicious actions by evaluating TCP/IP connections or log files, for instance. Once an IDS 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 usually focus on detecting attack types, but not on distinguishing between different attack instances. In addition, even low rates of false alerts could easily result in a high total number of false alerts if thousands of network packets or log file entries are inspected. As a consequence, the IDS creates many alerts at a low level of abstraction. It is extremely difficult for a human security expert to inspect this flood of alerts, and decisions that follow from single alerts might be wrong with a relatively high probability. In our opinion, a "perfect" IDS should be situation-aware [2] in the sense that at any point in time it should "know" what is going on in its environment regarding attack instances (of various types) and attackers. In this paper, we make an important step toward this goal by introducing and evaluating a new technique for alert aggregation. Alerts may originate from low-level IDS such as those mentioned above, from firewalls (FW), etc. Alerts that belong to one attack instance must be clustered together and meta-alerts must be generated for these clusters. The main goal is to reduce the amount of alerts substantially without losing any important information which is necessary to identify ongoing attack instances. We want to have no missing metaalerts, but in turn we accept false or redundant meta-alerts to a certain degree. This problem is not new, but current solutions are typically based on a quite simple sorting of alerts, e.g., according to their source, destination, and attack type. Under real conditions such as the presence of classification errors of the low-level IDS (e.g., false alerts), uncertainty with respect to the source of the attack due to spoofed IP addresses, or wrongly adjusted time windows, for instance, such an approach fails quite often.

Our approach has the following distinct properties:

- It is a generative modeling approach [3] using probabilistic methods. Assuming that attack instancescan be regarded as random processes "producing" alerts, we aim at modelingtheseprocessesusing approximative maximum likelihood parameter estimation techniques. Thus, the beginning as well as the completion of attack instances can be detected.

*International Journal of Research in Advent Technology (IJRAT) Special Issue*
*E-ISSN: 2321-9637*
*National Conference "MOMENTUM-17", 14th & 15th February 2017*
*Available online at www.ijrat.org*

- It is a data stream approach, i.e., each observed alert is processed only a few times [4]. Thus, it can be applied online and under harsh timing constraints.

The remainder of this paper is organized as follows: In Section 2 some related work is presented. Section 3 describes the proposed alert aggregation approach, and Section 4 provides experimental results for the alert aggregation using various data sets. Finally, Section 5 summarizes the major findings.

## 2. LITERATURE REVIEW

Most existing IDS are optimized to detect attacks with high accuracy. However, they still have various disadvantages that have been outlined in a number of publications and a lot of work has been done to analyze IDS in order to direct future research [5], for instance). Besides others, one drawback is the large amount of alerts produced. Recent research focuses on the correlation of alerts from (possibly multiple) IDS. If not stated otherwise, all approaches

outlined in the following present either online algorithms or as we see it can easily be extended to an online version. Probably, the most comprehensive approach to alert correlation is introduced in [6]. One step in the presented correlation approach is attack thread reconstruction, which can be seen as a kind of attack instance recognition. No clustering algorithm is used, but a strict sorting of alerts within a temporal window of fixed length according to the source, destination, and attack classification (attack type). In [7], a similar approach is used to eliminate duplicates, i.e., alerts that share the same quadruple of source and destination address as well as source and destination port. In addition, alerts are aggregated (online) into predefined clusters (so-called situations) in order to provide a more condensed view of the current attack situation. The definition of such situations is also used in [8] to cluster alerts. In [9], alert clustering is used to group alerts that belong to the same attack occurrence. Even though called clustering, there is no clustering algorithm in a classic sense. The alerts from one (or possibly several) IDS are stored in a relational database and a similarity relation which is based on expert rules is used to group similar alerts together. Two alerts are defined to be similar, for instance, if both occur within a fixed time window and their source and target match exactly. As already mentioned, these approaches are likely to fail under real-life conditions with imperfect classifiers (i.e., low-level IDS) with false alerts or wrongly adjusted time windows. Another approach to alert correlation is presented in [10]. A weighted, attribute-wise similarity operator is used to decide whether to fuse two alerts or not. However, as already stated in [11] and [12], this approach suffers from the high number of parameters that need to be set. The similarity operator presented in [13] 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 [14], 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 [15], 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 perceptrons, 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, labeled training data need to be generated which could be quite difficult in case of various attack instances. The same or quite similar techniques as described so far are also applied in many other approaches to alert correlation, especially in the field of intrusion scenario detection. Prominent research in scenario detection is described in [8],[9], [10], for example. More details can be found in [8]. In [8], 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 described in [11] 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 thereasons for the existence of the majority of alerts, the socalled 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 [10] root causes are regarded as "generally persistent" that does not hold for attack instances that occur only within a limited time window. Furthermore, only root

*International Journal of Research in Advent Technology (IJRAT) Special Issue*
*E-ISSN: 2321-9637*
*National Conference "MOMENTUM-17", 14th & 15th February 2017*
*Available online at www.ijrat.org*

causes that are responsible for a majority of alerts are of interest and the attribute-oriented induction algorithm is forced "to find large clusters" as the alert load can thus be reduced at most. 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 [12] is based on [11] 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 [10].

A completely different clustering approach is presented in [09]. There, the reconstruction error of an autoassociator neural network (AA-NN) is used to distinguish different types of alerts. Alerts that yield the same (or a similar) reconstruction error are put into the same cluster. The approach can be applied online, but an offline training phase and training data are needed to train the AA-NN and also to manually adjust intervals for the reconstruction error that determine which alerts are clustered together. In addition, it turned out that due to the dimensionality reduction by the AA-NN, alerts of different types can have the same reconstruction error which leads to erroneous clustering. In our prior work, we applied the well-known c-means clustering algorithm in order to identify attack instances [23]. However, this algorithm also works in a purely offline manner.

## 3. A NOVEL ONLINE ALERT AGGREGATION TECHNIQUE

In this section, we describe our new alert aggregation approach which is—at each point in time—based on a probabilistic model of the current situation. To outline the preconditions and objectives of alert aggregation, we start with a short sketch of our intrusion framework. Then, we briefly describe the generation of alerts and the alert format. We continue with a new clustering algorithm for offline alert aggregation which is basically a parameter estimation technique for the probabilistic model. After that, we extend this offline method to an algorithm for data stream clustering which can be applied to online alert aggregation. Finally, we make some remarks on the generation of meta-alerts.

### 3.1. *Collaborating Intrusion Detection Agents*

In our work, we focus on a system of structurally very similar so-called intrusion detection (ID) agents. Through self-organized collaboration, these ID agents form adistributed intrusion detection system (DIDS). Fig. 1 outlines the layered architecture of an ID agent: The sensor layer provides the interface to the network and the host on which the agent resides. Sensors acquire raw data from both the network and the host, filter incoming data, and extract interesting and

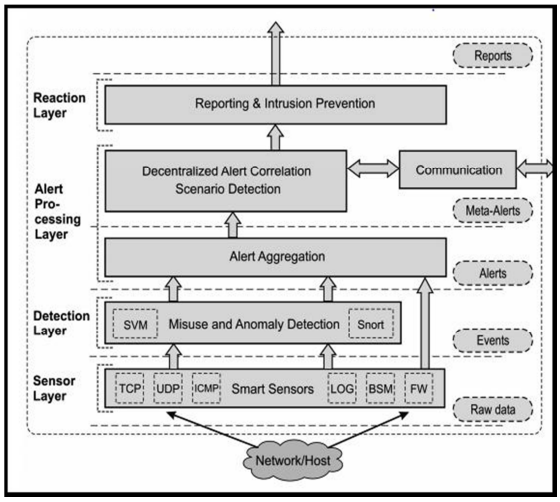


Fig.1. Architecture of an intrusion detection agent.

potentially valuable (e.g.,statistical) information which is needed to construct an appropriate event. At the detection layer, different detectors, e.g., classifiers trained with machine learning techniques such as support vector machines (SVM) or conventional rule-based systems such as Snort, assess these events and search for known attack signatures (misuse detection) and suspicious behavior (anomaly detection). In case of attack suspicion, they create alerts which are then forwarded to the alert processing layer. Alerts may also be produced by FW or the like. At the alert processing layer, the alert aggregation module has to combine alerts that are assumed to belong to a specific attack instance. Thus, so called meta-alerts are generated. Meta-alerts are used orenhanced in various ways, e.g., scenario detection or decentralized alert correlation. An important task of

*International Journal of Research in Advent Technology (IJRAT) Special Issue*
*E-ISSN: 2321-9637*
*National Conference "MOMENTUM-17", 14th & 15th February 2017*
*Available online at www.ijrat.org*

the reaction layer is reporting. The overall architecture of the distributed intrusion detection system and a framework for large-scale simulations are described in in more detail.

In our layered ID agent architecture, each layer assesses, filters, and/or aggregates information produced by a lower layer. Thus, relevant information gets more and more condensed and certain, and, therefore, also more valuable. We aim at realizing each layer in a way such that the recall of the applied techniques is very high, possibly at the cost of a slightly poorer precision [12]. In other words, with the alert aggregation module on which we focus in this paper we want to have a minimal number of missing meta-alerts (false negatives) and we accept some false metaalerts (false positives) and redundant meta-alerts in turn.

### 3.2. *Alert Generation and Format*

In this section, we make some comments on the information contained in alerts, the objects that must be aggregated, and on their format. As the concrete content and format depend on a specific task and on certain realizations of the sensors and detectors, together with the experimental conditions. At the sensor layer, sensors determine the values of attributes that are used as input for the detectors as well as for the alert clustering module. Attributes in an event that are independent of a particular attack instance can be used for classification at the detection layer. Attributes that are

(or might be) dependent on the attack instance can be used in an alert aggregation process to distinguish different attack instances. A perfect partition into dependent and independent attributes, however, cannot be made. Some are clearly dependent (such as the source IP address which can identify the attacker), some are clearly independent such as the destination port which usually is 80 in case of webbased attacks), and lots are both (such as the destination port which can be a hint to the attacker's actual target service as well as an attack tool specifically designed to target a particular service only). In addition to the attributes produced by the sensors, alert aggregation is based on additional attributes generated by the detectors. Examples are the estimated type of the attack instance that led to the generation of the alert (e.g., SQL injection, buffer overflow, or denial of service), and the degree of uncertainty associated with that estimate.

### 3.3. *Offline Alert Aggregation*

In this section, we introduce an offline algorithm for alert aggregation which will be extended to a data stream algorithm for online aggregation .Assume that a host with an ID agent is exposed to a certain intrusion situation as sketched in algorithm shown in Fig. 2: One or several attackers launch several attack instances belonging to various attack types. The attack instances each cause a number of alerts with various attribute values. Only two of the attributes are shown and the correspondence of alerts and (true or estimated) attack instances is indicated by different symbols. Fig.2. shows a view on the "ideal world" which an ID agent does not have. The agent only has observations of the detectors (alerts) in the attribute space without attack instance labels as outlined in Fig. 2. The task of the alert aggregation module is now to estimate the assignment to instances by using the unlabeled observations only and by analyzing the cluster structure in the attribute space. That is, it has to reconstruct the attack situation. Then, meta-alerts can be generated that are basically an abstract description of the cluster of alerts assumed to originate from one attack instance. Thus, the amount of data is reduced substantially without losing important information. Fig. 2 shows the result of a reconstruction of the situation. There may be different potentially problematic situations:

1. **False alerts are not recognized** as such andwrongly assigned to clusters: This situation is acceptable as long as the number of false alerts is comparably low.
2. **True alerts are wrongly assigned to clusters:** Thissituation is not really problematic as long as the majority of alerts belonging to that cluster is correctly assigned. Then, no attack instance is missed.
3. **Clusters are wrongly split:** This situation isundesired but clearly unproblematic as it leads to redundant meta-alerts only. Only the data reduction rate is lower, no attack instance is missed.
4. **Several clusters are wrongly combined into one:** This situation is definitely problematic as attack instances may be missed.

According to our objectives (cf. Section 3.1) we must try to avoid the latter situation but we may accept the former three situations to a certain degree. How can the set of samples be clustered (i.e.,

*International Journal of Research in Advent Technology (IJRAT) Special Issue*
*E-ISSN: 2321-9637*
*National Conference "MOMENTUM-17", 14th & 15th February 2017*
*Available online at www.ijrat.org*

aggregated) to generate meta-alerts? Here, the answer to this question is identical to the answer to the following: How can an attack situation be modeled with a parameterized probabilistic model and how can the parameters be estimated from the observations?

### 3.4. *Data Stream Alert Aggregation*

In this section, we describe to an online approach working for dynamic attack situations. Assume that in the environment observed by an ID agent attackers initiate new attack instances that cause alerts for a certain time interval until this attack instance is completed. Thus, at any point in time the ID agent which is assumed to have a model of the current situation shown in Fig. 2 has several tasks, Fig. 3:

1. Component adaption: Alerts associated with already recognized attack instances must be which has several tasks. Identified as such and assigned to already existing clusters while adapting the respective component parameters.
2. Component creation (novelty detection): The occurrence of new attack instances must be stated. New components must be parameterized accordingly.
3. Component deletion (obsoleteness detection): The completion of attack instances must be detected and the respective components must be deleted from the model.

A Public Cloud Server is an element of the infrastructure provided by a cloud service supplier, like Amazon S31, for storing and rendering of volumes. It stores (encrypted) volumes and access policies accustomed regulate access to the degree and the rendered image. It performs most of the rendering on keep volumes and produces the partially rendered information.

### 3.5. *Meta-Alert Generation and Format*

With the creation of a new component, an appropriate met alert that represents the information about the component in an abstract way is created. Every time a new alert is added to a component, the corresponding meta-alert is updated incrementally, too. That is, the meta-alert "evolves" with the component. Meta-alerts may be the basis for a whole set further tasks.

Meta-alerts could be used at various points in time from the initial creation until the deletion of the corresponding component (or even later). For instance, reports could be created immediately after the creation of the component or which could be more preferable in some cases a sequence of updated reports could be created in regular time intervals. Another example is the exchange of met alerts between ID agents: Due to high communication costs,meta-alerts could be exchanged based on the evaluation of their interestingness.

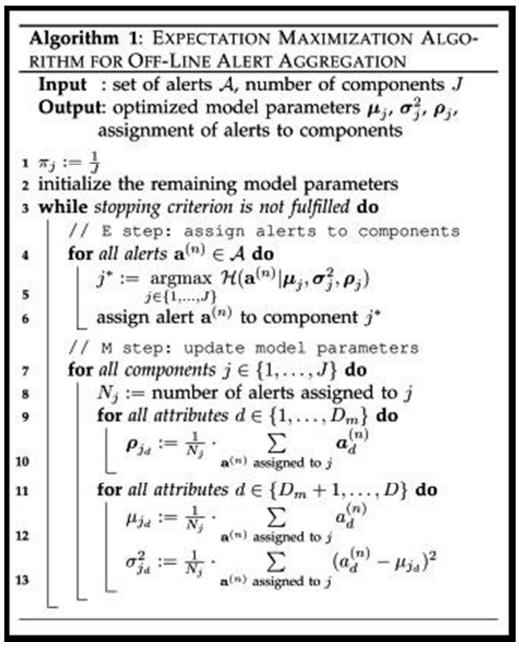


Fig.2. Algorithm for expectation Maximization.

## CONCLUSION

The experiments demonstrated the broad applicability of the proposed online alert aggregation approach. We analyzed three different data sets and showed that machine-learning-based detectors, conventional signature based detectors, and even firewalls can be used as alert generators. In all cases, the amount of data could be reduced substantially. Although there are situations as described in Section 3.3—especially clusters that are wrongly split—the instance detection rate is very high: None or only very few attack instances were missed. Runtime and component creation delay are well suited for an online application.

## 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.

*International Journal of Research in Advent Technology (IJRAT) Special Issue*
*E-ISSN: 2321-9637*
*National Conference "MOMENTUM-17", 14th & 15th February 2017*
*Available online at www.ijrat.org*

[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.ComputerSecurityApplications 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 alse 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 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.