Online Intrusion Alert Aggregation with Generative Data Stream Modeling

M.Hanock[1], K.Srinivas[2], A.Yaganteeswarudu[3]

[12]Kottam College of engineering Kurnool, india
[3]SJCET Kurnool, india

Abstract
Alert aggregation is an important subtask of intrusion detection. The goal is to identify and to cluster different alerts—produced by low-level intrusion detection systems, firewalls, etc.—belonging to a specific attack instance which has been initiated by an attacker at a certain point in time. Thus, meta-alerts can be generated for the clusters that contain all the relevant information whereas the amount of data (i.e., alerts) can be reduced. substantially. Meta-alerts may then be the basis for reporting to security experts or for communication within a distributed intrusion detection system. We propose a novel technique for online alert aggregation which is based on a dynamic, probabilistic model of the current attack situation. Basically, it can be regarded as a data stream 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 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. 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 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 meta alerts, 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 using probabilistic methods. Assuming that attack instances can be regarded as random processes "producing" alerts, we aim at modeling these processes using approximative 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.

Manuscript received September 5, 2013
Manuscript revised September 20, 2013

2.RELATED WORK

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 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. 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).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] to cluster alerts. 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. 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. 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, 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. 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..

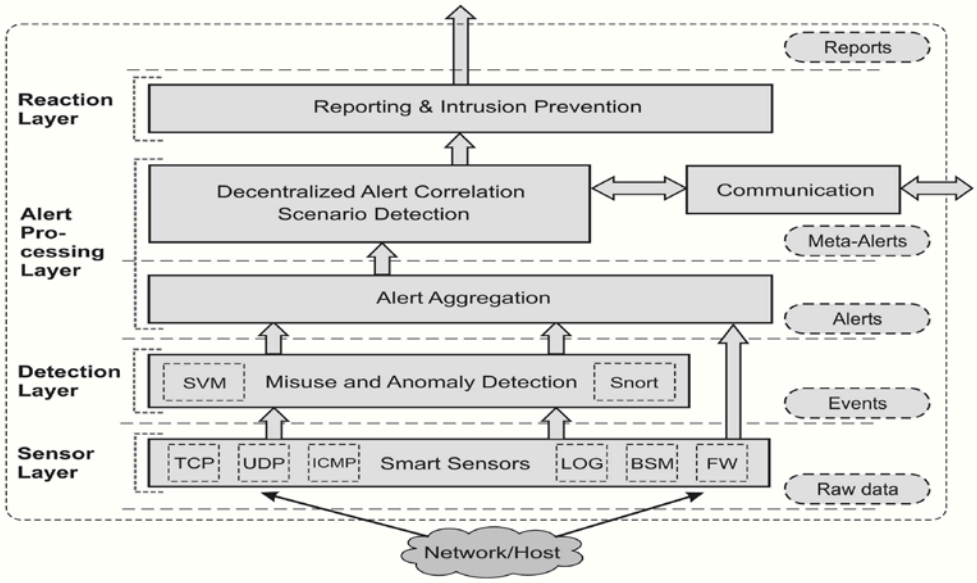


Fig. 1. Architecture of an intrusion detection agent.

3. ANOVEL ONLINE ALERT AGGREGATION TECHNIQUE

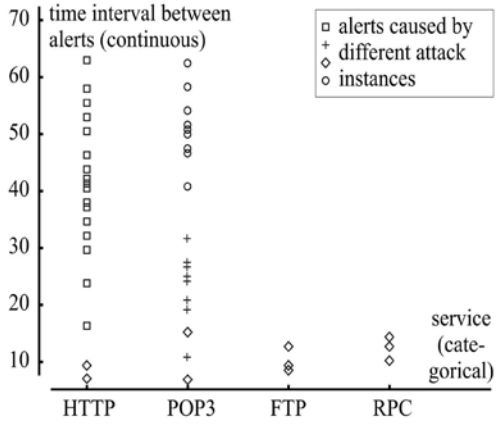
In this section, we describe our new alert aggregation approach which is—at each point in time—based on a probabilistic model of the current situation. To outline the preconditions and objectives of alert aggregation, we start with a short sketch of our intrusion framework. Then, we briefly describe 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 ake some remarks on the generation of meta-alerts.

3.1 Collaborating Intrusion Detection Agents

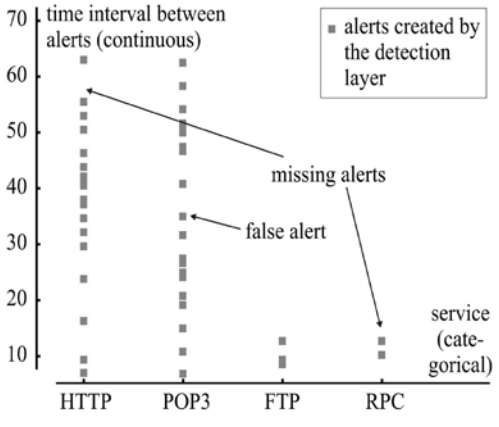
In our work, we focus on a system of structurally very similar so-called intrusion detection (ID) agents. Through self-organized collaboration, these ID agents form a distributed intrusion detection system (DIDS). Fg. 1 outlines the layered architecture of an ID agent: The sensor layer provides the interface 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 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, socalled meta-alerts are generated. Meta-alerts are used or enhanced in various ways, e.g., scenario detection or decentralized alert correlation. An important task of the reaction layer is reporting. The overall architecture of the distributed intrusion detection system and a framework for large-scale simulations are described 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 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, some more details will be given in Section 4 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.



(a)



(b)

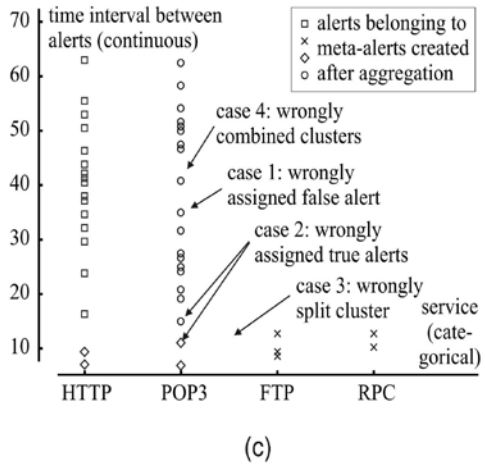


Fig. 2. Example illustrating the alert aggregation task and possible problems (artificial attack situation). (a) Idealized world: In the idealized IDS, the detectors do not make errors (no false and missing alerts) and the correct assignment of alerts to attack instances is known (indicated by different symbols). (b) Actual observations: The alerts produced by a real detection layer. The task of the alert aggregation is to reconstruct the attack situation by means of these observations only (including false alerts). (c) Reconstruction: The result of the aggregation (correspondence of alerts and clusters/meta-alerts) together with four different types of problems that may arise.

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 in Section 3.4. Assume that a host with an ID agent is exposed to acertain intrusion situation as sketched 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. 2a 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. 2b. 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. 2c shows the result of a reconstruction of the situation. There may be different potentially problematic situations:

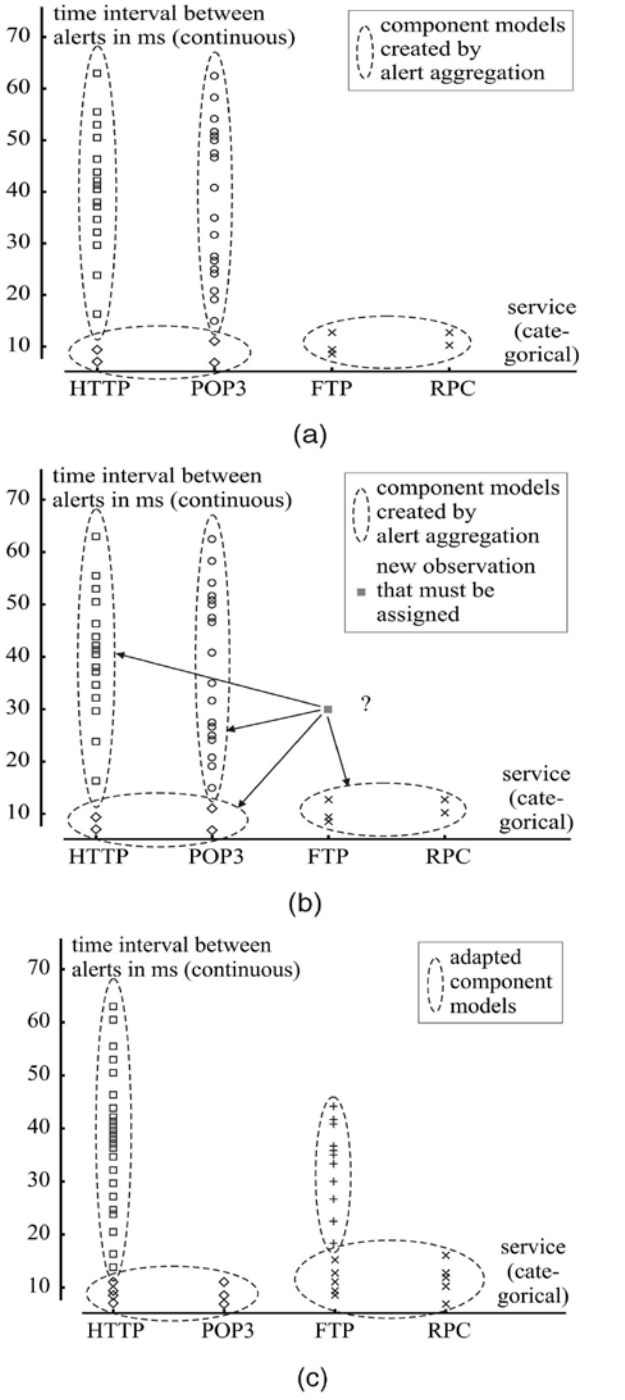


Fig. 3. Example illustrating the principle of online alert aggregation (artificial attack situation). (a) Existing model: Components have been created by the alert aggregation module. These components are the basis for meta-alert generation. (b) Assignment problem: New observations must either be assigned to an existing component which is then adapted or a new component must be created. Also, outdated components must be deleted. (c) Adapted model: The new situation after a few steps. One component has been created, one component has been deleted, and the other components have been adapted accordingly.

1. False alerts are not recognized as such and wrongly 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: This situation 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 is undesired 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

3.4 Data Stream Alert Aggregation

In this section, we describe how the offline approach is extended 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 acertain 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, cf. Fig. 3a—has several tasks, cf. Fig. 3b:
1. Component adaption: Alerts associated with already recognized attack instances must be 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. HOFMANN AND SICK: ONLINE INTRUSION ALERT AGGREGATION WITH GENERATIVE DATA STREAM MODELING 287 Fig. 3. Example illustrating the principle of online alert aggregation (artificial attack situation). (a) Existing model: Components have been created by the alert aggregation module. These components are the basis for meta-alert generation. (b) Assignment problem: New observations must either be assigned to an existing component which is then adapted or a new component must be created. Also, outdated components must be deleted. (c) Adapted model: The new situation after a few steps. One component has been created, one component has been deleted, and the other components have been adapted accordingly.
3. Component deletion (obsoleteness detection): The completion of attack instances must be detected and the respective components must be deleted from the model. That is, the ID agent must be situation-aware and try to keep his model of the current attack situation permanently up to date, see Fig. 3c. Clearly, there is a trade-off between runtime (or reaction time) and accuracy.

4. EXPERIMENTAL RESULTS

This section evaluates the new alert aggregation approach.We use three different data sets to demonstrate the HOFMANN AND SICK: ONLINE INTRUSION ALERT AGGREGATION WITH GENERATIVE DATA STREAM MODELING 289 feasibility of the proposed method: The first is the wellknown DARPA intrusion detection evaluation data set for the second we used real-life network traffic data collected at our university campus network, and the third contains firewall log messages from a commercial Internet service provider. All experiments were conducted on an PC with 2.20 GHz and 2 GB of RAM.

4.1 Description of the Benchmark D

4.1.1 DARPA Data

For the DARPA evaluation several weeks of training and test data have been generated on a test bed that emulates a small government site. The network architecture as well as the generated network traffic has been designed to be similar to that of an Air Force base. We used the TCP/IP network dump as input data and analyzed all 104 TCP-based attack instances (corresponding to more than 20 attack types) that have been launched against the various target hosts. As sketched in Section 3.2, sensors extract statistical information from the network traffic data. At the detection layer, we apply SVM to classify the sensor events. By applying a varying threshold to the output of the classifier, a so-called receiver operating characteristics (ROC) curve can be created The ROC curve in Fig. 4 plots the true positive rate (TPR, number of true positives divided by the sum of true positives and false negatives) against the false positive rate (FPR, number of false positives divided by the sum of false positives and true negatives) for the trained SVM. Each point of the curve corresponds to a specific threshold. Four operating points (OP) are marked. OP 1 is the one with the smallest overall error, but as we want to realize a high recall, we also investigate three more operating points which exhibit higher TPR at the cost of an increased FPR. We will also investigate the aggregation under idealized conditions where we assume to have a perfect detector layer with no missing and no false alerts at all. As attributes for the alerts, we use the source and destination IP address, the source and destination port, the attack type, and the creation time differences (based on the creation time stamps). Table 1 shows the number of alerts produced for the different OP and also for the idealized condition, i.e., a perfect detection layer. In addition, the number of attack instances for which at least one alert is generated by the detector layer is also given. Note that we have 104 attack instances in the data set altogether. For OP 1, there are three attack instances for which not even a single alert is created i.e., these instances

are already missed at the detection layer. We are aware of the various critique on the DARPA benchmark data and the limitations that emerge thereof.

4.1.2 Campus Network Data

To assess the performance of our approach in more detail, we also conducted own attack experiments. We launched several brute force password guessing attacks against the mail server (POP3) of our campus network and recorded the network traffic. The attack instances differed in origin, start time, duration, and password guessing rate. The attack schedule was designed to reflect situations which we regard as being difficult to recognize. In particular, we have

1. Several concurrent attack instances (up to seven),
2. Partially and completely overlapping attack instances,
3. Several instances within a short time interval,
4. Different attack instances from similar sources,
5. Different attack durations, and
6. An attacker that changes his IP address during the attack.

In order to demonstrate that the proposed technique can also be used with a conventional signature-based detector, the captured traffic was analyzed by the open source IDS Snort which detected all 17 attack instances that have been launched and produced 128,816 alerts (cf. Table 1). The alert format equals the one used for the SVM detector, i.e., the alerts exhibit the source and destination IP address, TABLE 1 Input of the Alert Aggregation Algorithm the source and destination port, the attack type, and creation time differences. Snort was configured to match our network topology and we turned off rudimental alert aggregation features. In order to achieve a high recall, we activated all available rule sets—the official rule sets as well as available community rules, which both are available at the Snort web page Activating all rules leads to a false alert rate of 0.33 percent. The FPR is based on the assumption that all alerts that are not classified with the attack type that we launched are false alerts. There is no missing alert rate given in Table 1 for this data set for two reasons: First, it cannot be guaranteed that there are unknown attacks in the data set that were started by real attackers and second, we do not know exactly how many alerts should be created by the attacks we launched. 4.1.3 Internet Service Provider Firewall Logs The third data set used here differs from the previous ones as we actually do not have a detector layer that performs a classification and searches for known attacks. Here, the alerts consist of the source and destination IP address, the source and destination port, the creation time differences, and the PIX message type.

4.2 Performance Measures

In order to assess the performance of the alert aggregation, we evaluate the following measures: Percentage of detected instances (p). We regard an attack instance as being detected if there is at least one meta alert that predominantly contains alerts of that particular instance. The percentage of detected attack instances p can thus be determined by dividing the number of instances that are detected by the total number of instances in the data set. The measure is computed with respect to the instances covered by the output of the detection layer, i.e., instances missed by the detectors are not considered. Number of meta-alerts (MA) and reduction rate (r). The number of meta-alerts (MA) is further divided into the number of attack meta-alerts MAattack which predominantly contain true alerts and the number of nonattack meta-alerts MAnonattack which predominantly contain false alerts. The reduction rate r is 1 minus the number of created metaalerts MA divided by the total number of alerts N.Average runtime (tavg) and worst case runtime (tworst).

4.3 Results

In the following, the results for the alert aggregation are presented. For all experiments, the same parameter settings are used. We set the threshold _ that decides whether to add a new alert to an existing component or not to five percent, and the value for the threshold _ that specifies the allowed temporal spread of the alert buffer to 180 seconds. _ was set that low value in order to ensure that even a quite small degrade of the cluster quality, which could indicate a new attack instance, results in a new component. A small value of _, of course, results in more components and, thus, in a lower reduction rate, but it also reduces the risk of missing attack instances. The parameter _, which is used in the novelty assessment function, controls the maximum time that new alerts are allowed to reside in the buffer B. In order to keep the response time short, we set it to 180 s which we think is a reasonable value. For both parameters, there were large intervals in which parameter values could be chosen without deteriorating the results.

5.SUMMARY AND OUTLOOK

We presented a novel technique for online alert aggregation and generation of meta-alerts. We have shown that the sheer amount of data that must be reported to a human security expert or communicated within a distributed intrusion detection system, for instance, can be reduced significantly. The reduction rate with respect to the number of alerts was up to 99.96 percent in our experiments. At the same time, the number of missing attack instances is extremely low or even zero in some of our experiments and the delay for the detection of attack instances is within the range of some seconds only. In the future, we will develop techniques for interestingness- based communication strategies for a distributed IDS.

ACKNOWLEDGMENTS

We express our sincere thanks to Kottam College of engineering providing us good lab facilities. A heart full and sincere gratitude to our beloved parents and friend for their tremendous motivation and moral support.

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.

A.YAGANTEESWARUDU received the B.TECH from St.Johns college of engg&tech 2009.After Worked as assistant professor in SJCET, Now studying M.Tech in St.Johns college of engg&tech

M.Hanock completed B.Tech in St.Johns college of engineering&tech.Now studying M.Tech in Kottam college of engineering

K.Sreenivas, completed M.Tech, persuiving Ph.D,Working as HOD,in Kottam college of engineering