ISSN:0975-9646

Online Intrusion Alert Aggregation Using DGDSM Approach

J.Poongodi[1], Mrs.C.Vimalarani[2], Dr.S.Karthik[3]

CSE, SNS College of Technology,
Coimbatore,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.

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 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** 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.

2 A NOVEL ONLINE ALERT AGGREGATION TECHNIQUES

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.

2.1 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).

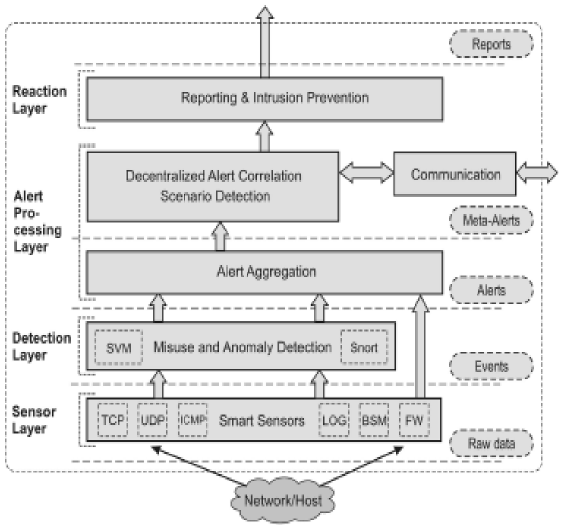

Fig. 1. Architecture of an intrusion detection agent.

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 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.

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 . 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 meta alerts (false positives) and redundant meta-alerts in turn.

2.2 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 next Section.

Assume that a host with an ID agent is exposed to a certain 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:

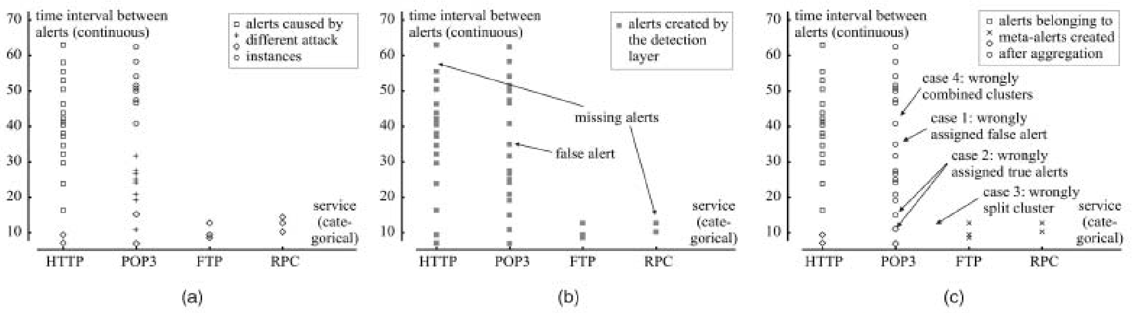

Fig. 2. Example illustrating the alert aggregation task and possible problems (artificial attack situation). (a) Idealized world: In the idealized IDS, the detectors do not make errors (no false and missing alerts) and the correct assignment of alerts to attack instances is known (indicated by different 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.

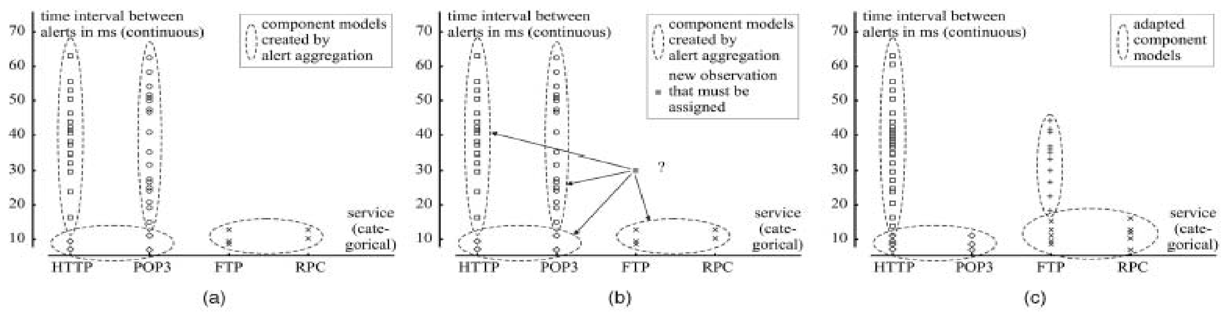

Fig. 3. Example illustrating the principle of online alert aggregation (artificial attack situation). (a) Existing model: Components have been created by the alert aggregation module. These components are the basis for meta-alert generation. (b) Assignment problem: New observations must either be assigned 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.

2.3 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 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, 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.
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.

2.4 Meta-Alert Generation and Format

With the creation of a new component, an appropriate metaalert 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 (cf. Fig. 1): .

- Sequences of meta-alerts may be investigated further in order to detect more complex attack scenarios (e.g., by means of hidden Markov models).
- Meta-alerts may be exchanged with other ID agents in order to detect distributed attacks such as one-to many attacks.
- Based on the information stored in the meta-alerts, reports may be generated to inform a human security expert about the ongoing attack situation.

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 metaalerts between ID agents: Due to high communication costs, meta-alerts could be exchanged based on the evaluation of their interestingness. According to the task for which meta-alerts are used, they may contain different attributes. Examples for those attributes are aggregated alert attributes (e.g., lists or intervals of source addresses or targeted service ports, or a time interval that marks the beginning and the end—if available—of the attack instance), attributes extracted from the probabilistic model (e.g., the distribution parameters or the number of alerts assigned to the component), an aggregated alert assessment provided by the detection layer (e.g., the attack type classification or the classification confidence), and also information about the current attack situation (e.g., the number of recent attacks of the same or a similar type, links to attacks originating from the same or a similar source).

3. CONCLUSION

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 2.2—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.
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.