Online Intrusion Detection System Modeling

Md Nurul Hasan

1MTech, Department of Computer Science, Bharat University, Khujuti Para, West Bengal, India

Abstract

An intrusion detection system (IDS) is a security layer used to detect ongoing intrusive activities in information systems. Traditionally, intrusion detection relies on extensive knowledge of security experts, in particular, on their familiarity with the computer system to be protected. 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. 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.

Keywords—Attack detection model, Intrusion detection, alert aggregation, generative modeling, data stream algorithm

I. Introduction

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, “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 met 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 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.

www.ijreat.org

Published by: PIONEER RESEARCH & DEVELOPMENT GROUP (www.prdg.org)
II. 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, 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. 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, 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. In, 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. A weighted, attribute-wise similarity operator is used to decide whether to fuse two alerts or not. However, as already stated in and, 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 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, for example. More details can be found in. In, an online 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 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. Furthermore, only root 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 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. A completely different clustering approach is presented in . 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.

III. Proposed System

Alerts may also be reproduced 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 in enhanced 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 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. 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. However, this algorithm also works in a purely offline manner.

Web search engines provide an efficient interface to this vast information. Page counts and snippets are two useful information sources provided by most web search engines. Page count of a query is an estimate of the number of pages that contain the query words.

IV. 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.

Sensor Networks

The extensive number of research work in this area has appeared in the literature. Due to the limited energy budget available at sensor nodes, the primary issue is how to develop energy-efficient techniques to reduce communication and energy costs in the networks. Approximate-based data aggregation techniques have also been proposed. The idea is to tradeoff some data quality for improved energy efficiency. Silberstein et al. develop a sampling-based
approach to evaluate approximate top-k queries in wireless sensor networks. Based on statistical modeling techniques, a model-driven approach was proposed in [5] to balance the confidence of the query answer against the communication cost in the network. Moreover, continuous top-k queries for sensor networks have been studied in [28] and [29]. In addition, a distributed threshold join algorithm has been developed for top-k queries. These studies, considering no uncertain data, have a different focus from our study.

### V. Offline Alert Aggregation

In this section, we introduce an offline algorithm for alertAggregation which will be extended to a data stream algorithm for online aggregation in [30]. Assume that a host with an ID agent is exposed to a certain intrusion situation as sketched in: One or several attackers launch several attack instances belonging to various attack types. The attack instances each cause anumber 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. 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 attributespace without attack instance labels as outlined in [31]. The task of the alert aggregation module is now to estimate the assignment to instances by using the labeled 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. shows the result of the reconstruction of the situation. There may be different potentially problematic situations:

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 incorrectly 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 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., 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? First, we introduce a model for a single attack instance. We regard an attack instance as a random process generating alerts that are distributed according to a certain multivariate probability distribution. The alert space is composed of several attributes.

### VI. Associated Work

In recent years, many works have been done to process in wireless sensor networks, and processing on uncertain data. Top-k query processing in sensor networks. An extensive number of research works in this area has appeared in the literature [21], [24], [25], [26]). Due to the limited energy budget available at sensor nodes, the primary issue is how to develop energy-efficient techniques to reduce communication and energy costs in the networks.

TAG [21] is one of the first studies in this area. By exploring the semantics of aggregate operators (e.g., sum, avg, and top-k), in-network processing approach is adopted to suppress redundant data transmissions in wireless sensor networks. Approximate-based data aggregation techniques have also been proposed [27], [25]. The idea is to trade off some data quality for improved energy efficiency. Silberstein et al. develop a sampling-based approach to evaluate approximate on statistical modeling techniques, a model-driven approach was proposed in [5] to balance the confidence of the query answer against the communication cost in the network. Moreover, continuous top-k queries for sensor networks have been studied in [28] and [29]. In addition, a distributed threshold join algorithm has been developed for top-k queries [24]. These studies, considering no uncertain data, have a different focus.
from our study. Top-k query processing on uncertain data. While research works on conventional top-k queries are mostly based on some deterministic scoring functions, the new factor of tuple membership probability in uncertain databases makes evaluation of probabilistic top-k queries very complicated since the top-k answer set depends not only on the ranking scores of candidate tuples but also their probabilities [8]. For uncertain databases, two interesting top-k definitions (i.e., U-Topk and U-kRanks) and A_-like algorithms are proposed [17]. U-Topk returns a list of k tuples that has the highest probability to be in the top-k list over all possible worlds. U-kRanks returns a list of k tuples such that the ith record has the highest probability to be the ith best record in all possible worlds. In [13], PT-Topk query, which returns the set of tuples with a probability of at least p to be in the top-k lists in the possible worlds, is studied. Inspired by the concept of dominate set in the top-k query, an algorithm which avoids unfolding all possible worlds is given. Besides, a sampling method is developed to quickly compute an approximation with quality guarantee to the answer set by drawing a small sample of the uncertain data. In [19], the expected rank of each tuple across all possible worlds serves as the ranking function for finding the final answer. In [30], U-Topk and U-kRank queries are improved by exploiting their stop conditions. In [31], all existing top-k semantics have been unified by using generating functions. Recently, a study on processing top-k queries over a distributed uncertain database is reported in [14] and [23]. Li et al. [14] only support top-k queries with the expected ranking semantic. On the contrary, our proposal is a general approach which is applicable to probabilistic top-k queries with any semantic. Furthermore, instead of repeatedly requesting data which may last for several rounds, our protocols are guaranteed to be completed within no more than two rounds. These differences uniquely differentiate our effort from [14]. Our previous work [23] as the initial attempt only includes the concept of sufficient set. In this paper, besides of sufficient set, we propose another important concept of necessary set. With the aid of these two concepts, we further develop a suite of algorithms, which show much better performance than the one in [23]. Probabilistic ranked queries based on uncertainty at the attribute level are studied in [32], [33], and [19]. A unique study that ranks tuples by their probabilities satisfying the query is presented in [12]. Finally, uncertain top-k query is studied under the setting of streaming databases where a compact data set is exploited to support efficient slide window top-k queries [18]. We will apply sufficient set and necessary set to sensor networks with tree topology, to further improve query processing performance by facilitating sophisticated in-network filtering at the intermediate nodes along the routing path to the root.

VII. Experimental Setup And Result

This section evaluates the new alert aggregation approach. We use three different data sets to demonstrate the Feasibility of the proposed method: The first is the well-known DARPA intrusion detection evaluation data set [32], 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.

Description of the Benchmark Data Sets

DARPA Data

For the DARPA evaluation [32], 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 networkdump 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 [38]. 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. Eachpoint 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
omissioning 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,
the 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 dataset 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. The three instances are missed because there are
only a few training samples of the corresponding
attack types in the dataset which results in a bad
generalization performance of the SVM. Switching
from OP 1 to OP 2 alerts are created for one more
instance. OP 3 and OP 4 do not yield any further
improvement. We are aware of the various critique on
the DARPA benchmarks (e.g., published in [39],
[40]) and the limitations that emerge thereof. Despite
all disadvantages, this data set is still frequently
used—not least because it is well-documented and
publicly available data set that allows the comparison
of results of different researchers. In order to achieve
fair and realistic results, we carefully analyzed
all known deficiencies and omitted features that could
bias the detector classification performance. Besides,
most of the deficiencies of this data set have an
impact on the detector layer but only little influence
on the alert aggregation results. A detailed discussion
can be found in [19] and [41].

<table>
<thead>
<tr>
<th>OP</th>
<th>FPR</th>
<th>TPR</th>
<th># Alerts</th>
<th># Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0%</td>
<td>0.0%</td>
<td>1562/604</td>
<td>104</td>
</tr>
<tr>
<td>OP 1</td>
<td>0.3%</td>
<td>99.22%</td>
<td>1563/995</td>
<td>101</td>
</tr>
<tr>
<td>OP 2</td>
<td>1.0%</td>
<td>99.66%</td>
<td>1565/844</td>
<td>102</td>
</tr>
<tr>
<td>OP 3</td>
<td>2.0%</td>
<td>99.78%</td>
<td>1575/885</td>
<td>102</td>
</tr>
<tr>
<td>OP 4</td>
<td>4.0%</td>
<td>99.89%</td>
<td>1576/610</td>
<td>102</td>
</tr>
</tbody>
</table>

Table 2.

VIII. 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 to a 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. 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 intervals
in which parameter values could be changed without
deteriorating the results. A detailed analysis
and discussion on the effects of different parameter
settings can be found in [19].

DARPA Data

Results for the DARPA data set are given in
Table 2. First of all, it must be stated there is an
operation point of the SVM at the detection layer (OP
1) where we do not miss any attack instances at all (at
least in addition to those already missed at the
detection layer). The reduction rate is with 99.87
percent extremely high, and the detection delay
is only 5.41 s in the worst case (d100%). Average and
worst caseruntimes are very good, too. All OP will now be analyzed in much more detail.

All attack instances for which the detector produces at least a single alert are detected in the idealized case and with OP 1 and OP 2. Choosing another OP, the rate of detected instances drops to 98.04 percent (OP 3) and 99.02 percent (OP 4). In OP 3, a FORMAT instance and a MULTIHOP instance are missed. In OP 4, only the FORMAT instance could not be detected. A further analysis identified the following reasons: The main reason in the case of the FORMAT instance is the small number of only four alerts. Those alerts are created by the detector layer for all OP, i.e., there is obviously no benefit from choosing another OP. All OP are now analyzed in much more detail.

Results of the Online Alert Aggregation for Three Benchmark Data Sets

<table>
<thead>
<tr>
<th>OP</th>
<th>p [h]</th>
<th>MA</th>
<th>M_A_data</th>
<th>M_A_struct</th>
<th>p [%]</th>
<th>l_mean [ms]</th>
<th>d_mean [s]</th>
<th>d_median [s]</th>
<th>d_std [s]</th>
<th>d_max [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.00</td>
<td>324</td>
<td>324</td>
<td>0</td>
<td>99.96</td>
<td>0.3</td>
<td>1.5</td>
<td>3.9</td>
<td>4.1</td>
<td>6.2</td>
</tr>
<tr>
<td>2</td>
<td>10.01</td>
<td>316</td>
<td>316</td>
<td>0</td>
<td>99.97</td>
<td>0.1</td>
<td>1.5</td>
<td>3.9</td>
<td>4.1</td>
<td>6.2</td>
</tr>
<tr>
<td>3</td>
<td>98.94</td>
<td>399</td>
<td>399</td>
<td>0</td>
<td>99.98</td>
<td>0.0</td>
<td>0.3</td>
<td>1.5</td>
<td>2.1</td>
<td>7.0</td>
</tr>
<tr>
<td>4</td>
<td>98.96</td>
<td>399</td>
<td>399</td>
<td>0</td>
<td>99.98</td>
<td>0.0</td>
<td>0.3</td>
<td>1.5</td>
<td>2.1</td>
<td>7.0</td>
</tr>
</tbody>
</table>

Table 3.

IX. 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. 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. This IDS will be based on organic computing principles [44]. In addition, we will investigate how human domain knowledge can be used to improve the detection processes further. We will also apply our techniques to benchmark data that fuse information from heterogeneous sources (e.g., combining host and network-based detection).

References


