IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor : 1.479)
www.ijreat.org

Online Intrusion Detection System Modeling

Md Nurul Hasan[1]

[1]MTech, 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 differentalerts—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, Authenticationmechanisms or encryption techniques veryImportant 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 Detectingattack types, but not on distinguishing between differentattack 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.

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 Futureresearch , 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 approachesare 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 userdefined 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 begenerated 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 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 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 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 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 beproduced by FW or the like. At the alert processing layer,the alert aggregation module has to combine alerts that areassumed to belong to a specific attack instance. Thus, so-calledmeta-alerts are generated. Meta-alerts are used orenhanced in various ways, e.g., scenario detection ordecentralized alert correlation. An important task of thereaction layer is reporting.The overall architecture of the distributed intrusiondetection system and a framework for large-scale simulationsare 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 missingmeta-alerts (false negatives) and we accept some false metaalerts (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 offlinemanner.

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.

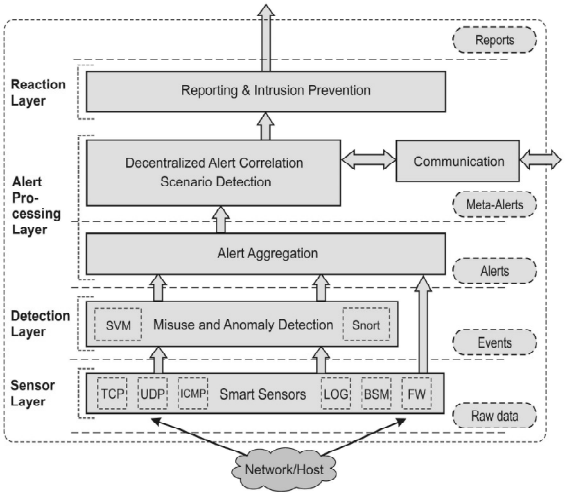

Fig. 1 Shows the Architecture diagram.

IV. A Novel Online Alert AggregationTechnique

In this section, we describe our new alert aggregationApproach which is—at each point in time—based on aprobabilistic model of the current situation. To outline thepreconditions and objectives of alert aggregation, we startwith a short sketch of our intrusion framework. Then, webriefly describe the generation of alerts and the alert format.We continue with a new clustering algorithm for offlinealert aggregation which is basically a parameter estimationtechnique for the probabilistic model. After that, we extendthis offline method to an algorithm for data stream clusteringwhich can be applied to online alert aggregation. Finally, wemake 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 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 and. 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 streamalgorithm for online aggregation in .Assume that a host with an ID agent is exposed to acertain intrusion situation as sketched in : One orseveral attackers launch several attack instances belongingto various attack types. The attack instances each cause anumber of alerts with various attribute values. Only two ofthe attributes are shown and the correspondence of alertsand (true or estimated) attack instances is indicated bydifferent symbols. shows a view on the "idealworld" which an ID agent does not have. The agent onlyhas observations of the detectors (alerts) in the attributespace without attack instance labels as outlined in .The task of the alert aggregation module is now to estimatethe assignment to instances by using the unlabeled observations only and by analyzing the cluster structurein the attribute space. That is, it has to reconstruct the attacksituation. Then, meta-alerts can be generated that arebasically an abstract description of the cluster of alertsassumed to originate from one attack instance. Thus, theamount of data is reduced substantially without losing important information. shows the result of areconstruction of the situation. There may be differentpotentially problematic situations:

1. False alerts are not recognized as such and wronglyassigned to clusters: This situation is acceptable aslong 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 themajority of alerts belonging to that cluster iscorrectly assigned. Then, no attack instance ismissed.

3. Clusters are wrongly split: This situation is undesiredbut clearly unproblematic as it leads toredundant meta-alerts only. Only the data reductionrate is lower, no attack instance is missed.

4. Several clusters are wrongly combined into one: This situation is definitely problematic as attackinstances may be missed.According to our objectives we must try toavoid the latter situation but we may accept the formerthree situations to a certain degree.How can the set of samples be clustered (i.e., aggregated)to generate meta-alerts? Here, the answer to thisquestion is identical to the answer to the following: Howcan an attack situation be modeled with a parameterizedprobabilistic model and how can the parameters beestimated from the observations?First, we introduce a model for a single attack instance.We regard an attack instance as a random process generatingalerts that are distributed according to a certain multivariateprobability distribution. The alert space is composedof several attributes.

VI. Associated Work

In recent years, many works have been done to Here; we review representative work in the areas of processing in wireless sensor networks, and 2 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 tradeoff 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

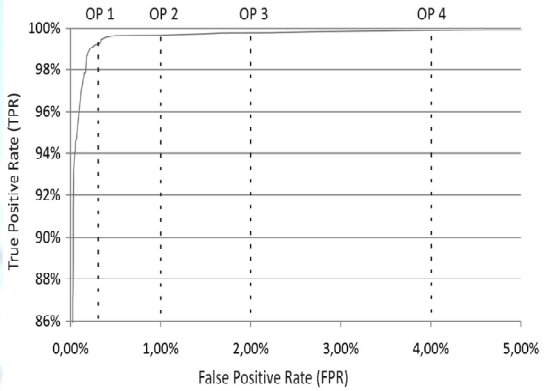

Table 1.ROC curve for the SVM detector

Feasibility of the proposed method: The first is the well-knownDARPA intrusion detection evaluation data set [32],for the second we used real-life network traffic datacollected at our university campus network, and the thirdcontains firewall log messages from a commercial Internetservice provider. All experiments were conducted on an PCwith 2.20 GHz and 2 GB of RAM.

Description of the Benchmark Data Sets

DARPA Data

For the DARPA evaluation [32], several weeks of trainingand test data have been generated on a test bed that emulatesa small government site. The network architecture as well asthe generated network traffic has been designed to be similarto that of an Air Force base. We used the TCP/IP networkdump as input data and analyzed all 104 TCP-based attackinstances (corresponding to more than 20 attack types) thathave been launched against the various target hosts. Assketched in Section 3.2, sensors extract statistical informationfrom the network traffic data. At the detection layer, weapply SVM to classify

the sensor events. By applying avarying threshold to the output of the classifier, a so-calledreceiver operating characteristics (ROC) curve can becreated [38]. The ROC curve in Fig. 4 plots the true positiverate (TPR, number of true positives divided by the sum oftrue positives and false negatives) against the false positiverate (FPR, number of false positives divided by the sum offalse positives and true negatives) for the trained SVM. Eachpoint of the curve corresponds to a specific threshold. Fouroperating points (OP) are marked. OP 1 is the one with thesmallest overall error, but as we want to realize a high recall,we also investigate three more operating points whichexhibit higher TPR at the cost of an increased FPR. We willalso investigate the aggregation under idealized conditionswhere we assume to have a perfect detector layer with nomissing and no false alerts at all. As attributes for the alerts,we use the source and destination IP address, the source anddestination port, the attack type, and the creation timedifferences (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., aperfect detection layer. In addition, the number of attackinstances for which at least one alert is generated by thedetector layer is also given. Note that we have 104 attackinstances in the data set altogether. For OP 1, there are threeattack 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 fewtraining samples of the corresponding attack types in thedata set which results in a bad generalization performanceof the SVM. Switching from OP 1 to OP 2 alerts are createdfor one more instance. OP 3 and OP 4 do not yield anyfurther improvement.We are aware of the various critique on the DARPAbenchmark data (e.g., published in [39], [40]) and thelimitations that emerge thereof. Despite all disadvantages,this data set is still frequently used—not least because it is awell-documented and publicly available data set that allowsthe comparison of results of different researchers. In orderto achieve fair and realistic results, we carefully analyzed allknown deficiencies and omitted features that could bias thedetector classification performance. Besides, most of thedeficiencies of this data set have an impact on the detectorlayer but only little influence on the alert aggregationresults. A detailed discussion can be found in [19] and [41].

Input of the Alert Aggregation Algorithm

OP	FPR	TPR	# Alerts	# Instances
DARPA Data				
Idealized	0.0%	0.0%	1 562 604	104
OP 1	0.3%	99.22%	1 563 995	101
OP 2	1.0%	99.66%	1 565 484	102
OP 3	2.0%	99.78%	1 575 955	102
OP 4	4.0%	99.89%	1 576 610	102
Campus Network Data				
n/a	0.33%	unknown	128 816	17
Internet Service Provider Firewall Logs				
n/a	n/a	n/a	4 898	n/a

Table 2.

VIII.Results

In the following, the results for the alert aggregation arepresented. For all experiments, the same parameter settingsare used. We set the threshold _ that decides whether to adda new alert to an existing component or not to five percent,and the value for the threshold _ that specifies the allowedtemporal spread of the alert buffer to 180 seconds. _ was setthat low value in order to ensure that even a quite smalldegrade of the cluster quality, which could indicate a newattack instance, results in a new component. A small valueof _, of course, results in more components and, thus, in alower reduction rate, but it also reduces the risk of missingattack instances. The parameter _, which is used in thenovelty assessment function, controls the maximum timethat new alerts are allowed to reside in the buffer B. In orderto keep the response time short, we set it to 180 s which wethink is a reasonable value. For both parameters, there werelarge intervals in which parameter values could be chosenwithout deteriorating the results. A detailed analysis anddiscussion on the effects of different parameter settings canbe found in [19].

DARPA Data

Results for the DARPA data set are given in Table 2. First ofall, it must be stated there is an operation point of the SVMat the detection layer (OP 1) where we do not miss anyattack instances at all (at least in addition to those alreadymissed at the detection layer). The reduction rate is with99.87 percent extremely high, and the detection delay isonly 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

Results of the Online Alert Aggregation for Three Benchmark Data Sets

OP	p [%]	MA	MA_{attack}	$MA_{non\text{-}attack}$	r [%]	t_{avg} [ms]	t_{worst} [s]	$d_{90\%}$ [s]	$d_{95\%}$ [s]	$d_{100\%}$ [s]
DARPA Data										
Idealized	100.00	324	324	0	99.98	0.13	8.54	1.79	3.17	96.2
OP 1	100.00	1976	368	1598	99.87	0.19	11.00	1.64	1.82	5.41
OP 2	100.00	3186	369	2817	99.80	0.28	11.83	1.18	1.95	7.92
OP 3	98.04	7946	339	7607	99.50	0.81	19.50	1.73	2.47	17.1
OP 4	99.02	8588	348	8240	99.46	0.97	16.05	1.94	2.47	16.2
Campus Network Data										
n/a	100.00	52	20	32	99.96	0.75	2.87	1.35	1.35	1.61
Internet Service Provider Firewall Logs										
n/a	n/a	56	n/a	n/a	98.86	1.53	0.27	n/a	n/a	n/a

Table 3.

IX. Conclusion

The experiments demonstrated the broad applicability ofthe proposed online alert aggregation approach. We analyzed three different data sets and showed thatMachine-learning-based detectors, conventional signature baseddetectors, and even firewalls can be used as alertgenerators. In all cases, the amount of data could bereduced substantially. Although there are situations asdescribed in Section 3.3—especially clusters that arewrongly 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 foran online application. We presented a novel technique for online alert aggregationand generation of meta-alerts. We have shown that thesheer amount of data that must be reported to a humansecurity expert or communicated within a distributedintrusion detection system, for instance, can be reducedsignificantly. The reduction rate with respect to the numberof alerts was up to 99.96 percent in our experiments. At thesame time, the number of missing attack instances isextremely low or even zero in some of our experimentsand the delay for the detection of attack instances is withinthe range of some seconds only.In the future, we will develop techniques for interestingness-based communication strategies for a distributedIDS. 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

[1] S. Axelsson, “Intrusion Detection Systems: A Survey and
Taxonomy,” Technical Report 99-15, Dept. of Computer Eng.,
Chalmers Univ. of Technology, 2000.

[2] M.R. Endsley, “Theoretical Underpinnings of Situation Awareness: A Critical Review,” Situation Awareness Analysis and Measurement, M.R. Endsley and D.J. Garland, eds., chapter 1, pp. 3-32, Lawrence Erlbaum Assoc., 2000.

[3] C.M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.

[4] M.R. Henzinger, P. Raghavan, and S. Rajagopalan, Computing on Data Streams. Am. Math. Soc., 1999.

[5] A. Allen, “Intrusion Detection Systems: Perspective,” Technical Report DPRO-95367, Gartner, Inc., 2003.

[6] F. Valeur, G. Vigna, C. Kru¨ gel, and R.A. Kemmerer, “A
Comprehensive Approach to Intrusion Detection Alert Correlation,” IEEE Trans. Dependable and Secure Computing, vol. 1, no. 3, pp. 146-169, July-Sept. 2004.

[7] H. Debar and A. Wespi, “Aggregation and Correlation of
Intrusion-Detection Alerts,” Recent Advances in Intrusion Detection, W. Lee, L. Me, and A. Wespi, eds., pp. 85-103, Springer, 2001.

[8] D. Li, Z. Li, and J. Ma, "Processing Intrusion Detection Alerts in Large-Scale Network," Proc. Int'l Symp. Electronic Commerce and Security, pp. 545-548, 2008.

[9] F. Cuppens, "Managing Alerts in a Multi-Intrusion Detection Environment," Proc. 17th Ann. Computer Security Applications Conf. (ACSAC '01), pp. 22-31, 2001.

[10] A. Valdes and K. Skinner, "Probabilistic Alert Correlation," Recent Advances in Intrusion Detection, W. Lee, L. Me, and A. Wespi, eds. pp. 54-68, Springer, 2001.

[11] K. Julisch, "Using Root Cause Analysis to Handle Intrusion Detection Alarms," PhD dissertation, Universita¨ t Dortmund, 2003. HOFMANN AND SICK: ONLINE INTRUSION ALERT AGGREGATION WITH GENERATIVE DATA STREAM MODELING 293

[12] T. Pietraszek, "Alert Classification to Reduce False Positives in Intrusion Detection," PhD dissertation, Universita¨ t Freiburg, 2006.

[13] F. Autrel and F. Cuppens, "Using an Intrusion Detection Alert Similarity Operator to Aggregate and Fuse Alerts," Proc. Fourth Conf. Security and Network Architectures, pp. 312-322, 2005.

[14] G. Giacinto, R. Perdisci, and F. Roli, "Alarm Clustering for Intrusion Detection Systems in Computer Networks," Machine Learning and Data Mining in Pattern Recognition, P. Perner and A. Imiya, eds. pp. 184-193, Springer, 2005.

[15] O. Dain and R. Cunningham, "Fusing a Heterogeneous Alert Stream into Scenarios," Proc. 2001 ACM Workshop Data Mining for Security Applications, pp. 1-13, 2001.

[16] P. Ning, Y. Cui, D.S. Reeves, and D. Xu, "Techniques and Tools for Analyzing Intrusion Alerts," ACM Trans. Information Systemsn Security, vol. 7, no. 2, pp. 274-318, 2004.

[17] F. Cuppens and R. Ortalo, "LAMBDA: A Language to Model a Database for Detection of Attacks," Recent Advances in Intrusion Detection, H. Debar, L. Me, and S.F. Wu, eds. pp. 197-216, Springer, 2000.

[18] S.T. Eckmann, G. Vigna, and R.A. Kemmerer, "STATL: An Attack Language for State-Based Intrusion Detection," J. Computer Security, vol. 10, nos. 1/2, pp. 71-103, 2002.

[20] M.S. Shin, H. Moon, K.H. Ryu, K. Kim, and J. Kim, "Applying Data Mining Techniques to Analyze Alert Data," Web Technologies and Applications, X. Zhou, Y. Zhang, and M.E. Orlowska, eds. pp. 193-200, Springer, 2003.

[21] J. Song, H. Ohba, H. Takakura, Y. Okabe, K. Ohira, and Y. Kwon, "A Comprehensive Approach to Detect Unknown Attacks viaIntrusion Detection Alerts," Advances in Computer ScienceASIAN2007, Computer and Network Security, I. Cervesato, ed., pp. 247-253,Springer, 2008.

[22] R. Smith, N. Japkowicz, M. Dondo, and P. Mason, "UsingUnsupervised Learning for Network Alert Correlation,"Advances in Artificial Intelligence, R. Goebel, J. Siekmann, andW. Wahlster, eds. pp. 308-319, Springer, 2008.