Online Incursion Aware Aggregation With Generative Facts Issue Modeling

[1]Bejjam Naresh, [2]R. Naveen

[1,2]Dept. of CSE, Swarna Bharathi Institute of Technology and Science, Khammam, AP, India

Abstract

Meta-alerts is the basis for reporting to security experts or for communication within a distributed intrusion detection system. . 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. 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. Intrusion detection can be used to identify the types of hackers attempting to tress pass into the system, thus we use the concept of alerts to cluster the types of attacks and the further counter measures, by using the concept of firewalls. . 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

Keywords

Meta-alerts, Intruding Alerts, Intrusion Detection, Hacking

I. 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. At present, most IDS are quite reliable in detecting suspicious actions by evaluating TCP/IP connections or log files, for instance. Once AN ID finds a suspicious action, it immediately creates an alert which contains information about the source, target, and estimated type of the attack. 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.

This problem is not new, but current solutions are typically based on a quite simple sorting of alerts, e.g., according to their source, destination, and attack type. Under real conditions such as the presence of classification errors of the low-level IDS (e.g., false alerts), uncertainty with respect to the source of the attack due to spoofed IP addresses, or wrongly adjusted time windows, for instance, such an approach fails quite often. Our approach has the following distinct properties:

- It is a generative modeling approach [3] Using probabilistic methods. Assuming that attack 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 [4]. Thus, it can be applied online and under harsh timing constraints.

II. Related Works

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 (cf. [5], for instance). Besides others, one drawback is the large amount of alerts produced. Recent research focuses on the correlation of alerts from (possibly multiple) IDS.Probably, the most comprehensive approach to alert correlation is introduced In [7], a similar approach is used to eliminate duplicates, i.e., alerts that share the same quadruple of source and destination address as well as source and destination port. In addition, alerts are aggregated (online) into predefined clusters (so-called situations) in order to provide a more condensed view of the current attack situation. 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. 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.

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. 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.

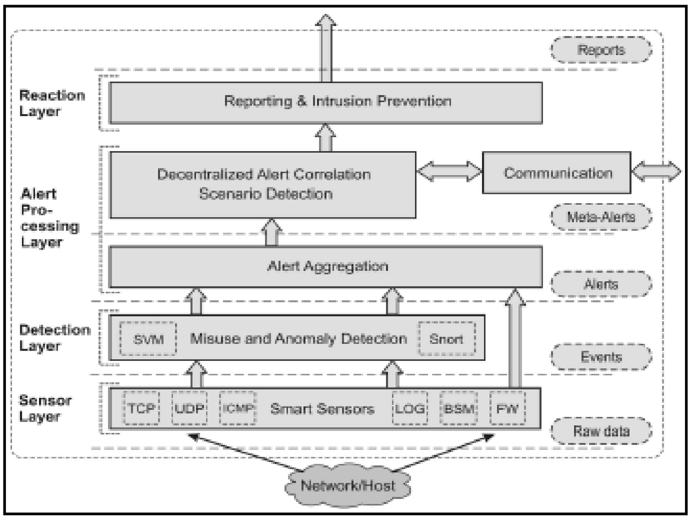


Fig. 1: Architecture of an Intrusion Detection Agent

III. Anovel 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. To outline the preconditions and objectives of alert aggregation, we start with a short sketch of our intrusion framework.

A. Collaborating Intrusion Detect-ion 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) 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.

B. 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 4together 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.

C. Offline Alert Aggregations

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.Only two of the attributes are shown and the correspondence of alerts and (true or estimated) attack instances is indicated by different symbols.has observations of the detectors (alerts) in the attribute space without attack instance labels as outlined in Fig. 2b. 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

- 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.
- 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.

Some additional remarks must be made:

1. Initialization of Model Parameters

The aim of the initialization is to find good initial values. Instead of using a random initialization which results in higher runtimes and sub-optimal solutions, we use a heuristic which we have successfully applied to the training of radial basis function neural networks. This heuristic selects as initial cluster centers a set of alerts with a large spread in the attribute space.

2. Hard Assignment of Alerts to Components

More general EM algorithms make a gradual assignment of alerts to components in the E step (cf. responsibilities in [3]). In practical applications, a hard assignment reduces the runtimes significantly at the cost of slightly worse solutions in some situations. In our case, this is acceptable as we do not want to find the optimal model parameters at the end, but to generate the optimal set of meta-alerts.

3. Stopping Criterion

An EM algorithm guarantees that the set of parameters is improved in each step. In addition, due to the hard assignment of alerts, there exist a limited number of possible assignments. Thus,

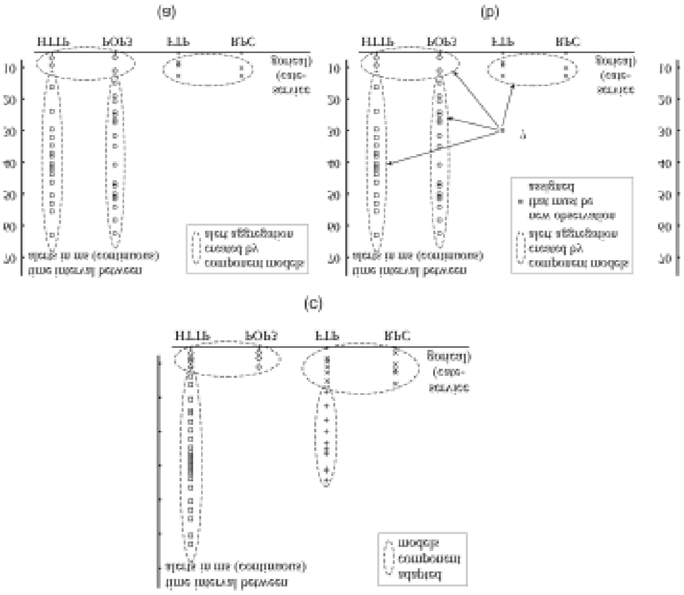


Fig. 2: Example Illustrating the Principle of online alert aggregation (artificial attack situation). (a) Existing model: Components have been created by the alert aggregation module. These components are the basis for meta-alert generation. (b) Assignment problem: New observations must either be assigned 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.ne important task has not been mentioned so far: The estimation of the number of components (clusters) J from the set of observed samples A. Up to now, we assumed that this number is given. In our case, the number of clusters shall correspond to the number of attack instances in an ideal case. To solve this problem, cluster validation measures can be used to assess the quality of a clustering result. A clustering algorithm is started several times with a varying number of clusters; the quality for each clustering result is assessed Numerically, and finally the results are compared and the number that optimizes this measure is chosen (so-called relative criterion, cf.).

D. 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. 2a—has several tasks, cf. Fig. 2b:

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)

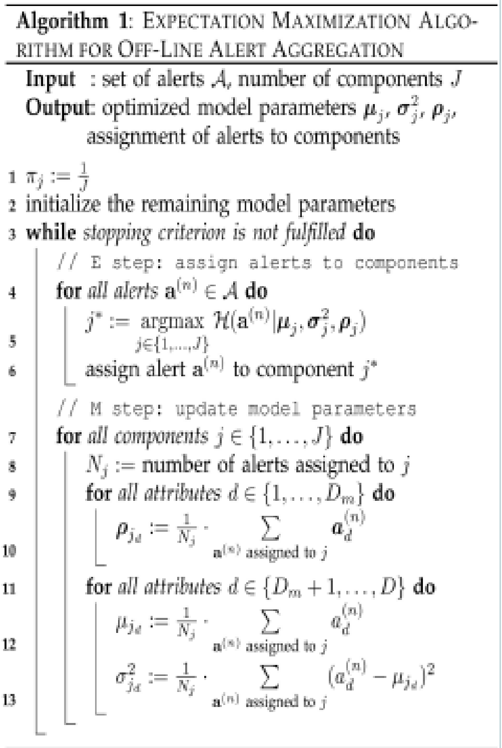
The occurrence of new attack instances must be stated. New components must be parameterized accordingly.

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. 2c.Clearly, there is a trade-off between runtime (or reaction time) and accuracy. For example, it is hardly possible to decide upon the existence of a new attack instance when only one observation is made. From the viewpoint of our objectives (cf. Section 3.1), the tasks 1 and 2 are more time critical than task 3.

Algorithm 2 describes the online alert aggregation. If a new alert is observed we first have to decide whether a first component has to be created. In this case, we initialize its Parameters with information taken from this alert. Random, small values are added, for example, to prevent any subsequent optimization steps from running into singularities of the respective likelihood function [3]. Otherwise, we have to decide whether the alert has to be associated with an existing component or not, i.e., whether we believe that it belongs to an ongoing attack instance or not. Provisionally, we assign the alert to the most likely component (E step) and optimize the parameters of this component (M step). For the reason of temporal efficiency, we do not conduct a sequence of E and M steps for the overall model. In some tests [9], it turned out that our main goal—not to miss any attack instances, see Section 3.1—can be achieved this way with substantially lower runtimes but at the cost of some redundant meta-alerts (due to split of clusters, see Section 3.3).

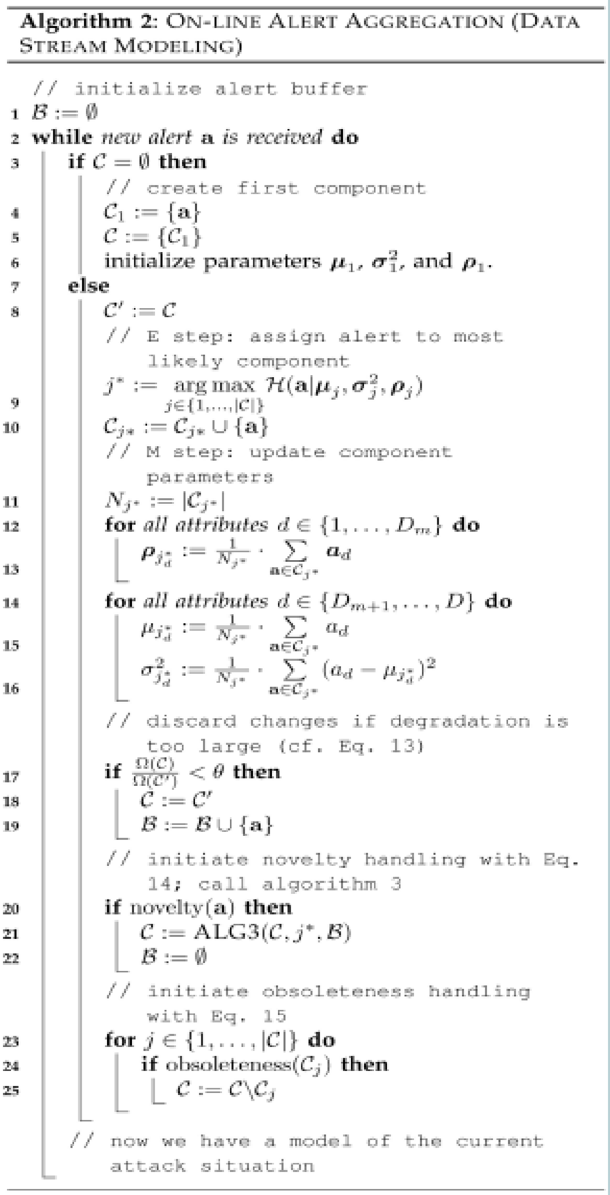
D. Meta-Alert Generation and Format



Algorithm 1: Expectation Maximization Algorithm for Off-Line Alert Aggregation

```
Input  : set of alerts $\mathcal{A}$, number of components $J$
Output: optimized model parameters $\mu_j, \sigma_j^2, \rho_j$,
        assignment of alerts to components
$\pi_j := \frac{1}{J}$
initialize the remaining model parameters
while stopping criterion is not fulfilled do
    // E step: assign alerts to components
    for all alerts $\mathbf{a}^{(n)} \in \mathcal{A}$ do
        $j^* := \operatorname{argmax}_{j \in \{1,\dots,J\}} \mathcal{H}(\mathbf{a}^{(n)} | \mu_j, \sigma_j^2, \rho_j)$
        assign alert $\mathbf{a}^{(n)}$ to component $j^*$
    // M step: update model parameters
    for all components $j \in \{1,\dots,J\}$ do
        $N_j :=$ number of alerts assigned to $j$
        for all attributes $d \in \{1,\dots,D_m\}$ do
            $\rho_{j_d} := \frac{1}{N_j} \cdot \sum_{\mathbf{a}^{(n)} \text{ assigned to } j} a_d^{(n)}$
        for all attributes $d \in \{D_m + 1,\dots,D\}$ do
            $\mu_{j_d} := \frac{1}{N_j} \cdot \sum_{\mathbf{a}^{(n)} \text{ assigned to } j} a_d^{(n)}$
            $\sigma_{j_d}^2 := \frac{1}{N_j} \cdot \sum_{\mathbf{a}^{(n)} \text{ assigned to } j} (a_d^{(n)} - \mu_{j_d})^2$
```

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



Algorithm 2: On-line Alert Aggregation (Data Stream Modeling)

```
// initialize alert buffer
$\mathcal{B} := \emptyset$
while new alert $\mathbf{a}$ is received do
    if $\mathcal{C} = \emptyset$ then
        // create first component
        $\mathcal{C}_1 := \{\mathbf{a}\}$
        $\mathcal{C} := \{\mathcal{C}_1\}$
        initialize parameters $\mu_1$, $\sigma_1^2$, and $\rho_1$.
    else
        $\mathcal{C}' := \mathcal{C}$
        // E step: assign alert to most
           likely component
        $j^* := \arg\max_{j \in \{1,\dots,|\mathcal{C}|\}} \mathcal{H}(\mathbf{a} | \mu_j, \sigma_j^2, \rho_j)$
        $\mathcal{C}_{j*} := \mathcal{C}_{j*} \cup \{\mathbf{a}\}$
        // M step: update component
           parameters
        $N_{j^*} := |\mathcal{C}_{j^*}|$
        for all attributes $d \in \{1,\dots,D_m\}$ do
            $\rho_{j_d^*} := \frac{1}{N_{j^*}} \cdot \sum_{\mathbf{a} \in \mathcal{C}_{j^*}} a_d$
        for all attributes $d \in \{D_{m+1},\dots,D\}$ do
            $\mu_{j_d^*} := \frac{1}{N_{j^*}} \cdot \sum_{\mathbf{a} \in \mathcal{C}_{j^*}} a_d$
            $\sigma_{j_d^*}^2 := \frac{1}{N_{j^*}} \cdot \sum_{\mathbf{a} \in \mathcal{C}_{j^*}} (a_d - \mu_{j_d^*})^2$
        // discard changes if degradation is
           too large (cf. Eq. 13)
        if $\frac{\Omega(\mathcal{C})}{\Omega(\mathcal{C}')} < \theta$ then
            $\mathcal{C} := \mathcal{C}'$
            $\mathcal{B} := \mathcal{B} \cup \{\mathbf{a}\}$
        // initiate novelty handling with Eq.
           14; call algorithm 3
        if novelty($\mathbf{a}$) then
            $\mathcal{C} :=$ ALG3($\mathcal{C}, j^*, \mathcal{B}$)
            $\mathcal{B} := \emptyset$
        // initiate obsoleteness handling
           with Eq. 15
        for $j \in \{1,\dots,|\mathcal{C}|\}$ do
            if obsoleteness($\mathcal{C}_j$) then
                $\mathcal{C} := \mathcal{C} \setminus \mathcal{C}_j$
    // now we have a model of the current
       attack situation
```

IV. Experimental Results

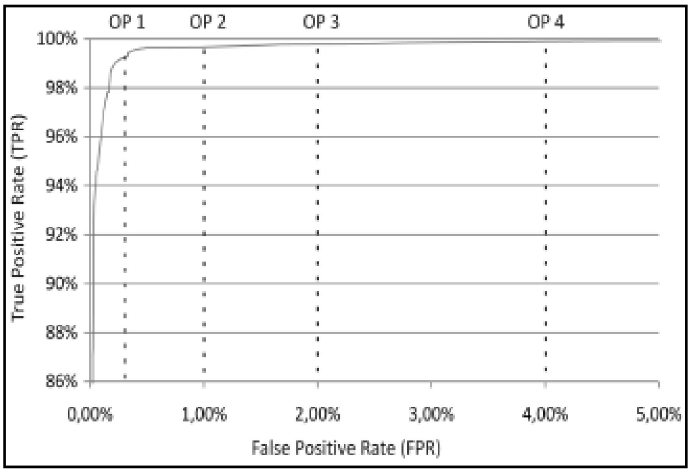


Fig. 3: ROC Curve for the SVM Detector

A. Description of the Benchmark Data Sets

1. DARPA Data

For the DARPA evaluation , several weeks of training and test data have been generated on a test bed that emulates 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 that have been launched against the various target hosts. At the detection layer, we apply SVM to classify the sensor events. Four operating points (OP) are marked.

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.

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. Instead, we use log messages that were generated by a CISCO PIX system, which is a well-known commercial FW and network address translation (NAT) device.

B. 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.

Several independent attackers. In the DARPA data set, some attack instances are labeled as a single attack instance although they are in fact comprised of the actions of several independent attackers. A typical example is a WAREZCLIENT instance in week four: There, attackers download illegally provided software from a compromised FTP server.

Long attack duration. Attack instances with a long duration are often split into several meta-alerts. Typical examples are slow or hidden port scans or (distributed) denial of service attacks which can last several hours.

Bidirectional communication.TCP/IP-based 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.

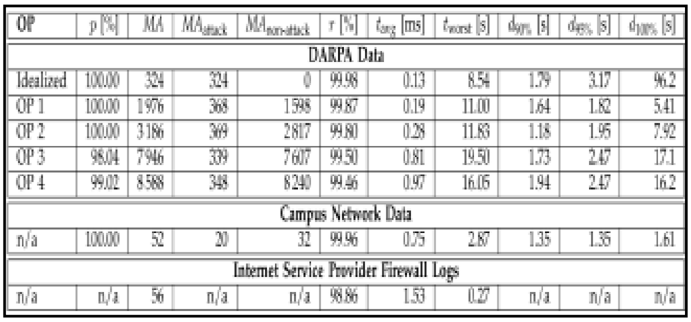
C. Results

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.

1. 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)and 99.02 percent (OP 4). In OP 3, a FORMAT instance and a MULTIHOP instance are missed. 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 an OP with higher FPR. Hence, as the false alerts easily outnumber the four true FORMAT alerts within this cluster, the FORMAT instance gets lost. For the MULTIHOP instance, for which we have 19 alerts, the situation is more complex.

Table 2:



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

Communication between two hosts results in packets transmitted in both directions. If the detector layer produces alerts for both directions (e.g., due to malicious packets), the source and destination IP address are swapped, which in the end results in two meta-alerts.

2. Campus Network Data

For the campus network data, for which the IDS Snort was used to create alerts, quite similar results could be achieved (see Table 2). All attack instances that have been launched were correctly detected. For the 17 attack instances with 128,816 alerts, 52 meta-alerts were created, which is equivalent to a reduction rate of 99.96 percent. Again, the majority of meta-alerts is caused by false alerts.

3. Internet Service Provider Firewall Logs

For the firewall log data, the proposed alert aggregation could also be applied successfully. As Table 2 shows, 56 meta-alerts were created for the 4,989 alerts, which is a reduction rate of 98.86 percent. As it is not possible to specify a percentage of detected attack instances, we analyzed the content of the 56 resulting meta-alerts: In many cases, it is possible to find a particular reason for the meta-alerts

D. 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.

V. 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.

VI. Acknowledgments

This work was partly supported by the German Research Foundation (DFG) under grant number SI 674/3-2. The authors would like to thank D. Fisch for his support in preparing one of the data sets. The authors highly appreciate the suggestions of the anonymous reviewers that helped them to improve the quality of the article.

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