Alert Verification-Determining the Success of Online Intrusion

1N. Sravani, 2M. Anuraghamayi
1,2Dept. of CSE, A.S.R College of Engineering & Tech., Tetali, Tanuku, AP, India

Abstract
Intrusion Detection System (NIDS) [8] that tries to detect malicious activity such as denial of service attacks, port scan or even attempts to crack into computer by monitoring network traffic. Network Intrusion detection is mainstream to identify alert aggregation and to cluster different alerts produced by low-level intrusion detection systems firewalls etc. Belonging to a specific attack instance which has been initiated by an attacker at a certain point in time, thus, meta-alerts can be generated for the clusters that contain all the relevant information whereas the amount of data (i.e., alerts) can be reduced substantially. Meta-alerts may then be the basis for reporting to security experts or for communication within a distributed intrusion detection system. We propose a novel technique for online alert aggregation which is based on a dynamic, probabilistic model of the current attack situation. Basically, it can be regarded as a data stream version of a maximum likelihood approach for the estimation of the model parameters. 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
Intrusion Detection, Probabilistic Techniques, Alert Aggregation

I. Introduction
An intrusion Detection System (IDS) is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a Management Station. Some systems may attempt to stop an intrusion attempt but this is neither required nor expected of a monitoring system Intrusion Detection and Prevention Systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, and reporting attempts. In addition, organizations use IDPSes for other purposes, such as identifying problems with security policies, documenting existing threats, and deterring individuals from violating security policies. IDPSes have become a necessary addition to the security infrastructure of nearly every organization

Types of Intrusion Detection System:

A. Network Intrusion Detection System (NIDS)
It is an independent platform that identifies intrusions by examining network traffic and monitors multiple hosts, developed in 1986 by Pete R. Network intrusion detection systems gain access to network traffic by connecting to a network hub, network switch configured for port mirroring or network tap. In a NIDS, sensors are located at choke points in the network to be monitored, often in the demilitarized (DMZ) or at network borders. Sensors capture all network traffic and analyzes the content of individual packets for malicious traffic. An example of a NIDS is Snort.

B. Host based Intrusion Detecting System (HIDS)
It consists of an agent on a host that identifies intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files, capability databases, Access control list, etc.) and other host activities and state. In a HIDS, sensors usually consist of a software agent. Some application-based IDS are also part of this category. Examples of HIDS are Tripwire and OSSEC.

C. Stack based Intrusion Detecting System (SIDS)
This type of system consists of an evolution to the HIDS systems. The packets are examined as they go through the TCP/IP stack and, therefore, it is not necessary for them to work with the network interface in promiscuous mode. This fact makes its implementation to be dependent on the Operating System that is being used. Intrusion detection systems can also be system-specific using custom tools and honeypots.

D. Comparison with Firewalls
Though they both relate to network security, an intrusion detection system (IDS) differs from a firewall in that a firewall looks outwardly for intrusions in order to stop them from happening. Firewalls limit access between networks to prevent intrusion and do not signal an attack from inside the network. An IDS evaluates a suspected intrusion once it has taken place and signals an alarm. An IDS also watches for attacks that originate from within a system. This is traditionally achieved by examining network communications, identifying heuristics and patterns (often known as signatures) of common computer attacks, and taking action to alert operators. A system that terminates connections is called an intrusion prevention system, and is another form of an application layer firewall.

All Intrusion Detection Systems use one of two detection techniques:
A statistical anomaly-based IDS determines normal network activity like what sort of bandwidth is generally used, what protocols are used, what ports and devices generally connect to each other- and alert the administrator or user when traffic is detected which is anomalous (not normal Signature based IDS monitor’s packets in the Network and compares with pre-configured and pre-determined attack patterns known as signatures. The issue is that there will be lag between the new threat discovered and Signature being applied in IDS for detecting the threat. During this lag time your IDS will be unable to identify the threat.

Fig. 1: Intrusion Detection System
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 create 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. Alerts may originate from low-level IDS such as those mentioned above, from firewalls, etc. Alerts that belong to one attack instance must be clustered together and meta-alerts must be generated for these clusters. It is a generative modeling approach using probabilistic methods.

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 [5]. 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 [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. The definition of such situations is also used in [8] to cluster alerts. In [9], alert clustering is used to group alerts that belong to the same attack occurrence.

In [20], an off-line 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 [11] 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.

A completely different clustering approach is presented in [22]. There, the reconstruction error of an Auto-Associator 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 off-line 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. In addition, it 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.

III. Existing System

Existing IDS are optimized to detect attacks with high accuracy still have various disadvantages that have been outlined in a number of publications and a lot of work has been done to analyze IDS. 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 [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. The definition of such situations is also used in [8] to cluster alerts. In [9], 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 an approach to alert correlation is presented in [10]. A weighted, attribute-wise similarity operator is used to decide whether to fuse two alerts or not. However, as already stated in and [5], this approach suffers from the high number of parameters that need to be set. The similarity operator presented in [6] 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 [7], 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.

IV. Proposed System

Network Security is an important issue in our current Task. Providing certain firewall, antivirus, security for password become hectic even though we cant control the network type of anomalies or misuse activities. To maintain the security issue at a time we introduce a novel method call online alert aggregation which is based on dynamic or probabilistic model of the current attack situation.

A. Application Specific for Alert Aggregation

Server module is the main module for this project. This module acts as the Intrusion Detection System. This module consists of four layers viz. sensor layer (which detects the user/client etc.), Detection layer, alert processing layer and reaction layer. In addition there is also Message Log, where all the alerts and messages are stored for the references. This Message Log can also be saved as Log file for future references for any network environment.

Client module is developed for testing the Intrusion Detection System. In this module the client can enter only with a valid
user name and password. If an intruder enters with any guessing passwords then the alert is given to the Server and the intruder is also blocked. Even if the valid user enters the correct user name and password, the user can use only for minimum number of times. For example even if the valid user makes the login for repeated number of times, the client will be blocked and the alert is sent to the admin. In the process level intrusion, each client would have given a specific process only. For example, a client may have given permission only for P1 process. If the client tries to make more then these processes the client will be blocked and the alert is given by the Intrusion Detection System. In this client module the client can be able to send data. Here, when ever data is sent Intrusion Detection System checks for the file. If the size of the file is large then it is restricted or else the data is sent.

The DARPA Data Set is used to check the technique of the Online Intrusion Alert Aggregation with Generative Data Stream Modeling. The DARPA data set is downloaded and separated according to each layers. So we test the instance of DARPA Dataset using the open file dialog box. Whenever the dataset is chosen based on the conditions specified the Intrusion Detection System works.

Mobile- The traditional system uses the message log for storing the alerts. In this system, the system admin or user can get the alerts in their mobile. Whenever alert message received in the message log of the server, the mobile too receives the alert message. Whenever an attack is launched the Intrusion Detection System must be capable of detecting it. So our system will also be capable of detecting such attacks. For example if an IP trace attack is launched, the Intrusion Detection System must detect it and must kill or block the process.

B. IDS Algorithm
- Select the n’layers needed for the whole IDS.
- Build Sensor Layer to detect Network and Host Systems.
- Build Detection Layer based on Misuse and Anomaly detection technique.
- Classify various types of alerts. (For example alert for System level intrusion or process level intrusion)
- Code the system for detecting various types of attacks and alerts for respective attacks.
- Integrate the system with Mobile device to get alerts from the proposed IDS.
- Specify each type of alert on which category it falls, so that user can easily recognize the attack type.
- Build Reaction layer with various options so that administrator/user can have various options to select or react on any type of intrusion.
- Test the system using Attack Simulation module, by sending different attacks to the proposed IDS.
- Build a log file, so that all the reports generated can be saved for future references.

C. Alert Aggregation when the user is at Offline
Assume that a host with an ID agent is exposed to a certain intrusion situation with one or several attacks launch several attack instances belonging to various attack types. The attack instances each cause a number of alerts with various attributes values.
- 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.
- 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.
- Several clusters are wrongly combined into one: This situation is definitely problematic as attack instances may be missed.

C. Dataset Alert Aggregation
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 at point in time the ID agent which is assumed to have a model of the current situation.

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.

Component creation (novelty detection): The occurrence of new attack instances must be stated. New components must be parameterized accordingly.

Component deletion (obsoleteness detection): The completion of attack instances must be detected and the respective components must be deleted from the model.

V. Conclusion
Intrusion alert aggregates varies subtask of intrusion detection. Different alerts produced by low-level intrusion detection systems, firewalls, etc is evaluated to make the system more foolproof. The project identifies and cluster intrusion alerts to make the segregation of various attacks being generated. To improve the efficacy of intrusion detection system, Meta alerts are generated which contain all the relevant information. 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.

We extend our idea of sending Intrusion alerts to the mobile. This makes the process easier and comfortable. Online Intrusion Alert Aggregation with Generative Data Stream Modeling does not degrade system performance as individual layers are independent and are trained with only a small number of features, thereby, resulting in an efficient system. Online Intrusion Alert Aggregation with Generative Data Stream Modeling is easily customizable and the number of layers can be adjusted depending upon the requirements of the target network. Our framework is not restrictive in using a single method to detect attacks. Different methods can be seamlessly integrated in our framework to build effective intrusion detectors. Our framework has the advantage that the type of attack can be inferred directly from the layer at which it is detected. As a result, specific intrusion response mechanisms can be activated for different attacks. Along With Alert aggregation Improving Efficiency of finding Attacks In Different types of Layers in intrusion detection System
References


Ms. N. Sravani is a student of A.S.R College of Engineering & Technology, Tamuku. Presently she is pursuing her M.Tech (C.S.E) from this college and she received her B.Tech(I.T) from JNTU Kakinada, in the year 2010. Her area of interest includes Computer Networks and Object oriented Programming languages in Computer Applications.

Mrs. N. Anuragamayi, excellent teacher Received M.Tech (CSE) from JNTUK university is working as Assistant Professor in Akula Sree Ramulu College Engineering & Technology. She has 5 years of teaching experience in various engineering colleges. Her area of Interest includes Information security, Data Warehouse and Data Mining, flavors of Unix Operating systems and other advances in computer Applications.