European Journal of Molecular & Clinical Medicine
ISSN 2515-8260 Volume 07, Issue 02, 2020

AN EFFICIENT IOT BASED ALERT INTRUSION SYSTEM FOR GENERATIVE MOBILE DATA STREAM APPLICATION

[1]Dr. Lokesh P Gagnani,[2]Ramesh S, [3]R.Senthil, [4]Krishnakumar V, [5]Dr.SyedKhasim

[1]Associate Professor, Department of Information Technology, SVBIT, Vasan, Gandhinagar -382650. India.

[2]Associate Professor, Department of Electronics and Communication Engineering, Mailam Engineering College, Mailam-604304, Tindivanam, Villupuram District, Tamilnadu,India.

[3]Assistant Professor,Information Technology,SRM Institute of Science and Technology, Delhi-NCR Campus, Delhi-Meerut Road, Modinagar, Ghaziabad-201204, Uttar Pradesh.

[4]Assistant Professor, Electronics and Communication Engineering, Kongunadu College of Engineering and Technology,Namakkal to Trichy Mainroad,Trichy-621215.

[5]Professor, Computer Science & Engineering, Dr.Samuel George Institute of Engineering & Technology, Markapur, Prakasam District, 523316, India,

Abstract

IOT Alert based aggregation is an important subtask of intrusion detection. The goal is to identify and to cluster different alert produced by low-level intrusion detection systems, firewalls, etc.belonging to a specific attack instance which has beeninitiated by an attacker at a certain point in time. Thus, meta-alerts can be generated for the clusters that contain all the relevantinformation whereas the amount of data (i.e., alerts) can be reduced substantially. Meta-alerts may then be the basis for reporting tosecurity experts or for communication within a distributed intrusion detection system. This method proposes a novel technique for online alertaggregation which is based on a dynamic, probabilistic model of the current attack situation. Basically, it can be regarded as a datastream version of a maximum likelihood approach for the estimation of the model parameters.It describes the problem of intrusion detection in detail and analyze various well known methods for intrusion detection with respect to two critical requirements using SparkV Dataset.

Index: IoT Model, Intrusion Detection, Generative Mobile data stream modeling, SparkV Dataset

1. Introduction

Intrusion detection systems (IDS), along with other security measures such as virtual private networks, authentication systems or encryption methods, are very important to ensure information security. They detect the activities of attackers or attacking devices on the network or in a host-based manner and help prevent various threats to networks and hosts by detecting malicious or malicious detection strategies. Currently, most IDSs are highly reliable for detecting suspicious activity on TCP / IP connections or log files. If an IDS detects suspicious activity, it immediately generates an alert containing information about the source, target and type of attack (e.g., SQL injection, buffer overflow or service rejection). Intrusion caused by a single attack incident - a specific type of attacker initiated by a particular attacker at a particular time often spreads across multiple network connections and log file entries. An attack often causes hundreds or thousands of warnings [1]. IDS generally focus on identifying the types of attacks, but not distinguishing between different attack events. Also, if you check thousands of network packets or log file entries, even the smallest false alarms can lead to a large number of false alarms. As a result, the IDS produces many alerts of abstract magnitude. This flood of warnings is very difficult for a human safety expert to diagnose, and decisions from a single warning are at high risk of being misplaced [2].

In our opinion, one should be aware of the "perfect" IDS situation and at any given time be aware of the events of the attack and the (different types) of the attacker and what is going on in its environment. In this paper is taking an important step towards this goal by introducing and evaluating new technology for warning compilation. Low level IDS and firewalls (FW) alerts may occur as mentioned above [3]. Alerts from an attack incident should be grouped together and meta alerts will be created for these groups. The main goal is to significantly reduce the number of alerts without losing important information needed to detect an ongoing attack. I do not want to miss meta alerts, but I accept somewhat false or unnecessary meta alerts. This problem is not new, but current solutions are usually based on very general warnings, e.g. Depending on their source, target and type of attack. In real situations such as low-level IDS classification errors (e.g., false alarms), such an approach often fails due to uncertainty associated with the source of the attack or incorrectly adjusted time windows due to spoofed IP addresses [4].

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" alert modeling 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.

Intrusion detection warning is a comprehensive approach to interaction

Alert correlation is the process of analyzing alerts generated by one or more intrusion detection systems and providing a more concise and high-level view of what has been entered or attempted. Although the interrelationship process is often presented as a single step, the analysis actually involves several elements, each of which has a specific purpose. Unfortunately, most approaches to interaction focus only on certain aspects of the process, providing formal approaches and methods that address only specific interrelated issues. This paper provides a general interrelationship model, consisting of integral parts and a framework based on this model [5]. The tool that uses the framework to determine how each component contributes to the overall objectives of the interaction is applied to data sets that detect several known intrusions. The results of these experiments show that correlated factors are effective in reducing alertness and abstraction. They show that the effect of a component depends on the nature of the data set analyzed.

Process alerts that detect intrusions on large-scale networks

The intrusion detection system generates a lot of alerts, most of which are false positives. This paper seeks to connect multiple intrusion detection systems on a large scale network to minimize excessive false alarms and detect real-time real-time security events [6]. To process these alerts, two algorithms called minimize and cluster are developed in this paper, which can remove false alarms and cluster multiple identical alerts, respectively. Experiment shows that 90% of raw alerts are filtered and less than 1% are fully processed for analysis.

2. System Analysis

Wong et al,Most existing IDSs are adapted to detect attacks with high accuracy. However, they still have the various disadvantages described in many publications and much effort has been put into analyzing IDS to guide future research. Among other things, Swang et al,a large error is a large number of warnings [7]. Alerts are issued only on system logs. The current IDS does not have a common framework for adding domain-specific knowledge to the specific needs of users or network administrators. Online intrusion alert aggregation with productive data stream modeling is a productive modeling approach using potential methods. Corg et al, suppose incidents of violence can be considered random processes as "productive" warnings; The goal of this model is to approximate and process using the maximum potential parameter calculation methods. Therefore, the beginning and end of the attack can be found [8]. This is a data stream approach, meaning that each monitor alert is processed only a few times. Therefore, it can be applied online and within strict time limits.

Albert Gho et al, in a specific project of online intrusion alert aggregation with generative data stream modeling, to expand our concept of sending intrusion alerts to mobile. This makes the process easier and more convenient [9]. Because individual layers are free and trained with very few features, online intrusion alert system performance with productive data stream modeling does not degrade. Online intrusion alert can be easily customized with product data stream modeling and the number of layers can be adjusted to suit the needs of the target network. Our framework is not limited to using a single method to detect attacks. Different methods can be integrated indefinitely in our framework to create effective intrusion detectors. Our framework has the advantage of being able to retrieve the type of attack directly from the detected layer. As a result, specific infiltration response systems can be activated for different attacks [10].

3. A Novelty-Driven Approach to Intrusion Alert Correlation

Distribution intrusion detection and prevention play an even more important role in securing computer networks. In a distributed intrusion detection system, alerts or high-level meta alerts are transmitted, integrated, and interconnected to overcome the limitations of traditional intrusion detection systems. Significant progress has been made, but existing systems still suffer from a variety of shortcomings: most of them are for data collection and distribution only, not analysis, or they rely on hierarchical or centralized organization and / or communication architecture. In addition, alerts or meta alerts are usually stored in a pre-defined area so that large amounts of alerts are not reduced before delivery. As a result, scalability is limited, and any central element in the structure provides a "single point of failure". Collaborative, self-organizing, load-balanced approach.

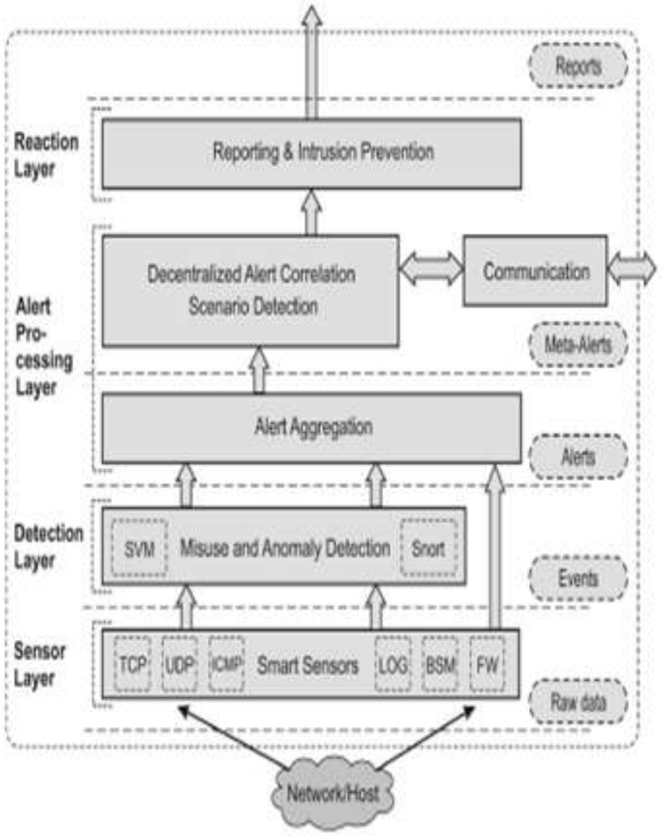

Figure 1: IoT based alter aggeration Intrusion system

4. System Implementation

The implementation phase involves careful planning and investigation of the existing system, including barriers to implementation, design of methods to achieve change and evaluation of change methods.

Server: The main module of this project is the server module. This module acts as an intrusion detection system. This module consists of four layers. Sensor layer (which finds the user / client), Detection layer, Alert processing layer and Response layer. There is also a message log, where all alerts and messages are stored for reference. This message can be saved as a log file for future reference to any network environment.

Client: Developed a client module to test the intrusion detection system. In this module the client can only log in with a valid username and password. If the intruder logs in with any S password, it will alert the server and block the intruder. Even if a valid user enters the correct username and password, the user can only use it a few times. For example, even if a valid user logs in several times, the client will intercept and send an alert to the administrator. In the process level intrusion, each client only registers a specific process. For example, only one client may authorize the P1 process. If the client tries to do more, these processes are blocked by the client and a system is provided to detect alert intrusions. The client can send data in this client module. Here, the intrusion detection system checks for a file whenever data is sent. If the file size is large it will be limited or data will be sent.

Spark v Dataset: This module is integrated into the server module. This is an offline type for intrusion testing. In this module, Spark v datasetdata set is used to test the technology of online intrusion alert aggregation along with product data stream modeling. The Spark v Datasetdata set is downloaded and sorted according to each layer. So I will check the example of Spark V dataset using open file dialog box. The intrusion detection system works whenever the dataset is selected based on specific circumstances.

Mobile: This module was developed using Android. The traditional system uses the message log to store alerts. In this system, the system admin or user can receive alerts on their mobile. Whenever a warning message arrives in the server's message log, the mobile also receives an alert message.

Attack simulation: In this module, we build our self-attack attack simulator to test the system. Attacks can be classified and simulated here. The intrusion detection system must be able to detect it whenever an attack begins. So our system can detect such attacks. For example, if an IP trace attack is triggered, the intruder detects it and must eliminate or prevent that process.

4.1 MISUSE BASED DETECTION ALGORITHM

Step 1: Select the ‘n’ layers needed for the whole IDS.
Step 2: Build Sensor Layer to detect Network and Host Systems.
Step 3: Build Detection Layer based on Misuse and Anomaly detection technique.
Step 4: Classify various types of alerts. (For example alert for System level intrusion or process level intrusion)
Step 5: Code the system for detecting various types of attacks and alerts for respective attacks.
Step 6: Integrate the system with Mobile device to get alerts from the proposed IDS.
Step 7: Specify each type of alert on which category it falls, so that user can easily recognize the attack type.
Step 8: Build Reaction layer with various options so that administrator/user can have various options to select or react on any type of intrusion.
Step 9: Test the system using Attack Simulation module, by sending different attacks to the proposed IDS.
Step 10: Build a log file, so that all the reports generated can be saved for future references.

There may be differentpotentially problematic situations:

1. False alerts are not recognized as such and wrongly assigned 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.

Types of Attacks	Number of Samples
Normal	Normal
Probe	Satan, Ipsweep, Portsweep, Nmap
DoS	Smurf, Neptune, Back, Teardrop, Pod, Land
R2L	Warezclient, Guess passwd, Imap, ftp write, Multihop, Phf, Spy
U2R	Buffer overflow, Rootkit, loadmodule, perl

Table1. Attack Types and Number of samples in 10% KDD Dataset

4.2 Performance Evaluation of a Collaborative Intrusion Detection System

Intrusion detection systems include two technologies: abuse detection and abuse detection. There are benefits to detecting abuse and detecting misconduct. Currently, these two technologies are being developed in conjunction with each other, although intrusive detection systems are not an effective way to assess the performance of collaborative identification. The rigorous mathematical evaluation equation needs to be established and analyzed. Depending on the information theory approach to analyze this problem, the ability to detect intrusions can be used for analysis and evaluation. In contrast, a system based on two intrusion detection systems, abuse and misuse detection, has better identification results.

The attack schedule is designed to reflect situations that I find difficult to identify. Most importantly, it is us
1. Multiple attacks at once (up to seven),
2. Incidents of violence that are partially or completely overlapping,
3. Most events in a short period of time,
4. Incidents of different attacks from similar sources,
5. Different attack periods, and
6. An attacker who changes his IP address during an attack.

To demonstrate that specific technology can also be used with traditional signature-based detectors, the extracted traffic open source IDS snore was analyzed, which detected a total of 128 attacks and generated 128,816 alerts [Table 1].

Percentage of cases detected (P): I think an attack incident is detected, especially if there is a specific meta warning. The percentage of detected events can be determined by dividing the number of detected events by the total number of events detected in the dataset. Sizes are calculated relative to the events that have the by-output of the detection layer, i.e., detected events are not considered detectors.

Meta Alert Number (MA) Reduction Rate (R): The number of meta alerts (MA) is mainly divided into the number of attack meta alerts that contain real alerts and the MA nontak of nontok meta alerts that contain mainly false alerts. Reduction rate 1 divides the number of meta alerts minus MA by the total number of alerts.

Average Runtime (TOW) Worst Runtime (Torst): The average runtime for each alert is measured in milliseconds. If uming has millions of alerts per day, the tow should be less than 100 ms for one alert. The worst runtime torque, measured in seconds, is how long the loop takes to execute, which involves running Algorithm 1 (the second several times).

Meta alert creation delay (d): It is clear that there is a definite delay until a meta alert is created for a new attack event. The meta-warning creation delay d measures the delay between the actual start of the example (i.e., the creation time of the first warning) and the creation of the first meta-warning for that example.

Measures to determine the type of attack: Often, the division of attack events into more meta alerts is due to the nature of the attack. In fact, many types of attacks have different and clearly identifiable stages. For example, an FTP-write attack performs three such steps: FTP login on port 21, FTP data transfer on port 20, and remote login on port 513. Therefore, it is very natural to divide it into three related meta warnings. Subsequent tasks in the warning processing layer handle such multistep attack conditions.

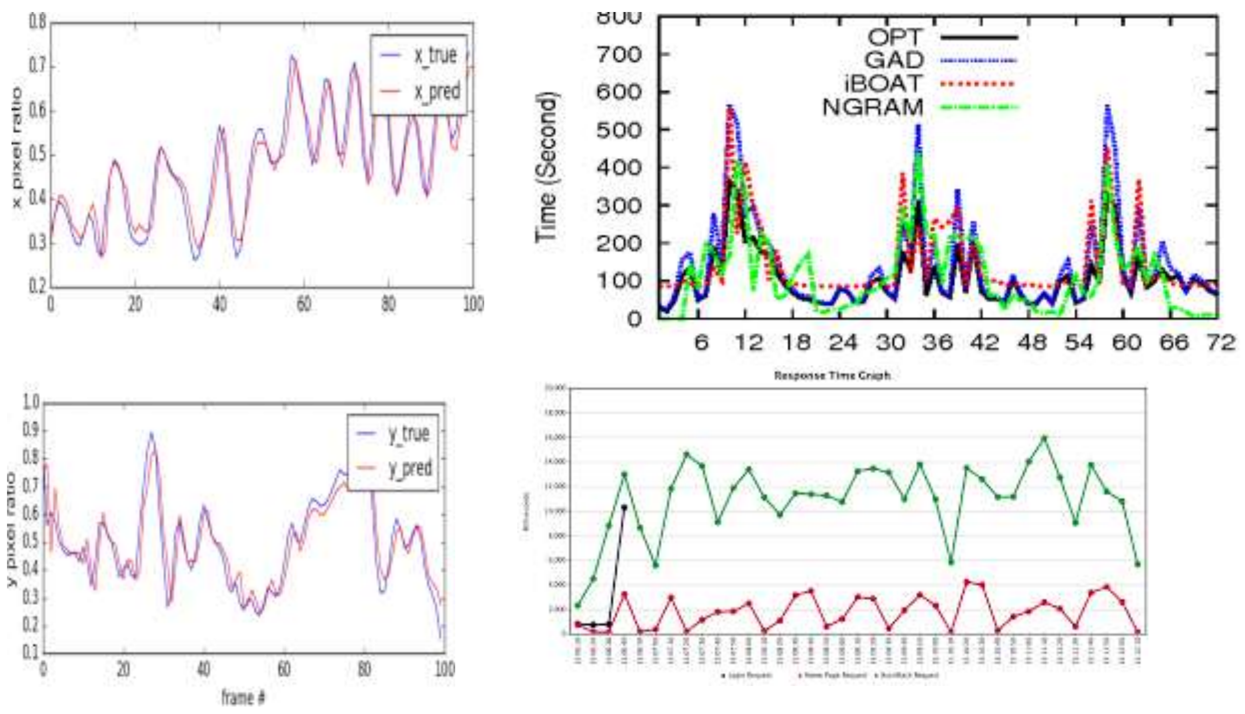

Figure 2: Simulation result for Spark V Model Dataset using Amazon Redshit

Many were independent attackers: In the Spark V dataset, some attack incidents are labeled as a single attack example, but in reality they contain the activities of many independent attackers. A typical example is Warezclientin the fourth week: there, the attacker downloads illegally provided software from an uncompromising FTP server. For example, an individual meta alert is generated for each attacker because coordinated rejection of a service attack means there is no cooperation between attackers.

5. Conclusion

In this workpresented a novel technique for online alert aggregation. We also addressed the problem of accuracy and efficiency of Intrusion Detection System. We developed the presented architecture and tested the system with the misuse based anomaly detection technique. We also proposed a misuse based anomaly detection algorithm for our system. As our contribution, to make the system more efficient in identify the intrusion alerts and also we extend this work by sending the Alerts as Message to the Network Administrator who governs the Network or Intrusion Detection System. Most of the present existing Intrusion Detection System does not have a generalized framework. Our proposed architecture is similar to layers, so according to the network environment, the network administrator can add or remove the layers. If a new updated version of detection comes in future, then it will be very easy to add the layer with our proposed system. As future enhancement, to test our system by launching various attacks to the system and we find how the system detects and reacts according to the developed IDS. Then mobile alert based part will be developed as part of future enhancements.

6. References

1. R. Agrawal, T. Imielinski, and A. Swami, “Mining Association Rules between Sets of Items in Large Databases,” Proc. ACM SIGMOD, vol. 22, no. 2, pp. 207-216, 2017.

2. Manikandan.S, ManikandaKumaran.K, Palanimurugan.S, Aravindan.S and Praveen Kumar.S, "Detecting and Preventing Distributed Denial of Service (DDOS) Attacks using BOTNET Monitoring System", International Journal of Engineering and Computer Science, ISSN:2319-7242, Vol.3,Issue.12,pp:9717-9720,December;'2019.

3. R. Bace and P. Mell, Intrusion Detection Systems, Computer Security Division, Information Technology Laboratory, Nat’l Inst. of Standards and Technology, 2020.

4. KDD Cup 1999 Intrusion Detection Data,http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2017.

5. Haripriya S, Indumathi , S.Manikandan, "Virtual Network Connection Using Mobile Phones", COMPUSOFT, An International Journal of Advanced Computer Technology, ISSN:2320-0790, Vol.03, Issue: 06, pp-980-984, June-2016.

6. Y. Bouzida and S. Gombault, “Eigenconnections to Intrusion Detection,” Security and Protection in Information Processing Systems, pp. 241-258, 2014.

7. J.P. Anderson, Computer Security Threat Monitoring and Surveillance, http://csrc.nist.gov/publications/history/ande80.pdf, 2010.

8. Autonomous Agents for Intrusion Detection,http://www.cerias.purdue.edu/research/aafid/, 2010.

9. S. Manikandan, K. Manikanda Kumaran, "Performance Analysis of Mobile Ad-Hoc Network Routing Protocols using Network Simulator – 2", COMPUSOFT, An International Journal of Advanced Computer Technology, ISSN:2320-0790, Vol.03, Issue: 06, pp-957-960, June-2014.

10. B. Bohem, Software Engineering Economics, Prentice Hall, Englewood Cliffs, 1981 [Briand et al 94] L. Briand, S. Morasca, V. Basili, Defining and Validating High-Level Design Metrics, Tech. Rep. CS TR-3301, University of Maryland, 2009.

11. Boehm, Barry W., Software Engineering Economics, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 2006

12. CRF++: Yet Another CRF Toolkit, http://crfpp.sourceforge.net/, 2010.

13. Overview of Attack Trends, http://www.cert.org/archive/pdf/attack_trends.pdf,2015.