REAL TIME CLASSIFICATION AND CLUSTERING OF IDS ALERTS USING MACHINE LEARNING ALGORITHMS

T. Subbulakshmi[1] , George Mathew[2] ,Dr. S. Mercy Shalinie [3]
[1]*SeniorGrade Lecturer, Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai*
subbulakshmitce@yahoo.com
[2] *Lecturer , Department of Computer Science and Engineering,* Rajagiri Engineering College, *Cochin.*
geomat@gmail.com
[3]*HODCSE, Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai*
shalinie_m@yahoo.com

ABSTRACT

Intrusion Detection Systems (IDS) monitor a secured network for the evidence of malicious activities originating either inside or outside. Upon identifying a suspicious traffic, IDS generates and logs an alert. Unfortunately, most of the alerts generated are either false positive, i.e. benign traffic that has been classified as intrusions, or irrelevant, i.e. attacks that are not successful. The abundance of false positive alerts makes it difficult for the security analyst to find successful attacks and take remedial action. This paper describes a two phase automatic alert classification system to assist the human analyst in identifying the false positives. In the first phase, the alerts collected from one or more sensors are normalized and similar alerts are grouped to form a meta-alert. These meta-alerts are passively verified with an asset database to find out irrelevant alerts. In addition, an optional alert generalization is also performed for root cause analysis and thereby reduces false positives with human interaction. In the second phase, the reduced alerts are labeled and passed to an alert classifier which uses machine learning techniques for building the classification rules. This helps the analyst in automatic classification of the alerts. The system is tested in real environments and found to be effective in reducing the number of alerts as well as false positives dramatically, and thereby reducing the workload of human analyst.

KEYWORDS

Alert Classification, Alert Generalization, Alert Verification, False positives, Intrusion Detection, machine learning

1. INTRODUCTION

The explosive increase in the number of networked computers and sophisticated attack strategies made intrusion detection one of the most challenging fields in computer security. Intrusion Detection Systems (IDS) aim at detecting intrusions, i.e. actions that attempt to compromise the confidentiality, integrity and availability of a computer resource. As the signature based IDS became more popular, its limitations and problems have become apparent.

One of the major problems faced by IDS is huge number of false positive alerts, i.e. alerts that are mistakenly classified normal traffic as security violations. A perfect IDS does not generate false or irrelevant alarms. In practice, signature based IDS found to produce more false alarms than expected. This is because of the overly general signatures and lack of built in verification tool to validate the success of the attack. The huge amount of false positives in the alert log makes the process of taking remedial action for the true positives, i.e. successful attacks, delayed and labor intensive.

Same intrusion event can trigger hundreds of similar alerts. For example, a single network scan may cause to generate several alerts which differ by a small amount of time. These alerts can be fused together before passing to human analyst. Also, different types of alert will be having same underlying event as the root cause. We can generalize each attributes of all alerts to find out the correlated alerts. This will help in the process of root cause analysis and hence eliminate more number of false positives. Alert generalization also helps to speed up alert verification some times. For example, suppose a large number of IIS exploit attack comes to port 80 of a particular machine which is running an Apache web server and Linux, obviously all of these can be marked as irrelevant since they are not successful.

2. RELATED WORKS

Alert aggregation and verification are often part of Alert correlation [4], but it has a different goal, to reconstruct incidents from alerts and to find attack scenarios. From the processed alerts the correlation engine tries to find the incident by reconstructing attack threads and sessions. We have no intension for alert correlation; even then this work can be utilized as a front end for high level alert correlation.

Kruegel, Robertson and Vigna [6] proposed a method for alert verification using both active and passive verification methods. In their implementation they use active alert verification extensively while we only use passive alert verification.

Pietrazek and Tanner [3] propose a two stage alert classification mechanism for reducing false positives. In the filtering stage they use CLARAty for generalization and in classification stage they use RIPPER for building classifier rule base. Our work differs from there work in the first stage, where they lack the process of alert verification. We also have automatic labeling of alerts for second stage for the purpose of learning phase of classifier.

3. SYSTEM ARCHITECTURE

In this paper we describe a two phase alert classification system. Figure 1 gives an overview about this architecture. The first phase preprocesses and normalizes the alerts, fuse them and generalize them for root cause analysis and alert verification. After the first phase, alerts which are marked as false positives can be safely removed or labeled alerts can be passed to second phase.

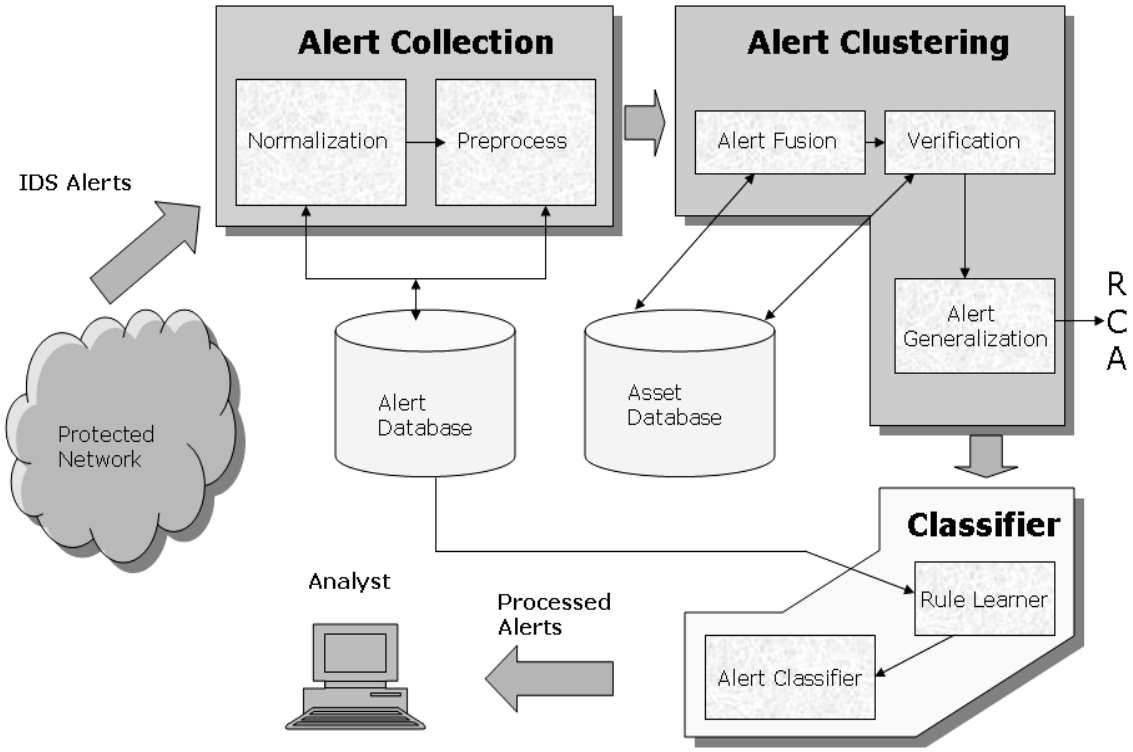

Figure 1: Real-time alert clustering and classification system architecture

In second phase, we make use of machine learning techniques to build a classifier that automatically distinguishes true and false positives. It helps the human analyst by providing an option to discard the false positives it has classified with high classification confidence. The labeled alerts from first phase are used for the purpose of learning. Upon arrival of next batch of alerts, the classifier can be updated in a batch incremental manner, since the classification algorithm we are using, RIPPER, supports incremental learning. The classification rules learned by RIPPER for each batch of alerts are mentored by a human analyst. This ensures the correctness of the classifier.

3.1 Alert Clustering

Alerts generated by one or more IDS can be set to log into a centralized database. If we are using different types of IDS (Application, Network and Host based) the attack messages also will be in different formats. So we need a preprocessing step to be run, preferably in batch mode, before passing into the clustering component. While preprocessing the alert we try to supply best effort values for the missing attributes. Also the timestamp is converted into seconds for the purpose of comparison.

Since different IDS may use different naming conventions for the same event, we need to standardize the messages. For example, the messages ‘scanning’, ‘nmap scan’, ’port scan’ all belongs to the category ‘port scan’. The standard names are chosen either from CVE or Bugtraq

and in some cases names from one of the IDS is taken as standard. In addition, a unique id is also added to every alert for the purpose of tracking the alerts.

Once the alerts are preprocessed and normalized, it is passed to the first phase for the purpose of filtering and labeling. First, alerts with same attributes other than time and which differ only by a small amount of time are fused together for the purpose of alert reduction. This is possible since multiple IDS may be there in the network which produces redundant alerts and same event may cause to trigger hundreds of similar alerts. Alert fusion also makes the process of generalization fast.

For the purpose of generalization of alerts, we need to incorporate hierarchical background knowledge for each attribute. A sample hierarchy is shown in Fig.2. We prefer human understandable descriptions of alert clusters since human intervention may be required for advanced analysis. We carry out generalization as a step by step process. On every iteration, one of the selected attribute is generalized to the next higher level of hierarchy and those alerts which have become similar by this generalization are grouped together. This process is repeated until one of the generalized alerts reach a threshold count. The selection of this threshold is left as a design choice.

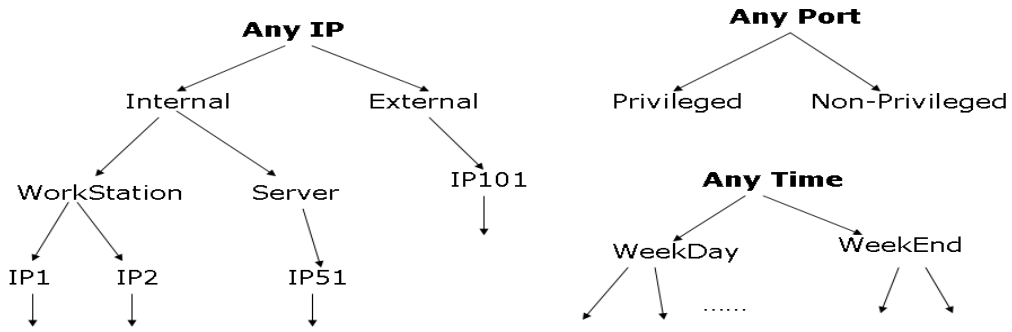

Figure 2: Generalization hierarchies for IP address, port and timestamp

Alert verification is done based on the static asset information collected about the machines inside the network. The process of collecting services and vulnerability information is done with Nessus [9] client. The false positives and irrelevant alerts are marked separately for further analysis. Alert verification is of two types, active and passive. In active verification, whenever an alert is received an information gathering process is initiated to verify the correctness of alert. This method requires more resources and it may slow down the whole alert management process. As an alternative, we have passive alert verification system which depends on a priori gathered information about the host and network. A drawback of passive alert verification is that the information may be redundant. But still, for real-time environments, the performance of passive verification suits well.

After the process of verification each alert is labeled as true positive, false positive, or irrelevant. The human analyst can optionally examine the output of the first phase for advanced

root cause analysis and for updating firewall and IDS rules for avoiding irrelevant and false positive alerts. The labeled alerts are passed to the next phase.

3.2 Alert Classification

Unfortunately, alerts generated by IDS have to be reviewed by a mentor since no rule can assure hundred percent true positive or true negative rates. In the second phase, the labeled alerts from first phase are used for training the automatic classifier which uses RIPPER algorithm for learning the classification rules. The main aim of this phase is to build an automatic alert classifier that reduces the workload of the human analyst.

The analyst examines the rules formed by the classifier and modifies if required. The qualified rules are updated to an Alert filter which classifies the alerts as true and false positives. Some algorithms can give confidence level of classification. In this case, the one with high level of confidence can be safely removed as false positives. The alerts which have been classified as false positive by the human analyst can be considered for training purpose. In addition to training examples, background knowledge is used to learn improved classification rules. These rules are then used by the classifier to classify alerts. The analyst can inspect the rules to make sure they are correct.

4. IMPLEMENTATION DETAILS

We used Snort, an open source light weight intrusion detection system, to detect attacks and generate alerts. The default rule set of Snort is used because we do not have any intension to evaluate the performance of Snort as IDS. We placed separate IDS for wireless and wired networks inside the campus. The alerts generated are logged into a file. Each alert is represented as a seven filed record containing the following attributes: timestamp, signature ID, Source and destination IP addresses, message name and protocol.

The alerts generated for a period of 24 hours have been collected for the purpose of analysis. Alerts from multiple IDS are combined and considered for preprocessing. The preprocessed alerts are then normalized with standard naming conventions as discussed earlier. After normalizing similar alerts are fused together for eliminating redundancy. The alerts are then generalized for root cause analysis and alert verification. The background knowledge for generalization is represented as a tree structure. The whole system was developed in java.

In the second phase we need to add some more information as background knowledge for the purpose of learning. Attribute valued representation of background knowledge [1], which is most suitable for machine learning algorithms, is used in this experiment. Eight more attributes to the alerts generated by snort has been added. The background knowledge sets used are IP address classification and operating system for source and destination IP addresses, Aggregate 1 containing number of alerts in the time window of one minute with same source or destination IP address and total number of alerts in the same time window, Aggregate 2 contains fields same as Aggregate 1 but within the time window of 5 minutes. We used WEKA implementation of RIPPER algorithm in our experiments.

5. RESULTS AND DISCUSSION

The result obtained after the first phase using both real-time dataset and DARPA 1999 intrusion detection evaluation dataset is given in Table 1. For the purpose of comparison, we did not use alert verification and generalization for the DARPA 1999 dataset. Instead we used the truth

table provided by MIT Lincoln laboratory which contain information about the successful attacks. All the other alerts are marked as false positives.

Table 1: Alert statistics after the first phase

	Real-time Dataset	DARPA 1999 Dataset
Preprocessed Alerts	1201	1201
No: of Alerts after Fusion	531	566
Total FP Alerts	525	564
Total TP Alerts	6	2
Base rate	1.13%	0.35%

We can clearly see that alert fusion increases the basic rate of incidence (base rate) of true positive alerts and thereby increases the performance of the rule learner and classifier. It also helps to avoid redundancy and eliminate more than half of the alert after fusing. This introduces performance and time benefits for both the classifier and human analyst. Alert verification marks out most of the common and known false positives by comparing the vulnerability listing of every machine. Alert generalization helps to reduce the number of alerts further to make the process of verification easier for the analyst. In the case of our real-time dataset, 94.83% of total alerts were reduced after the process of generalization. Analyst can use active alert verification with the help of Nessus scanner along with file integrity checkers to find out unknown false positives.

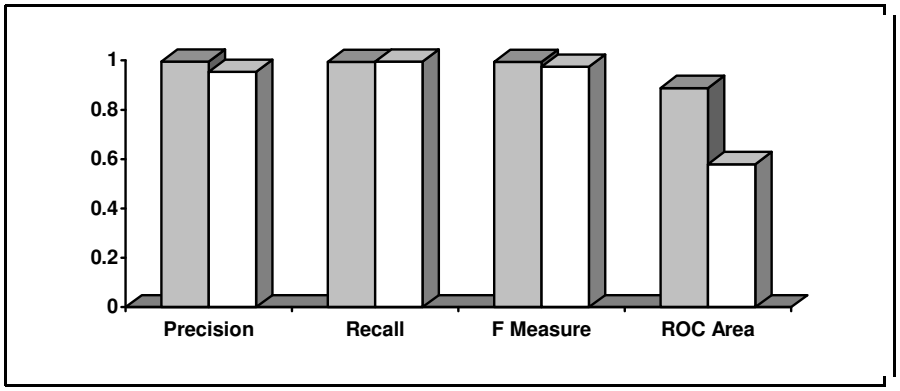

Figure 3: Performance of the classifier using RIPPER algorithm for rule learning

Figure 3 shows the details of classification of both the datasets using RIPPER rule learner. RIPPER rules are easily interpreted and can be modified by the human analyst. From the graphs we can see that RIPPER performs better with real world data rather than standard dataset. The reason for this might be the presence of redundant alerts which are more in the real-time dataset. We have also compared the suitability of different classification algorithms as the rule learner in second phase. Table 2 shows the results obtained for both real-time and DARPA datasets.

Table 2: Comparison of results of classification using various classification schemes

	Precision		Recall		F-Measure		ROC Area	
	Real	DARPA	Real	DARPA	Real	DARPA	Real	DARPA
Random Forest	0.993	0.958	0.992	0.965	0.991	0.961	0.997	0.625
Decision Stump	0.978	0.954	0.989	0.996	0.983	0.975	0.905	0.473
RIPPER	0.991	0.954	0.991	0.996	0.991	0.975	0.889	0.580
NNge	0.993	0.959	0.992	0.954	0.991	0.956	0.667	0.554
oneR	0.993	0.954	0.992	0.993	0.991	0.973	0.667	0.496
PART	0.989	0.955	0.989	0.985	0.989	0.970	0.896	0.668

By analyzing the results, we can see that algorithms like Random Forest perform very well, but the rules created by them are not easy to interpret compared to PART, NNge and RIPPER. The rules created by NNge are not very concise and it need not help the human analyst to build the correct classifier rules every time. RIPPER and PART (a decision tree based rule learner in WEKA) can provide confidence of classification which would be very much useful once the system is deployed to work in automatic mode. We should also consider the algorithms with minimum false negative rates since false negatives are considered more serious than false positives in intrusion detection. Figure 8 gives the average false negative rates of different schemes. We can see that RIPPER has lesser false negative rates compared to other algorithms. Hence we choose RIPPER to be our default rule learner.

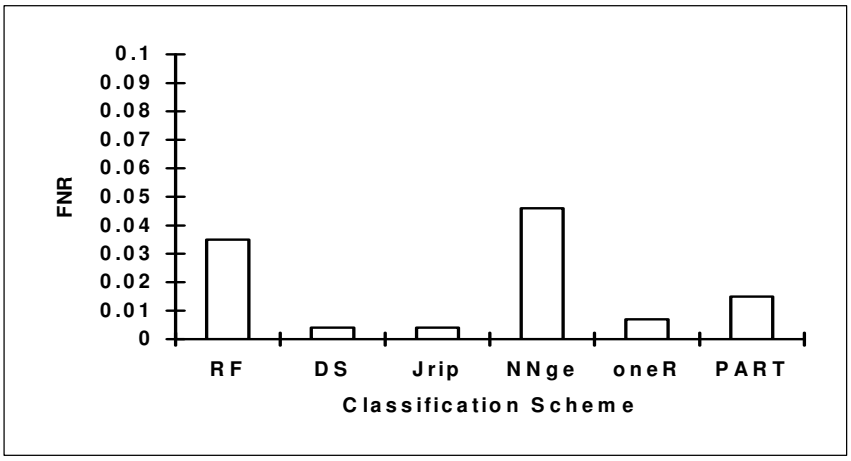

Figure 4: Average False Negative Ratio for different classification schemes

6. CONCLUSION

We have discussed an architecture for a two phase alert management system that assists a human analyst to eliminate false positives as fast as possible. The conventional method of using an alert classifier has the disadvantage of lacking an alert labeling system for real-time data. For the purpose training the classifier, the analyst has to manually verify thousands of alert which can be time consuming and annoying if the training has to be done in regular intervals. We proposed a method for correct labeling of alert with the help of human analyst by alert fusion

and alert generalization, which reduces the workload of the analyst considerably. The system is implemented in real-time and compared with other systems with different classification schemes and datasets. The results show that the system can be deployed safely after sufficient amount of training.

This work may be extended by identifying the most important attributes for intrusion detection using feature selection. Concepts from rough set theory can be used for this purpose. Also, the binary classification can be extended to multi-classes which include the causes of false positives and categories. Another possible extension is to develop a feedback system that modifies the IDS parameters and rule base to eliminate false positive based on the feedback given by the alert management system.

REFERENCES

[1]Tadeusz Pietraszek. "Using adaptive alert classification to reduce false positives in intrusion detection", In Recent Advances in Intrusion Detection (RAID2004), volume 3324 of Lecture Notes in Computer Science, pages 102–124, Sophia Antipolis, France, 2004. Springer-Verlag.

[2]William W. Cohen. "Fast effective rule induction". In Armand Prieditis and Stuart Russell, editors, Proceedings of the 12th International Conference on Machine Learning, pages 115–123, Tahoe City, CA, 1995. Morgan Kaufmann Publishers.

[3] Tadeusz Pietraszek and Axel Tanner. Data mining and machine learning-Towards reducing false positives in intrusion detection. Information Security Technical Report, 10:169–183, 2005

[4]Herv´e Debar, Andreas Wespi, “Aggregation and correlation of intrusion detection alerts”, In Recent Advances in Intrusion Detection (RAID2001), volume 2212 of Lecture Notes in Computer Science, pages 85–103. Springer-Verlag, 2001

[5] Guy Helmer, Johny S.K. Wong, Vasant Honavar, Les Miller, “Automated discovery of concise predictive rules for intrusion detection”, The Journal of Systems and Software, 60(2):165–175, 2002.

[6] Christopher Kruegel, W. Robertson and Giovanni Vigna, “Using alert verification to identify successful intrusion attaempts”, K.G. Saur Verlag, Munchen, 2004

[7] Weka 3: Data Mining Software in Java, Web page at http://www.cs.waikato.ac.nz/ml/weka, 2008

[8] Snort Intrusion Detection and Prevention system, Web page at http://www.snort.org, 2008.

[9] Nessus vulnerability scanner, Web page at http://www.nessus.org
ensemble of soft computing paradigms”, *Third international conference on intelligent systems design and applications advances in soft computing,* Germany: Springer Verlag; pp: 239e48.

Authors

T. Subbulakshmi is working as a Senior Grade Lecturer in department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, Tamilnadu. She has published papers in conferences and Journals. She is currently pursuing Ph. D in the area of Information Security. Her research interests includes information Security and Machine learning algorithms

Dr. S. Mercy Shalinie, is currently heading the Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, Tamilnadu. She has published 50 papers in International Journals. Her research interests include Application of Neuro Fuzzy systems to various research problems.

George Mathew is working as a lecturer in Rajagiri Engineering College, Cochin. He has published papers in Conferences and Journals. His research interests includes Machine Learning and Information Security