http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (4): 798-806, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.798.806

Filtering Events using Clustering in Heterogeneous Security Logs

H.Asif-Iqbal, Nur Izura Udzir, Ramlan Mahmod and Abdul Azim Abd.Ghani
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia,
43400 UPM Serdang, Selangor Darul Ehsan, Malaysia

Abstract: Log files are rich sources of information exhibiting the actions performed during the usage of a computer system in our daily work. In this study we concentrate on parsing/isolating logs from different sources and then clustering the logs using data mining tool (Weka) to filter the unwanted entries in the logs which will greatly help in correlating the events from different logs. Unfortunately parsing heterogeneous logs to extract the attribute values becomes tedious, since every type of log is stored in a proprietary format. We propose a framework that has the ability to parse and isolate a variety of logs, followed by clustering the logs to identify and remove unneeded entries. Experiments involving a range of logs, reveals the fact that clustering has the capacity to group log entries with a higher degree of accuracy, thereby assisting to identify correctly the entries to be removed.

Key words: Multi-level clustering, heterogeneous log parsing, event filtering, false positive rate, false negative rate

INTRODUCTION

Logs are repositories of actions provided by the operating system, applications and devices (Souppaya and Kent, 2006). Logs recorded by diverse entities (OS, application, devices) serve as critical resource in identifying intrusions. A glimpse at the logs manually, conceals the existence of intrusion, but careful examination reveals the existence of intrusion traces. Identifying the traces of problems right from the root may help to design a more complete solution to detect intrusions. Analyzing a single type of log, to generate intrusion alerts results in more number of false positives and false negatives (Abad *et al*., 2003). From this fact it is evident that multiple types of logs need to be considered and also correlating such logs from myriad of resources becomes necessary.

Events related to system and security is logged by all operating systems. Massive records of operations are also maintained by servers. Bringing all these information together will facilitate a proficient attack detection mechanism (Abad *et al*., 2003). Data obtained with a considerable delay, will delay the detection process and hence become useless. Moreover, incomplete data might degrade the detection capabilities (Spafford and Zamboni, 2000). This could be the reason that why some of the intrusion based systems do not provide the functionality it promises.

Every type of log records the information which they deem as the most important in their context, for instance host IP address and username (Souppaya and Kent, 2006). Usability of log data depends on the way that each log is interpreted and normalized into a stanch representation (The MITRE Corporation, 2008). Event correlation is defined by Tiffany (2004) as associating events with one another in useful ways. It is almost impossible to treat every individual event separately. The idea of event correlation is to attempt to isolate larger problems which causes countless symptoms to emerge (Tiffany, 2004). Correlating logs to ascertain intrusion attacks is vital to enhance the effectiveness of the intrusion detection systems. The ability of the intrusion detection systems can be improved by correlating information from different logs, especially regarding the effectiveness in generating alarms (Abad *et al*., 2003). Inconsistent log formats and data field representations is a challenge for log reviewers, which desires a better understanding of various data fields in each log to perform an exhaustive review (Souppaya and Kent, 2006). Besides, correlation makes detection more effective and imparts information needed for future prevention and reaction (Abad *et al*., 2003).

The majority of logging mechanisms available today is not crafted by computer security experts and hence inappropriate for intrusion detection (Barse and Jonsson, 2004). Very little discussion regarding the log mechanism is found in intrusion detection literature. Most of the log related discussions consider the type of data used and leave behind, the technique used to produce such data (Larson and Jonsson, 2006). The two common dilemmas in event logging are the varied log formats and the

Corresponding Author: H.Asif-Iqbal, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor Darul Ehsan, Malaysia Tel: +60-3-8946-6521 Fax: +60- 3-8946-6577

massive number of recorded events. No establishments are available to standardize formats, hence logs are written in a proprietary fashion (Herrerias and Gomez, 2007). The underlying problem is that although the log sources contain all the information for a successful recovery from an intrusion, this information is drowned inside mammoth log entries generated by the legitimate use of the system (Fredrikson *et al*., 2010). Even though the tool Slog designed by Fredrikson *et al*. (2010) provides the user with a simplified view of the events related to the intrusion, the need of manual analysis is still expected. A common restriction in majority of the tools and sources of information is that they blend together the actions of the intruder with that of legitimate users. In spite of the availability of adequate information regarding logs and disk state to empathize an attack, identifying the chain of actions from the initial compromise to the detection point, is still a manual process (King and Chen, 2003), furthermore in multipart datasets, the margin between normal and abnormal events are not apparent (Yu *et al*., 2009).

The vicinity of intrusion event correlation is immense and we do not concentrate on each facet of it. The focus is streamlined towards:

- Collection of different logs from standard datasets available online (Chuvakin, 2009) and also the logs generated especially by lightweight components or applications, without forgetting the essence of the information to be extracted for further process
- Extracting or parsing the data from the logs and store it in a format, to induce the process of segregating the log entries using clustering algorithms
- Reducing unneeded entries in logs to a format that, ease the normalization process, without losing the intrusion context

RELATED WORK

Statistical approaches lean towards the process that generated the data, but machine learning techniques spotlights on building a system that increases its performance based on prior results which means that the systems based on machine learning paradigm has the skill to amend their execution approach on the basis of newly acquired information (Patcha and Park, 2007). The technology of data mining and machine learning have lead many researches to overcome the restriction of signature and anomaly detection with a better detection capacity for unknown attacks and reduce false alarm rate (Muda *et al*., 2010). Grouping and clustering the alerts based on their feature similarities actually can reveal the attack steps launched by the attackers (Siraj *et al*., 2009). A cluster is a group of likely objects sharing some common characteristics. The aim of cluster analysis is the classification of network connections, or objects, according to similarities among them and organizing objects into groups.

Clustering algorithms can aid existing intrusion detection systems with the ability to detect unseen anomalous behavior, where unknown patterns or signatures can be detected (Zhong *et al*., 2007). The nature of the data to be clustered should be given paramount significance in selecting the appropriate algorithm for clustering (Vaarandi, 2003). Clustering spotlights on finding patterns in the input data and clusters instances with similar properties into groups (Nguyen and Armitage, 2008). So clustering algorithms can greatly help in dividing the normal behavior from the anomalous one. Those clusters which are having minimum number of events need more attention so as to identify the anomalies correctly. Clustering provides the ability to learn from and detect intrusions in the audit data, without requiring the system administrator to give clear descriptions of a range of attack classes/types (Patcha and Park, 2007). More over clustering algorithms are unsupervised and they do not need labeled data as input. They have the ability to learn by itself in finding patterns with the provided data.

Chandola *et al*. (2009) categorize the cluster based anomaly detection techniques based on three assumptions. They are: (1) normal events form clusters whereas anomalous events will not form clusters, (2) normal events are closer to centroid, whereas anomalous events are far away from the centroid and (3) normal events form bigger clusters, whereas anomalous event form smaller clusters. The third could be the right assumption for our work and the experiments are continued with that assumption. The event clusters that are bigger are filtered to reduce the unimportant or unneeded events based on the assumption mentioned before. Feedback Learning IPS (Locasto *et al*., 2006) is a mixture approach to host security to prevent binary code injection attacks, comprises of three key mechanisms namely an anomaly-based classifier, a signature-based filtering scheme and a supervision framework that employs Instruction Set Randomization (ISR) to identify injected code and prevent code injection attacks which allows FLIPS to construct signatures for zero-day exploits. This work concentrates only on a specific type of attack especially code injection attacks and the rest of the attacks are not addressed.

The Slog devised by Fredrikson *et al*. (2010) allows users to state the syntactic and semantic information about the events for a given random set of log sources and performs thorough and accurate analysis of the

events. Besides, it offers services to perform semantic abstraction for a series of events gathered from log sources, allows the user to simplify the view of the events. This tool intends to provide a straightforward view of the events and objects related to an intrusion and do not concentrate on generating alarms with respect to an intrusion. Moreover the user has to provide the syntax and semantic description of the log sources.

The majority of the clustering algorithms are designed for general data sets such as market basket data, where no exact assumptions are being made on the nature of data (Vaarandi, 2003). Siraj *et al.* (2009) used an unsupervised clustering algorithm, Expectation Maximization (EM) which has a better performance, in terms of ability to combine redundant alerts and abandon false positives and worthless alerts. The originality of this work is the combination of Improved Unit Range (IUR), Principal Component Analysis (PCA) and unsupervised learning algorithm (Expectation Maximization) as a solution to group intrusion alerts and to clean the unnecessary alerts. Moreover the experiment uses DARPA 2000 Scenario Specific Dataset.

Yen and Lee (2009) implemented a Cluster-based under-sampling approach by which all the training data is clustered first. Every cluster becomes the representative of the class where majority of the sample belongs to. In other words, the cluster having more majority class samples and less minority class samples, acts like the majority class samples. The cluster having more minority class samples and less majority class samples, acts like minority class samples. So the approach selects a certain number of majority class samples and minority class samples from each cluster based on the ratio between them. This approach has a better prediction accuracy and stability than other methods, high classification accuracy on predicting the minority class samples and also has a fast execution time.

Chimphlee *et al.* (2005) implemented a method that combines Rough set and Fuzzy Clustering, where Rough set is to decrease the amount of data and get rid of redundancy and Fuzzy *c*-means clustering allow objects to belong to several clusters simultaneously, with different degrees of membership. The results are very promising in terms of detection accuracy at low false-positive rates when tested on KDD Cup 1999 intrusion detection dataset. Sequential Information Bottleneck (sIB) Clustering (Panda and Patra, 2009) provides better detection accuracy with comparatively low false positive rate in comparison to other existing unsupervised clustering algorithms. The algorithm is able to fix the number of clusters by itself and specifying the number of clusters is not needed and it uses a division of KDD Cup 1999 intrusion detection benchmark dataset.

Zhou *et al.* (2010) use a simple filtering technique whereby repeated events and periodic events are filtered. It uses a simple statistical algorithm and a simple clustering algorithm to filter the events, which results in the number of events in the log decline by 96%. Liang *et al.* (2005) proposed a method called Adaptive Semantic Filtering (ASF) which uses the semantic context of event descriptions to decide on the redundant events. This involves three steps: (1) building a dictionary of keywords that appear in event descriptions, (2) translating event descriptions into a binary vector in which each element of the vector represents the status of the appearance of keyword in the description and (3) filtering events using adaptive semantic correlation thresholds.

Our work uses the clustering algorithm to form natural clusters before the filtering process starts. The filtering process is done based on the assumption that the dense clusters are operational noise and the thin clusters are possibly the cause of an anomalous activity. Moreover, multilevel clustering is used to identify those events, which changes the cluster at different levels, to make the filtering process easier. Most of the related work mentioned above uses a dataset which has been collected from a single source, which means that, the data is collected from a single device or a single application which is already labeled as normal or intrusive. Since our work revolves around correlating the entries from different log sources captured from different applications or devices, the usage of KDD cup dataset is not amicable.

PROPOSED FRAMEWORK

Log management infrastructures usually carry out functions similar to parsing, filtering, aggregating logs without losing the originality of the logs (Souppaya and Kent, 2006). Correlation takes place after collecting, filtering and normalizing the log files (Herrerias and Gomez, 2007). Data mining techniques would be very useful in processing massive number of entries present in the logs (Abad *et al.*, 2003) and Weka is used for this purpose in the proposed framework. The work done by Abad *et al.* (2003) which is the first of this kind of work, exhibits the importance of correlating heterogeneous logs to identify anomalies, but did not suggest a concrete framework to implement the same. It also gives an insight of the different type of logs that can be considered for correlation. The framework proposed by Herrerias and Gomez (2007) collects log files from different sources through agents, stores it in container and filters the log before normalizing the events. Semantic abstraction is the technique introduced by Fredrikson *et al.* (2010) to filter syslog events by condensing linked sequences of events

so as to create an abstracted view of the events that maintains the semantics of the underlying events.

Our framework illustrated in Fig. 1 differs from Herrerias and Gomez (2007) and Fredrikson *et al.* (2010) by using multiple logs from different sources (Zhou *et al.*, 2010) and by specifying the types of log used. The parsing done by Fredrikson *et al.* (2010) is a joint effort of the user and the tool, whereas the tool is fully responsible for parsing logs in our case, but limited only to the logs stated in our framework. Filtering is performed by Herrerias and Gomez (2007) by compressing, counting, suppressing and generalizing events, whereas in our case the filtering process starts initially by clustering the events in individual logs followed by reducing the changing clusters and big clusters from the logs for further process.

The algorithm to implement the proposed framework is as follows:

- **Collecting:** Collect unlabelled heterogeneous logs
- **Parsing:** Parse raw logs individually and isolating log entries if needed
- **Clustering:** Grouping the log entries before filtering
  (a) Finding the ideal number of clusters (K) for every parsed log dataset using EM clustering algorithm
  (b) Run K-Means algorithm on parsed logs using K clusters generated in (a)
  (c) Repeat (b) twice
- **Parsing:** Parse the clustered logs to make it viable for filtering
- **Filtering:** Filter the clustered events
  (a) Filter-in the events which switch the clusters during successive iterations
  (b) Filter-in smaller clusters that satisfies the given thresholds:
  (c) Clusters containing events that are less than or equal to 10% of the total number of events clustered, for that particular dataset
  (d) IF number of events yielded by (a) and (c) is 0 THEN
  Clusters containing events that are less than or equal to the total number of events clustered for that particular dataset divided by number of clusters for that particular dataset and then proceed to aggregating
  ELSE proceed to aggregating
- **Aggregating:** Combining filtered events in which the attribute values are exactly alike and also is recorded in the same minute of time

Every log is individually treated since every log has different number of attributes and also attribute values as well as shown in Fig. 2. So parsing is done separately for each type of log and stored in a separate file. In some cases, the entries belong to the same log is isolated because, the same log contains different attributes for different entries which mandates the same log to be stored in separate files. For example, IP Table syslog entry has varying number of attributes for different protocols (TCP, ICMP, UDP has 24, 22, 21, respectively). In some other cases, the data for a long duration is given in a single large file, which has to be divided into smaller sets for a specific duration to ease further process.

Apache logs are more or less having similar attributes, but the rest of the logs need to be separately parsed since every log has different number of attributes. Parsing a log which has different punctuations for different attributes in the same record is very difficult. Moreover the format of date in different log remains different, which needs to be parsed to a single common format.

Log parsing is a process of extracting data from a log, which could be used as input for another logging process. An example of parsing is reading a text-based log file that holds n comma-separated values in each line and extracting the n values from each line. Parsing is a part of numerous logging functions, like log conversion and log viewing (Souppaya and Kent, 2006). Since every log records different attributes, separate parsers needs to be written to parse the logs to a format, where every attribute in a record is separated by commas. The reason for separating attributes using commas is to ease and expedite the preprocessing by the data mining tool (WEKA). A parser is typically developed using a script language like Perl or Python (Forte, 2004). Zhou *et al.* (2010) use Python to parse about 977,858 events in 4 min 24 sec and each event contains nine attributes and save the logs into the MySQL database. Perl is chosen to write the parser because of its text processing capabilities at a faster pace compared to Python.

The time taken to parse 179752 entries with every entry having 20-24 attribute values is only 27 sec as demonstrated in Fig. 3. The parser also separates entries based on the protocols (TCP, UDP and ICMP) and stores it in 3 different files, which is appropriate for the data mining tool for further training process. If the parsed log file has more entries, it should be divided and made into smaller files to overcome the problems regarding memory constraints posed by the data mining tool (WEKA).

The logs are clustered using unsupervised learning, especially using clustering algorithms because of the reason it can work well on unlabelled data. The clustered output consists of additional attributes such as instance number and cluster number apart from those attributes

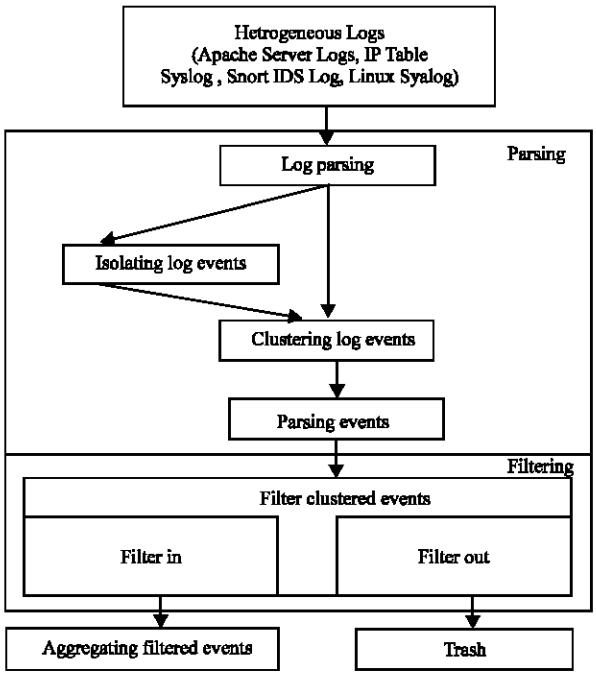

Fig. 1: Architecture of the proposed framework

Apache HTTP Access Log
24.196.254.170- - [06/Mar/2005:05:28:52 -0500] "GET / HTTP/1.1" 403 2898 "-" "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)"
Apache HTTP Error Log
[Sun Mar 06 04:05:53 2005] [notice] Digest: generating secret for digest authentication ...
Apache HTTP SSL Error Log
[Mon Mar 07 00:07:01 2005] [error] Spurious SSL handshake interrupt [Hint: Usually just one of those OpenSSL confusions!?]
Linux Syslog
Mar 6 08:54:47 combo sshd[6396]: Failed password for root from 210.125.27.175 port 1510 ssh2
Snort Logs
Feb 25 12:21:33 bastion snort: [1:483:5] ICMP PING CyberKit 2.2 Windows [Classification: Misc activity] [Priority: 3]: {ICMP} 70.81.243.88 -> 11.11.79.100
Firewall log
Feb 25 12:11:24 bridge kernel: INBOUND TCP: IN=br0 PHYSIN=eth0 OUT=br0

Fig. 2: Sample logs

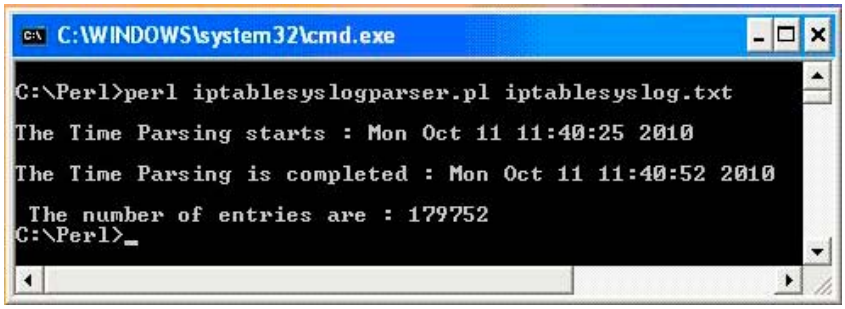

Fig. 3: Time duration for parsing logs

present in a log entry. Additionally it contains information which is recorded in the file as a part of processing. Parsing has to be done on clustered output to make it concise for the filtering process. The big clusters possibly contain the entries of the operational noise that is generated by the legitimate use of the system. The smallest clusters need to be identified for further processing and biggest clusters needs to be filtered. After filtering the unneeded entries from each log, the attributes common in all the logs is found.

EXPERIMENTS AND RESULTS

For the experiments the logs provided by Chuvakin (2009) is used. The experiments were conducted at the Faculty of Computer Science and Information Technology, University Putra Malaysia from 2009 to 2010. The reason for choosing a particular type of log is based on the promising results tabulated by Abad *et al.* (2003) which illustrate the ability of the logs in showing the traces of different types of attacks. Parsing of individual logs was done using Perl. The HTTP server logs were provided in several files, in which each file contains entries for a particular duration. IP Table syslog was provided in a single bunch, but it was isolated and stored in different files after parsing, similar to the duration followed in the previous case, which also applies Linux Syslog. Parsing every group of HTTP server logs consumed less than a second which has approximately 2000 entries. Table 1 provides the type of logs and the number of entries for each type of log used for the experiments. All the logs listed in Table 1 were captured during a specific time (20th to 27th February, 2005).

The attack detection rate of unsupervised algorithms are higher than that of the supervised ones (Gogoi *et al.*, 2010) and since the clustering algorithms are unsupervised, we have chosen to use the most popular and the simplest partitional algorithm K-means (Jain, 2010) which clusters individual logs and the number of clusters to be formed has to be given in advance (Zurutuza *et al.*, 2010). Finding the number of clusters automatically is the most complex dilemmas in data clustering (Jain, 2010), so the number of clusters K for each and every log was determined using Expectation Maximization (EM) algorithm, because it can automatically divide the instances into a definite number of clusters, considered as the best one for the particular dataset (Zurutuza *et al.*, 2007). The number of clusters generated using EM is shown in Table 2.

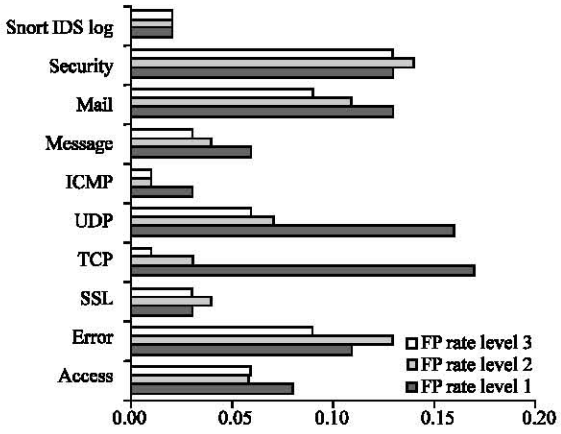

Fig. 4a: False positive rate of multiple levels of clustering

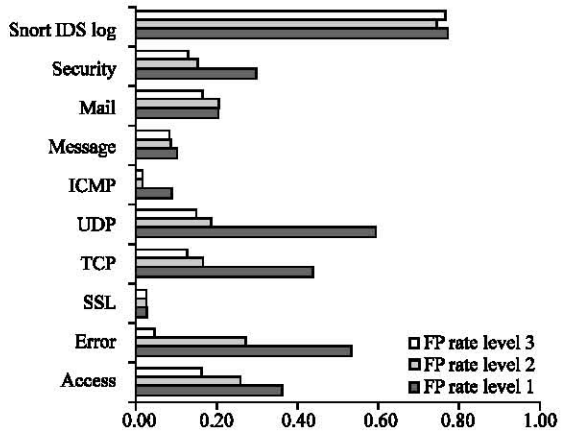

Fig. 4b: False negative rate of multiple levels of clustering

Table 1: Details of dataset used

	Log type	No. of entries
HTTP server logs	Access	260
	Error	461
	SSL error	76
IP table firewall log	TCP	9632
	UDP	613
	ICMP	171
Linux syslog	Message	144
	Mail	57
	Security	148
Snort IDS log		4423

Table 2: Ideal number of clusters using EM

	Log type	No. of clusters
HTTP server logs	Access	7
	Error	4
	SSL error	4
IP table firewall log	TCP	6
	UDP	6
	ICMP	2
Linux syslog	Message	7
	Mail	4
	Security	4
Snort IDS log		12

```
4,22/Feb/2005,11:04:44,error,211.59.0.40,' File does not exist: /var/www/html/
MSADC',cluster0,cluster3,cluster3
5,22/Feb/2005,11:04:49,error,211.59.0.40,' File does not exist: /var/www/html/
c',cluster0,cluster3,cluster3
6,22/Feb/2005,11:04:53,error,211.59.0.40,' File does not exist: /var/www/html/
d',cluster0,cluster3,cluster3
7,22/Feb/2005,11:04:57,error,211.59.0.40,' File does not exist: /var/www/html/
scripts',cluster0,cluster0,cluster0
```

Fig. 5: Cluster changes after multilevel clustering

Table 3: Performance of K means with multiple levels of clustering

	False positive rate			False negative rate		
Log types	First level clustering	Second level clustering	Third level clustering	First level clustering	Second level clustering	Third level clustering
Access	0.08	0.06	0.06	0.37	0.26	0.17
Error	0.11	0.13	0.09	0.54	0.27	0.05
SSL	0.03	0.03	0.03	0.03	0.03	0.03
TCP	0.17	0.03	0.01	0.44	0.17	0.13
UDP	0.16	0.07	0.06	0.60	0.19	0.15
ICMP	0.03	0.01	0.01	0.09	0.02	0.00
Message	0.06	0.04	0.03	0.11	0.09	0.09
Mail	0.13	0.11	0.09	0.21	0.21	0.17
Security	0.13	0.14	0.13	0.30	0.15	0.13
Snort IDS	0.02	0.02	0.02	0.78	0.75	0.77

Table 4: Filtered and aggregated events

			Events in small clusters					
Log type	Total events	No. of events changing clusters (5.1)	5.2.1	5.2.2	Total events filtered-in	Percentage of events filtered-in	Events after aggeration 6	Percentage of events after aggretation
Access	260	0	67	65	67	25.77	33	12.69
Error	461	77	10	59	87	18.87	64	13.88
SSL	76	0	6	0	6	7.89	6	7.89
TCP	9632	4633	2023	0	6656	69.10	6632	68.85
UDP	613	0	110	0	110	17.94	110	17.94
ICMP	171	0	0	75	75	43.86	75	43.86
Message	144	0	34	20	34	23.61	34	23.61
Mail	57	0	0	27	27	47.37	27	47.37
Security	148	0	13	46	13	8.78	13	8.78
Snort IDS	4423	0	1760	0	1760	39.79	1655	37.42

The K value thus generated was used for forming clusters using K means. So every log will form different number of clusters and the resulting clusters were stored in ARFF (Attribute-Relation File Format) which is ideal for further experiments. The reason for preferring K means over EM for the experiments is because of its ability in handling large datasets with less processing time. Hence only the cluster number generated by EM is used to cluster the raw dataset again using simple K means. The clustered output generated by the Weka explorer is given as input for the experimenter. The number of clusters (K) used for all the logs for the experiments is same as stated in Table 2. The number of folds and the iterations are set to 10 for all the experiments to allow every part of the log to be used alternatively for training and testing to produce more accurate results.

The results of the experiments using K-Means as illustrated in Table 3 and Fig. 4a-b reveals the fact that the false positive and false negative rate decreases when the clusters are clustered again. Moreover the true positive and true negative rate increases after re-clustering, which means that the accuracy of the clustering increases. The clustering is done until the third level, because the FN and FP rate becomes stabilized without significant rate changes. Another interesting fact noticed is that, some of the entries which were grouped in cluster X in the first level clustering changes to cluster Y in the second level. The samples depicting the cluster changes are given in Fig. 5.

An entry changes from one cluster to another cluster in the next iteration was because after clustering for the first time every entry is given an extra attribute specifying the cluster which it belongs to. And this helps the clustering algorithm to better decide the cluster which the entry should be in. Moreover the above cluster reveals the fact that, those entries that move to a different cluster

in the next level is a sign of abnormal behavior apart from the small clusters and needs to be considered for correlating with other types of logs.

Based on the algorithm proposed earlier, individual logs are filtered to reduce the number of unwanted events which may be operational noise. Table 4 illustrates the number of events filtered as per the algorithmic criteria followed with the aggregation of filtered events. The filtering result shows that an average of 30% of the events is filtered-in for further processing, which can be considered significant. Events thus filtered are sanitized and then aggregated to remove duplicate entries which are recorded in the same minute of time which reduces another 2% of the events in average.

CONCLUSION

The experimental findings encourages advancing the work to test various clustering algorithms with different datasets based on our proposed framework. Since the goal of our work is to reduce the false positives and false negatives and the results shows the direction that clustering can help to reduce false positives and false negatives. More experiments needs to be conducted on the clustering algorithms to verify the efficiency for different parameters. After clustering individual logs, the redundant entries were filtered, leaving the candidate entries. Filtering criteria proposed needs to be polished and new criteria has to be framed to reduce the number events further. Besides, aggregation has to be fine-tuned to combine events having inconsequential difference in attribute values. The common fields available in different logs, is identified for further correlation which will help to combine the filtered events from different logs into a single set for clustering again.

REFERENCES

Abad, C., J. Taylor, C. Sengul, W. Yurick, Y. Zhou and K.E. Rowe, 2003. Log correlation for intrusion detection: A proof of concept. Proceedings of the 19th Annual Computer Security Applications Conference, Dec. 8-12, IEEE Computer Society, Las Vegas, USA., pp: 255-265.

Barse, E.L. and E. Jonsson, 2004. Extracting attack manifestations to determine log data requirements for intrusion detection. Proceedings of the 20th Annual Computer Security Applications Conference, Dec. 06-10, IEEE Computer Society, Tucson, Arizona, pp: 158-167.

Chandola, V., A. Banerjee and V. Kumar, 2009. Anomaly detection: A survey. ACM Computing Survey, 41: 58-58.

Chimphlee, W., A.H. Abdullah, M.N.M. Sap, S. Chimphlee and S. Srinoy, 2005. Unsupervised clustering methods for identifying rare events in anomaly detection. Proceedings of the 6th International Enformatika Conference, Aug. 26-28, Enformatika, Canakkale, Turkey, pp: 26-28.

Chuvakin, A., 2009. Public security log sharing site. http://log-sharing.dreamhosters.com/.

Forte, D.V., 2004. The art of log correlation: tools and techniques for correlating events and log files. Computer Fraud Security, 2004: 7-11.

Fredrikson, M., M. Christodorescu, J. Giffin and S. Jhas, 2010. A declarative framework for intrusion analysis. Adv. Inform. Security, 46: 179-200.

Gogoi, P., B. Borah and D.K. Bhattacharyya, 2010. Anomaly detection analysis of intrusion data using supervised and unsupervised approach. J. Convergence Inform. Technol., 5: 95-110.

Herrerias, J. and R. Gomez, 2007. A log correlation model to support the evidence search process in a forensic investigation. Proceedings of the 2nd International Workshop on Systematic Approaches to Digital Forensic Engineering, April 10-12, IEEE Computer Society, Seattle, Washington, USA., pp: 31-42.

Jain, A.K., 2010. Data clustering: 50 years beyond K-means. Pattern Recognition Lett., 31: 651-666.

King, S.T. and P.M. Chen, 2003. Backtracking intrusions. Proceedings of the 19th ACM Symposium on Operating Systems Principles, Oct. 19-22, ACM, Bolton Landing, New York, USA., pp: 223-236.

Larson, U.E. and E. Jonsson, 2006. An intrusion detection-centric taxonomy and survey of data log mechanisms. Technical Report 06-07, Chalmers University of Technology, Goteborg.

Liang, Y., Y. Zhang, H. Xiong and R. Sahoo, 2005. Filtering failure logs for a BlueGene/L prototype. Proceedings of the International Conference on Dependable Systems and Networks, June 8-July 1, IEEE Computer Society, Yokohama, Japan, pp: 476-485.

Locasto, M.E., K. Wang, A.D. Keromytis and S.J. Stolfo, 2006. FLIPS: Hybrid Adaptive Intrusion Prevention. In: Lecture Notes in Computer Science, Valdes, A. and D. Zamboni (Eds.). Springer Verlag, Berlin, Heidelberg, pp: 82-101.

Muda, Z., W. Yassin, M.N. Sulaiman and N.I. Udzir, 2010. A K-means and naive bayes learning approach for better intrusion detection. Inform. Technol. J., 10: 648-655.

Nguyen, T.T.T. and G. Armitage, 2008. A survey of techniques for internet traffic classification using machine learning. IEEE Commun. Surveys Tutorials, 10: 56-76.

Panda, M. and M.R. Patra, 2009. A novel classification via clustering method for anomaly based network intrusion detection system. Int. J. Recent Trends Eng., 2: 1-6.

Patcha, A. and J. Park, 2007. An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput. Networks, 51: 3448-3470.

Siraj, M.M., M.A. Maarof and S.Z.M. Hashim, 2009. Intelligent alert clustering model for network intrusion analysis. Int. J. Adv. Soft Computing Appl., 1: 1-16.

Souppaya, M. and K. Kent, 2006. Guide to computer security log management. White Paper, NIST Special Publication 800-92, Computer Security, http://permanent.access.gpo.gov/lps69969/LPS69969.pdf.

Spafford, E.H. and D. Zamboni, 2000. Data collection mechanisms for intrusion detection systems. CERIAS Technical Report 2000-08, Purdue University. https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2000-08.pdf.

The MITRE Corporation, 2008. Common event expression. White Paper. http://cee.mitre.org/docs/Common_Event_Expression_White_Paper_June_2008.pdf.

Tiffany, M., 2004. A survey of event correlation techniques and related topics. http://www.cc.gatech.edu/fac/Russell.Clark/papers/tiffany-netman.html.

Vaarandi, R., 2003. A data clustering algorithm for mining patterns from event logs. Proceedings of the 2003 IEEE Workshop on IP Operations and Management, Oct. 1-3, IEEE, pp: 119-126.

Yen, S.J. and Y.S. Lee, 2009. Cluster-based under-sampling approaches for imbalanced data distributions. Expert Syst. Appl. Int. J., 36: 5718-5727.

Yu, X., L.A. Tang and J. Han, 2009. Filtering and refinement: A two-stage approach for efficient and effective anomaly detection. Proceedings of th 9th IEEE International Conference on Data Mining, Dec. 6-9, IEEE Computer Society, Miami, USA., pp: 617-626.

Zhong, S., T.M. Khoshgoftaar and N. Seliya, 2007. Clustering-based network intrusion detection. Int. J. Reliability Qual. Safety Eng., 14: 169-187.

Zhou, W., J. Zhan, D. Meng, D. Xu and Z. Zhang, 2010. LogMaster: mining event correlations in logs of largescale cluster systems. http://arxiv.org/abs/1003.0951.

Zurutuza, U., R. Basagoiti and A. Aztiria, 2010. Behavior analysis of domain servers through windows security event log mining. J. Inform. Assurance Security, 5: 418-425.

Zurutuza, U., R. Uribeetxeberria, E. Azketa, G. Gil, J. Lizarraga and M. Fernandez, 2007. Combined data mining approach for intrusion detection, secrypt. Proceedings of the International Conference on Security and Cryptography, June 28-31, INSTICC Press, Barcelona, Spain, pp: 67-73.