Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 1117 – 1134
© TÜBİTAK
doi:10.3906/elk-1302-19

An unsupervised heterogeneous log-based framework for anomaly detection

Asif Iqbal HAJAMYDEEN*, **Nur Izura UDZIR, Ramlan MAHMOD,**
Abdul Azim ABDUL GHANI
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang,
Selangor Darul Ehsan, Malaysia

Received: 03.02.2013 • **Accepted/Published Online:** 14.02.2014 • **Final Version:** 23.03.2016

Abstract: Log analysis is a method to identify intrusions at the host or network level by scrutinizing the log events recorded by the operating systems, applications, and devices. Most work contemplates a single type of log for analysis, leading to an unclear picture of the situation and difficulty in deciding the existence of an intrusion. Moreover, most existing detection methods are knowledge-dependent, i.e. using either the characteristics of an anomaly or the baseline of normal traffic behavior, which limits the detection process to only anomalies based on the acquired knowledge. To discover a wide range of anomalies by scrutinizing various logs, this paper presents a new unsupervised framework, UHAD, which uses a two-step strategy to cluster the log events and then uses a filtering threshold to reduce the volume of events for analysis. The events from heterogeneous logs are assembled together into a common format and are analyzed based on their features to identify anomalies. Clustering accuracy of K-means, expectation-maximization, and farthest first were compared and the impact of clustering was captured in all the subsequent phases. Even though log events pass through several phases in UHAD before being concluded as anomalous, experiments have shown that the selection of the clustering algorithm and the filtering threshold significantly influences the decision. The framework detected the majority of anomalies by relating the events from heterogeneous logs. Specifically, the usage of K-means and expectation-maximization supported the framework to detect an average of 87.26% and 85.24% anomalous events respectively with various subsets.

Key words: Unsupervised anomaly detection, heterogeneous log, feature analysis, filtering threshold, generic format log

1. Introduction

An intrusion detection system (IDS) examines computer systems for intrusive activities that violate the established security model [1]. Signature-based detection detects only known attacks based on recognized patterns, whereas anomaly-based detection is able to detect unknown attacks, i.e. traffic that differs from the baseline (normal) behavior, but it suffers from a high false alarm rate. Most unsupervised anomaly detectors use clustering, an unsupervised learning algorithm [1,2–7] that finds instances in an unlabeled dataset with strange properties or models with unforeseen behaviors.

Examining a single type of log results in more false positives and false negatives, whereas correlating events from several logs facilitates in exposing most attacks [8]. Every computer network environment is distinctive and has various prospective data sources for security investigation [9]. Therefore, the ability of an anomaly detector to handle multiple log sources will help to detect a wide range of intrusions.

*Correspondence: hasifiqbal@yahoo.com

As we use multiple logs for anomaly detection, it is impractical to learn and maintain a precise and up-to-date traffic profile for every such log. Moreover, most approaches available today are knowledge-based, i.e. using signatures or learned traffic models, which restricts the detector to only known anomalies. To overcome these limitations, the process of anomaly detection should be devised in such a way that it must be able to detect intrusions from fresh data by learning from it, rather than building a model to detect intrusions in the future traffic data.

To address these problems, we propose a new unsupervised anomaly detection framework called UHAD, which scrutinizes heterogeneous logs without using a trained model on traffic behavior or the knowledge of the anomalies, and uses a two-step strategy in clustering to segregate normal and abnormal events. A new algorithm for filtering is introduced, whereby the filtering threshold is calculated based on the volume of log events and the number of log event clusters. To collect the events from multiple logs into a common setup, we frame a generic format comprising important features that are discriminating in detecting anomalies. New algorithms are introduced to analyze the events to discover anomalous events.

The rest of this paper is structured as follows: in Section 2, we introduce related work on unsupervised anomaly detection methods using clustering and also highlight the variations of our framework from the others. Section 3 describes the components of UHAD and the metrics used to measure the performance of these components. The results produced by the framework components are discussed in Section 4. Finally, conclusions and future work are presented in Section 5.

2. Related work

The importance of using multiple logs for intrusion detection was introduced and emphasized by Abad et al. [8]. They correlated system calls with network logs to detect a specific anomaly, i.e. the Yaha virus. Synthetic data were used to train and test the model. Detection accuracy increased with log correlation, but the “predict the next system call” method underperformed for this problem. UCLog [10], a unified logging architecture, correlates events from various logs for intrusion detection and was further extended as UCLog+ [9], a security data management system to correlate raw data, alerts, and incidents. Alerts and incidents were stored in a centralized database and were queried to perform security analysis, which was then related to raw logs stored at a remote location. Correlations were focused on timestamps and host IP addresses to present the perspective of attack trends on the network. These works on correlating logs inspired us to use multiple logs for anomaly detection.

An unsupervised anomaly detection model (training and testing) [6] compared five clustering algorithms to select the best based on detection accuracy. Even though this model resulted in higher accuracy in detecting unknown intrusions than the signature-based detection model, it was not feasible for real-time detection due to the high false positives. This is the common limitation in using trained models when testing with fresh data. To overcome this limitation, we use clustering algorithms only to group the log events that need to be examined for detecting anomalies. This also avoids the need of having two different datasets for training and testing. An unsupervised anomaly detection method [11] consisting of three steps in the training phase, namely filtering, clustering, and modeling, initially filters out the attack data to retain normal data for clustering, and the clusters generated were modeled using a one-class SVM to form a hypersphere. The data points that fell inside the hypersphere were considered as normal. In the testing phase, those data points in the test data that fell outside this hypersphere were concluded as an attack. Although clustering and filtering are used in UHAD, it differs from [11] in the order in which these functions are applied to the log data and the strategy followed in implementing these functions. Unlike [11], all the events in a particular log are used for clustering to enable filtering to identify the sparse clusters to be filtered in for further processing. Therefore, no models were built to classify fresh data and no adjustments were made to the amount of attack events available in the logs.

A fully unsupervised attack detection approach was proposed by Casas et al. [7] that uses parallel multiclustering uniting the concept of density-based clustering, subspace clustering, and evidence accumulation (SSC-EA) to overcome common clustering problems. The performance of the method was measured by computational time, which increases when the number of features in the events clustered increases. This approach detects network attacks without relying on labeled data or training, and our framework also has a similar knowledge-independent approach to detect successful and unsuccessful intrusions. In our case, the existing iterative clustering methods, i.e. K-means, expectation-maximization (EM), and farthest first (FF), were tested with different parametric settings to recognize their appropriateness in terms of accuracy in grouping log events. The clustering problems were handled by calculating the initialization parameters based on log events clustered and the suitability of the parameters was tested with different settings to evaluate the accuracy in producing better clusters. Unlike [7,11], clustering was used twice at different points in UHAD to classify the events: 1) to assist the filterer to identify the abnormal events and 2) to support the analysis method in detecting anomalies.

The clustering and filtering strategies used in UHAD were initially proposed in [12], but they are implemented differently in UHAD. The first step in the clustering strategy is similar to that of [12], but in the second step, the logs are clustered only once in UHAD since the volume of events that changes clusters in successive iterations is considerably less with inconsequential detection. Instead, the logs are clustered with various initial parameters in order to evaluate the role of seed and K in forming precise clusters. The filtering rules formulated in [12] were refined in UHAD to adapt to the changes made in clustering and also to avoid hardcoding the filtering threshold. Therefore, the filtering threshold was automatically calculated from the clustered events in UHAD. As we think that clustering alone cannot serve as a solution for anomaly detection, several functions have been proposed in UHAD to analyze the clustered events as detailed in Section 3.

3. UHAD

The goal of the framework is to discover anomalous events from heterogeneous logs by identifying the relationships between them, without knowing the characteristics of the anomaly or the model of network traffic behavior. It primarily relies on the pattern of log events and its features to detect anomalies, making it applicable to the evolving network traffic environment. We made two assumptions about the datasets used, which are: 1) the dataset received from the logging sources has recorded every action and is free from modification(s) or fabrication(s) by the adversary, and 2) the majority of the recorded events in logs are normal traffic with a minority percentage of malicious traffic.

We used Scan of the Month #34 (SOTM#34; http://old.honeynet.org/scans/scan34/) from the Honeynet project, which has been analyzed and validated by multiple contestants (http://old.honeynet.org/scans /scan34/sols/3/sotm, http://old.honeynet.org/scans/scan34/sols/2/proc.pdf, http://www.honeynet.org/scans/ scan34/sols/1/index.html) of the challenge, and also examined in [13,14]. The dataset includes evidence from several logs, i.e. Apache Server, Linux syslogs, Snort NIDS, and iptables firewall, which were recorded by different sources in a Honeynet system. To achieve better confidence of the framework in identifying anomalies, four subsets covering different durations were tested, which exhibit different anomalies.

3.1. Log preprocessing

A custom written parser was used to extract the data from the logs, which is incorporated with additional functions, i.e. Isolator and Timestamp Synchronizer. To facilitate further processing, the isolator separated the events of the iptables firewall log according to connections (TCP, UDP, and ICMP) in separate files, since the number of features in the events of this log was not even. The Apache server and Linux syslog events differed by 287 min against the iptables firewall log [14] and therefore the synchronizer adjusted the Apache server logs and Linux syslogs for the difference towards the iptables firewall log. After synchronizing the time, some of the log events were moved to the respective subsets to match the time duration covered by each subset. The preprocessed logs were maintained in comma-separated values (CSV) files. Feature selection [15] was used to eliminate insignificant features and boost the precision in predicting and classifying abnormal events, and so Weka's [16] *CfsSubsetEval* and *BestFirst* search method was used to implement this step. Feature selection was performed by setting every feature in the log as class attributes to know the relationship of the set feature with other features, which will assist in precisely identifying the important features. The features selected from each log used for further processing are provided in Table 1.

Table 1. Features selected by *CfsSubsetEval*.

Log type	Selected features
Access	Date, Time, IPClient, ClientRequestLine, StatusCode, ObjectSize, Agent
Error	Date, Time, Severity, ClientIP, Msg
SSL_Error	Date, Time, Severity, Msg
TCP	Date, Time, Direction, PHYSIN, PHYSOUT, LEN, TOS, PREC, TTL, ID, DF, SPT, DPT, WINDOW, RES, STATUS, URGP, SRC, DST
UDP	Date, Time, Direction, PHYSIN, PHYSOUT, LEN, TOS, PREC, TTL, ID, DF, SPT, DPT, LEN, SRC, DST
ICMP	Date, Time, LEN, TOS, PREC, TTL, ID, DF, id, seq, SRC, DST
Message	Date, Time, Daemon, PID, Operation, User, Tty, UID, EUID, Remotehost, Systemmessage
Mail	Date, Time, From, To, Daemon, Mailer, Stat, Priority, Protocol, Message_ID, Relay, Control_address, DSN, Queue_ID, Messages_queued, Messages_delivered, Bytes_queued, Bytes_delivered
Security	Date, Time, Daemon, PID, Operation, User, Source, Systemmessage
Snort IDS	Date, Time, Rulenumber, Rule, Classification, Priority, Protocol, SourceIP, Sourceport, DestinationIP, Destinationport

Keeping the information in one universal format by bringing all events in diverse logs together facilitates the analysis process in identifying most of the anomalies. However, including every feature in the logs will increase the schema size, making it difficult to manage. To analyze the events collectively, a generic format (GF) was framed with common and important features that were available in the logs, which were also used in [17–20], i.e. timestamp, source IP, destination IP, source port, destination port, and protocol. Additionally, Message was also included in the GF as it expresses the action performed by the event and the patterns were entirely different for benign and malignant events. The framed GF specifies the features not only for GFL but also the features from various logs that can fit into it. The GFL features and the corresponding features chosen from various logs that fit in GFL are tabulated in Table 2. The reader should note that the selection of features for GFL and the relevant features picked from various logs was based on the logs considered and it may vary with different logs.

Table 2. GFL features.

Generic format (GF)	Date	Time	Source IP	Destination IP	Source Port	Destination Port	Protocol	Message
Access	Date	Time	IP client					Client Request Line
Error	Date	Time	Client IP					Msg
SSL_Error	Date	Time						Msg
TCP	Date	Time	SRC	DST	SPT	DPT	Protocol	STATUS
UDP	Date	Time	SRC	DST	SPT	DPT	Protocol	
ICMP	Date	Time	SRC	DST			Protocol	
Message	Date	time						
Mail	Date	time	Remote host					System message
Security	Date	time	Source		Port			System message
Snort IDS	Date	time	Source IP	Destination IP	Source port	Destination port	Protocol	Rule

3.2. Architecture of UHAD

UHAD consists of several cooperating components to identify anomalies. Figures 1a and 1b depict the framework: Figure 1a shows the flow of the process before framing the generic format log (GFL), and Figure 1b demonstrates the procedure of detecting anomalous events based on GFL. Only Component (1) in Figure 1a and Component (5) in Figure 1b use existing algorithms available in Weka, while the other components are newly proposed algorithms written in Perl.

UHAD defines a configuration C that considers logs from several sources S:

$$C = \{ S_1, S_2, \ldots, S_m \}. \tag{1}$$

Each log source provides one or more logs:

$$S_i = \{ L_{i1}, L_{i2}, \ldots, L_{in} \}, \; i \in [1, m], \tag{2}$$

where i specifies the source a particular log belongs to.

Every log event (e) in L_{ij} is extracted to a set of events $E_{ij,}$ and every individual event in E_{ij} is denoted by e_{ijo}:

$$L_{ij} \rightarrow E_{ij} = \{ e_{ij1}, e_{ij2}, \ldots, e_{ijo} \}, \; i \in [1, m], \; j \, [1, n], \tag{3}$$

where i and j indicate the source and log that an event belongs to, and each e_{ij} is composed of a series of features F.

$$E_{ij} \, or \, e_{ij} = \{ F_{ij1}, F_{ij2}, \ldots, F_{ijp} \}, \; i \in [1, m], \; j \, [1, n] \tag{4}$$

Every log (E_{ij}) is subject to feature selection and the resulting features are denoted by:

$$E_{ij} \, or \, e_{ij} = \{ f_{ij1}, f_{ij2}, \ldots, f_{ijp} \}, \; i \in [1, m], \; j \, [1, n]. \tag{5}$$

Log events E_{ij} can be grouped into a specific number of clusters K_{ij}, and the actual number of clusters formed is specified by k_{ij}:

$$E'_{ij} = \{ Ek_{ij1}, Ek_{ij2}, \ldots, Ek_{ijn} \}, \tag{6}$$

where k_{ijn} refers to the number of events in cluster n and Ek_{ijn} is used to specify a set of events belonging to a particular cluster n.

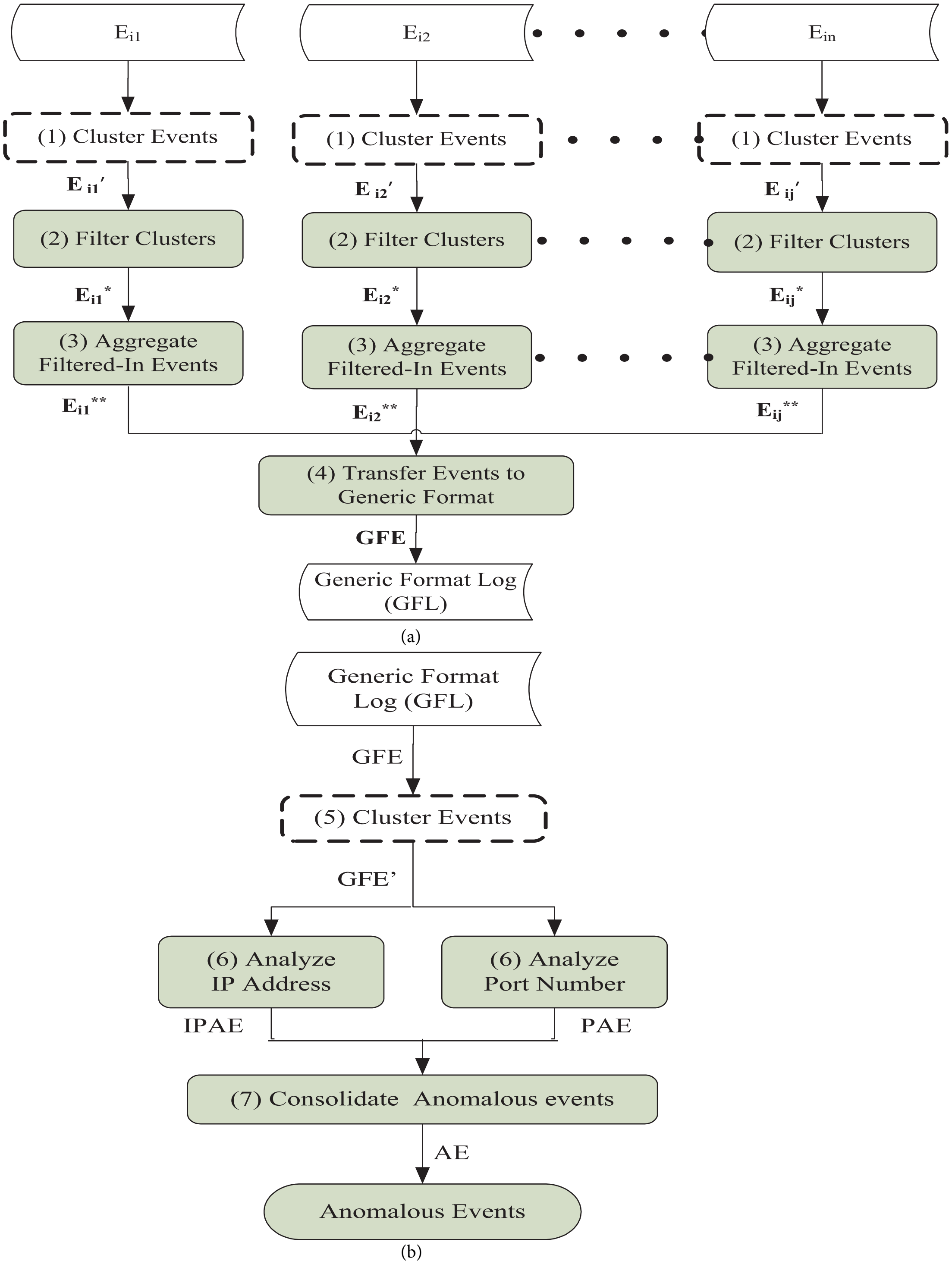

Figure 1. a) UHAD - Event reduction and integration. b) UHAD - Anomaly detection from GFL.

E'_{ij}: set of clustered events, E^{*}_{ij}: set of filtered events, and E^{**}_{ij}: set of aggregated events.

We define a GFL consisting of a set of events referred to as generic format events (GFE) with specific features extracted from E_{ij}^{**}.

$$GFL \rightarrow GFE \leftarrow \left\{ {E_{11}}^{**}, {E_{12}}^{**}, \ldots, {E_{ij}}^{**} \right\}, i \in [1, m], j \in [1, n] \tag{7}$$

GFE consists of a set of events, where GFe_q represents every single event in *GFE.*

$$GFE = \{ GFe_1, GFe_2, \ldots, GFe_q \} \; or \, GFe_q \in GFE \tag{8}$$

GFe_q is composed of eight principal features:

$$GFe_q = \{ f_{da} f_{ti}, f_{sip}, f_{dip}, f_{sp}, f_{dp}, f_{pr}, f_{me} \}, \tag{9}$$

where f_{da}, f_{ti}, f_{sip}, f_{dip},f_{sp},f_{dp}, f_{pr}, and f_{me} refer to date, time, source ip, destination ip, source port, destination port, protocol, and message, respectively. GFE can be grouped into a specific number of clusters K_g, and the actual number of clusters formed is specified by k_g.

$$GFE' = \{ GFEk_1 GFEk_2, ..., GFEk_n \}, \tag{10}$$

where k_n refers to the number of events in cluster n, $GFEk_n$ is used to specify a set of events belonging to a particular cluster n, and *GFE'* represents all the clustered events.

The set of anomalous events identified by IP Address Analysis (IPAE) is composed of several events and is defined as:

$$IPAE = \{ IPAe_1 IPAe_2, \ldots, IPAe_x \}, \tag{11}$$

where $IPAe_x$ refers to individual events in *IPAE.*

The set of anomalous events identified by port analysis (PAE) is composed of several events and is defined as:

$$PAE = \{ PAe_1 PAe_2, \ldots, PAe_y \}, \tag{12}$$

where PAe_y refers to individual events in *PAE.*

The set of anomalous events in *IPAE* and *PAE* is consolidated to produce a set of anomalous events (*AE*), and is composed of several events defined as:

$$AE = \{ Ae_1 Ae_2, \ldots, Ae_z \}, \tag{13}$$

where Ae_z refers to individual events in *AE.*

3.3. UHAD components

The implementation of individual components (as labeled in Figures 1a and 1b) of the anomaly detection framework is detailed as follows.

3.3.1. Clustering events [Component (1)]

This process separates abnormal events from the normal ones for various logs. In our case, the normal events must be identified and reduced prior to analysis to decrease processing overhead. The clustering algorithms used require the number of clusters (K) to group the events. As we are using multiple logs, clustering every log with different clusters (K) to choose the best cluster is time-consuming. Since we want to separate different event patterns in separate clusters, K was calculated based on the patterns in the log. An arrangement consisting of two steps was used for clustering which are: 1) predicting the best number of clusters (K_{ij}) for the given E_{ij}, and 2) clustering E_{ij} using the manipulated K_{ij} [12]. Applying the clustering algorithm separately for different services improves the detection quality [5], and therefore every single log was treated separately for this step.

EM clustering [21] has the ability to find the ideal number of clusters [16] using cross-validation, and therefore EM was used to predict the best number of clusters in step 1, as every cluster relates to a pattern. Apart from that, EM can also cluster when the number of clusters (K) is provided. There are many clustering algorithms and every algorithm produces clusters differently for the same dataset, making it hard to decide an appropriate algorithm for a particular context. K-means [22] treats all features equally [23] and FF [24] is a fast, simple, approximate [16], hierarchical, and distance-based clustering using a distance measurement similar to K-means [25]. The calculation of cluster centroids in successive iterations by FF is opposite to K-means, and hence we chose to test both in this step. Even though EM is slower in clustering compared to K-means and FF, it was opted to test its clustering accuracy. Detail descriptions of these algorithms are available in [21,22,24]. To assess the accuracy of clustering algorithms in grouping events, the Weka implementation of K-means, EM, and FF were used.

Clustering algorithms use K in forming the number of clusters and seed value to decide initial cluster centers. Even the same algorithm generates different clusters with different parameter initializations, which led to difficulty in choosing the appropriate parameters. To capture the impact of K and *seed* in clustering, the algorithms were tested with four different parameter settings. The default seed value of K-means, EM, and FF are 10, 100, and 1, respectively. The settings used are as follows:

Setting-1 (S1): K_{ij} from step 1 of the clustering strategy with the default seed of the respective clustering algorithm.

Setting-2 (S2): K_{ij} from step 1 of the clustering strategy with the seed value set to the number of events in the log (L_{ij}).

Setting-3 (S3): Doubling up K_{ij} from step 1 of the clustering strategy with the default seed of the respective clustering algorithm.

Setting-4 (S4): Doubling up K_{ij} from step 1 of the clustering strategy with the seed value set to the number of events in the log (L_{ij}).

The quality of clusters produced by these algorithms with the respective parameter settings was calculated using the Weka Experimenter with 10-fold cross-validation and 10 iterations to allow every part of the log to be tested. True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) were measured to calculate the accuracy and false positive rate (FPR) of clustering, i.e. Accuracy = (TP + TN) / (FP + FN + TP + TN), FPR = FP / (FP + TN). Apart from that, the clusters generated were examined to evaluate the precision of the algorithms in separating abnormal events.

3.3.2. Filtering clustered events [Component (2)]

The goal of this step is to remove the normal events (noise) while retaining the abnormal events for further processing. The threshold (E_t) for identifying sparse clusters was calculated based on the number of events and clusters for a particular log, i.e. $E_t = E_{ij} / k_{ij}$. Hence, the characteristics of anomalies were not required for filtering events. The entire step was automated using a script written in Perl that receives a clustered file in ARFF format as input and produces the filtered-in events in CSV format. The performance of this step was evaluated by the number of events reduced and the number of abnormal events retained for further processing.

3.3.3. Aggregating filtered-in events [Component (3)]

This component combines the redundant events, thereby reducing the events in the filtered log. Aggregation merges redundant records into a single record [26]. A set of two or more events was combined if all the features in the events were exactly similar to the following event(s), and the aggregated event serves as the representative. Although much work used sampling methods for data reduction, it was noted by Tavallaee et al. [27] that sampling methods in anomaly detection introduce a significant bias that degrades the performance. Hence, we chose aggregating instead of data reduction.

3.3.4. Transferring events [Component (4)]

The purpose of this step is to extract the features from various aggregated logs as stated in GF to store in GFL. Unavailability of a particular feature in a log was buffered with a hyphen (-) during the transfer. The events in Apache server, Linux syslog, and Snort IDS logs were transferred to GFL, excluding Apache SSL_Error and Linux mail log, since many of the GFL features were not available in these logs, but they were maintained separately to be used in the analysis. As the iptables firewall log was also used only during analysis, these events were stored separately as per the features specified in GF.

3.3.5. Clustering GFE [Component (5)]

Even though the logs were clustered and filtered before, there are chances of having less number of normal or insignificant events due to the inaccuracy in clustering and the subsequent retention by the filtering threshold. To discover such events and also to find the relationship between the events in GFL, which contains events from various logs, GFE was reclustered. The GFE was clustered using the same two-step strategy, algorithms, and parameter settings used previously for the individual logs and the clustered events (GFE') were stored in ARFF format. The accuracy of clustering was evaluated using the same metrics used before.

3.3.6. Detecting anomalous events by analyzing features [Component (6)]

The clustered events of GFL was analyzed based on features to detect anomalies. We performed two types of analysis: IP address analysis and port number analysis.

IP address analysis identifies the relation between the IP address of the events from various logs to discover the presence of intrusive events in multiple logs. Therefore, the IP address existing in Linux and Apache events that also existed in Snort IDS events was identified. Not every IP address in Apache and Linux log has a match with the Snort IDS log, because some of the abnormal activities may have been overlooked by Snort IDS, but the anomalous patterns must have joined together in the same cluster during clustering. Therefore, to cover those abnormal activities that were missed by the Snort IDS log, all events in those clusters to which the identified IP address belonged to were extracted and considered as anomalous. During the analysis process three features were manipulated: IP address (f_{sip}), destination IP address (f_{dip}), and cluster number (f_k). Although checking the IP address of the events was the focus, the cluster number of these events was also manipulated to get an overall picture of the events related to intrusions. The analysis results were evaluated based on the volume of anomalous events identified. Additionally, it was compared with the results of the SOTM#34 challenge to substantiate the coverage of anomalies by UHAD.

Port number analysis identifies anomalous events based on port numbers. This was not an alternative to IP address analysis, but was rather done to capture those anomalous events that were available in those clusters that were not captured by it. Most anomalous activities were launched from a host by utilizing the unassigned and dynamic ports, since no predetermined service was running on these ports. In the case of inbound connections the source port number of the events was checked against the listing of the dynamic and unassigned port numbers as per the Internet Assigned Numbers Authority (IANA; http://www.ietf.org/assignments/port-numbers). Subsequently, the destination ports of the matching events were checked for the availability of well-known port numbers like 20, 21, 22, 23, 25, and 80 to generate an anomalous event list, as intrusive events mostly succeed through these ports on the victim host. On the other hand, for outbound connections, the destination port number of the events was checked against the listing of the dynamic and unassigned port numbers as per IANA and all the matching events were deemed as anomalous. The same set of GFL events (*GFE'*) used for IP address analysis was also used for port analysis. Naturally, events without port numbers will be automatically excluded from analysis. The anomalous events detected were evaluated with the same metrics used for IP address analysis.

3.3.7. Consolidating anomalous events [Component (7)]

This step consolidates the anomalous events to serve as the output of the framework by performing three operations, namely combining, correlating, and concentrating the events. Some of the events identified during IP analysis may also have been identified during port analysis. Therefore, these events were compared to distinguish them and distinct events were brought together for correlation. To recognize the relation between the events of the Linux syslog, Apache logs, and Snort IDS log with the iptables firewall log, the IP address (f_{sip} and f_{dip}, excluding internal IPs) of these events was compared with the iptables firewall log and the matching events were extracted and appended with the previously combined events. To concentrate on the most critical anomalous events, less significant events were eliminated using a threshold (A_t) on the occurrence of events related to an IP address. Those events containing IPs that appear A_t times or more were extracted to serve as the result of the framework. The performance of this step was evaluated by the retention of the most critical anomalous events with the usage of threshold.

4. Results and discussion

To evaluate the capacity of the framework in detecting anomalies, we extracted and combined the events from various sources, and a total of four subsets were tested. The performance of the framework in detecting various anomalies was evaluated based on the challenge results of SOTM#34 provided at Honeynet.org.

4.1. Clustered events

EM clustering was applied with the default values for the parameters (i.e. seed = 100, K = –1) on the selected features of the individual logs (E_{ij}) to estimate the best number of clusters (K_{ij}). For example, when predicting the number of clusters for the Apache Access log, EM initially selects eight clusters by cross-validation but ends up forming only five clusters (0, 1, 2, 6, 7). Thus, the ideal number of clusters chosen for this log was five, and the same strategy was followed for all the logs in calculating the ideal number of clusters. The cluster generated by the EM reveals that the number of clusters generated was not influenced by the number of features and events in a log, but rather by the patterns of the events. The time taken to predict the ideal number of clusters by EM for each log varied from seconds to minutes depending on the number of features and volume of events. Using the clusters (K_{ij}) manipulated, the logs were clustered and the clustered events ($E_{ij}{}'$) with different parameter settings were evaluated. The accuracy and FPR of the algorithms for the respective parameter settings is illustrated in Figure 2. The clustering strategy implemented showed accuracy of greater than 80% and FPR of less than 20% for all the log subsets clustered. Due to space constraints, we discuss the results of Apache access for subset-1 in this phase. FF and K-means always return the same number of clusters as requested, whereas EM produces fewer clusters than requested when there are insufficient distinct patterns in the log. Because of this nature, EM produced clusters separating the normal and abnormal events resulting in better accuracy for all parameter settings, but then it fails to clearly separate anomalous events from the others. Even though FF has a lower accuracy compared to EM, it produced better clusters.

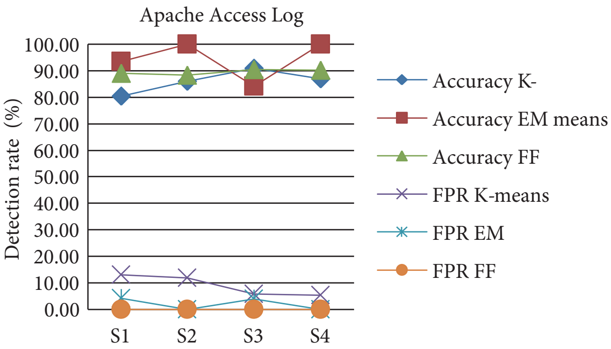

Figure 2. Clustering accuracy and FPR with different parameter settings (Access Log).

The clusters generated by the algorithms for various settings were verified to substantiate the ability of the algorithm in classifying the abnormal events in separate clusters. In the Apache Access log, there were 16 successful intrusive events and 42 unsuccessful events. Within the context of our research, both were considered as anomalies. The Apache Access log, consisting of 256 events clustered using FF with K = 5 and the default seed = 1 (Setting-1), resulted in Clusters 0 to 4 consisting of 189, 3, 48, 13, and 3 events, respectively. FF managed to group the successful intrusive events in smaller clusters 1 and 3. Out of the 42 unsuccessful intrusive events 33 were grouped together with other events in Cluster 0, i.e. the biggest cluster, and nine events in a smaller cluster, i.e. Cluster 2.

The same log when tested with parameter setting-2 also yielded similar clusters as setting-1, showing the insensitiveness of FF towards the increase in seed value. Settings 3 and 4 also resulted in similar clusters of successful intrusive events, but then the increase in K had an impact in the formation of clusters, especially in grouping normal and unsuccessful intrusive events. All the 42 unsuccessful intrusive events were precisely grouped in a separate cluster, i.e. Cluster 5. EM grouped all the 16 successful intrusive events together in a smaller cluster, i.e. Cluster 2 for setting-1, Cluster 1 for setting-3, and Cluster 5 for setting-4. This shows the precision of EM in grouping successful intrusive events in separate clusters. However, for setting-2, the successful intrusive events were grouped into two clusters, i.e. 15 events in Cluster 2 and 1 event in Cluster 3, together with other events in both clusters. The majority of the unsuccessful events were assembled in smaller clusters together with other events for settings 1 and 2, whereas for settings 3 and 4, they were grouped in separate clusters. There was a significant effect on the formation of clusters when seed and K values were increased. K-means grouped only two of the successful intrusive events in smaller clusters for setting-1 and all the 16 events for setting-2, but together with other events. For setting-3, K-means grouped only three of the successful intrusive events in the smaller cluster and another 13 events in the biggest cluster, i.e. Cluster 0, but with setting-4, all the successful intrusive events were grouped into two separate clusters, i.e. Clusters 2 and 3. All 42 unsuccessful intrusive events were grouped in small clusters together with other events for settings 1 and 2, but in a separate cluster for settings 3 and 4. This shows the sensitive nature of K-means to K and seed values in the formation of clusters. The results show that FF performed well in correctly classifying successful intrusive events in separate clusters, whereas EM maintained a balance between classifying both successful and unsuccessful intrusions in appropriate clusters. EM achieved the maximum accuracy with settings 2 and 4 for most of the logs in this subset, which indicates the impact of seed values in forming better clusters. With all the four subsets tested, EM accurately clustered most of the individual logs with a minimum number of FP and FN.

4.2. Filtered events

In this phase, the clustered logs were scanned to identify the number of events and clusters to calculate the threshold, and the events were filtered accordingly. For the Apache error log clustered using FF with setting-1, there were 433 events in seven clusters (104, 45, 32, 9, 67, 41, 135). The threshold calculated was 62 (E_t = 433 / 7), and there were four clusters satisfying the threshold that retained 127 events while eliminating 306 events. The volume of events filtered out depends on the number of clusters and its size. The use of a threshold managed to filter out an average of 65% events, while retaining an average of 35% events for further scrutiny with all the four subsets tested. Out of 209 intrusive events available in the Apache Error log, 32 events were successful while 177 were unsuccessful. All the successful intrusive events were filtered in by all the algorithms for the majority of the settings. Most or all unsuccessful intrusions were filtered out because of the high number of such events having similar patterns recorded in this log; it resulted in forming bigger clusters that did not satisfy the threshold. The percentages of anomalies retained by the algorithms with the respective settings are illustrated in Figure 3. Filtering events using the calculated threshold managed to retain all the successful intrusive events, but failed to retain the majority of unsuccessful intrusions as the volume of such events was high. This is the limitation of using a calculated threshold (E_t) to filter the log events.

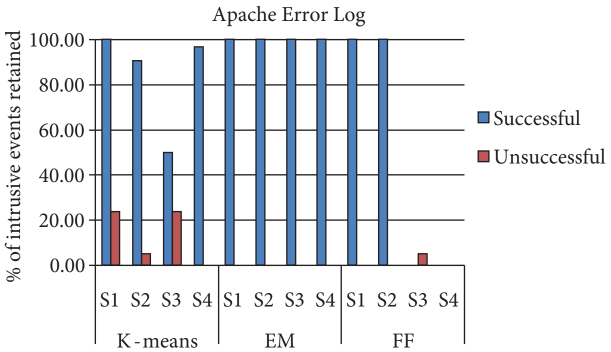

Figure 3. Retention of anomalies after filtering (Error Log).

4.3. Aggregated events

Since all the intrusive events filtered in were distinctive, aggregation did not remove any of these events. Less numbers of filtered-in events were reduced in logs, but further processing was not affected by this reduction. Apache access and error log were trimmed to an average of 12.75%, whereas it was an average of 2% in Linux syslog. Nearly 75% of the filtered-in events were reduced in the SSL_Error log because of similar event patterns recorded in this log with the same timestamp. Aggregation reduced an average of 20% filtered-in events in the Snort IDS log. A similar percentage of reduction was achieved by aggregation with the other three subsets as well. All the events that were retained by the respective logs after it had been filtered and aggregated were abnormal events of some kind.

4.4. Transferred events to the framed GFL

Even though all the logs with the stated features in GFL were used for analysis to detect intrusions, only certain logs were transferred to GFL format to be used in the clustering phase. Apache and Linux events were captured directly by the victim system and every abnormal activity expressed by Apache server logs and Linux syslog should have generated events in the Snort IDS log, as it has been designed to capture such activities. These events were transferred to GFL. Since the events of the Apache SSL_Error log and Linux mail log did not have IP addresses and port numbers, these events were not transferred. However, the respective features of the excluded logs were maintained separately to be used in the analysis phase.

4.5. Clustered GFL events

The accuracy and FPR of clustering by various algorithms and settings for subset-1 is illustrated in Figure 4. In spite of the differences in the nature of events and the absence of values for some features, EM and FF had a maximum accuracy of greater than 99% and a lowest FPR of 0% with setting-2. This exhibits the impact of seed value together with ideal clusters (K_g) in clustering. Despite the number of clusters requested, EM always clusters based on the available patterns and hence increasing clusters (k) for settings 3 and 4 decreased the accuracy, but an increase in seed value resulted in higher accuracy. Like EM, the accuracy of FF decreased with the increase in clusters. As K-means is very sensitive to initializations the accuracy decreased much with the raise in K and seed. FF achieved the maximum accuracy with GFL for most of the settings with all the four subsets, exposing its capacity in clustering heterogeneous log events despite the absence of values for some of the features.

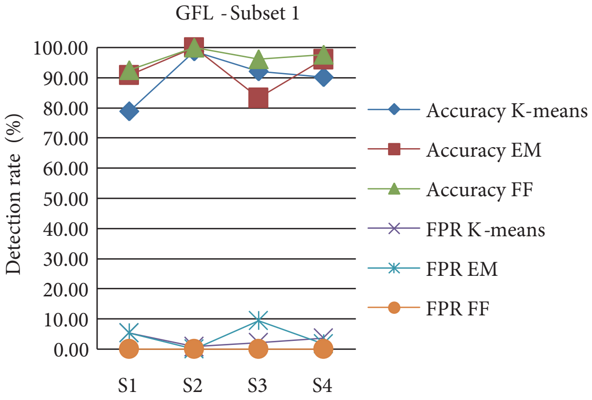

Figure 4. Clustering accuracy and FPR of GFL.

4.6. IP address analysis

IP analysis showed its ability in identifying most anomalous events in the logs and the volume of anomalous events identified was affected by the formation of clusters in individual logs and GFL. The anomalous events identified by IP analysis for various algorithms with different settings are illustrated in Figure 5. The best performance was achieved with EM clusters by discovering most of the anomalous events. This was due to the formation of accurate clusters by EM that were superior to the other algorithms that we tested. In this subset there were 15 IP addresses from which the anomalous events originated. Out of these 15 IPs, three IPs existed only in the Snort IDS log and five existed only in Linux syslog, and still the analysis managed to detect the anomalous events related to these IPs as well. This is because the analysis considers not only the anomalous events of IPs that exist in multiple log sources but also the events belonging to clusters of these IPs. The analysis detected anomalous events from 14 of these IPs with EM clusters. The only IP that was not identified by the analysis was available only in Linux security log and was filtered out by the threshold (E_t), making it unavailable for analysis. A total of 1129 anomalous events from these IPs were detected by analysis with EM clusters for setting-2, whereas 697 and 331 events were detected with K-means and FF clusters respectively for setting-1. The better performance with EM clusters with setting-2 also confirms the impact of seed in the formation of clusters. IP analysis detected the maximum number of anomalous events from different IP address with EM clusters for subsets 1 and 4 and with K-Means clusters for subsets 2 and 3.

4.7. Port number analysis

Only the Snort IDS events in GFE' were scrutinized, since it contains both source and destination ports. The volume of anomalous events identified by port analysis is illustrated in Figure 6. Even though the number of anomalous events identified by port analysis was relatively small, it still managed to retain the anomalous events, especially the ones related to AWStats vulnerability. A total of 162 anomalous events were detected, which originated from six different IPs in EM clusters with setting-4. This was because of the highest accuracy, i.e. 99.46%, achieved in clustering Snort IDS logs, thereby enabling the filtering threshold to retain the events for analysis. Even though the volume of anomalous events detected with FF and K-Means clusters was less compared to EM clusters, the coverage of anomalies from different IPs was more. The anomalous events originating from seven different IPs were detected with FF and K-means clusters for settings 4 and 3, respectively. A range of 1 to 22 anomalous events that were missed during IP analysis were discovered by port analysis with K-means and FF clusters. All the events identified by port analysis for EM clusters were already identified during IP analysis, implying that the accurate cluster formation reduces the need for port analysis. The majority of the anomalous events from different IPs were detected by port analysis with EM clusters for subsets 1 and 2 and with K-means clusters for subsets 3 and 4.

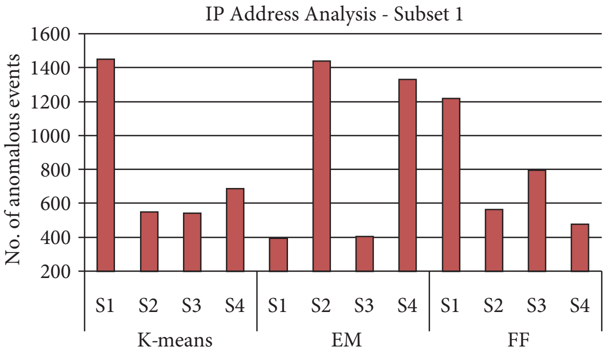

Figure 5. Anomalous events of IP analysis.

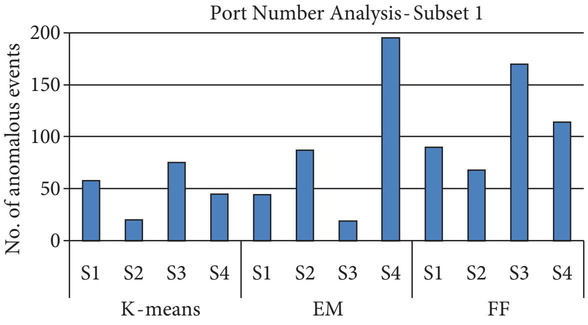

Figure 6. Anomalous events of port number analysis.

4.8. Consolidated anomalous events

Application of a threshold to consolidate anomalous events reduced a minimum number of anomalous events apart from reducing insignificant events. The reduction was minimal with the analysis results of EM and FF clusters but slightly higher with K-means clusters. The consolidated anomalous events were scrutinized to verify whether the usage of a threshold eliminated the significant events related to those IPs from which the intrusions originated. The effect of the threshold in retaining significant anomalous events is presented in Figure 7. A very small percentage of significant anomalous events was reduced by the threshold, and the increase in the number of clusters to group the events with settings 3 and 4 had an impact on the percentage of reduction. The percentage of anomalous events reduced by the threshold with settings 1 and 2 was less than 2%, whereas it reached a maximum of 16% with the other two settings. Apart from this, IPs represented by these anomalous events also gain significance, since they help to understand the actions taken by the intruder. All the 14 IPs detected by IP analysis with EM clusters for setting-2 from which the intrusion attempt was launched were retained before the application of the threshold. As the threshold was based on the occurrence of events pertaining to an IP address, an increase in threshold failed to retain anomalous events from three of these IPs as they were less than the threshold. Though the usage of a threshold is not viable at this end, it assists in focusing on the most critical events that need to be immediately addressed.

UHAD detected a wide range of anomalies including Nimda scans, CodeRed worms, SSH Brute Force scans, CONNECT scans, IRC Traces, and especially the AWStats vulnerability scans through which the intrusion succeeded. The samples of the anomalous events detected by UHAD that exploit the AWStats vulnerability are tabulated in Table 3. The percentage of anomalous events detected with various subsets is presented in Figure 8. The detection performance in using a particular clustering algorithm in UHAD was not sturdy with different subsets. This was influenced by the capacity of a clustering algorithm in handling varying patterns of events in different subsets.

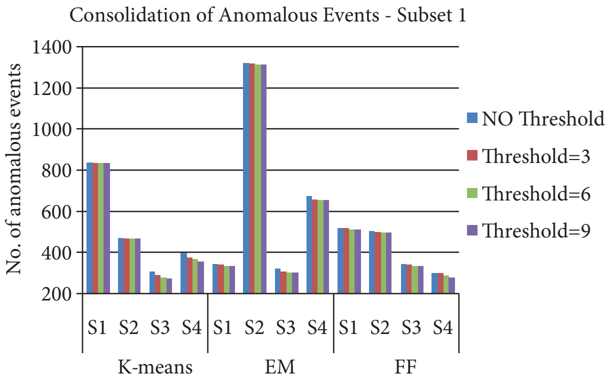

Figure 7. Impact of threshold in retaining significant anomalous events.

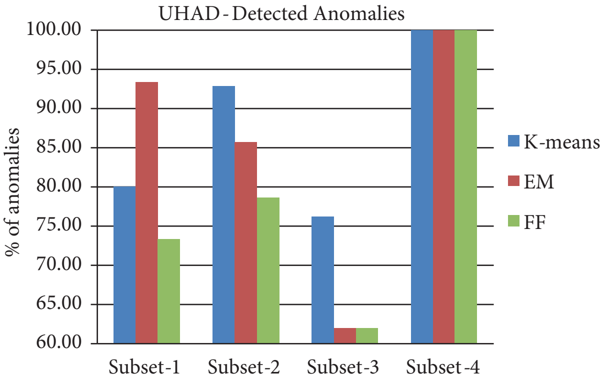

Figure 8. Anomalies detected by UHAD.

Due to the lack of availability of the latest datasets containing heterogeneous logs, SOTM#34 was used to test the capacity of the proposed framework in detecting anomalies. Although this dataset contained more anomalous events, UHAD managed to detect most of the critical events by discovering the relationship between the events from various logs. We think that UHAD will be able to detect most of the anomalies even in the latest traffic logs, as the decision is based on the pattern of events and the basic features it contains.

5. Conclusion and future work

We have presented UHAD, a new unsupervised anomaly detection framework to scrutinize logs from heterogeneous sources, without using any knowledge base (signature) or the usual method of training and testing commonly used in anomaly detectors. The efficiency and accuracy of UHAD in identifying intrusions were examined using three clustering algorithms with four parametric settings and the results were compared with the work of others using the same dataset. The detected anomalies were parallel with the output of other methods, hence proving the accuracy of UHAD in detecting anomalies. Among the three algorithms used by the framework, EM and K-means produced better clusters, assisting the analysis process to detect the majority

Table 3. Intrusion through AWStats.

Date	Time	Source IP	Destination IP	Source Port	Destination Port	Protocol	Message
26-Feb-05	18:57:37	213.135.2.227	-	-	-	-	GET/cgi-bin/awstats.plHTTP/1.0
26-Feb-05	18:57:37	213.135.2.227	11.11.79.89	49727	80	TCP	SYN
26-Feb-05	19:00:43	213.135.2.227	11.11.79.89	50860	80	TCP	BLEEDING-EDGE EXPLOIT Awstats Remote Code Execution Attempt
26-Feb-05	19:01:44	82.55.78.243	11.11.79.67	24934	80	TCP	SYN
26-Feb-05	19:01:47	82.55.78.243	11.11.79.67	24934	80	TCP	WEB-ATTACKS rm command attempt
26-Feb-05	19:02:00	82.55.78.243	-	-	-	-	–14:14:45– http://www.shady.go.ro/aw.tgz
27-Feb-05	2:55:37	212.203.66.69	11.11.79.67	33847	80	TCP	SYN
27-Feb-05	2:55:40	212.203.66.69	11.11.79.67	33847	80	TCP	BLEEDING-EDGE EXPLOIT Awstats Remote Code Execution Attempt
27-Feb-05	2:55:43	212.203.66.69	-	-	-	-	GET /cgi-bin/awstats.pl? configdir=7cecho203becho20 b_exp3bcd202ftmp3bcurl3becho 20e_exp3b2500 HTTP/1.1

of the anomalies in all the four subsets. The coverage of anomalies with FF clusters were marginally lower than EM and K-means, but is also applicable for this framework due to its faster clustering even with larger datasets.

All the parameters used in UHAD were calculated based on the tested dataset alone. Hence, the accuracy of clusters and subsequent retention of anomalies by the threshold were influenced by these parameters. None of the algorithms showed a consistent accuracy with a particular parameter setting, and the performance of a particular algorithm was not steady with different subsets. Because of this, some of the anomalous events were filtered out by the threshold, making it unavailable for further investigation. The mechanism of calculating the clustering parameters and filtering threshold should be refined to improve the precision of anomaly detection and more clustering algorithms need to be tested to evaluate their applicability for the framework. A recommender method should be designed to choose the most accurate clusters for a particular log among those generated by various algorithms with different parametric settings. Consequently, these chosen clusters can be used for further analysis to detect almost all the anomalies.

References

[1] Yu X, Tang LA, Han J. Filtering and refinement: a two-stage approach for efficient and effective anomaly detection. In: Proceedings of the 9th IEEE International Conference on Data Mining; December 2009; Miami, FL, USA. New York, NY, USA: IEEE. pp. 617–626.

[2] Chimphlee W, Abdullah AH, Sap MNM, Chimphlee S, Srinoy S. Unsupervised clustering methods for identifying rare events in anomaly detection. In: Proceedings of the 6th International Enformatika Conference; 26–28 August 2005; Çanakkale, Turkey. pp. 26–28.

[3] Erman J, Arlitt M, Mahanti A. Traffic classification using clustering algorithms. In: Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data; 2006. New York, NY, USA: ACM. pp. 281–286.

[4] Münz G, Li S, Carle G. Traffic anomaly detection using k-means clustering. In: Proceedings of Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen 4, GI/ITG-Workshop MMBnet; September 2007; Hamburg, Germany.

[5] Smith R, Japkowicz N, Dondo M, Mason P. Using unsupervised learning for network alert correlation. Lect Notes Artif Int 2008; 308–319.

[6] Syarif I, Prugel-Bennett A, Wills G. Unsupervised clustering approach for network anomaly detection. Comm Com Inf Sc 2012; 293: 135–145.

[7] Casas P, Mazel J, Owezarski P. Steps towards autonomous network security: unsupervised detection of network attacks. In: 4th IFIP International Conference on New Technologies, Mobility and Security; 2011; Paris, France. pp. 1–5.

[8] Abad C, Taylor J, Sengul C, Yurick W, Zhou Y, Rowe KE. Log correlation for intrusion detection: a proof of concept. In: Proceedings of the 19th Annual Computer Security Applications Conference; December 2003; Las Vegas, NV, USA. pp. 255–264.

[9] Yurcik W, Abad C, Hasan R, Saleem M, Sridharan S. UCLog+ : A security data management system for correlating alerts, incidents, and raw data from remote logs. Arxiv Preprint cs/0607111, 2006.

[10] Li Z, Taylor J, Partridge E, Zhou Y, Yurcik W, Abad C, Barlow JJ, Rosendale J. UCLog: A unified, correlated logging architecture for intrusion detection. In: 12th International Conference on Telecommunication Systems – Modeling and Analysis (ICTSM), 2004.

[11] Song J, Takakura H, Okabe Y, Nakao K. Toward a more practical unsupervised anomaly detection system. Inform Sciences 2013; 231: 4–14.

[12] Asif-Iqbal H, Udzir NI, Mahmod R, Ghani AAA. Filtering events using clustering in heterogeneous security logs. Inform Technol J 2011; 10: 798–806.

[13] Panichprecha S. Abstracting and correlating heterogeneous events to detect complex scenarios. PhD Thesis, Queensland University of Technology, Brisbane, Australia, 2009.

[14] Herrerias J, Gomez R. Log analysis towards an automated forensic diagnosis system. In: International Conference on Availability, Reliability and Security; February 2010; Krakow, Poland. pp. 659–664.

[15] Amiri F, Rezaei Yousefi MM, Lucas C, Shakery A. Mutual information-based feature selection for intrusion detection systems. J Netw Comput Appl 2011; 34: 1184–1199.

[16] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter 2009; 11: 10–18.

[17] Sinclair C, Pierce L, Matzner S. An application of machine learning to network intrusion detection. In: Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC '99); 6–10 December 1999, Phoenix, AZ, USA. New York, NY, USA: IEEE. pp. 371–377.

[18] Barbara D, Wu N, Jajodia S. Detecting novel network intrusions using bayes estimators. In: Proceedings of the First SIAM International Conference on Data Mining (SDM 2001); 2001; Chicago, IL, USA. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics (SIAM). pp. 1–17.

[19] Li Y, Wu N, Jajodia S, Wang XS. Enhancing profiles for anomaly detection using time granularities. Journal of Computer Security 2002; 10: 137–158.

[20] Staniford S, Hoagland JA, McAlerney JM. Practical automated detection of stealthy portscans. Journal of Computer Security 2002; 10: 105–136.

[21] Dempster AP, Laird NM, Rubi DB. Maximum likelihood from incomplete data via the EM algorithm. J Roy Stat Soc B Met 1977; 39: 1–38.

[22] MacQueen J. Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability; 1967; Berkeley, CA, USA. pp. 281–297.

[23] Tjhai GC, Furnell SM, Papadaki M, Clarke NL. A preliminary two-stage alarm correlation and filtering system using SOM neural network and K-means algorithm. Comput Secur 2010; 29: 712–723.

[24] Dasgupta S, Long PM. Performance guarantees for hierarchical clustering. J Comput Syst Sci 2005; 70: 555–569.

[25] Zheng X, Cai Z, Li Q. An experimental comparison of three kinds of clustering algorithms. In: International Conference on Neural Networks and Brain; 13–15 October 2005; Beijing, China. New York, NY, USA: IEEE. pp. 767–771.

[26] Kent K, Souppaya M. Guide to Computer Security Log Management. White Paper, NIST Special Publication 800-92, 2006.

[27] Tavallaee M, Stakhanova N, Ghorbani AA. Toward credible evaluation of anomaly-based intrusion-detection methods. IEEE T Syst Man Cy C 2010; 40: 516–524.