Scenario-Based Insider Threat Detection From Cyber Activities

Pratik Chattopadhyay, Lipo Wang, and Yap-Peng Tan, *Senior Member, IEEE*

***Abstract*— An insider threat scenario refers to the outcome of a set of malicious activities caused by intentional or unintentional misuse of the organization's systems, networks, data, and resources. Prevention of insider threat is difficult, since trusted partners of the organization are involved in it, who have authorized access to these confidential/sensitive resources. The state-of-the-art research on insider threat detection mostly focuses on developing unsupervised behavioral anomaly detection techniques with the objective of finding out anomalousness or abnormal changes in user behavior over time. However, an anomalous activity is not necessarily malicious that can lead to an insider threat scenario. As an improvement to the existing approaches, we propose a technique for insider threat detection from time-series classification of user activities. Initially, a set of single-day features is computed from the user activity logs. A time-series feature vector is next constructed from the statistics of each single-day feature over a period of time. The label of each time-series feature vector (whether malicious or nonmalicious) is extracted from the ground truth. To classify the imbalanced ground-truth insider threat data consisting of only a small number of malicious instances, we employ a cost-sensitive data adjustment technique that undersamples the nonmalicious class instances randomly. As a classifier, we employ a two-layered deep autoencoder neural network and compare its performance with other popularly used classifiers: random forest and multilayer perceptron. Encouraging results are obtained by evaluating our approach using the CMU Insider Threat Data, which is the only publicly available insider threat data set consisting of about 14-GB web-browsing logs, along with logon, device connection, file transfer, and e-mail log files. We observe that both deep autoencoder and random forest classifiers classify the data-adjusted time-series feature set with high precision, recall, and f-score. Although multilayer perceptron has a high recall, it suffers from a lower precision and f-score compared to the other two classifiers.**

***Index Terms*— Cost-sensitive learning, imbalanced data, insider threat, time-series classification.**

I. Introduction

INSIDERS are the trusted partners of an organization who have authorized access to the organization's resources, data, and network. In unwanted situations, these insiders loose liability toward their employers either out of anger or out of greed for making personal benefits. This provokes them to behave maliciously and intentionally harm the organization's resources and lower its reputation. According to IBM 2016 Cyber Security Intelligence Index (a survey made by IBM about the threat) [1], among all the security breaches or attacks recorded in 2015 worldwide, more than 60% are caused by insiders only. Thus, the prevention of insider threat is a matter of serious concern.

The main objective of this paper is to develop an effective insider threat detection algorithm to classify a sequence of insider activities as malicious or nonmalicious. In this paper, we will focus on the analysis of human behavioral activities only. Analysis of psychometry, emotion, and other nonbehavioral factors is outside the scope of this paper.

Depending on the sequence of malicious activities involved, insider threat can be categorized into a number of threat scenarios. Providing proper definition to these threat scenarios is challenging, since it requires past experience as well as extensive manual effort. In this paper, instead of defining the threat scenarios, we focus on developing an effective machine learning algorithm to achieve our objective. We assume that the definition of the threat scenarios and the ground-truth information pertaining to the different threat scenarios is available to us.

The CMU Insider Threat Data [2] is the public insider threat data available till date. Hence, the evaluation of the approach is done using these data. Version 4.2 of the data is used in this paper, since it focuses only on behavioral characteristics. This data set consists of the user-activity logs of 1000 insiders over a period of 17 months from January 2010 to May 2011. The different user activity logs captured in the CMU Data are as follows.

1) The logon activity details are stored in a file named logon.csv. For each user, the file stores the computer ids in which he/she has logged in or logged off along with the corresponding time stamps.
2) The device.csv file contains the device connection information for each user, including the computer ids and the time stamps at which a device is connected/disconnected.
3) The file transfer details for each user are stored in file.csv. Along with the time stamp, it also lists the type (pdf, doc, jpg, and so on) of file/files transferred.
4) The e-mail exchange details are contained in a file named email.csv. It provides information, such as the number of e-mails sent by a user on a day, e-mail size, attachment count, whether the recipient is from the same organization or not, and so on.
5) The http log is provided by the file http.csv. It specifies the URLs visited, the machine id accessed to visit each

Manuscript received January 10, 2018; revised May 17, 2018; accepted July 1, 2018. Date of publication August 23, 2018; date of current version September 11, 2018. *(Corresponding author: Lipo Wang.)*

P. Chattopadhyay is with the Department of Computer Science and Engineering, IIT (BHU) Varanasi, Varanasi 221005, India (e-mail: pratik.cse@iitbhu.ac.in).

L. Wang and Y.-P. Tan are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: elpwang@ntu.edu.sg; eyptan@ntu.edu.sg).

Digital Object Identifier 10.1109/TCSS.2018.2857473

web page, and some keywords selected from the content of the web page.

6) The psychometric.csv contains the big five psychometric scores (i.e., O, C, E, A, and N).[1]

Three threat scenarios are defined in the CMU Data based on the above set of user activities. These are as follows.

1) *Scenario S_1:* User who has never used removable drives previously or worked overtime starts behaving uncharacteristically by logging in after office hours using a removable drive and uploading data to wikileaks.org. He/she leaves the organization shortly after this.
2) *Scenario S_2:* User begins searching for new employment opportunities by surfing job websites. Before leaving the organization, he/she steals confidential information using a thumb drive. The frequency of the thumb drive usage is higher than what he/she used to do previously.
3) *Scenario S_3:* User downloads keylogger and obtains a list of passwords of different employees of the organization. Next, he/she uses a thumb drive to transfer this list of passwords to the supervisor's machine and tries searching for the supervisor's password. Once successful, he/she logs into the supervisor's machine and broadcasts an alarming mass e-mail creating panic in the organization. This type of malicious activity comes from system administrators when they become disgruntled with their supervisors.

In addition to the threat scenarios, the CMU Insider Threat Data provide a set of ground truth for each of the above-mentioned threat scenarios along with the date and time stamp of each activity.

From the definition of the threat scenarios, it is observed that none of these deals with psychometric information. Hence, the log file named "psychometric.csv" has not been used in this paper. Section I-A provides the background knowledge required to carry out this paper. This includes the description of the previous work done on insider threat detection, their shortcomings, and how the proposed method is capable of overcoming these drawbacks.

A. Background

Several factors can trigger an insider threat attack. A few of these, as mentioned by Liang and Biros [3], are personality problems, mental health disorders, social isolation, emotional characteristics, disgruntlement, social and cultural conflicts, and so on. These factors have often been seen to affect an insider's behavior adversely, provoking him/her to behave maliciously. Majority of the work done on insider threat detection are unsupervised [4]–[12], while a few supervised methods [13]–[15] have also been developed. In Section I-A1, we provide an overview of the state-of-the-art research on behavioral-based insider threat detection.

1) Behavioral-Based Insider Threat Detection: Unsupervised Methods: A research team from Leidos, an American defense company, developed an unsupervised anomaly detection system termed PROactive Detection of Insider threats with Graph Analysis and Learning,[2] which accepts user activity logs as inputs and uses high-performance computing for anomaly detection. Memory *et al.* [4] described an anomaly detection technique by analyzing user roles and their historical access patterns from the computer usage and network activity logs. The list of blackboard models given in [4] can be used as a baseline to set up a new insider threat detection system. Young *et al.* [5] proposed a structural anomaly detection method by combining structural and semantic information from user activities. The features used in this paper are: 1) count of attachments on e-mails sent; 2) count of file transfers to removable drives; 3) count of distinct machines logged onto; 5) count of print jobs submitted; 6) count of blacklisted web pages visited; 6) ratio of file transfers on removable drives to all file transfers; 7) ratio of the number of upload activities to the number of download activities; and 8) ratio of the number of distinct removable drives used for uploading to that used for downloading. Predicting anomalousness in behavior by following an ensemble-based clustering approach has been studied by the same research team in [6] and [16]. Performance evaluation of these approaches on an extensive data set consisting of computer usage activity logs of 5500 people has shown encouraging results. However, this data set has not been made available to the public, and hence could not be used in this paper.

Researchers from the Palo Alto Research Center, Palo Alto, CA, USA, proposed a proactive insider threat detection approach in [8] by combining structural anomalies obtained from social and information networks, along with the psychological profile extracted from user behavioral patterns. A time-series-based anomaly detection method has been proposed in [7], which concatenates the statistics (mean, weighted mean, and sum of weighted differences) of single-day activities over a period of time to generate a time-series feature vector. An unsupervised isolation forest algorithm is next employed to predict if an insider behavior is anomalous from the above statistical features.

The Cyber Security Centre of the University of Oxford developed another anomaly detection tool termed Corporate Insider Threat Detection [9] to detect anomalous activities. Three activity indicators have been considered here: 1) user's daily activities; 2) comparison between user's daily activity and his/her previous activity; and 3) comparison between the user's daily activity and the previous activity of their role. The active learning strategy employed here allows the analyst to incorporate knowledge back into the system via an interactive manner. Similar to [4], Nurse *et al.* [10] have also described a grounded framework for insider threat detection by identifying the key factors that have triggered insider threat scenarios in the past. This paper does not focus on user activity analysis but only provides an overview of the several factors responsible for triggering an attack as well as an understanding of whether an insider threat activity has been done deliberately or unintentionally.

Magklaras and Furnell [14] proposed an approach that performs file content analysis and integrity checking to monitor

[1] https://en.wikipedia.org/wiki/Big_Five_personality_traits

[2] https://en.wikipedia.org/wiki/Proactive_Discovery_of_Insider_Threats_Using_Graph_Analysis_and_Learning

misuse of computer systems. It takes into account several human behavioral characteristics, such as user knowledge of a file system, network interaction patterns, and so on. Other insider threat detection approaches include [17] that deals with the integration of user taxonomy and psychological profiling to identify user misbehavior in real time from his/her technological trait. However, experimental evaluation of the algorithm has not been done, and no results have been reported. Along with anomaly detection, Eldardiry *et al.* [18] proposed a method for detecting spoofing attacks, i.e., identifying suspicious insiders who pretend to behave in a manner similar to the group they belong to. The final anomaly score for a user is computed as the weighted sum of the anomaly scores obtained from the different activities, e.g., logon, file transfer, e-mail, web browsing, and so on.

The ELICIT anomaly detection system described by Maloof and Stephens [11] aims to detect insiders who violate the "need to know" principle. Usually, any security organization defines a clearance level for its users. An attempt to seek information beyond the clearance level is said to be a violation of the "need to know" principle. A real-world corporate intranet consisting of 3900 users has been used to test the performance of the ELICIT model in [11]. The anomaly score has been computed from Bayesian ranking of the scores obtained from 76 different anomaly detectors. The approach in [12] describes the identification of anomalous user activities by assigning a threat rank to each user on each day on the basis of the following factors: severity of the threat and frequency of anomalous behavior. A high disparity in the magnitude of the threat rank implies significant deviation of the single-day user activities from their usual pattern. The work in [19] discusses about the detection of anomalies from single modalities, e.g., web access patterns, printing, file access, and so on, and how to fuse them using a domain-knowledge driven Bayesian network.

Supervised Methods: To date, only a few supervised insider threat detection approaches have been developed, which are described next. The work in [14] describes a taxonomy for the development of a supervised insider threat detection system, but does not focus on the relevant features to be computed for identifying the threat scenarios. As described before in Section I, an insider threat database is generally imbalanced with only a small number of malicious instances and a comparatively large number of nonmalicious instances. Learning from imbalanced data poses significant challenges. Although important, this issue has not been given significant attention in the past. Only a few works [13], [15] have given insights to this problem. Pfleeger [13] explained the concept of insider threat and discusses about the several classification challenges arising out of the scarcity of malicious ground truth data. However, the paper provides only a conceptual framework, and no details have been given on the technical aspects required for effectively handling imbalanced data. Azaria *et al.* [15] presented a framework for the behavioral analysis of insider threat (BAIT) using a semisupervised bootstrapping-based classification method that learns from highly imbalanced data. The data set used for performance evaluation in [15] has no resemblance with that of the CMU Data, which consists of user-activity logs, and hence, the features described in [15] are not suitable for this paper. Specifically, Azaria *et al.* [15] designed a one-person game and recruited 795 subjects to play the game on Amazon Mechanical Turk. To introduce imbalancedness in the data, only a few subjects were asked to behave maliciously. The model is used to predict a user as malicious or nonmalicious from the way he/she plays the game. Simple aggregated behavioral features have been used here, e.g., frequency with which each and every activity (fetching data from CD/DVD/USB, sending e-mail with encrypted data, and so on) is executed. Seven variations of the BAIT algorithm have been described in this paper, out of which the best one has a recall of 0.7 and a precision of only 0.07.

TABLE I
CONFUSION MATRIX FOR BINARY CLASSIFICATION

Predicted \ Actual	Malicious	Non-malicious
Malicious	True Positive (TP)	False Positive (FP)
Non-malicious	False Negative (FN)	True Negative (TN)

Similar to the data set used in [15], the ground truth of CMU Insider Threat Data is also highly imbalanced with only a small number of malicious class instances and a comparatively larger number of nonmalicious class instances. Cost-sensitive learning [20]–[29] has been extensively used in the past to effectively handle imbalanced data. Section I-A2 focuses on describing the various cost-sensitive learning techniques in the literature, along with useful metrics for evaluating classification performance on imbalanced data.

2) Class-Imbalance Problem and Existing Solutions: A confusion matrix [30] (refer to Table I) is commonly used to compute metrics for performance evaluation of classification algorithms. Let, for any class, $\mathcal{C}$, TP, FP, FN, and TN, respectively, denote the number of true positives, false positives, false negatives, and true negatives. Single class metrics, namely, precision, recall, and f-score (and not overall accuracy) are generally recommended for evaluating classifier performance on imbalanced data. With reference to Table I, precision ($\mathcal{P}_\mathcal{C}$), recall ($\mathcal{R}_\mathcal{C}$), and f-score ($\mathcal{F}_\mathcal{C}$) for class $\mathcal{C}$ are mathematically expressed as follows:

$$\mathcal{P}_\mathcal{C} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{1}$$

$$\mathcal{R}_\mathcal{C} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2}$$

$$\mathcal{F}_\mathcal{C} = 2\frac{(1+\beta^2)\mathcal{P}_\mathcal{C}\mathcal{R}_\mathcal{C}}{\mathcal{P}_\mathcal{C} + \beta^2\mathcal{R}_\mathcal{C}} \tag{3}$$

where β is a positive constant. Macroaveraged precision ($\mathcal{P}_{\text{AV}}$), recall ($\mathcal{R}_{\text{AV}}$), and f-score ($\mathcal{F}_{\text{AV}}$) are often used to study the overall classifier performance. These are computed by averaging the precision, recall, and f-score values obtained from each class.

In contrast to the traditional classification algorithms, cost-sensitive classification requires a cost matrix (more specifically a misclassification cost matrix) as an input during the training phase. In general, for C classes, the cost matrix is a square matrix of dimensions $C \times C$, where the element corresponding to row i and column j indicates the cost of misclassifying a sample from class j as class i. In the case of

binary classification problems such as insider threat detection, the cost matrix is of dimensions 2×2. Several categories of cost-sensitive learning techniques have been introduced by researchers over the years [24], [29]. Among these, the most important are: 1) algorithm-level adjustments; 2) altering class distribution of the ground truth data; and 3) ensemble-based learning. The first category of approaches is mainly applicable for classification using neural networks or other classifiers. The original backpropagation learning algorithm is unsuitable for classifying data with imbalanced class distribution. A cost-sensitive version of the backpropagation learning algorithm, described in [31], has taken into account the adaptivity of learning rate and the adaptivity of output neuron values by means of expected misclassification costs. Another cost-sensitive learning technique using a radial basis neural network has been described by Wan *et al.* [20], where the learning rate for the minority class has been chosen to be higher than that for the majority class. This causes the backpropagated error to have a higher influence on the minority class than on the majority class, and the network gradually gets trained with the minority class samples accurately. In another work on cost-sensitive neural network [21], instead of adjusting the learning rate, the misclassification cost for each class has been multiplied with the predicted value at the corresponding output neuron. Likewise, any classifier can be transformed to its corresponding cost-sensitive version. Two such techniques are: 1) [29], which describes a method based on fuzzy SVM for class imbalanced learning and 2) [32], which shows the application of genetic algorithm to generate a population of biases for a decision tree induction algorithm. Another framework for a cost-sensitive neural network termed MetaCost has been described in [33], which is based on relabeling of training examples with their estimated minimal-cost classes and repeatedly applying the classifier on the newly labeled training set.

In the second category of cost-sensitive learning approaches, data-level adjustments are done to reduce the imbalanced data distribution. These methods resample (i.e., either oversample or undersample) the training data in proportion with the misclassification cost provided by the cost matrix. Both undersampling and oversampling have their own benefits and limitations. While undersampling of majority class might lose important information, oversampling of the minority class can lead to overfitting [34].

One of the earliest data adjustment algorithms is termed Synthetic Minority Over-sampling Technique (SMOTE) [25]. It oversamples the minority class by synthetically adding new samples to the training set. The algorithm selects one sample at a time, extracts feature vector from the minority sample, finds the nearest neighbor of this feature vector, computes the vector difference between the two, multiplies this difference with a random number generated in the range *(0,1)*, and finally adds this weighted difference to the feature vector. Using this technique, new minority samples are generated in the neighborhood of the original sample until the data set becomes balanced. However, the SMOTE algorithm is nonselective in nature, where any of the minority class samples can be used for oversampling.

As an improvement, a selective oversampling technique has been presented in [35]. Since the boundary samples of the minority class are prone to get misclassified, in [35], the oversampled set of minority class samples has been constructed by synthetically generating data from these boundary samples. Selective oversampling of the minority class in [35] has also been seen to perform better than that of nonselective oversampling as done for SMOTE [25]. Another improvement to the SMOTE algorithm, termed the Modified Synthetic Minority Oversampling (MSMOTE) [26], performs adaptive mediation using K-nearest neighbor classifier to remove noisy data. Three strategies for rebalancing the training set as given in [23] are undersampling, random resampling, and a recognition-based induction scheme. In random undersampling, the majority class samples are undersampled randomly until their number becomes equal to that of the number of minority class samples. On the other hand, random resampling is done by sampling the minority class randomly until the number of samples in the two classes match each other.

The third category of approaches, namely, the ensemble-based approaches [27], [28], [36] fuses predictions from multiple classifiers by computing a weighted vote of the predictions. A variety of techniques for combining multiclassifier outputs exist in the literature, such as bagging, boosting, and majority voting [37]. AdaBoost learning algorithm [38] has often been used to improve classification accuracy by fusing the outputs from several weak classifiers. Performances of both SMOTE and MSMOTE have been seen to improve, once these are integrated with the AdaBoost learning algorithm. The resulting algorithms have been named SMOTEBoost [27] and MSMOTEBoost [26], respectively. A clustering-based subset ensemble learning technique has been described in [39], which clusters majority class samples into a fixed number of groups and subsequently reduces the number of majority class samples. Next, predictions from an ensemble of the classifiers, decision tree, Naive Bayes, K-nearest neighbor, and support vector machines, are fused by employing a combined bagging approach for final prediction.

Recently, Seiffert *et al.* [28] proposed the RUSBoost ensemble-based cost-sensitive classification algorithm, which uses the AdaBoost algorithm while combining the different classifier outputs. Instead of oversampling the minority class as in [25]–[27], in RUSBoost, the majority class samples are undersampled randomly. This causes the RUSBoost algorithm to be more time-efficient than that of SMOTEBoost. Experiments conducted on 15 different data sets in [28] have revealed that the performance of RUSBoost is marginally better than that of SMOTEBoost. An extensive survey of the commonly used data adjustment techniques for handling imbalanced data can also be found in [22]. In this paper, we use the random undersampling-based cost-sensitive learning strategy [23], which falls under the second category of cost-sensitive learning techniques, and obtain significantly high precision and recall in detecting each of the threat scenarios of the CMU Data.

Generally, insider threat scenarios do not materialize within a single day, but rather occur over a period of time. Hence, we felt it necessary to also review the important feature

extraction methods from time-series data. This is described in Section I-A3.

3) Review on Time-Series Analysis Methods: Depending on the data types, time-series data [40] can be classified into a number of categories: 1) discrete-valued and continuous-valued and 2) uniformly sampled and nonuniformly sampled [41]. While uniformly sampled time-series signals are easy to compare by means of standard distance metrics, comparison between nonuniformly sampled signals is difficult due to the lack of one–one correspondence among the sampled points. The dynamic time warping algorithm [42]–[44] is frequently used to compare nonuniformly sampled sequences, by finding matching points between sequences.

Shapelet-based features have been used for time-series classification in [45]–[47]. These approaches use the similarity index between a shapelet (subsequence of a time series) and a time series for classification. Ji *et al.* [48] proposed a pattern identification method called minimal distinguishing subsequence with the objective of estimating a minimal subsequence that occurs frequently in sequences of one class and infrequently in sequences of the other class. The gap constraint within the subsequences, as considered in this paper, makes it suitable for analyzing only protein sequence type of structures.

Other frequently used time-series analysis methods include the application of discrete Fourier transform or discrete wavelet transform [49]–[52], which transform the time-domain signal into frequency domain. A detailed survey of the techniques useful for time-series analysis as well as different frequency-domain representations of a time series can be found in [53]. In [54] and [55], the applicability of fractals in time-series prediction has been studied. Fractal dimension provides the intrinsic dimension of a signal in the hyperspace. A time-series signal, which undergoes very little or almost no variation across the sampled times, will have a fractal dimension close to one. As variability in the signal becomes higher, its intrinsic dimensionality increases. The book on *Data Mining with Computational Intelligence* by Wang and Fu [56] describes several time-series prediction and cost-sensitive classification techniques.

Statistical features extracted from a time series (for example, mean, standard deviation, skewness, kurtosis, and so on) preserve important structural properties of the series. In [57], a number of statistical features (including structural features, entropy, and scaling properties) have been used to do multivariate time-series clustering. Once these features are computed, a greedy forward feature selection strategy is employed to select a subset of features that best represents the time series. A book on time-series analysis by Chatfield [58] lists the different statistical and frequency-based properties that can be extracted from a time-series signal, such as Fourier transform, Laplace transform, Z-transform, spectral power analysis, and so on.

From the extensive literature survey on behavioral-based insider threat detection in Section I-A1, it can be seen that the unsupervised approaches are mostly anomaly-detection-based, which compute features depicting the variability of user activities over a period of time. However, a high anomaly score cannot guarantee the occurrence of a malicious activity. On the other hand, supervised approaches mostly compute frequency-based features and do not take into consideration the variation of insider behavior over time. In this paper, we combine the advantages of both the unsupervised and supervised techniques by following an algorithm based on the classification of time series of user activities. An overview of our approach and the main contributions of this paper are specified next in Section I-B.

B. Summary of Our Approach

As an initial step, we construct scenario-specific single-day features from the user activity logs. By single-day features, we mean features specific to a single day, such as the first logon time in a day, last logoff time in a day, or aggregated information of certain activities in a day, such as the number of computers accessed in a day, the number of external devices connected in a day, and so on. To preserve information about the variation of user behavior over a period of time, we compute a set of statistical measures from the time series of each single-day activity feature and concatenate them into a vector, which we term as the time-series feature vector. Classification of this time-series feature set is done after randomly undersampling the nonmalicious class samples to make the feature set balanced.

The main contributions of this paper can be summarized as follows.

1) This paper describes a method of classifying time series of user activities to predict whether an insider is malicious or not. As mentioned in Section I-A, previous methods to insider threat detection are mostly unsupervised, which just compute a set of anomaly scores to depict the variability observed in user behavior over time. In these approaches, unusual high variability in user activity over a time period is said to correspond to malicious behavior. However, a high variability in user behavior does not necessarily correspond to malicious activity. The issue has been addressed in this paper by first computing a set of anomaly scores from statistics of user activity over a time period. A statistical feature vector formed from these statistical parameters is next used to predict if an insider activity set can be mapped to some of the known threat scenarios.
2) A few supervised insider threat detection approaches developed till date carry out classification of single-day activities to predict if an insider is malicious or not. However, an insider usually does not exhibit radical changes in behavior within a single day. The gradual change in his/her behavior can be best captured by computing features from the time series of user activities. Evaluation of the proposed technique in Section III shows that the proposed algorithm outperforms the state-of-the-art insider threat detection approaches by a substantially high margin.
3) It is observed from the ground truth of CMU Insider Threat Data that although certain insider threat scenarios can occur within a few days, some other may need a larger time period to materialize. In Section III, we have described a method to determine the approximate time

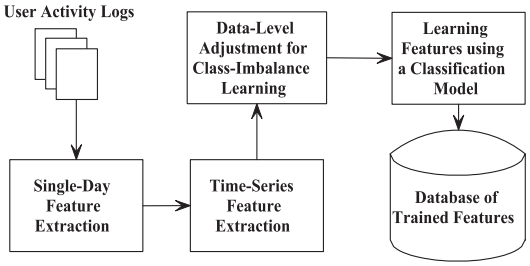


Fig. 1. Block diagram for building the classification model for insider threat detection.

window needed to analyze a given threat scenario from the available ground truth information.

4) Results show that the proposed insider threat detection algorithm is significantly accurate. It has the potential to suit the needs of any organization only with necessary modifications in the feature set.

The block diagram in Fig. 1 shows an overview of the sequence of steps followed in building the system. Further details of each of the blocks are given in Section II.

The rest of this paper is organized as follows. In Section II, the proposed algorithm is explained in two different sections. Section II-A describes the extraction of appropriate single-day features for identifying each threat scenario present in the CMU Insider Threat Data, whereas the construction of the time-series features from the above set of single-day features is discussed in Section II-B. Classifiers used and testing protocol along with experimental results are presented in Section III. Concluding remarks and scope for further research are finally pointed out in Section IV.

II. Time-Series Classification of User Activities

Here, we describe the proposed feature extraction technique to identify the different threat scenarios of the CMU Data (refer to Section I for detailed specification of the scenarios).

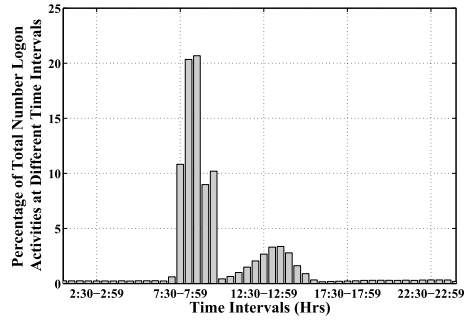
A. Extraction of Single-Day Features

The single-day features extracted from the CMU Data are listed in Table II. The first column of the table shows the activity type, the second column specifies the features extracted from each activity log along with the corresponding symbols, while the last three columns indicate, which features are appropriate for identifying the threat scenarios. Symbols "$\surd$" and "$\times$" in each row represent whether a feature is appropriate to identify a threat scenario or not. It needs to be mentioned here that the set of features in Table II are specific to the CMU Insider Threat Data and its associated scenarios. While working on a different data set, these features must be updated accordingly. However, the overall algorithm described in this paper (refer to Fig. 1) has the potential to work effectively on any given insider threat data set.

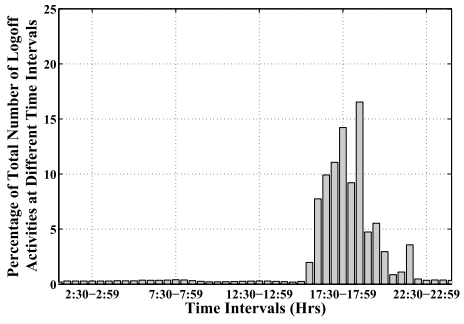
With reference to Table II, the extraction of first 18 single-day features is straightforward given the logon, device, file, e-mail, and http user activity logs. However, to compute the features L_1–L_4, L_6, L_8, L_9, and D_1, shown in the second column of the table, knowledge of the office start and end times is needed, which is not provided by the CMU Data. The logon log-file from this data set (which contains the logon and logoff times for all users) is used to obtain an estimate of the office start and end times. The percentage of the total number of logon activities at consecutive time intervals of 30 min starting from 12:01 to 0:00 h is provided in Fig. 2(a). Fig. 2(b), on the other hand, shows the percentage of logoff activities corresponding to the same time intervals.

TABLE II
Single-Day Features

Activity Logs	List of Single-Day Features	S_1	S_2	S_3
Logon	1. Difference of office start time from first login time (L_1)	$\surd$	$\times$	$\surd$
	2. Difference of last login time from office end time (L_2)	$\surd$	$\times$	$\surd$
	3. Average time difference between office start time and login times before office hours (L_3)	$\surd$	$\times$	$\surd$
	4. Average time difference between office end time and login times after office hours (L_4)	$\surd$	$\times$	$\surd$
	5. Total number of sessions (or, number of logins) (L_5)	$\surd$	$\times$	$\surd$
	6. Number of sessions (i.e., number of logins) outside office hours (L_6)	$\surd$	$\times$	$\surd$
	7. Total number of computers accessed (L_7)	$\surd$	$\times$	$\surd$
	8. Number of computers accessed outside office hours (L_8)	$\surd$	$\times$	$\surd$
	9. Average session duration outside office hours (L_9)	$\surd$	$\times$	$\surd$
Device	10. Number of times a thumb drive used outside office hours (D_1)	$\surd$	$\surd$	$\surd$
	11. Total number of times a device is used (D_2)	$\surd$	$\surd$	$\surd$
File	12. Number of .exe files downloaded (F_1)	$\times$	$\times$	$\surd$
Email	13. Number of emails sent outside the organization domain (E_1)	$\times$	$\surd$	$\times$
	14. Number of emails sent within the organization domain from supervisor's account (E_2)	$\times$	$\times$	$\surd$
	15. Number of attachments (E_3)	$\times$	$\times$	$\surd$
	16. Average email size (E_4)	$\times$	$\times$	$\surd$
	17. Number of recipients (E_5)	$\times$	$\times$	$\surd$
Http	18. Number of times *wikileaks.org* is visited (H_1)	$\surd$	$\times$	$\times$
	19. *TF-IDF* based feature for job-advertisement related web-pages (H_2)	$\times$	$\surd$	$\times$
	20. *TF-IDF* based feature for key-logger downloading sites (H_3)	$\times$	$\times$	$\surd$



(a)



(b)

Fig. 2. Bar diagram showing (a) percentage of the total number of logon activities and (b) percentage of the total number of logoff activities at different time intervals of 30 min from CMU Data version 4.2.

It is seen from Fig. 2(a) that more than 10% of the total number of logon activities occur within the time interval 7:30–7:59 h. The previous interval (i.e., 7:00–7:29 h) shows a much smaller percentage of logon activities, whereas the next two intervals (i.e., 8:00–8:29 and 8:30–8:59 h) show the highest percentage of logon activities. Thus, office start time can be suitably considered to be 8:00 h (i.e., 8:00 A.M.). Using a similar argument, the office end time can be chosen as 19:00 h (i.e., 7:00 P.M.) from Fig. 2(b).

The last two features, namely, H_2 and H_3, are http log-based features and are computed from the term frequency–inverse document frequency (*TF-IDF*) scores[3] of the keywords in a visited web page. Identification of CMU threat scenario S_2 (refer to Section I for a description of the threat scenarios) requires information about the frequency of visit to job websites, whereas for detecting scenario S_3, the frequency of visiting key generator downloading sites is an important cue. However, CMU Data do not specify the type of website that is being visited. In the absence of this information, the ground truth http log file can be used to prepare a corpus of documents, say $\mathcal{D}_1$ and $\mathcal{D}_2$, consisting of keywords from the web pages visited by malicious and nonmalicious users, respectively. Thus, in the corpus for scenario S_2, $\mathcal{D}_1$ mostly consists of terms related to job opportunities. On the other hand for scenario S_3, $\mathcal{D}_1$ contains terms, such as keylogger, password, crack, and so on. Given a new web page, the *TF-IDF* score of the keywords in this page can reflect whether the page is similar to $\mathcal{D}_1$ or $\mathcal{D}_2$. A higher similarity score for $\mathcal{D}_1$ than $\mathcal{D}_2$ indicates the possibility of a suspicious web-activity.

The http log of the CMU Data provides a keyword list for each visited page, which we directly use to compute the *TF-IDF* features. If only the URLs of the visited web pages were captured in the activity logs, instead of the keywords, standard keyword extraction algorithms [59], [60] could have been employed to do this task. The Porter stemming algorithm [61] is used to remove suffixes and find the base form of each word in $\mathcal{D}_1$ and $\mathcal{D}_2$. Suppose a test page consists of n keywords, $w_1, w_2, \ldots w_n$. Furthermore, let $f_{w_j,\mathcal{D}}$ be the frequency of word w_j in document $\mathcal{D}$, where $j = 1, 2, \ldots, n$, and $\mathcal{D} \in \{\mathcal{D}_1, \mathcal{D}_2\}$. The term frequency $[tf(w_j, \mathcal{D})]$ of word w_j in document $\mathcal{D}$ is given by the normalized frequency of this word in document $\mathcal{D}$, i.e.,

$$tf(w_j, \mathcal{D}) = \frac{f_{w_j,\mathcal{D}}}{|\mathcal{D}|} \tag{4}$$

where $|\mathcal{D}|$ represents the number of words in document $\mathcal{D}$. The inverse document frequency $[idf(w_j)]$ of term w_j in the corpus is mathematically expressed as follows:

$$idf(w_j) = \frac{\mathcal{N}}{1 + |\mathcal{D} \in \{\mathcal{D}_1, \mathcal{D}_2\} : w_j \in \mathcal{D}|}. \tag{5}$$

In (5), the numerator $\mathcal{N}$ denotes the number of documents in the corpus (in the present case $\mathcal{N} = 2$, since there are two documents $\mathcal{D}_1$ and $\mathcal{D}_2$), whereas the denominator is one added with the number of documents in which the term w_j appears at least once. Addition with one avoids division by zero in case the term does not appear in any of the documents in the corpus. The *TF-IDF* score of term w_j in a document $\mathcal{D}$ within the corpus is henceforth computed as

$$\text{TF-IDF}(w_j, \mathcal{D}) = tf(w_j, \mathcal{D}) idf(w_j). \tag{6}$$

The magnitude of this statistic increases proportionally with the frequency of occurrence of the word in a document, but is offset by its frequency in the corpus. From (6), it can be seen that the value of *TF-IDF*$(w_j, \mathcal{D})$ will be zero if w_j is present in both the documents or does not belong to any of them. A similarity score ($\mathcal{S}_\mathcal{D}$) of the web page with respect to document $\mathcal{D}$ is computed by summing up the *TF-IDF* values of each word present in the page, as shown in (7):

$$\mathcal{S}_\mathcal{D} = \sum_{j=1}^{n} \textit{TF-IDF}(w_j, \mathcal{D}). \tag{7}$$

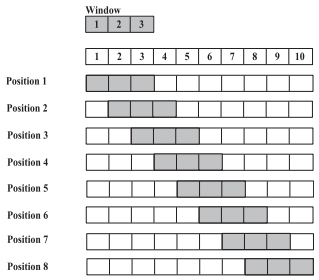


Fig. 3. Sliding window mechanism.

B. Construction of the Time-Series Feature Vector

Suppose the data set contains user activity logs captured over a period of T days. Scenario-based insider threat detection is done by analyzing appropriate user activity logs (refer to Table II) in small time intervals over the entire length of the time line. We employ a time window of length W (where $W \leq T$) and slide it over the time line, shifting the window by one day at each step, as shown in Fig. 3. The figure explains the sliding window mechanism with $W = 3$ days and $T = 10$ days. In the figure, each small-sized box represents a single day. For a time line of 10 days, there are 10 such boxes. The gray region on the time line represents the days covered by the time window at a particular position. There will be a maximum number of $(T-W+1)$ different positions of the window on the time line. Separate time-series feature vectors are computed for each window position.

Since the single-day features are computed from different user activities, their magnitudes are not comparable. Therefore, before computing the time-series feature vector, we divide each single-day feature attribute with the maximum value of the attribute within the time window to normalize it within the range *(0,1)*. In the rest of this paper, by single-day feature attributes, we will refer to the normalized single-day feature attributes.

1) Statistical Measures: To construct the time-series feature vector, we compute a set of statistical measurements corresponding to each single-day feature attribute for each window position. The time-series feature vector for the

[3]https://en.wikipedia.org/wiki/Tf%E2%80%93idf

dth day is computed from the statistics of single-day features corresponding to the previous $(d-W+1)$ days, inclusive of the dth day, where $d = W, W+1, \ldots T$. The statistical measurements computed from the time series of each single-day feature are: 1) mean; 2) variance; 3) Katz fractal dimension; and 4) total power corresponding to the top five frequencies in the power spectrum of the time-series signal.

Mean provides the average value of an attribute within a time window, whereas variance shows the amount of dispersion in the data. Fractal dimension [54], [55], [62] is a statistical index measuring the complexity (i.e., the amount of details) of a time series. Suppose that K single-day feature attributes are required to detect a threat scenario (refer to Table II). Let $[X_{i1}\ X_{i2}\ \ldots\ X_{iW}]$ be the feature vector containing the magnitudes of a certain single-day feature attribute X_i at a particular position of the time window, $i = 1, 2, \ldots, K$. The mean $(\mathcal{M}_i)$ and variance $(\mathcal{V}_i)$ of the attribute i are computed according to the following two equations:

$$\mathcal{M}_i = \frac{1}{W}\sum_{j=1}^{W} X_{ij} \tag{8}$$

$$\mathcal{V}_i = \frac{1}{W}\sum_{j=1}^{W}(X_{ij} - \mathcal{M}_i)^2. \tag{9}$$

The Katz fractal dimension $\mathcal{F}_i$ of the time series of this attribute is computed as

$$\mathcal{F}_i = \frac{\log_{10}\left(\frac{Z}{\mathcal{Z}}\right)}{\log_{10}\left(\frac{\mathfrak{D}}{\mathcal{Z}}\right)} \tag{10}$$

where Z is the sum of the distances between each pair of consecutive points, $\mathcal{Z}$ is the average distance between successive point pairs, and $\mathfrak{D}$ is the diameter of the time series computed as

$$\mathfrak{D} = \max_{2\le j\le W}\{|X_{i1} - X_{ij}|\}. \tag{11}$$

The magnitude of the Katz fractal dimension is close to one when the time series has almost uniform value throughout the time window. It increases in magnitude as more and more variability is introduced into the time series, i.e., when the behavior changes from malicious to nonmalicious or from nonmalicious to malicious.

The time-series feature vector $[X_{i1}\ X_{i2}\ \ldots\ X_{iW}]$ corresponding to the single-day feature attribute X_i can also be viewed as a time-varying signal. We first decompose the signal into its different frequency components and obtain its equivalent frequency-domain representation, i.e., periodogram, using discrete Fourier transformation, as given by (12). Let us denote the periodogram corresponding to X_i as $\mathcal{X}_i(\omega)$

$$\mathcal{X}_i(\omega) = \sum_{w=1}^{W} X_{iw}\exp^{-j\frac{2\pi(\omega-1)(w-1)}{W}} \tag{12}$$

where $\omega \in \mathbb{Z}$ (the set of integers). The power of the signal at each frequency is given by

$$\mathcal{P}_i(\omega) = \mathcal{X}_i(\omega)\overline{\mathcal{X}_i(\omega)} \tag{13}$$

where $\overline{\mathcal{X}_i(\omega)}$ is the complex conjugate of $\mathcal{X}_i(\omega)$. If a user activity remains almost uniform throughout the time window, the power at the high-frequency components of the periodogram will be nil. On the other hand, if significant variations are present in the user activity within the time window, the power at the high-frequency components of the signal will be greater. As features, we consider the total power at the top five frequency components from the periodogram.

Suppose $\mathcal{M}_i$, $\mathcal{V}_i$, $\mathcal{F}_i$, and $\mathcal{E}_i$, respectively, denote the mean, variance, Katz fractal dimension, and spectral power density of the time series of the feature attribute X_i. The time-series feature vector V_i for this attribute is constructed by concatenating the above statistical measures, as shown in (14):

$$V_i = [\mathcal{M}_i\ \ \mathcal{V}_i\ \ \mathcal{F}_i\ \ \mathcal{E}_i]. \tag{14}$$

The time-series feature vectors corresponding to the K single-day feature attributes are concatenated to construct the final feature vector for identifying a threat scenario. If V denotes this vector, then

$$V = [V_1\ \ V_2\ \ \ldots\ \ V_K]. \tag{15}$$

With reference to Table II, 12 single-day features are needed to identify scenario S_1. The time-series feature vector for this scenario will thus have a total dimensionality of 48. Likewise, the dimensionality of the time-series feature vectors for scenarios S_2 and S_3 is 16 and 68, since the corresponding single-day feature vectors are of dimensionality 4 and 17, respectively. The complete time-series feature vector preserves useful dynamics of a user behavior within the time window that cannot be captured by single-day features alone. The imbalanced time-series feature set is next balanced by randomly undersampling the nonmalicious class instances to achieve a desired imbalance ratio. Imbalance ratio is defined as the ratio of the number of majority class samples (i.e., nonmalicious class) to the number of minority class samples (i.e., malicious class) [63].

2) Selection of Appropriate Time-Window Length: Malicious insiders do not execute threatful activities every day. They tend to perform abnormally for a very short period of time (typically for a few days) and start behaving normally after that. With reference to Fig. 3 and the associated discussions in the first paragraph of Section II-B, if $W = T$, then the window covers the entire length of the time line. Since T is usually large, it would be difficult to precisely figure out at which period of time certain insider threat-related activities have occurred. On the contrary, the use of a smaller time window will help in getting such information more precisely. However, W cannot be too small either, since in that case the behavioral changes of an insider from malicious to nonmalicious, or from nonmalicious to malicious, might not get accurately captured. Corresponding to each threat scenario, the minimum window length for which each of the performance metrics, precision, recall, and f-score, is greater than or equal to γ $(0\le\gamma\le1)$ is chosen as the optimal window length. This condition helps in selecting the classifier configuration with a less number of false positive cases and a high number of true positive cases. In Section III, we will discuss more on this issue with suitable experimental results.

TABLE III

NUMBER OF NONMALICIOUS AND MALICIOUS SAMPLES IN THE ORIGINAL TIME-SERIES FEATURE SET FOR THE DIFFERENT WINDOW LENGTHS AND FOR DIFFERENT SCENARIOS ALONG WITH THE NUMBER OF UNDERSAMPLED NONMALICIOUS INSTANCES FOR THE DIFFERENT IMBALANCE RATIOS *C1–C7*

Scenario	Window Length	Number of Malicious Instances	Number of Non-malicious Instances	Under-sampled Non-malicious Samples						
				C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*
S_1	5	271	325997	135771	81571	54471	27371	13821	271	136
	10	421	320487	210921	126721	84621	42521	21471	421	211
	20	497	310771	248997	149597	99897	50197	25347	497	249
	40	498	290838	249498	149898	100098	50298	25398	498	249
S_2	5	60	325696	30060	18060	12060	6060	3060	60	30
	10	72	320684	36072	21672	14472	7272	3672	72	36
	20	72	310684	36072	21672	14472	7272	3672	72	36
	40	72	290753	36072	21672	14472	7272	3672	72	36
S_3	5	12	335300	6012	3612	2412	1212	612	12	6
	10	12	330300	6012	3612	2412	1212	612	12	6
	20	12	320300	6012	3612	2412	1212	612	12	6
	40	12	300057	6012	3612	2412	1212	612	12	6

III. EXPERIMENTAL RESULTS

We evaluate this paper through extensive experiments using the CMU Data version 4.2 and present the important findings. All simulations are carried out in MATLAB 2017a on a system having 2.4-GHz Intel Xeon Processor [E5-2640 (v4)] and 128-GB RAM. As a classifier, we employ a deep autoencoder neural network and compare its performance with other commonly used classifiers, namely, random forest [64] and multilayer perceptron [65].

Recently, autoencoders have been extensively used as tools for learning deep neural networks [66], [67]. The hidden/encoding layers of a deep autoencoder are trained separately in an unsupervised manner to learn the input features at different levels of abstraction. The final layer of the deep network is a soft-max classification layer, which is trained in a supervised manner using the outputs of the last encoding layer. Let us assume that a K-dimensional feature vector $[V_1\ V_2\ \ldots\ V_K]$ is provided as input to the deep autoencoder, and $[\widehat{V_1}\ \widehat{V_2}\ \ldots\ \widehat{V_K}]$ is the corresponding desired output vector. If N denotes the total number of samples in the feature set, the cost function for training each encoding layer of the deep network is as follows:

$$\mathcal{L}_1 = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}(V_{kn} - \widehat{V_{kn}})^2 + \beta\Omega_{\text{sparsity}} \tag{16}$$

where β denotes the coefficient for the sparsity regularization term and Ω_{sparsity} is the sparsity regularization term given by

$$\Omega_{\text{sparsity}} = \sum_{i=1}^{\mathcal{H}}\left(\rho\log\left(\frac{\rho}{\widehat{\rho_i}}\right) + (1-\rho)\log\left(\frac{1-\rho}{1-\widehat{\rho_i}}\right)\right) \tag{17}$$

where ρ_i is the average activation value at a neuron and ρ is its desired value. Cross-entropy loss function is next used to train the final soft-max classification layer using the features at the last encoding layer.

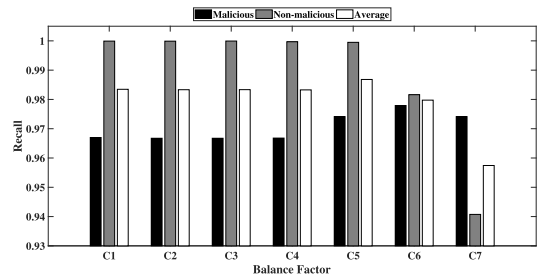
The deep learning toolbox[4] of MATLAB 2017a is used to construct a two-layer deep autoencoder neural network and train it with the data-adjusted time-series feature set. For each insider threat scenario (refer to Section I), we compute the performance metrics (refer to Section I-A2) by varying the length of the time window and also by reducing the imbalance ratio gradually. We experiment with the window lengths of 5, 10, 20, and 40 days, and for each of these window lengths, the nonmalicious class is randomly undersampled to the following imbalance ratios: $C1 = 500$, $C2 = 300$, $C3 = 200$, $C4 = 100$, $C5 = 50$, $C6 = 1$, and $C7 = 0.5$. Table III shows the original number of malicious and nonmalicious instances in the time-series feature set for each scenario corresponding to each window length as well as the number of undersampled nonmalicious class samples for the different imbalance ratios. The data-adjusted time-series feature set is next used for training the deep autoencoder neural network. Any instance of this time-series feature set is labeled as malicious, if certain malicious activity has occurred within the corresponding time window.

To choose the optimum network parameters through experimentation, we vary the number of neurons in the hidden layers of the deep autoencoder from 5 to $2K$ (where K is the dimensionality of the feature set) in the steps of 5, with the constraint that the second hidden layer has a lesser number of neurons than the first one. Fivefold cross validation is carried out, and finally, the optimal window length for each scenario is determined from the values of average recall $\mathcal{R}_{\text{AV}}$, average precision $\mathcal{P}_{\text{AV}}$, and average f-score $\mathcal{F}_{\text{AV}}$, following the condition stated in Section II-B2. In this paper, we consider γ to be 0.8.

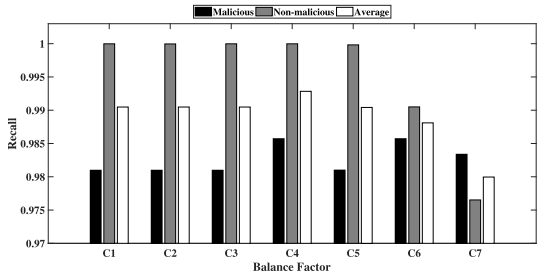
Fig. 4(a)–(d) shows the recall for malicious class $\mathcal{R}_M$ (black bars), recall for nonmalicious class $\mathcal{R}_{\text{NM}}$ (gray bars), and average recall $\mathcal{R}_{\text{AV}}$ (white bars) corresponding to scenario S_1 for each experimental setup, as described earlier. Similarly, Figs. 5(a)–(d) and 6(a)–(d) show the precision ($\mathcal{P}_M$, $\mathcal{P}_{\text{NM}}$, and $\mathcal{P}_{\text{AV}}$) and f-score ($\mathcal{F}_M$, $\mathcal{F}_{\text{NM}}$, and $\mathcal{F}_{\text{AV}}$) for the two classes along with their averages. It is seen from the figures that for scenario S_1, significantly high values of precision, recall, and f-score (all greater than γ) are obtained for a small window length of only five days. Thus, the optimal window length is set to five days. It is also observed that, for this window length, the best value of $\mathcal{R}_M$ is obtained when the imbalance ratio is set to *C5*. The averages of the recall, precision, and f-score for the chosen window length and imbalance ratio are 0.9868, 0.9870, and 0.9742, respectively.

Figs. 7(a)–(d), 8(a)–(d), and 9(a)–(d) show the recall, precision, and f-score values for the malicious and the nonmalicious classes along with their average values for scenario S_2.

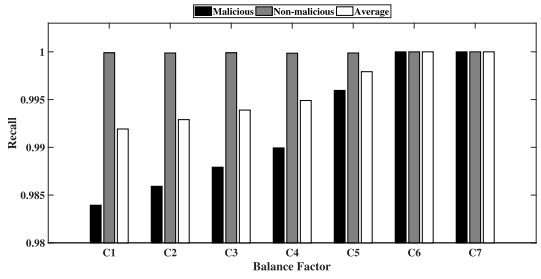
[4]https://www.mathworks.com/discovery/deep-learning.html



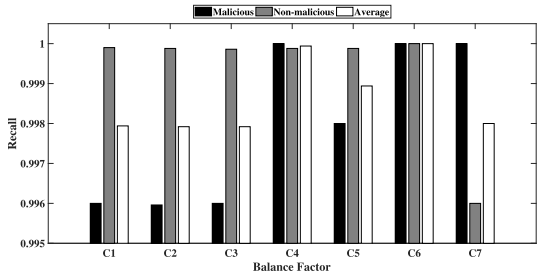
(a)



(b)

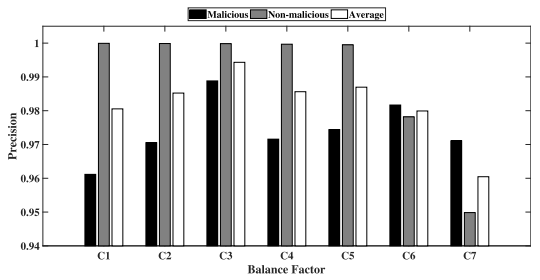


(c)

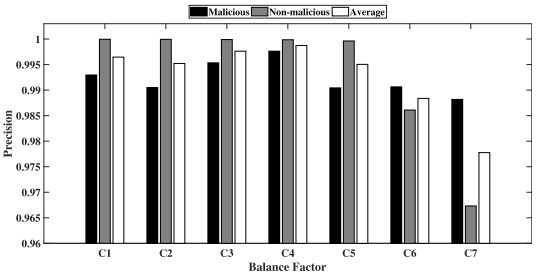


(d)

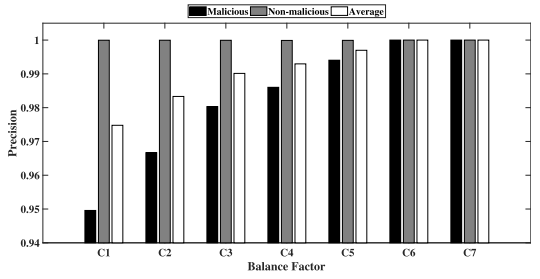
Fig. 4. Scenario S_1 detection: variation of recall for malicious class, recall for nonmalicious class, and average recall as imbalance ratio decreases from C1 to C7 for different window lengths. (a) Window length = 5 days. (b) Window length = 10 days. (c) Window length = 20 days. (d) Window length = 40 days.



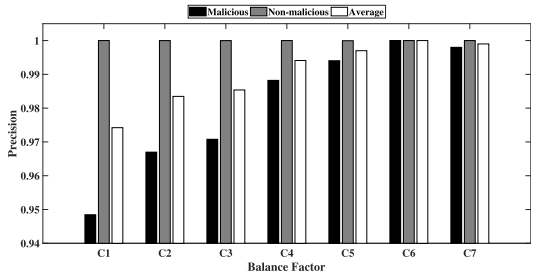
(a)



(b)



(c)



(d)

Fig. 5. Scenario S_1 detection: variation of precision for malicious class, precision for nonmalicious class, and average precision as imbalance ratio decreases from C1 to C7 for different window lengths. (a) Window length = 5 days. (b) Window length = 10 days. (c) Window length = 20 days. (d) Window length = 40 days.

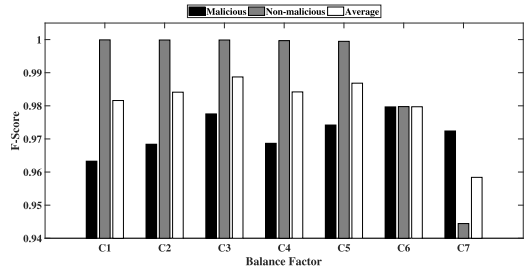
It is observed from the figures that each of the performance metrics is greater than γ for all the imbalance ratios, if the window length is set to 40 days. The best recall for the malicious class at this particular window size is obtained for the imbalance ratio *C7*. For this particular combination of window length and imbalance ratio, each of the metrics, $\mathcal{R}_{\text{AV}}$, $\mathcal{P}_{\text{AV}}$, and $\mathcal{F}_{\text{AV}}$, attains the maximum possible value, i.e., 100%.

We also carry out similar experiments for scenario S_3 and found that the optimal window length and the imbalance ratio for identifying this scenario are 10 days and *C6*, respectively. For this configuration, each of the above metrics attains a value of 100%. Table IV shows the optimal time-window lengths

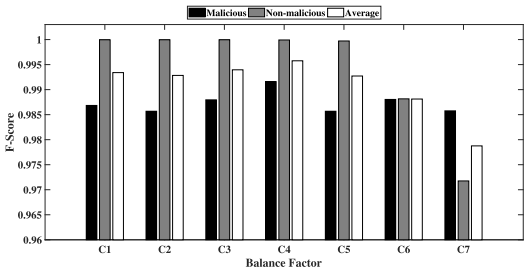
TABLE IV

OPTIMAL TIME-WINDOW LENGTHS AND IMBALANCE RATIOS FOR THE CMU DATA THREAT SCENARIOS

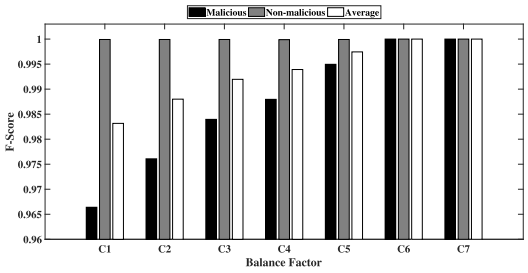
Scenario	Optimal Window Length (Days)	Optimal Imbalance Ratio
S_1	5	C_5
S_2	40	C_7
S_3	10	C_6



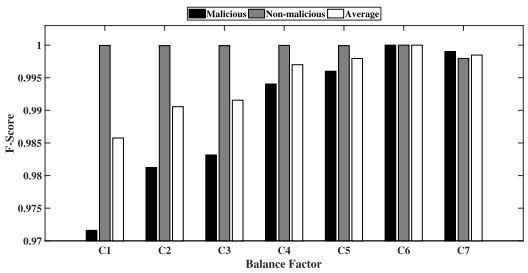
(a)



(b)

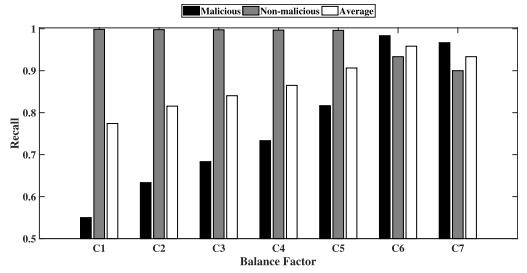


(c)

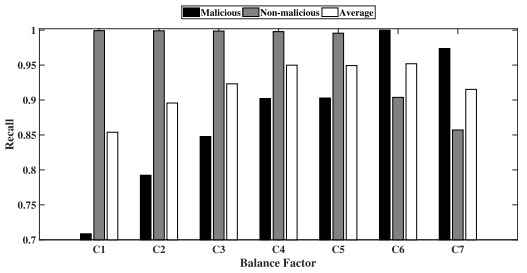


(d)

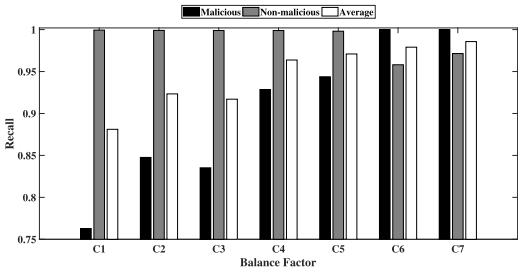
Fig. 6. Scenario S_1 detection: variation of f-score for malicious class, f-score for nonmalicious class, and average f-score as imbalance ratio decreases from C1 to C7 for different window lengths. (a) Window length = 5 days. (b) Window length = 10 days. (c) Window length = 20 days. (d) Window length = 40 days.



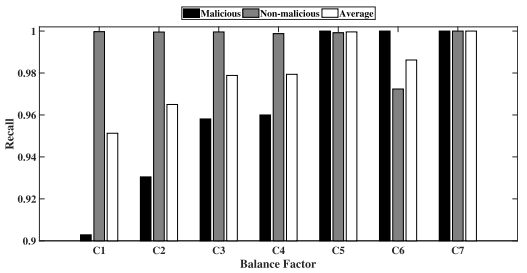
(a)



(b)



(c)

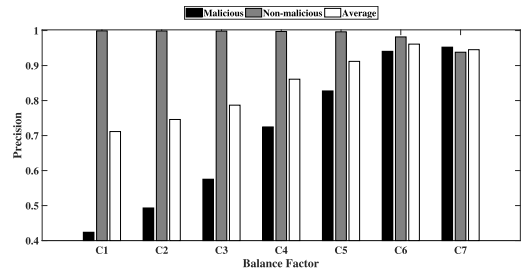


(d)

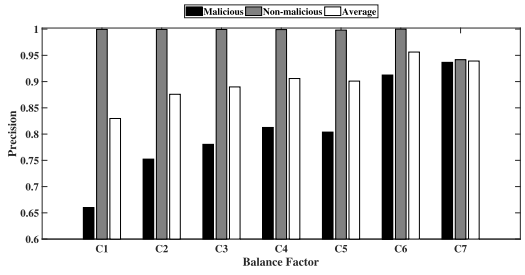
Fig. 7. Scenario S_2 detection: variation of recall for malicious class, recall for nonmalicious class, and average recall as imbalance ratio decreases from C1 to C7 for different window lengths. (a) Window length = 5 days. (b) Window length = 10 days. (c) Window length = 20 days. (d) Window length = 40 days.

and the imbalance ratios for the three scenarios determined in the previous experiment. It may be noted that the criterion for optimal window length selection (described in Section II-B2) could have been satisfied for some intermediate window length as well (e.g., 15 days or 25 days). But for most practical purposes, such a minor difference in the lengths of the time window will not significantly affect the localization of an insider threat event.

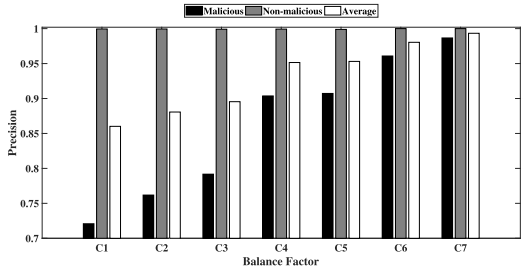
An important observation from Figs. 4 and 7 is that, for a fixed window length, as the imbalance ratio is gradually decreased from C_1 to C_7, the value of $\mathcal{R}_M$ tends to increase. However, the magnitude of $\mathcal{R}_{\text{NM}}$ falls if the imbalance ratio is made very low. This is due to the fact that, with the reduction of imbalance ratio, a more number of nonmalicious instances get eliminated, and the classifier learns to predict the malicious class with higher accuracy, whereas its prediction accuracy on the nonmalicious class reduces gradually. The phenomenon is more prominent for small values of the window lengths than for higher values. With reference to Table III, for large time-window lengths, the number of malicious samples in the feature set is more compared to that for small lengths of the time window. Thus, when the time-window size is set to a



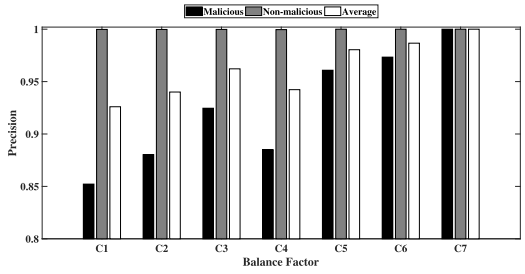
(a)



(b)

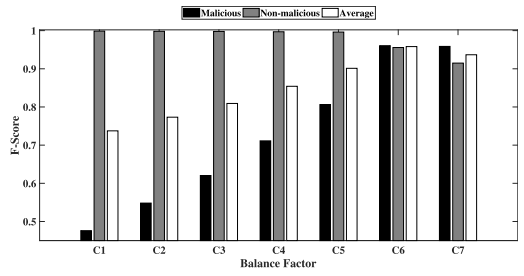


(c)

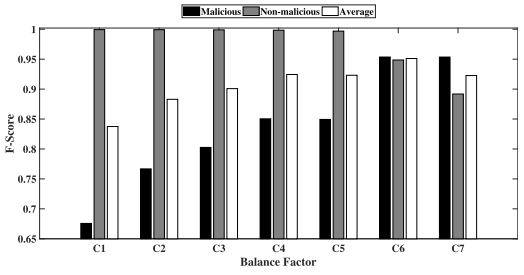


(d)

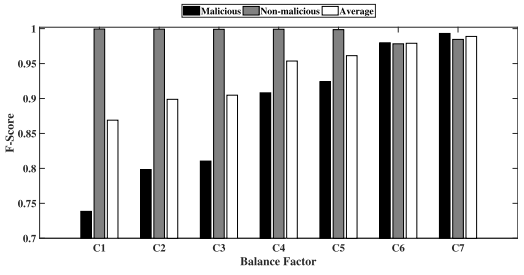
Fig. 8. Scenario S_2 detection: variation of precision for malicious class, precision for nonmalicious class, and average precision as imbalance ratio decreases from C1 to C7 for different window lengths. (a) Window length = 5 days. (b) Window length = 10 days. (c) Window length = 20 days. (d) Window length = 40 days.



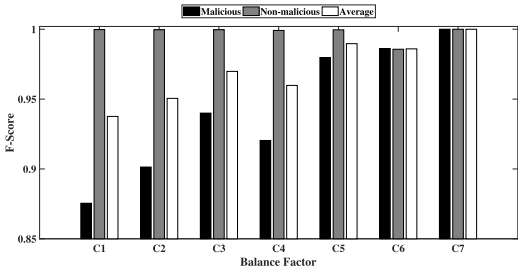
(a)



(b)



(c)



(d)

Fig. 9. Scenario S_2 detection: variation of f-score for malicious class, f-score for nonmalicious class, and average f-score as imbalance ratio decreases from C1 to C7 for different window lengths. (a) Window length = 5 days. (b) Window length = 10 days. (c) Window length = 20 days. (d) Window length = 40 days.

large value, the classifier shows a higher value of $\mathcal{R}_M$ even for higher values of imbalance ratio.

As explained before, time-series features are more appropriate for identifying insider threat scenarios than single-day features. To experimentally verify this fact, we compare the overall classification performance of deep autoencoder when trained with both these types of feature sets. Average f-score ($\mathcal{F}_{\text{AV}}$) is an appropriate metric to measure the overall classification performance. So, we present this metric for the three scenarios and also for the different imbalance ratios C_1, C_2, …, C_7 in Fig. 10(a)–(c). The optimal window length corresponding to each scenario specified in Table IV has been used to obtain the above results.

As expected, it is observed from the figures that the time-series-based classification of user activities outperforms single-day classification. The difference in the average f-score values is small for each of scenarios S_1 and S_3, since in this case insiders execute malicious activities mostly within a single day. However, this difference is substantially high for scenario S_2, in which there is a gradual change in insider behavior over time.

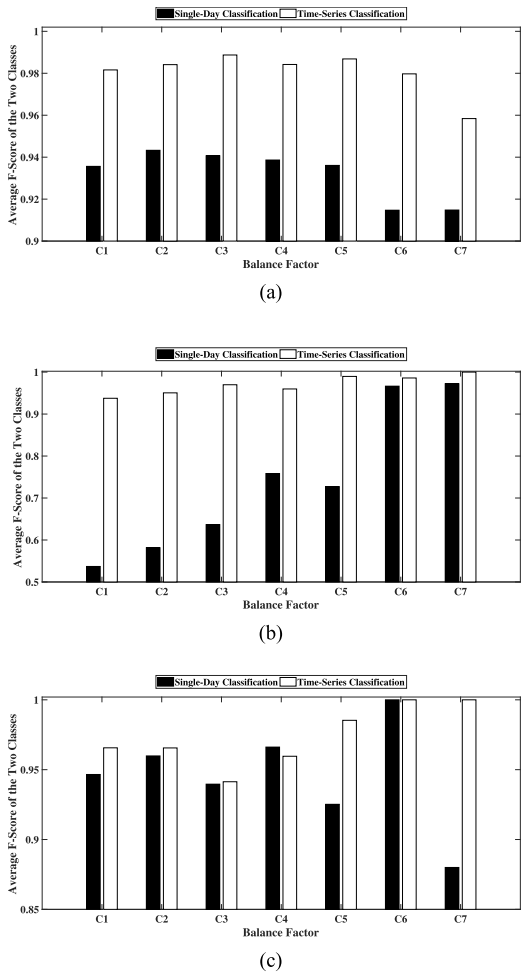


Fig. 10. Comparison between the single-day classification and the time-series classification of user activities in terms of average f-score for scenarios (a) S_1, (b) S_2, and (c) S_3.

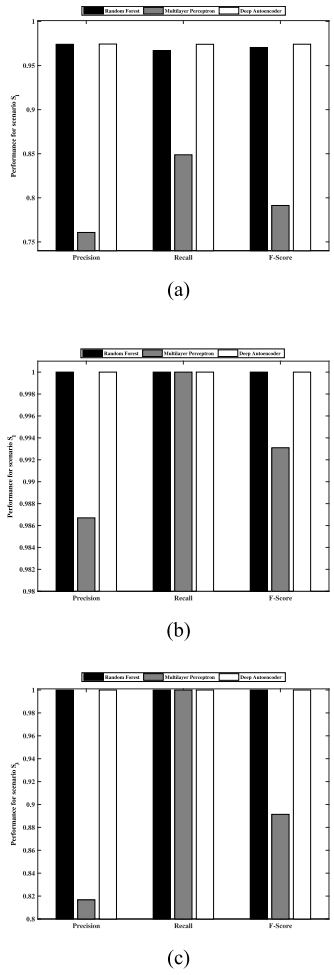


Fig. 11. Comparative study of the performances of three classifiers: random forest, multilayer perceptron, and deep autoencoder on the time-series feature set for scenarios (a) S_1, (b) S_2, and (c) S_3.

We also make a comparative study of the performances of the deep autoencoder neural network with two other popularly used classifiers, namely, random forest[5] and multilayer perceptron with backpropagation.[6] In Fig. 11(a)–(c), we present the precision, recall, and f-score for the malicious class corresponding to each of the three scenarios. The optimal values for the window length and the imbalance ratio (refer to Table IV) are also used in this experiment. In each figure, the bars corresponding to the random forest, multilayer perceptron, and deep autoencoder classifiers are colored with "black," "gray," and "white," respectively. Each classifier is subjected to fivefold cross validation repeatedly by varying the classifier parameters. While carrying out classification using multilayer perceptron, we vary the number of hidden layer neurons from 5 to $2K$ in the steps of 5, and for each of these network configurations, four different learning rates 0.05, 0.25, 0.5, and 1 are considered. In the case of random forest classifier, we vary the number of decision trees in the forest from 10 to 150 in the steps of 10, and for each of these ensembles, the same four learning rates are considered, as mentioned earlier. Performance metrics presented in Fig. 11(a)–(c) are computed from the network configuration that provides the best value of the metric $\mathcal{R}_M$. From the figures, it is observed that the deep autoencoder classifier performs with significantly high precision, recall, and f-score. The performance metrics for the random forest classifier are also reasonably high, and for scenarios S_2 and S_3, it is comparable with that of the deep autoencoder. On the other hand, multilayer perceptron has a lower precision than either of the other two classifiers used in this paper. The reason for this is explained next. High-dimensional data are difficult to classify, and both deep autoencoder and random forest reduce the original high-dimensional data into smaller dimensions. While deep autoencoder aims to find a smaller representation of the original feature vector, each tree within a random forest works on different subsets of the original attribute set. In contrast, multilayer perceptron performs classification using the complete set of feature attributes, and hence, suffers from poor classification performance.

Majority of the existing approaches on insider threat detection aim at detecting only anomalousness, and users exhibiting high degree of anomaly score are suspected as probable insiders. None of the existing techniques on insider threat detection has performed threat scenario-based classification on time series of user activities. The two approaches closest to this paper are [15], which carried out cost-sensitive

[5]https://www.mathworks.com/help/stats/treebagger.html
[6]https://www.mathworks.com/help/nnet/ref/train.html

TABLE V
COMPARATIVE STUDY WITH EXISTING TECHNIQUES

Scenario	Method	Percentage of Training Data (%)								
		30			50			70		
		$\mathcal{P}_{AV}$ (%)	$\mathcal{R}_{AV}$ (%)	$\mathcal{F}_{AV}$ (%)	$\mathcal{P}_{AV}$ (%)	$\mathcal{R}_{AV}$ (%)	$\mathcal{F}_{AV}$ (%)	$\mathcal{P}_{AV}$ (%)	$\mathcal{R}_{AV}$ (%)	$\mathcal{F}_{AV}$ (%)
S_1	[15]	50.02	54.01	51.94	50.01	51.44	50.72	50.01	51.29	50.64
	[7]	50.71	69.43	58.61	50.95	70.78	59.25	51.44	82.09	63.25
	Random Forest	50.61	83.48	63.02	50.82	88.18	64.48	51.24	89.58	65.19
	Deep Autoencoder	50.16	73.41	59.60	50.31	83.00	62.64	50.42	90.25	64.69
S_2	[15]	50.00	51.32	50.65	50.00	50.11	50.05	50.01	54.60	52.20
	[7]	62.35	79.68	69.96	70.96	91.61	79.98	71.71	95.68	81.98
	Random Forest	95.88	80.75	87.67	99.21	84.12	91.04	99.34	85.94	92.16
	Deep Autoencoder	89.52	98.97	94.01	91.76	99.14	95.31	92.88	99.48	96.06
S_3	[15]	50.00	52.17	51.06	50.00	49.74	49.87	50.00	49.04	49.52
	[7]	50.11	59.97	54.60	50.17	62.82	55.79	50.54	71.08	59.07
	Random Forest	50.31	77.90	61.14	51.67	92.62	66.34	52.14	96.35	67.66
	Deep Autoencoder	50.16	86.12	63.40	50.15	88.50	64.02	50.18	94.10	65.46

classification on single-day features, and [7], which computed time-series features from the first-order statistical measurements and employed isolation forest algorithm to obtain the anomaly scores. To verify the effectiveness of the proposed algorithm, we compare it with both these techniques. The best two classifiers obtained in our previous study, i.e., random forest and deep autoencoder, have been used in the comparison.

The protocol followed here is different from the previous experiments. We randomly select a percentage (say $r\%$) of samples from the data set as training set. Hyperparameters of each classifier are chosen from the configuration that provides the minimum cross-validation error. Experiments have been conducted with the values of $r = 70$, 50, and 30, and the results are presented in Table V.

It can be seen from the table that the proposed approach always outperforms both [15] as well as [7] by a significantly high margin in terms of average recall and average f-score. The effect is more pronounced for scenario S_2 than that for S_1 and S_3, since, in the case of S_2, user behavior changes gradually, and higher order statistical measurements are needed to preserve the dynamics of this change at a high resolution. The fact that scenario S_2 in the CMU data deals with gradual and slow changes in user behavior is also evident from the fact that a larger time window is needed to detect insiders accurately. The use of only the first-order statistics [7] is insufficient to capture the dynamics of behavioral changes accurately. As in previous experiments, here again the performances of deep autoencoder and random forest have been found to be comparable. The results in Table V also prove that both the cost-sensitive data-adjustment and the time-series classification of user activities are essential for accurate identification of insider threats.

So far, we have presented the results of insider threat detection from the ground truth knowledge of predefined threat scenarios. However, it is also possible to come across situations in which no threat scenarios are present in the data. In our next experiment, we evaluate the performance of the proposed algorithm on the CMU Insider Threat Data with the assumption that the scenario-specific ground truth knowledge of insider threat data is not available to us. The time-series

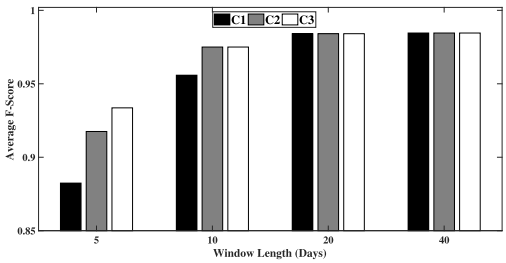


Fig. 12. Average f-score for the different window lengths and imbalance ratios when no threat scenarios are considered.

feature set is thus formed from the complete set of single-day features specified in Table II. To make this feature set scenario-independent, each feature vector in this feature set is labeled as malicious if it corresponds to any one of the threat scenarios S_1–S_3 (refer to Section I), else it is labeled as nonmalicious. In Fig. 12, we report the average f-score for the imbalance ratios: C_5–C_7, and window lengths: 5, 10, 20, and 40 days. As expected, it can be seen from the results that the maximum value of f-score (i.e., 0.98) is achieved if the window length is set to 20 days and the imbalance ratio is set to C_1. On increasing the length of the window beyond 20 days, no noticeable improvement in the value of f-score is observed.

IV. CONCLUSION AND FUTURE WORK

We have proposed an accurate scenario-based insider threat detection approach from human behavioral activities. Existing algorithms are mostly unsupervised and aim to find anomalies in human behavior. The few supervised approaches developed till date do not take into account the temporal changes in user activities. This paper is an improvement over the state-of-the-art algorithms. It constructs a time-series feature vector from the statistics of single-day user activities within a time window, employs random undersampling of the nonmalicious class samples to make the feature set balanced, uses the ground truth knowledge to label each vector in the balanced feature set as malicious or nonmalicious, and finally trains a classifier using the reduced feature set. The algorithm proposed in this paper can also be potentially deployed in other related domains, such as intrusion detection, even if no threat scenarios are defined in the data set.

Experimental evaluation of our algorithm in Section III shows that it has the capability to identify malicious insiders with significantly high precision, recall, and f-score if a suitable classifier is used. In the future, it can be studied if increasing the number of hidden layers of the deep autoencoder neural network can further improve the classification performance metrics. The effectiveness of our algorithm in handling challenging real-world data sets will also be studied in the future. We have started building our own data set from user-activity logs, and once we come up with an extensive data, we will evaluate the proposed algorithm on this data set and study the performance. Also, further improvements to the algorithm will be proposed, if needed.

ACKNOWLEDGMENT

The authors would like to thank the Director of IIT (BHU) Varanasi, Varanasi, India, for helping us with the facilities needed to complete the final part of the project.

REFERENCES

[1] I. X.-F. R. Team. (2016). *2016 Cyber Security Intelligence Index*. [Online]. Available: http://www-03.ibm.com/security/data-breach/cyber-security-index.html

[2] A. P. Moore *et al.*, "A preliminary model of insider theft of intellectual property," Softw. Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. , 2011.

[3] N. Liang and D. Biros, "Validating common characteristics of malicious insiders: Proof of concept study," in *Proc. 49th Hawaii Int. Conf. Syst. Sci.*, Jan. 2016, pp. 3716–3726.

[4] A. Memory, H. G. Goldberg, and T. E. Senator, "Context-aware insider threat detection," in *Proc. Workshop Activity Context Syst. Archit.*, 2013, pp. 44–47.

[5] W. T. Young, H. G. Goldberg, A. Memory, J. F. Sartain, and T. E. Senator, "Use of domain knowledge to detect insider threats in computer activities," in *Proc. Secur. Privacy Workshops*, May 2013, pp. 60–67.

[6] T. E. Senator *et al.*, "Detecting insider threats in a real corporate database of computer usage activity," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 1393–1401.

[7] G. Gavai, K. Sricharan, D. Gunning, R. Rolleston, J. Hanley, and M. Singhal, "Detecting insider threat from enterprise social and online activity data," in *Proc. 7th ACM CCS Int. Workshop Manage. Insider Secur. Threats*, 2015, pp. 13–20.

[8] O. Brdiczka *et al.*, "Proactive insider threat detection through graph learning and psychological context," in *Proc. Symp. Secur. Privacy Workshops*, May 2012, pp. 142–149.

[9] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Caught in the act of an insider attack: Detection and assessment of insider threat," in *Proc. IEEE Int. Symp. Technol. Homeland Secur.*, Apr. 2015, pp. 1–6.

[10] J. R. C. Nurse *et al.*, "Understanding insider threat: A framework for characterising attacks," in *Proc. Secur. Privacy Workshops*, May 2014, pp. 214–228.

[11] M. A. Maloof and G. D. Stephens, "ELICIT: A system for detecting insiders who violate need-to-know," in *Proc. Int. Workshop Recent Adv. Intrusion Detect.* Berlin, Germany: Springer, 2007, pp. 146–166.

[12] K. Bhavsar and B. H. Trivedi, "An insider cyber threat prediction mechanism based on behavioral analysis," in *Proc. Int. Conf. ICT Sustain. Develop.*, 2016, pp. 345–353.

[13] C. P. Pfleeger, "Reflections on the insider threat," in *Insider Attack and Cyber Security*. Boston, MA, USA: Springer, 2008, pp. 5–16.

[14] G. B. Magklaras and S. M. Furnell, "Insider threat prediction tool: Evaluating the probability of IT misuse," *Comput. Secur.*, vol. 21, no. 1, pp. 62–73, 2001.

[15] A. Azaria, A. Richardson, S. Kraus, and V. S. Subrahmanian, "Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data," *IEEE Trans. Comput. Social Syst.*, vol. 1, no. 2, pp. 135–155, Jun. 2014.

[16] W. T. Young, A. Memory, H. G. Goldberg, and T. E. Senator, "Detecting unknown insider threat scenarios," in *Proc. Secur. Privacy Workshops*, May 2014, pp. 277–288.

[17] M. Kandias, A. Mylonas, N. Virvilis, M. Theoharidou, and D. Gritzalis, "An insider threat prediction model," in *Proc. Int. Conf. Trust, Privacy Secur. Digit. Bus.* Berlin, Germany: Springer, 2010, pp. 26–37.

[18] H. Eldardiry, E. Bart, J. Liu, J. Hanley, B. Price, and O. Brdiczka, "Multi-domain information fusion for insider threat detection," in *Proc. Secur. Privacy Workshops*, May 2013, pp. 45–51.

[19] A. Coden *et al.*, "Uncovering insider threats from the digital footprints of individuals," *IBM J. Res. Develop.*, vol. 60, no. 4, pp. 8:1–8:11, Jul./Aug. 2016.

[20] C. Wan, L. Wang, and K. M. Ting, "Introducing cost-sensitive neural networks," in *Proc. 2nd Int. Conf. Inf., Commun. Signal Process.*, 1999, pp. 445–449.

[21] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63–77, Jan. 2006.

[22] N. Chawla, "Data mining for imbalanced datasets: An overview," in *Data Mining and Knowledge Discovery Handbook*. New York, NY, USA: Springer, 2005, pp. 853–867.

[23] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *Proc. Int. Conf. Artif. Intell., Special Track Inductive Learn.*, 2000, pp. 111–117.

[24] Z. Qin, C. Zhang, T. Wang, and S. Zhang, "Cost sensitive classification in data mining," in *Proc. 6th Int. Conf. Adv. Data Mining Appl.* Berlin, Germany: Springer-Verlag, 2010, pp. 1–11.

[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.

[26] S. Hu, Y. Liang, L. Ma, and Y. He, "MSMOTE: Improving classification performance when training data is imbalanced," in *Proc. 2nd Int. Workshop Comput. Sci. Eng.*, Oct. 2009, pp. 13–17.

[27] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proc. 7th Eur. Conf. Princ. Pract. Knowl. Discovery Databases*, 2003, pp. 107–119.

[28] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.

[29] R. Batuwita and V. Palade, "FSVM-CIL: Fuzzy support vector machines for class imbalance learning," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 3, pp. 558–571, Jun. 2010.

[30] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009.

[31] M. Kukar and I. Kononenko, "Cost-sensitive learning with neural networks," in *Proc. 13th Eur. Conf. Artif. Intell.* Hoboken, NJ, USA: Wiley, 1998, pp. 445–449.

[32] P. D. Turney, "Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm," *J. Artif. Intell. Res.*, vol. 2, pp. 369–409, Mar. 1995.

[33] P. Domingos, "MetaCost: A general method for making classifiers cost-sensitive," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 1999, pp. 155–164.

[34] C. Drummond and R. Holte, "C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling," in *Proc. Int. Conf. Machine Learn., Workshop Learn. Imbalanced Data Sets (II)*, 2003, pp. 1–8.

[35] J. Stefanowski and S. Wilk, "Selective pre-processing of imbalanced data for improving classification performance," in *Proc. Int. Conf. Data Warehousing Knowl. Discovery*. Berlin, Germany: Springer, 2008, pp. 283–292.

[36] T. Dietterich, "Ensemble methods in machine learning," in *Proc. 1st Int. Workshop Multiple Classifier Syst.* Berlin, Germany: Springer, 2000, pp. 1–15.

[37] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann, "Review of classifier combination methods," in *Machine Learning in Document Analysis and Recognition*. Berlin, Germany: Springer, 2008, pp. 361–386.

[38] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Ann. Statist.*, vol. 26, no. 5, pp. 1651–1686, 1998.

[39] X.-S. Hu and R.-J. Zhang, "Clustering-based subset ensemble learning method for imbalanced data," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Jul. 2013, pp. 35–39.

[40] S. Lee, D. Kwon, and S. Lee, "Minimum distance queries for time series data," *J. Syst. Softw.*, vol. 69, nos. 1–2, pp. 105–113, 2004.

[41] T. W. Liao, "Clustering of time series data—A survey," *Pattern Recognit.*, vol. 38, no. 11, pp. 1857–1874, Nov. 2005.

[42] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern Recognit.*, vol. 44, no. 9, pp. 2231–2240, Sep. 2011.

[43] T. M. Rath and R. Manmatha, "Lower-bounding of dynamic time warping distances for multivariate time series," Multi-Media Indexing Retr. Group, Center Intell. Inf. Retr., Univ. Massachusetts, Amherst, Amherst, MA, USA, Tech. Rep. MM 40, 2003.

[44] J. Mei, M. Liu, Y.-F. Wang, and H. Gao, "Learning a Mahalanobis distance-based dynamic time warping measure for multivariate time series classification," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1363–1374, Jun. 2016.

[45] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 289–297.

[46] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining Knowl. Discovery*, vol. 28, no. 4, pp. 851–881, 2014.

[47] L. Ye and E. Keogh, "Time series shapelets: A novel technique that allows accurate, interpretable and fast classification," *Data Mining Knowl. Discovery*, vol. 22, nos. 1–2, pp. 149–182, 2011.

[48] X. Ji, J. Bailey, and G. Dong, "Mining minimal distinguishing subsequence patterns with gap constraints," in *Proc. 5th IEEE Int. Conf. Data Mining*, Nov. 2005, pp. 259–286.

[49] C. Jentsch and S. S. Rao, "A test for second order stationarity of a multivariate time series," *J. Econometrics*, vol. 185, no. 1, pp. 124–161, Mar. 2015.
[50] X. Zhu, W. Luan, and Y.-S. Zhu, "Frequency domain method for analysis of macroeconomic time series," in *Proc. Int. Conf. Multimedia Technol.*, Oct. 2010, pp. 1–4.
[51] K. K. Teo, L. Wang, and Z. Lin, "Wavelet packet multi-layer perceptron for chaotic time series prediction: Effects of weight initialization," in *Proc. Int. Conf. Comput. Sci.* Berlin, Germany: Springer, 2001, pp. 310–317.
[52] L. Wang, K. K. Teo, and Z. Lin, "Predicting time series with wavelet packet neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2001, pp. 1593–1597.
[53] P. Esling and C. Agon, "Time-series data mining," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 12:1–12:34, 2012.
[54] D. Chakrabarti and C. Faloutsos, "F4: Large-scale automated forecasting using fractals," in *Proc. 11th Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, 2002, pp. 2–9.
[55] W. Lutzenberger, H. Preissl, and F. Pulvermüller, "Fractal dimension of electroencephalographic time series and underlying brain processes," *Biol. Cybern.*, vol. 73, no. 5, pp. 339–350, 1997.
[56] L. Wang and X. Fu, *Data Mining With Computational Intelligence*. Berlin, Germany: Springer-Verlag, 2006.
[57] X. Wang, A. Wirth, and L. Wang, "Structure-based statistical features and multivariate time series clustering," in *Proc. 7th IEEE Int. Conf. Data Mining*, Oct. 2007, pp. 351–360.
[58] C. Chatfield, *The Analysis of Time Series: An Introduction*, 6th ed. Boca Raton, FL, USA: CRC Press, 2016.
[59] Y. Matsuo and M. Ishizuka, "Keyword extraction from a single document using word co-occurrence statistical information," *Int. J. Artif. Intell. Tools*, vol. 13, no. 1, pp. 157–169, 2004.
[60] Y.-H. Chen, E. J.-L. Lu, and M. F. Tsai, "Finding keywords in blogs: Efficient keyword extraction in blog mining via user behaviors," *Expert Syst. Appl.*, vol. 41, no. 2, pp. 663–670, Feb. 2014.
[61] P. Willett, "The porter stemming algorithm: Then and now," *Program*, vol. 40, no. 3, pp. 219–223, 2006.
[62] H. Preißl, W. Lutzenberger, F. Pulvermüller, and N. Birbaumer, "Fractal dimensions of short EEG time series in humans," *Neurosci. Lett.*, vol. 225, no. 2, pp. 77–80, Apr. 1997.
[63] A. Fernández, S. García, M. J. del Jesus, and F. Herrera, "A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets," *Fuzzy Sets Syst.*, vol. 159, no. 18, pp. 2378–2398, 2008.
[64] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[65] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997, ch. 4, pp. 81–126.
[66] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Feb. 2010.
[67] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

Pratik Chattopadhyay received the B.Tech. degree in computer science and engineering from the Institute of Engineering and Management, Kolkata, India, in 2009, the M.E. degree in computer science and engineering from Jadavpur University, Kolkata, in 2011, and the Ph.D. degree from IIT Kharagpur, Kharagpur, India, in 2015.

He was a Post-Doctoral Researcher with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He is currently an Assistant Professor with the Department of Computer Science and Engineering, IIT (BHU) Varanasi, Varanasi, India. His current research interests include machine learning, image processing, and data mining.

Lipo Wang received the bachelor's degree from the National University of Defense Technology, Changsha, China, and the Ph.D. degree from Louisiana State University, Baton Rouge, LA, USA.

He has authored or co-authored over 300 papers, of which more than 100 are in journals. He holds a U.S. patent in neural networks and a Chinese patent in very large scale integration. He has co-authored two monographs and (co-)edited 15 books. His current research interests include intelligent techniques with applications to communications, image/video processing, biomedical engineering, and data mining.

Dr. Wang was the President of the Asia–Pacific Neural Network Assembly (APNNA). He was an AdCom Member of the IEEE Computational Intelligence Society from 2010 to 2015 and served as the CIS Vice President for Technical Activities. He was a member of the Board of Governors of the International Neural Network Society from 2011 to 2016 and an AdCom Member of the IEEE Biometrics Council. He received the APNNA Excellent Service Award. He served as the Chair of the Emergent Technologies Technical Committee. He was the Founding Chair of both the EMBS Singapore Chapter and the CIS Singapore Chapter. He served as the Chair of the Education Committee, IEEE Engineering in Medicine and Biology Society (EMBS). He serves/served as the chair/committee member of over 200 international conferences. He was/will be a keynote/panel speaker for 35 international conferences. He is/was an associate editor/editorial board member of 30 international journals, including four IEEE TRANSACTIONS, and a guest editor of 10 journal special issues.

Yap-Peng Tan (SM'97) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1993, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA, in 1995 and 1997, respectively, all in electrical engineering.

From 1997 to 1999, he was with Intel Corporation, Chandler, AZ, USA, and Sharp Laboratories of America, Camas, WA, USA. In 1999, he joined Nanyang Technological University, Singapore, where he is currently a Professor and an Associate Chair (Academic) of the School of Electrical and Electronic Engineering. His current research interests include image and video processing, content-based multimedia analysis, computer vision, pattern recognition, machine learning, and data analytics.

Dr. Tan served a member of the Multimedia Systems and Applications Technical Committee and Visual Signal Processing and Communications Technical Committee of the IEEE Circuits and Systems Society, and a member of the Image, Video and Multidimensional Signal Processing Technical Committee of the IEEE Signal Processing Society. He served as a voting member of the IEEE International Conference on Multimedia & Expo Steering Committee from 2011 to 2012, the Chairman of the IEEE Signal Processing Singapore Chapter from 2009 to 2010, the Chair of the Visual Signal Processing and Communications Technical Committee of the IEEE Circuits and Systems Society from 2012 to 2014, the Chair of the Nominations and Elections Subcommittee of the Multimedia Signal Processing Technical Committee of the IEEE Signal Processing Society from 2012 to 2013, and the Chair of the Membership & Election Subcommittee of the Multimedia Systems and Applications Technical Committee of the IEEE Circuits and Systems Society from 2013 to 2017. He was the Finance Chair of ICIP 2004, the General Co-Chair of ICME 2010, the Technical Program Co-Chair of ICME 2015, and the General Co-Chair of the 2015 IEEE International Conference on Visual Communications and Image Processing 2015. He is the Technical Program Co-Chair of ICME 2018 and the 2019 IEEE International Conference on Image Processing 2019, the Chair of the ICME Steering Committee from 2018 to 2019. He has also served as an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE SIGNAL PROCESSING LETTERS, and the IEEE ACCESS, an Editorial Board Member of the *EURASIP Journal on Advances in Signal Processing*, and *EURASIP Journal on Image and Video Processing*, a Guest Editor for special issues of several journals including the IEEE TRANSACTIONS ON MULTIMEDIA.