This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

Analyzing Data Granularity Levels for Insider Threat Detection using Machine Learning

Duc C. Le, *Student Member, IEEE,* Nur Zincir-Heywood, *Member, IEEE,* and Malcolm I. Heywood, *Senior Member, IEEE*

***Abstract*—Malicious insider attacks represent one of the most damaging threats to networked systems of companies and government agencies. There is a unique set of challenges that come with insider threat detection in terms of hugely unbalanced data, limited ground truth, as well as behaviour drifts and shifts. This work proposes and evaluates a machine learning based system for user-centered insider threat detection. Using machine learning, analysis of data is performed on multiple levels of granularity under realistic conditions for identifying not only malicious behaviours, but also malicious insiders. Detailed analysis of popular insider threat scenarios with different performance measures are presented to facilitate the realistic estimation of system performance. Evaluation results show that the machine learning based detection system can learn from limited ground truth and detect new malicious insiders in unseen data with a high accuracy. Specifically, up to 85% of malicious insiders are detected at only 0.78% false positive rate. The system is also able to quickly detect the malicious behaviours, as low as 14 minutes after the first malicious action. Comprehensive result reporting allows the system to provide valuable insights to analysts in investigating insider threat cases.**

***Index Terms*—Insider threat, machine learning, data granularity.**

I. Introduction

INSIDER threats are one of the most dangerous and prevalent security threats that various institutions, companies and government agencies are facing, where the malicious acts are performed by authorized personnel inside the organization. Due to the fact that insiders are authorized to access an organization's networked systems and knowledgeable about its structure and security procedures, an insider threat is one of the most costly types of attacks and hardest to detect. Cybersecurity reports show that 53% of organizations and 42% of U.S. federal agencies are suffering from insider threats every year [1], [2]. In a recent survey, insider threat attacks are shown to account for 25% of all attacks to organizations and their frequency is on the rise [3]. Some insider threat related incidents reported, to date, include data compromised, such as customer records, trade secrets or intellectual property, theft of personally identifiable information, and IT system sabotage [3], [4]. In short, an integral part to any organizations are networked systems, where data, such as intellectual property and sensitive data of employees and customers, is stored, shared, and processed. As such, the insider threat poses a serious cybersecurity challenge, which needs to be addressed with high priority to ensure the continued security of such systems and therefore an organization's functionality.

The authors are with Faculty of Computer Science, Dalhousie University, Canada (e-mail: lcd@dal.ca, zincir@cs.dal.ca, mheywood@cs.dal.ca).

The insider threat is defined in a recent technical report by the CERT Insider Threat Center as threats that are carried out by malicious or unintentional insiders, whose authorized access to the organization's network, system, and data is exploited to negatively affect the confidentiality, integrity, availability, or physical well-being of the organization's information, information systems, or workforce [4]. Malicious activities related to an insider threat can be carried out both intentionally by malicious insiders, such as information system sabotage, intellectual property theft and disclosure of classified information, as well as by an unintentional insider, such as user negligence in using authorized resources. Distinct from traditional intrusion detection tasks, many challenges of insider threat detection come from the fact that the insider is authorized to access the organization's computer systems and has familiarity with the organization's security layers. Furthermore, in most organizations, the activities of insiders with malicious intent occur infrequently. Thus, data available to describe the activity is usually rare and not well documented [5]. Finally, challenges of insider threat detection may arise from the need to process and investigate a wide range of data types in organizational environments, from network traffic, web and file access logs, to email history, or employee information. The available data also differs significantly by organization. Hence only a small fraction of organizations have tools and (the human) resources to interpret user's behavior and intent from the monitoring data collected [3].

This research evaluates the application of machine learning (ML) techniques in detecting malicious insiders in networked systems of corporations and organizations. We propose and evaluate a workflow for user-centered insider threat detection, from data collection and pre-processing, to data analysis using ML models, and alert reporting and analysis. The proposed system aims to *assist* cybersecurity analysts in learning from only a small number of normal user behaviors and detected malicious insider actions to identify threats in unknown data, and provide useful insights.

To this end, the research aims to make the following contributions: (i) We assume a realistic condition for training ML models in order to obtain results reflecting real-world environments, and highlight the differences compared to training under ideal (traditional ML) conditions; (ii) We explore data processing techniques to enable the extraction of multiple levels of data granularity with rich details for data analytics; (iii) We aim to address one of the major

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

concerns of security analysts via the introduction of a detection delay metric. This reflects the elapsed time between malicious behaviour and detection. Currently, this might be as much as 108 days [3]; (iv) We present a comprehensive result reporting process, where instance and user based results are reported, and malicious cases are investigated, for a better view of system performance.

The remainder of the paper is organized as follows. Section II summarizes previous research in applications of ML for insider threat detection. Section III presents the proposed system as well as data processing steps and the ML algorithms employed. Section IV details the employed dataset, experimental settings, and evaluation results, including the new metrics introduced for measuring the delay in detection and the detection rate per detected malicious insider. Finally, conclusions are drawn and the future work is discussed in Section VI.

II. Related Work

Insider threat detection is a challenging research problem, not only to the research community but also government agencies and cybersecurity firms. CERT Insider Threat Center and the U.S. National Insider Threat Task Force have issued common guidelines to help prevent and mitigate insider threats in organizational environments [4], [6]. The guidelines in [4] describes 20 practices that organizations should implement across the enterprise to prevent and detect insider threats, as well as case studies of organizations that failed to do so. Recent surveys, [7], [8], provide an overview of the insider threat research literature. In [7], Liu et al. reviewed insider threats and related cybersecurity problems, such as malware and advanced persistent threats, while in [8], Homoliak et al. proposed a structural taxonomy and novel categorization of insider threat related research.

As the insider threat problem can be related to human factors, many researchers approach the problem using psychological models and decision-making theories [9], [10], [11], [12]. In [9], Padayachee applied opportunity theories from criminology for conceptualizing insider threats, as well as opportunity-reducing measures for assisting insider threat mitigation. In [10], Greitzer et al. proposed a predictive modeling framework that integrates multiple data sources and psychological/motivational factors for assisting the data analyst in detecting high-risk behaviors. Legg et al. in [11] proposed a framework for modeling the insider threats based on behavioral and psychological observations. The framework allows an analyst to reason and describe potential insider threats from different domains, such as human behaviors and organizational policies.

Considering the huge amount of data acquired daily in any organization, ML based solutions are one of the most promising approaches for solving cybersecurity challenges in the current era [13], [14]. The advantage of ML is the ability to automatically learn from a large amount of data and identify patterns potentially characterizing malicious activities or anomalous behaviors. In [15], Eberle et al. employed a graph based anomaly detection algorithm for insider threat detection. Caputo et al. introduced Elicit, a system for monitoring user activities and indicating malicious activities based on Bayesian Networks [16]. They also identified some malicious user indicators via an empirical study that may benefit insider threat detection system design.

Anomaly detection is a popular approach employing ML in insider threat detection, where normal user behavior models are built and anomalies are identified as deviations from the normal behavior. An anomalous alert in this case may indicate changes in employee behaviors as a potential early indicator of insider threats. Based on graph models, Parveen et al. assumed an incremental learning approach to insider threat detection under streaming data [17]. To do so, quantized dictionaries of patterns are created for each data chunk and test data is considered an anomaly if it has a large edit distance from all patterns in the dictionary. Another approach to anomaly detection is to model the sequences of user actions and employ them to detect unusual sequences [18], [19]. A Hidden Markov Model was applied by Rashid et al. in [19] to capture each user's normal weekly activity sequences of common activities. Anomalous action sequences detected by the model may potentially indicate insider threats. Many insider threat detection systems, [18], [20], [21], [22], were supported by DARPA's project ADAMS [23]. The project aimed to "identify patterns and anomalies in very large datasets" in order to detect and prevent insider threats. In [18], [22], various anomaly detection algorithms, including Hidden Markov Models and Gaussian Mixture Models, were employed in an ensemble on user activity log data for identifying insider threat indicators. A visual language for describing anomalies was proposed in the work of [18]. In [20], Eldardiry et al. employed a hybrid combination of anomaly detectors on several different types of user activity logs to detect two classes of insiders – blend-in malicious insiders, and insiders with an unusual change in behaviors. Salem et al. [24] and Toffalini et al. [25] proposed masquerader detection approaches based on anomaly detection in user search and file access behaviors.

Other recent ML based approaches to insider threat detection include supervised learning [21], [26], [27], and stream online learning [28], [29], [30]. Gavai et al. applied different ML methods on organizational data to detect not only anomalies but also early "quitter" indicators, where both may possibly suggest insider threats [21]. Capabilities of different ML techniques, such as decision tree [27], Bayesian-based approaches [31], [32], and self-organizing map [26] were also examined for insider threat detection. Stream online learning approaches were employed in an attempt to characterize conditions of non-stationary user behaviors. With this in mind, the moving weighted average is a common technique to support anomaly detection under streaming conditions [17], [28]. In [28], Tuor et al. proposed an anomaly detection approach using a deep neural network (one model for the organization), or recurrent neural networks (one model per user), to produce anomaly scores. Bose et al. proposed a system employing scalable supervised and unsupervised learning algorithms on a fusion of heterogeneous data streams to detect anomalies and insider threats [30]. On the other hand, Le et al. benchmarked genetic programming algorithms under

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

two behavioral assumptions: stationary and non-stationary, to examine the prospect of evolutionary computation in insider threat detection [29].

In this paper, as distinct from previous work in the literature, we propose a user-centered insider threat detection system, where data analytics is employed on multiple levels of data granularity under different training conditions. We evaluate the proposed system on publicly available CERT insider threat datasets [33]. To the best of our knowledge, this is the first work comprehensively assessing the effect of different data granularity levels and training conditions of ML in insider threat detection. Assuming the point of view of a security analyst in evaluating suspicious user activities, the system provides rich information to guide the analysis, where the results are reported by not only the correctly detected data instances, but also the correctly identified normal and malicious insiders, as well as insider threat scenarios. Well-known security metrics are adopted and new ones introduced to show practical performances of the system, while also presenting insider threat indicators helping to understand the user behaviors.

III. Methodology

The principal interest of this research is to assess the capability of ML techniques in detecting insider threats in a corporate and organizational network. To this end, in this section, the workflow for applying ML techniques is presented (Section III-A). The workflow is designed to be modular and easily expandable for a wide range of corporate environments, data acquisition conditions, as well as learning and analysis methods. Section III-B details the data collection and pre-processing steps, where features are constructed and different levels of data granularity are defined. Finally, Section III-C presents the ML algorithms used in this research for data analytics.

A. System Overview

The proposed approach of a system for malicious behavior and insider threat detection is illustrated in Fig. 1. The system process is as follows:

1) Data collection: Data from multiple sources are collected and stored in unified formats. The two main sources are:
   - User activities, e.g. network traffic, email, file logs.
   - Organization structure and user profile information.
2) Data pre-processing: aggregated data are processed to construct feature vectors representing user activities and profile information in different granularity levels.
3) ML algorithms are employed to data analytics based on the constructed feature vectors.
4) Results are presented in different formats, and detailed analysis is provided to the system analyst.

The system is designed to operate with the participation / supervision of security analysts in many steps, especially in initial detection, where the early signs of malicious behaviors and abnormal activities are investigated. Human analysts play an important role not only in the analysis of system warnings and alerts, but also in performing the necessary actions to return the system to 'normal' operation after an attack.

In the work reported here, we assume benchmarking datasets provided by CERT (Section IV-A) for the specific purpose of assessing the data analytics component of Fig. 1. Specifically, we are interested in evaluating the ML algorithms trained with limited amount of ground truth for detecting unknown malicious insiders. To this end, supervised learning algorithms are employed to learn from the obtained knowledge (ground truth) on malicious / normal user behaviors. Then, we explore how well the learned solution would be able to generalize for detecting unknown malicious insider cases. The benefit from using supervised learning is that we need not assume that data clusters are always synonymous with distinct behaviors. This potentially leads to higher precision than unsupervised learning / anomaly detection algorithms [13] (IV-C2).

In addition, our analysis will distinguish between malicious actions detected and malicious users detected, where the two are not necessarily the same. That is to say, the diversity in a user's role within an organization can impact on the number / types of actions performed, both normal and malicious. In many cases, user actions can vary over time and have multiple contexts that need to be taken into account in order to process an alert about a suspicious behavior [34]. Thus, high malicious instance detection rates in this case may not necessarily translate to all malicious insiders being detected. Furthermore, a seemingly small false positive rate may still require a lot of attention from the security analyst if it flags many distinct normal users as anomalous. In short, we believe that results that highlight malicious users rather than events represent a more important measure of system performance.

Finally, several measures are presented, such as detection delay per malicious insider, or the support for each malicious insider alert. By providing these measures, we aim to provide a better support to security analysts and positively contribute to a successful application of the proposed system in real-world scenarios.

B. Data Collection and Pre-processing

Data collection and pre-processing is crucial for insider threat detection in particular but also for cybersecurity tasks in general. A good monitoring procedure in combination with adequate data collection enables a successful application of ML techniques and support security analysts in making correct decisions.

Data collected from organizational environments may come from a wide variety of sources and have many different forms [4], [35]. This research assumes that organizational data are collected in two main categories: (i) activity log data, and (ii) organization's structure and users' information. Data from the first category comes from different logging systems such as network traffic capture, firewall logs, email, web, and file access. These represent real-time sources of data that often need to be collected and processed in a timely manner in order to quickly detect and respond to malicious and/or anomalous behaviors. The second category of data represents background

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

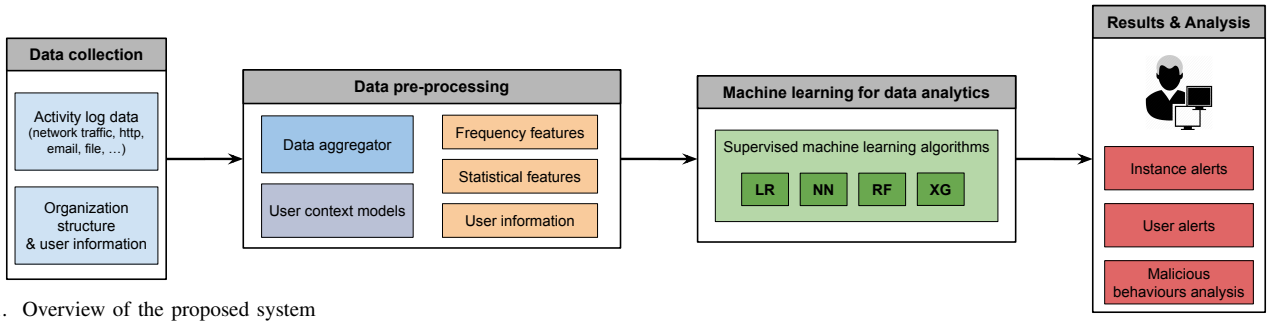


Fig. 1. Overview of the proposed system

or context data, which can be employee information, role in the organization, relationships to other users. In many cases, this category also consists of more complex data, such as psychometric and behavioral models of users.

To assist data processing and feature contruction, user context models are created for each user in the organization. The models consist of auxiliary information related to each user, such as assigned machines, relationships with other users, roles, work hours, permitted access and so on. Based on the user context models, feature vectors summarizing user's actions can be quickly and orderly created from incoming data.

1) Feature Extraction: From the collected data and user context models, feature extraction can be performed to create data vectors that are suitable for training ML algorithms. Firstly, data from different sources are aggregated based on user id given an aggregation condition c, such as time duration or number of actions performed. Subsequently, feature extraction is performed on the aggregated data to generate numerical vectors x_c, which are also called data instances, of fixed length N summarizing user actions. Each vector consists of user information – mostly categorical data encoded in numeric format for providing context for ML algorithms – and two types of features:

- Frequency features, which are the count of different types of actions the user performed in the aggregation period, e.g. *number of emails sent, number of file accesses after work hour*, or *number of websites visited on a shared PC*.
- Statistical features, which are descriptive statistics, such as mean, median, standard deviation, of data. Examples of data that are summarized in statistical features are email attachment sizes, file sizes, and the number of words in websites visited.

Fig. 2 demonstrates the feature creation process in the case of CERT dataset employed in this work. The process allows creating information-rich features consisting of many details, such as PC, time, and action specific characteristics. Up to three connected pieces of information presented in Fig. 2 are combined to generate a feature, such as the *number of actions on a shared PC, number of HTTP downloads after workhour, mean attachment size of sent emails.* Thus, the set of features constructed is essentially an enumeration over the pieces of information[1].

HTTP and file features require that we define sets of website and file categories in corporate environments that are

[1]The extracted features and data are available upon request.

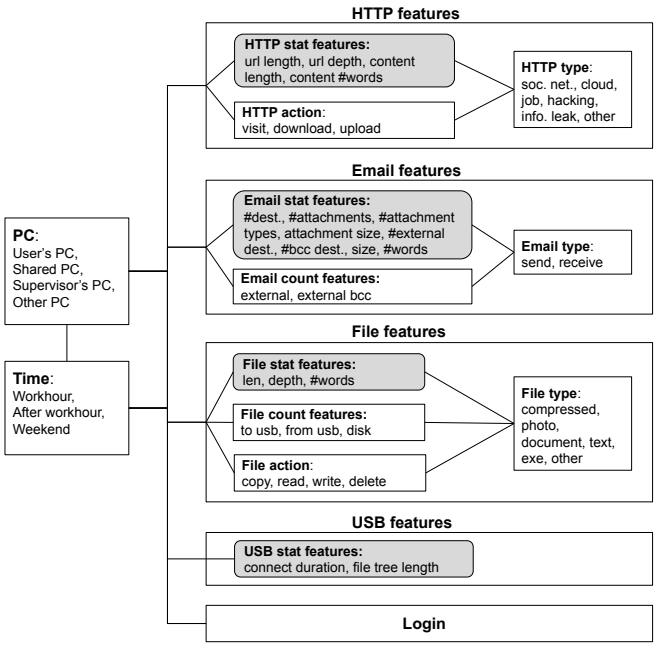


Fig. 2. Illustration of the feature extraction process

TABLE I
DATA GRANULARITY LEVELS

Data type	Notation	Aggregation criterion c
User-Week	$\mathbf{x}_w$	Week of user actions on all PCs
User-Day	$\mathbf{x}_d$	Day of user actions on all PCs
User-Session	$\mathbf{x}_s$	Session of user actions, from login to logoff on a PC
User-Subsession Ti	$\mathbf{x}_{t=i}$	i hours of user actions in each session
User-Subsession Nj	$\mathbf{x}_{n=j}$	j user actions in each session

potentially helpful to insider threat detection. Furthermore, a well designed categorizing scheme for collecting activity information directly facilitates privacy preserving user monitoring, as specific websites and files that users visited, as well as their contents are not inspected in data pre-processing [36].

2) Data Granularity: Based on the aforementioned data aggregation condition c, extracted features could have different levels of granularity. We explore two main criteria for c: time duration and the number of actions performed. Table I summaries the data types extracted in this research based on different granularity levels.

In the case of time duration, three data aggregations of user activities are assumed: week, day, or session [27], [29]. User-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

week and user-day data instances summarize users' activities over the corresponding time period. These coarse-grained types of data provide a high-level overview of behaviors in a day or a week with a higher feature count than session and sub-session data. As such, they can potentially accelerate learning process through lowering the amount of extracted data instances. On the other hand, user-session data points provides higher data fidelity by capturing user actions on a PC, from Login to corresponding Logoff; or from one Login to next Login. Session-based data has utility for isolating malicious actions, since malicious users tend to perform malicious actions in particular sessions whereas other sessions in the same day or week may still be normal [8]. Furthermore, as the duration of a session is typically much shorter than a day, this data type may also allow quicker system responses when a malicious instance is detected.

As a session may last many hours and comprises of hundreds of actions, we explore further the balance between the amount of data summarized in each data instance and potential system response time to malicious behaviors. This is done by using time duration and number of actions performed as the criteria for separating a user-session data instance into sub-session data instances. This way, we control the amount of information embedded into each data instance. Thus, if the ML based system could successfully learn from the short-lived sub-session data to detect malicious behaviors, response time of the system might be improved. As presented in Table I, based on duration, a user-subsession Ti data instance is created every i hours from the start time of a user's session on a PC. Similarly, a user-subsession Nj data instance is created after every j actions by a user on a PC, starting from the Login action. The smaller i and j, the higher fidelity the data, but also the smaller amount of user activity information that is summarized in an instance. Empirical analysis is performed in Section IV-A to identify the most suitable values for i and j for the CERT dataset.

C. ML for Data Analytics

In this research, the following four well-known and widely-used ML algorithms are employed: Logistic Regression, Random Forest, Neural Network, and XGBoost [7], [13], [37], [38]. Brief descriptions of the algorithms are presented below, while more detailed descriptions can be found in [39].

1) Logistic Regression (LR): LR is a linear statistical model that uses a logistic function to model a binary dependent variable based on independent variables. Specifically, in this work, the logistic function, σ, is used to model the probability of normal or malicious insider behavior for each input x.

$$\sigma(w^T x) = (1 + e^{-w^T x})^{-1} \tag{1}$$

Logistic regression training (with $l2$-regularization) results in the identification of a weight vector, w, that minimizes the sum of squared errors between logistic function $\sigma(w^T x)$ and target labels. Logistic regression has the advantage of being highly interpretable as a linear model. Furthermore, it returns a probability of an input vector belonging to a class, thus facilitating the prioritization of the most suspicious actions for investigation. In this work, logistic regression is included as a baseline model [40].

2) Neural Network (NN): The NN assumed in this work takes the form of a multi-layer perceptron with up to three hidden layers.[2] Neural networks with at last a hidden layer provide the ability to model a wide range of non-linear properties [41]. Reliably training such architectures has recently been made possible through developments in approaches to credit assignment and representation. Specifically, back-propagation with the Adam formulation of stochastic gradient decent is assumed in this work [42], where this implies that each weight in the network has its own (adaptable) learning rate. Thus, given a mini-batch sample of training exemplars, statistics are collected to enable second order information to be collected, i.e. the second moments of the gradients are inferred. The resulting combination of per weight learning rate adaption and (stochastic) back-propagation of the error represents a more robust scheme for weight updating than many previous more computationally expensive methods [42]. The use of rectified linear activation functions (as part of the representation) in the hidden layers also accelerates learning across multi-layer architectures [43].

3) Random Forest (RF): RF represents a process for building an ensemble of decision tree classifiers that collectively 'vote' to provide a single class label for each exemplar [44]. Given p training exemplars, each described in terms of d attributes, each decision tree is define by: (1) selecting a random *subset* of the p training exemplars; (2) identify a random subset, D, of the d attributes ($D \ll d$); (3) each of the D selected attributes are used to parameterize a new decision node in the decision tree using the the Gini index [39].

The entire process is repeated to create a user specified number of decision trees. The selection of attributes (step 2) and design of decision tree nodes (step 3) is contextual on the subset of exemplars to build each decision tree, and subset of attributes used to build each decision node. These properties have been formally shown to preclude over learning, making the predictions of the RF robust [44].

4) XGBoost (XG): Similar to RF, XG also assumes decision trees as the classifiers to compose an ensemble [45]. However, XG assumes a gradient boosting method, where the combination of simple decision trees into a strong ensemble is guided by the optimization of a differentiable loss function. Moreover, in boosting, the classifier output predicts class labels through probabilities obtained using the logistic transformation of a linear combination of each decision tree output. XGBoost provides improvements over traditional gradient boosting methods to allow a highly scalable tree boosting system. Examples include regularization, a mechanism for operating under sparse and weighted data, and adopting a block structure for parallel learning. The algorithm has been successfully applied to applications in a wide range of data mining tasks [46], [47]. Due to its popularity, in this work, we include XG to test its capability in insider threat detection.

[2]Larger architectures could be trained, but at the expense of computational cost and interoperability, factors that are also potentially important in real-world applications [13].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

IV. Experimental Evaluations

In this section, we present the evaluation of the proposed system for insider threat detection using the CERT insider threat dataset. Specifically, Section IV-A details the dataset, as well as data pre-processing based on the identification of multiple levels of data granularity (Section III-B). In Section IV-B, the experiment settings and performance measures are introduced. Sections IV-C, IV-D, IV-E, and IV-F detail the experiment results.

A. Dataset

Obtaining data for evaluating insider threat detection systems faces additional challenges over typical difficulties observed in cybersecurity. This is due to concerns involving user privacy and intellectual properties of organizations, such as corporations and government agencies. In this work, we employ the CERT insider threat dataset, which is a publicly available dataset for research, development, and testing of insider threat mitigation approaches [33], [48]. The dataset consists of multiple releases, each of which characterizes an organization with 1000 to 4000 employees. Release 5.2 of the dataset (CERT r5.2) employed in this work simulates an organization with 2000 employees over the period of 18 months. CERT r5.2 consists of user activity logs, categorized as follows: login/logoff, email, web, file and thumb drive connect, as well as organizational structure and user information. Each malicious insider in CERT r5.2 belongs to one of four popular insider threat scenarios: data exfiltration (scenario 1), intellectual property theft (scenarios 2, 4) to IT sabotage (scenario 3). A detailed description of the insider threat scenarios can be found in Section IV-E and [33].

Following the data pre-processing steps described in Section III-B, feature construcction is performed to obtain data in multiple levels of granularity (Table I and Fig. 2). To build user context models for feature extraction, we manually examine the CERT data to approximately determine the missing information, such as user-user relationships, user-PC relationships, regular working hours, and website and file categories.

Fig. 3 shows the distribution of user-session data by the number of actions, the duration of each session, and the relationship between the two features. It is now apparent that the majority of user-session data has less than 300 actions, and more than a half of the sessions has less than 100 actions. We therefore conclude that $j = \{25, 50\}$ for extracting user-subsession Nj data. On the other hand, duration of a session is closer to a uniform distribution, with a large proportion lasting longer than 8 hours. Moreover, as shown in Fig. 3, many sessions with less than 50 actions may last longer than 10 hours. Hence, we explore the values of $i = \{2, 4\}$ for extracting subsession data by time.

Table II provides an overview of the data types and the number of normal and malicious users. It is apparent that the data distribution is extremely skewed, where malicious insider related data only accounts for 0.39%, 0.19%, and 0.18% of user-week, user-day, and user-session data, respectively. On sub-session data, the figure is even smaller, ranging from 0.09% to 0.15%. Moreover, on examining different insider

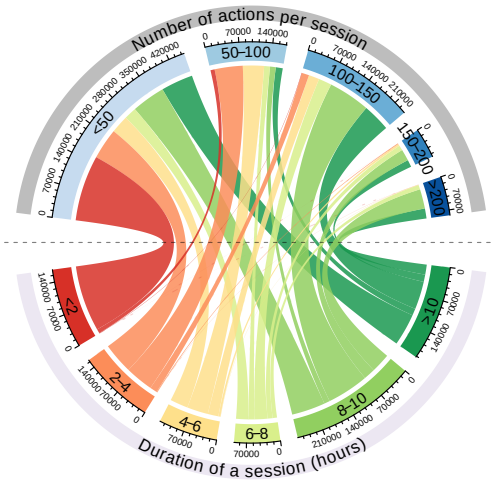


Fig. 3. Relationship between the duration and the number of actions in user-session data

TABLE II
Summary of the extracted data. (Sc: Insider threat scenario)

	Data	# features	Data distribution by class				
			Normal	Sc 1	Sc 2	Sc 3	Sc 4
higher granularity ↓	$\mathbf{x}_w$	1092	139,572	49	245	10	248
	$\mathbf{x}_d$	824	692,342	85	863	20	339
	$\mathbf{x}_s$	221	1,002,616	65	1,070	33	678
	$\mathbf{x}_{n=50}$	222	2,164,629	70	2,018	48	678
	$\mathbf{x}_{n=25}$	222	3,710,139	89	2,841	55	679
	$\mathbf{x}_{t=4}$	222	2,153,840	93	1,884	47	678
	$\mathbf{x}_{t=2}$	222	3,713,142	119	2,811	59	678
	# users by scenario:		**1901**	**29**	**30**	**10**	**30**

threat scenarios, different patterns appear to be present. Scenario 3 (IT sabotage related malicious actions) has the least amount of users and data instances. On the other hand, the malicious actions in scenarios 2 and 4 (actions for different types of intellectual property thefts) span long periods – 8 weeks (more than 240 malicious user-week data instances / 30 users). This may indicate that malicious insiders are attempting in avoid detection by performing malicious actions over a long period of time. However, different characteristics between scenario 2 and 4 appear when single malicious sessions are considered. While almost all individual malicious sessions of scenario 4 are short (less than 2 hour or 25 actions), a large percentage of scenario 2 sessions are more than 50 actions and span over 4 hours.

B. Experiment Settings

In this work, our aim is to obtain a realistic estimation of the proposed system's performance on real-world networked systems, based on scenarios characterized by limitations to the amount of ground truth data available for training the ML algorithms. Specifically, in real-world environments, labeled (ground truth) data for training detection systems is scarce. Thus, ground-truth is only obtainable from a limited set of verified users, while behaviors of others are generally unknown [14], [48].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

To emulate this condition, we assume a primary configuration – namely *realistic* condition thereafter – where ground-truth is obtained from only a restricted set of users over a given time period. Thus, the ground truth data for training the ML algorithms is limited to the data of 400 identified "normal" and "malicious" users (among 2000 users in the organization), based on the *first* 37 weeks – 50% of the time period that the dataset covers. By user count, this allows the ML algorithms to learn from data representing 18% of "normal" users and 34% of malicious insiders. It is noteworthy that from a detector's point of view, the "normal" users in the training data are only guaranteed to be benign in the first 37 weeks, while later in the testing weeks, they may or may not turn "malicious". Additionally, we further ensure experiments are realistic by presenting results obtained solely from unknown users, i.e users that have not performed any malicious actions in the first 37 weeks. By excluding known malicious users, i.e. users whose malicious actions are included in training data, from system performance measures, we believe the evaluations performed reflect the real life situations as well as cybersecurity analyst's interest [34].

The evaluation results are obtained from a series of experiments, where each setting – a ML algorithm on a data type – is randomly repeated 20 times. In the first experiment, to show the contrast between traditional ML applications and real-world cybersecurity situations, we compare the *realistic* setting above with an *idealistic* (traditional) setting, where a random 50% of data from the whole data set is used to train the ML algorithms. This is done at three levels of data granularity: user-week, user-day, and user-session. The second experiment evaluates the ML algorithms in *realistic* setting on all aforementioned data granularity levels to obtain detailed results, both instance-based and user-based. Detailed analysis is performed on the results for each insider threat scenario provided in the data set. Furthermore, models trained on CERT r5.2 are also used to test against other versions of CERT insider data for exploring the generalization of the trained models' performances under new / unseen environments (different version of the CERT data emulates different organizations).

1) ML Training Configuration and Parameterization: In this research, Python 3.7 is used for data pre-processing steps and Scikit-learn [49] and XGBoost [45] are used for implementing ML algorithms. The training data are normalized, per attribute, to zero mean and unit variance before being used to train the ML algorithms. The ML algorithms are trained in a binary setting, where the two classes are: malicious (*positive*) and normal (*negative*).

Logistic Regression assumes the *lbfgs* solver [50] and appeared to perform best under default parameters. In the case of the three remaining algorithms, we perform parameter search with cross-validation using hyperopt, which is a parameter tuning solution based on the tree-structured Parzen estimator [51]. Specifically, for RF, we tune the number of decision tree estimators (50 to 300), the number of features to consider when looking for the best tree split (all features, square root and log base-2 of all features), and the depth of individual trees (3 to 10, or unlimited). Similarly, we tune the number of estimators in XG (50 to 300), the depth of each tree (3 to 25), the feature and sample subset size (0.5 to 1). Finally, for the NN, a computational limit of 250 epochs was assumed. A search between 1 to 3 for number of hidden layers was conducted, where each hidden layer has the size set to a half of the previous layer.[3] Different mini-batch sizes were tested (32 to 256), and L2 regularization penalty (10^{-6} to 10^{-1}) are also tuned. In each case, parameter tuning is limited to training data alone.

2) Performance Metrics: In cybersecurity applications of ML, detection rate (DR), which is also called recall, and false positive rate (FPR) are widely used [13].

$$DR = \frac{TP}{TP + FN}, \quad (2) \qquad FPR = \frac{FP}{TN + FP}, \quad (3)$$

where TP, TN, FP, FN are True Positive, True Negative, False Positive, and False Negative, respectively. In our work, TP represents the number of malicious samples that are correctly classified as "malicious", and FN represents the number of malicious samples that are incorrectly classified as "normal". On the other hand, TN (FP) are the numbers of normal data samples that are correctly (incorrectly) classified.

In addition to DR and FPR, we also report the system performance by Precision (Pr) and F1-score (F1). In particular cases, Receiver operating characteristic (ROC) curves and Area under the curve (AUC) are also presented.

$$Pr = \frac{TP}{TP + FP}, \quad (4) \qquad F1 = \frac{2}{Pr^{-1} + Recall^{-1}}, \quad (5)$$

Precision represents the percentage of malicious alarms generated by the system that are true. F1-score summarizes both DR, or recall, and Pr as a harmonic mean. Due to the extremely skewed data (Table II), a relatively small FPR may still translate to a large amount of false alarms. By reporting results using Pr and F1, the cost of false alarm investigation is better presented. For the sake of brevity, in the following, we report performance metrics in percent (%).

As mentioned in Section III-A, system performance is reported in terms of *both* data instances correctly detected (instance-based results), and users correctly detected, (user-based results). For user based results, a normal user is misclassified if at least one of their data instances is classified as "malicious", while a malicious insider is identified if at least one of their malicious data instances is classified as "malicious" by the system. We therefore have two sets of performance metrics: Instance based (IDR, IFPR, IPr, IF1) and User based (UDR, UFPR, UPr, and UF1).

To compare between the algorithms, data types, or experiment settings, we adopt F1 performance metric for its expressive power. Naturally a system achieves high F1 when it does well on both recall (DR) and precision, which means high malicious detection rate at a cost of low false alarm rate. Comparisons are supported by Wilcoxon sign-ranks test, which is the nonparametric analogue to the paired t-test for statistical significance tests [52].

Finally, to further analyze the insider threat cases and provide better insights into the effectiveness of the approach,

[3]Enforces a 'bottleneck' effect that encourages the network to discover a suitable encoding.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

we introduce the following measures: *detection delay* (DD) and *detection rate per detected malicious insider* (DR/DMI). DD can be defined as the time duration between the first malicious action performed by a malicious insider until he/she is detected (if ever). On the other hand, DR/DMI is the percentage of malicious instances detected per malicious user. For example, if a malicious insider performs data exfiltration over 5 weeks and only one user-week data instance of the user is flagged as "malicious", DR/DMI for this case is $1/5 = 0.2$. These additional metrics could be helpful in evaluating insider threat detectors' performances, where DD demonstrates how quickly the system is able to detect a malicious insider, and DR/DMI represents the extent of the user's malicious actions uncovered by the system.

C. Results by Training conditions

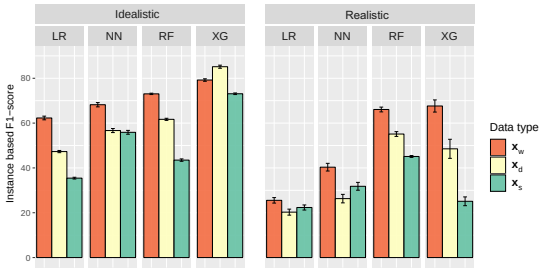


Fig. 4. Instance based F1-score by data types and algorithms under *realistic* and *idealistic* training conditions. Error bars show 95% confidence intervals.

1) Realistic vs Idealistic: As defined in Section IV-B, in this experiment we compare instance based results on user-session, user-day, and user-week data between the two training conditions (realistic vs idealistic) in order to demonstrate the challenges in applying ML solutions in real-world environments. Instance based results obtained from the two settings are shown in Table III and Fig. 4. It is clear that results obtained under the *idealistic* setting - 50% of all data instances are used for training, sampled across the entire temporal period - are significantly better on almost all measures than results obtained in the *realistic* setting - training data is limited to data of only 20% of users in the first half of the dataset. Performing Wilcoxon sign-ranks test returns $p = 0.003$. Thus, this rejects the null hypothesis: ML algorithms under the two training conditions (*idealistic* and *realistic*) perform similarly with regards to IF1 at the 1% significance level.

By employing *idealistic* training condition, most ML algorithms achieved near perfect IFPR, and higher IDR than under the *realistic* setting. The only exception is RF over user-session data, where IDR and IF1 obtained from *realistic* setting is better. Fig. 4 also shows that RF achieves the most similar performances between *idealistic* and *realistic* training conditions, which suggests RF as a good candidate for learning from limited data conditions. On the other hand, while XG performs better under the *idealistic* setting than the remaining algorithms, its performance degrades greatly under *realistic* training condition, especially on higher granularity data, i.e user-session data.

From a ML standpoint, all ML algorithms achieve relatively low instance based false positive rates under *realistic* condition, where IFPR < 0.5% in most cases. However, in practice, due to the fact that normal data instances account for more than 99.6% of all data instances, the amount of false positive alarms may still pose a challenge to analysts. For example, a 0.14% IPFR by NN on user-session data is equivalent to 1400 false malicious instance alerts.

The presented results highlight the challenges observed in many real-world applications of ML, especially in cybersecurity, where ground-truth is limited and typical ML setting (random training / testing splitting of data) can not be satisfied, given that such a split assumes global information. Thus, results obtained in a typical ML setting may not necessarily reflect real-world cybersecurity performances, and realistic settings need to be adopted in system design in order to ensure a smooth transition to deployment environments.

2) Learning algorithm – Supervised vs unsupervised: While the focus of this work is in employing ML algorithms for learning from limited amount of labeled data to detect unseen malicious insiders, in this section we compare in *realistic* training condition the performances of employed ML algorithms and Isolation Forest (IF) [53], a prominent unsupervised learning approach that has been employed in many network anomaly detection work recently [54]. IF assumes that the anomalous data instances are easier to isolate from the rest of the data than normal instances, hence shorter path length to the corresponding leaves of anomalous instances. For training IF models, the number of trees is tuned for each data type. We assume three different thresholds (1%, 5%, and 10%) for warning data instances as "anomaly", based on different available budgets for investigating flagged anomalous events. Table IV and Fig. 6 illustrate the results achieved by IF. The results clearly demonstrate that when label information, albeit limited, is available for training ML algorithms, guided search as in supervised learning will achieve superior performances, especially at very low FPRs.

D. Results by Data Granularity Levels

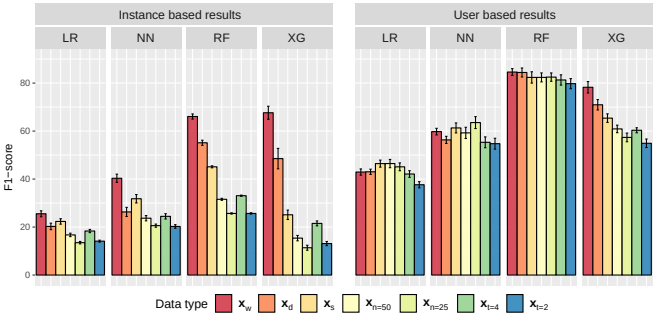


Fig. 5. Instance based and user based F1-score by data types and ML algorithms. Error bars show 95% confidence intervals.

In this section, we investigate the impact of different data granularity levels on performance of a ML based insider threat detection system under the *realistic* training condition.

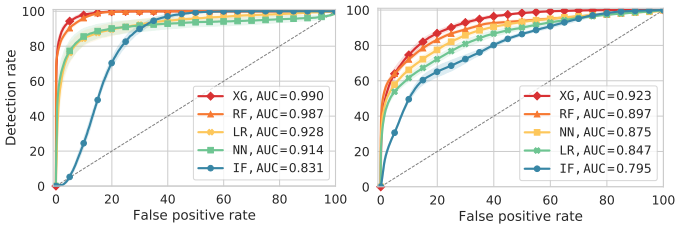
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

TABLE III
INSTANCE BASED RESULTS: REALISTIC VS IDEALISTIC

Setting	Alg.	User-Week				User-Day				User-Session			
		IFPR	IDR	IPr	IF1	IFPR	IDR	IPr	IF1	IFPR	IDR	IPr	IF1
Idealistic	LR	0.09	55.13	71.70	62.28	0.01	33.26	81.94	47.30	0.01	22.13	89.04	35.44
	NN	0.03	55.82	89.27	68.20	0.05	48.99	68.57	56.68	0.04	46.90	69.99	55.88
	RF	0.00	57.55	99.97	73.05	0.00	44.60	99.98	61.68	0.00	27.80	99.96	43.49
	XG	0.01	66.65	97.69	79.22	0.00	74.85	98.81	85.16	0.00	58.76	96.70	73.09
Realistic	LR	1.40	53.77	16.94	25.53	0.70	43.88	13.44	20.27	0.27	26.49	19.82	22.34
	NN	0.41	45.23	38.03	40.33	0.41	39.90	20.54	26.30	0.14	29.28	37.20	31.80
	RF	0.00	49.54	99.39	66.07	0.03	41.97	83.70	55.12	0.02	31.54	81.98	45.10
	XG	0.14	63.37	74.10	67.63	0.20	56.39	44.84	48.52	0.31	31.76	22.09	25.11

TABLE IV
INSTANCE BASED RESULTS BY ISOLATION FOREST

Threshold	User-Week			User-Day			User-Session		
	IFPR	IDR	IPr	IFPR	IDR	IPr	IFPR	IDR	IPr
1%	0.94	0.38	0.22	1.17	3.01	0.59	1.14	12.54	2.50
5%	4.98	6.31	0.62	4.96	24.30	1.09	4.90	31.66	1.44
10%	9.42	26.42	1.32	9.23	50.99	1.17	9.08	49.87	1.17

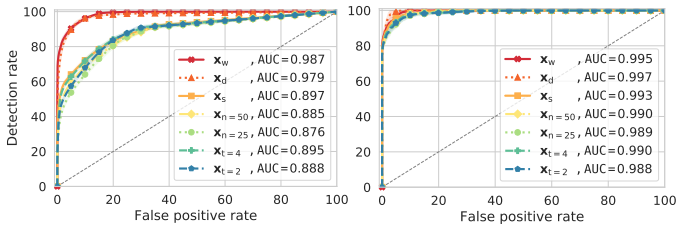


(a) User-week data (b) User-session data

Fig. 6. Instance-based ROCs and AUCs of ML algorithms on user-week and user-session data

1) Instance-based results: Instance based results are shown in Table V and Fig. 5. One noticeable trend overall is that ML algorithms' instance based performances are degrading (w.r.t. IDR and IF1) by higher data granularity levels. Specifically, significant differences can be observed when comparing instance based results (IF1) of different data granularity levels in almost all of the cases. For example, comparing RF's IF1 between user-week and user-session, or between user-session and user-subsession T2 both yield $p = 9e^{-5}$. Fig. 6 further demonstrates the observation, where AUCs are higher on user-week data than on user-session data. This can be explained through the amount of information embedded in each data instance of different data types (see §III-B), where coarse-grained data types, such as user-week and user-day, covers a longer period and summarize more behavioral information, i.e. user actions, than fine-grained data types, such as user-session and user-subsession. Furthermore, much larger instance count and higher imbalanced data distribution in fine-grained data types (Table IV-A) also likely contribute to the observed degradation in instance based results.

2) User-based results: Tables V and Fig. 5 and 7 show user based results by the ML algorithms on different data granularity levels. In contrast to the trend observed in instance based results, user based results (UDR, UF1) are generally more robust to different data granularity levels. Other than the XG algorithm, no large changes (>5%) can be found



(a) Instance-based (b) User-based

Fig. 7. Instance-based vs User-based ROCs and AUCs of RF on different data granularity levels

between the measures of the data types in most of the cases. Furthermore, despite a relatively low proportion of detected malicious insider data instances (Table V), the classifiers could learn to detect at least one malicious instance for most of the malicious insiders (80 to 90%). User based result reporting also adjusts false positive rates considerably. For example, the NN achieves only a 0.14% IFPR, but 3.44% UFPR on user-session data. These observations show the shortcomings of simply reporting results per data instance rather than per user, where the former may not necessarily demonstrate a true estimation of the detector's capability in detecting malicious users. In practice, it seems that user based metrics justify the use of fine-grained data types, such as user-session and user-subsession. On these data types, despite lower IF1 and IDR than for coarse-grained data types, the UF1 and UDR performance remains the same. Fine-grained data types also provide further advantages to detection systems, such as allowing a faster response (investigated in Section IV-E).

3) ML algorithms' performances: As discussed above, among the ML algorithms, RF presents the best results in terms of F1-score, precision, and false positive rates, both on user based and instance based metrics (Fig. 7). NN shows promising UDR – about 4% higher than RF – at a cost of higher UFPR. On the other hand, LR suffers from high UFPR and low F1-scores. Finally, while XG achieves the best performance (IF1) on user-week data, fine-grained data types have a much higher negative impact on it than on either RF or NN. Based on the observations, the rest of this section presents RF performances only, due to the space limitation.

E. Analysis of Insider Threat Scenarios

As mentioned in Section IV-A, there are four distinct insider threat scenarios in the dataset. In this section, we analyze

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

TABLE V
INSTANCE BASED RESULTS BY DATA GRANULARITY LEVELS AND ML ALGORITHM

Data type	Logistic Regression				Neural Network				Random Forest				XGBoost			
	IFPR	IDR	IPr	IF1	IFPR	IDR	IPr	IF1	IFPR	IDR	IPr	IF1	IFPR	IDR	IPr	IF1
$\mathbf{x}_w$	1.40	53.77	16.94	25.53	0.41	45.23	38.03	40.33	0.00	49.54	99.39	66.07	0.14	63.37	74.10	67.63
$\mathbf{x}_d$	0.70	43.88	13.44	20.27	0.41	39.90	20.54	26.30	0.03	41.97	83.70	55.12	0.20	56.39	44.84	48.52
$\mathbf{x}_s$	0.27	26.49	19.82	22.34	0.14	29.28	37.20	31.80	0.02	31.54	81.98	45.10	0.31	31.76	22.09	25.11
$\mathbf{x}_{n=50}$	0.14	16.72	17.60	16.72	0.09	20.24	32.39	23.69	0.01	19.96	80.09	31.57	0.27	22.12	12.14	15.38
$\mathbf{x}_{n=25}$	0.07	11.22	17.84	13.49	0.04	14.75	39.61	20.59	0.01	15.56	78.98	25.67	0.20	15.74	9.24	11.37
$\mathbf{x}_{t=4}$	0.13	18.20	19.13	18.35	0.09	22.02	29.77	24.46	0.01	21.65	74.77	33.06	0.12	21.45	22.93	21.53
$\mathbf{x}_{t=2}$	0.08	12.53	16.65	14.13	0.05	15.77	32.02	20.26	0.01	16.13	67.86	25.68	0.16	16.33	11.26	13.13

TABLE VI
USER BASED RESULTS BY DATA GRANULARITY LEVELS AND ML ALGORITHM

Data type	Logistic Regression				Neural Network				Random Forest				XGBoost			
	UFPR	UDR	UPr	UF1	UFPR	UDR	UPr	UF1	UFPR	UDR	UPr	UF1	UFPR	UDR	UPr	UF1
$\mathbf{x}_w$	7.77	91.31	28.12	42.93	2.82	77.00	49.00	59.77	0.02	73.85	99.33	84.60	1.00	82.00	75.52	78.25
$\mathbf{x}_d$	7.75	91.54	28.17	43.03	4.13	86.15	41.96	56.32	0.43	81.62	88.04	84.42	1.97	85.46	61.03	70.96
$\mathbf{x}_s$	6.54	89.00	31.49	46.44	3.44	87.54	47.47	61.31	0.78	84.62	81.45	82.36	2.49	82.77	54.39	65.39
$\mathbf{x}_{n=50}$	6.44	87.77	31.72	46.46	3.71	86.46	45.43	59.24	0.54	80.15	85.89	82.37	2.51	75.15	51.49	60.90
$\mathbf{x}_{n=25}$	6.55	85.54	30.73	45.10	2.83	83.54	51.89	63.54	0.51	79.85	86.49	82.49	2.58	69.77	48.91	57.36
$\mathbf{x}_{t=4}$	7.83	89.54	27.60	42.11	4.56	88.38	40.55	55.32	0.79	83.08	80.67	81.30	2.30	71.54	52.50	60.33
$\mathbf{x}_{t=2}$	9.36	89.62	23.87	37.63	4.47	86.15	40.38	54.74	0.92	82.92	77.80	79.77	2.66	66.62	46.94	54.91

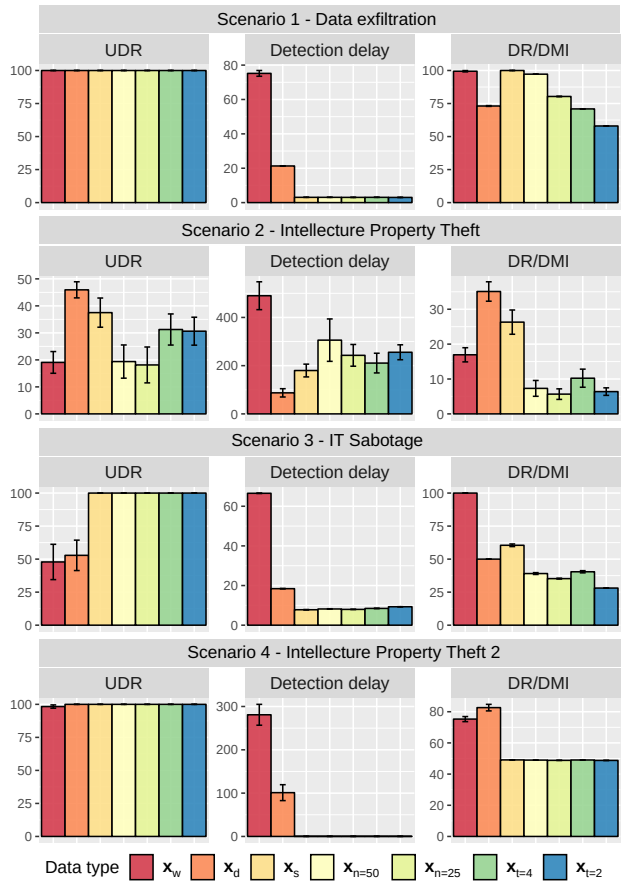


Fig. 8. Insider threat scenario detailed results by RF. Error bars depict 95% confidence intervals. UDR: User-based Detection Rate, DR/DMI: detection rate per detected malicious insider. The unit of detection delay (DD) is hour. Note that for DD, lower is better.

the performance of the ML models on each scenario based on UDR, detection delay, and detection rate per detected malicious insider. The results by RF are shown in Fig. 8.

1) Scenarios 1 and 4 – Data exfiltration: It is readily apparent that insider threat scenarios 1 and 4 are reliably detected on most of the data types. Furthermore, these scenarios are detected quickly by the system with high confidence, particularly on user-session and user-subsession data granularity levels. In these cases, the detection delays by RF are 3 and 0.23 hours (14 minutes) on average, and DR/DMI mostly above 70% and 48%, respectively. This suggests that the two scenarios can be easily detected using ML based systems. The insider threat scenarios depict data exfiltration attempts. In scenario 1, "user begins to log in after hours, using a USB, and uploading data to wikileaks.org", and scenario 4 shows an another example: "user logs into another user's machine and searches for interesting files, emails them to his/her home email" [33]. Thus, it may be assumed that well chosen / derived features (Section III-B) contribute to the detection of behavioral shifts (abrupt changes in the user behavior) in these scenarios. Some examples of these are in *after workhour* information, or in *PC* and *external email* information.

2) Scenario 2 – Intellectual property thief: Scenario 2 appears to be the hardest to detect using ML based detectors. The best results are obtained on user-day data, where 46% and 65% of malicious insiders are detected using RF and NN respectively. Detection also takes a longer time and with less confidence (DR/DMI < 36%) – averaging 154 hours (NN) and 86 hours (RF) after the first malicious behavior. Dataset description states that in this scenario, "user surfing job websites and soliciting employment from a competitor, and use a thumb drive to steal data" [48]. This implies

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

less obtrusive actions than other scenarios. Furthermore, the insiders in scenario 2 possibly hid their malicious actions intentionally by performing them over a longer period of time (2 months in average, §IV-A), hence making it harder to detect.

3) Scenario 3 – IT sabotage: Scenario 3 represents an IT sabotage behavior, where "disgruntled system administrator uses keylogger to collect keylogs then log in as his supervisor to send out an alarming mass email" [33]. Based on Fig. 8, it is apparent that this scenario can be detected easily with fine-grained data types (user-session and user-subsession). On the other hand, coarse-grained data types result in lower performance under this scenario. This observation can be explained through distinct characteristics particular to this scenario, where a malicious user performs only a small number of actions (download keylogger, copy to USB) under their own account. Fine-grained data types are able to isolate those activities in terms of their short duration and PC specific nature, leading to high detection rates. However, it should be noted that although this scenario can be detected effectively using fine-grained data types, detection delay and DR/DMI are not as good as that of scenarios 1 and 4, at approximately 8 hours and 40%, respectively.

4) Traitors vs masqueraders: Depends on the modeling approach, scenarios 3 and 4 may be considered masqueraders [55], [8], as they involve insiders performing malicious actions on other users' machines. The two remaining scenarios are purely traitors' actions. Results shows that traitors' actions – when performed in a less obtrusive manner and over a long period – are harder to detect (Scenario 2, Fig. 8). On the other hand, both masquerader and traitor cases (scenarios 1, 3, and 4) can be detected easily using our system if significant change in user behaviors can be detected through the ML models.

5) Data granularity effects: Finally, with respect to granularity levels, fine-grained data types perform well with high malicious insider DR, low detection delay, and high DR/DMI on almost all situations. However, user-subsession data types show no advantage over user-session data in all measures. On coarse-grained data types, user-day data has an edge in detecting insider threat scenario 2. On the other hand, when user-week data is used, lower performance results, especially in terms of detection delay (>280 hours and 360 hours in scenarios 4 and 2), despite the best instance based performances (Table V).

F. Test Results on Different Organizational Data

As mentioned in Section IV-A, there are different versions of the CERT insider threat datasets. In the previous sections, training and test partitions are defined in terms of CERT r5.2. We will now train ML classifiers on CERT r5.2 and test them on CERT r5.1 and r6.2. CERT r5.1 simulates a similar organizational structure as CERT r5.2, with the same number of users, while CERT r6.2 has a different organizational structure and more users (4000), compared to CERT r5.2. Both CERT r5.1 and r6.2 simulate only one malicious user per insider threat scenario, significantly reducing the proportion of malicious data, which makes the detection task more challenging. Table VII presents the results of these tests.

TABLE VII
USER-BASED TEST RESULTS OF THE TRAINED MODELS ON CERT R5.1 & CERT R6.2

Test data	Data type	Neural Network				Random Forest			
		UFPR	UDR	UPr	UF1	UFPR	UDR	UPr	UF1
CERT r5.1 (2000 users, 4 malicious insiders)	$\mathbf{x}_w$	3.56	70.00	3.75	7.12	0.05	55.00	75.42	62.16
	$\mathbf{x}_d$	5.87	63.75	2.06	3.99	0.68	65.00	21.81	30.44
	$\mathbf{x}_s$	5.14	76.25	2.84	5.48	0.84	77.50	22.15	32.68
	$\mathbf{x}_{n=50}$	4.50	75.00	3.28	6.28	0.58	75.00	31.09	41.62
	$\mathbf{x}_{n=25}$	3.57	75.00	4.33	8.15	0.67	75.00	28.13	37.97
	$\mathbf{x}_{t=4}$	5.78	80.00	2.71	5.24	0.96	76.25	20.31	29.97
	$\mathbf{x}_{t=2}$	6.18	81.25	2.55	4.94	1.17	75.00	13.87	22.90
CERT r6.2 (4000 users, 5 malicious insiders)	$\mathbf{x}_w$	0.05	10.00	26.10	13.39	2.75	52.00	4.43	7.79
	$\mathbf{x}_d$	2.03	40.00	2.97	5.45	2.99	62.00	5.96	10.10
	$\mathbf{x}_s$	18.57	59.00	0.34	0.68	16.02	57.00	0.38	0.75
	$\mathbf{x}_{n=50}$	17.58	63.00	0.38	0.75	6.18	61.00	1.36	2.65
	$\mathbf{x}_{n=25}$	16.46	57.00	0.38	0.75	6.51	60.00	1.17	2.30
	$\mathbf{x}_{t=4}$	22.56	59.00	0.26	0.52	8.16	60.00	0.92	1.81
	$\mathbf{x}_{t=2}$	26.29	60.00	0.21	0.43	8.38	59.00	0.87	1.71

On CERT r5.1, it is apparent that the models perform well with low false alarm rates and 75% malicious insider detection rate. Typically on all data types, only the insider in scenario 2 is missed. It is noteworthy that while UDR and UFPR are similar to that observed on CERT r5.2, UPr and UF1 are lower, given the lower number of malicious users in CERT r5.1. On the other hand CERT r6.2 appears to present new challenges, possibly due to a different organizational structure. Notably, a new insider threat scenario in CERT r6.2 (scenario 5) is undetected. This scenario depicts behaviors of "a user decimated by layoffs uploads documents to Dropbox, planning to use them for personal gain" [33]. Not only does this scenario represent a novel malicious behavior to the trained model, it also shows less obtrusive actions than the remaining four scenarios.

On data granularity, user-session shows lower performance than the other data types from CERT r6.2. This suggests that on a different organization with different user behavior models, a session of user data may represent a different course of actions. In this case, a more aggregated data type, such as user-day and/or user-week may demonstrate better results.

Overall, results in this section indicate that models trained for insider threat detection in an organization can only be used as an initial detection step in a different environment. Specific models need to be re-trained from ground up or evolved from existing models for better accuracy. Furthermore, anomaly detection is needed to identify novel malicious behaviors.

V. DISCUSSION

A. DR - FPR trade-off

Throughout the experiments and results presented above, it is clear that there is a trade-off between the ability to detect malicious behaviors (DR) and maintaining a low false alarm rate (FPR). RF achieves the lowest FPRs and good DRs in most of the cases, explaining the good performances under the F1-score. Wilcoxon sign-ranks tests between RF and LR, NN, XG return p values 0.015, 0.015 and 0.03, respectively. This provides strong evidence against the null hypothesis that the methods are equivalent with respect to IF1. NN and LR post higher malicious DR at a cost of lower precision. Table V shows that instance based false alarm rates are approximately

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

5/6 for LR, and 2/3 for NN. In comparison, the false alarm rate is only 1/5 for RF, both for instance based and user based. On the other hand XG shows good results on week data and in the *idealistic* condition only, which suggests that it may not be a suitable candidate for insider threat detection, under limited ground truth conditions.

On data granularity levels, the experiments show that data extracted in user-session format allows good detection performance and low delay in most of the cases. Specifically, using RF classifier, 85% of malicious users are detected at a low cost of 0.78% false malicious user alarms. Classifiers using RF on user-session data also achieves the lowest delay in detecting most of the insider threat scenarios. User-day data may provide an advantage over user-session data in detecting scenario 2, while user-week data shows promising results on unseen data of different organizational structure (Table VII). On the other hand, pre-processing user data into subsessions does not improve the results w.r.t. session data in any metrics. This is probably due to the lower amount of information embedded in each instance of subsession data.

B. Data Imbalance Problem

As the data is highly imbalanced (Table II), in this section we discuss the effect on ML algorithms. In this work, training the ML algorithms as binary classifiers, in which minority malicious classes are combined to a single positive class, may help reducing the negative impacts of data imbalance. Furthermore, good results obtained by RF, as shown in IV, may be attributed to its resistance to data imbalance [56].

We further examine the ML algorithms performances with oversampling techniques. Results with random oversampling (increase positive class to 20% of training data) are shown in Table VIII. Although not shown due to space limitation, other oversampling techniques, such as SMOTE [57], and different training settings yield very similar results. Overall, it seems that the DR-FPR trade-off is maintained, and the results obtained with oversampling is similar to what achieved with original training data composition with an adjusted decision threshold (Fig. 6 and 7).

TABLE VIII
USER-BASED RESULTS WITH OVERSAMPLING ON USER-SESSION DATA

Algorithm	UFPR	UDR	UPr	UF1
LR	32.55	100.00	6.90	12.91
NN	17.26	95.08	14.03	24.44
RF	1.36	86.46	72.00	77.63
XG	5.04	88.31	37.83	52.72

C. Overfitting and Effect of Training Data

We performed permutation test [58] in order to validate the trained classifiers. Test results in all case return p-values less than 0.01 w.r.t F1, indicating the ML algorithms have learned to recognize meaningful classifier from training data without overfitting.

We further explore the effect of training data by comparing test results on test data of "normal" training users

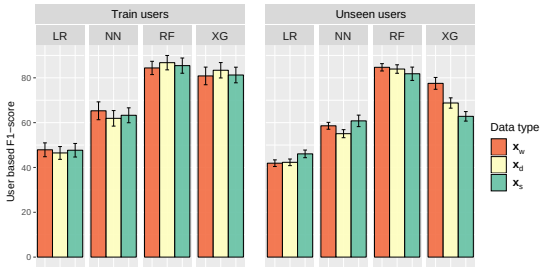
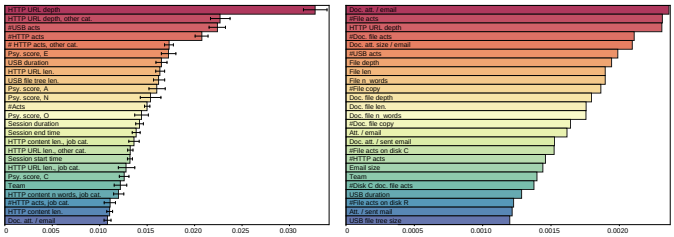


Fig. 9. User based F1-score on test data of "normal" train users and unseen users. Error bars show 95% confidence intervals.



(a) Feature importance by RF (with 95% CIs)

(b) Feature importance by Information gain

Fig. 10. Feature importance in user-session data

(user set U_{train}), and of unseen test users (user set U_{unseen}, $U_{unseen} \cup U_{train}$ = All test users). Recall that in the *realistic* training condition, data from only a limited subset of users are included for training. U_{unseen} is the set of users whose data in the first 37 weeks are not used in training the ML models. Fig. 9 shows user-based F1-score on test data of the two user sets. Overall, the ML algorithms are able to generalize well, where UF1 is only slightly higher on U_{train} than on U_{unseen} (p=0.012 w.r.t UF1). RF shows most similar performance between U_{unseen} and $U_{results}$ (p = 0.66 w.r.t UF1), which suggests that it is more robust to overfitting problem than the remaining algorithms.

D. Feature Analysis

We perform feature analysis based on the feature importance output of RF, which is calculated using Gini impurity, and information gain. Definition of the metrics can be found in [39]. Fig. 10 illustrates the 25 most important features identified by the two metrics in user-session data.

Fig. 10a shows that HTTP and USB features are important to RF classifiers, specifically URL depth, and in job category. This is expected, considering the malicious actions in the insider threat scenarios (Section IV-E), where web and usb are two common means for carrying out malicious actions, such as copying files, and searching job offers. Features related to HTTP category "other" are important too. This suggests the use of a more detailed HTTP categorization model for better description of user activities might be appropriate. Furthermore, the importance of team and psychometric scores indicates the usefulness of the constructed user context models (Section III-B). On the other hand, information gain shows a slightly different picture, Fig. 10b, where features describing document related activities are the most important ones.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.2967721, IEEE Transactions on Network and Service Management

TABLE IX
RESULTS OF OTHER WORKS

Work	Data & Preprocessing	Method	Results
[31]	CERT r4.2, *monthly* data instances	Bayesian network models are trained on 3 months of monthly user data instances and test on 5 months	Instance-based Recall 100%, Precision 29.8%
[19]	CERT r4.2, weekly data instances	Hidden Markov Models for modelling each user weekly activity sequences	Instance-based AUC = 0.83
[21]	Vegas dataset, daily data instances	Supervised quitter detection and unsupervised insider threat detection	Instance-based AUC = 0.77
[28]	CERT r6.2, daily data instances	unsupervised deep learning (RNN and LSTM)	Instance-based recall up to 35.7% at 1000 alerts daily
[30]	CERT r2, session of user data	Streaming anomaly detection algorithms	Instance-based recall $\approx$ 50%, precision $\approx$ 8%
[18], [22]	other data, not publicly available	Ensemble of anomaly detection approaches	Instance-based AUC up to 0.97 on monthly data

E. Comparison with Other Works

While there are no other work in the literature for exact comparison in data, methods, and even the aims, we summarize some results from related work in Table IX for a general comparison.

Compared to the previous works, we are able to match or better previous results while learning from a much more limited amount of user data. Furthermore, many detailed analysis in user-based results and insider threat scenarios are performed for better insights into the system's detection performance.

VI. CONCLUSION AND FUTURE WORK

In this research, a ML based system for insider threat detection in networked systems of organizations is presented. The work benchmarks four different ML algorithms – LR, NN, RF, and XG – on multiple levels of data granularity, limited ground truth, and different training scenarios in order to support cybersecurity analysts for detecting malicious insider behaviors in unseen data. Evaluation results show that the proposed system is able to successfully learn from the limited training data and generalize to detect new users with malicious behaviors. The system achieves a high detection rate and precision, especially when user based results are considered.

Among the four ML algorithms, RF clearly outperforms the other algorithms, where it achieves high detection performance and F1-score with low false positive rates in most of the cases. On the other hand, NN allows slightly better insider threat detection performance at a cost of higher false alarm rates. On data granularity, user-session data provides high malicious insider detection rates and the minimum delay. On the other hand, user-day data shows slightly better performance on detecting a particular insider threat scenario, aka scenario 2 - intellectual property theft. Moreover, it seems to be more generalizable when applied to a different organization's data.

Future work will investigate the use of temporal information in user actions. Specifically, all the models in this work provided labels based on a state description limited to a single exemplar. Enabling models to see multiple exemplars or retain state (recurrent connections) has the potential to enable models to make non-Markovian decisions.

ACKNOWLEDGMENT

This research is supported by Natural Science and Engineering Research Council of Canada (NSERC). Duc C. Le gratefully acknowledges the supports of the Killam Trusts, Mitacs, and the province of Nova Scotia. The research is conducted as part of the Dalhousie NIMS Lab at: https://projects.cs.dal.ca/projectx/.

REFERENCES

[1] Meritalk, "The 2017 federal insider threat report," Symantec, Tech. Rep., 2017. [Online]. Available: https://www.meritalk.com/study/inside-job-the-sequel/

[2] Crowd Research Partners, "2018 insider threat report," 2018. [Online]. Available: https://crowdresearchpartners.com/insider-threat-report/

[3] CSO, U.S. Secret Service, CERT Division of SEI-CMU, KnowBe4, "The 2018 U.S. state of cybercrime survey," IDG, Tech. Rep., 2018. [Online]. Available: https://www.idg.com/tools-for-marketers/2018-u-s-state-of-cybercrime/

[4] M. L. Collins, M. C. Theis, R. F. Trzeciak, J. R. Strozer, J. W. Clark, D. L. Costa, T. Cassidy, M. J. Albrethsen, and A. P. Moore, "Common sense guide to mitigating insider threats, fifth edition," The CERT Insider Threat Center, Tech. Rep. CMU/SEI-2015-TR-010, 2016.

[5] 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.

[6] National Cybersecurity and Communications Integration Center, "Combating the Insider Threat," The US Department of Homeland Security, Tech. Rep., 2014.

[7] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and Preventing Cyber Insider Threats: A Survey," *IEEE Commun. Surveys Tuts.*, 2018.

[8] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Computing Surveys*, vol. 52, no. 2, pp. 30:1–30:40, Apr. 2019.

[9] K. Padayachee, "An assessment of opportunity-reducing techniques in information security: An insider threat perspective," *Decision Support Systems*, vol. 92, pp. 47–56, 2016.

[10] F. L. Greitzer and R. E. Hohimer, "Modeling human behavior to anticipate insider attacks," *J. Strategic Security*, vol. 4, no. 2, 2011.

[11] P. Legg, N. Moffat, J. Nurse, J. Happa, I. Agrafiotis, M. Goldsmith, and S. Creese, "Towards a conceptual model and reasoning structure for insider threat detection," *J. Wireless Mobile Netw., Ubiquitous Comput., & Depend. Appl. (JoWUA)*, vol. 4, no. 4, pp. 20–37, 2013.

[12] O. Brdiczka, J. Liu, B. Price, J. Shen, A. Patil, R. Chow, E. Bart, and N. Ducheneaut, "Proactive insider threat detection through graph learning and psychological context," in *IEEE SPW*, 2012, pp. 142–149.

[13] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2016.

[14] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 2014.

[15] W. Eberle, L. Holder, and D. Cook, "Identifying threats using graph-based anomaly detection," in *Machine Learning in Cyber Trust*. Springer, 2009, pp. 73–108.

[16] D. Caputo, M. Maloof, and G. Stephens, "Detecting insider theft of trade secrets," *IEEE Security & Privacy Magazine*, vol. 7, no. 6, 2009.

[17] P. Parveen and B. Thuraisingham, "Unsupervised incremental sequence learning for insider threat detection," in *IEEE Int. Conf. on Intelligence and Security Informatics*, 2012.

[18] T.E. Senator *et al.*, "Detecting insider threats in a real corporate database of computer usage activity," in *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2013, pp. 1393–1401.

[19] T. Rashid, I. Agrafiotis, and J. R. Nurse, "A new take on detecting insider threats," in *Int. Workshop on Managing Insider Security Threats*, 2016.

[20] H. Eldardiry, E. Bart, J. Liu, J. Hanley, B. Price, and O. Brdiczka, "Multi-domain information fusion for insider threat detection," in *IEEE Security and Privacy Workshops (SPW)*, 2013.

[21] G. Gavai, K. Sricharan, D. Gunning, J. Hanley, M. Singhal, and R. Rolleston, "Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data," *JoWUA*, vol. 6, no. 4, pp. 47–63, 2015.

[22] H. G. Goldberg, W. T. Young, M. G. Reardon, B. J. Phillips, and T. E. Senator, "Insider threat detection in PRODIGAL," in *Annual Hawaii Int. Conf. on System Sciences*, 2017.

[23] Defense Advanced Research Projects Agency, "Anomaly detection at multiple scales (ADAMS)," https://www.darpa.mil/program/anomaly-detection-at-multiple-scales.

[24] M. B. Salem and S. J. Stolfo, "Modeling user search behavior for masquerade detection," in *Int. Symposium on Recent Advances in Intrusion Detection*. Springer Berlin Heidelberg, 2011, pp. 181–200.

[25] F. Toffalini, I. Homoliak, A. Harilal, A. Binder, and M. Ochoa, "Detection of masqueraders based on graph partitioning of file system access events," in *IEEE SPW*, 2018, pp. 217–227.

[26] D. C. Le and A. N. Zincir-Heywood, "Evaluating insider threat detection workflow using supervised and unsupervised learning," in *IEEE SPW*, 2018.

[27] D. C. Le and A. N. Zincir-Heywood, "Machine learning based insider threat modelling and detection," in *IFIP/IEEE Int. Symposium on Integrated Network Management*, Washington DC, USA, 2019.

[28] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *AAAI Workshop on Artificial Intelligence for Cyber Security*, 2017.

[29] D. C. Le, S. Khanchi, A. N. Zincir-Heywood, and M. I. Heywood, "Benchmarking evolutionary computation approaches to insider threat detection," in *ACM Genetic and Evolutionary Computation Conf.*, 2018.

[30] B. Bose, B. Avasarala, S. Tirthapura, Y. Y. Chung, and D. Steiner, "Detecting insider threats using radish: A system for real-time anomaly detection in heterogeneous data streams," *IEEE Systems Journal*, 2017.

[31] S. C. Roberts, J. T. Holodnak, T. Nguyen, S. Yuditskaya, M. Milosavljevic, and W. W. Streilein, "A model-based approach to predicting the performance of insider threat detection systems," in *IEEE SPW*, 2016.

[32] W. Meng, K. R. Choo, S. Furnell, A. V. Vasilakos, and C. W. Probst, "Towards bayesian-based trust management for insider attacks in healthcare software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 2, pp. 761–773, Jun. 2018.

[33] CERT and ExactData, LLC, "Insider threat test dataset," https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099.

[34] R. Werlinger, K. Hawkey, K. Muldner, P. Jaferian, and K. Beznosov, "The challenges of using an intrusion detection system: Is it worth the effort?" in *Symposium on Usable Privacy and Security*. USENIX, 2008.

[35] K. Ball, "Workplace surveillance: an overview," *Labor History*, vol. 51, no. 1, pp. 87–106, 2010.

[36] K. Kenthapadi, I. Mironov, and A. G. Thakurta, "Privacy-preserving data mining in industry," in *ACM Int. Conf. on Web Search and Data Mining*, 2019, pp. 840–841.

[37] F. Haddadi and A. N. Zincir-Heywood, "Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification," *IEEE Systems Journal*, vol. 10, no. 4, pp. 1390–1401, Dec. 2016.

[38] D. C. Le, A. N. Zincir-Heywood, and M. I. Heywood, "Unsupervised monitoring of network and service behaviour using self organizing maps," *J. Cyber Security and Mobility*, vol. 8, no. 2, 2018.

[39] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. Pearson, 2018.

[40] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Int. Conf. on Machine Learning (ICML)*, 2006, pp. 161–168.

[41] A. Barron, "Approximation and estimation bounds for artificial neural networks," *IEEE Trans. on Inf. Theory*, vol. 39, pp. 930–944, 1993.

[42] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. on Learning Representations*, 2015.

[43] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Int. Conf. on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[44] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, 2001.

[45] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *KDD*, 2016, pp. 785–794.

[46] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "Xgboost classifier for ddos attack detection and analysis in sdn-based cloud," in *IEEE Int. Conf. on Big Data and Smart Computing*, 2018, pp. 251–256.

[47] L. Torlay, M. Perrone-Bertolotti, E. Thomas, and M. Baciu, "Machine learning–XGBoost analysis of language networks to classify patients with epilepsy," *Brain Informatics*, vol. 4, no. 3, pp. 159–169, Sep. 2017.

[48] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *IEEE SPW*, 2013, pp. 98–104.

[49] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research (JMLR)*, vol. 12, 2011.

[50] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, Aug. 1989.

[51] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *ICML*, 2013, pp. 115–123.

[52] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *JMLR*, vol. 7, pp. 1–30, 2006.

[53] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *IEEE Int. Conf. on Data Mining*, 2008, pp. 413–422.

[54] J. Ahmed, H. H. Gharakheili, Q. Raza, C. Russell, and V. Sivaraman, "Monitoring enterprise dns queries for detecting data exfiltration from internal hosts," *IEEE Trans. Netw. Service Manag.*, 2019.

[55] M. B. Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research," in *Insider Attack and Cyber Security: Beyond the Hacker*. Boston, MA: Springer US, 2008, pp. 69–90.

[56] T. M. Khoshgoftaar, M. Golawala, and J. V. Hulse, "An empirical study of learning from imbalanced data using random forest," in *IEEE Int. Conf. on Tools with Artificial Intelligence*, 2007.

[57] 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, Jun. 2002.

[58] M. Ojala and G. C. Garriga, "Permutation tests for studying classifier performance," *JMLR*, vol. 11, pp. 1833–1863, Aug. 2010.

Duc C. Le is a Ph.D. student at Dalhousie University, Halifax, Canada. He received the Master degree in computer science from the same university in 2017, and the B. Eng. degree in electronics and telecommunications engineering from Posts and Telecommunications Institute of Technology, Ha Noi, Vietnam, in 2015. His research focuses on machine learning and its applications in computer and network security and analysis.

Nur Zincir-Heywood is a Full Professor of Computer Science with Dalhousie University, Canada. Her research interests include machine learning and data mining techniques for networks, services and cybersecurity. She has published over 200 fully reviewed papers and has been a recipient of several best paper awards. She is an Associate Editor of the IEEE Transaction on Network and Service Management and the General Co-Chair of the International Conference on Network and Service Management 2020. She is a member of the IEEE and the ACM and a recipient of the 2017 DNS Women Leaders in the Digital Economy Award.

Malcolm I. Heywood (M'95–SM'06) received the Ph.D. degree, his work on movement invariant pattern recognition using neural networks from the University of Essex, Colchester, U.K., in 1994. He is currently a Full Professor of Computer Science at Dalhousie University, Halifax, NS, Canada. His current research investigates the application of coevolutionary methods to reinforcement learning tasks as encountered in high-dimensional spaces, and streaming data applications (Intrusion Detection and Financial Services). Dr. Heywood is a member of the Editorial Board for GPEM (Springer). He was a Co-Chair for the GECCO GP Track in 2014 and Co-Chaired the European Conference on Genetic Programming in 2015 and 2016. He was a co-recipient of the Humies Silver Medal in