International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:13, No:10, 2019

Design of an Ensemble Learning Behavior Anomaly Detection Framework

Abdoulaye Diop, Nahid Emad, Thierry Winter, Mohamed Hilia

Abstract—Data assets protection is a crucial issue in the cybersecurity field. Companies use logical access control tools to vault their information assets and protect them against external threats, but they lack solutions to counter insider threats. Nowadays, insider threats are the most significant concern of security analysts. They are mainly individuals with legitimate access to companies information systems, which use their rights with malicious intents. In several fields, behavior anomaly detection is the method used by cyber specialists to counter the threats of user malicious activities effectively. In this paper, we present the step toward the construction of a user and entity behavior analysis framework by proposing a behavior anomaly detection model. This model combines machine learning classification techniques and graph-based methods, relying on linear algebra and parallel computing techniques. We show the utility of an ensemble learning approach in this context. We present some detection methods tests results on an representative access control dataset. The use of some explored classifiers gives results up to 99% of accuracy.

Keywords—Cybersecurity, data protection, access control, insider threat, user behavior analysis, ensemble learning, high performance computing.

I. Introduction

IN the information and technology domain, data assets and their usage represent an immensely lucrative market. Cyber threats which target these resources are continuously becoming a more significant issue. Companies understand the strategic importance to heavily protect their information systems and use cybersecurity tools to shield themselves against all kinds of cyber attacks. Hackers understand the value of data assets and are continuously upgrading and widening their attack options. For a long time, the main focus of security specialists was the defense against external threats. They used tools based on databases of known threats patterns, and experts written-rules against cyber attacks. They also monitored access to their internal networks, servers, and endpoints, to control the access to their data, using intrusion detection systems(IDS).

Identity and access management (IAM) solutions, also known as logical access control tools, were developed to restrict the access of organization information assets to only authorized individuals. These tools offer solutions to control access to critical information and services. Their use reduced the risk of loss, misuses, and sabotage of data drastically. However, these solutions cannot identify individuals that maliciously use company assets when they own legitimate access rights. In the cybersecurity domain, this issue is defined as an insider threat problem.

In a company environment, insiders are employees who misuse their access rights. A typical example of insider activities is the illegal share of confidential business information to competitor organizations in exchange for compensation (i.e. industrial espionage). Another example is the sabotage of an organization proprietary tools by an insider hired by a rival company. According to IBM [1] in 2015, 60% of cyber threats were due to insiders (i.e. 44.5% of malicious insiders and 15.5% inadvertent actors). In 2018 studies from the *Ponemon Institute* evaluated the loss related to insider threat accidents for some relatively big company, in the range of millions of dollars [12].

The upsurge of insider threats made companies realize the need to protect themselves efficiently against the internal as well as external threats. The only way to counter the security problem is through employee monitoring and behavior analysis. Cyber attacks specialists developed tools such as user and entity behavior analytics (UEBA) software to monitor company systems and their employee's activities. Their goal is to detect if their behaviors are normal or anomalous and harmful to their organizations. User and entity behavior analytics is a term coined by Gartner in 2015 [14]. It is referring to the use of advanced analytics methods to detect insiders efficiently. Before the advent of data analysis methods, these tools were primarily based on expert written rules. They now integrate more intelligent and adaptive algorithms. According to Gartner [13], most of the UEBA software uses a combination of analysis techniques such as machine learning, and statistical analysis.

In this paper, we present the step toward the construction of a behavior anomaly detection framework combining machine learning classification methods, with graph-based methods relying on linear algebra and parallel computing techniques. The proposed software structure would utilize concepts such as ensemble learning and high-performance computing to manage big data volume of user activities data and get better behavior classification results. We also proposed insights on intelligent risk scoring approaches to measure insiders threat levels. Our contribution lies into the exploration and the test of multiple behavior analysis methods and the proposition of their combination using parallel implementation techniques, as an all-around better insider detection tools.

The rest of this paper is organized as follows. Section II

This project is funded by Atos, and the University of Versailles Paris Saclay.

Abdoulaye Diop is with the Li-PaRAD and Maison de la Simulation laboratories, and Atos Evidian R&D, Les Clayes-sous-Bois 78340, France (e-mail: mamadou-abdoulaye.diop@atos.net).

Nahid Emad is a member of the Li-PaRAD and Maison de la simulation laboratories, Versailles 78310 (e-mail: nahid.emad@uvsq.fr).

Thierry Winter is with Atos Evidian R&D, Les Clayes sous Bois 78340, France (e-mail: thierry.winter@atos.net).

Mohamed Hilia is with the Ippon Technologies, Paris 75016, France (e-mail: mhilia@ippon.fr).

presents some related work. In Section III, we define the insider threat problem and offer a small review of the detection models. Section IV presents the ideas of framework models and explores of some behavior anomaly detection methods. In Section V, we explain the programming paradigm we advocate and the corresponding implementation. Section VI will present some classification results on two access control datasets collected from an IAM software editor. Finally, Section VII concludes this article and give indications about the future directions of our work. I

II. Related Work

In behavior analysis, many works were realized in the intrusion detection systems field [4]. IDS vendors are the first to use machine learning for anomaly detection. They understood the added value of machine learning in the cybersecurity domain. Their algorithms monitor and profile companies internal networks to detect anomalies. They can discover new zero-day attacks, and with training time, most of the network-based cyber attacks (e.g. distributed and non-distributed denial of service attacks, port scan attacks).

Sun et al. [10] presented an unsupervised ensemble based outliers detection framework, which is based on the isolation forest (IForest) algorithms to detect insiders in a company. They tested their solution on a dataset of the staff accessing a payroll system. Their algorithm extracts feature sets and build users profiles based on the collection of extended isolation forest trees. Those trees are classic isolation trees modified to support categorical data. An isolation forest is an anomaly detection algorithm based on the use of multiple binary search trees, and the observation of the branches depth. Branches with small extent, when compared to the average tree branches depth, are considered anomalous [22][10]. One drawback, using an isolation forest algorithm lies in the fact that the training data has to be considered as healthy.

Moriano et al. [11] used bipartite graphs to detect insider threat. They proposed to identify malicious behavior over time considerations. They studied precipitating events and used them to detect anomalous behavior. They defined this type of events as ”key events that have the potential to trigger insiders to become a threat to their employer” [11]. They applied their model on a dataset of a version control tool for software development. They compared the volume of interactions between the users and the components before and after precipitating events. They found out that it was increasing, hence establishing the precipitating events as a sign off an upcoming insider attack. This detection method may encounter some issues if there is a gradual and non-abrupt change in the behavior of an employee with malicious intent. If there is no found precipitating events the model would be unable to detect the insider attack since there is no significant trigger.

Gamachchi et al. [8] proposed to use a GPU to extract graph-based features from a graph built with a multidimensional dataset from the CERT of Carnegie Melon university data. They fed the features to an IForest algorithm to detect outliers without profiling normal behavior. This technique focuses on the study local graph anomalies as a potential indicator of deviant behavior. In this work, the behavioral study parameters are chosen in a certain way by a human operator who is an expert in the field(i.e. graphs initial features are chosen) This method can be sufficient to stop insider based on expert experience but does not take into account all the parameters that could be indicators of internal threats. Some more elaborate threats and a new types of attack could slip through this detection method.

Chen et al. [9] used bipartite graphs to map user access log, then use the cosine similarity method to get the similarity between users and to detect if specific access is malicious or not. They based their approach on a correlation method used in collaborative filtering for recommender systems. These systems are dependant on the amount of information used to determine the model for reliable prediction results. It’s a common problem for recommendation systems.

Haidar et al. [7] proposed an ensemble based methods, using base classifiers such as a one class support vector machine (OCSVM) and an IForest on data clusters (k-data). Besides, they added a progressive update method using false positive (FP) chunks (i.e. false positive results identify by a human domain expert) to refine their pre-generated models. After using their base classifiers on the entire dataset, they executed a class decomposition method using the k-means algorithm on the data labeled as normal behavior to get k-clusters. On those clusters, they applied the base algorithm again, to detect anomalies at a more local level. They fine-tuned their algorithm by oversampling the FP results which as the effect to create a more accurate the decision boundary. This algorithm presents an interesting idea to solve the internal threat problem, but it requires the intervention of a human operator for his optimal functioning.

Parveen et al. [5] proposed an ensemble learning based framework. It combined an unsupervised learning method using a graph-based anomaly detection technique based on minimum description length (MDL)[25] and an OCSVM as a supervised learning method. They prove that an OCSVM outperforms classic support vector machine and their graph-based anomaly detection methods. They previously proposed a model solely based on the combination of multiple one class support vectors machine and a stream mining approach to take account of behavior changes over time (i.e. concept drift) [6]. Starting with an ensemble learning method, Parveen et al. decided to use a traditional voting method to detect insiders. Using this kind of approach, if the accuracy of some classifiers performance is very weak compared to the others, they can disrupt the quality of the final results. This highlight that choice and the number of the methods to combine is crucial for maximum efficiency.

Yuan and al. [21] presented a deep learning approach to detect insiders. They use a combination of Long Short Term Memory (LSTM) neural network to learn the language of user behavior, and a Convolutional Neural Network (CNN) to identify the abnormal behavior in multiple scenarios. They obtained an area under the curve(AUC) of 94 % for their classifier. A drawback of deep learning methods is they need of consistent amount of data, and advanced material like GPUs to

have excellent performance. If one the goal is the framework portability, this might represent a bottleneck.

III. Insider Threat Problem Definition and Detection Method Review

A. Insider Threat Problem

In an organization, the IAM software has the role to manage, the identification, the authentication, and the authorization process for each employee. Most of these systems follow the role-based access control paradigm [2]. They are working by attributing roles to employees and permissions to roles. Permissions give access to the organization data and applications. In this context, an insider can be described as a user who is essentially an employee which access his company informational assets through the authentication process of an IAM software, and which is not respecting the data policy usage of his company. This policy is in place to prevent the risk of data loss or misuse, which could have a nefarious impact on companies. The problem can also be formulated more practically. Given the activity records of individuals working in a company and feature information characterizing them (e.g. organization name, role, permission, operation ...), the goal is to use anomaly detection tools to determine when these individuals present an anomalous behavior, in the way they are using their company's information assets. Behavior anomaly detection is a binary classification problem. The employee conduct is either classified as normal or abnormal.

To model normal behavior, the primary method is to start by collecting users historical activities or the patterns they follow during their daily work routines. Users with the same well-defined roles (i.e. role in organization represented in an IAM software) have mainly the same tasks. They usually repeat the same actions and trigger the same processes. A profile realized with the collection of their interactions with the system represents their typical behavior [3]. A mathematically understandable model can be established, using data analysis algorithms. With an established user baseline, all the patterns deviation from this model can be suspected to be an anomalous behavior.

Another way to detect abnormal behavior is to compare an employee activities to his colleagues(i.e. other users with the same roles). As an example, two analyst programmers in an IT company, should have access to the same assets and present sensibly the same behavior patterns. The process of detecting insider in a group of regular user can be seen as an outlier detection problem. The distance between an outlier and the model (i.e. a normal users aggregation) is an indicator of the level of menace that the user represents for the company.

B. Detection Methods Review

In this domain, most of the detection methods are usually in the spectrum of supervised and unsupervised learning methods. Besides classical data analysis, graph-based methods are used to fix this issue. Before proposing a detection model, it's essential to understand that insider threats have a heterogeneous nature. There are multiple ways to classify behavior, and different types of data can be used to model it. Even though labeled data is not very common in this domain [3] (i.e. researchers usually test their methods using artificial data), the supervised anomaly detection algorithm often outperforms the unsupervised learning and the graph-based algorithms in terms of low false positive rate. To counter insiders, new works were developed combining these supervised approaches with graph-based methods in an ensemble learning fashion.

They showcased promising results to fix this cybersecurity issue. According to Gartner [13], existing insider threat detection tools already rely on different models to detect anomalous behavior. For each case, the input dataset can be different; hence, the most efficient algorithm depend sometimes on the input data. Most of the software already use a combination of multiple techniques to detect an insider(e.g. combination of statistical analysis, expert-written rules, machine learning).

A classical ensemble learning methods can be presented as multiple classifiers working in parallel to analyze the same user behavior. A voting system is established at the output level of each classifier to elect the class of the user depending on the biggest number of occurrence. An example of classifier combination can be the use of graph features as the input data of another classifier. These features can be put into a dataset format and can be used to feed a standard machine learning algorithm to detect anomalies (e.g. the use of IForest [8]), OCSVM, or another type of classifier, which is known to be efficient for outliers detection). These algorithms would detect the anomalous parts of the graph, which are representing users and their actions.

IV. Framework Modeling

A. Framework Abstraction

The proposed framework will follow a model combining prepossessing, processing and postprocessing steps. The prepossessing part has a role in preparing data entries. They will be mainly a feed composed by users and their associated activities. They are mostly, IAM datasets transformed in a format usable by analysis technique (e.g. dataframes). The processing part will be centered around the notions of the modeling a typical behavior profile for each user and the classification of their behavior. The establishment of this profile would depend on their past actions, or their peers recorded activities. An insider menace will be detected if an employee showcase a significant difference between his present behavior and his profiled behavior, or the activity patterns of his colleagues. In an extreme case where the behavior profile was built from data where the individual behaved systematically in a malicious way, a comparison with his peers should mark him as suspicious. At the postprocessing level, we will rank users, depending on the danger their supposedly represent, using a computed threat score for each of them. We essentially see this approach as multiple boxes of methods able to individually give indications on the nature of the conduct of a studied user. Their results are combined to have a stronger opinion on their behavior.

B. Ensemble Learning

To model a framework for behavior anomaly detection, we propose an ensemble learning approach. As we will see, this approach presents a significant potential for parallel computing and consequently for its efficiency in terms of time to solution. This model of learning is based on the combination of weak classifiers to get better classifications results for a given problem. It needs more computation power since classifiers are mostly working in parallel, but helps to avoid issues such as overfitting, and the bias-variance trade-off.

Example based on Kumar et al. book [15]: Assuming we have an ensemble learning algorithm with *n* classifiers and the error rates of each is $\epsilon = 0.35$. Let's calculate the probability to have *m* classifiers with the wrong result. Then for n = 25 and m = 13 the error is approximately equal to 0.06 which smaller than ϵ.

$$\sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} \approx 0.06 << \epsilon \quad (1)$$

This example proves that it is less likely to have wrong classification results with an ensemble learning framework, than a single classifier.

At the data level, there are multiple ways to apply ensemble learning. The main methods are *Bagging* (i.e. split the data set into samples and train each classifier with a different sample) and *Boosting* (i.e. based on an iterative systems which introduce at each iteration wrongly classified element in the new training data to get better models). A combination of these two approaches can be considered in order to get better results.

Starting from the ensemble learning idea, we will primarily feed user activity data through a preprocessing pipeline to multiple behavior classifiers and get their results for each user. These classifiers would belong to the families of methods mentioned before to detect the insiders (i.e. supervised learning, unsupervised learning, graph-based methods)(see Fig. 1).

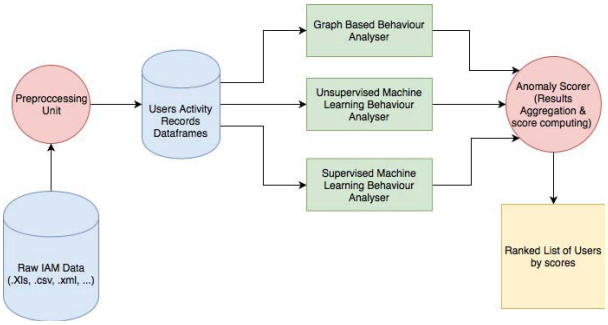

Fig. 1 Proposed Ensemble Learning Framework

The obtained result of each classifier will contribute to the risk scoring of every employee monitored. Each classifier result would be given a weight, which will represent its impact on the computed risk score. These weights will be chosen in relation to the studied accuracy (i.e. recall, precision) of the classifiers.

C. Supervised Learning

The models used in supervised learning characterize a typical behavior of user by affecting classes to each users depending on their data entries. They create a user profile based on their activity records and try to determine if a new record corresponds to a specific pattern in that profile. For this part we propose to study the classification result of multiple supervised learning algorithms to choose methods to use for our framework (see Section VI). The use of models such as OCSVM, IForest, ensemble tree-based algorithms (random forest or gradient boosting[19]), artificial neural networks[20], [21] recently showed promising results.

D. Unsupervised Learning

Using an unsupervised learning approach, modeling user behavior is mostly done at the same time that the detection of bad behavior. These methods are based on clustering and outliers detection. The goal is to group users who showcase the same practices as one entity (i.e. clusters). The detection process will be based on the assumption that users who don't belong to the main employee groups are insiders. We will essentially use two types of clustering methods (i.e. hierarchical and flat clustering). Hierarchical clustering would be by the use of the *Mean shift* algorithm. This method can identify the number of clusters on a given dataset. Flat clustering would be by the use of K-Means. In our problem, the supposed maximum number of groups is two. Either the users' activities is normal or malicious. Using *K-means* after *Mean shift* would allow double-checking the clustering results, and will showcase uniform values or aberration on the dataset that will indicate suspicious activities. As presented in the related work [7], clustering methods can also be used as a class decomposition tools in our framework, to detect anomalies at a more local level. This idea is interesting to refine models. We could also think about apply clustering to a graphical representation of users.

E. Graph-Based Methods

In some other IT domains such as social networks, telecommunication, sale sites, search engines, and recommendation systems, fraudulent user activities is an issue that cybersecurity experts are continuously facing. Similar in some aspects to the insider threat problem, this issue leverage the use of behavior analysis. In these fields, the term user behavior modeling refers to the study of users behaviors, in order to classify them as good or fraudulent. These actions are behavior anomalies same as insider activities in an organization. With some considerations, the fraudsters can be seen as the equivalent of to the insiders of these domains. The methods proposed to deal with this issue uses graph representations, analysis techniques, and linear algebra. Their functional entities and their connections are represented under components such as nodes and edges. Graph features represent a good information sources to detect anomalies.

To build a framework using graph-based methods, and still following the principle of ensemble learning, we can combine

three types of graph-based methods to work as a much stronger classifier (i.e. *Subgraph Analysis*, *Propagation Methods* and *Latent factor models*). The goal is mostly to identify the nature of the nodes and edges in the graphs. Normal nodes and edges represent normal behavior, and the anomalous ones, abnormal behavior.

Strictly following a role-based access control type of architecture, a graph model, would be close to the representation proposed in Fig. 2. In this figure, organizations are represented in green, users in blue cyan, roles in light blue, permissions in magenta, applications in purple, and operations in grey.

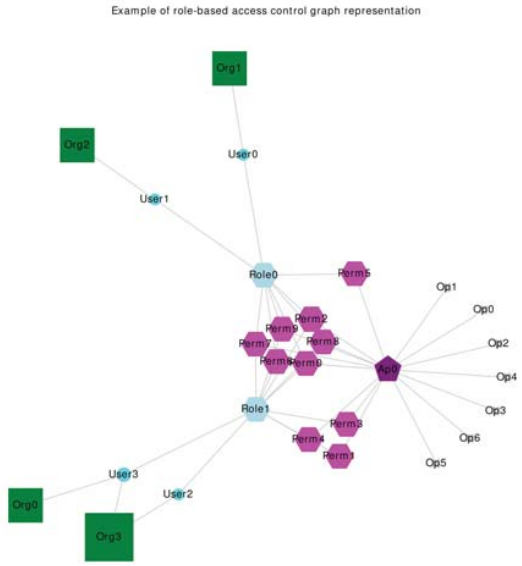

Fig. 2 Example of graph modelization of a IAM system

An induced subgraph is a subset of a graph. It is defined as a set of selected nodes or a specific region of a graph. The goal of their study is to highlight local graph properties in order to detect if they present a divergence, compared to the global graph. This is an indicator of behavior anomaly at the local levels.

Using an ego-network (egonet) solution, the central nodes can represent a user, and his neighbor nodes can be the features associated with his connection or access to his company resources. Starting with the computation of some subgraphs features, we can represent them into two dimension axis and build regression lines to characterize the model and his boundaries. This is inspired by the work of [23] and [8]. Akoglu et al. studied features of egonets to detect anomalies in a graph. They developed a scalable and unsupervised learning method for behavior anomaly detection. The first step of their work was to compute features that characterized the central nodes of an egonet and his neighborhood. Those features can be the number of neighbors, the number of edges, the total weight of the egonets, and the principal eigenvalues (i.e. the biggest) of his adjacency matrix. The second step was to combine these features to find normal behavior patterns characterized by these subgraphs. They established a list of laws and observations followed by the majority of the egonets, and use the principles of outliers detection to find anomalous egonets. This method would represent an outliers detection tool which could identify users at the boundaries of the models as an insider menace.

Propagation methods are a family of techniques that are useful to spot anomalies (i.e. fraud attempts) on web pages distribution (e.g. spams), auction sites, review site, and social networks. The most known propagation methods are the hyperlink-induced topic search (HITS) [17] and the PageRank(PR)[18] algorithms. Using these propagation methods, a directed graph can be created highlighting the connection or their transitions from a resource to another. The belief propagation [24] principle can also be considered as a propagation method. Using this principle, we can start with the unsupervised learning methods presented before, and try to identify groups of insider based on a neighborhood study. This mechanism of detection is labeled as a guilt-by-association method. For this process, in the case of IAM, all the nodes in the graph can be connected by their belonging to the same entity(e.g. organization department).

Latent factor models are the third branch of graph-based methods for user behavior modeling. They are mostly based on the use of singular value decomposition (SVD). This decomposition is used in the domain of recommender systems as an alternative to the collaborative filtering method based on user correlations. The graph models related to these methods become more in the form of a bipartite graph. A recommender system is a tool able to recommend items (e.g. applications, sites, movies) to an individual. For a recommender system, a bipartite graph structure is used to highlight the common topic of interest between individuals. In the domain of recommender systems, the most common method is collaborative filtering [16]. It determines future preferences in a set of items for a user, based on the similarity of the user to another (i.e. user-based), or the similarities between items (i.e. Item-based). This method represents another tool to detect malicious users. In the process of filtering, starting with a utility matrix (i.e. relation matrix between users and items) *Pearson correlation*(2) and *cosine similarity*(3) can be used to determine the similarity between users.

$$U_{i,j} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}} \tag{2}$$

$$cos(u_i, u_j) = \frac{\sum_{k=1}^{m} (v_{ik} v_{jk})}{\sqrt{\sum_{k=1}^{m} (v_{ik}^2) \sum_{k=1}^{m} (v_{jk}^2)}} \tag{3}$$

In these equations $u_{i,k}$ denotes the similarities between the users i and k. $v_{i,j}$ represent the rating that user i gave to item

j. To predict user opinion on an unrated item, we apply the following formula:

$$V_{ij}^{*} = K \sum_{v_{kj} \neq ?} U_{jk} V_{kj} \tag{4}$$

The implementation of this method is straightforward but has the issue to not take into account the change in user preference over time (i.e. concept drift, which is a common problem in outliers detection tools). The same process can be used to calculate the similarity between items, hence recommend a list of items to the users. The problem here is the scalability and the sparsity of the problem. The computation needs grow proportionaly to of the number of user and items. The sparsity taints the similarity results between items (i.e. when there are far more items than users, and the majority of them are not rated).

SVD is used to avoid the $cold-start$ problems linked to the sparsity of the utility matrix. This lack of rating gives result inaccurate calculated similarity between users or items. This decomposition helps to reduce the dimension of the problem, hence contributing to characterize easily users, based on the latent factor. It transforms the recommendation systems into an optimization problem using the properties of the singular vectors and the sum of square error (SSE).

We are then considering M_{ij} as the utility matrix with missing values. We want to use the singular value decomposition on M, to minimize the SSE to get the best approximation of the rating.

$$min(U, V) \sum_{(i,j) \in M} (M_{i,j} - U_i \Sigma V_i)^2 \tag{5}$$

$$\widehat{M_{i,j}} = U_i . \Sigma . V_i \tag{6}$$

Based on the [9] proposal, the computation of the similarities between user could indicate if a user is behaving normally or abnormally through a comparison with a list of users. The usual actions are highly rated in comparison to the abnormal ones, hence giving insights on insider detection. For this method, we would propose to build a detection system based on the efficient SVD computation, as one of the pieces of the ensemble learning framework.

F. Risk Scoring

After the application of multiple classifiers, there is a need to establish a risk scoring algorithm to identify a user as an insider effectively. Employees will be ranked in the function they score. Using the main principles of ensemble learning, looking at the classification average result could be applied to spot a threat effectively. However, we can use more advanced techniques such as a weighted linear regression method, based on the studied accuracy of the individual classifiers. Those weights would represent the influence of a specific classifier on a user score. Another solution would be to apply another classifier taking as input the results of the behavior classifiers to decide if a user is an insider and to generate the indicator. Some Algorithm like the IForest propose a metrics to quantify the anomalies scores [22]. It also could be used as a base. Additionally, to classify employees behavior visually, we can use a color code. Green would represent a clean user, orange a suspicious user, and red a confirmed malicious user. Finally, thresholds would be chosen to pick specific responses depending on the threat level. These responses would be considered as contextual grey decision(i.e. a black and white systems would demonstrate more binary responses such as allowing or denying user access to the network).

V. Framework Implementation Ideas

The implementation of this framework will leverage the use of *python* code and *C* code. *Python* can be chosen for his common use in the machine learning domain. it's mainly through libraries such as *pandas*, *scikit-learn*, *tensorflow*, and especially for graph modeling, the *networkx* library. A combination of these libraries will allow creating a pipeline able to preprocess data and apply numerous classifiers. A pipeline using *pandas* functions, will be able to support multiple data entry format of security datasets. The raw datasets would much likely be under databases format extraction (i.e. .xls, .xml or .csv files). The preprocessed data would be then be fed to the classifiers, in a distributed and parallel manner.

The *C* code will allow us to build memory optimized custom implementation of our classifiers and also the eigenvalues computation methods used on our graph-based classifiers. The eigenvalues computation would use *Arnoldi* based methods (e.g. Implicitly Restarted Arnoldi Methods(IRAM)). The use of the *cython* library will serve as an interface between the *c* and the python code. Custom implementation or The *scikit-learn*, and *tensorflow* libraries can be used to apply classifiers to activities data. These libraries are widely used on the machine learning communities, but a custom implementation is more suitable to build optimized software.

After a study of most demanding computation aspects of the framework, high-performance computing methods (i.e. using the *pyMPI* library) can be used to do distributed calculations on the ensemble learning model and the eigenvalues problems. In the case of the use of neural network methods, an efficient implementation might require a GPU distributed implementation to get the best performance.

In the context of the learning method, it would represent the usage of a different process for each classifier, data duplication, and result gathering at the end. An MPI(i.e. Message Passing Interface) architecture would be perfectly suitable for this kind of software architecture. Parallel implementations of the eigenvalue problem are standard in the high-performance computing domain and proven useful to provide better computation performance. In our case, the user activity data can be significant depending on the size of the studied companies(i.e. big enterprises tend to have many employees, so a lot of entries in datasets).

The performance study would also assess the feasibility of a combination of a *MPI* and a *OpenMP*(Open Multi-Processing) implementation. This hybrid implementation would manage at the same time the eigenvalues, the forests algorithm, and the

ensemble learning structure. Running on a cluster architecture, the node would be in charge of a single classifier, and the threads would manage the distributed part of the base algorithms.

The parallelism present in this framework is the large coarse grain with asynchronous communications between the large-size components. This kind of parallelism is well adapted to current petascale and future exascale supercomputers.

VI. Test Results

In this section, we present the results of some of the previously explored classification methods used on two different datasets. This part has the goal to give insights on the accuracy of the classifiers, before choosing to use them in our future framework. We used an IAM software company data to classify users, and their roles related to their activities to test some supervised learning algorithm. Classifying users, using their actions is the base of anomaly detection in IAM context. In order to test unsupervised learning methods, we applied the *Mean shift* algorithm using *scikit-learn*(see Fig. 3). The dataset used is an organization policy dataset of 51 entries containing: *User Id*, *Organization code*, *Organization inheritance*,*Role and Role ID*, and *Permission and Permission ID*.

The policy represents the list of roles and permission of the users. We applied a principal component analysis (PCA) to represent the data in a 2D format.

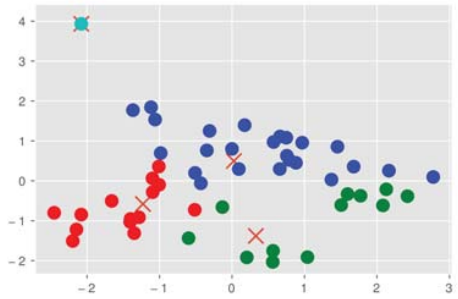

Fig. 3 Mean shift on Policy dataset

In Fig. 3, we considered the dataset as healthy. Therefore we expected to have only one cluster detected (i.e. when regrouping the users) by the unsupervised learning approach. This was not the case; four groups were found. This was due to the fact the users in the data can be cluster into a subgroup. However, one clear outsider was found at the edge of the domain. This user has not the same organization inheritance as the rest of the users, which are all sensibly in the same area.

We applied on the same dataset the isolation forest algorithm (see Fig. 4), we found a list of outliers, which also contain the same individuals at the edge of the domain. This allowed us to confirm in a way the result of the clustering algorithm, and showcase one of the utilities of an ensemble learning approach.

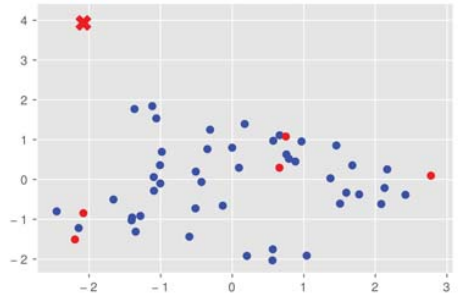

Fig. 4 Isolation forest on Policy dataset

To test supervised learning methods, we used a user activities dataset of 1000 entries, 26 unique users, representing respectively: *User Name*, *Role*, *Permission*, *Resources*, and *Operation*. We used multiple classification methods to classify the users (i.e. Logistic Regression, Decisions trees, K-Nearest Neighbors, Linear Discriminant Analysis, Gaussian Naive Bayes, Support Vector Machine, Random Forest, Bagging, and Gradient boosting classifier using decision trees as base classifiers). The main goal was to determine the users' classification accuracy using this dataset. At the view of the

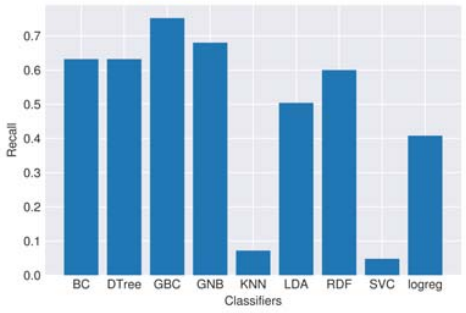

Fig. 5 User classification recall

result in Fig. 5, we found out that this dataset allows to classify users up to 75% of recall, which is pretty accurate. However, the classification algorithms tend to perform worse to identify users for all the methods if we choose a more significant data sample(e.g. 11380 entries, 301 unique users, 42% using Gradient boosting). If we look at the problem in an access control context, a smart solution would be trying to classify users with the same role. So we would mostly deal with small data samples, and avoid the poor accuracy problem. However, we can do further studies to identify the source of these poor results correctly. As a solution, we are thinking to improve the hyperparameters of each algorithm using *bayesian optimization*. Whatever the size of the dataset, the gradient boosting ensemble-based method tends to perform better. We might get even better results if we change his base classifiers.

For a second series of test we worked on a subset of user activity dataset of 364 entries, composed by 251 entries for *User1*, 93 entries for *User2*, 10 entries for *User3* and 10 entries

for *User4*. Our goal was to simulate a scenario where *User2*, *User3* and *User4* entries would be idealy detected as outliers of this subset.

At first, we applied to our dataset four outliers detection methods, which are an OCSVM, an IForest algorithm, a robust covariance method (i.e. elliptic envelope (EE)), and the local outlier factor(LOF). We then use a voting classifier combining the base-classifiers with the best recall, hoping we get a better result. We finally test 5 binary classifiers method: An artificial neural network(ANN) with two hidden layers, a Gaussian naive Bayes(Gnb) algorithm, and ensemble learning Bagging classifiers (Bgc), random forest (Rdf) and gradient boosting (Gbc). We present the results in Table I (see Fig. 6 for the ROC curves).

TABLE I
CLASSIFIERS' ACCURACIES

Classifers(%)	Precision	Recall	F1-score	AUC-score
IForest	74	74	68	59
OCSVM	64	48	49	55
LOF	73	74	69	60
EE	54	62	57	47
ANN	94	95	94	93
ODVtg	60	64	61	51
Gnb	72	71	63	54
Bgc	99	99	99	99
Rdf	99	99	99	99
Gbc	99	99	99	99

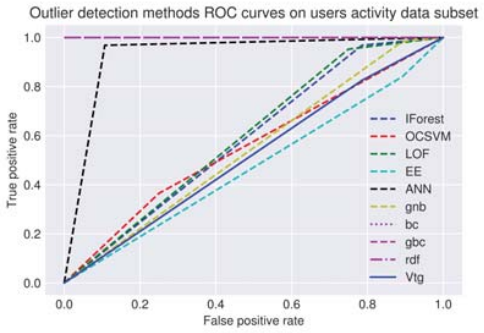

Fig. 6 Receiver Operating Characteristic curve for outliers detection methods

It is important to note that changing the studied subset(i.e. choosing different users in the subset) as an impact on the accuracy of the detection methods, but overall the ensemble-based methods always outperform the other algorithms, closely followed by the artificial neural network (i.e. in our tests the ensemble base method varied from 97% to 99% and the ANN accuracy varied from 60% to 99%). It might be interesting to use an ANN combine with bagging or boosting methods to check if we can obtain even more better performance.

Our next steps is to convert the dataset at our disposal into a graph or a data frame format, to matrix format (e.g. adjacency, utility matrix). This will allow us the test the graph based methods we studied, and assess their detection accuracies for the insider threat problems. It will also be necessary to consider more algorithm taking into account the behavior evolution over time, to build more diverse framework detection capabilities.

VII. Conclusion

Due to the evolution of the threat landscape and continuous hacker innovation, behavior analysis has become an essential tool. Based on the monitoring of user, systems, and entity activities, its goal is to detect intruders, insiders, or fraudsters in multiple application domains. These threats can pass through classical cybersecurity tools such as IAM, SIEM software, and fraud detection programs. This is mainly because classic tools are based on expert-written rules and correlation studies. These tools present a high cost of maintenance and a lack of flexibility.

In this article, we explored some basics about the core themes of data protection, cybersecurity threats, and behavior analysis. Starting with the study of behavior analysis application in cybersecurity, we presented ideas to develop a framework able to detect and stop insiders. This study allowed to highlight the usefulness of the ensemble learning approach base on the combination of multiple classifiers combine with a parallel implementation. We pointed out the detection methods based on graph modeling, supervised and unsupervised learning methods, that can be used as classifiers into an ensemble learning framework implemented with high-performance computing methods. We tested unsupervised learning methods and supervised learning classifiers accuracy for this problem in an IAM context. We obtained the best results using an ensemble-based boosting technique for both test (75%-99%) followed closely by an artificial neural network for our second test (60%-99%).

To complete our the proposed framework model, as future work we are going to test other ensemble type solutions, graph-based detection algorithms we presented, other emerging deep learning solutions, and exploit the potential parallelism of the proposed ensemble solution. These are essential steps to finalize our efficient framework architecture.

References

[1] IBM-Security, IBM 2015 Cybersecurity Intelligence Index, Managed Security services, https://securityintelligence.com/media/cyber-security-intelligence-index-2015/, 2016.
[2] P. Bradford and J. Lui, Applying role based access control and genetic algorithm to insider threat detection, 44th annual Southeast regional conference, pp 1–7, 2016.
[3] J. Peng, K. R. Choo and H. Ashman, User profiling in intrusion detection: A Review, Journal of Network and Computer Applications, vol. 72, pp 14–27, 2016.
[4] A. L. Buczak and E. Guven, A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection Systems, IEEE Communications surveys and Tutorials, vol. 18, no. 2, pp 1153–1178, 2016.
[5] P. Pallabi, N. Mcdaniel and Z. R. Weger, Evolving Insider Threat Detection Stream mining Perspective, International Journal on Artificial Intelligence Tools vol. 22, no. 5, 2013.
[6] P. Pallabi, Z. R. Weger, et al., Supervised Learning for Insider Threat Detection Using Stream mining, 23rd International Conference on Tools with Artificial Intelligence, 2011.
[7] D. Haidar, and M. M. Gaber, Adaptive One-Class Ensemble-based Anomaly Detection: An Application to Insider Threats, Internationnal Joint conference on Neural Networks(IJCNN), 2018.

[8] A. Gamachchi, L. Sun, and S. Boztas, A graph based framework for malicious insider threat detection, Hawai International conference on system sciences, (HICSS), 2017.
[9] Y. Chen, S. Nyemba, W. Zhang, and B. Malin, Specializing network analysis to detect anomalous insider actions, Security Informatics, vol. 1, no. 1, pp 5, 2012.
[10] I. Sun, S. Versteeg, S. Boztas, and A. Rao, Detecting Anomalous User Behavior Using an Extended Isolation Forest Algorithm: An Enterprise Case Study, In Computer Research Repository(CoRR), 2016.
[11] P. Moriano, J. Pendleton, S. Rich, and L. Jean Camp, Stopping the Insider at the Gates: Protecting Organizational Assets through Graph Mining, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 9, pp 4–29, 2018.
[12] Ponemon, 2018 Coast of Insider Threat Global organizations, Ponemon Insitute Research report, https://www.observeit.com/ponemon-report-cost-of-insider-threats/. Last accessed 4, 2018.
[13] A. Chuvakin and A. Barros, A Comparison of UEBA Technologies and Solution, Gartner Technical Professional Advice, pp 1–45, https://www.gartner.com/doc/3645381/comparison-ueba-technologies-solutions, 2017.
[14] S. Gopalakrishnan, Data Science & Machine Learning in Cybersecurity, In: AT&T Business, vol. 3, pp 1–15, 2017.
[15] V. Kumar, P-N. Tan, M. Steinbach and A. Karpatne, Introduction to data mining 2nd edition, https://www-users.cs.umn.edu/~kumar001/dmbook/index.php, 2018.
[16] S. Hung, Introduction to collaborative filtering Part1, in hackernoon.com, hackernoon.com, 2018.
[17] J. M. Kleinberg, Authoritative Sources in a Hyperlinked Environment, Journal of the ACM", vol. 46, pp 604–632, 1999.
[18] L. Page and S. Brin, Anatomy of a Large-Scale Hypertextual Web Search Engine, Proceedings of the seventh international conference on World Wide Web(WWW) 7", vol. 46, pp 107–117, 1999.
[19] A. Ravanshad, Gradient boosting versus random forest, https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80, 2018.
[20] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols and S. Robinson, Deep learning for unsupervised insider threat detection in structured cybersecurity data streams, AAAI Conference on Artificial Intelligence, 2017.
[21] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan and B. Fang, Insider Threat Detection with Deep Neural Network. International conference on Computationnal Science (1), pp 43–54, 2018.
[22] E. Lewinson, Outlier Detection with Isolation Forest, https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e, 2018.
[23] L. Akoglu, M. McGlohon, and C. Faloutsos, Oddball, Spotting anomalies in weighted graphs, Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), vol. 46, pp 1–12, 2010.
[24] P. P. Talukar and K. Cramer, New Regularized Algorithms for Transductive Learning, Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, Part II, vol. 5782, pp 442–457, 2009.
[25] W. Eberle, and L. Holder, Insider Threats Detection Using Graph-Base approaches, Cyber security Application & technologies Conference for homeland security, vol. 5782, pp 1–5, 2009.