Improving Intrusion Detection Systems using Zero-Shot Recognition via Graph Embeddings

Saber Zerhoudi*§, Michael Granitzer*
*Department of Computer Science and Mathematics
University of Passau, Germany
Innstr. 41, 94032 Passau, Germany
{saber.zerhoudi, michael.granitzer}@uni-passau.de

Mathieu Garchery*†§
†Atos Information Technology GmbH
Otto-Hahn-Ring 6, 81739 Munich, Germany
Email: mathieu.garchery@atos.net

***Abstract*—In order to detect insider threats, anomaly-based intrusion detection must learn profiles of normal user behavior. However this is particularly difficult when historical audit data is scarce. Zero-shot learning can address this limitation by compensating the absence of examples with semantic knowledge, allowing to better estimate behavior of unknown users. In this paper, we address insider threat detection in two use cases where historical user data is unavailable or obsolete. We extend an existing intrusion detection system by adding information describing user positions, roles and projects assignments. These semantic descriptions are encoded via graph embeddings. Experimental results show that providing this additional context improves insider threat detection significantly. This suggests that zero-shot learning is a promising way of improving intrusion detection systems.**

***Index Terms*—zero-shot learning, intrusion detection system, graph embedding, security, anomaly detection**

I. Introduction

Nowadays, organizations are becoming more aware of cyber-security threats especially the insider ones. The number of insider threat breaches rises year-on-year. Recently, the Verizon 2019 data breach investigations [1] reported that 34% of all breaches in 2018 were caused by insiders. Additionally, the Accenture & Ponemon's 2019 report about the cost of cyber-crime [2] stated that the average cost of insider attacks rose 15% from 2018 to 2019.

According to [3], " An intrusion detection system (IDS) is software that automates the intrusion detection process, [i.e.] monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices ". However, it's really hard to detect insider-related incidents because they are performed by people working within an organization and trusted by having access and knowledge of multiple important assets. For this reason, some breaches can stay undetected for months or even years.

As the threats come from the inside, it's important to shift the focus toward learning user behavior in order to improve detection. In this regard, various anomaly-based intrusion detection methods have been proposed. However they are unable to address challenging scenarios in which historical data is missing or too scarce.

Therefore, zero-shot learning can be used to help providing context from user meta-data (i.e. not audit log data). Zero-shot learning (ZSL) is inspired from the way human identify new and unseen classes when a high-level description is provided. This high-level description is then used to identify new classes without any training samples by relating them to classes that we previously learned during the training. A very simple example: humans who have never seen a zebra would be able to identify one, when told that it looks like a horse with black and white stripes. In parallel, few-shot learning works the same as zero-shot except that it requires a few training samples. ZSL has two main stages: the attribute learning stage *(training phase)* in which knowledge about the attributes is captured, and then the inference stage where this knowledge is used to classify instances among a new set of classes.

Zero-shot learning has been widely applied to areas such as image classification tasks [4], especially computer vision [5], object recognition [6] and document classification [7]. The example that kicked off the whole area of zero-shot learning is fMRI mind reading [8], where ZSL was used to estimate brain activity for unseen concepts by using combinations of brain activities of known concepts. A very few approaches have been proposed to utilize zero-shot and few-shot learning in intrusion detection systems [9, 10, 11]. In this context, zero-shot learning can help refining user behavior modelisation when few or nonexistent logs are available.

In this paper we introduce two scenarios where user historical data is unavailable. The first scenario will represent the case where a new employee joins an organization, and the second one will reflect the changing of behavior that occurs when an existing employee changes the project he's working on. The two experimental scenarios used in this work are realistic and focus on a specific type of cases where we expect the system to have some difficulties to distinguish threats and normal user behavior rather because historical data is not available (new user) or obsolete (project change).

In order to characterize users and their expected behavior before any historical data is available, we use graph embedding techniques to learn user vector representations which encode

§These two authors contributed equally to the work.

978-1-7281-7303-0/20/$31.00 ©2020 IEEE
DOI 10.1109/COMPSAC48688.2020.0-165

their position and relations in the organization's structure. Graph embedding is a widely used technique in organizational network analysis to learn low-dimensional representations for nodes in graphs. In this work, we first start by extracting user context from LDAP *(Lightweight Directory Access Protocol)* attributes, which are key-value pairs used to store user data. We then use this context in a feed-forward neural network algorithm to strengthen intrusion detection abilities.

Throughout this work, we describe the extension of an unsupervised machine learning system [12] which analyzes data from user logs and detect malicious insider threat activities. The performance of this system is then tested by applying different sets of experiments. Analysis shows that our model is able to achieve better results than the baseline in both evaluation scenarios.

The contributions of this work can be summed up as follows:

- We extend an existing intrusion detection system with ZSL capabilities by integrating LDAP attributes as graph embeddings.
- We propose two realistic scenarios in which ZSL allows to better model user behavior.
- We benchmark the extended system against its baseline in both scenarios and show that ZSL capabilities improve intrusion detection performance.

II. Related works

In this section, we introduce related works in two domains; the first concerns research on intrusion detection systems coupled with zero-shot and few-shot approaches, while the second one is related to graph embedding methods.

A. Zero-shot learning/Few-shot learning and security applications

Most of the algorithms which are implemented for zero-shot learning are used in images classification tasks. They aim to model the relationships among features, attributes, and classes in order to automatically recognize features based on the attribute extraction. Such a strategy makes it difficult to apply zero-shot learning to other kind of problems like intrusion detection systems due to the lack of attribute learning algorithms for non computer vision related tasks.

However, few works coupled one-shot [9] and few-shot [10] learning with intrusion detection. Authors in [9] proposed to train a deep-Q network to obtain a policy which is ignorant of the unseen classes and then map the current state of the one-shot learning process to operational actions in real-time to maximize the objective function. In [10], authors have trained a deep convolutional neural network structure and used it as a general feature extractor. The introduced methods have proven to be effective when the data is highly imbalanced in terms of overall and class-wise test accuracy, which makes it a good candidate for imbalance learning an intrusion detection. Nonetheless, it detects the minor class more accurately at the expense of compromising test accuracy performances of majority classes.

Authors in [13] developed a system which creates role-based profiles that can be used to describe all activities that are performed by users within an organization and compares daily observations to find anomalies so that whenever it detects an anomaly, an alarm is raised. However, the system has several limitations when initializing role profiles since it has little data to compare with, which results alarms being frequently raised. Plus, if a malicious action occurs within the initialization phase, it is most likely to be ignored or missed by the experts.

Likewise, [12] introduces user-based profiling system to detect anomalies in online fashion. In this work, we extend their system with semantic user attributes learned using graph embeddings.

In conclusion, only few works have applied zero or few-shot learning to intrusion detection, and no commonly accepted approach has emerged yet.

B. Graph embeddings

Graph Embedding techniques transform nodes within a graph into low-dimensional dense vectors, so that nodes which are close in the original graph (different methods have different definitions) are also close in the low-dimensional expression space. Authors in [14] offer an extended survey of the literature in graph embedding where they compared embedding techniques in term of their advantages and disadvantages. In this work, we utilize DeepWalk [15], LINE [16], Node2vec [17], SDNE [18] and Struc2Vec [19] to learn the embedding vector of each node.

III. Methods

In this section, we first describe our baseline system (lower box in Fig. 1) then its extension based on zero-shot learning (upper box in Fig. 1).

A. Baseline

The starting point for our new approach is the framework used by *Tuor et al.* (*SafeKit*) [12], which models users' normal behavior based on their logs and uses anomaly as an indicator of potential malicious behavior.

Tuor et al.'s framework utilizes raw events that are extracted from user logs and aggregates their counts using a feature extraction system in order to construct user-day vectors.

Each user ends up having multiple feature vectors (one vector for each day) that are fed into an auto-encoder to predict the next user-day vector. The anomaly is then computed based on the difference between predicted and observed user activity.

B. Zero-shot learning extension

1) Overview: The overall overview shown in Fig. 1 can be divided into two main components: graph construction, graph embeddings and their integration in the pipeline.

Basically, our contribution consists in utilizing graph embedding techniques to extract embeddings that represent similarities based on user's relations in the organization *(Sec.*

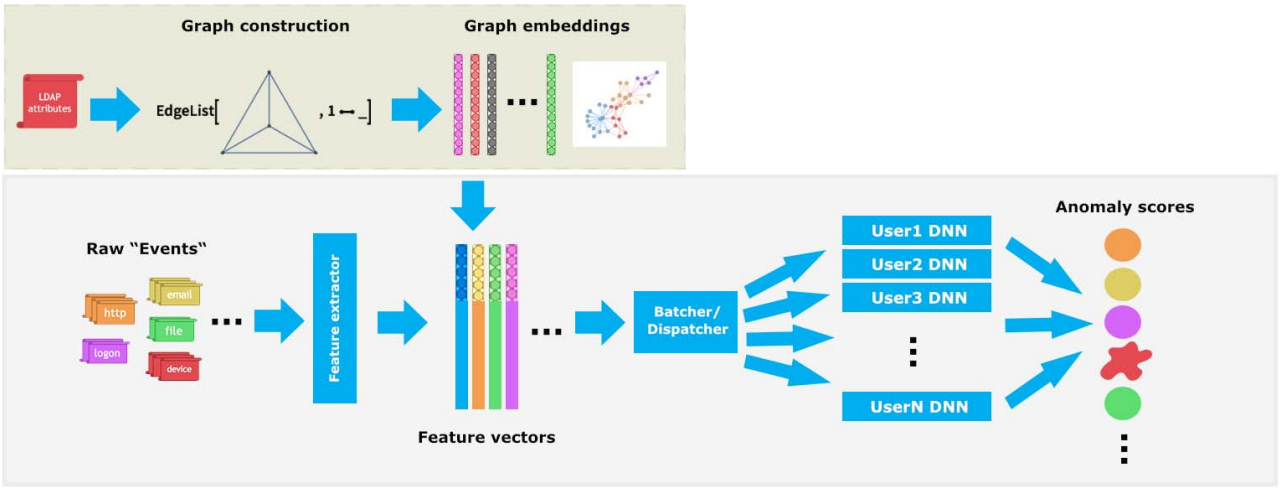

Fig. 1. Overview schema of the implemented pipeline.

III-B2, III-B3) and then feed them to the existing framework that outputs anomaly scores.

Our approach includes a number of additional and augmented steps compared to *Tuor et al.*'s. As shown in Fig. 1, the process is as follows: we extract first raw events from user logs, and then we feed them to a feature extraction system that aggregates their counts and outputs one vector per user per day. This first step is identical to *Tuor et al.*'s system. Each user's feature vectors are then aggregated with their matching user embedding before they are fed into a neural network to create a set of networks, one per user. These neural networks learn to model users' normal behavior and are tasked to predict the next vector in the sequence. The anomaly score refers to the probability of the observed value (same as in *Tuor et al.*'s system).

When the model is fully trained it should do a good job at reconstructing commonly found patterns in the data and a poor job at reconstructing unusual events. Anomaly scores are computed based on reconstruction errors as in *Tuor et al.*'s system.

2) Graph Construction: Graphs can be constructed in different ways. During this work we focus on building graphs using nodes and edges directly. Nodes represent employees and LDAP attributes combinations (role and project, role and department, etc.), while edges serve connecting those entities. We choose to work with edges to obtain one graph where embeddings are in the same vector space for all employees.

3) Graph Embeddings: The embedding maps each node to a low dimensional feature vector and tries to preserve the connection strengths between nodes. Once we get the nodes and edges from the graph built before (as described in III-B2), we test different graph embedding methods (deepWalk, node2vec, LINE, SDNE and struc2vec) and analyze their visual representation in order to choose which on preserve most of the connection between nodes.

Once we have extracted our user embeddings, we set them aside ready to be aggregated with the users' feature vectors. The system that we are willing to improve, *Tuor et al.* [12], receives as an input raw events from system user logs. Individual log lines are then transformed into a vector of categorical and continuous variables. We aggregate statistics for the log lines for each user over a time window along with the extracted user embeddings and then insert them into a vector for later use as an input to the neural network.

The networks are trained using mini-batches of samples and are fed into the (feed-forward) neural network with L hidden layers. Weights are updated with gradients computed by standard back-propagation. For a sequence length T, the model is trained to reproduce the input (series of T feature vectors x concatenated to user embeddings vector y, $x_1^u \frown y^u, x_2^u \frown y^u, ..., x_T^u \frown y^u$ for a user u) as output (series of T hidden state vectors h, $h_1^u, h_2^u, ..., h_T^u$).

IV. Evaluation

The focus of our experiments is to determine whether zero-shot learning improves intrusion detection performance when user historical data is unavailable. We first introduce two scenarios where this is the case (IV-A). We then present our evaluation setting (IV-B) and finally discuss our experimental results for both scenarios (IV-C).

A. Scenarios

We here detail the two zero-shot learning scenarios used in our evaluation.

1) First Scenario: Adding a new user: This scenario reflects the case where a new user joins the organization. When a "new" user is inserted into the system, it has no activity trace. Therefore, we rely on user embeddings which we have extracted from the LDAP attributes graph to reflect the

expected activity. However, the dataset we are working with starts with a fixed number of *4000 users*. This number drops each month until it reaches *3640 users*. To address this issue, we remove users from the train set and introduce them as new users to the test set. We first experiment by removing users randomly, then we make sure that the extracted users includes all the ones with malicious events.

2) Second Scenario: Changing the current project: The second scenario reflects the case where users are changing their assigned projects. New projects are added to the dataset, then users are affected to the newly created projects. We keep the dataset as it is and we utilize the user embeddings only if an existing user is assigned to a new project. In this case, we expect a change of behavior to resemble more to users working on the same project.

In this scenario, the dataset starts with a list of *100 projects* which only a portion of users are affected to. The remaining users are on a "stand-by" status, which means they are waiting to be affected to a project. The list of projects increases until it reaches *366*. Throughout the course of time we also notice that some projects cease to exist marking the end of a project cycle and leaving place of a new one to appear.

B. Setting

1) Dataset: To evaluate the proposed approach, we use an intrusion detection system benchmark dataset, namely CERT (v6.2) [20]. The dataset contains multiple files that represent a collection of synthetic insider threat test information. This synthetic data is generated by modeling user activities on a network and injecting expert designed insider threat events into the normal activities. There are 516 days of data and 135 million log lines in total. The events are from five sources: email, web, logon, file and device usage logs.

The split into train and test set is done chronologically and similarly to the work of [12], which results in two subsets: train (85% of the data - from day 1 to 418) and test (15% of the data - from day 419 to 516) excluding the weekends.

2) Graph Construction: The dataset is accompanied by user meta data (LDAP attributes: user role, project, department, etc.) and mainly composed of information that define users relationship with different entities. For instance, their role in the organization, the project they are assigned to and the team they are working with.

When constructing our user graph, nodes represent user's meta data and edges translate the relationship between these meta data. This will leaves us with a variety of ways to construct our graph. However, when applied to our case, LDAP attributes are not equally important to build up user's context. Therefore, we set employee, project and role as important nodes. The rest of LDAP attributes, such as supervisor, team, email, department and functional units will represent the less important nodes. We test different combinations of less important nodes to determine which one correlate the most with the important ones. For simplicity purposes we visualized the graph with the help of t-SNE to reduce the dimensionality, we then inspected the similarity between different nodes empirically and with the support of the structural identity which determine nodes that are similar based on the network structure as a measure for evaluating embeddings.

We notice that using employee, project, role and team connections gives a better outcome mainly because employees relations are defined in an accurate manner. In contrast, when replacing teams with bigger entities (departments, functional units), embeddings tend to lose user identity.

Therefore, when constructing the graph, nodes will represent employees, supervisors, teams and projects. For each project, we create a list of edges where, for each team, each employee is connected: to the project he's working on, to his team, to his role, to the combination of role and project, to his supervisor, to his coworkers, to his coworkers' role and project, to the combination of his coworkers' role and team, to the combination of his coworkers' role and project as illustrated in Fig. 2.

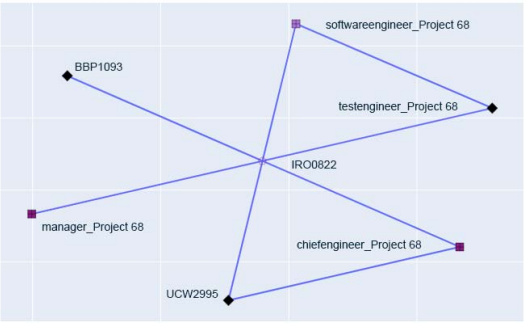

Fig. 2. Simplified tree structure of the constructed graph around the user 'IRO0822' working on the project '68'. Nodes represent employees and role-project combinations and edges serve connecting those entities. In this example, IRO0822 is working as a test engineer on project 68 under the supervision of his chief engineer UCW2995. IRO0822's team contains people working as software engineers (e.g. BBP1093), test engineers and manager.

3) Graph Embeddings: We tested different variants of graph embedding techniques (deepWalk, node2vec, LINE, SDNE and struc2vec) where nodes are connected by adding an edge connection based on the relations that we determined previously. Using the resulting 2-dimensional space of t-SNE visualization and the structural identity, we empirically determined that the best results were obtained with Struc2vec. Furthermore, we notice that struc2vec outperforms other graph embedding techniques when learning the latent distributed vectors that capture the structural similarity based on graph structure, regardless of node and edge labels.

4) Baseline hyperparameters: In order for our results to be comparable to the baseline model presented in [12], we need to make sure that we extract the same features type and

count, work with the same fixed time window (24 hours) and tune our models on the training phase using the same hyper-parameters. Therefore, we tune the number of hidden layers (between 1 and 6) and the hidden layer dimension (between 20 and 500) as in [12]. We also fix the batch size to 256 samples (user-days) and the learning rate to 0.01. We reproduced *Tuor et al.*'s model which we consider as our baseline and we re-implemented the pre-processing part described in [12] where some features such as *(common/uncommon)* are not well defined, hence not exactly reproducible. We also use `tahn` for the hidden activation function and we train using the ADAM variant of gradient descent. Similarly to the experiments done in [12], we produce anomaly scores that are used to rank user-days from most anomalous to least.

Unlike [12], we made the choice of keeping the categorical variables and adding user embeddings in inputs and outputs, hence we need to additionally tune a hyper-parameter which determines the size of the input embedding vector of a category in relation to how many classes in that category (between 0.25 and 1).

5) Metrics: To evaluate our model, we adapt a recall-oriented metric introduced by *Tuor et al.* in their work [12]: cumulative recall at a daily budget k (CR-k). Working with recall-oriented metrics is more suitable for rating the performance of a model compared to precision-oriented ones. In fact, precision represents the fraction of all detected anomalies that are real, while recall is the fraction of all real anomalies that are successfully detected.

A daily budget k represents the number of most anomalous user's logs which we can afford to investigate each day. Hence, recall at a daily budget k is the number of real anomalies which are successfully detected while checking a k number of user-day pairs. We examine the Cumulative Recall (CR-k) which is a way of averaging over all budgets (i.e. over different decision thresholds) by summing up the recalls for all daily budgets up to and including k and then we normalize all the values to obtain results in range 0 to 1.

C. Results and Discussion

The focus of the experiments is to compare our system with the baseline in different scenarios.

1) Adding new users:

a) Results: We conduct the first experiment where we adapt the dataset to our first scenario. We delete a randomly selected portion of users with their logs before training our model. Once it's done, we reintroduce them during the test phase joined with their user embeddings that we have extracted separately from only the LDAP attributes. We iterate this process varying each time the number of "new" (present only in the test set) users (200, 500, 1000).

In table I we report results for different number of random new users (200, 500 and 1000) to determine its effect on CR-k score. In table II we report results for 200 new users including all malicious insiders. We choose here two arbitrary k values

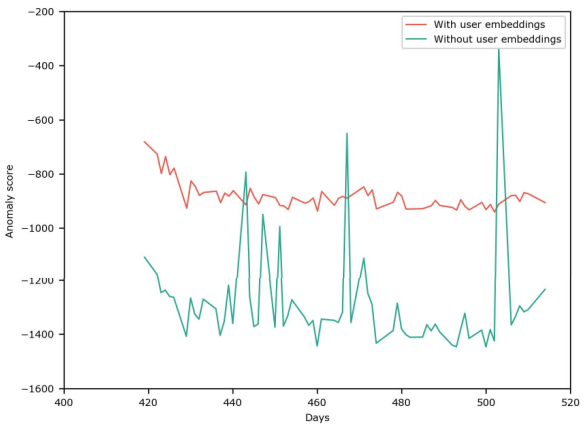

Fig. 3. An in-depth look of the evolution of anomaly score in the course of days for a picked normal user (not malicious). The model starts with no prior logs about the user (completely untrained), it is translated by a very high anomaly score with multiple spikes (more anomalous) in the first few days which will probably lead to false alerts. As the model sees examples of user behavior, it quickly learns what is "normal" which pushes the anomaly score down (less anomalous). Furthermore, when using embeddings the curves fluctuate less compared to when we exclude them. This is due to the fact that the model has more context about a user's behavior and can quickly recognize his normal behavior.

1500 and 3500 to compare the normalized cumulative recall. In both tables, results show that using embeddings improves significantly the performance compared to excluding them.

Model	CR-1500	CR-3500
Without Embeddings (Baseline)	0.0	0.123
With Embeddings (Ours)		
200 "new" users	0.343	0.457
500 "new" users	0.226	0.270
1000 "new" users	0.136	0.263

TABLE I. Normalized Cumulative Recall (CR-k) for budgets of 1500 and 3500. Comparing the performance of the model with and without embeddings (adding 200, 500 and 1000 random new users respectively).

Model	CR-1500	CR-3500
Without Embeddings (Baseline)	0.109	0.126
With Embeddings (Ours)	0.471	0.630

TABLE II. Normalized Cumulative Recall (CR-k) for budgets of 1500 and 3500. Comparing the performance of the model with and without embeddings (adding 200 new users including all users with malicious events).

Figure 3 plots the evolution of anomaly score for a picked user during the test period. We notice that in the absence of user embeddings (green curve), the fluctuation of the anomaly score is more important especially in the first days since the model starts with no prior logs about the user. As the model sees examples of user's behavior, it learns his "normal" behavior which affects the anomaly score to fluctuate less. But when user embeddings are introduced (red curve), the fluctuation becomes less important allowing the upcoming

user-day vectors not to be flagged as malicious events. Thus, false alerts are avoided.

b) Discussion: The cumulative recalls follow the same trend independently from the number of added users. The only thing that varies is the number of daily budget required to find all the malicious events. The more unknown users are added, the more detection performance decreases. This proves that having a small base of users to learn behavior from can make it harder to predict the next user-day vector of the new users. In fact, if the number of added users increases, the daily required budget to find all malicious events also increases.

On the other hand, when we exclude all malicious events from the training set (table II), the model properly predict the next user-day vector which results higher CR-k values. We believe that in a real-world dataset or even in an improved synthetic dataset which translates better user context within an organization, including embeddings would outperform excluding them in both aspects.

2) Changing project:

a) Results: We conduct a second set of experiments to see how our model behaves when users change projects. The dataset starts with 100 available projects, where each project is related to at least one user. New projects are added as time goes by and old ones are terminated. We describe the context of newly created projects mainly using two different approaches. First using role-team connection, where we utilize a user's role and team combination to try and find a similar project from those that we have already seen in the training set and mimic his vectorial representation. Secondly, using user assignments, where we utilize other attributes (user role, team, department, functional unit, etc.) to try and find similar projects. In this approach, we start by looking for projects that share multiple attributes (an example would be two projects sharing the same functional unit, department and team). If nothing is found, we downscale and look for projects sharing smaller number of attributes until we end up using the role-team connection to determine project similarity if we found no common attributes.

In Fig. 4, we compare the variation of anomaly score for all users that changed their project (day 486) with a matching number of random users that didn't change their project during the course of the test set. When plotting the anomaly score variation for users that didn't change their project, we notice that the fluctuation is not affected much with the inclusion or exclusion of user embeddings. With the exception of the red spikes that refers to when malicious events occur. On the other hand, when project change occurs, including and excluding user embedding make a noticeable difference in the curve. In the absence of user embeddings (yellow curve), the fluctuation of the anomaly score is more important when users change their project. Moreover, shortly after the project change, anomaly scores spike which means that the model struggles in predicting the next user-day vector once a user has changed his project and flags the upcoming vectors as malicious events. But when user embeddings are introduced

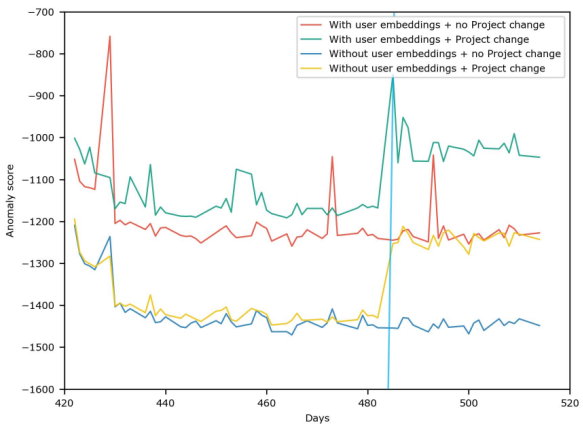

Fig. 4. The variation of anomaly score in the test set period (between day 419 and 516) comparing users that are affected by a project change and an equal portion of those who are not. The model uses the user assignments approach. The day where the project change occurs (day 486) is highlighted with a light blue vertical line. Blue and yellow represent the variation of anomaly score without adding user embeddings while green and red refer to variation of anomaly score when including them. Anomaly scores are relative, the fluctuation matter more than the absolute value

(green curve), the fluctuation has become less important after the project change has occurred and the anomaly scores fluctuate normally allowing the upcoming user-day vectors not to be flagged as malicious events. We also note that the obtained scores with embeddings are higher than without mainly because the anomaly scores are relative and the fluctuation matter more than the absolute value.

Furthermore, table III shows the normalized cumulative recall for k budget, with k being 1500 and 3500. These results show that using embeddings significantly improves the performance compared to excluding them. In addition, it gives a high CR-k score with low daily budgets.

Model	CR-1500	CR-3500
Role-team connection approach		
Without Embeddings (Baseline)	0.0	0.062
With Embeddings (Ours)	0.088	0.198
User assignments approach		
Without Embeddings (Baseline)	0.138	0.140
With Embeddings (Ours)	0.460	0.658

TABLE III. Normalized Cumulative Recall (CR-k) for budgets of 1500 and 3500. Comparing the performance of the model with and without embeddings using two different approaches (Changing project).

b) Discussion: We experiment the effect of using different approaches when defining similar projects on the model overall performance. We notice that the anomaly score curves follow the same trend independently with the used approach. Nevertheless, using user assignments as a method to define similar projects leads to better results. In addition, table III shows that the cumulative recalls are clearly better in the

user assignments approach since we reached a high recall percentage with only few daily budget. This demonstrates that setting up an approach which defines the best a project context can help improving the model performance when dealing with project changing events.

We note that the CR-k scores that we obtained in our experiments are much different from the results presented in [12] mainly due to the re-implementation of the pre-processing phase where some features (e.g. common/uncommon) are not exactly defined and to the inclusion of categorical features.

V. Conclusion

A. Key findings

We extended an intrusion detection system [12] by using zero-shot learning to improve insider threat detection performance for cases where user historical data is not available. More particularly, we used graph embeddings to encode relations from the organization structure. We introduced two realistic evaluation scenarios: the first one boils down to when new users are added to the dataset and the second one to when users change their projects.

Results illustrate that the system was able to detect malicious insider activities by exploiting the graph embeddings in scenarios where it has failed without them. For instance, in the project change scenario, the normalized cumulative recall jumped from 0.14 to 0.65 for the 3500 budgets and from 0.12 to 0.63 in the user adding scenario. Besides, anomaly scores fluctuate less with embeddings allowing anomalies to be better separated from normal behavior. Hence, graph embedding has brought a new context definition to the system enabling it to flag less irrelevant malicious events.

We conclude that zero-shot learning reveals some promising results when applied to insider threat detection, especially when including more substance to the selected features, such as a detailed LDAP file for projects, teams and departments.

B. Future work

In this work, we have limited our experiments to the feed-forward neural network algorithm from [12]. However, it would be interesting to compare the results from the recurrent neural network as well, which is expected to better capture temporal patterns.

We have been able to test several graph embedding methods and techniques but we were lacking an accurate metric to evaluate the quality of our user embeddings. A suggestion would be to test and see if a measure for evaluating user embeddings based on similarities of their behavior can be suitable for our experiment. An example can be the work of *Blandfort et al.* [21] where they presented a new metric Pair-Distance Correlation measure (PDC) to measure the quality of learned embeddings. This may improve the overall results since we tend to select models only based on their prediction ability, but we neglect the "meaningfulness" of embeddings or learned input sensitivities.

Acknowledgment

S. Zerhoudi acknowledges the financial support provided by LIS Funding Programme: e-Research Technologies. In addition, this work has been partially carried out within the Internet-Kompetenzzentrum Ostbayern (IKZ_Ostbayern) project funded by the Bavarian State.

References

[1] "Verizon: 2019 data breach investigations report," *Computer Fraud Security*, vol. 2019, no. 6, p. 4, 2019.

[2] "Accenture/ponemon institute: The cost of cybercrime," *Network Security*, vol. 2019, no. 3, p. 4, 2019.

[3] K. Scarfone and P. M. Mell, "Guide to intrusion detection and prevention systems (idps)," 2012.

[4] D. Das and C. S. G. Lee, "Zero-shot image recognition using relational matching, adaptation and calibration," 2019.

[5] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 13:1–13:37, Jan. 2019.

[6] X. Wang, Y. Ye, and A. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[7] J. Zhang, P. Lertvittayakumjorn, and Y. Guo, "Integrating semantic knowledge to tackle zero-shot text classification," *Proceedings of the 2019 Conference of the North*, 2019.

[8] M. Palatucci, D. Pomerleau, G. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, ser. NIPS'09. USA: Curran Associates Inc., 2009, pp. 1410–1418.

[9] A. Puzanov and K. Cohen, "Deep reinforcement one-shot learning for artificially intelligent classification systems," 2018.

[10] M. M. U. Chowdhury, C. Xin, J. Li, and H. Wu, "A few-shot deep learning approach for improved intrusion detection," 10 2017.

[11] J. L. R. Pérez and B. Ribeiro, "Attribute learning for network intrusion detection," in *INNS Conference on Big Data*. Springer, 2016, pp. 39–49.

[12] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," 2017.

[13] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Automated insider threat detection system using user and role-based profile assessment," *IEEE Systems Journal*, vol. 11, no. 2, pp. 503–512, June 2017.

[14] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," 2017.

[15] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, 2014.
[16] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," 2015.
[17] A. Grover and J. Leskovec, "node2vec," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016.
[18] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," 08 2016, pp. 1225–1234.
[19] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec," *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, 2017.
[20] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *2013 IEEE Security and Privacy Workshops*, May 2013, pp. 98–104.
[21] P. Blandfort, T. Karayil, F. Raue, J. Hees, and A. Dengel, "Fusion strategies for learning user embeddings with neural networks," *CoRR*, vol. abs/1901.02322, 2019.