Value Embedding Approach for Access Control

Thanh Duc Bui* and Brajendra Panda

Dept. of Electrical Engineering and Computer Science, University of Arkansas, Fayetteville, 72701, AR, USA.

*Corresponding author(s). E-mail(s): tbui@uark.edu;
Contributing authors: bpanda@uark.edu;

Abstract

Due to the swift progress in computing and information technologies, traditional access control frameworks struggle to encompass the subtle security needs of emerging applications. An alternative, the attribute-based access control (ABAC) model, offers greater adaptability in meeting the authorization demands of intricate and evolving systems. This paper presents a framework suggesting the application of advanced Natural Language Processing (NLP) techniques to generate embedding vectors for attribute requests, employing a Skip-gram architecture to capture contextual nuances. With well-defined features extracted from these requests, a robustly trained machine learning classifier authorizes each request appropriately. We envision that embedding layers could enhance and potentially become integral to traditional access control models, aiding in capturing nuanced features over time. Through experimentation on real-world datasets, we validate the feasibility of our approach, benchmarking it against the State-of-the-Art (SOTA) model. Additionally, we discuss the challenges encountered and propose avenues for future research.

Keywords: Cyber Security, Access Control, Machine Learning, Big Data, Language Model Application

1 Introduction

Access control is essential in cybersecurity as it dictates users' access to resources and has been extensively studied. While various methods exist, popular approaches include Access Control Lists (ACLs) [1], Role-Based Access Control (RBAC) [2], and Attribute-Based Access Control (ABAC) [3]. Despite numerous studies, a longstanding issue persists. That is the reliance on human intervention, typically from skilled

security administrators, to devise detailed and intricate policies catering to diverse individual needs [4]. Larger organizations are especially prone to errors and inefficiencies in this regard. To mitigate this, system engineers often grant users excessive access to resources to reduce administrative burden or restrict users from accessing resources they are entitled to, aiming for tighter security [5]. However, this challenge escalates with the introduction of additional system elements like Application Programming Interfaces (APIs), Internet of Things (IoT) devices, and cloud systems.

One way to find a proper solution for this persisting issue is to build a model that allows the computing system to capture the underlying contextual information from each user's request. Nevertheless, many attributes of users' requests do not have complete numerical properties. While they can be expressed as numbers, those values are likely used to represent categorical information rather than having magnitude. Hence, it is challenging for experts to instruct computing systems so that they can understand them fully. Fortunately, a similar problem has been occurring within Natural Language Processing (NLP). There have been several approaches in the field of NLP to teach computers to understand human language, and one of the breakthroughs in this field was the capability to represent human-language tokens in the form of numerical vectors [6]. This approach is effective with several NLP models that can generate texts that closely resemble human languages, such as GPT-4, Llama 2, and Gemini 1.5. Subsequently, token embedding has been applied in tabular data; one of them was TabTransformer, which has shown a significant impact in the field of machine learning data structure by exploring the underlying feature of categorical data [7].

Inspired by the success of NLP and its application with token-vectorizing approaches, we propose an embedded method for access requests from known data. Section 2 discusses the related works on different access control approaches, utilization of Natural Language Processing (NLP) in cybersecurity, and our contribution with an attribute-embedding approach. In Section 3, we define the parameters for access control, our proposed approach, and the candidate embedding model. Section 4 discusses real-world datasets and candidate models for attribute embedding. In the next section 5, we present the experimental result of our approach compared to other State-of-the-Art (SOTA) approaches. Finally, we discuss limitations and possible future expansion for our research in Section 6.

2 Related Work

The access control model is a well-researched topic in the field of cybersecurity; therefore, there is plenty of work on establishing access control policies/rules. We will review the two classes of access control model as below:

- **Classical Policy Mining Approach** Numerous studies have investigated policy extraction employing traditional methodologies such as Rule-based access control (RBAC), attribute-based access control (ABAC), and relationship-based access control (ReBAC). Various algorithms have been devised to capture RBAC policies effectively, employing diverse methodological approaches, including hybrid methods, combining aspects of multiple approaches. Researchers have also applied

different criteria for evaluating the quality of extracted policies [8] or imposed constraints to refine the extraction process [9]. Notably, Xu and Stoller introduced an ABAC extraction algorithm [10]. Similarly, for ABAC, several approaches have been employed to derive meaningful policies, including frameworks for optimizing multiple objectives [11], exploring table dependencies for rule extraction [12], and algorithms for extracting both positive and negative rules [13]. Bui et al. utilized Xu and Stoller's algorithm to derive access policies based on system entity relationships and devised a greedy algorithm for ReBAC policy extraction [14]. Iyer et al. proposed a ReBAC extraction approach for systems with inconsistent variables using graph-based mining by extending it with an active learning method for ReBAC policies [15]. Cotrini et al. introduced an adaptable access control policy extraction method named Unicorn, which is capable of building policies across various access control models, including RBAC, ABAC, and ReBAC [16].

- **Machine Learning for Mining Policies/Rules** Numerous scholars have investigated the utilization of machine learning (ML) algorithms in the extraction of access control policies. Frank et al. introduced a probabilistic model addressing the role-mining problem derived from the logical structure of RBAC [5]. Alternative techniques, such as classification trees [17], deep recurrent neural networks (RNN) [18], K-Nearest Neighbor (KNN) [19], and Decision Trees [20], have also been employed in ABAC policy extraction. The initial unsupervised learning-based ABAC extraction method utilized k-mode clustering to derive rules from historical operational data [21]. Naroui et al. suggested enhancing existing ABAC policies through ML-driven policy extraction [22]. Cotrini et al. proposed an ABAC policy extraction algorithm named Rhapsody, leveraging APRIORI-SD, an ML algorithm for subgroup discovery [23]. Karimi et al. introduced an automated approach for learning ABAC policies by extracting rules incorporating both positive and negative attributes and relationship filters [24]. Another recent method involves employing deep learning architectures to glean insights from metadata associated with users and resources for making appropriate decisions, as suggested by Nobi et al. [25].

2.1 Our Contribution

We focus on capturing the context of the access requesting use and the current system status to issue the correct authorization. Hence, our contributions are:

- We propose using an embedding model training from historical data of access by various users to different resources to learn the contextual meaning of each status value.
- We deploy the previously trained embedding vectors of status values to construct a proper machine-learning model to understand the underlying context of an upcoming request and issue a valid authorization decision.
- We benchmark our proposed method-trained model against other state-of-the-art methods. To obtain this goal, we generated various datasets with different user and system statuses balanced and imbalanced between the total count of decision types. We also test the models using real-world datasets provided by Amazon.

2.2 Problem Definition

Before processing our proposed method, we will define some assumptions and definitions presented throughout the paper. In our assumption, each request consists of 3 main parts: the user's current information, the system's current statuses, and authorization provided by system administrators. We will define each as a set and how they interact to make the definition proper.

Definition 1: Let S_i is a set of all possible attribute for any user u; for a given attribute $a \in S_i$, V_a is defined as a range of all possible values for a. Similarly, we also define the set S_j as a set of all possible states for any resource r; for a given state $s \in S_i$, V_s is defined as a range of all possible values for s.

Definition 2: We define a request q as $q\langle u, r\rangle$ as request made by a given user u, each request contains user's attributes a and the object/resource r they wish to grant access to. For example, a user with the ID of 0001 is an Intern Engineer from Project 1 wishing to have access to use sector 2 of the company A's supercomputer can be represented as:

$$r\ \langle\{\textit{ID: 0001, Pos: Intern Engineer, Project: R1}\},\ \{\textit{Sector 2}\}\rangle$$

We assume each attribute has only one value per row (atomic). However, this idea should also apply to multi-value instances, as they can be transformed into multiple single-value instances.

Definition 3: A tuple of authorization showing decision d for user u requesting on using resource r is defined as $t\langle q, d\rangle$, for q, is a request notation mentioned in Definition 2, and d is a decision value.

Definition 4: A log file L is a collection of authorization tuple t obtained from historical requesting events. We created a training and a testing dataset from this log file L, denoted as L_t and L_v respectively.

Table 1 summarizes the defined notations used in this paper.

2.3 Evaluation Metrics

To evaluate our proposed method, we measure how similar our generated access permissions are to the original authorization. We expect our model-predicted decision to resemble the original decision rule in the testing data closely. For example, if the decision of the testing dataset for an access request q is permission granted, the decision predicted by our model must be "granted" for the same access request. However, if our model denies the same access request, it would score an incremental value for *False Negative*. We calculate our model's *Accuracy*, *Precision*, *Recall*, and *F1-score* using the standard definition [24].

While we prioritize the *Accuracy* metric in balanced test cases, the metric of *Recall*, *Precision*, and *F1-score* are focused more in the imbalanced instances.

3 Proposed Approach

The steps of our proposed classification process are described in Fig 1. We consider each access request similar to a text sentence, where each attribute's value mimics a word. Since each token includes a meaning and performs a specific role in requesting access

Table 1: Summary of Notations

Notation	**Definition**
U, R, D	Set of users, objects, and decisions
S_i, S_j	Set of user's attributes, resource's states
V_a, V_s, V_d	Value range for a given attribute a, state s, decision r
$q\langle u, o\rangle$	Access request
$t\langle q, d\rangle$	Authorization tuple, showing decision d with respect to request r
L	Historical access log L
L_t,L_v	training data and testing data derived from access log L
d_p	Predicted decision for a request q
L^+	Set of true accepted request q
L^-	Set of true denied request q
$TP_{d_p\|L}$, $FP_{d_p\|L}$,$TN_{d_p\|L}$,$FN_{d_p\|L}$	True Positive, False Positive, True Negative, False Negative
$A_{d_p\|L}$	Accuracy rate, proportion predicted decision d_p that are correct
$R_{d_p\|L}$	Sensitivity, the proportion of predicted decisions correctly classified as accepted in the set of truly accepted request
$P_{d_p\|L}$	Precision, the proportion of predicted decisions correctly classified as accepted in the set of predictively accepted requests
$F_{d_p\|L}$	F-score

to the system, a specifically trained word-embedding model is an excellent candidate to capture the meaning of each token. First, we will separate our data into two datasets for training and testing purposes. For the training process, we split each dataset into two parts: the first section consists of the requests' information, while the second part only includes the permission corresponding to each request. The training dataset's first part trains the embedding layer using a skip-gram model. This embedding model is then employed to convert each token to a numerical vector in training data. This process recasts each request tuple into a matrix. At the same time, the permission label of the training set is transformed into a binary encoding label and rejoined with the matrix-transformed requests' tuples to form a final set of training data. By applying this process, we expect to embed underlying contextual meaning for each token so that our classification model can understand different access requests and issue the correct permissions. Finally, we apply a tree-based classification model to train on the transformed training dataset.

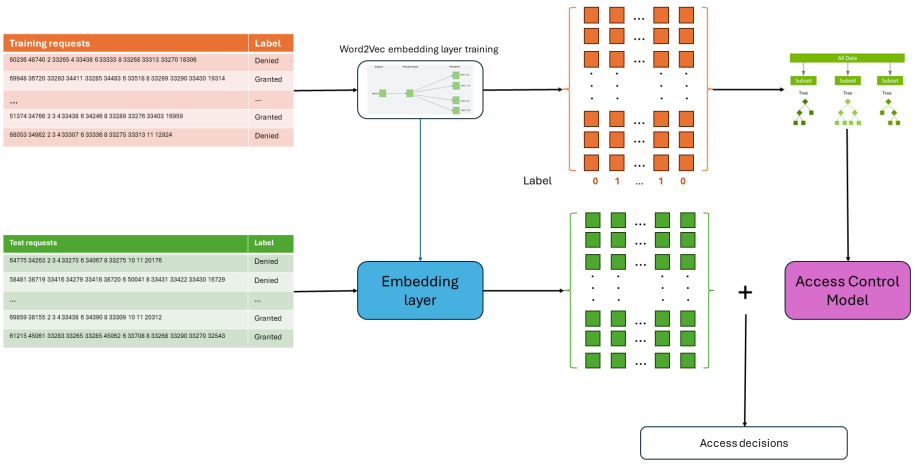

Fig. 1: Pipeline for modeling with proposed method

In the second half of the process, we perform testing to validate the effectiveness of our proposed approach. The testing dataset undergoes the same splitting process as the training dataset. Unlike the training part, we apply the embedding model from the training set directly to the first part of testing data to obtain featured matrices. Our trained access control model uses these matrices to generate access permissions corresponding to each request. Finally, the predicted permission is compared with the actual decision from the testing data to benchmark our model's performance.

3.1 Skip-gram Model

Our candidate model for embedding vector training is the skip-gram structure, and the most known model is Word2Vec [26]. Each token is linked to an indexed vector in this model. These indexed vectors are constructed through the Word2Vec skip-gram algorithm. The embedding for a specific token consists of fixed-dimensional vectors that are initially randomized. Each vector represents the quantitative contextual significance of its corresponding token. The context for each token is determined by a predefined number of tokens preceding and following it, referred to as a "window" in this context. To achieve this objective, the neural network treats all surrounding tokens as labels for the target token and uses the target token as input to predict each label as an output. Following each iteration, all values within each vector are updated using the back-propagation.

3.2 Gradient Boosting (GB) Tree

Gradient Boosting Tree is a form of ensemble learning derived from the decision tree model. Instead of creating a single extensive decision tree with multiple steps, this technique employs boosting, which involves constructing weaker decision trees based on the errors of previous iterations of weak decision trees [27]. Additionally, it often outperforms Random Forest, another ensemble learning method that employs bagging

(where multiple weak decision trees are built in parallel) [28]. Unlike a conventional decision tree, a gradient-boosting tree complicates the training process of a classification model, leading to reduced traceability and significantly longer training times. Nevertheless, the model remains robust and adaptable. Therefore, it is one of the stable options for predicting and classifying tabular datasets [29].

4 Experimental Evaluation

4.1 Experiment Data

We perform our experiments on real datasets provided by Amazon in the UCI archive, which contains the historical access log of Amazon employees. The data documented over $36,000$ employees from Amazon, collected from 2010 to 2011 [30]. While the dataset is over a decade old, it is still relevant and used to experiment with different ABAC models. There are over $33,000$ columns, but they can be characterized into four main categories: users' information, access permissions to system supports, access permissions to host servers, and access permissions to groups.

The dataset we will use for our experiments is created by transforming the Amazon UCI into the form described in the definition section. We extract users' information and store them separately. Next, we obtain the host, resource, and system support IDs from the attribute names and transpose them into a single column. Afterward, we perform a cross-product using the users' information data and the resource data to generate access information of each user to each resource. Finally, we match the cross-product dataset row with the corresponding access decision from the original dataset. This process propagates over 1 billion rows of access requests with decisions. While the number is enormous, it is not feasible for us to train on all the given data. As a proof of concept for our proposed method, we create multiple subsets by extracting rows of data from this extensive dataset. Therefore, subsets with $16,000$, $32,000$, $64,000$, and $128,000$ rows of data are used for experimenting with our approach, namely $16k$, $32k$, $64k$, and $128k$ datasets, respectively. Since the acquired dataset is significantly smaller than the populated dataset, we can generate datasets with approximately equal numbers of rows for each class of decisions. Additionally, we construct two skewed datasets, where the numbers of granted accesses are significantly larger than the denied accesses, to test the robustness of our model, namely $16skew$ and $32skew$ ($16,000$ and $32,000$ rows of data).

4.2 Experimental Setup

To set up our experiment for the proposed model, we randomly split each dataset into testing and training sets, where 80% of the data will be used for training, and the remaining 20% will be used for testing. Since the testing dataset is entirely unseen in the training process, the evaluation should be fair and generalized for our approach and the comparing methods.

We merge all users' attributes into sentence form with spaces to disjoin each token from one another. This process transforms each dataset into two attributes of request q and decision d for each tuple, in which only the request string is passed into the

Word2Vec model to train for the embedding vectors. We also test with different vector dimensions and plot the change in training loss and training time after each 10 epoch to find an approximation of the optimal dimension for each token's vector.

The embedding vectors were developed in *Python* using the *Gensim* library and trained on Arkansas High-Performance Computing Center (AHPCC) using condo nodes (E5-2650v4 2.20 GHz CPUs, 128 GB main memory, and dual NVidia K80 Tesla GPU). For the classification process, we employ an extreme gradient boosting tree model from the *Extreme Gradient Boosting (XGBoost)* library and optimize the parameter using a cross-validation grid-search function using the *Scikit-learn* library with the area under the Receiver Operating Characteristic (ROC) curve (AUC-ROC) score as a benchmark for selecting the best parameters for the tree-based model.

As far as we know, there haven't been any efforts to utilize NLP word embedding specifically for access control tokens. This makes it challenging to compare our approach directly with other embedding techniques. Consequently, we evaluate our results based on the same datasets and setup, comparing them with algorithms applicable in a broader context. Despite numerous studies on ABAC systems, our choice of model is primarily influenced by the availability of source code and our capacity to adapt the algorithm to achieve desired outcomes. Hence, we decided to use the deep-learning approach ABAC proposed by Nobi et al. as the benchmarking model for our proposed method, which we refer to as the $DLBAC_\alpha$ model [25].

***DLBAC*$_\alpha$**: A deep-learning ABAC model is built on Python using the *Keras* library with *Tensorflow* backend. We utilize the architecture and hyperparameters that are faithful to the source code. The model is also trained using the same hardware setup as ours to reduce bias when comparing the results. The training process is the same as described in the paper, with a batch size of 16, 60 epochs, Adam optimizer, and an initial learning rate of 0.001, decay in 10 folds every 10 epochs.

4.3 Evaluation Metrics

We use *Accuracy*, *Recall*, *Precision*, and *F1-score* as the benchmark to measure how effective a model performs. Since our experiments were conducted in balanced and imbalanced datasets, *Accuracy* can indicate a certain level of correctness of each method's prediction. However, for the imbalanced dataset, *Precision* is a more accurate measurement than *Accuracy* on how a model can grant access for requests. *Recall* is also an excellent metric to ensure that a model does not unconditionally allow requests to enter a system. Finally, *F1-score* is considered an overview metric to determine how well a model grants or denies an access request, as it is the harmonic mean of *Precision* and *Recall*, which can give an overall outlook on how both metrics were calibrated.

5 Evaluation

5.1 Parameter Tuning

We construct several embedding models with different settings. Since each request tuple is relatively short, picking a *window size* value to be small so that Word2Vec can

better understand each token's surroundings is reasonable. Another critical parameter we must be attentive to is the *vector size*. While a higher number of dimensions for each vector can yield better results, there is a certain threshold where the diminishing returns will take effect. Also, the training resources increase significantly as each vector's dimension expands. To obtain a close approximation of the optimal number of the vector's dimensions, we perform a grid search with an incremental of 50 for this parameter. Next, we collect the training loss and elapsed time to plot training graphs and decide on a candidate value for the vector's size. Training loss and time are shown in Fig 2

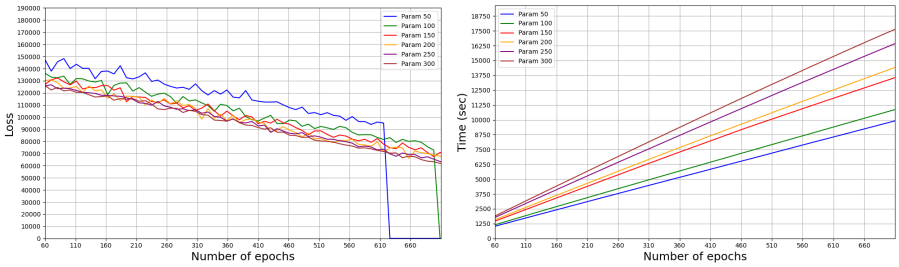

Fig. 2: Word2Vec Training Loss and Time

The graphs indicate that the loss values are close among different parameter settings. However, there were abrupt drops of loss when the vector sizes were 50 and 100, possibly due to an overfitting issue or inability to learn after a certain number of epochs. Considering the training time, a vector size of 150 seems to be a good candidate for our experiment.

Additionally, we conduct a grid search to optimize our XGB classification model, exploring various configurations. We employ 5-fold cross-validation for each configuration to ensure robust performance and select the setting with the highest average benchmark score. Our experimentation indicates that the XGB model achieved the highest AUC-ROC score with the following settings: a learning rate of 0.3, a maximum tree depth of 6, a minimum child weight of 1, and a subsample ratio of 1.

5.2 Result

We evaluate our proposal against baseline models using different datasets derived from the Amazon UCI access samples dataset. Table 2 shows the results of these experiments across different metrics for our model and $DLBAC_\alpha$. We can infer the following remarks based on the experimental results:

- *Contextual embedding layer achieves significantly better results across different datasets and experimental setups.* Our approach performs better than the $DLBAC_\alpha$ model for the balanced and imbalanced datasets in all metrics across different experimental datasets. While $DLBAC_\alpha$ has lower Accuracy across different datasets, its Precision and Recall are slightly better. This is likely because the model's setting

Table 2: Experimental Results

Method	Dataset	Accuracy	Precision	Recall	F1-Score
Proposed Approach	16k	90.66%	90.86%	90.71%	90.65%
	32k	91.89%	92.02%	91.90%	91.89%
	64k	92.08%	92.21%	92.07%	92.07%
	128k	93.28%	93.37%	93.27%	93.28%
	16skew	94.14%	94.35%	94.14%	93.76%
	32skew	94.27%	94.41%	94.27%	93.89%
DLBAC	16k	81.03%	81.76%	81.15%	80.96%
	32k	81.84%	82.26%	81.86%	81.79%
	64k	82.05%	82.53%	82.03%	81.97%
	128k	82.09%	82.63%	82.05%	82.00%
	16skew	85.50%	85.19%	85.50%	82.61%
	32skew	85.80%	85.23%	85.80%	82.62%

was optimized to train using a skewed dataset, shown as $DLBAC_\alpha$ performance is better on our imbalanced setup. On the other hand, our approach obtains similar consistency in results across all the metrics on all datasets, demonstrating the ability to learn and understand the context of each request of our model to assign proper permission.

- *Experimental outcomes improve as the dataset size increases.* Both our model and $DLBAC_\alpha$ score the least with $16k$ dataset and the most with $128k$ dataset for the balanced setup. A similar trend can be seen with the imbalanced datasets. However, the improvement rates are much slower even though the dataset doubles in size in every setup. This suggests that both models can be effectively trained with a few inputs and are generalized.
- *Imbalanced setup yields better results than the balanced setup.* As the results indicate, the imbalanced setup averages over 1% better in overall metrics compared to the balanced instances. While the difference is not quite significant, it is impressive, considering that the size of datasets for the imbalanced experiments is much smaller than the balanced setup. Additionally, in real-world scenarios, it is unlikely that a user will try to submit an access request to unassigned resources, which makes the number of denied accesses naturally much lower than the number of access granted ones. Our proposed model and $DLBAC_\alpha$ show the capabilities to learn and understand the context of access requests and assign proper permission for each, even on rare occasions.

Overall, our proposed approach shows significantly better results in different setups across all metrics, suggesting the capability of learning the underlying context of the access requests. Moreover, the knowledge of embedding vectors can be transferred to similar instances, and we also observe a significantly lower training time for our approach. This suggests the potential of applying our proposed method as a lightweight approach for access control in cloud or fog computing, where the hardware capacity is limited.

6 Conclusions and Future Researches

In this section, we discuss some challenges of our approach, possible improvements, and future research directions for applying NLP to the access control model.

Unknown Token: While the proposed approach shows promising results in our experiments, it suffers from issues when new values are introduced when policies are added or updated. This can be resolved by recalibrating new tokens' empirical vectors in tandem with the old tokens. This process can be designed by training newly introduced tokens with the old tokens to generate representative vectors associated with new tokens, which can be done using a continual learning approach. Furthermore, a pipeline of frequent updates with supervision is an excellent long-term solution for this problem.

Tokens Representing Different States: One major drawback of the Word2Vec model is its limitation, where each token corresponds to only one vector, creating difficulties when a token has multiple meanings. Although this issue is relatively uncommon in access control requests, there's no guarantee it won't arise in the future. Meanwhile, Bidirectional Encoder Representations from Transformers (BERT) presents a more encouraging approach for word embedding in NLP. It adeptly deals with homonyms, unlike Word2Vec [31]. Nonetheless, the training process for BERT from scratch could be expensive and requires further exploration as an application for access control issues.

Proposed Approach with Traditional Access Control: The experimental results from our model suggest the potential of applying NLP techniques to address access control and possibly other cybersecurity concerns. Nevertheless, we do not propose that our method will replace conventional access control methods anytime soon. Instead, we advocate for integrating our model with traditional access control systems, leveraging their established practicality. Conflicting decisions between these models can serve as valuable insights to refine our approach for better outcomes.

7 Acknowledgment

This research is supported by the Arkansas High-Performance Computing Center, which is funded through multiple National Science Foundation grants and the Arkansas Economic Development Commission.

The research work of Brajendra Panda was supported in part by grant H98230-22-1-0321 issued by the National Security Agency as part of the National Centers of Academic Excellence in Cybersecurity's mission to expand cybersecurity research and education for the Nation.

References

[1] Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: On protection in operating systems. SIGOPS Oper. Syst. Rev. **9**(5), 14–24 (1975) https://doi.org/10.1145/1067629.806517

[2] Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **29**(2), 38–47 (1996) https://doi.org/10.1109/2.485845

[3] Hu, V., Ferraiolo, D., Kuhn, D., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to attribute based access control (abac) definition and considerations. National Institute of Standards and Technology Special Publication, 162–800 (2014)

[4] Karp, A., Haury, H., Davis, M.H.: From abac to zbac: The evolution of access control models. ISSA (Information Systems Security Association). Journal **8**, 22–30 (2010)

[5] Frank, M., Basin, D., Buhmann, J.M.: A class of probabilistic models for role engineering. In: Proceedings of the 15th ACM Conference on Computer and Communications Security. CCS '08, pp. 299–310. Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1455770.1455809 . https://doi.org/10.1145/1455770.1455809

[6] Asudani, D., Nagwani, N.: Impact of word embedding models on text analytics in deep learning environment: a review. Artificial Intelligence Review **56**, 1–81 (2023) https://doi.org/10.1007/s10462-023-10419-1

[7] Huang, X., Khetan, A., Cvitkovic, M., Karnin, Z.S.: Tabtransformer: Tabular data modeling using contextual embeddings. CoRR **abs/2012.06678** (2020) 2012.06678

[8] Molloy, I., Li, N., Li, T., Mao, Z., Wang, Q., Lobo, J.: Evaluating role mining algorithms. In: Proceedings of the 14th ACM Symposium on Access Control Models and Technologies. SACMAT '09, pp. 95–104. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1542207.1542224 . https://doi.org/10.1145/1542207.1542224

[9] Jafarian, J.H., Takabi, H., Touati, H., Hesamifard, E., Shehab, M.: Towards a general framework for optimal role mining: A constraint satisfaction approach. In: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies. SACMAT '15, pp. 211–220. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2752952.2752975 . https://doi.org/10.1145/2752952.2752975

[10] Xu, Z., Stoller, S.D.: Mining attribute-based access control policies. CoRR **abs/1306.2401** (2013) 1306.2401

[11] Medvet, E., Bartoli, A., Carminati, B., Ferrari, E.: Evolutionary inference of attribute-based access control policies, vol. 9018 (2015). https://doi.org/10.1007/978-3-319-15934-8_24

[12] Talukdar, T., Batra, G., Vaidya, J., Atluri, V., Sural, S.: Efficient bottom-up

mining of attribute based access control policies. 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC) (2017) https://doi.org/10.1109/cic.2017.00051

[13] Iyer, P., Masoumzadeh, A.: Mining positive and negative attribute-based access control policy rules. In: Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies. SACMAT '18, pp. 161–172. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3205977.3205988 . https://doi.org/10.1145/3205977.3205988

[14] Bui, T., Stoller, S.D., Li, J.: Mining relationship-based access control policies. CoRR **abs/1708.04749** (2017) 1708.04749

[15] Iyer, P., Masoumzadeh, A.: Active learning of relationship-based access control policies. In: Proceedings of the 25th ACM Symposium on Access Control Models and Technologies. SACMAT '20, pp. 155–166. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3381991.3395614 . https://doi.org/10.1145/3381991.3395614

[16] Cotrini, C., Corinzia, L., Weghorn, T., Basin, D.: The next 700 policy miners: A universal method for building policy miners. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. CCS '19, pp. 95–112. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3354196 . https://doi.org/10.1145/3319535.3354196

[17] Chari, S.N., Molloy, I.M.: Generation of attribute based access control policy from existing authorization system (2016)

[18] Narouei, M., Khanpour, H., Takabi, H., Parde, N., Nielsen, R.: Towards a top-down policy engineering framework for attribute-based access control. In: Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies. SACMAT '17 Abstracts, pp. 103–114. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3078861.3078874 . https://doi.org/10.1145/3078861.3078874

[19] Hadj, M., Benkaouz, Y., Freisleben, B., Erradi, M.: Abac rule reduction via similarity computation, pp. 86–100 (2017). https://doi.org/10.1007/978-3-319-59647-1_7

[20] Bui, T., Stoller, S.D.: A decision tree learning approach for mining relationship-based access control policies. In: Proceedings of the 25th ACM Symposium on Access Control Models and Technologies. SACMAT '20, pp. 167–178. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3381991.3395619 . https://doi.org/10.1145/3381991.3395619

[21] Karimi, L., Joshi, J.: An unsupervised learning based approach for mining attribute based access control policies, pp. 1427–1436 (2018). https://doi.org/10.

1109/BigData.2018.8622037

[22] Narouei, M., Takabi, D.: A Nature-Inspired Framework for Optimal Mining of Attribute-Based Access Control Policies, pp. 489–506 (2019). https://doi.org/10.1007/978-3-030-37231-6_29

[23] Kavšek, B., Lavrac, N., Jovanoski, V.: Apriori-sd: Adapting association rule learning to subgroup discovery, vol. 20, pp. 230–241 (2008). https://doi.org/10.1007/978-3-540-45231-7_22

[24] Karimi, L., Aldairi, M., Joshi, J., Abdelhakim, M.: An automatic attribute based access control policy extraction from access logs. CoRR **abs/2003.07270** (2020) 2003.07270

[25] Nobi, M.N., Krishnan, R., Huang, Y., Shakarami, M., Sandhu, R.: Toward deep learning based access control. In: Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy. CODASPY '22. ACM, ??? (2022). https://doi.org/10.1145/3508398.3511497 . http://dx.doi.org/10.1145/3508398.3511497

[26] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space (2013). https://arxiv.org/abs/1301.3781

[27] Friedman, J.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics **29** (2000) https://doi.org/10.1214/aos/1013203451

[28] Hastie, T., Tibshirani, R., Friedman, J.: Boosting and Additive Trees, pp. 337–387. Springer, New York, NY (2009). https://doi.org/10.1007/978-0-387-84858-7_10 . https://doi.org/10.1007/978-0-387-84858-7$_1$0

[29] Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., Kasneci, G.: Deep neural networks and tabular data: A survey. IEEE Transactions on Neural Networks and Learning Systems **35**(6), 7499–7519 (2024) https://doi.org/10.1109/tnnls.2022.3229161

[30] Montanez, K.: Amazon Access Samples. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5JW2K (2011)

[31] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). https://doi.org/10.18653/v1/N19-1423 . https://aclanthology.org/N19-1423