Adversarial Attacks in Machine Learning Based Access Control

Mohammad Nur Nobi[1,2,4], Ram Krishnan[1,2,3,5] and Ravi Sandhu[1,2,3,4]

[1]The University of Texas at San Antonio (UTSA), Texas, USA 78249
[2]Institute for Cyber Security (ICS), UTSA, Texas, USA 78249
[3]NSF Center for Security and Privacy Enhanced Cloud Computing (C-SPECC), UTSA, Texas, USA 78249
[4]Department of Computer Science, UTSA, Texas, USA 78249
[5]Department of Electrical and Computer Engineering, UTSA, Texas, USA 78249

Abstract

While machine learning-based access control systems demonstrated their potential in large-scale and complex applications context for accurate and generalized access decisions, the security of such systems is equally important due to the vulnerability of a machine learning (ML) model to adversarial attacks. A small modification to the input could lead the ML model to make a completely inaccurate access decision. This paper explores ML-based access control's adversarial attack problem, focusing on manipulating information of users and resources to gain unauthorized access. We experiment with this problem in two simulated systems where a ResNet model decides access. We demonstrate that it is possible to design adversarial attacks for ML models deployed for access decisions. Also, there is potential to reduce adversarial attacks to some extent by utilizing access control-specific constraints.

Keywords

authorization, access control, machine learning, adversarial attack

1. Introduction

With advances in Big data, the Internet of Things (IoT), cloud computing, etc., the demand for a more dynamic and efficient access control system is escalating. The traditional systems, such as RBAC [1], ABAC [2], ReBAC [3], etc., have demonstrated their effectiveness in the access control arena for a long time. However, with the increased complexity of computing systems, their generality, flexibility, and usability come at a higher cost and are somewhat inadequate [4, 5, 6]. To deal with such large-scale access control systems, the use of Machine Learning (ML) is becoming common in different areas such as policy mining [7, 8, 9], attribute engineering [10], role mining [11], etc. Contemporary researches also manifest the advantages of using an ML model for more accurate access control decision-making [4, 6, 12, 13, 14, 15]. These systems decide accesses based on a trained ML model instead of a written access control policy [16]. We refer to such systems as machine learning based access control (MLBAC). In

ITADATA2022: The 1st Italian Conference on Big Data and Data Science, September 20–21, 2022, Milan, Italy
mohammadnur.nobi@my.utsa.edu (M. N. Nobi); ram.krishnan@utsa.edu (R. Krishnan); ravi.sandhu@utsa.edu (R. Sandhu)
http://profsandhu.com (R. Sandhu)
0000-0002-2974-552X (M. N. Nobi); 0000-0002-7402-553X (R. Krishnan); 0000-0002-3165-1813 (R. Sandhu)

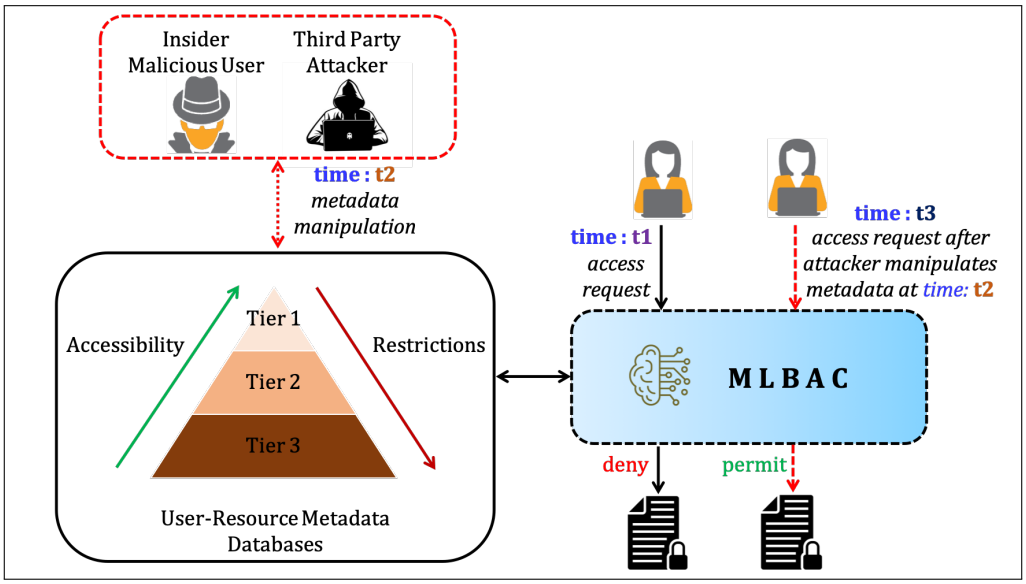


Figure 1: The Adversarial Attack Problem in MLBAC.

MLBAC, the access control decisions (grant or deny) are made using user and resource metadata and attributes. Metadata and attributes are the user/resource features that an ML model learns for subsequent access decisions in MLBAC. For simplicity, we refer to both metadata and attributes as 'metadata'. These metadata could be expressed in categorical (e.g., 'security_level,' 'designation') and continuous (e.g., 'age,' 'salary') data.

Among different ML models, neural networks are prevalent for obtaining a generalized and accurate MLBAC system due to their ability to capture features from complex input [4, 6, 14]. Such quality of the neural networks poses some security concerns as they are highly sensitive to the minor changes in the input— that is, a slight manipulation or introduction of additional information to the input may result in an unintended output [17]. In ML, this issue is known as an *adversarial attack*, and the manipulated data is an *adversarial example*. For instance, in MLBAC, an attacker could perturb the user/resource metadata to gain access to a resource forcibly. Figure 1 demonstrates the adversarial attack problem in MLBAC. An MLBAC is deployed in a system where a user requests to access a resource at time 't1', which the system denies. At time 't2', the user by itself or a third-party adversary purposefully manipulates the respective user and resource metadata to gain access. As illustrated in the figure, system administrators (sysadmin) may store metadata in the databases with different levels of security restrictions, e.g., Tier 1, Tier 2, etc. Therefore, the adversary may or may not equally access and modify each metadata. For example, an adversary may not have access to the 'job_role' metadata as a more restricted database could store them. On the contrary, the adversary may manipulate (e.g., 'expertise' metadata) or influence (e.g., login_frequency) some metadata to which the user may have direct or indirect access. As shown in the figure, after the manipulation, the user requests the same access at the time 't3', which is eventually granted by the MLBAC system.

This area is studied carefully and comprehensively in the domains where inputs are generally images [18, 19], e.g., computer vision. However, this problem is equally important in the access

control domain, especially for MLBAC, where the input data is *non-image*. As discussed, in MLBAC, the input is the user/resource's metadata which is *tabular* data expressed in terms of a set of categorical and continuous values. Also, as shown in Figure 1, metadata may have different levels of restrictions. Therefore, the adversarial attack needs to be investigated concerning the access control domain. This paper investigates adversarial attacks in MLBAC. In particular, we consider an MLBAC where the ML model is a deep neural network. To the best of our knowledge, this is the first work of its kind in the access control domain. We summarize our contributions as follows.

- We define adversarial attack problems in the MLBAC context. We also propose customized objective functions for imposing additional constraints in the same context.
- We simulate the adversarial attack cases in the deep neural network-based MLBAC with two complex and large-scale systems. Also, we consider underlying user/resource metadata in both systems contain a mix of categorical and continuous data.
- We evaluate the performance of our experimentation and demonstrate that applying accessibility constraints makes it harder for the adversarial attack generation.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 defines the adversarial attack problem in the context of MLBAC. Later, we discuss overall experimentation methodologies in Section 4. We present evaluation metrics and results in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

2.1. Machine Learning Based Access Control (MLBAC)

In MLBAC, an ML model decides accesses based user and resource's metadata and attributes. We briefly discuss them below.

Cappelletti et al. [6] experiment with multiple ML techniques, including Decision Tree [20], Random Forest (RF) [21], Support Vector Machines (SVM) [22], and Multi-Layer Perceptron (MLP) [23], for making access control decisions. The authors train the ML models using access history. The authors suggest using neural networks for complex systems. Chang et al. [12] present an SVM-based time-constraint access control approach associated with the time (e.g., a user may only have access to a resource during office hours). When an access request is made, the trained SVM grants security access based on the user's information and given time. Liu et al. [13] propose a technique utilizing an RF algorithm. The authors build RF based on the access logs of an existing access control system.

A more advanced ML algorithm is also becoming common to tackle situations where access control requirements are complex. For example, Karimi et al. [4] establish a framework using a reinforcement learning (RL) algorithm that maps access requests to the access decisions. An RL agent learns access control policies through feedback (rewards for the right decision, otherwise penalties) from the users/administrators and subsequently makes access decisions. Srivastava et al. [15] developed risk adaptive access control (RAdAC) using a neural network

and an RF algorithm for dynamic access decisions. Before deciding on accesses, the RAdAC assesses the genuineness of the user and the risk of the access based on metadata such as access time, location, user history, resource sensitivity, etc. Recently, Nobi et al. [14] proposed a Deep Learning Based Access Control (DLBAC) using deep neural networks, including ResNet [24], DenseNet [25], etc. The DLBAC train ML models using authorization tuples that represent the existing access control state. The evaluation results suggest that DLBAC can make more generalized and accurate access decisions than an ABAC system and other ML methods. The authors also provide two strategies for understanding DLBAC decisions by exploiting deep learning interpretation techniques.

2.2. Adversarial Attack for Tabular Data

As discussed, the input data in the access control domain is in tabular format—this section reviews existing adversarial attacks in the tabular data context.

There are tremendous research efforts for the adversarial attacks and defenses in the different domains, such as image, graph, text, etc. [18, 19, 26]. However, we do not see similar efforts for the tabular data domain. Ballet et al. [17] propose an imperceptible adversarial attack for the tabular data domain. The authors manipulate less important features to a greater extent, adding a minimal perturbation to the more important features to ensure imperceptibility. Cartella et al. [27] present an adversarial attack for imbalanced tabular data for fraud detection. The proposed method obtains adversarial examples that are less perceptible when analyzed by humans. Kumar et al. [28] propose an adversarial attack for a payment system, focusing on generating adversarial examples on the tabular dataset with limited resources (the least number of queries used). The authors experiment with a gradient-free approach in black-box settings. Mathov et al. [29] propose a framework for adversarial examples in heterogeneous tabular data, including discrete, real-number values, categorical, etc. The proposed framework defines a distribution-aware constraint and then incorporates them by embedding the heterogeneous input into a continuous latent space.

3. Adversarial Attack in MLBAC

3.1. Adversarial Attack Problem

This section discusses the MLBAC adversarial attack problem. For any access request to the MLBAC, the input to the model is both user and resource metadata, and the MLBAC decides whether the user has access or not. Given the ground truth of an access decision of a request and respective user/resource metadata, we can express them as a tuple of user metadata, resource metadata, and an access decision. We refer to such tuple as an *authorization tuple*. For example, an authorization tuple $\langle$u1, r1, {grant}$\rangle$ indicates that a user u1 has access to the resource r1. The u1 and r1 also represent the set of their respective metadata and metadata values.

We consider $\mathbb{X}$ represents a set of authorization tuples where user and resource metadata are denoted by x_a with $a \in [1 \ldots N]$ and associated with a binary access decision y_a representing either grant (1) or deny (0). Also, x_a is defined by a 'vector' of user and resource metadata values and expressed as: $x_a = \left\langle m_1^u : v_1, m_2^u : v_2, \ldots, m_i^u : v_i, m_1^r : v_1, m_2^r : v_2, \ldots, m_j^r : v_j \right\rangle$. The pair

$m_i^u : v_i$ indicates v_i is the value of i^{th} user metadata m_i^u. Similarly, the pair $m_j^r : v_j$ indicates v_j is the value of j^{th} resource metadata m_j^r.

Also, we consider $f : \mathbb{X} \longrightarrow \{0, 1\}$, a binary classifier mapping to access decisions where the grant is represented by '1' and deny as '0', which is obtained from comparing the probability of access at the output of the classifier with a *threshold*.

For a given user-resource metadata values x, the access decision $y = f(x)$, and target access decision $t \neq y$, we aim to generate an optimal perturbation x_p such that

$$f(x) = y \neq f(x + x_p) = t$$

Our goal is to generate an adversarial example such that we can minimize the amount of perturbation (x_p) and gain the desired access t. To accomplish that, we define our objective function as the accumulation of access change constraint and minimization of x_p:

$$g(x_p) = \mathcal{L}(x + x_p, t) + \omega \| x_p \|$$

Where $\mathcal{L}(x, t)$ is the value of loss of the binary classifier f calculated for the input x and target access decision t. Also, $\| x_p \|$ measures the l_p norm of perturbation and $\omega > 0$ is the weight associated to the amount of perturbation.

3.2. Accessibility Constraint

The objective function we discussed above considers that the attacker has the flexibility to modify any user/resource metadata to obtain their desired access. In MLBAC, there are metadata with different levels of restrictions, as discussed in Section 1 and illustrated in Figure 1. Therefore, an adversary can not access and modify every metadata equally. A sysadmin could explicitly impose such restrictions, and the adversary could not directly access and modify some of the metadata. For example, a user's 'job_role' information comes from the system, which is difficult, if not impossible, to modify and will take more effort. On the other hand, the adversary could influence some metadata, such as 'expertise', and maybe modify it with less effort. Without loss of generality, we refer to such notion of restrictions and the ability to access them as the *accessibility* constraint.

To impose the accessibility constraint while generating perturbation (x_p), we extend our objective function as below:

$$g(x_p) = \mathcal{L}(x + x_p, t) + \omega \| x_p \circ c \|$$

Where c is the accessibility constraint expressing the magnitude of restrictions with respect to each metadata, $\circ$ is the element-wise product operator, and $\omega > 0$ is the weight for the penalty of perturbing metadata with higher accessibility constraint. We hypothesize that exploiting accessibility constraint with the proposed objective function will make the perturbation generation harder.

3.3. Adversarial Attack Approach

After formulating objective functions, we aim to develop an approach to optimize the objective function minimization problem and generate perturbed examples. Researchers have proposed numerous approaches, such as the Fast Gradient Sign Method (FGSM) [30], Projected Gradient Descent (PGD) [31], Carlini & Wagner Attacks [32], etc., to generate adversarial attacks that can deceive a trained deep neural network with higher accuracy. However, most of the algorithms have been proposed for the image domain. Due to the data characteristics in the access control domain, these algorithms may not readily mislead the ML model in MLBAC. A few works consider the non-image tabular data domain [17, 27, 33]. These works have demonstrated their potential in generating adversarial attacks for ML models applied in multiple use-cases such as financial services [17], fraud detection [27], etc.

Ballet et al. [17] propose the LowProFool algorithm to solve the optimization problem for the continuous tabular data using a gradient descent approach. As MLBAC input is tabular data containing both *categorical* and *continuous* metadata, and we have a similar optimization problem, we adopt their algorithm for minimizing our objective function g. We customize the algorithm and adjust the parameters such that we can optimize our objective function and supports continuous and categorical data. We further discuss this in Section 4.5.

4. Methodology

4.1. Determining Accessibility Constraint

In MLBAC system, the access decisions are made based on user/resource metadata. In practice, this metadata could come from different sources, where some sources are more secured than others. For example, a highly restricted database may store a user's sensitive information, such as 'job_role.' In contrast, a less secured database may store less sensitive data such as 'expertise.' Also, not every metadata/attributes equally influences the access decisions [14]. Therefore, the sysadmin may restrict influential and sensitive metadata/attributes with more security. We utilize the notion of accessibility constraint (as discussed in Section 3.2) to *simulate* such security restriction levels for each of the metadata.

Instead of randomly determining some metadata as more restricted and others as less restricted, we measure the correlation for each user and resource metadata for a decision. More concretely, we calculate the absolute value of Pearson's [34] correlation coefficient of each metadata with respect to the access decision. For this purpose, we could utilize other methods crafted specifically for the access control domain, such as the *global interpretation*-based method developed by Nobi et al. [14], where they determine the influence of each metadata for a specific decision for an ML model. However, we rely on Pearson's correlation as it can better simulate human intuition through a linear correlation based on the data [17]. For our proposed objective function, we generate accessibility constraint values in the range of 0 to 1 for each metadata. A higher constraint value of metadata indicates the respective metadata is more secured, thereby less accessible by the adversary while generating perturbation.

4.2. Dataset for MLBAC Adversarial Attack Experimentation

MLBAC makes access control decisions based on user and resource metadata and attributes values. Before that, MLBAC needs training with available *ground truth* to decide on subsequent access control requests. Generally, the ground truth could be access history, existing authorization information, etc., communicated through authorization tuples [6, 13, 14]. (We discussed the authorization tuple in Section 3.1.) We call such a collection of authorization tuples a *dataset* that represents the current access control state of a system.

Each dataset represents the metadata information of users and resources and their access control state. However, metadata values could be a *mix* of continuous (e.g., 'age,' 'salary') and categorical (e.g., 'security_level,' 'designation') data. We are unaware of any publicly available access control dataset representing mixed metadata values. There are two access control datasets from Amazon [35, 36]. Both of these datasets contain only categorical metadata/attributes values. As a result, for this work, we experiment with two simulated datasets[1], named *u5k-r5k-auth12k* and *u5k-r5k-auth19k*, developed by Nobi et al. [14]. The authors create these datasets using the data generation method proposed in [37] and introduce different complexities to simulate some real-world scenario. Each dataset has around five thousand users, five thousand resources, and four different operations. The *u5k-r5k-auth12k* dataset, we refer to as **System-1**, has around 12K authorization tuples. Each user has eight user metadata (*umeta*0, *umeta*1, ..., *umeta*7), and each resource has eight resource metadata (*rmeta*0, *rmeta*1, ..., *rmeta*7). On the other hand, the *u5k-r5k-auth19k* dataset has around 19K authorization tuples, whereas a user has ten metadata (*umeta*0, *umeta*1, ..., *umeta*9) and a resource has ten metadata (*rmeta*0, *rmeta*1, ..., *rmeta*9). We refer to this dataset as **System-2**.

However, both datasets contain *categorical* metadata values, as integers, of which each value represents a category. In practice, one could anticipate a mix of continuous and categorical data [29]. To simulate the notion of mixed data, we consider metadata values of both datasets as mixed data. In particular, we assume that the first half of the user and resource metadata are continuous and consider an integer a real value. We also consider the rest of the metadata as nominal categorical data, where each integer represents a category, and the order of the category does not matter. As the metadata contains mixed data, they need to be processed efficiently such that the underlying ML model can properly consume the training data [38]. The following section discusses their preprocessing.

4.3. Data Preprocessing

This section explains the processing of the System-1, which is equally applicable to System-2. $\Big\langle 1011|2021|30\ 49\ 5\ 26\ 63\ 129\ 3\ 42\ |\ 43\ 49\ 5\ 16\ 63\ 108\ 3\ 3\ |\langle 1\ 1\ 0\ 1\rangle\Big\rangle$ is a sample authorization tuple of the System-1 dataset where 1011 and 2021 are the user and resource's unique numbers. The next eight elements indicate the metadata values of a user, the following eight elements represent the resource's metadata values, and the final four binary digits (1 for grant, 0 for deny) signify four different operation's access to the resource.

For our experiment, we consider the first four user metadata (*umeta*0 to *umeta*3) and the first four resource metadata (*rmeta*0 to *rmeta*3) to be continuous, and the rest of the metadata are

[1] https://github.com/dlbac/DlbacAlpha/tree/main/dataset/synthetic

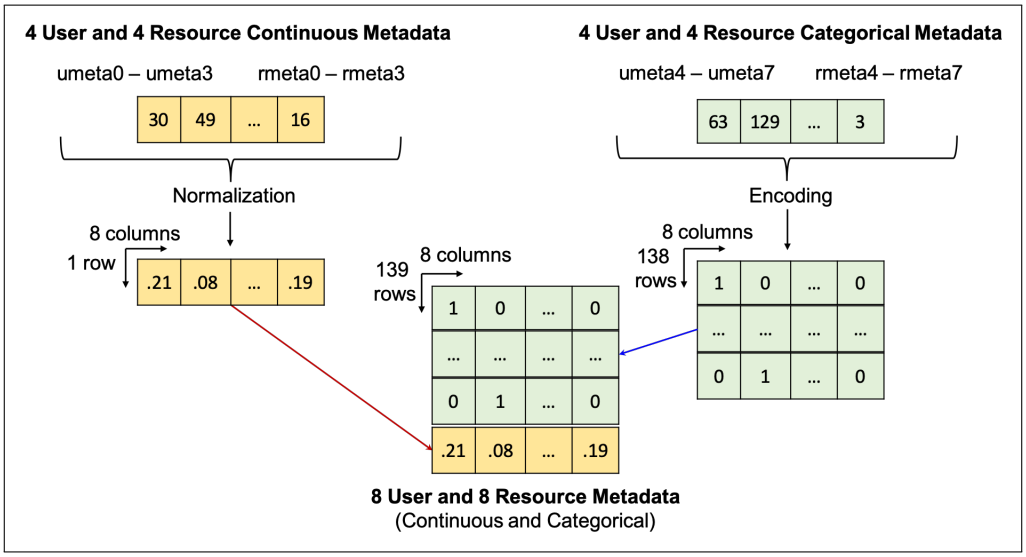


Figure 2: Input Preprocessing in System-1 Dataset.

categorical. Also, we utilize only *one* operation (out of four operations) that indicates whether the user has access to the respective resource or not (grant/deny). Figure 2 illustrates the data preprocessing of System-1 dataset. We normalize continuous data using the *MinMaxScaler* approach [39] and apply one-hot encoding [38] for the categorical portion of the metadata. As shown in the figure, we merge continuous normalized data, a vector, with the one-hot encoded categorical data matrix. The merged matrix, holding values between 0 and 1, acts as the ML model's input.

4.4. Candidate ML Model for MLBAC

A deep neural network has significant benefits for controlling access in dynamic, complex, and large-scale systems [14]. Considering the scale of our datasets and data characteristics, we utilize a deep neural network for MLBAC. In particular, we consider ResNet [24] as our candidate ML model. We note that one could use other deep neural networks, including Xception [40], DenseNet [25], etc., although we do not anticipate any significant changes in our results.

For System-1 and System-2 datasets, we exploit the ResNet ML models with 8 and 50 residual layers, respectively, as suggested in [14]. There is a convolution, a batch normalization operation [41], and a ReLU [42] activation function in each layer. The final activation layer's output is flattened and fed into a dense layer with a *sigmoid* activation function as the MLBAC needs to make a binary access decision (grant or deny).

4.5. Customization of LowProFool Algorithm

As discussed in Section 3.3, we exploit the LowProFool [17] algorithm for generating adversarial examples for the MLBAC. We modify the algorithm and parameters to support continuous and categorical data and help minimize our proposed objective functions discussed in Section 3.1

and 3.2. We replace the objective function with our proposed one and take accessibility constraint (c) as input. We define the loss function $\mathcal{L}$ as the binary cross-entropy function. Also, we guide the perturbation to the *positive* of the gradient if our target access is a 'grant (1)'. Otherwise, we guide to the *negative* of the gradient. As discussed, our experiment's preprocessed input holds values between 0 and 1. Therefore, we modify the $clip(x)$ function to control the value of each metadata such that it does not cross the range. We floor the value to 1 if it exceeds one, and we ceil it to zero if the value becomes negative.

In addition, the algorithm controls the growth of the perturbation using the scaling factor parameter α. Also, we need to provide a weight parameter ω that helps to minimize perturbing the metadata with higher accessibility constraints. For the System-1, we choose the parameters α and ω as 0.2 and 6, respectively. These values are 0.7 and 5.7 for the System-2. We follow a trial-and-error process for choosing them. We have created a GitHub repository consisting of our experimentation's source code and datasets[2].

5. Evaluation

5.1. Evaluation Metrics

The efficiency of an adversarial attack is measured as the ratio of successfully crafted adversarial examples, and the total number of samples attempted to prepare the adversarial example. This ratio is known as the **Success Rate** [17]. Let us consider $\mathbb{X}$ as the number of samples attempted for the adversarial example creation. We also consider $\mathbb{A}$ containing every tuple $(x,\ x_a)$ such that $x \in \mathbb{X}$ and x_a a successfully crafted adversarial examples from x with $f(x) \neq f(x_a)$. We define the Success Rate as follows:

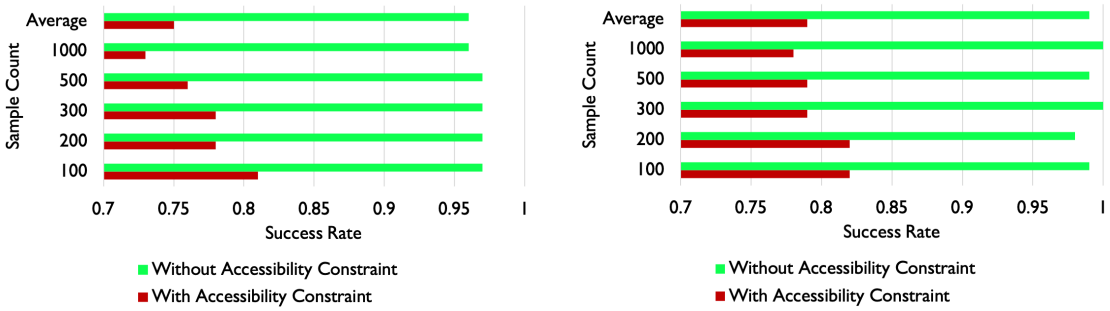
$$Success\,Rate\ =\ \frac{|\mathbb{A}|}{|\mathbb{X}|}$$

5.2. Evaluation Methodology

We propose two different objective functions for the MLBAC system, as discussed in Section 3.1 and 3.2. In one case, we consider that adversary has equal access to every metadata and, thereby, could change any metadata at any level. In another case, we consider a scenario where each metadata may have been stored in places with different levels of restrictions (accessibility constraint). As a result, an adversary could not access and modify every metadata equally. In such a case, the adversarial example generation would be harder, and we anticipate a lower Success Rate. We evaluate both cases and refer to the first as the 'Without Accessibility Constraint' and the latter as the 'With Accessibility Constraint.'

As discussed, we experiment with two datasets System-1 and System-2, having a different number of samples and data characteristics. We evaluate the performance of both of the datasets. Also, we create a different subset of samples measured by sample count and eventually determine the average performance for the respective dataset.

[2]https://github.com/dlbac/MLBAC-AdversarialAttack



(a) Performance Evaluation of System-1.

(b) Performance Evaluation of System-2.

Figure 3: MLBAC Adversarial Attack Performance Evaluation.

5.3. Results

Figure 3 illustrates the overall performance of MLBAC adversarial attack simulation. Figure 3a demonstrates the performance of System-1. As shown in the figure, the success rate of adversarial attacks without imposing any additional constraint is above 95%. However, when we restrict the system considering accessibility constraint, the success rate reduces to an average of 75%. We observe a similar result in the System-2 case, as demonstrated in Figure 3b. In this case, without any constraint, the success rate is above 98%. We observe an identical trend for the scenario when we apply the constraint. The average success rate goes down to 79%. In both cases, we observe that the proposed adversarial method could successfully design adversarial examples at a higher success rate. However, by restricting the flexibility of the adversary by limiting their access to different metadata at a different level, we see a dramatic reduction in the performance, justifying our hypothesis made in Section 3.2.

6. Conclusion and Future Work

This paper defines the adversarial attack problem in the MLBAC context where a trained ML model decides access. We customize objective functions by imposing additional accessibility constraint to reflect the access control context. We experiment with adversarial attack cases in the deep neural network-based MLBAC with two complex, large-scale systems. Also, we consider systems where underlying user/resource metadata contains both categorical and continuous data. We thoroughly evaluate the performance of MLBAC adversarial attack experimentation and demonstrate that it is possible to restrict an adversarial attack to some extent by introducing a security constraint.

In this work, we measure accessibility constraint using the correlation method. Exploring other methods to simulate such restrictions and accessibility would be interesting. Also, the adversarial attack and the proposed objective functions have dependencies on multiple parameters. The performance of the attack method has a strong connection with these parameter values, which we determine using a trial-and-error approach. A further investigation is needed to minimize dependencies and find their values more systematically. Besides, we consider

the white-box attack scenario, where we assume the adversary has access to the ML model in MLBAC. It would be interesting to explore a black-box scenario where the adversary cannot access the MLBAC model. Also, the proposed accessibility constraint exhibits its prospect of safeguarding against adversarial attacks, which may not be sufficient in some cases. A thorough investigation is needed to determine defense techniques in the access control context.

Acknowledgments

We would like to thank the CREST Center For Security And Privacy Enhanced Cloud Computing (C-SPECC) through the National Science Foundation (NSF) (Grant Award #1736209) and the NSF Division of Computer and Network Systems (CNS) (Grant Award #1553696) for their support and contributions to this research.

References

[1] R. S. Sandhu, et al., Role-based access control models, Computer (1996).
[2] V. C. Hu, D. Ferraiolo, et al., Guide to attribute based access control (abac) definition and considerations (draft), NIST special publication (2013).
[3] P. W. Fong, I. Siahaan, Relationship-based access control policies and their policy languages, in: The 16th ACM SACMAT, 2011, pp. 51–60.
[4] L. Karimi, M. Abdelhakim, J. Joshi, Adaptive abac policy learning: A reinforcement learning approach, arXiv (2021).
[5] N. Baracaldo, J. Joshi, An adaptive risk management and access control framework to mitigate insider threats, Computers & Security (2013).
[6] L. Cappelletti, et al., On the quality of classification models for inferring abac policies from access logs, in: IEEE Big Data, 2019.
[7] A. A. Jabal, E. Bertino, et al., Polisma-a framework for learning attribute-based access control policies, in: ESORICS, 2020.
[8] C. Cotrini, et al., Mining abac rules from sparse logs, in: Euro S&P, IEEE, 2018.
[9] L. Karimi, M. Aldairi, J. Joshi, M. Abdelhakim, An automatic attribute based access control policy extraction from access logs, IEEE TDSC (2021).
[10] M. Alohaly, H. Takabi, E. Blanco, A deep learning approach for extracting attributes of abac policies, in: ACM SACMAT, 2018.
[11] Q. Ni, J. Lobo, S. Calo, P. Rohatgi, E. Bertino, Automating role-based provisioning by learning from examples, in: ACM SACMAT, 2009.
[12] C.-C. Chang, I.-C. Lin, C.-T. Liao, An access control system with time-constraint using support vector machines., Int. J. Netw. Secur. 2 (2006) 150–159.
[13] A. Liu, X. Du, N. Wang, Efficient access control permission decision engine based on machine learning, Security & Communication Networks (2021).
[14] M. N. Nobi, et al., Toward deep learning based access control, in: ACM CODASPY, 2022.
[15] K. Srivastava, N. Shekokar, Machine learning based risk-adaptive access control system to identify genuineness of the requester, in: Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough, Springer, 2020, pp. 129–143.

[16] M. N. Nobi, M. Gupta, L. Praharaj, M. Abdelsalam, R. Krishnan, R. Sandhu, Machine learning in access control: A taxonomy and survey, arXiv (2022).
[17] V. Ballet, et al., Imperceptible adversarial attacks on tabular data, arXiv:1911.03274 (2019).
[18] I. J. Goodfellow, et al., Explaining and harnessing adversarial examples, arXiv (2014).
[19] H. Xu, et al., Adversarial attacks and defenses in images, graphs and text: A review, International Journal of Automation and Computing 17 (2020) 151–178.
[20] S. R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE transactions on systems, man, and cybernetics (1991).
[21] L. Breiman, Random forests, Machine learning (2001).
[22] C. Cortes, V. Vapnik, Support-vector networks, Machine learning (1995).
[23] J. Schmidhuber, Deep learning in neural networks: An overview, Neural networks (2015).
[24] K. He, et al., Deep residual learning for image recognition, in: IEEE CVPR, 2016.
[25] G. Huang, et al., Densely connected convolutional networks, in: IEEE CVPR, 2017.
[26] S. G. Finlayson, et al., Adversarial attacks on medical machine learning, Science (2019).
[27] F. Cartella, et al., Adversarial attacks for tabular data: Application to fraud detection and imbalanced data, arXiv (2021).
[28] N. Kumar, S. Vimal, K. Kayathwal, G. Dhama, Evolutionary adversarial attacks on payment systems, in: 2021 20th IEEE ICMLA, IEEE, 2021.
[29] Y. Mathov, E. Levy, Z. Katzir, A. Shabtai, Y. Elovici, Not all datasets are born equal: On heterogeneous tabular data and adversarial examples, Knowledge-Based Systems (2022).
[30] C. Szegedy, et al., Intriguing properties of neural networks, arXiv (2013).
[31] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, arXiv (2016).
[32] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: S&P, IEEE, 2017.
[33] R. S. S. Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, et al., Adversarial machine learning-industry perspectives, in: 2020 IEEE S&P Workshops (SPW), IEEE, 2020.
[34] J. Benesty, J. Chen, Y. Huang, I. Cohen, Pearson correlation coefficient, in: Noise reduction in speech processing, Springer, 2009, pp. 1–4.
[35] U. Amazon, Amazon access samples data set, 2011. URL: http://archive.ics.uci.edu/ml/datasets/Amazon+Access+Samples.
[36] K. Amazon, Amazon employee access challenge in kaggle, 2013. URL: https://www.kaggle.com/c/amazon-employee-access-challenge/.
[37] Z. Xu, S. D. Stoller, Mining attribute-based access control policies, TDSC (2014).
[38] J. T. Hancock, T. M. Khoshgoftaar, Survey on categorical data for neural networks, Journal of Big Data 7 (2020) 1–41.
[39] J. Yoon, et al., Anonymization through data synthesis using generative adversarial networks (ads-gan), Journal of biomedical and health informatics (2020).
[40] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: IEEE CVPR, 2017.
[41] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: ICML, PMLR, 2015.
[42] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: ICML, 2010.