# Environment Aware Deep Learning Based Access Control Model

Pankaj Chhetri
Department of Electrical and Computer Engineering
University of Texas at San Antonio
Texas, USA
chhetri.pankaj11@gmail.com

Smriti Bhatt
Department of Computer and Information Technology
Purdue University
West Lafayette, Indiana, USA
bhatt32@purdue.edu

Paras Bhatt
Department of Information Systems, Supply Chain and Analytics
University of Alabama at Huntsville
Alabama, USA
paras.bhatt@uah.edu

Mohammad Nur Nobi
FedEx Dataworks
Texas, USA
mohammad.nobi@FedEx.Com

James Benson
Institute for Cyber Security (ICS) and Department of Computer Science
University of Texas at San Antonio
Texas, USA
james.benson@utsa.edu

Ram Krishnan
Department of Electrical and Computer Engineering and Institute for Cyber Security (ICS)
University of Texas at San Antonio
Texas, USA
ram.krishnan@utsa.edu

## ABSTRACT

Recently Deep Learning based Access Control (DLBAC) model has been developed to reduce the burden of access control model engineering on a human administrator, while managing accurate access control state in large, complex, and dynamic systems. DLBAC utilizes neural networks for addressing access control requirements of a system based on user and resource metadata. However, in today's rapidly evolving, dynamic, and complex world with billions of connected users and devices, there are various environmental aspects in different application domains that affect access control rights and decisions. While Attribute-Based Access Control (ABAC) have captured environmental factors through environmental attributes, DLBAC still lacks the capabilities of capturing any environmental factors and its use in access control decision making. In this paper, we propose an environment aware deep learning based access control model (DLBAC-Env) which includes environmental metadata in addition to user and resource metadata. We present an Industrial Internet of Things (IIoT) use case to demonstrate the need for DLBAC-Env and show how different types of environmental aspects in a specific domain are necessary towards making dynamic and autonomous access control decisions. We enhance the DLBAC model and dataset to incorporate environmental metadata and then implement and evaluate our DLBAC-Env model. We also present a reference implementation of DLBAC-Env in an edge cloudlet using AWS Greengrass.

## CCS CONCEPTS

• **Security and privacy → Access control**; • **Computing methodologies → Machine learning**.


*SaT-CPS '24, June 21, 2024, Porto, Portugal*

ACM ISBN 979-8-4007-0555-7/24/06.
https://doi.org/10.1145/3643650.3659105

## KEYWORDS

Deep Learning Based Access Control, Internet of Things, Environmental Metadata, Autonomous Access Control, DLBAC-Env, AWS Greengrass

**ACM Reference Format:**
Pankaj Chhetri, Smriti Bhatt, Paras Bhatt, Mohammad Nur Nobi, James Benson, and Ram Krishnan. 2024. Environment Aware Deep Learning Based Access Control Model. In *Proceedings of the 2024 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (SaT-CPS '24), June 21, 2024, Porto, Portugal*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3643650.3659105

## 1 INTRODUCTION AND MOTIVATION

In the highly dynamic and rapidly evolving world of today, there is a need for an autonomous and dynamic access control approach that reduces the load of access control engineering and policy mining on a human administrator. Towards this aim, Nobi et al have developed a Deep Learning Based Access Control (DLBAC) model [23] leveraging the advances in Deep Learning (DL) that enables autonomous and dynamic access control in any domain. DLBAC reduces the burden of access control engineering on human administrators and addresses issues, such as over provisioning or under provisioning permissions in a system. Currently, DLBAC model is based on users' and resources' metadata and makes an access control decision (specific operation allowed or denied) based on the input given to the model. The input dataset include user and resource metadata, operations and some constraints, however, DLBAC still lacks the capability to capture the dynamic environmental aspects or factors in which users and resources exists and operate within a system.

In any application domain, environmental or contextual information are essential aspects of the access control mechanism being used in that domain/system. A user's or subject's access on requested objects is not just dependent on the user and object but also the environment in which the user and object reside as well as specific environmental conditions at the time access is being requested. Some of these environmental factors could be time, location, and state of system [29]. Attribute-Based Access Control

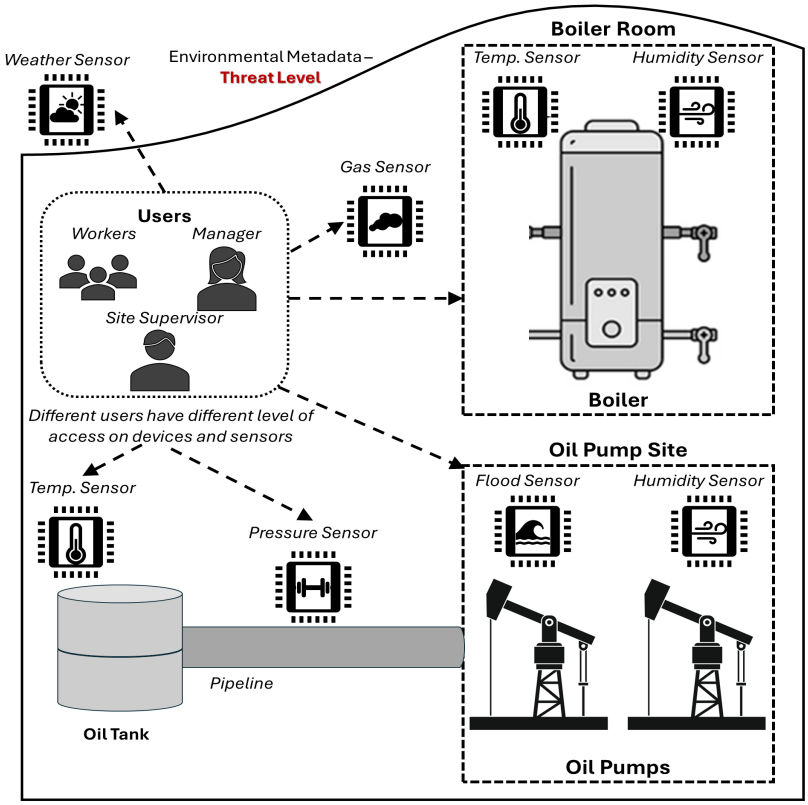


**Figure 1: A Industrial IoT Use Case with Different Types of Users, Sensors, Pumps, and Machinery**

(ABAC) [18, 19] model incorporates environmental aspects as environmental attributes which basically represent the environmental conditions or characteristics of the system and are used while defining access control policies. Environmental conditions or contextual information can be activated during an entire session in a system or can be based on changing contexts or conditions, such as temperature, ambient noise, and need to be checked at the time of access request [12]. It is essential for a deep learning based access control model to incorporate such environmental or contextual information to provide a comprehensive, dynamic and autonomous access control mechanism for a system. The neural network based access control model needs to capture dynamic environmental conditions through environmental metadata and include this information in the access control decision making process.

## 1.1 Motivation

We present an Industrial Internet of Things (IIoT) use case about a smart oil factory, as shown in Figure 1, to motivate the need for incorporating the environmental information in DLBAC model. IoT is highly dynamic domain which includes various components, such as users, devices, cloud and edge computing components, such as gateways, applications, and all of these working in continuously changing environments or contexts. For instance, as shown in Figure 1, there are various users (e.g., plant workers, managers, site supervisor, etc.), resources such as wireless sensors and factory equipment, e.g., oil pumps, tanks, temperature sensors, and the working environment in a smart factory, which is captured using temperature, humidity, pressure level, oxygen level sensors, and gas and weather sensors to monitor environment in the boiler room, pump room, and pump site. Since each of the sensors and smart devices are associated with critical operations within the factory, any abnormal changes in the environment can affect the access control decisions. Any abnormal behavior, either user or device level, or environmental factors, constitutes an elevated threat level in the factory. For example, if there is a weather advisory warning (e.g., flooding, wind storm), then access to certain resources will be restricted for some or all the users.

Another scenario that requires autonomous and on-demand changes in access control decisions based on environmental factors would be when the threat level of the factory is triggered based on the data collected by environmental sensors. For instance, the temperature sensor senses unusually high temperatures in the Boiler room, and the threat level is set high. There might be a fire situation in the factory, so if the threat level is high, then certain operations for some or all users would be denied (e.g., restrict all access to the boiler rooms, any oil pumps or tanks, and allow access to site supervisor only to restricted areas for checking the sensors or other devices). Using such environmental metadata, we can define access control rules for specific users and resources and automatically

grant or deny operations based on dynamic environmental changes, as shown in the IIoT use case.

To incorporate the environmental information in the DLBAC model, in this paper, we propose an environment aware DLBAC approach by adding environmental metadata, known as DLBAC-Env. These environmental metadata are independent of the user and resource metadata and rather capture different types of contextual aspects or environmental conditions related to various application domains including cloud computing, Internet of Things (IoT), or any other domain. Every domain has different types of users, resources, devices, and different types of contexts or environments. Access Control (AC) needs for different types of domains, such as IoT, which has various sub application domains within it including but not limited to Medical IoT, Vehicular IoT, and Industrial IoT, depends on users and resources or devices, and different types of environment these users and resources/devices are operating in, therefore, metadata for various entities including users, IoT devices, and specific environment metadata must be taken into consideration towards making dynamic access control decisions.

To develop our environment aware DLBAC model (DLBAC-Env), we extend the DLBAC model to include the environmental metadata by generating a dataset that includes environmental metadata, and then define a set of access control rules based on these environmental metadata, user metadata, and resource metadata. The updated dataset includes additional environmental metadata and authorization tuples based on the new rules defined. We generate synthetic dataset to run our DLBAC-Env model and implement a proof-of-concept of our model in an edge computing platform, AWS Greengrass [2], provided by AWS cloud [1]. We also conduct performance evaluation of our DLBAC-Env model which shows that the level of accuracy, explainability and generalizability are comparable to the original DLBAC model. The contributions of this research are outlined as follows.

- We propose an environment aware DLBAC model, named as DLBAC-Env, which can capture the environmental aspects of the system and utilize them in the access control process.
- We generate a set of rules for different scenarios and update the dataset with environmental metadata while including user and resource metadata. We also evaluate the performance of DLBAC-Env with different dataset and acceptable results were obtained.
- We propose a reference proof-of-concept implementation of our model in edge cloudlet using AWS Greengrass, an edge computing service provided by Amazon Web Services.

The rest of the paper is organized as follows. Section 2 defines the background and related work. Section 3 describes the environment aware DLBAC model and dataset generation. Section 4 presents the decision making process in DLBAC-Env, and Section 5 contains a discussion and evaluation of the model. Section 6 presents the details of the implementation in AWS Greengrass. Finally, we conclude the paper with future work in Section 7.

## 2 BACKGROUND AND RELATED WORK

### 2.1 Deep Learning for Access Control

The need for robust and fault tolerant security has given rise to an increasing need for automating and enhancing various access control approaches. Access control administration, policy generation, and deployment are critical tasks that, if automated, can enable effective management of users' access rights. The use of deep learning techniques to enable this automation can ensure the effectiveness of access control in a wide variety of contexts, from extracting useful attributes that enable the generation of access control policies to policy generation that take into consideration the dynamic nature of various application domains. Access control management relies on identification of robust attributes and research on using deep learning for identifying and extracting such attributes has risen in the past few years [4]. Several works have attempted to use methods such as natural language processing (NLP) to generate synthetic natural language access control policies and mine attributes [4, 5].

Prior studies have also attempted to generate policies from logs [22]. The authors proposed a model that generalizes the knowledge from the logs to yield a good set of candidates rules in a binary vector format. The key idea is to extract insights about different parameters that are important for obtaining a well-defined set of rules. Upon such extraction, the set of candidate rules from the binary vector format are transformed to the format acceptable by Xu-Stoller [30] and performance comparisons are conducted. An important aspect to consider while designing access control is the mobility of today's networks which leads to a dynamic environment that includes devices on-the-move, such as smart watches and specialized communication systems, such as air-ground coordinated communications [14]. In this respect, deep learning has emerged as an effective solution to implement access control not only on traditional networks, but also in non-traditional networks, such as wireless networks [24], WLANs [32], non-terrestrial networks [32], open radio access networks [10].

Another application of deep learning in access control has been in ensuring the security of IoT systems ([27]; [3]). IoT technologies play a critical role in upgrading various real-world smart applications that can improve the quality of life, however, the multidisciplinary nature of IoT systems and the different components utilized in their implementation present novel security issues. Thus, it is essential to implement security measures such as access control for securing IoT devices and their inherent vulnerabilities. As a result, existing security solutions need be improved to successfully safeguard the IoT environment. Also, deep learning is increasingly becoming a powerful method to add intelligence to IoT networks that have large-scale topology and complex radio conditions [21].

### 2.2 Environment Aware Access Control

The concept of environmental attributes was introduced with the ABAC model [29, 31] and has been ever since used in various versions of ABAC. A contextual attribute access control model was developed in [12]. The authors also discuss the dynamic nature of these environmental variables, such as temperature or ambient noise in the environment and how it can be captured using sensors in the environment and then be used in generating access control policies. Environmental attributes are a distinct type of attributes in ABAC which represents the environmental factors affecting the access control decisions, such as time of the day, location (user location or object location) [13, 18, 19, 25, 28]. These attributes are

mostly dynamic in nature but may also be static in some situation, i.e., might have a fixed value once defined in the system. They can also be associated with users and resources, but mostly are independent components of the surrounding environment where the access control is being deployed. ABAC, in its original form, or some modified form, has also been applied in the IoT domain by several researchers [6–9, 15, 17]. It is necessary to incorporate environmental attributes in the access control model and decision making process for dynamic application domains as IoT.

### 2.3 Deep Learning Based Access Control (DLBAC) Model

DLBAC [23] is an automated and dynamic access control mechanism based on advances in deep learning technology that can complement or potentially replace human administrators. It is different from classical access control approaches, such as Role Based Access Control (RBAC) [16, 26] and ABAC in that it uses a neural network in lieu of attribute and policy mining and engineering. It is also a more generalizable approach as access control rules in ABAC and RBAC focus on accurately capturing the access control state. This leads to poor generalization when making access control decisions on users and resources with attributes that are not explicitly seen during the mining process. A neural network is consistently better at making decisions on unseen data provided the training data follows the same format as the unseen data. It replaces access control policies with a neural network that makes access control decisions and trains this network using raw metadata instead of access control attributes. For example, metadata could include logs, employee details, access timestamps, network access profiles, etc. Examples of metadata for users are *night_shift, title,* and *department*, and resource metadata are *created_by, owner,* and *type*. Once the user and resource metadata are created, access control rules are generated based on these metadata, and once the rules are generated, authorization tuples cane created or updated with a new operation based on the rules for a user, resource, and operation combination.

## 3 ENVIRONMENT AWARE DLBAC MODEL

Here, we present an environment aware DLBAC model, known as DLBAC-Env, for encompassing different types of environmental or contextual information while making access control decisions in a system. In a dynamic and complex domain, there are various environmental aspects among which some are continuously changing and access of users/subjects/actors on specific objects/resources is dependent on such environmental factors. As discussed in earlier sections, with respect to the traditional access control models, environmental attributes are already captured in ABAC model. These environmental attributes are defined and mined by a human administrator and are used while defining access control policies in a system. The environmental attributes may be static, i.e., remain the same once defined in the system, or may change across time and will have different values based on current time of access request. Such dynamic environmental attributes can be captured using sensors (e.g., temperature, ambient noise) [12] or other system variables, e.g., location, time, security level of the system.

It is crucial to include these factors in the neural network based DLBAC model so that it can meet the access control requirements of highly dynamic current and futuristic application domains, such as a fully connected smart community or smart city, smart transportation where vehicles are constantly moving and operating in dynamic environmental scenarios, smart industries - manufacturing or factories, where environmental factors can determine if it is even safe to work in that environment. For example, if there is a gas leak detected in a smart factory, then all access to gas chamber and pumps must be immediately and autonomously restricted. To enable this, we propose and implement DLBAC-Env model which incorporates such environmental information in the original DLBAC model.

Traditional AC models have significant limitations to address the access control needs of dynamic application domains, such as IoT, where there are a large number of users, smart devices and tremendous amount of data being generated from these devices, and different types of environmental factors. Therefore, we need a neural network based approach to automate access control and minimize the time and resources spent on defining access control policies based on a specific traditional AC model, access control policy engineering, attribute engineering, policy review and updates. Administrators may not be properly skilled and may need additional training to design and implement access control policies for new emerging application domains, such as IoT. Generally, administrators are overburdened which leads to security holes in the system. Thus, the goal of DLBAC-Env is to address fine-grained, accurate, and autonomous access control for any dynamic and complex application domain, such as industrial IoT, medical IoT/smart healthcare scenario (for e.g., smart hospital, remote sensing and patient monitoring), and mission critical application domains, as well as other domains, university or organization, by incorporating environmental metadata along with user metadata and resources metadata in access control decision making process. DLBAC-Env captures environmental information in access control decision making process and provides this information to the neural network which enables it to make accurate and automatic access control decisions in a dynamic system with evolving entities and their contexts, and large amount of sensitive data.

Figure 2 depicts the decision making process in DLBAC-Env with new type of environmental metadata in addition to the user metadata and resource metadata. In the DLBAC-Env, the user metadata and resource metadata remains the same along with four different operations as in the original DLBAC model, but the model is augmented with environmental metadata to represent additional environmental information that is specific to the system and entities in that system. The wide range of environmental metadata that can be captured by various sensors or system information are used along with user and resource metadata in generating new access control rules or updating existing rules which in turn updates permissions on operations (i.e., grant or deny operations). The decision engine neural network gets user metadata (umeta), resource metadata (rmeta), and environmental metadata (emeta) in form of encoded metadata as input and gives output, i.e., operations (op1,op2,op3,or op4) that are allowed or denied based on the values of these metadata.

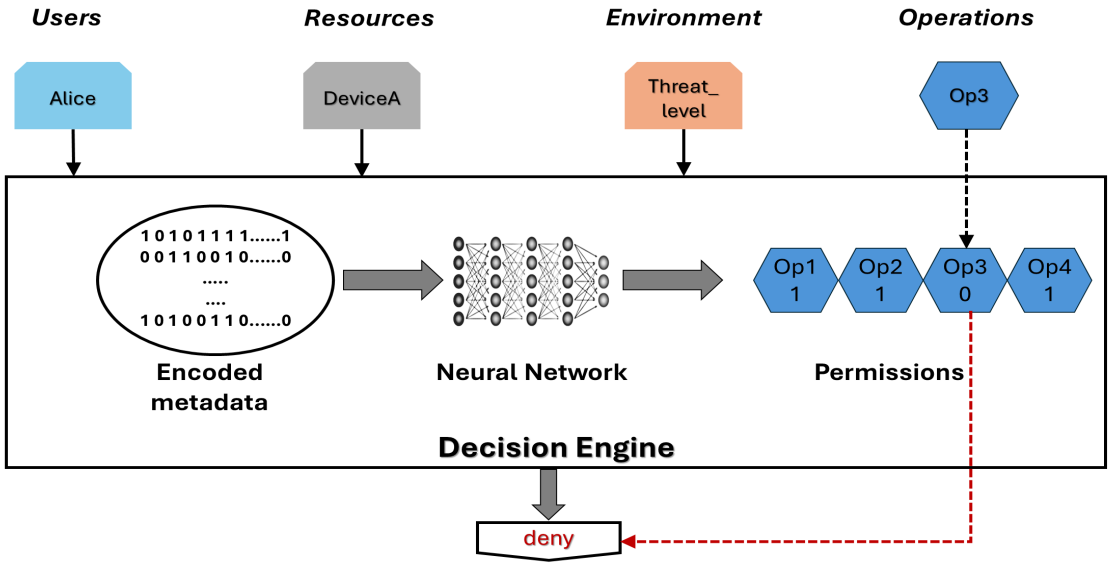


**Figure 2: Decision Making in DLBAC-Env**

## 3.1 Environmental Metadata

Now, we discuss further on environment metadata and in order to do that, we leverage the industrial IoT use case discussed in Section 1. Our environment aware DLBAC (DLBAC-Env) model can accurately address access control requirements in a continuously evolving ecosystem, such as IoT, by capturing the environmental aspects of the application domain in which the entities reside and operate within that domain. For example, in the smart factory use case shown in Figure 1, there are various sensors (e.g., temperature sensor, humidity sensor, pressure sensor, gas sensor, and weather sensor) which can capture environmental factors inside and outside of the factory. One of the environmental metadata (emeta) would be threat_level which can be triggered (0 or 1) based on sensor values and other surrounding conditions, such as weather conditions. So, an access control rule based on the emeta can be defined as: if *emeta0* = 1 (which means that threat_level is high), due to extreme weather condition or wind storm, then specific permissions can be allowed or denied for certain users. Another example can be when the temperature sensor senses extremely high values, then close out oil pumps and restrict all access for workers. These scenarios can be accurately captured with the help of environmental metadata, such as wind speed, fire alert, and threat level. Let us consider, threat level is an environmental metadata, *emeta0*, with values 1 and 0 as possible values, so if *emeta0* = 0, it signifies the absence of any risk or normal levels of operations, and if *emeta0*= 1, then risk/threat level is high. Now, access control rule can be defined as if *emeta0* = 1, then deny access to the boiler room, and any oil pumps or tanks.

To enable fine-grained access control, it is essential to capture the environmental information, especially in a large-scale, dynamic and rapidly evolving domain, such as IoT ecosystem. There are multiple contexts within IoT that also need to be considered while making access control decisions. There are various IoT application domains, e.g., smart health, smart home, vehicular IoT, and they have different types of environments that need to be considered to provide fine-grained and robust access control within these domains. For example, in a smart health scenario, when the details for a new doctor Bob are entered into the hospital system, his joining date could be user metadata and the location of his wearable IoT device would be environmental metadata. So, if there is an emergency situation and Bob is the only doctor on call at that specific time and location of wearable device shows them in hospital, then allow Bob to access patient's health records and take necessary actions. Similar to user and resource metadata, environmental metadata would be immediately available to DLBAC-Env model once the system is implemented and based on these environmental metadata, we can define specific access control rules based on all three types of metadata values and determine specific operations are allowed or denied.

In Figure 2, we illustrate how environmental metadata are used to make decisions in DLBAC-Env. In addition to using the available metadata for the user and resource, the neural network in DLBAC-Env is configured to use data on environmental metadata while making access control decisions. A non-IoT scenario where environmental metadata would impact access control decisions is a university scenario. In the university scenario, there can be two types of environment, one with normal operations, and second, when the University is under a cybersecurity attack. In normal environment, users (Administrative, faculty, staff) would have access on sensitive and non sensitive data/files and servers. However, when the university is under a cybersecurity attack, then, the access will be modified for the users, such as only admin and essential users will have access on sensitive files and servers. The environment metadata will represent normal operations or under cyber attack and access control decisions are dependent on these environmental factors.

## 3.2 Data Generation

Using a similar approach adopted by Nobi et al, we developed a rule based dataset using the additional environmental metadata needed for different application domains. We adopt Nobi et al's datasets used in the DLBAC paper and update it with environmental metadata, *emeta1*, *emeta2*, ... and then use these environmental metadata attributes to write new access control rules for the dataset in any domain. The rules are generated based on metadata for users, resources, environmental, and thereafter the dataset is updated either by creating new tuples and updating existing tuples in the dataset to ensure that all matching data points adhere to the specified rules. An example of such a rule is – if *emeta1* = 0, and *umeta3* = 84 and *rmeta5* = 132, then deny operation *op3*, i.e., op3 = 0, where *emeta1* is the *threat_level* environmental metadata and *op3* is denied.

The dataset rules are defined using the value of environmental metadata in a tuple format as $<UA; RA; EA; OP>$, where:UA is the user attribute, RA is the resource attribute, EA is the environmental attribute, and OP is the operation. The rule based syntax of the DLBAC-Env dataset contain a set of authorization tuples in the form of: $< uid|rid|eid|m^{u_1} : v_1, m^{u_2} : v_2, ..., m^{u_i} : v_i|m^{r_1} : v_1, m^{r_2} : v_2, ..., m^{r_j} : v_j|m^{e_1} : v_1, m^{e_2} : v_2, ..., m^{e_k} : v_k| < op1, op2, op3, op4 >>$. The uid, rid, eid in the tuple indicates the unique id of a user, a resource and environment. The next part gives the metadata values of all $i$ metadata of a user and $m^{u_1}$ indicates the first user metadata name (e.g., umeta0) whereas its value is indicated by $v_1$. The next part presents the metadata values of all $j$ metadata of a resource, and first resource metadata name (e.g., rmeta0) and its value are represented by $m^{r_1}$ and $v_1$ respectively. The subsequent part presents the metadata values of all $k$ metadata of a environmental attribute, and first environmental metadata name (e.g., emeta0) and its values are represented by $m^{e_1}$ and $v_1$ respectively. The last part is a binary sequence with a '0' meaning 'deny' and a '1' meaning 'grant' for the particular operation.

For example, *⟨3744 | 3889 | 31 84 7 31 43 105 48 43 | 43 48 122 31 33 105 48 6 | 0 0 |⟨1 0 1 1⟩⟩* is a sample authorization tuple in our dataset where 3784, 3889 are the user and resource unique number. The next set of eight numbers indicate the metadata values of a user, the following eight numbers represent resource's metadata values, thereafter the following two numbers represent the environment's metadata values, and the last four binary digits denote that the user has *op1*, *op3*, *op4* access to the resource.

These environmental metadata attributes depend on specific application domain and scenario that is being considered while designing access control system. The environmental information can be extracted or captured accordingly from the environment and/or system. For instance, environmental metadata attributes for medical IoT or smart health will be different than environmental metadata attributes for a smart connected IIoT use case. The environmental conditions can also be captured using sensors in the environment and the captured information can be directly provided to the deep learning model in the form of environmental metadata. An example of a environmental metadata for a smart oil factory is *threat_level* with possible values as high and low which indicates if the *threat_level* is high than certain operations can be restricted or certain operations can be enabled. For instance, if there is a malfunction in the factory, then the *threat_level* will be high, so it will lockdown the critical areas, such as pump site and boiler room, and any operation that tries to interact with the oil pump will be denied, on the other hand, in the facility, all exit doors will open and access should be allowed for all the users to get out of the facility. Similarly, for medical IoT scenario, an environmental metadata can be *emergency_level* which will be either True (1) or False (0) when the patient is in critical condition. If there is a medical emergency or accident, then allow read operation for on-call doctor on patient's health records and medical history, so that the doctor can treat the patient and take necessary action, even allow access for surgery, if needed.

## 3.3 Decision Making in DLBAC-Env

In DLBAC-Env, the Access Control Decision Engine (Decision Engine) component is responsible for receiving and authorizing access requests. In DLBAC-Env, we update the Decision Engine (DE), as shown in Figure 2, to take four inputs (user, resource, environment and operation). The DE obtains the corresponding binary representation of these inputs by encoding the user, resource and environment metadata that it retrieves from the internal databases. The neural network (ResNet) then receives this encoded input and predicts the access authorization for a matching request. Consequently, access data for each operation is output by the network. Finally, the requested access and the network's output are used by DE to determine the actual access authorization. For example, in Figure 2, Alice is requesting to perform op3 on deviceA. The predicted output of the neural network for op3 is 0, which indicates that Alice does not have access to deviceA. Therefore, the DE denies this request for Alice.

To illustrate the decision making in DLBAC-Env, let us consider the following real world scenario: in a smart home, there can be an environment metadata regarding risk level that represents the security risk to the smart home devices. In this scenario, when the home owner is not a technical expert and/or an elderly user, then the network devices (e.g., router) and IoT devices (smart thermostats, IP cameras) are likely to have default password and configurations. Therefore, these devices are operating in a high security risk environment and only the home owner must be allowed any access on these devices until the risk level is normal. Therefore, if the environmental metadata. i.e., risk level is high (emeta1 = 1) in the smart home, then allow all access to the home owner and restrict access to all the other users (e.g., guests, visitors, or even family members). The home owner then may request the technicians (for example, Bestbuy technician) to update the default password and configurations, and once it is done, the risk level will become normal (emeta1 = 0), thus allowing access to other users.

In DLBAC-Env, in addition to user and device attributes the model also considers the environment attributes (environmental metadata) while making access control decisions. The inclusion of the additional environment metadata enables DLBAC-Env to consider the dynamic changes in access control environments. Also, since a neural network forms the core of the DLBAC model, one of its main challenges is elucidating the reasoning and mechanisms behind the DLBAC's decisions. In other words, explainability is a core challenge for the model. In this regard, by incorporating the environment metadata, the DLBAC-Env model attempts to provide

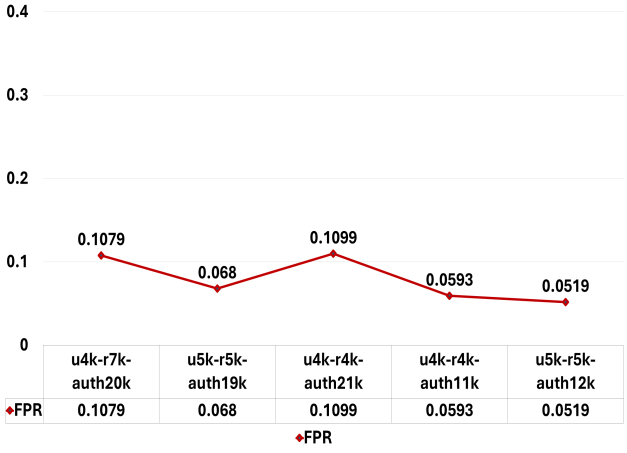


Figure 3a – FPR Scores

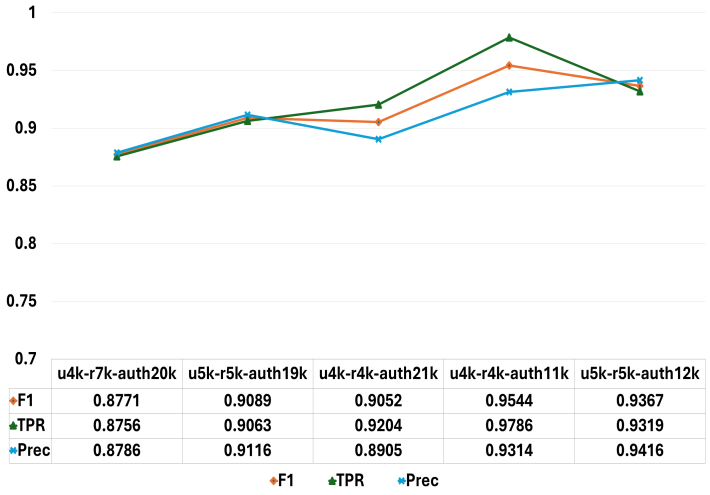


Figure 3b – F1, TPR, Precision Scores

**Figure 3: Performance Evaluation for DLBAC-Env**

justification for the decisions made on different authorization requests with reference to the environment metadata. For example, in Figure 2, Alice wants op3 access to deviceA. This request is sent to the DE and based on the neural network's output, the request is denied. Considering the smart factory use case, we have an access control rule: if threat level is high (emeta2=1) due to flood warning, then only the site supervisor can access the oil pump. Thus, Alice's, who is the manager, cannot do any operation on oil pump when there is a critical environmental threat. By incorporating the environment metadata, DLBAC-Env can take different environmental conditions or contextual information into consideration while making access control decisions.

## 4 EVALUATION AND RESULTS

We generated 5 synthetic datasets as per the process described in Section 3.2. We then evaluate the performance of the DLBAC-Env model using these datasets. Each dataset can be considered as an organization with its own set of distinct user, resource and environmental attributes. The datasets were divided into training (80 percent) and validation (20 percent) sets. Since the validation dataset is completely unknown during training, DLBAC-Env generalization should be accurately measured with such evaluation experiments.

For the experiment and evaluation, we implement DLBAC-Env model on the different synthetic datasets generated using the environment metadata rules. To conduct a comprehensive evaluation of DLBAC-Env, we calculate different performance metrics such as the F1 score, Precision, True Positive Rate (TPR), and False Positive Rate (FPR) obtained for each dataset. We take into account the accepted definitions of these performance metrics [11] [20]. In this regard, a high F1 score is associated with better generalization of the model. With respect to users, resources, and environment metadata that are not directly visible in the training process, the model is able to make more accurate decisions about access control. Additionally, higher TPR and Precision scores show how precisely and effectively the model can grant or deny access. Conversely, a lower FPR score signifies that the model is less likely to grant access to requests that should be denied based on the ground truth access control policy.

The DLBAC-Env model achieves consistently high performance in all the datasets. In this regard, Figure 3 illustrates the overall performance of DLBAC-Env on the different synthetic datasets with respect to F1 score, FPR, TPR, and Precision. Overall, the DLBAC-Env model achieved in general a high performance and the results demonstrate the effectiveness of using DLBAC-Env as an access control system in which can enable more accurate access control decision making, given the various different environmental considerations.

## 5 A REFERENCE IMPLEMENTATION IN EDGE CLOUDLET USING AWS GREENGRASS

In this section, we provide a proof-of-concept implementation of our DLBAC-Env model. The purpose of this is to validate the applicability of this model in more realistic IoT use case. Initially, for implementing the DLBAC-Env model it was trained using a local machine with an 80/20 training/testing split. The model resulted with an accuracy of 95.90 percent, 97.11 percent precision, 95.51 F1 score and 93.96 percent recall.

However, we propose an edge computing architecture for deploying our DLBAC-Env model in real-world to reduce latency and faster response time. Implementing DLBAC-Env at the edge also assists in enhancing model security and data privacy by providing local model computation. This is essential for IoT domains since the response time is crucial for making critical decisions in real-time. For example, in autonomous self driving cars, access control decision making needs to be done in real-time, rather than waiting for round-trip time for device to cloud and cloud to device communications.

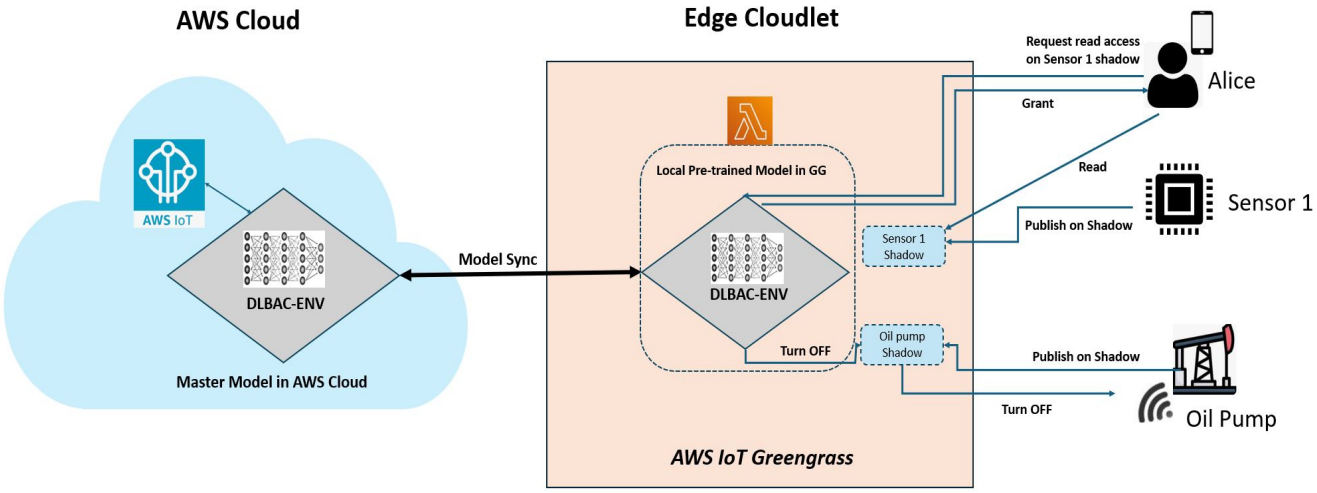


**Figure 4: AWS Greengrass Reference Implementation**

To achieve this, the DLBAC-Env pre-trained model converted to a tensorflow-lite model and was placed into an edge device as shown in Figure 4. According to this proposed architecture, the IoT devices are communicating with their shadows (virtual devices) in the Greengrass core where our pre-trained model will also be running on the GreenGrass Core. This core will then evaluate our DLBAC-Env model reducing latency, increasing response time, and can enable the IoT devices to have their requests granted or denied as needed.

The Amazon Web Services (AWS) IoT GreenGrass (GG) service is running on a dedicated Raspberry Pi 2 with a 900 MHz quad-core ARM Cortex-A7 CPU, 1 GB of RAM, and 32GB microSD card. The operating system is Raspbian 11 Bullseye, connected to a 1 Gbps network. The AWS Lambda is the service that runs inside of the GG Core orchestrating our code in tandem with a MQTT server that communicates with the IoT devices. Our model runs in a long-lived (pinned), uncontainerized environment, meaning that the code and deep learning model is preloaded and waiting for data input to begin processing data with the full resources of the Raspberry Pi to compute. This GG Lambda is running locally on the Raspberry Pi which is on same network as the devices and not in the cloud.

In our experiment, we ran 2117 trials in rapid succession by submitting one set of data parameters as input: "3434 3410 32 84 32 23 56 109 15 39 32 84 65 40 56 109 3 25 0 1". Once the input was provided, a clock was started, the model with said input was evaluated and results were returned. An example of the returning results are: "0 0 1 0". Once the output of the model was calculated, the timer was stopped and across the 2117 trials, it took on average 1.82 ms to run each model, with a minimum of 0.978 ms and a maximum of 3.18 ms with a standard deviation of 0.342 ms. The tensorflow-lite model takes approximately 48 ms to load, and is loaded prior to any inputs into the model.

Overall, our experiments demonstrate that it is feasible and applicable to run a pre-trained DLBAC-Env model in an edge cloudlet for an IoT use case scenario. The performance evaluation results are also promising and can be further improved with larger dataset and a more complete use case scenario with several IoT devices, users, and other entities where access requests are being granted or denied based on DLBAC-Env model.

## 6 CONCLUSION AND FUTURE WORK

There has been significant progress in both domains of deep learning and access control in recent years. We developed a environment aware DLBAC model, known as DLBAC-Env model. We introduced environmental metadata into the DLBAC model and tested it on 5 synthetic datasets which were generated using rules based on the environmental metadata along with user and resources metadata. We believe that with a change in environment, the DLBAC-Env model can be applied to different contexts in any domain, a university scenario, or a more dynamic scenario as IoT and cloud computing, based on different types of environmental metadata identified for each domain. Moreover, the DLBAC-Env model with environment metadata can make dynamic and autonomous access control decisions as it can adapt to meet any specific context in any system since User, Resource, Environment, Operation (UREO) format remains consistent across the different application domains and relevant contexts or environments. In the future work, we plan to create larger datasets and to run them for enhancing our model. We will test the DLBAC-Env model with new dataset including different environments for different use cases. We also plan to work towards enhancing security of the DLBAC-Env model and protect against adversarial attacks, such as model poisoning attacks from malicious entities in the system.

## ACKNOWLEDGMENTS

We would like to thank the National Science Foundation (NSF) for the CREST Grant Award 1736209 (CREST Center For Security and Privacy Enhanced Cloud Computing (C-SPECC)), NIST grant 70NANB23H133, and Cisco grant 87979263.

# REFERENCES


[1] 2024. Amazon Web Services (AWS) Cloud. https://aws.amazon.com/.

[2] 2024. AWS IoT Greengrass. https://aws.amazon.com/greengrass/.

[3] Mohammed Ali Al-Garadi, Amr Mohamed, Abdulla Khalid Al-Ali, Xiaojiang Du, Ihsan Ali, and Mohsen Guizani. 2020. A survey of machine and deep learning methods for internet of things (IoT) security. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 1646–1685.

[4] Manar Alohaly, Hassan Takabi, and Eduardo Blanco. 2018. A deep learning approach for extracting attributes of ABAC policies. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*. 137–148.

[5] Manar Alohaly, Hassan Takabi, and Eduardo Blanco. 2019. Automated extraction of attributes from natural language attribute-based access control (ABAC) policies. *Cybersecurity* 2, 1 (2019), 2.

[6] Safwa Ameer, James Benson, and Ravi Sandhu. 2022. An attribute-based approach toward a secured smart-home IoT access control and a comparison with a role-based approach. *Information* 13, 2 (2022), 60.

[7] Bruhadeshwar Bezawada, Kyle Haefner, and Indrakshi Ray. 2018. Securing home IoT environments with attribute-based access control. In *Proceedings of the Third ACM Workshop on Attribute-Based Access Control*. 43–53.

[8] Smriti Bhatt, Thanh Kim Pham, Maanak Gupta, James Benson, Jaehong Park, and Ravi Sandhu. 2021. Attribute-based access control for AWS internet of things and secure industries of the future. *IEEE Access* 9 (2021), 107200–107223.

[9] Smriti Bhatt and Ravi Sandhu. 2020. Abac-cc: Attribute-based access control and communication control for internet of things. In *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies*. 203–212.

[10] Yang Cao, Shao-Yu Lien, Ying-Chang Liang, and Kwang-Cheng Chen. 2021. Federated deep reinforcement learning for user access control in open radio access networks. In *ICC 2021-IEEE International Conference on Communications*. IEEE, 1–6.

[11] Carlos Cotrini, Thilo Weghorn, and David Basin. 2018. Mining ABAC rules from sparse logs. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 31–46.

[12] Michael J Covington and Manoj R Sastry. 2006. A contextual attribute-based access control model. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 1996–2006.

[13] Ni Dan, Shi Hua-Ji, Chen Yuan, and Guo Jia-Hu. 2012. Attribute based access control (ABAC)-based cross-domain access control in service-oriented architecture (SOA). In *2012 International Conference on Computer Science and Service System*. IEEE, 1405–1408.

[14] Ruijin Ding, Yadong Xu, Feifei Gao, and Xuemin Shen. 2021. Trajectory design and access control for air–ground coordinated communications system with multiagent deep reinforcement learning. *IEEE Internet of Things Journal* 9, 8 (2021), 5785–5798.

[15] Sheng Ding, Jin Cao, Chen Li, Kai Fan, and Hui Li. 2019. A novel attribute-based access control scheme using blockchain for IoT. *IEEE Access* 7 (2019), 38431–38441.

[16] David Ferraiolo, Janet Cugini, D Richard Kuhn, et al. 1995. Role-based access control (RBAC): Features and motivations. In *Proceedings of 11th annual computer security application conference*. 241–48.

[17] Maanak Gupta, Feras M Awaysheh, James Benson, Mamoun Alazab, Farhan Patwa, and Ravi Sandhu. 2020. An attribute-based access control for cloud enabled industrial smart vehicles. *IEEE Transactions on Industrial Informatics* 17, 6 (2020), 4288–4297.

[18] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. 2013. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication* 800, 162 (2013), 1–54.

[19] Xin Jin, Ram Krishnan, and Ravi Sandhu. 2012. A unified attribute-based access control model covering DAC, MAC and RBAC. In *Data and Applications Security and Privacy XXVI: 26th Annual IFIP WG 11.3 Conference, DBSec 2012, Paris, France, July 11-13, 2012. Proceedings 26*. Springer, 41–55.

[20] Aodi Liu, Xuehui Du, and Na Wang. 2021. Efficient access control permission decision engine based on machine learning. *Security and Communication Networks* 2021 (2021), 1–11.

[21] Qian Mao, Fei Hu, and Qi Hao. 2018. Deep learning for intelligent wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 2595–2621.

[22] Decebal Mocanu, Fatih Turkmen, Antonio Liotta, et al. 2015. Towards ABAC policy mining from logs with deep learning. In *Proceedings of the 18th International Multiconference, ser. Intelligent Systems*.

[23] Mohammad Nur Nobi, Ram Krishnan, Yufei Huang, Mehrnoosh Shakarami, and Ravi Sandhu. 2022. Toward deep learning based access control. In *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*. 143–154.

[24] Diego Pacheco-Paramo, Luis Tello-Oquendo, Vicent Pla, and Jorge Martinez-Bauset. 2019. Deep reinforcement learning mechanism for dynamic access control in wireless networks handling mMTC. *Ad Hoc Networks* 94 (2019), 101939.

[25] Qasim Mahmood Rajpoot, Christian Damsgaard Jensen, and Ram Krishnan. 2015. Attributes enhanced role-based access control model. In *Trust, Privacy and Security in Digital Business: 12th International Conference, TrustBus 2015, Valencia, Spain, September 1-2, 2015, Proceedings 12*. Springer, 3–17.

[26] Ravi S Sandhu. 1998. Role-based access control. In *Advances in computers*. Vol. 46. Elsevier, 237–286.

[27] Ahmed Sedik, Mohamed Hammad, Ahmed A Abd El-Latif, Ghada M El-Banby, Ashraf AM Khalaf, Fathi E Abd El-Samie, Abdullah M Iliyasu, et al. 2021. Deep learning modalities for biometric alteration detection in 5G networks-based secure smart cities. *IEEE Access* 9 (2021), 94780–94788.

[28] Daniel Servos and Sylvia L Osborn. 2017. Current research and open problems in attribute-based access control. *ACM Computing Surveys (CSUR)* 49, 4 (2017), 1–45.

[29] Hai-bo Shen and Fan Hong. 2006. An attribute-based access control model for web services. In *2006 Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)*. IEEE, 74–79.

[30] Zhongyuan Xu and Scott D Stoller. 2014. Mining attribute-based access control policies. *IEEE Transactions on Dependable and Secure Computing* 12, 5 (2014), 533–545.

[31] Eric Yuan and Jin Tong. 2005. Attributed based access control (ABAC) for web services. In *IEEE International Conference on Web Services (ICWS'05)*. IEEE.

[32] Yizhe Zhao, Jie Hu, Kun Yang, and Shuguang Cui. 2020. Deep reinforcement learning aided intelligent access control in energy harvesting based WLAN. *IEEE Transactions on Vehicular Technology* 69, 11 (2020), 14078–14082.