Adaptive Access Control Policies for IoT Deployments

1st Ashraf Alkhresheh
School of computing
Queen's University
Kingston, ON, Canada
khashraf@cs.queensu.ca

2nd Khalid Elgazzar
Department of Electrical, Computer and Software Engineering
University of Ontario Institute of Technology)
Oshawa, ON, Canada
Khalid.Elgazzar@uoit.ca

3rd Hossam S. Hassanein
School of computing
Queen's University
Kingston, ON, Canada
hossam@cs.queensu.ca

***Abstract*—In the era of the Internet of Things (IoT), it has become possible for a set of smart devices to collaborate autonomously and communicate seamlessly to achieve complex tasks that require a high degree of intelligence. Unlike traditional internet devices, a compromised IoT device can cause real-world damages. The severity of these damages increases dangerously in sensitive contexts especially when these devices are controlled by system insiders. Detecting abnormal access behaviors in such environments is quite challenging, due to frequent changes in the access contexts under which the IoT device can be accessed. In this paper, we propose an adaptive access control policy framework that dynamically refines the system access policies in response to changes in the device-to-device access behavior. We apply supervised machine learning to model and classify the device access behavior based on a real-life data set. We provide a use case scenario of a door locking system to validate our work. Results show that our framework provides improved security, dynamic adaptability and sufficient scalability to the target application domain.**

***Index Terms*—**

I. Introduction

Since the term was first coined by Kevin Ashton in 1999, IoT has received a considerable attention from the research community and industry [1]. IoT is a communication paradigm which connects *'things'* to the Internet and enables them to communicate and share data to achieve complex tasks. *Things* can be any object such as TVs, refrigerators, light controls, security systems, baby monitors and automobiles. The autonomous interaction among these things creates what is called *'smart spaces'* commonly known as smart homes, smart cars, smart cities; the essential constituents of IoT environments.

IoT opens the door for a wide range of smart applications and services such as traffic monitoring, remote medical care and management of supply chains. A recent study conducted by GrowthEnabler [2] shows that the global IoT market will grow from $157 billions to $457 between the years 2016 and 2020, and that the top three sub-sectors that dominate the market share are Smart Cities (26%), Industrial IoT (24%) and Connected Health (20%).

However, recent studies on IoT security show that a high percentage of connected IoT devices can be easily attacked due to many security vulnerabilities associated with these devices. For example, HP IoT [3] tested 10 of the most commonly used home security IoT devices and found that 70% of these devices are vulnerable to attacks. The omnipresence of IoT devices in individuals' surroundings allows for the collection of a very sensitive information and hence serious privacy violations. Therefore, it has become evident that security and privacy are major concerns that impede the widespread adoption of the IoT technology, and that access to IoT devices must be tightly controlled to prevent unauthorized accesses, preserve individuals' privacy and use the IoT technology to its fullest potential.

Existing access control models such RBAC [4], ABAC [5] and TBAC [6] rely on access control policies to regulate access to protected resources. The main issue in these access policies is that, oftentimes, they assign more access privileges than that actually needed by the requesting entity, which exposes systems' resources to insider attacks [7]. In addition, these access policies are manually specified and maintained, assuming operation in closed environments with infrequent changes in interaction conditions. The highly dynamic nature of IoT environments creates access contexts in which pre-defined access control policies can not meet the security and privacy objectives of the policy administrator. In IoT context, access control policies become obsolete quickly due to frequent changes in security and privacy requirements, which increases the risk of insider attacks and makes the policy management and maintenance a tedious and error-prone task. Therefore, there is a need for an adaptive access control mechanism that can react instantly to changes in IoT context and refine the access control policies at run time with minimal or no human intervention.

In this paper, we present a generic adaptive access control framework for IoT deployments. The framework uses machine learning techniques to detect abnormal behaviors of IoT devices accessing system's resources and refine the system's access control policies at run time to cope with behavioral changes. The framework introduces restrictions to access control policies based on behavioral features that significantly contribute to abnormal behaviors, and provides access policy refinements as recommendations to the policy administrator for final approval. With this, we aim to prevent users of IoT devices from abusing their access privileges or exploiting obsolete access control policies to gain unauthorized access,

978-1-7281-3129-0/20/$31.00 ©2020 IEEE

and simplify the management of access control policies. To the best of our knowledge, this work is the first to apply machine learning techniques to refine access control policies at run time based on a real-life data set.

II. Background and Related Work

Access control is a combination of three security concepts [8], [9]: (1) authentication that is the process by which a system verifies the identity of an entity (e.g., user, application, IoT device, etc.); (2) authorization determines whether an accessing entity has sufficient privileges to access system resources and what operations are allowed or prohibited for this specific entity on the resources of interest; (3) accountability ensures that the actions of an entity can be solely traced back to this entity. It guarantees that all operations carried out by individuals, systems or processes can be identified and that the trace to the entity and the operation is maintained.

The main objective of the access control is to enforce the system security and privacy requirements on protected services and resources [8]. The level of authorization an entity can be assigned is determined by evaluating its associated properties (e.g., identity, roles, proximity, access history) against a set of predefined access control policies.

Recently, detection and prevention of insider threats has attracted the interest of researchers in the IoT security field. In IoT context, insider threats come from current or former connected IoT devices which have or had access privileges on the protected resources, and the users in control of these devices, hereinafter referred to as (IoT users), intentionally abuse these access privileges in a manner that negatively affect the security and privacy objectives of the resource's administrator.

Traditional access control approaches such as [10]–[12] do not cope with insider threats and usually provide exessive access privileges to IoT users. In these approaches, researchers focus on improving access control policies by considering more factors (e.g., time, location and environmental conditions) in access decision-making to narrow down the access permissions assigned to IoT users on protected resources, the literature refers to these factors collectively as context. However, these approaches can still allow unauthorized access because they do not consider changes in the access behavior of IoT users while resources are in use.

A number of solutions have been proposed to approach the insider threats in IoT environments based on behavioral models and anomaly detection techniques. Meidan et al. [13] use deep autoencoders [14] to detect anomalous network traffics generated by compromised IoT devices. In their experiments, they trained one autoencoder to construct the normal traffic behavior for each IoT device. If the autoencoder fails to reconstruct the normal behavior on a new traffic sample generated by the corresponding device, it indicates that the observed behavior is anomalous and all subsequent traffic from this device is blocked. Hafeez et al. [15] use semi-supervised machine learning to detect malicious device-to-device communications in IoT. They use the Fuzzy C-Mean (FCM) algorithm to perform clustering of the network activities and determine the degree of maliciousness; thus handling different types of malicious traffic. Although these approaches do not focus on the adaptation of the access control policies, they give great insights for insider threat detection in IoT environments.

In this paper, we leverage the experience from the field of anomaly detection to build an adaptive and accurate policy refinement framework. The framework classifies the device access behaviors based on proactive measures and uses the knowledge it acquires from detected abnormal accesses to generate policy refinements at run time.

III. Adaptive Access Control Framework

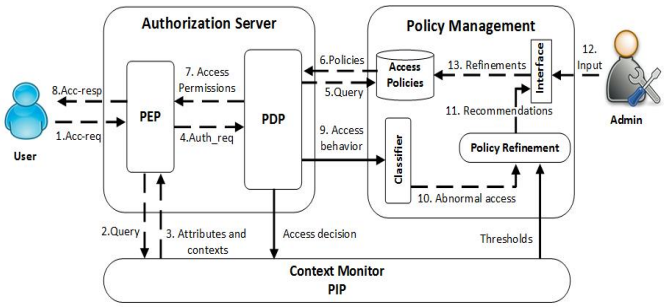

Fig. 1. Adaptive access control framework architecture.

Figure 1 depicts the architecture of the access control framework. The framework consists of:

1) A standard ABAC authorization server based on XACML access control infrastructure described in [16].
2) A policy management module which implements the access policy adaptation functionalities including the access behavior classifier and the policy refinement components.
3) A context monitor, which provides the access the contextual information required for the authorization server to evaluate access requests, and the policy refinement component to generate context-aware policy controls.

The framework encompasses four phases: (1) the offline training, during which the classifier models the access behaviors based on the historical access information (i.e., user access logs). (2) the access authorization phase, during which an access request is evaluated following the standard ABAC policy evaluation framework (`steps 1-8`); (3) the access behavior classification phase, during which the classifier component classifies ongoing access sessions and reports abnormal access behaviors to the policy refinement component (`steps 9-10`); (4) the policy refinement phase, in which the policy refinement component analyzes the anomalous access behaviors, recommends proper policy controls based on the proactively monitored contextual features (e.g., number of denials) and sends these recommendations to the policy administrator for final approval (`steps 11-13`).

The context monitor continuously feeds the policy refinement component with the contextual information required to

keep pace with changes in the normal access behavior. For example, if the rate of access denials increases significantly, this indicates that either the system is being attacked or the classifier model becomes obsolete due to emergent of new but normal access behaviors. In both cases, the policy refinement component notifies the policy administrator to restart the learning process and update the statistics of the classification model. In case of system attack, the policy administrator can simply ignore the alert. Otherwise, the policy administrator needs to label the emergent access behaviors as normal ones and feed the classifier with the new data to update.

A. Access behavior model

An access behavior represents how users utilize the system resources. We define the access behavior as the sequence of access requests submitted by the user to the authorization server within a predefined time window. Formally, we define the access sequence, denoted by *AB*, as follows:

$$AB = \{AR_{t-k}, AR_{t-k+1}, \ldots, AR_t\} \quad (1)$$

where AR_{t-k} is the first access request in the access sequence AB in a time window k. Each access request in the sequence is represented by the set of attributes that characterize the elements of the access request (i.e., user, resource and operation) and the context information that describes the environment in which the access request takes place (e.g., time and location). Formally, we define the access request, denoted by *AR*, as follows:

$$AR = \{AT_u, AT_r, AT_o, AT_c\} \quad (2)$$

where $AT_u = \{at_1^u, ..., at_x^u\}$ is the set of user attributes, $AT_r = \{at_1^r, ..., at_y^r\}$ is the resource attributes, $AT_o = \{at_1^o, ..., at_z^o\}$ is the operation attributes, and $AT_c = \{at_1^c, ..., at_w^c\}$ is the context information. An attribute is expressed as $at = name\ op\ value$, where op is a relational operator (e.g.,$=, \neq, <$) between the attribute name and a value from the range of possible values of this attribute.

B. Access behavior classification

The task of the classifier component is to classify the current access sequence into one of two behavioral classes: *normal* or *abnormal* given the historical access sequences submitted to the system (i.e., training data).

One approach to implement the classifier component is to use standard classification algorithms. In this regards, data science provides a plethora of classification algorithms such as Naive Bayes Classifier (NBC), Support Vector Machine (SVM) [16], and decision trees (DT) [17]. But near the top of the classifier hierarchy is the Random Forest classifier (RF) [18]. RF is an addition to the DT; it creates multiple decision trees and merges them together to obtain a more stable and accurate prediction. In general, the more trees in the forest, the more robust the prediction would be and thus higher accuracy. However, the complexity of the random forest grows significantly as the size of the data set increases because more decision trees need to be built in order to maintain stability and accuracy of the prediction. In addition, the RF classifier makes predictions based on the correlation between the input and output variables as well as among input variables within individual data points (i.e., one access request in our case), it does not consider correlations across data points nor their time order.

Another approach for the classification task is to use artificial neural networks such as Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs) [19]. These classes of neural networks provide a lot of flexibility and have proven usefulness and reliability in a wide range of problems. In particular, Recurrent Neural Networks (RNNs) [20] were designed to work with sequence prediction problems. Sequence prediction problems come in many forms including:

- **Many-to-One:** A sequence of multiple steps as input mapped to a class or quantity prediction.
- **One-to-Many:** An observation as input mapped to a sequence with multiple steps as an output.
- **Many-to-Many:** A sequence of multiple steps as input mapped to a sequence with multiple steps as output.

Our classification problem belongs to the first category. We want our classifier to take a sequence of multiple access requests as input and map it to one behavioral class as output. However, we want the classifier to predict the next access behavior every time a new access denial is recorded. Therefore, the length of the input sequence is variable in our case. This requires careful engineering of the RNN input layer.

C. Access policy refinement

Once the classifier model is built, we need to apply the knowledge it acquires during the training phase to refine the access control policies. Listing 1 shows the access policy refinement algorithm.

Listing 1: Access policy refinement algorithm.

```
   input : Abnormal Access Sequence AS
   output: Access Control Refinements ACR
1  begin
2      ACR ← {}   // initialize
3      for all ar ∈ AS
4          for all features ∈ ar
5              if P(ar.denied|ar.feature) ≥ threshold
6                  predicate ← ¬(features =
                     ar.feature.value)
7                  ACR ← ACR ∪ {predicate}
8              end
9          end
10     end
11     Return ACR
12 end
```

The refinement algorithm takes the abnormal access sequence reported by the classifier as input, and returns a set of access policy refinements in a form of negation predicates. The algorithm performs two steps:

1) It extracts the access attributes and contextual features (i.e., user, resource, operation, location and time) of each request in the abnormal access sequence, and for each individual feature, it calculates the likelihood probability

of access denial given that the feature is present in the access request, `lines 3-5`.

2) It generates new policy controls, `lines 6 and 7`, in a form of negation predicates. The negation predicate is defined on the feature(s) that contribute most to the abnormal behavior.

We calculate the conditional probability, `line 5`, as follow:

$$Pr(D|F) = \frac{Pr(F \cap D)}{Pr(F)} \tag{3}$$

Where:

$Pr(D)$ is the probability of access denial in the access sequence, defined as follow:

$$Pr(D) = \frac{\#\ of\ denied\ requests}{sequence\ length} \tag{4}$$

$Pr(F)$ is the probability of feature presence in the access sequence, defined as follow:

$$Pr(F) = \frac{\#\ of\ feature\ presences}{sequence\ length} \tag{5}$$

$Pr(F \cap D)$ is the probability of feature F to present in the a denied access in the sequence, defined as follow:

$$Pr(F \cap D) = \frac{\#of\ feature\ presences\ in\ denied\ requests}{sequence\ length} \tag{6}$$

The refinement $threshold$ is a predefined value that is set by the policy administrator. Setting the refinement threshold to small value increases the probability for the refinement component to pose more restrictions on the access policies. For example, If the probability of the feature (*subject:"Bob"*) to present in a denied access in an abnormal access sequence is greater than the refinement threshold, the refinement component will recommend to update all applicable policies (i.e., all policies whose subject is Bob) with the following predicate:

$$predicate \leftarrow \neg(user, "Bob")$$

Therefore, all access permissions that Bob has on the system resources will be revoked. In fact, a negation predicate can be defined to restrict access per user, operation, resource, attribute, context or any combination of them.

IV. Use case: Door Locking System (DLS)

We use the School of Computing door locking system (DLS) at Queen's university as our use case to demonstrate the utility and usability of the proposed adaptive access control framework. DLS controls a total of 30 doors located in a seven story building at the University main campus. Figure 2 depicts the architecture of DLS. The architecture encompasses the following entities: humans (i.e., the policy administrator and users); Authorization Server (AS) which makes the access decisions; Gateways (G), which are Raspberry PI [21] and Arduino [22] microcomputers. These microcomputers intercept the access requests that come from the door controllers and forward them with other information (e.g., time and location) to the AS. In addition, they also serve as backup for the AS during power outage should it occurs; Door Controllers (DC), which read users' information stored in their Ibutton microchips [23] (i.e., a form of access key), and lock/unlock the doors based on the system's access control policy. There are four types of doors in the school building: Exterior Doors (ED), Floor Doors (FD), Lab Doors(LD) and Office Room Doors (OD).

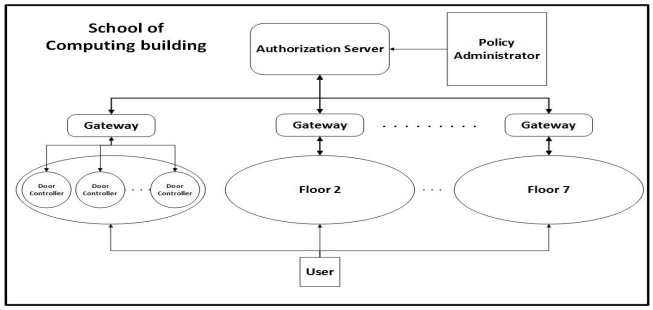

Fig. 2. DLS architecture.

A. DLS access control policy

DLS uses the XACML framework as its access control infrastructure. It employs a Policy Decision Point (PDP) to make access decisions (i.e., AS), a Policy Enforcement Point (PEP) to enforce the decision (i.e., door locks), and a Policy Information Point (PIP) to manage the collection of attributes and context information required for that decision (i.e., gateway). At the most basic level, a user makes an access request by touching his/her key to the Ibutton probe located on each door's frame. The Ibutton prob is connected to the gateway via 1-wire communication protocol [24]. The gateway timestamps the access request and sends it in a form of authorization request to the AS. The AS evaluates the request against preconfigured ABAC policies, makes the access decision, and returns the decision through the gateway to the door lock. The door lock in turns, permits or denies access. The access to the building, floors and rooms is determined by evaluating different attributes of the access request against the system's policies. These attributes are: the user attributes (e.g., role: faculty member); the door attributes (e.g.,lab and office door) and the contextual information (e.g., door time schedule and controller IP address).

B. Motivating access scenario

Assume Bob, a graduate student at the school of computing, goes to the school in the late evening hours on weekdays. Bob uses the main entrance to enter the school building, takes the elevator up to the sixth floor, opens the floor main entrance and heads towards his lab room. The following is an example of existing access control policies in DLS:

```
policy: policy 1 { deny-unless-permit
  rule rule1 ( permit
  target:
    equal(subject/role,"Grad student")
    && equal(subject/depart,"Computing")
    && equal(operation,"Unlock")
    && equal(resource,"LD")
    && equal(time,"Any")
```

```
    && equal(location,"6th floor"))}
```

Policy 1 allows graduate students of the school of computing a 24 hours access to all lab rooms in the sixth floor. Let us assume that while in the sixth floor, Bob attempts to access lab room 601. The following request will be submitted to the AS :

```
request:
    (subject/role,"grad student")
    (subject/depart,"computing")
    (operation,"Unlock")
    (resource,"room 601")
    (time,"20:00")
    (location, "601 door controller IP")
```

We can easily tell that the attributes and contexts of Bob's access request match the policy *target*, and satisfy all conditions of *policy 1*. Therefore, Bob's access request is granted. It worth mentioning that the access request is only evaluated against the access policies whose *targets* match the parameters of the access request, where the target is the set of user attributes, resource attributes and contextual information upon which conditions of access policies are defined.

Suppose that Bob, either accidentally or intentionally, attempts to access an OD without plausible reasons. Although, *policy 1* would deny Bob's access request, this scenario may indicate that Bob is either abusing his access privileges for personal benefits or Bob's Ibutton is being used by an attacker.

Existing access control approaches cannot prevent such kind of insider attacks. They lack the ability to detect abnormal access behaviors and adapt the access control policies to prevent these behaviors. Therefore, we equip our adaptive access control framework with a learning mechanism that models the normal access behaviors of the IoT devices (e.g., Ibutton), and use this knowledge to detect abnormal access behaviors and refine the access policies accordingly.

For example, if DLS monitors the access behavior based on the number of denied accesses, and observes that the majority of graduate students have been denied access to school doors no more than three times. Our system can use this information to refine access *policy 1* as follow:

```
policy: policy 2 { deny-unless-permit
  rule rule1 (permit
  target:
     equal (subject/role,"grad student")
     && equal (subject/depart,"computing")
     && equal (operation,"Unlock")
     && equal ( of access denials ,"<3"))}
```

In *policy 2*, the predicate in bold is the restriction our system imposes on *policy 1* based on the newly acquired knowledge. According to *policy 2*, graduate students whose access history (i.e., access logs) contains three access denials, will be denied access to all lab rooms, including those doors they were authorized to, until further review/auditing is conducted.

In the previous access scenario, we only looked into the number of access denials to refine the access policies regardless of the access context. However, it is applicable that the access control system applies more complex measures such as the correlation of abnormal access behaviors with sensitive contexts (e.g., weekends). In such contexts, our system can pose more restrictive conditions to the access policies, for instance, lowering the number of allowed access denials in weekend days, all applied by the policy administrator. Another interesting feature that the proposed system supports is the detection of abnormal access sequences. For example, if Bob chooses to change the doors he usually uses to get to the lab room, for instance, if Bob takes the elevator up to the fifth floor, then takes the stairs to get to the sixth floor. Our system can detect such change in access behavior and classify it as abnormal access sequence. Many other scenarios are also possible.

V. Dataset and Feature Selection

Our dataset consists of 180,000 access logs collected over the time period from December 2016 to April 2019. Each access log contains the following parameters: user ID (i.e., key ID), door ID, user attributes, door attributes, access time and access value. First, we selected the top five users based on the total number of accesses. This cuts down the data size to 13,296 access logs. All parameters in the access logs do not provide quantitative information. Therefore, we categorized these parameters and used the one hot encoder to perform "binarization" of each category and include it as a feature to train the classifiers.

Next, we defined 27 input variables as follows: 5 categories for the users, 3 categories for user attributes (i.e., faculty member, staff member, graduate student and undergraduate student), 2 categories for the day of access (i.e., weekday and weekend), 4 categories to represent the access hour (i.e., morning, afternoon, evening and night), 13 categories to represent the door ID which also represents the door location, and 1 category for the access value. We constructed an output label "Abnormal/Normal" behavior to the data set, where 1 represent abnormal and 0 represent normal. The default value to this column is 0 to all access logs. We set the value to 1 (abnormal behavior) in two cases: (1) if the user is denied access three times in a row with normal access context (i.e., weekdays and/or during morning, afternoon and evening times of day), (2) if the user is denied access two times in a row in sensitive context settings (i.e., weekends and/or during night time of the day). Otherwise, the output label is always set to 0 to denote normal access.

VI. Evaluation and Results

We use the Python libraries Scikit-learn [25] and Keras [26] to implement the RF and RNN classifiers. For the RNN approach, we use Long Short-Term Memory (LSTM) [27] network. LSTM overcomes the training problem associated with the RNN and in turn has been used in a wide range of applications that involve large data sets.

We chose the Receiver Operating Characteristic (ROC) curve as our evaluation metric. ROC does not depend on the class distribution, which makes it useful for evaluating classifiers predicting rare events such as abnormal access behavior in our case. In contrast, evaluating performance using accuracy

would favor classifiers that always predict a negative outcome (i.e. normal access behaviors) over rare positive outcome (i.e., abnormal access behaviors). To compare different classifiers, it is useful to summarize the performance of each classifier into a single measure. One common approach is to calculate the area under the ROC curve, which is abbreviated to AUC. AUC is equivalent to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance.

To evaluate the performance of the RF and LSTM classifiers. We conduct two experiments on two different data sizes: 6000 and 10000 access logs. For both experiments we performed the following steps: First, we split the data set into training and testing sets with the percentages 80 and 20, respectively. For the RF classifier, we fed the training data without modifications. For the LSTM classifier, however, we group the access logs between each subsequent abnormal behaviors into one access sequence. The resultant access sequences have different lengths, therefore, we calculate the length of the longest sequence and pad the short sequences with zeros. Next, we optimized the hyper-parameters for both models on the testing data. Then, we generate synthetic data that has different distribution (e.g., uniform) from that of the original data, and re-evaluate the two classifiers on the synthetic data. Figure 3 shows the performance of the two classifiers when trained and tested on small number of access logs. Approximately, the two classifiers score the same AUC over variable values of decision thresholds. The AUCs achieved show that both classifiers poorly distinguish one normal behavior from one abnormal behavior.

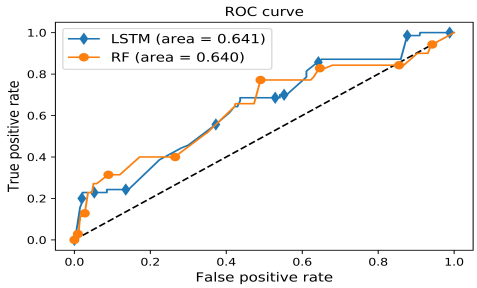

Fig. 3. The performance of LSTM VS RF, training data = 500 logs

Figure 4 shows the performance of the two classifiers when applied to data of different distribution. The RF outperforms the LSTM because there is less chance for short training data to contain long access sequences.

Figure 5 shows the performance of the two classifiers when trained and tested on relatively large number of access logs. Again, the two classifiers score comparable AUCs. However, The AUC results show that both classifiers can distinguish the two access behaviors with high accuracy. The LSTM outperforms the RF because of the increased chance for the LSTM classifier to learn from longer access sequences in larger training data. Therefore, we conclude that the LSTM classifier can scale better to IoT environments; it classifies the

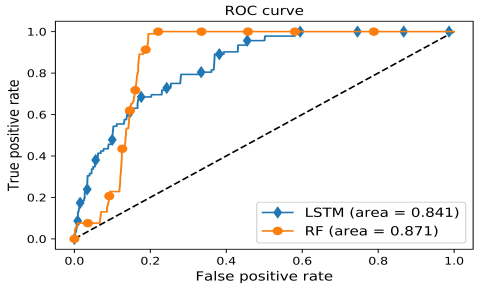

Fig. 4. The performance of LSTM VS RF, validation data = 150 logs.

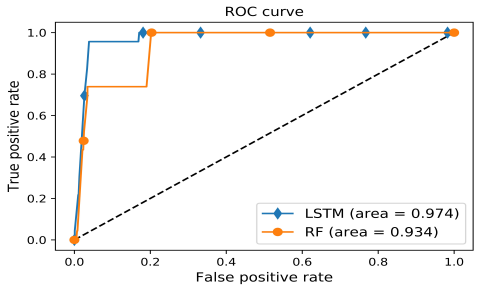

Fig. 5. The performance of LSTM VS RF, training data = 8000 logs.

access behaviors irrespective of the number of users, attributes, and access contexts. Figure 6 shows the performance of the two classifiers when applied to data of different distribution. To

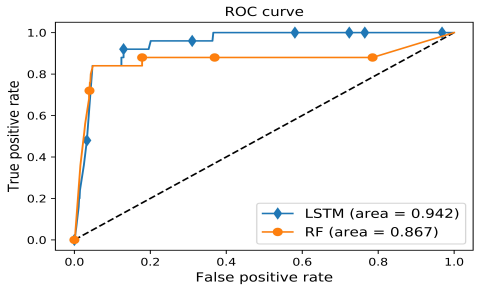

Fig. 6. The performance of LSTM VS RF, validation data = 3000 logs

evaluate the results of the policy refinement component, we run the LSTM classifier on the entire data set (i.e., 458 users). The classifier successfully reports 466 abnormal access sequences out of 495 actual ones. We set the refinement threshold to the minimum value (0.024), which is the maximum allowable number of access denials for a user (3 in our case) divided by the maximum sequence length (124 access requests). With this threshold we allow the policy refinement component to generate the largest number of possible recommendations. Table 1 summarizes the results of the policy refinement. We only present the type and number of distinct predicates generated by the policy refinement component. A total of 279 negation predicates are generated based on the userID only, 21 predicates for users on specific doors and 115 predicates

for users in specific time of the day distributed as follows: 26, 127, 163, 50 predicates for night, morning, afternoon and evening times respectively.

TABLE I
POLICY REFINEMENTS

Feature	userID	DoorID	UserID&DoorID	UserID &Time
predicates	279	301	21	115

For simpler analysis, we make a list of the predicates that associate two features in one predicate (i.e., userID&DoorID and userID &Time). We present the list to the policy administrator for analysis. The following is the feedback points from the policy administrator:

- 18 of (userID&DoorID) predicates are labeled "necessary". The rest of predicates require more analysis because the permissions of some users were escalated to access more doors, while others were replaced lost keys.
- 97 of UserID&Time predicates are labeled "unnecessary". Among the accepted 18 predicates, there are 15 that recommend restricting access on afternoon times. One possible justification is that there is an increased possibility that people go for launch around noon time, which increases the chances for accidental or intentional use of wrong doors.

VII. CONCLUSIONS

In this work, we introduced an adaptive access control policy framework for IoT deployments. The framework dynamically refines the access policies based on access behaviors of the IoT devices. We implemented and evaluated two machine learning classification approaches on a real life data set. The results show that both approaches perform well on small and unbalanced data sets, and provided accurate policy refinements at run time. However, LSTM shows improved performance on large data sets because of the increased chances for the LSTM classifier to learn from longer access sequences. For future work, we plan to build a user friendly interface to visualize the policy refinement recommendations. In addition, We plan to extend our framework with an anomaly detection component to detect changes in normal access behaviors (i.e., unseen behaviors) to maintain accurate and up-to-date access control policies.

REFERENCES

[1] K. Ashton, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[2] "Market Pulse Report,IOT - UK." [Online]. Available: https://growthenabler.com/reports/IOT.html

[3] "HP News - HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack." [Online]. Available: https://www8.hp.com/us/en/hp-news/press-release.html?id=1744676

[4] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[5] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, 2015.

[6] J.-B. Deng and F. Hong, "Task-based access control model," *Journal of software*, vol. 14, no. 1, pp. 76–82, 2003.

[7] "What is an Insider Attack? - Definition from Techopedia." [Online]. Available: http://www.techopedia.com/definition/26217/insider-attack

[8] M. Benantar, *Access control systems: security, identity management and trust models*. Springer Science & Business Media, 2005.

[9] N. R. Council, S. S. S. Committee *et al.*, *Computers at risk: safe computing in the information age*. National Academies Press, 1990.

[10] M. J. Moyer and M. Abamad, "Generalized role-based access control," in *21st International Conference on Distributed Computing Systems*, Apr. 2001, pp. 391–398.

[11] B. Anggorojati, P. N. Mahalle, N. R. Prasad, and R. Prasad, "Capability-based access control delegation model on the federated iot network," in *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on*. IEEE, 2012, pp. 604–608. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6398784

[12] P. N. Mahalle, B. Anggorojati, N. R. Prasad, R. Prasad *et al.*, "Identity authentication and capability based access control (iacac) for the internet of things," *Journal of Cyber Security and Mobility*, vol. 1, no. 4, pp. 309–348, 2013.

[13] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[14] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[15] I. Hafeez, A. Y. Ding, M. Antikainen, and S. Tarkoma, "Toward secure edge networks taming device to device (d2d) communication in iot," *arXiv preprint arXiv:1712.05958*, 2017.

[16] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[17] W. A. Belson, "Matching and prediction on the principle of biological classification," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 8, no. 2, pp. 65–75, 1959.

[18] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[19] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[20] "Raspberry Pi Documentation." [Online]. Available: https://www.raspberrypi.org/documentation/

[21] "Arduino - Home." [Online]. Available: https://www.arduino.cc/

[22] "What is iButton? - Definition from WhatIs.com." [Online]. Available: https://whatis.techtarget.com/definition/iButton

[23] S. Godik and T. Moses, "Oasis extensible access control markup language (xacml)," *OASIS Committee Secification cs-xacml-specification-1.0*, 2002.

[24] "1-Wire Communication with a Microchip PICmicro Microcontroller - Application Note - Maxim." [Online]. Available: https://www.maximintegrated.com/en/app-notes/index.mvp/id/2420

[25] "scikit-learn: machine learning in Python — scikit-learn 0.21.2 documentation." [Online]. Available: https://scikit-learn.org/stable/

[26] "Home - Keras Documentation." [Online]. Available: https://keras.io/

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.