Qualitative Intention-aware Attribute-based Access Control Policy Refinement

Shohei Mitani
NEC Corporation
Tokyo, Japan
s.mitani@nec.com

Jonghoon Kwon
ETH Zürich
Zürich, Switzerland
jong.kwon@inf.ethz.ch

Nakul Ghate
NEC Corporation
Tokyo, Japan
nak14@nec.com

Taniya Singh
NEC Corporation
Tokyo, Japan
taniya@nec.com

Hirofumi Ueda
NEC Corporation
Tokyo, Japan
h-ueda_cb@nec.com

Adrian Perrig
ETH Zürich
Zürich, Switzerland
adrian.perrig@inf.ethz.ch

ABSTRACT

Designing access control policies is often expensive and tedious due to the heterogeneous systems, services, and diverse user demands. Although ABAC policy and decision engine creation methods based on machine learning have been proposed, they cannot make good access decisions for applications and situations not envisioned by the decision-makers who provide training examples. It results in over- and under-permissiveness. In this paper, we propose a framework that refines pre-developed policies. It creates a decision engine that makes better decisions than those policies. Inspired by multiple criteria decision theory, our method uses the policy manager's qualitative intentions behind their judgments to guide access decisions so that more benefits are expected. In the evaluation, we prepare a coarse and relatively elaborate policy. We refine the coarse policy to obtain a decision engine that is compared for the similarity in access decisions with the elaborate policy using AUC as a measure. The results show that our method improves the coarse policy by a difference of 12–26% in AUC and outperforms the conventional machine learning methods by a difference of 3–11% in AUC.

CCS CONCEPTS

• **Security and privacy** → **Authorization**; **Access control**; • **Computing methodologies** → *Planning under uncertainty*.

KEYWORDS

ABAC policy, Machine learning, Decision theory, Actionable AI

ACM Reference Format:
Shohei Mitani, Jonghoon Kwon, Nakul Ghate, Taniya Singh, Hirofumi Ueda, and Adrian Perrig. 2023. Qualitative Intention-aware Attribute-based Access Control Policy Refinement. In *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies (SACMAT '23), June 7–9, 2023, Trento, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3589608.3593841

SACMAT '23, June 7–9, 2023, Trento, Italy

ACM ISBN 979-8-4007-0173-3/23/06…$15.00
https://doi.org/10.1145/3589608.3593841

1 INTRODUCTION

The fundamental idea of creating effective access control policies for information systems relies on the principle of least privilege. The principle requires allowing entities only a minimum amount of access privileges necessary to perform the designated tasks. The principle's underlying thought is balancing security risk and operational need [17]. For instance, access rights to read a customer list may be limited to a small number of employees who highly need it to reduce the risk of impact from misuse and information leakage. In this paper, the term "effective" describes the adequacy of access control policies. The term implies that making allow/deny decisions in the policy is expected to yield more business benefits and less loss associated with security incidents. The effective access control policy allows access if the operational need exceeds the risk, and vice versa [8]. Policy managers must thoughtfully evaluate the trade-offs between the security risk and the operational need for all possible accesses. Balancing these two competing goals with fine granularity minimizes over- and under-permissiveness in the policy. However, given the complicated modern information systems, evaluating the trade-offs and developing effective access control policies have become expensive, tedious, and error-prone.

In order to reduce the cost, state-of-the-art approaches employ Attribute-Based Access Control (ABAC) policy creation methods with Machine Learning (ML) techniques. Some of these approaches are policy extraction by rule mining. The ABAC policy is typically a set of access permission rules expressed with various attributes, e.g., document owner, read/write, user's department, and location. Each access is accompanied by a set of those attribute values, i.e., attribute vector. The ABAC policy generation approaches use pairs of attribute vectors and the corresponding "allow" or "deny" access control actions, i.e., decision examples. Xu and Stoller had proposed an ABAC policy mining from Access Control List (ACL) policies [22]. Each ABAC policy (e.g., "A user can read notifications from his/her department.") represents the purpose of access control, while the original ACL policy (e.g., "User Alice can read the resource B.") does not. Karimi et al. proposed an ABAC policy mining approach from access logs [13]. The access logs reflect initially deployed access control policies in the system (i.e., original policies). Another ML-based approach had been proposed to train a classifier (e.g., decision tree) to create an ABAC decision engine from access logs [4]. The decision engine uses the classifier to classify attribute vectors into allow/deny. The classifier works as ABAC policies but does not have

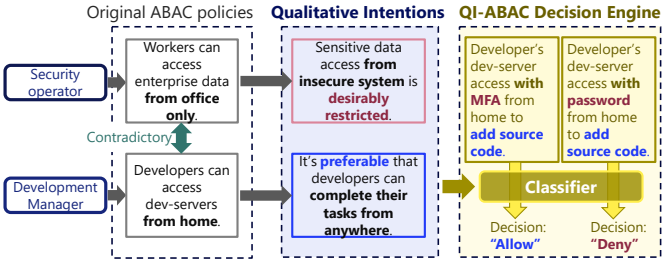


Figure 1: QI-ABAC policy refinement overview.

to be in the form of rules. In some of these log-based approaches, under the assumption that infrequent accesses are unnecessary, the ABAC policy is improved to prohibit low-need access.

Nonetheless, these log-based approaches share the following drawbacks: First, infrequent accesses are often ignored even if they are essential such as maintenance access. Second, there are many ignored criteria other than frequency for estimating the access need and risk. With these existing methods, appropriate decisions can only be made for the access purposes envisioned by the original policy designer. For example, if an ABAC policy is extracted from a policy intended only for development tasks, appropriate decisions regarding sales work cannot be made. These limitations are because the conventional methods learn decision patterns without consideration of balancing profit and loss.

Therefore, we propose Qualitative Intention-aware ABAC (QI-ABAC) policy refinement framework. In a nutshell, the proposed method systematically refines a given original policy (in the shapes of ACL policy, access logs, and even ABAC policy) to create an ABAC decision engine that makes more effective access decisions than the original policy. Our framework utilizes/infers the underlying qualitative intentions of the policy managers and designers. In our framework, the qualitative intentions systematically guide the direction of decisions, in addition to learning the decisions in original policies. We leverage ML models with monotonic constraints for this purpose. Inspired by Multiple Criteria Decision Making (MCDM) theory [7], QI-ABAC represents the process of evaluating the trade-offs between multiple criteria (i.e., the qualitative intentions) related to security and needs. Then the decision engine selects appropriate actions through the estimated values of granting accesses.

There can be various criteria. For instance, a ***quantitative*** criterion can be "allowing the resource owner's access contributes to a business benefit of $100." Obtaining such information would be difficult without in-depth investigation. Instead, QI-ABAC uses a ***qualitative*** criterion such as "the owner's access contributes more to the business benefit than access that is not." The qualitative criterion describes the order of good and bad from a given perspective (i.e., the owner's access is better than others). QI-ABAC utilizes policy designers' "qualitative intention," which has not been systematically handled due to its qualitative and ambiguous nature.

Fig. 1 provides an overview of how the proposed framework will work. The described system can have multiple goals: ensuring security and developing products. A security administrator designs an ABAC policy enforcing data access from the office only, whereas a development manager aims to permit developers access from their homes. They contradict each other. A cause of the contradiction is that the policy designers are shortsighted, i.e., the security administrator ignores the details of development tasks. Likewise, the development manager ignores insecure cases (i.e., limited scope in decisions). In common practice, they are aligned through "deny (or allow) override" [21], i.e., a policy aggregation rule that access denied (or allowed) by at least one of the policies is shortly denied (or allowed). Compared with the common practice, our framework utilizes the security administrator's qualitative intention to reduce sensitive data access from the insecure system. Another qualitative intention of the development manager is to let developers complete their tasks from as many locations as possible. QI-ABAC balances the risks and the needs, e.g., credential leakage, untrusted location, and development need. As a result, for instance, the security operator's qualitative intention is reflected in the Multi-Factor Authentication (MFA) requirement as Fig. 1.

We present three applications: to create an ABAC decision engine from access logs for reference (QI-ABAC$_0$), from pre-developed ABAC policies (QI-ABAC$_{\mathrm{I}}$), and through interactions with the operators (QI-ABAC$_{\mathrm{II}}$). In all the cases, the ABAC policy is represented as a classifier[4] instead of a set of rules. To show the performance, we use a real access log data set [9] and synthetic ABAC policies (sets of rules) in an academic paper [22]. In evaluating the QI-ABAC$_{\mathrm{I}}$ and QI-ABAC$_{\mathrm{II}}$, the synthetic policy is necessary. For the evaluation, we assume that a policy consisting of many rules is more effective because they have been designed in detail at a cost. Therefore, we pick a small number of rules from each synthetic ABAC policy as a poor baseline policy, ignoring other rules. We create a decision engine refining the small baseline policy. It is compared with the synthetic ABAC policy consisting of all the rules for reference. We interpret the higher the similarity of access decisions between the decision engine and the reference policy, the more effective the refinement. This paper makes the following contributions:

- Formulate a new problem of systematically refining policies and creating a decision engine so that adequate decisions can be made for tasks and situations that were not envisioned.
- Introduce three applications; 1) ACL policy/access logs to ABAC decision engine, 2) ABAC policy to ABAC decision engine, and 3) that with interacting with the operator.
- Propose the framework to achieve them with the concept of qualitative intentions. We propose an ML-based algorithm that represents an MCDM, and prove its efficacy.

2 RELATED WORK

This section overviews ML-based access control policy generation methods. Attribute-Based Access Control (ABAC) model defines permissions based on all available attributes of subject, object, operation, and environment [11]. Risk-Adaptive Access Control (RAdAC) explicitly estimates the risk. In attribute-based RAdAC models [12], the low-risk level is required to get permission (i.e., multilevel security). Some studies evaluate the trustworthiness (trust score) of entities [16] that can be an attribute. Several studies generate executable ABAC policies (e.g., XACML [21]) from those written in natural language [1]. Other approaches translate a high-level policy(e.g., service requirement) into lower-level enforceable policies (e.g., firewall rules) using the system models [2].

Table 1: Comparison of rule mining (e.g., Karimi), training a classifier (e.g., Cappelletti), and our approach (QI-ABAC).

Approach	From logs	From ABAC	Sparse sample	Fine-granular	Rational
Karimi [13]	✓	×	✓	×	×
Cappelletti [4]	✓	×	×	✓	×
QI-ABAC	✓	✓	✓	✓	✓

In these methods, an equivalent policy with the generated policy is necessary; in the shape of a document or high-level policy.

To overcome the issue, access log-based ABAC rule mining methods have been proposed. In [6, 20], the rules aim to grant the privileged operations exercised during a specified period, otherwise deny. Karimi [13] divides access logs (attribute and action pairs) into clusters to extract rules. These rule-mining methods rely on metrics to prevent the mined rule from being complex. Although the simplification helps to obtain adequate policies with few decision examples (i.e., sparse data), it may lead the policy to coarse granularity. Nobi [19] and Cappelletti [4] create classification models instead of rule sets. Those classifiers work as ABAC policies. The classifiers can perform complicated and fine-grained access decisions. Hence, we follow this classifier-based approach. Compared with these approaches, we aim to achieve rational decisions that choose actions to obtain large benefits. Tab. 1 shows the comparison with the rule mining [13] and with the classifier-based method [4].

3 PROBLEM DEFINITION

In this section, we explain the fundamental issues of the ABAC policy design/generation. Later, we identify the challenges.

3.1 ABAC Policy Creation Problem

We first formulate a general ABAC policy creation process. Let A be a set of all attributes. For each attribute, $a_n \in A$, the value x_n of the attribute varies in a set V_n. The set of all possible "attribute:value" combinations ξ is written with the cartesian product, $\xi = V_1 \times V_2 \times \cdots V_N$, where N is the number of all attributes. $\mathbf{x} = (x_1, x_2, \cdots, x_N) \in \xi$ describes a particular attribute vector. Not all the combinations are valid, e.g., if the value of an attribute "User position" is not "student", another attribute "Courses the student takes" may be empty. The valid privilege universe $\xi' \subset \xi$ is the set of attribute vectors that can appear [20]. ABAC policy divides the valid privilege universe into permission and prohibition. Hence, the ABAC policy is defined as a map $f : \xi' \to Y$ where Y is the set of action, i.e., $Y = \{$"allow", "deny"$\}$. The policy designer evaluates risk and needs for attribute vectors in ξ', deciding desirable actions.

3.2 Formulation of Decision Scope

We explain the cause of poor policy creation. We define the ideal ABAC policy as a hypothetical function $f^* : \xi' \to Y$. Assume an ideal decision maker who can accurately predict gains and losses from granting/denying attribute vectors. We assume that a real policy approximates the ideal policy in a limited scope: a subset of the privilege universe. We formulate the practical policy $\mathcal{P}$ as a function $f^{\mathcal{P}} : \xi' \to Y$ and a subset (scope of the policy) $S^{\mathcal{P}} \subset \xi'$ on which attribute vectors are well-evaluated to decide the optimal actions. Therefore, the policy $\mathcal{P} = (f^{\mathcal{P}}, S^{\mathcal{P}})$ satisfies the following.

$$\mathbf{x} \in S^{\mathcal{P}} \Longrightarrow f^{\mathcal{P}}(\mathbf{x}) = f^*(\mathbf{x}) \tag{1}$$

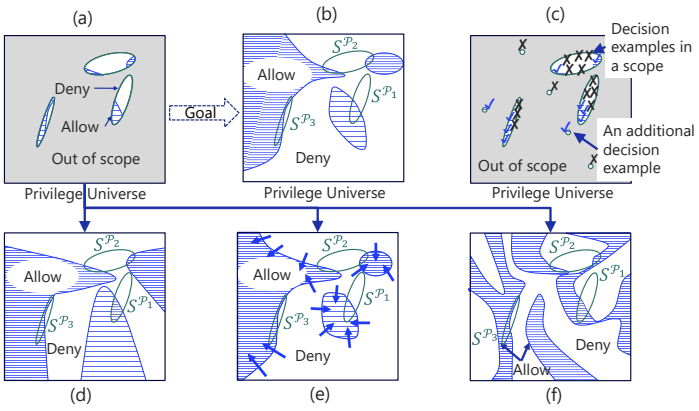


Figure 2: Conceptual diagram of "Scope" (a, b, c) and of policy design by humans and by ML-based methods (d, e, f).

Therefore, we assume the volume of each scope indicates the decision-making cost of the ABAC policy design. The scopes represent expertise areas of multiple policy designers. Here, we focus primarily on the up-front cost of evaluating benefits to allow accesses (i.e., decision-making cost) rather than describing policies (i.e., the complexity of rules). Fig. 2 shows the relationship between (a) Three scopes and action maps in three original policies, (b) the schematic of an ideal action map, and (c) example actions following the ideal ABAC policy (decision examples). This figure describes sparse sample ("an additional decision example") and biased sample ("decision examples in a scope") problems. The sparse sample describes a case where the desired action is known for few attribute vectors. The biased sample is a case where no desirable action in a task (e.g., development task) is known. The original policy might be designed for sales tasks. The following figures show action maps of various ABAC policy generation; (d) a naive rule aggregation, (e) QI-ABAC decisions guided by the qualitative intentions (blue arrows), and (f) general ML-based decisions. The last one has fine granularity but often outputs irrational decisions. QI-ABAC (e) approaches the ideal policy by allowing fine granularity and rationality.

3.3 Challenges

To obtain a more effective ABAC policy, we consider rationally extending the scope of the policy as the main challenge. The term "scope" describes attribute vectors within which adequate actions are defined to maximize expected business profit. As explored earlier, many attribute vectors remain out of scope due to sparse and biased samples. The most effective ideal policy can be achieved if the scope is extended to all the privilege universe. However, extending the scope without requiring extra decision examples is challenging. The sparse sample is interpolation problem from the similarity of access, while the biased sample is the nontrivial problem of extrapolation toward unforseen business scenes. To our best knowledge, this paper is the first work that deals with both problems.

4 OVERVIEW

This section provides an overview of the QI-ABAC policy refinement framework. We present an applications overview. Then, we describe our approach with prominent ideas.

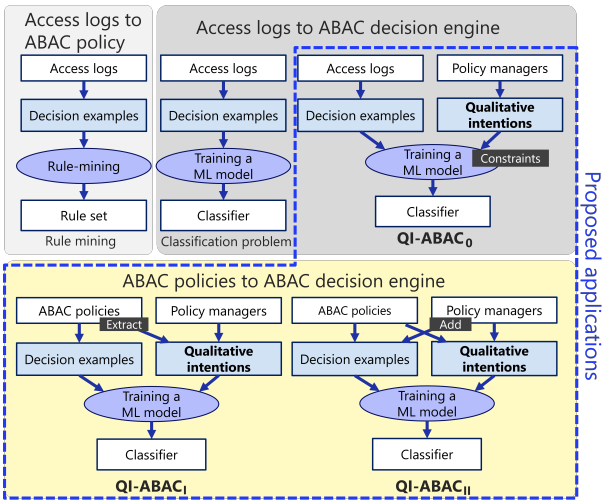


Figure 3: Comparison of existing approaches and our proposed QI-ABAC applications.

4.1 Applications overview

We first show QI-ABAC applications in Fig. 3. The upper side represents the applications to generate ABAC policies or decision engines from access logs or ACL policies. Two existing approaches are shown; rule mining (e.g., [13]) and classification (e.g., [4]). QI-ABAC$_0$ generates a classifier. Our framework provides an interface to reflect the policy managers' qualitative intentions as guidance information. The lower side in Fig. 3 shows our novel applications. QI-ABAC$_{\mathrm{I}}$ (on the left) samples decision examples from the given original ABAC policies, extracting qualitative intentions. Then, a classifier is trained to improve the decisions. In QI-ABAC$_{\mathrm{II}}$ (on the right), policy managers are asked to give a desirable action for an attribute vector (i.e., "Query and Response"). Their decisions are additionally learned. To clarify our focus, this paper does not discuss optimizing the choice of queries (i.e., active learning such as [10]).

4.2 Proposed Approach

QI-ABAC refines access decisions utilizing the qualitative intentions. Qualitative intentions are "the preference to grant access" regarding certain aspects of access regarding risk or need (e.g., Aspect 1: whether the resource is relevant to the user's organizational work and, Aspect 2: the user's job position). In order to reflect them in access control, we impose the following constraint as the minimum constraint representing a rational decision. "***Access that is preferable to grant in all aspects is, overall, more valuable to grant.***" The monotonicity is the mathematical representation of this relationship between the preferences and the value to allow.

We represent the preferences as a feature vector (i.e., intention vector). For example, Aspect 1-based preference is computed at the following five levels: L1) resources of the user's project, L2) team resources, L3) departmental resources, L4) organizational resources, and L5) other resources. We assign each level with a preference value: {1, 0.5, 0, -0.5, and -1} to represent the order of preference (L1 is preferable to L2, L2 is preferable to L3,...). In this case, two attributes "the user's department" and "the resource owner's department"

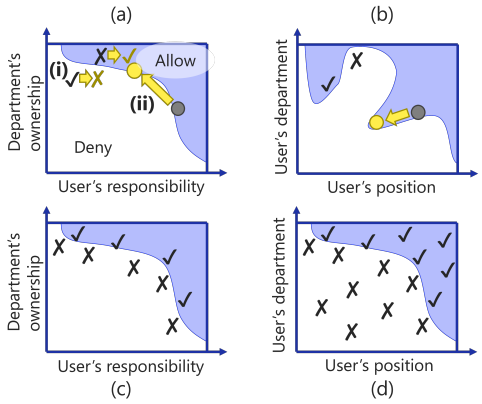


Figure 4: Effects of the qualitative intention-awareness (a, c) compared to using general attributes for decisions (b, d).

relate to the preference. We call the two attributes "associated attributes" with this qualitative intention. Similarly, we assign four levels of preference value: {1, 0.333, -0.333, and -1}, depending on whether the user's position (Aspect 2) is L1) managerial, L2) regular, L3) cooperative, or L4) other. Suppose the access user is a member of the same team as the resource owner and is a cooperative employee. The intention vector based on these two aspects, 1 and 2, is $(z_1, z_2) = (0.5, -0.333)$. We then train a function that computes the total permissive value from the feature vector. Pairs of an attribute vector and an action (allow/deny) must be given (i.e., decision examples).

Fig. 4 shows; (a) an action map for intention vectors (z_1, z_2) following two qualitative intentions above; the department's ownership (Aspect 1) and the user's responsibility (Aspect 2). The boundary between "allow" and "deny" is determined through the training. The following advantages of our proposal stem from the monotonicity: the total permissive value increases from the lower left to the upper right and never decreases.

(i) Correction of irrational actions from allow (✓) to deny (×) and vice versa. For example, where users with greater responsibility access more work-relevant resources, permissions should be broader (if other conditions are the same).
(ii) Proposals of alternative requirements. For instance, if a user has an insufficient responsible position, they can instead access other resources that are more work-relevant, i.e., with a greater ownerships. As another example, if a user does not have an MFA device required, security questions are used instead.

(c) shows the advantage from the viewpoint of ML. "✓" ("allow") and "×" ("deny") are decision examples. Since the total permissive value increases monotonically towards the top right, a few decision examples determines the decision boundary. It indicates that the decision scopes are rationally extended to any attribute vectors. (b) and (d) show there is no merit without the monotonicity, resulting in unexpected decisions, such as allowing access irrelevant to the department's business from a non-responsible position.

Original policies are imperfect. Therefore, our decision engine does not thoroughly learn policies but only in-scope attribute vectors. In addition, we define strict control of decisions. Our method estimates the value (between 0 and 1) to allow access. Access is allowed if the value exceeds a threshold (e.g., 0.5). The threshold is adjusted to control the strictness depending on the applied system.

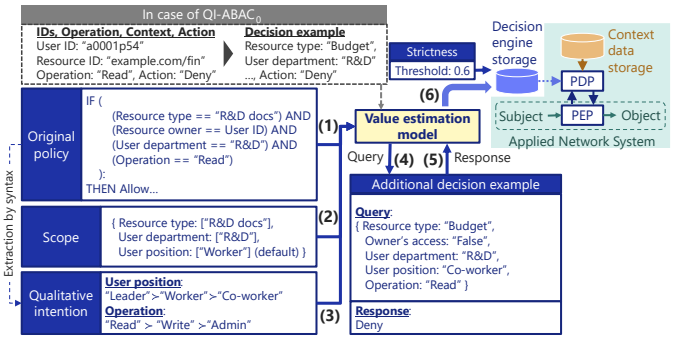


Figure 5: QI-ABAC policy refinement framework.

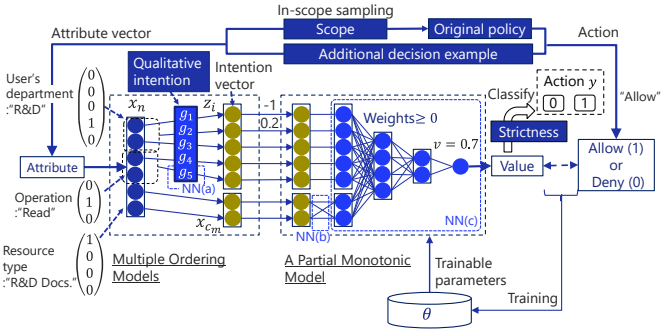


Figure 6: Machine learning-based value estimation model.

Examples of qualitative intention. A qualitative intention is defined as the preference order. Three examples follow. "≥" is an ordering operator. Note that intentions A and B are compatible.

Intention A :

{User ID == Resource owner} ≥ {NOT User ID == Resource owner}

Intention B :

{(NOT User ID == Resource owner) ∧ (Work flow : "Review")}

≥ {(User ID == Resource owner) ∧ (Work flow : "Review")}

Intention C : {Trust score : t} ≥ {Trust score : t'} for $t \geq t'$ (2)

Foundations as a decision-making problem. In MCDM [7], rationality is the choice of action to yield more benefits based on multiple criteria. Our methodology follows this basis. Assume that a designer created a policy to maximize benefits. The objective of our ML is to back-estimate and gather various operators' value functions. Our method should have the capability to express arbitrary policies. For instance, decision boundaries in Fig. 4 can vary, conditioned by services or applications. Such variables (e.g., service type) are not subject to monotonic constraints. Therefore, we use "partial" monotonic functions as the value function (Equation. 4, 5).

5 METHODOLOGY

We present our QI-ABAC framework. First, we overview the framework. Consequently, we disclose three options to define qualitative intentions. Lastly, ML-based value estimation model are shown.

5.1 Framework Overview

Fig. 5 shows our framework and the workflow. It outputs a classifier. In a typical case, Policy Decision Point (PDP) stores and utilizes the classifier. The workflow is as follows: (1) Define original ABAC policies designed by, e.g., security administrators, project managers, resource owners, or existing ABAC policy mining methods. It is a standard rule set, not unique to our proposal. (2) Define the scope(s) of those policies. For instance, a security operator may assume a general worker as the default assumption of the user. It is expressed in a scope. Operators should define and confirm the scope. (3) Define or extract the qualitative intentions. See also §4.2. (4) The system makes querying attribute vectors randomly to its operator, and (5) receives desirable actions as the response. Finally, (6) the training is run. A classifier is created as an ABAC decision engine according to the strictness. In QI-ABAC$_I$, steps (4) and (5) are skipped. In QI-ABAC$_0$, steps (1), (2), (4), and (5) are skipped, and decision examples converted into attribute vectors are used for the training.

5.2 Extraction of qualitative intention

Our framework has three strategies to define qualitative intentions: 1. Manual definition, 2. extraction from ABAC policy syntax, and 3. extraction during the training, i.e., unknown qualitative intention.

Manual definition. Qualitative intentions are defined based on the organization's high-level security policy, generic security guidelines, or expert intuition. For example, even if guidelines state that "multi-factor authentication should be required to access sensitive information," this alone does not lead to an access control policy. The policy depends on which resources are considered sensitive. Nonetheless, the guideline indicates a qualitative intention, {Auth. method: "MFA"} ≥ {Auth. method: "password"}. This example shows that defining qualitative intentions is effortless.

Extraction from ABAC policy syntax. Our fundamental insight is that each conditional statement represents the policy designer's intention. For instance, the original policies may have a condition "IF (User ID == Resource owner ID), and..., then allow the access." The situation that the user is the resource owner may contribute to the permit decision. Generally, we use the following two criteria: 1) statements that only or frequently appear with "allow" decision is qualitatively intended, and vice versa, e.g., Intention A in § 4.2, 2) a conflicting statement with the above is also qualitatively intended under other specific conditions, e.g., Intention B in § 4.2.

Unknown qualitative intention. This option extracts the qualitative intentions during the training. Assume several attributes (e.g., the user's project team and resource owner's department). A policy manager may know those attributes contributes to access decisions but does not know the concrete relationships between project teams and departments. One of our models (QI-ABAC-DNN in § 5.3) extracts the relationship by inferring the preference order. A policy manager should only identify the associated attributes.

5.3 ML-based value estimation model

As described in Fig. 6, the ML model estimates value (0–1) to permit a given attribute vector. It is optimized with the given decision examples sampled in scopes and the additional decision examples. We disclose the model, the preprocessing, and two implementations.

Architecture. Our proposed model has two major parts: multiple ordering and partial monotonic models. In the first part, the model converts the attribute vector to an intention vector. It computes a preference between -1 and 1 according to ordered conditions that represent a qualitative intention. See §4.2. The preference value for i^{th} qualitative intention (z_i) with its associated attributes (e.g., $\mathbf{x}_i = (x_3, x_7)$) is calculated as below, when the attributes $\mathbf{x}$ matches d_i^{th} condition in D_i number of ordered conditions.

$$z_i \leftarrow g_i(\mathbf{x}_i) = 1 - \frac{2(d_i - 1)}{D_i - 1} \tag{3}$$

They are input of the second part; partial monotonic function h along with conditional attributes ($x_{c_1}, x_{c_2}, \cdots$), e.g., service type.

$$v = h(z_1, z_2, \cdots, z_r, x_{c_1}, x_{c_2}, \cdots) \tag{4}$$

r is the number of all the qualitative intentions. The partial monotonic constraint of ML model h is defined with Equation (5) [14, 23].

$$h(z_1, \cdots, z_i \cdots, z_r, x_{c_1}, \cdots) \geq h(z_1, \cdots, z_i', \cdots, z_r, x_{c_1}, \cdots) \quad \text{if } z_i \geq z_i' \text{ for } 1 \leq i \leq r \tag{5}$$

v is discretized into an action. To extract unknown qualitative intentions, we replace the unknown g_i with an ML model [18].

Preprocessing. In QI-ABAC$_0$, IDs are converted into attributes, e.g., User ID "a001p54" is converted into the user's position ("worker") and department ("R&D") by referring ID management system (context data storage in Fig. 5). In QI-ABAC$_{\mathrm{I}}$ and QI-ABAC$_{\mathrm{II}}$, our proposed system randomly samples attribute vectors in a scope, and obtains the corresponding action following the original policy. They are a part of the decision examples. In order to deal with the conditional variables and to use the "unknown qualitative intention" extraction, the system transforms the attribute vectors into computable vectors. We use one-hot vector encoding, e.g., $x_3 \rightarrow (0, 0, 1, 0, 0, 0), x_7 \rightarrow (0, 1, 0, 0)$. Continuous variables (e.g., trust score) are individually normalized from -1 to 1. These encoded variables are inputs of "NN(a)" and "NN(b)" as described below.

Implementation. We apply a monotonic deep neural network [14] for feasibility (i.e., QI-ABAC-DNN) as our main proposal. Besides, we also use monotonic XGBoost [5] as another choice (i.e., QI-ABAC-XGB). Therefore, we can combine three applications and these two ML models, e.g., QI-ABAC$_0$-XGB, QI-ABAC$_{\mathrm{II}}$-DNN.

QI-ABAC-DNN The model consists of three types of fully connected neural networks ("NN(a)", "NN(b)", "NN(c)" enclosed by dotted blue rectangles in Fig. 6). NN(a) is the extractor(s) of the unknown qualitative intention(s). It substitutes for the function g_i of Equation 3. This neural network has three hidden layers, each 50 wide with Rectified Linear Unit (ReLU). It outputs a single value with Tanh function. Second, NN(b) is a conditional variables' encoder. It receives encoded conditional variables $x_{c_1}, x_{c_2}, \cdots$. The hidden layers have the same structure as NN(a). Lastly, NN(c) outputs a value to allow access. It is a fully monotonic neural network with all the weights set above zero. There are two hidden layers with 20 widths with LeakyReLU. It outputs a single value with a Sigmoid function. We use cross entropy as the loss function.

QI-ABAC-XGB The XGBoost-based model is an ensemble decision tree. We train the model with monotonic constraints. The parameters are set to the default values of the python package [5].

6 EVALUATION

This section evaluates the performance of our three applications (QI-ABAC$_0$, QI-ABAC$_I$, QI-ABAC$_{II}$).

6.1 Experimental Setup

Algorithms. We use two existing rule mining methods, two classifiers, and a naive rule aggregation method ("allow" override of rules) as a practical baseline. We do not use [6] and other methods that focus on the access frequency, which is not our focus.

- **Compared methods** Rule mining: Karimi [13], Regression Tree [3]. Classifier [4]: XGBoost [5], Deep neural network (DNN). Base-line: Aggregation [21].
- **Our methods** QI-ABAC-DNN: based on Multi-layer perceptron, QI-ABAC-XGB: based on XGBoost.

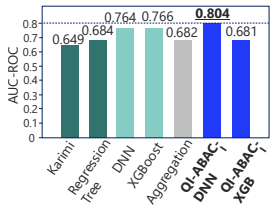
Dataset. We use three synthetic sample policies [22] and one realistic access log data [9].

- **University** ABAC policy with 10 number of rules [22].
- **Health care** The number of rules is 9 [22].
- **Project management** The number of rules is 11 [22].
- **Amazon** ABAC Access permissions and history [9] for reference evaluation. The evaluation is limited due to its anonymity.

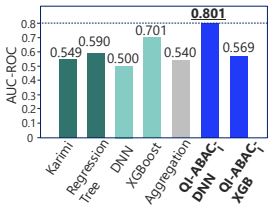
We define qualitative intentions for the university, health care, and project management sample policies as follows: 1) If the subject and object department are the same (i.e., owned by the department), they have higher order, and 2) the user with a higher position (e.g., chair) is ordered higher than other users (e.g., student). Following these two ways, we define six to seven qualitative effects for each sample policy. We select the first two out of 10, 9, and 11 rules in three sample policies. The selected rules are considered original policies, and decisions following the policies are used for the training. The selection of the "two rules" corresponds to a specific resource type (e.g., grade book). The small policy represents the assumption of disregard for other resource types. We evaluate how well our method can refine the policy and make it applicable to other resource types (e.g., application for admission). We define the scope of each rule based on the attributes used. These synthetic policies are defined with 11–14 attributes and have over 200,000 valid attribute vectors. In contrast, the evaluation using the Amazon dataset is performed the same way as the conventional methods, split into training and testing data. The Amazon dataset is anonymized. We need to extract unknown qualitative intentions.

6.2 Evaluation Methodology

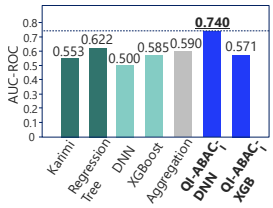
We define two evaluation tasks as follows. First, we evaluate the scope extension performance (QI-ABAC$_{\mathrm{I}}$). We input decision examples following a few rules (two rules). In addition, we also evaluate the classification performance using access logs of Amazon (QI-ABAC$_0$). Second, we feed additional decision examples and evaluate the number of them to obtain effective policies (QI-ABAC$_{\mathrm{II}}$). The additional decision examples follow the full-size sample policies, i.e., all rules are enabled. For QI-ABAC$_{\mathrm{I}}$ and QI-ABAC$_{\mathrm{II}}$, we evaluate the similarity between the classifier's decision and that of the full-size policy. The QI-ABAC$_{\mathrm{I}}$ evaluation aims to test biased sample problems. We evaluate QI-ABAC$_{\mathrm{II}}$ that solves sparse sample problems. QI-ABAC$_0$ has the same setup as existing methods.



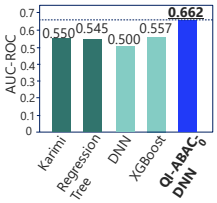
(a) AUC of QI-ABAC$_I$ with university policy.



(b) AUC of QI-ABAC$_I$ with health care policy.



(c) AUC of QI-ABAC$_I$ with project management policy.



(d) AUC of QI-ABAC$_0$ with amazon access samples.

Figure 7: The similarity between the ideal policy [22] (composed by 10 rules) and the created decision engine's decisions (a, b, c). Log-based decision engine's performance (d)[9].

We measure the performance with *Area Under the Curve of Receiver Operating Characteristic* (AUC of ROC curve). We randomly sample 10,000 valid attribute vectors (both for training and testing). We use AUC as a measure since our decision engine and other methods have controllable strictness. AUC represents the average True Positive Rate (TPR) for strictness. It equals the average True Negative Rate (TNR). For instance, the university sample policy has over 125,000 valid attribute vectors that should be denied. Hence, an increase of AUC by 1 % insists that 1250 additional unexpectedly privileged access requests can be prevented.

In the Amazon dataset, the "aggregation" method is excluded due to the lack of rules. Likewise, QI-ABAC$_I$-XGB is excluded because it can not extract unknown qualitative intentions. Also, we do not evaluate the second task as it is the same as the first task.

6.3 Evaluation Result

Our proposed method outperforms the existing methods. Fig. 7a, 7b, 7c presents the scope extending task. QI-ABAC$_I$-DNN performs the best. The advantage over the second-best method is at least 3 %, 10 %, and 11 % in each sample policy. Compared to the standard policy aggregation [21] (base-line), QI-ABAC$_I$-DNN performs better with 12 %, 26 %, and 15 % differences in AUC. In contrast, QI-ABAC$_I$-XGB performs poorly. The result for the Amazon access sample is shown in Fig. 7d. Even though the data set does not include any structured rules, QI-ABAC$_0$-DNN outperforms the compared methods with 10 % difference of AUC. Fig. 8a, 8b, 8c describes AUC with additional decision examples. In most cases, QI-ABAC$_{II}$-DNN and QI-ABAC$_{II}$-XGB perform better than other methods, rapidly approximating the ideal policies. The points at which the number of additional decision example is 0 is the same cases as those of Fig. 7a, 7b, 7c. In Fig. 8a, QI-ABAC$_{II}$-DNN achieves AUC > 0.9 at 40 number of additional samples. QI-ABAC$_{II}$-XGB also reaches the same line. Plain XGBoost achieves AUC > 0.9 at 120 additional examples, which indicates that our proposed method (QI-ABAC$_{II}$-DNN and QI-ABAC$_{II}$-XGB) requires 65 % fewer decision examples to approximate the ideal policy than the compared method. Although the proposed methods perform well, QI-ABAC$_{II}$-DNN has a similar performance with a compared method (XGBoost) in Fig. 8c as the number of additional examples increases. We proved three advantages as below.

- **Sparse sample** As shown in Fig. 8a, 8b, 8c, our proposed methods approached the ideal policy with a small number of additional decision examples.
- **Fine-granularity** Our proposed methods reached higher AUCs than the base-line ("Aggregation") in Fig. 7a, 7b, and 7c, and approximate the fine-grained policy with high AUC (> 0.9) as shown in Fig. 8a, 8b, 8c.
- **Rationality** The highest AUCs in Fig. 7a, 7b, and 7c are the result of rational extensions of scopes of the original policies.

7 DISCUSSION

In this section, we first interpret the evaluation results. Next, we discuss towards the validation of the decision engine.

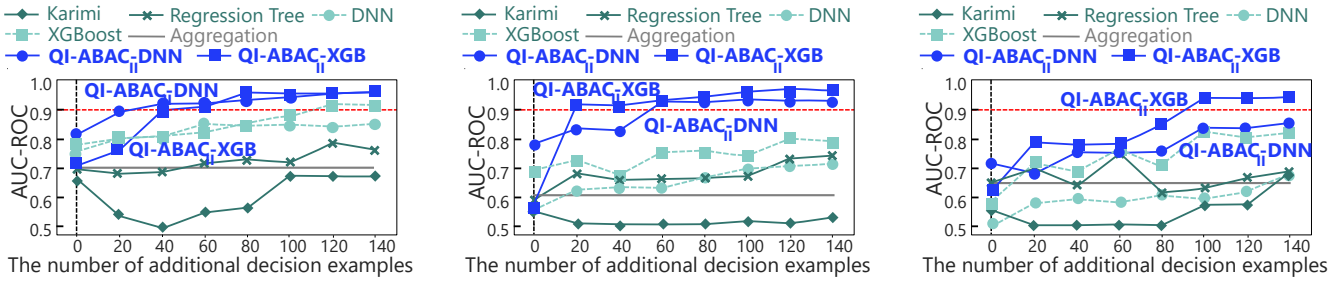
7.1 Discussion on Performance

In the first task to extend scopes of policies (Fig. 7a, 7b, 7c), QI-ABAC$_I$-DNN performs the best, while QI-ABAC$_I$-XGB does not. To explain this observation, we assume the value to allow access, the output of our model, is additive. For example, consider the case where an access user has greater business needs as a manager than a collaborator. Similarly, assume that the business needs are large if the user and the resource owner belong to the same department. If both conditions are met, the business benefit would be greater than if only one were met. The monotonic DNN-based classifier is non-linearly additive due to its smoothness, which makes it suitable for inferring the additive benefits, i.e., extrapolation task. In contrast, XGBoost is not. In case the number of additional examples is small, the QI-ABAC$_I$-XGB performs worse than a naive XGBoost. It may suffer the reduced representation ability due to the constraints. The second task in Fig. 8a, 8b, 8c directly benefit from monotonic constraints, which improve the robustness. As decision examples are added, QI-ABAC$_{II}$-XGB performs the best, followed by QI-ABAC$_{II}$-DNN. It is because of the interpolation task between sparse samples that are distributed throughout the privilege universe.

Our ultimate goal is to obtain a fine-grained effective decision engine for complex organizations. However, the synthetic policies we use in this paper are small. It is because we aim to prove a basic concept. Evaluation with large realistic policies is the future work.

7.2 Towards validation of decision engine

Since our method aims to refine given decisions, it is essential to ensure the validity of decisions. For example, it is crucial to identify accesses that should never be mistakenly allowed or denied to avoid fatal damage. However, it takes time and effort to identify all. Instead, the interpretability of access decisions will help. For instance, the value to allow can be interpreted as confidence in decisions. A value close to the threshold (e.g., 0.5) indicates unconfidence. Another approach is to convert the classifier into an interpretable one. In [19], the authors discuss converting into a decision tree.



(a) University policy **(b) Health care policy** **(c) Project management policy**

Figure 8: The ideal policy reconstruction performance of QI-ABAC$_{II}$ with varying the number of additional decision examples.

Our proposed method is interpretable. The intention vector shows which perspectives given access are desirable or undesirable. Furthermore, beyond interpretability, our constrained decision model ensures that access that is desirable in all respects compared to permitted access is permitted. Likewise, access that is less desirable in all aspects than prohibited access is prohibited. As previously discussed, it is the basis for extending decisions to not envisioned situations. Actionable AI (Artificial Intelligence) [15] concept had been proposed for systems that take actions through ML-based decisions. There, the system needs to take appropriate actions even for purposes and situations that were not envisioned. The MCDM is beneficial to acquire the actionability [15]. An underlying goal of our study is to achieve actionability, which we have obtained.

8 CONCLUSION

In this paper, we proposed a framework to refine access control policies (ACL policy, access logs, and ABAC policies) to an improved ABAC decision engine to balance security and needs better. In order to realize that, we formulated qualitative intentions behind human judgments, which have yet to be systematically handled, to refine the policy. It can guide the access decision to the appropriate one, even in situations involving access purposes or threats that the policy managers did not envision. We evaluated the proposed method and showed its efficacy using real access logs and synthetic ABAC policies. As we understand it, this study is the first to propose the concept of Actionable AI aiming to refine ABAC policies.

ACKNOWLEDGMENTS

We gratefully acknowledge support from ETH Zurich and the Zurich Information Security and Privacy Center (ZISC).

REFERENCES

[1] 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), 1–25.
[2] Wu Bei, Chen Xing-yuan, Wang Yong-liang, Dai Xiang-dong, and Peng Jun. 2008. Network system model-based multi-level policy generation and representation. In *Proceedings of the IEEE International Conference on Computer Science and Software Engineering.*
[3] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 2017. *Classification and regression trees.* Routledge.
[4] Luca Cappelletti, Stefano Valtolina, Giorgio Valentini, Marco Mesiti, and Elisa Bertino. 2019. On the quality of classification models for inferring ABAC policies from access logs. In *2019 IEEE International Conference on Big Data (Big Data).* IEEE, 4000–4007.
[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD).*
[6] Carlos Cotrini, Thilo Weghorn, and David Basin. 2018. Mining ABAC rules from sparse logs. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P).*
[7] Thierry Denoeux. 2019. Decision-making with belief functions: a review. *International Journal of Approximate Reasoning* 109 (2019), 87–110.
[8] Daniel Ricardo dos Santos, Roberto Marinho, Gustavo Roecker Schmitt, Carla Merkle Westphall, and Carlos Becker Westphall. 2016. A framework and risk assessment approaches for risk-based access control in the cloud. *Journal of Network and Computer Applications* 74 (2016), 86–97.
[9] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[10] Padmavathi Iyer and Amirreza Masoumzadeh. 2020. Active learning of relationship-based access control policies. In *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies.* 155–166.
[11] Xin Jin, Ram Krishnan, and Ravi Sandhu. 2012. A Unified Attribute-based Access Control Model Covering DAC, MAC and RBAC. In *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy.*
[12] Savith Kandala, Ravi Sandhu, and Venkata Bhamidipati. 2011. An attribute based framework for risk-adaptive access control models. In *2011 Sixth International Conference on Availability, Reliability and Security.* IEEE, 236–241.
[13] Leila Karimi and James Joshi. 2018. An unsupervised learning based approach for mining attribute based access control policies. In *Proceedings of the IEEE International Conference on Big Data (Big Data).*
[14] Bernhard Lang. 2005. Monotonic multi-layer perceptron networks as universal approximators. In *Proceedings of the International Conference on Artificial Neural Networks.*
[15] Igor Linkov, Stephanie Galaitsi, Benjamin D Trump, Jeffrey M Keisler, and Alexander Kott. 2020. Cybertrust: From explainable to actionable and interpretable artificial intelligence. *Computer* 53, 9 (2020), 91–96.
[16] Thomas Lukaseder, Maya Halter, and Frank Kargl. 2020. Context-based Access Control and Trust Scores in Zero Trust Campus Networks. *SICHERHEIT* (2020).
[17] Robert McGraw. 2009. Risk-adaptable access control (radac). In *Privilege (Access) Management Workshop. NIST–National Institute of Standards and Technology–Information Technology Laboratory*, Vol. 25. 55–58.
[18] An phi Nguyen and María Rodríguez Martínez. 2019. Mononet: towards interpretable models by learning monotonic features. *arXiv preprint arXiv:1909.13611* (2019).
[19] Mohammad Nur Nobi, Ram Krishnan, Yufei Huang, Mehrnoosh Shakarami, and Ravi Sandhu. 2022. Toward Deep Learning Based Access Control. In *Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy.* 143–154.
[20] Matthew W Sanders and Chuan Yue. 2019. Mining least privilege attribute based access control policies. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC).*
[21] OASIS Standard. 2013. extensible access control markup language (xacml) version 3.0. *A:(22)* (2013). http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.
[22] 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.
[23] Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya R Gupta. 2017. Deep lattice networks and partial monotonic functions. In *Proceedings of the International Conference on Neural Information Processing Systems.*