Received 30 August 2022, accepted 3 October 2022, date of publication 10 October 2022, date of current version 18 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3213645

RESEARCH ARTICLE

Integrating Cyber Deception Into Attribute-Based Access Control (ABAC) for Insider Threat Detection

MANAR ALOHALY[1], OLUSESI BALOGUN[2], (Student Member, IEEE), AND DANIEL TAKABI[2], (Member, IEEE)

[1]Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
[2]Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA

Corresponding author: Olusesi Balogun (obalogun6@student.gsu.edu)

This work was supported in part by the National Science Foundation under Grant 2006329.

ABSTRACT Insider threat is an ever-present challenge to corporate security. The availability of knowledge and privileges to insiders makes it extremely difficult to prevent, detect or deter malicious insider activities. In the literature, several studies have proposed deception-based approaches to mitigate insider threats through different layers of corporate systems. However, the integration of access control and cyber deception methods has not been adequately discussed. In this paper, we integrate Attribute-based Access Control (ABAC) with honey-based deception techniques to effectively track insiders, particularly in the context of a dynamic work environment. To the best of our knowledge, this is the first study to design, implement and evaluate this integration. Our evaluation results show that the proposed framework reliably identifies sensitive attributes in the system and generates indistinguishable honey values to protect them with an average similarity score of 0.90 to the truth.

INDEX TERMS Insider threat, defensive deception, attribute-based access control (ABAC), honey attribute, sensitivity estimation.

I. INTRODUCTION

Insider threat poses a serious security risk as it originates from within the organization itself. It typically occurs when a trusted employee (former or current) misuses the granted privileges to achieve a secondary unauthorized purpose. Unlike external threats that are often anticipated and prepared for, insider threat tends to go undetected for a long time leading to costly security breaches [5]. Besides, the complexity involved in locating the insider adversary extremely amplifies the difficulty of mitigating insider threats [11]. Several recent studies have highlighted the severity of insider threats towards organizations. In 2020, the Securonix Threat Research Team conducted a threat analysis of over 300 security incidents across several organizations from 8 different sectors. The study showed that privilege misuse is the second most common type of incidents, at 19% of all cases [6]. Similarly, a survey study conducted by the Cybersecurity Insiders, a community of 400,000 information security professionals [6], revealed that 72% –out of 373 participating organizations – have observed a frequent occurrence of insider attacks over the last 12 months, while 65% have actually experienced some sort of malicious insider activities over the same period [7].

In the literature, the malicious insider adversary is categorized into two classes, masqueraders and traitors [9]. Masqueraders are illegitimate users who impersonate legitimate users. Traitors, on the other hand, are legitimate users within an organization who misuse their access rights for their benefits [23]. To detect masqueraders, an anomaly-based intrusion detection approach is the commonly used solution. It functions by modeling the normal behavior of a legitimate user and capturing the deviation from that model [8]. It is worthy to note that external attackers may become masqueraders by

The associate editor coordinating the review of this manuscript and approving it for publication was Sathish Kumar.

stealing valid employee credentials [13]. However, our focus in this study is to mitigate the traitor threat. The term insider is used herein to refer to the traitor.

Several studies have researched the insider threat detection problem across different domains. The proposed approaches can broadly be categorized into four lines of research: psychological-based, behavioral-based, content-based, and deception-based. The psychological-based approach uses human bio-signals capturing the brain activity [15], heart rate [46], and eye movements [17] in building a framework to monitor and detect malicious insider threats [15]. The behavioral-based approach analyzes access logs collected from different systems and appliances to detect insiders activities [18]. On the other hand, the content-based approach applies machine learning and natural language processing techniques to textual content to build models for insider threat detection [2], [3], [4]. Finally, the deception-based approach uses decoy assets otherwise known as honey elements such as honey permissions [23], honeypots [25], honey files [19], honey documents [20], honey tokens [21], [26], honey words [24], and honey encryption [22] to attract and track insiders. Unlike other approaches, deception-based measures provide efficient early signs of insider incidents with the least amount of data collected from former or potential insiders [44]. However, none of the existing deception studies have considered the insider threat detection problem in the context of today's dynamic work environment.

To address this gap, we propose an ABAC-based defensive deception framework for insider threat detection. ABAC is an access control model that provides an unprecedented amount of flexibility to cope with dynamic and change-oriented business environments [1]. In particular, we extend and integrate the original ABAC model with decoys protecting sensitive objects. With this extension, access requests targeting sensitive objects are suspicious and indicate potential malicious activity. To the best of our knowledge, this is the first work that integrates cyber deception into ABAC for insider threat detection. The main contributions of this paper are:

1) Integrating the notion of honey elements with the standard ABAC model to build a defensive deception framework for insider threat detection.
2) Extending the standard ABAC with deception-related components such as sensitivity estimator, honey attribute generator, and monitoring unit.
3) Generating a realistic dataset to evaluate the proposed framework that could also be used for future studies in this area.

The remainder of this paper is organized as follows: Section II provides the background information. Section III summarizes the related work. Section IV outlines the threat model, and Section V describes the details of the proposed approach. Section VI discusses the experiments and results. Section VII presents a discussion and our future directions, and we conclude the study in Section VIII.

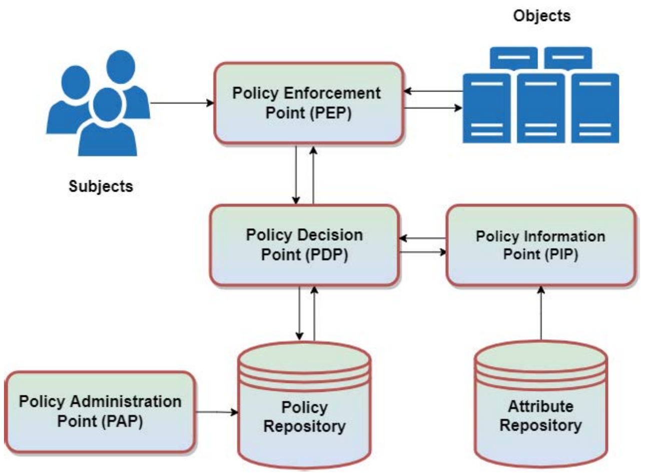

Objects
Subjects
Policy Enforcement Point (PEP)
Policy Decision Point (PDP)
Policy Information Point (PIP)
Policy Administration Point (PAP)
Policy Repository
Attribute Repository

FIGURE 1. **National institute of standards and technology (NIST) standard ABAC model [1].**

II. BACKGROUND

In the following subsections, we provide background information on key concepts used in this work, namely Attribute-based Access Control (ABAC) and genetic algorithm.

A. OVERVIEW OF ABAC

In this section, we provide an overview of ABAC architecture. To scale the compatibility and the applicability of the proposed framework, we focus on the National Institute of Standards and Technology (NIST) standardized ABAC model [1]. According to NIST definition, ABAC is an access control model in which the requests issued by subjects to perform actions on objects are granted or denied based on the attributes of subjects, objects, actions, and the environment conditions. The attributes of an object describe its characteristics such as its type, date of creation, etc. On the other hand, subjects in ABAC can be human or non-human (application) entities, and may have attributes such as name, office number, job title, etc. The actions performed on objects can be open, edit, execute, etc. Similarly, the environment conditions attributes indicate the current state of the system including current timestamp, location, threat level, and temperature, etc.

Authorization policies in ABAC are defined as a set of rules. Each rule is a conjunction of the attributes of subjects, objects, actions, and environment conditions. The attributes and the policies are respectively stored in the attribute and policy repositories, as illustrated in Figure 1. Also, these repositories contain information that describes the set of attributes and policies, referred to as meta-attributes and meta-policies.

In addition, ABAC model integrates four functional points to analyze and evaluate access requests. These functional points include: (1) Policy Information Point (PIP) which controls the access to attribute and policy repositories to get the data needed to evaluate access requests; (2) Policy Decision Point (PDP) which makes logical access decisions by evaluating the applicable policies and attributes against access requests; (3) Policy Enforcement Point (PEP) which enforces access authorization decisions; and (4) Policy Administration Point (PAP) which provides an interface for creating, managing, and testing policies. Figure 1 shows the connections of these functional points.

B. GENETIC ALGORITHM

Genetic Algorithm (GA) is a population-based algorithm that derives its inspiration from evolution theory [33] and was first introduced by John Holland [37]. GAs are well-suited for evolving solutions for real-world problems [38]. The algorithm starts by generating an initial population either randomly or with prior knowledge. Each individual solution in the population is represented as a chromosome. The goodness of the individual is evaluated using a fitness function, which is a problem-specific metric designed to estimate the closeness to the optimal solution. Then, the best-fit individuals are selected as intermediate parents to generate the next population. Once the parents are selected, GA performs two basic operations to produce new offspring, namely crossover, and mutation. The crossover operation splits up the parent chromosomes and recombines them to produce new individuals. The split is performed at a random point or based on standard heuristics [40]. After the crossover, a mutation operation is used to introduce randomness to the solution by altering some genes of the resulting chromosomes. The process is repeated until a predefined stopping criterion is met [28]. In this work, we apply GA to generate honey attributes as part of the proposed deception framework.

III. RELATED WORK

In this section, we discuss existing insider threat detection studies in four lines of research.

A. DECEPTION-BASED APPROACHES

Defensive deception has gained a significant attention from the cybersecurity community. Several studies have been published on this field. To understand the perspective of existing research and the contribution of our work, we introduced 8 factors to compare relevant studies: (1) the sensitive elements needed to be protected against insider attack; (2) the technique used to identify sensitive elements; (3) the honey elements; (4) the technique used to generate the honey elements; (5) the host system that is extended to incorporate the deception component; (6) the dataset; (7) the size of the dataset; (8) and the performance evaluation. Table 1 presents a summary of the findings, while the rest of this subsection discusses each study separately.

Bercovitch et al. [26] proposed ‘‘HoneyGen’’system which automates the production of honey tokens for any type of data. The authors’ proposed approach consists of three main steps:(1) mining set of rules that capture the characteristics of real data; (2) generating artificial or honey data items based on the mined rules; and (3) ranking the resulting honey items based on their similarity to the ground truth data. The high similarity scores between honey elements and the truth data suggest high degree of indispensability. The authors conducted a turing-like test to evaluate the quality of the produced honey elements. The evaluation showed a detection rate confidence interval of (0.47, 0.67) with a confidence level of 0.95 indicating a random detection rate. Although the results are promising, this study focused only on generating honey elements. Unlike our proposed framework, it did not address the problem of identifying sensitive data that needs to be protected with honey elements.

Bowen et al. [20] proposed a Decoy Document Distributor, D^3, system which automatically generates and distributes decoy documents across a file system aimed at luring malicious users. The authors employed a rule-based approach to design the decoy documents. For evaluation, they deployed the D^3 system on the honeynet platform and carried out a user study experiment with 20 participants. The results indicate that during the first week of the experiment, the system reported that around 30% of the participants have used/misused the bait data found in decoy documents suggesting the effectiveness of the D^3 system. However, Bowens’ proposal has limited application domains; it is only applicable in file and directory systems. Moreover, It distributes honey elements randomly across the system. Such random integration of honey elements worsens the space complexity of the system.

Bhagat et al. [25] proposed the use of honeypots for intrusion detection problem. The authors studied the interaction of attackers with the honeypots and analyzed data retrieved from honeypots for possible attacks. The results showed that the TCP, among other network protocols, can easily be compromised.

Srinivasa et al. [21] proposed a set of techniques, referred to as fingerprinting honey tokens, to compromise a decoy system. The authors of this study explicitly mentioned the difficulty of fingerprinting honey tokens created at data level. This finding supports the reliability our proposed solution

Kaghazgaran et al. [23] proposed an extension to the standard role-based access control (RBAC) to support insider threats detection. The proposed model introduced honey permissions and honey objects as new elements to protect sensitive objects. The authors evaluated the work using Americas Large dataset. The analysis of their results using the weighted structural complexity (WSC) showed that the overhead introduced by the proposed approach is minimal in comparison with the standard RBAC. Perhaps Kaghazgaran’s work is most similar in spirit to our study [23]. Unlike [23], which required a manually configured sensitivity level of objects, our proposed framework automatically computes sensitivity scores of attributes and objects. This is a significant improvement over [23] as the manual configuration is subjective, error-prone and difficult to maintain.

TABLE 1. Deception and insider threat detection related work. In this table, N/A stands for not applicable while N/R means not reported.

Study	Sensitive Element	Sens. Element Tech.	Honey Element	Honey Element Tech	Host	Dataset	Size	Performance
[26]	Organization resources	N/R	Honey Tokens	Rule mining, honey token generation, and the likelihood rating.	Honeypot	Profiles of Individuals	1,093	Detection rate confidence interval: (0.47, 0.67); Likelihood range: Real tokens - [-6.99,-6], Honeytokens - [-14.99,-14]
[20]	Enterprise Systems	N/R	Decoy Documents	Automatic generation of Decoy Documents using the D^3 System	File System	Honeynet containing several virtual machines running Linux	15 directories' and 50-100 files.	Detected Rate: 30%
[25]	Network System	N/R	Honeypot	N/R	Honeyd and Kfsensor	N/R	N/R	Most Compromised Port: TCP
[21]	Network System	N/R	Honeytoken	N/R	Canarytoken service	N/R	N/R	N/R
[23]	Organizational Data	manual	Honey Permissions and honey objects	N/R	Role Based Access Control	"Americas Large" dataset	404 roles, 3,485 users	Weight structural complexity: 90,143.

B. PSYCHOLOGICAL-BASED APPROACHES

This line of research studies the emotional and mental states of individuals to carry out a risk assessment and identify potential insider threats.

Brdiczka et al. [16] proposed a proactive method of detecting insider threats by integrating Structural Anomaly Detection (SA) from social and information networks with Psychological Profiling (PP) of individuals using their behavioral patterns. In [16], insider threats are predicted through a fusion and ranking of the outcomes from SA and PP. The authors evaluated the proposed approach using the World of Warcraft (WoW) dataset and they reported an average classification accuracy of 83.86%. Similarly, Greitzer et al. [14] proposed a psychosocial model to assess an employee's behavior suggesting an increased risk of insider abuse. The authors developed a list of indicators based on examination of published case studies and discussions with experienced human resources (HR) professionals. These include disgruntlement, anger management issue, performance, stress, confrontational behavior, and lack of dependability. based on the proposed psychosocial indicators and using Genie dataset, the authors built a Bayesian Network (BN) model to identify individuals who pose a possible insider threat. The results showed a high agreement between the model outcomes and the judgement human experts. More recently, Hashem et al. [15] developed a classification model using electrical signals, e.g., Electroencephalography (EEG), Electrocardiogram (ECG), and Electromyography (EMG), generated by humans biological activities to detect potential insiders. The authors reported a detection accuracy of about 86% indicating that this approach holds promise. However, it poses plethora of privacy concerns associated with the use of physiological signals.

While studies in this field heavily considered the dynamic nature of human psychology and its relation to insider attacks, they failed in capturing the dynamic nature of inter and intra business processes in today's work environment. It is worth mentioning that our proposed model complements this limitation with the use of ABAC.

C. BEHAVIORAL-BASED APPROACHES

This approach of insider threat detection studies the set of actions and reactions produced by individuals as they interact with their environment. The goal is to identify behavioral patterns that pose insider threats. Leg et al. [27] proposed a behavior profiling and anomaly detection model to identify a consistent and acceptable pattern of users' behavior. The model analyzes the daily activity of a user and produces three levels of alerts which are policy violations, threshold-based anomalies, and deviation-based anomalies. Yessir et al. [17] studied different sets of behaviors and their relation to insider threats. In [17], the authors used eye movement and pupil size responses to build a classification model to classify both malicious and benign computer-based activities. The model was developed and evaluated through a user study with 30 participants who were set to perform several scenarios of malicious and benign activities. The authors reported an accuracy of 85% using the proposed behavioral model. In addition, they empirically showed that the feature-level fusion of physiological and behavioral indicators yields at least a 10% improvement in classification accuracy. Jiang et al. [18] proposed a deep learning-based approach using Graph Convolutional Networks (GCN) to detect anomalous behaviors of users posing possible insider threats. The GCN model was evaluated on the CMU CERT v4.2 dataset consisting of 1000 users in a simulated network. The results showed a detection rate of 93% outperforming other machine learning models.

Similar to the psychological-based approaches, the behavioral-based approaches of insider threats detection focus solely on the human elements. However, it overlocks resources and other elements of the business environment. Our proposed approach complements this limitation by considering the sensitivity and dynamics of organizational resources.

D. CONTENT-BASED APPROACHES

This approach analyzes the content created by users to construct an insider threat detection model, mostly using machine learning and/or natural language processing techniques. A major challenge facing this line of research is the dearth of data arising from data protection legislation. An early effort to promote and advance this type of research was proposed by Glass et al. [4] to generate a synthetic textual dataset free of privacy concerns associated with real data. During the same period of time, Brown et al. [2] exploited the co-relation between the words use and a set of psychological and behavioral risk factors to develop a proactive insider threat detection framework. The authors evaluated their work on Enron emails, a real-world emails corpus. The evaluation results indicated that the proposed approach successfully identified insiders in about 66.7% of all cases. More recently, Paxton et al. [3] proposed a natural language processing framework to model incidents of insider attacks using written and recorded notes of an incident. Despite many years of research, the practical progress on this field is relatively slow due to the lack of data and limitations of the underling techniques.

IV. THREAT MODEL AND PROBLEM STATEMENT

The attacks from the inside of an organization are more challenging to manage because of the privileges given to insiders. In this work, we consider a threat model where insiders present risks to sensitive data of an organization by performing a set of actions. We assume that the actions have the potential of affecting the integrity and availability of these critical data. In addition, the actions vary across organizations and are dependent on the data type or content. For example, the action set may involve read, open, edit, delete, copy, or create operations. We also assume that the motivation for misuse of privileges may include the desire for illegal financial gains, promotion, political adversity, or satisfy curiosity.

To mitigate these risks, we present a defensive strategy by incorporating a deception mechanism into attribute based access control. As shown in Figure 2, we add the monitoring unit, sensitivity estimator, and honey attribute generator to the existing functional units of ABAC; PEP, PDP, PIP, and PAP. The sensitivity estimator estimates objects' sensitivities and identify the sensitive objects to be considered as candidate objects for deception purpose. The honey attribute generator generates a set of honey attributes for the candidate objects while the monitoring unit tracks the object's activities. In the

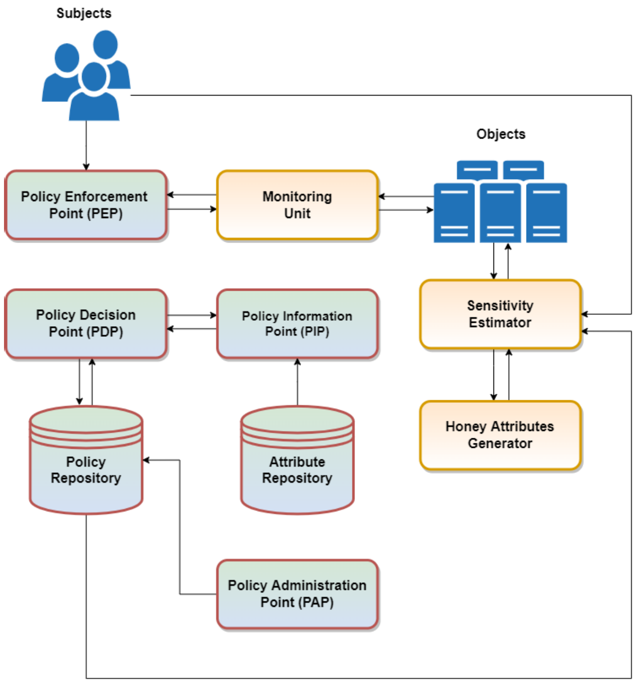

Subjects
Objects
Policy Enforcement Point (PEP)
Monitoring Unit
Policy Decision Point (PDP)
Policy Information Point (PIP)
Sensitivity Estimator
Honey Attributes Generator
Policy Repository
Attribute Repository
Policy Administration Point (PAP)

FIGURE 2. The proposed insider threat detection framework: it integrates the standard ABAC model with deception-related components.

process flow shown in Figure 3, the monitoring unit is integrated to the PEP of ABAC. It analyses the access decision from the PDP and tracks if honey attributes are used in access request to detect the presence of potential insider. In the event of detection of potential insider, we suggest that the system may take any of two actions: deny access and sends an alarm to the admin, or presents honey objects to the potential insider and thereafter sends an alarm to the admin.

One potential use case is a hospital environment. After successful login, all Nurses and Doctors are given permission to create new patients' health records. The sensitivity scores of each health records' attributes are estimated using the sensitivity estimator according to equation 5. Next, we estimate the overall sensitivity of health records using equation 7. If the score value is greater than sensitivity score threshold, the record will be regarded as candidate object. Thereafter, the PDP determines if a particular nurse or doctor can access patients' health records by evaluating the subject's attributes, requested object's attributes, environment's attributes, access policies, and requested operations/actions. The monitoring unit evaluates the access decision and tracks the user's operations. It detects the presence of potential insiders and takes necessary actions thereafter.

V. THE PROPOSED APPROACH

We extend the standard ABAC model with deception-related components to enable insider threat detection. We introduce a Honey-Attribute-based Access Control framework for insider threat detection. The proposed framework identifies attributes with relatively high confidentiality or integrity requirements, referred to as sensitive attributes. It, then, produces honey attributes to protect the sensitive attributes. If a subject uses a honey attribute to access an object, the system recognizes it as a potential insider attack and generates an alarm accordingly. The remainder of this section presents the formal definition of the extended ABAC system and its components.

A. BASIC COMPONENTS

1) BASIC ENTITIES

1) **Subjects (U):** set of users in an organization that place requests to access objects or resources. Each subject u is a member of the set U ($u \in U$).
2) **Objects (OBS):** set of protected resources such as data records or files in an organization. Each object ob is a member of the set OBS ($ob \in OBS$).
3) **Sensitive Objects (SO):** subset of resources, $SO \subseteq OBS$, such as a data record or a file of which the accumulated sensitivity scores of its attributes (see the definition of Sensitive Attributes in this section) exceeds a predefined threshold. Each sensitive object so is a member of the set SO ($so \in SO$)
4) **Actions (A):** set of operations applicable to an object such as read, write, create, delete, update etc. Each action a is a member of the set A ($a \in A$).
5) **Rule (r):** A logical combination of attributes, actions and access decisions. A rule r defines an authorization condition that has to be satisfied to execute an access request. Each rule r is a member of the set R ($r \in R$).
6) **Policies** (P): set of rules. Each policy p is a member of the set P ($p \in P$).
7) **Decisions (D):** set of access decisions produced by the Policy set such that each decision d is a member of the D set ($d \in D$).

2) ATTRIBUTES

1) **Subject Attributes (UA):** set of attributes applicable to each user in an organization. Each user attribute ua is a member of the UA set ($ua \in UA$) such that ua is a tuple, where $ua = (name : value)$.
2) **Object Attributes (OA):** set of attributes applicable to each object in an organization. Each attribute oa is a member of the OA set ($oa \in OA$) such that oa is a tuple, where $oa = (name : value)$.
3) **Environment Attributes (EA):** set of attributes that indicate the current state of the system. Each attribute ea is a member of the set EA ($ea \in EA$) such that ea is a tuple: $ea = (name : value)$
4) **Sensitive Attributes (SA):** subset of objects attributes with high confidentiality or integrity requirements making them a potential target to insider attacks. The set of sensitive attributes SA is defined as:
$SA = \{sa : sa \in OA \wedge Sensitivity(sa) \geq K\}$.
Where $Sensitivity$ is a function, discussed in Subsection V-A3, to measure the sensitivity score of an attribute and K is a predefined threshold. Each attribute $sa \in SA$ is a tuple: $sa = (name : value)$
5) **Honey Attributes (HA):** set of attributes generated for each sensitive object as honey elements. Each attribute ha is a member of the set HA ($ha \in HA$) such that ha is a tuple: $ha = (name : value)$

3) FUNCTIONS

To support insider threat detection, we extend the standard ABAC model with the following functions:

1) **Sensitivity:** Returns a real value indicating the sensitivity of object's attributes, as shown in the following:
$$Sensitivity : OA \rightarrow \mathbb{R} \tag{1}$$
The larger the value, the higher the sensitivity, i.e. the confidentiality and integrity requirements of the attribute.
2) **HoneyAttribute:** Produces honey attributes values for the attributes of a sensitive object, as shown the following:
$$HoneyAttributes : SO \rightarrow \mathbb{HA} \tag{2}$$
3) **CheckHoneyAccess:** Determines if a user activates honey attributes by requesting an unauthorized access to the corresponding candidate objects. Formally defined as indicated in Equation 3
$$CheckHoneyAccess(ua, oa, ea, a) = \begin{cases} True & \{\text{if } oa \in HA\} \\ False & \{\text{if } oa \notin HA\} \end{cases} \tag{3}$$

As illustrated in Figure 2, the proposed framework consists of, in addition to the core functional points of ABAC model, a sensitivity assessment component, honey attributes generation component, and monitoring components. These three components work together to: (1) automatically identify sensitive objects having relatively high confidentiality or integrity requirements; (2) and protect them against insider attacks using honey data. In the following subsections, we discuss the details of each component.

B. MODULE 1: SENSITIVITY ASSESSMENT

The type of actions the users may exercise on an object and the number of users authorized to perform the action provide key insights on the sensitivity of the object and its attributes. For instance, writable objects are more sensitive than objects with only read access due to a higher risk of integrity breaches. In addition, an object is likely to be more sensitive if it is authorized to few users indicating high confidentiality requirements. Further, the level of confidentiality and integrity implies uncertainty on existing information regarding the object and its attributes. In other words, a confidential object attribute is less predictable and hence more uncertain. Similarly, write access exposes object attributes to high uncertainty. Based on this reasoning, we use the Shannon information content measure "surprisal" [10], which measures the uncertainty in an event using its probability P as defined in Equation 4, to quantify the sensitivity (or the amount of surprise and uncertainty) in an attribute access.

$$IC = -\log(P) \tag{4}$$

Particularly, we compute the sensitivity of an attribute *attr* by summing over the information content in each access targeting *attr* to perform an action a as shown in Equation 5.

$$S_{Objattr} = -\sum_{a \in A} \log(P_{Obj_{attr}}) \tag{5}$$

where the probability $P_{Obj_{attr}}$ is the probability of authorized users targeting the object attribute Obj_{attr} with the action $a \in A$, defined as follows.

$$P_{Obj_{attr}} = \frac{n_{Obj_{attr}}}{N} \tag{6}$$

where $n_{Obj_{attr}}$ and N indicate the number of authorized and the total number of users, respectively. Then, we compute the sensitivity of an object as the Root Mean Square (RMS) function of the sensitivity scores of its attributes as defined in Equation 7

$$S_{Obj} = \sqrt{\frac{1}{n}\sum_{i} S_{Obj\,attri}} \tag{7}$$

where n is the number of attributes of an object and $S_{Obj\,attri}$ is the sensitivity score of the i^{th} attribute of the object such that $i \leq n$. An S_{Obj} value greater than a sensitivity threshold indicates a sensitive object. The threshold is calculated as the RMS value of sensitivity scores of objects. The proposed sensitivity assessment algorithm is shown in Algorithm 1. We note that we applied the Root Mean Square (RMS) in this algorithm due to its sensitivity to large values [12]. That is, if the majority of the attributes of an object are of low sensitivity scores and only a few with high values, then the values indicating a sensitive object.

C. MODULE 2: HONEY ATTRIBUTE GENERATION

We use the genetic algorithm (GA) (see Subsection II-B) to generate honey attributes values for each sensitive object. A key property of a honey element is its indistinguishability from the truth. Therefore, we optimize the algorithm to produce honey values with a relatively random detection rate as discussed in the following steps.

1) STEP 1: STARTING POPULATION

To seed the algorithm with an initial population, we use existing attributes values. The values are categorized into categorical or numerical. For categorical attributes, we tokenize[1] each value into words that constitute individuals in the population. Each individual is then encoded using word embeddings. On the other hand, the numerical values are kept unchanged and encoded as is. This step shown in Algorithm 2, lines 3-11.

Algorithm 1: Sensitivity Assessment

```
Input  : an object $Obj$
Output: The sensitivity of the object $Obj$
1  Function getObjectSensitivity (Obj)
2      $A = getAllActions()$
3      $N = countUsers()$
4      $AttributesSensitivity = List()$
5      for $Obj_{attr} \in Obj.attributes()$ do
6          $AttProbabilities = List()$
7          for $a \in A$ do
               // calculates the probability
                  of users targeting the
                  attribute $Obj_{attr}$ to perform
                  action $a \in A$
8              $P_{Obj_{attr}} = \frac{countAuthorizedUsers(a)}{N}$
9              $AttProbabilities.add(P_{Obj_{attr}})$
10         end
           // returns the sensitivity of
              object attribute $Obj\,attr$
11         $S_{Obj\,attr} = getEntropy(AttProbabilities)$
           $AttributesSensitivity.add(S_{Obj\,attr})$
12     end
13     $S_{Obj} = getRMS(AttributesSensitivity)$
14     return $S_{Obj}$
```

2) STEP 2: EVALUATE THE FITNESS

The fitness value of each individual in the population is calculated based on its semantic similarity to real attributes values in the system. A group of individuals is then selected as intermediate parents based on their fitness. In Algorithm 2 Lines 12-17, we define the *getFitnessScore* function to evaluate the fitness scores. The function takes pair of values of which one is real and the other is GA-generated. It calculates the semantic similarity between each pair using GLOVE embedding[2] and returns a value in the range of [0-1], where 0 indicates completely dissimilar while 1 indicates identical pair. In general, individuals with high similarity to the truth are more indistinguishable, and hence more fit as honey values. However, to preserve the secrecy of the true values, we omit identical or close-to identical individuals by resetting their fitness to 0 when the similarity scores are higher than 0.97. Note that we experimented with several random threshold values and obtained the best results with a threshold value of 0.97.

3) STEP 3: GENERATE A NEW POPULATION

This process goes through three sub-steps: parents selection, crossover, and mutation. For the first generation, we randomly select one parent from the real values and the other from the initial population. For the subsequent generations, one parent is selected from the resulting offspring and the

[1]Tokenization is a process of breaking of splitting strings of text into individual words or phrases [43].

[2]Glove Embedding is an unsupervised learning algorithm used for generating words' vector representations [34].

Algorithm 2: Honey Attributes Generation

```
Input  : an object Obj
Output: Honey Attributes of the object Obj
1  Function getHoneyAttributes(Obj)
2      $HoneyAttributes(Obj) = List()$
       // generates the population of attributes
3      Function getPopulation()
4          $Population = List()$
           // categorical attributes
5          for $Obj_{attr} \in Obj.attributes()$ do
6              $Population.add(tokenize(Obj_{attr}))$
7          end
           // numerical attributes
8          for $Obj_{attr} \in Obj.attributes()$ do
9              $Population.addObj_{attr})$
10         end
11         return Population
       // estimates and returns fitness score
12     Function getFitnessScore(var 1, var 2)
13         $FitnessScore = SemanticSimilarityScore(var1, var2)$
14         $FitnessScore = SequenceMatcherScore(var1, var2)$
           // score to 0 if above 0.97 to preserve secrecy
15         if $Score > 0.97$ then
16             $Score_{var1,var2} = 0$
17         end
18         return Score
       // Perform cross over and mutation of two parent words and returns the offspring
19     Function Crossover()
20         for $Obj_{attr} \in Obj.attributes()$ do
21             $Obj_{hattr} = null, Score = 0$
22             $P_{init} = random(Population)$
23             $C = UniformCrossover(Obj_{attr}, P_{init})$
               // C is the Offspring
24             $getMutation(C)$
25             $Score_{init} = getFitnessScore(C, P_{init})$
26             while True do
27                 $P1 = C$
28                 $P2 = select(Population) \ni getFitnessScore(P1, P2) > Score_{init}$
29                 $C = UniformCrossover(P1, P2)$
30                 $getMutation(C)$
31                 $Score_{init} = Score$
32             end
33             $Obj_{hattr} = C$
34             $HoneyAttributes(Obj).add(Obj_{hattr})$
35         end
36     return HoneyAttributes(Obj)
```

other from the existing population using the roulette wheel selection algorithm [41]. Then, we apply uniform crossover operation to swap genes between the two selected parents, and finally mutate some genes in the resulting offspring to diversify the results [39]. For mutation, we design two operations; one for alphanumeric attributes and the other for categorical attributes. Our alphanumeric mutation involves selecting a random point i in the interval of [0, N], where N is the length of the crossover output. Then, it replaces the i_{th} character of the output with a random character of the same data type of existing alphanumeric attribute. For categorical attributes values, we use semantic word substitutions with Glove [34] as a mutation operation. It is worth noting that we empirically evaluated the mutations obtained with different word vector representations, such as FastText [35], Word2Vec [36], and BERT [42]. The results show that GloVe embeddings achieved more semantically related substitutions.

Step 4: Convergence: The algorithm iterates over the selection, crossover, and mutation operations. It converges when the similarity score of the honey attribute is equal to or beyond a predefined threshold. In this work, we set the threshold value to 0.6 to ensure that there is a level of indistinguishability between the real and honey attributes.

Algorithm 3: Cyber Deception

```
1  $A$: Set of action combinations in the system $= \{a_1, a_2, \dots, a_n\}$
2  $S$: Set of Subjects in the system
3  $A_{req}$: Set of Access Request
4  $S_i$: Set of Insiders = null
5  for $req \in A_{req}$ do
6      $A_d = (\text{req}_s \in S) \xrightarrow{(req_a \in A)} (\text{req}_o \in O)$
       // The system checks if the subject is unauthorized to access the object.
7      if $A_d ==$ "False" then
8          System Presents honey attributes
9          $A'_d = (\text{req}_s \in S) \xrightarrow{(req_a \in A)} (\text{req}_o \in O)$
10         if $(req_o \in C)$ and $(A'_d ==$ "True"$)$ then
               // System Presents dummy data to the subject
11             $S_i = S_i \bigcup s$
               // System sounds an alarm
12         end
13     end
14 end
```

D. MODULE 3: MONITORING

We integrate the monitoring module into the policy decision point (PDP) to achieve our goal of detecting potential insiders. Figure 3 illustrates the operation of this unit along with other components in the system. As shown in the

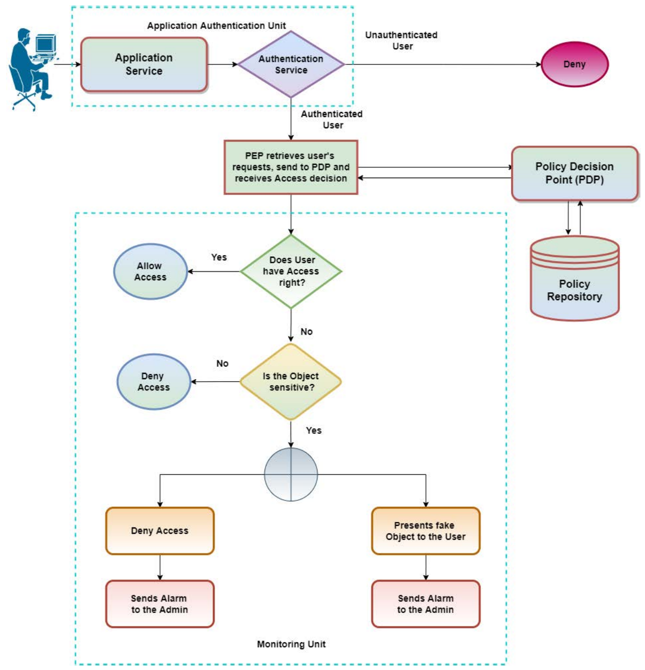


FIGURE 3. Process flowchart.

figure, the user must be authenticated before accessing the system's resources. Note that the authentication process is beyond the purview of this paper. Once authenticated, the monitoring workflow begins with the user submitting access requests. For each request, the PDP evaluates the user's and resource's attributes along with the requested action against the stored policy to make one of three possible decisions: (1) If the requesting user is authorized, the PDP returns ALLOW decision. Consequently, the user is granted the right to the submitted access request. (2) On the other hand, if the user is unauthorized and the requested resource is deemed sensitive, the PDP denies the access to the real resource and presents the user with an object of honey attributes as bait. The user is, then, monitored such that any attempt by the user to access the denied resource as enticed by the honey attribute indicates the presence of a potential insider. (3) Otherwise, the access request is denied. Algorithm 3 illustrates the overall deception process.

VI. EXPERIMENTS AND EVALUATION

This section presents setups and experiments conducted to assess the effectiveness of the proposed approach. Particularly, we focused on the following questions:

- **Q1:** How is the performance of the sensitivity estimator component (Module 1)?
- **Q2:** How is the performance of the honey attributes generator component (Module 2)?
- **Q3:** What is the complexity overhead induced by deception-related components?

A. DATASETS

Carrying out the experiments and answering our questions require a dataset that consists of subject and object attributes, subject and object attributes data (attributes values), and ABAC rules. To the best of our knowledge, there is no publicly available dataset with all of the required elements. For reliable evaluation, we constructed a novel dataset by integrating various real-world data/data resources in education domain as discussed in the following.

1) SUBJECT AND OBJECTS DATA

We obtained the subjects and objects attributes data using the California Basic Educational Data System (CBES) administrated by the Department of Education and released in 2018 [31]. The CBES provides publicly accessible datasets about students, staff, schools performance, course enrollments, …etc. In this work, we used the staff demographics and staff-assignment datasets as the subjects' data. The staff demographics dataset has 364,759 subjects with 16 attributes, while the staff assignment dataset has 1,269,836 unique records with 13 attributes. We linked the two datasets based on overlapping key attributes to generate one dataset of 1,269,836 unique records and 23 attributes. In addition, we used the course enrollment data, which contains 3,228,250 unique object records with 23 attributes, for the object dataset. To reduce the execution time, we performed a proportionate sampling from the resulting datasets.

As a result, we obtained 5 samples in the ration of 1:2.5 with 30000 and 75000 randomly selected subjects and objects of each sample. Using the sensitivity estimator in Subsection V-B, we identified 51675 sensitive and 18325 non-sensitive objects in the objects dataset, in a total of 75000 objects. Tables 2 and 3 describe the attributes of subjects and objects, respectively.

2) ABAC RULES

To achieve the goal of having holistic and realistic data for evaluation, we constructed the set of ABAC rules by drawing some policies from various sources including, the privacy of pupil records from the education code of the State of California [49073 -49079.7] [32], the Administration Assignment Manual of California Commission on Teacher Credentialing [29], the Data guide of California Longitudinal Pupil Achievement Data System (CALPADS) [30], and other policy documents of different educational institutions. Next, we reviewed the referenced sources to identify policies that can be defined using the subjects' and objects' attributes, discussed earlier. We note that some of our ABAC rules do not explicitly follow the contents in the policy sources, but rather we constructed the rules using those sources as guide and in consistence with our subject and objects datasets We identified 13 relevant policy sentences, and we defined them as JSON-based ABAC rules. Each rule has three sections; object attributes, subject attributes, and allowable actions. For action attributes, we identified three major actions; open, edit, and delete. We note that if any of the actions were not explicitly mentioned in the policies, we mapped those actions to open, edit, delete, or combination of two or all the actions.

TABLE 2. Subject dataset.

S/N	Attribute	Unique Values	Data Type	Description
1	Academic Year	1	Character	Academic year of the data
2	RecID	355617	Character	Random 7-digit ID assigned to each user
3	DistrictCode	1031	Character	A 7-digit code assigned to each district
4	SchoolCode	10108	Character	A 7-digit code assigned to each school
5	CountyName	58	Character	The county name for each school.
6	DistrictName	1019	Character	The district name for each school.
7	SchoolName	8834	Character	The name of each school
8	StaffType	3	Character	Unique value indicating the type of staff. A = Administrator, P = Pupil services, T = Teacher
9	AssignmentCode	95	Character	Code allocated based on each service type
10	ClassID	775146	Character	Unique code assigned to each classroom comprising of different course sections
11	CourseCode	555	Character	A 4-digit code mapped to each course
12	EstimatedFTE	1405	Numeric	Description
13	FileCreated	1	Date	The file creation date

TABLE 3. Objects dataset.

S/N	Attribute	Unique Values	Data Type	Description
1	Academic Year	1	Character	Academic year of the data
2	DistrictCode	1029	Character	A 7-digit code assigned to each district
3	SchoolCode	10052	Character	A 7-digit code assigned to each school
4	CountyName	58	Character	The county name for each school.
5	DistrictName	1017	Character	The district name for each school.
6	SchoolName	8778	Character	The name of each school
7	ClassID	800540	Character	Unique code assigned to each classroom comprising of different course sections
8	CourseCode	556	Character	A 4-digit code mapped to each course
9	ClassCourseID	1183571	Character	A unique ID identifying a state course code
10	GradeLevelCode	16	Numeric	Grade level for the course enrollment reported in this record
11	GenderCode	3	Numeric	The gender for the course enrollment reported in this record
12	EnrollNoEthRpt	58	Numeric	The number of students enrolled in a particular course no ethnicity/race reported.
13	EnrollAmind	17	Numeric	The number of American Indian or native Alaskan, not Hispanic, students enrolled in the particular course
14	EnrollAsian	72	Numeric	Number of Asian, not Hispanic, students enrolled in the particular course
15	EnrollPacIsl	13	Numeric	Number of Pacific Islander, not Hispanic, students enrolled in the particular course.
16	EnrollFilipino	31	Numeric	Number of Filipino, not Hispanic, students enrolled in the particular course.
17	EnrollHispanic	135	Numeric	Number of Hispanic or Latino students enrolled in the particular course.
18	EnrollAfrAm	55	Numeric	Number of African American, not Hispanic, students enrolled in the particular course.
19	EnrollWhite	95	Numeric	Number of white, not Hispanic, students enrolled in the particular course.
20	EnrollTwoOrMore	36	Numeric	Number of students reporting two or more races, not Hispanic, enrolled in the particular course.
21	EnrollTotal	219	Numeric	Number of all students, enrolled in the particular course.
22	EnrollEL	74	Numeric	Number of all English learner students, enrolled in the particular course.
23	FileCreated	1	Date	The file creation date

Our mappings are based on the assumption of possible privileges the subject(s) can have over specific resource(s). For example, we interpret "review" in rule 1 to open, edit, and delete actions because we assumed that a review will require an open action. Also, while reviewing, there may be a need to edit or delete the object. The list below outlines the rules:

1) Student Enrolment is reviewed by Administration staff to ensure that it has been completed correctly.
2) Academic and nominated administrative staff will be able to view and maintain student course enrolments.
3) An Admin can view, edit and delete the course enrollment of a student from the same county.
4) An Admin can view, edit and delete the course enrollment of a student from the same district.
5) A Teacher can view the record of students from the same school if he has a tenured position.
6) A Teacher can view the record of students if he has a tenured position and assigned the same course code as the object.
7) A Teacher can view the record of students if he has a tenured position and assigned the same class ID as the object.
8) A Teacher can view the record of students from the same school if he has a long term position.
9) A Teacher can view the record of students if he has a long term position and assigned the same course code as the object.
10) A Teacher can view the record of students if he has a long position and assigned the same class ID as the object.

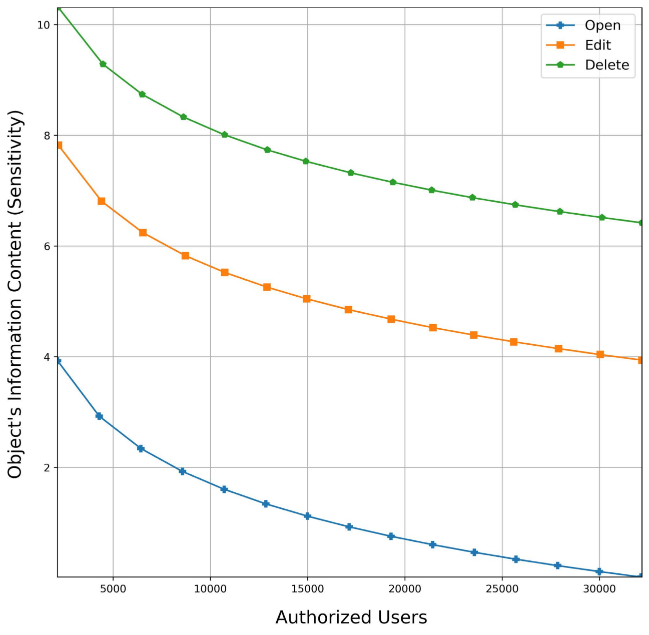


FIGURE 4. The distribution of sensitivity scores versus the probability of authorized users for different access rights as calculated.

11) A Teacher can view the record of students from the same school if he has a probationary position.
12) A Teacher can view the record of students if he has a probationary position and assigned the same course code as the object.
13) A Teacher can view the record of students if he has a probationary position and assigned the same class ID as the object.

B. IMPLEMENTATION

We implemented ABAC-related components of the proposed framework using *Py_ABAC*,[3] an open-source attribute-based access control toolkit written in the Python programming language. The toolkit gives a fine-grained control on the definition of ABAC rules that restrict access to resources. Then, we defined ABAC rules using a JSON-based Policy Language. We also used SQL along with *SQLAlchemy* database extension to store users' and objects' attributes and policies. In addition, we used Flask Python web application framework; a lightweight WSGI web application framework to an create access endpoint for users. To build the honey attribute generator model, we used Python programming language in Jupyter notebook environment. We note that we run all the experiments on AMD Ryzen 5 2600 Six-Core processor machine.

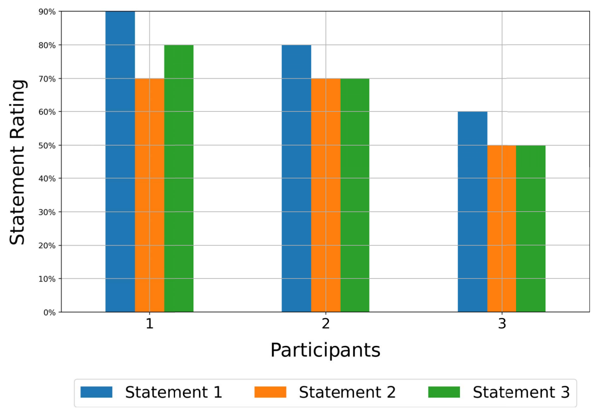
C. EVALUATION

To evaluate the performance of sensitivity estimator (Q1), we used the subject and object datasets as discussed in section VI-A. Intuitively, the higher the number of authorized users, the less confidential the object is, and hence the less the sensitivity. On the other hand, the higher the privilege, the higher the risk, and hence the higher the sensitivity. Therefore, we examined the effect of increasing the number of authorized users on the sensitivity scores of a resource attribute *A*. We also evaluated the change in the sensitivity scores for various access right which are open, edit, and delete. Access rights are granted to users in a hierarchical structure. For example, open access requires the least privileges, while edit access requires higher privileges and more trusted users. On the other hand, delete access requires the highest privileges of all access rights as it may cause some data/information to disappear. Note that, a user with edit access implicitly has open access to the same resource. Similarly, delete access indicates the user's right to open and edit the resource as well. Figure 4 shows the distribution of sensitivity scores versus the normalized form of authorized users for different access rights. Two observation can be drawn from the figure: (1) Actions that require higher privileges produces higher sensitivity scores. In our case, the sensitivity scores of an attribute targeted with delete access are higher than when targeted with edit access. And the scores obtained in the case of edit access are higher than scores with open access; (2) the higher the number of authorized users, the lower the sensitivity. These observations go inline with our initial intuition and suggest the effectiveness of using information content as a measure of the sensitivity.

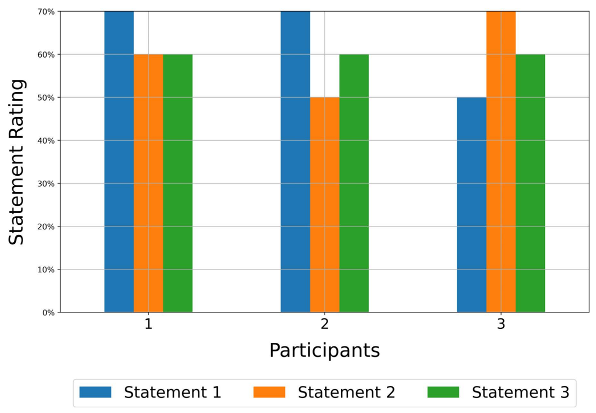
To address (Q2), we evaluated the performance of the proposed honey attributes generator component with respect to the indistinguishability between real and honey attributes. To quantify the indistinguishability, we measured the semantic similarity between pairs of honey and real values using GLOVE embedding. In our experiments, we run the generator algorithm (See Algorithm 2) 5 times on 75000 samples collected with random sampling technique from our object dataset. We note that we experimented with various Glove models and embedding dimensions to represent attributes values. Table 4 shows the basic statistics of the resulting similarity scores across 5 runs. The table shows that Glove 840B 300D have the highest mean, median, and standard deviation. This indicates that the similarity scores with Glove 840B 300D are centered around the mean value and therefore more effective. Based on this, we selected Glove 840B 300D as the best model and used it to generate the honey attributes. Table 5 shows examples of real and honey attribute values as well as the similarity scores generated using Glove 840B 300D word embedding as explained in section V-C. Note that we only show 5 values due to the space limit.

In addition, we carried out a pilot user study on a sample of the generated values to examine their indistinguishability from the truth. In the study, we compiled 20 real and honey values of two attributes, namely "school name"and "school district". We captioned them as *A* and *B*. Then, we presented the attributes to our three participants who were tasked to rate the following statements on a scale of 1 to 10, where 10 indicates strongly agree:

[3]https://py-abac.readthedocs.io/en/latest/



(a) Question 1



(b) Question 2

FIGURE 5. Survey results.

TABLE 4. Similarity of honey attributes.

S/N	Model	Mean	Median	Standard Deviation
1	Glove 840B 300D	0.9014	0.90683	0.0227
2	Glove 42B 300D	0.8772	0.8775	0.0071
3	Glove 6B 300D	0.8845	0.8844	0.0009
4	Glove 6B 200D	0.8869	0.8871	0.0028
5	Glove 6B 100D	0.8854	0.8036	0.0042
6	Glove 27B 200D	0.8837	0.8829	0.0080
7	Glove 27B 100D	0.8946	0.8941	0.0053
8	Glove 27B 50D	0.8872	0.8867	0.0011

TABLE 5. Real attributes, honey versions, and similarity scores.

S/N	Real Attribute	Honey Attribute
1	Cache Creek High (Continuation)	{0.9092: '(Proceeding) High Creek Cache'}
2	George Ellery Hale Charter Academy	{0.0.9527: 'Academy Charter Hale Ellery Leonard'}
3	Grover Cleveland Charter High	{0.9032: 'Heights Charter Cleveland Grover'}
4	Opportunities for Learning - Baldwin Park	{0.8900: 'Boulevard Baldwin - Learning for Opportunities'}
5	Phineas Banning Senior High	{0.8786: 'Grade Senior Banning Phineas'}

- Only A could be a valid value for a "school name"/ "school district"
- Only B could be a valid value for a "school name"/ "school district"
- Both A and B could be valid values for a "school name"/ "school district"

Figure 5a and 5b show ratings distributions by each participant for the two attributes. The varying values of each rating reveal that none of the participants could completely distinguish the actual attributes from the honey versions. For example, in figure 5a, participant 1 gave 90% agreement that only A could be a valid value for a school name. Also, participant 1 gave 80% agreement that both A and B could be valid values for a school name. This reveals that the real and honey attributes are indistinguishable.

TABLE 6. Overhead of sensitivity estimator.

S/N	Subject	Time(s)
1	2000	2.0731
2	4000	4.1921
3	6000	5.4501
4	8000	8.0365
5	10000	10.1476
6	12000	12.2749
7	14000	15.0184
8	16000	17.1985
9	18000	18.5376
10	20000	21.4582

To evaluate the complexity overhead induced by deception-related components (Q3). Table 6 shows the overhead added by the sensitivity estimator. The time complexity increases as the number of subjects requesting access grows because the access rule analyses every request from the subjects.The honey attribute generator adds an overhead of 2.08 seconds to generate one honey attribute using our selected model, Glove 840B 300D. As expected, the overhead increases as the number of attributes grows. Therefore, we limit this process to the sensitive objects to reduce the cumulative overhead to the system.

VII. DISCUSSION AND FUTURE DIRECTIONS

While the present study was carefully designed and implemented, it is not without limitations. In particular, we did not find any publicly available self-contained real-world dataset to carry out our experiments. However, the data concern is not unique to this problem, but rather is common to studies in the field of cybersecurity. To address this issue, we gathered elements of the required dataset from multiple real-world resources to construct a realistic dataset. The time complexity of the proposed approach is another potential limitation. We observed that the execution time of our approach increases as the number of objects' attributes grows because the sensitivity estimator analyzes each attribute separately. Similarly, the execution time increases as the number of subjects grows because the monitoring unit has more subjects to check. Although these increases may not be regarded as limitations, they may lead to scalability issues when used in large organizations with very large numbers of subjects and objects.

Our future work plan is focused on improving the proposed work in several directions. First, we will work on improving scalability of the proposed framework by developing more efficient components. Second, we will explore additional approaches for generating the honey attributes. One such approach would be utilizing Generative Adversarial Networks (GANs) which have shown significant results in a variety of domains. The GAN based approaches could improve quality of the generated honey attributes and result in better indistinguishability. Third, in a situation where the insider is aware of the deception, moving target defense [45] could provide another layer for defense. We will investigate integrating moving target defense approaches into the extended ABAC framework to increase the insider's cost to guess the actual attribute from the honey attribute values. We also plan to conduct a comprehensive user study to empirically understand users' behaviour while interacting with the proposed system.

VIII. CONCLUSION

In this paper, we proposed an approach for detecting insider threats by integrating cyber deception into standard Attribute-based Access Control (ABAC) framework. We introduced the notion of honey attributes to protect sensitive attributes in the system as opposed to all the attributes to mitigate the overhead. We extended the standard ABAC model with deception-related components such as sensitivity estimation component, honey attribute generator, and monitoring unit. We also proposed an approach based on genetic algorithm to generate honey attributes values. The findings of our implementation and experimental results are manifold. First, the attributes of objects in ABAC are not equally sensitive. Therefore, the sensitivity estimator can successfully identify the highly sensitive attributes. Second, the probability that insiders will access an object through sensitive attributes is high. Therefore, producing honey versions for sensitive attributes is more effective. Third, actions that require higher privileges produce higher sensitivity scores. Fourth, the higher the number of authorized users, the lower the sensitivity. Fifth, the results reveal that the genetic algorithm is a powerful technique for producing honey attributes that are indistinguishable from real attributes.

REFERENCES

[1] V. Hu, D. Ferraiolo, D. Kuhn, A. Schnitzer, K. Sandlin, R. K. Miller, and K. Scarfone, "Guide to attribute based access control (ABAC) definition and considerations," *Nat. Inst. Standards Technol. Special Publication*, vol. 800, pp. 1–54, Jan. 2014.

[2] C. R. Brown, A. Watkins, and F. L. Greitzer, "Predicting insider threat risks through linguistic analysis of electronic communication," in *Proc. 46th Hawaii Int. Conf. Syst. Sci.*, Jan. 2013, pp. 1849–1858.

[3] K. Paxton-Fear, D. Hodges, and O. Buckley, "Understanding insider threat attacks using natural language processing: Automatically mapping organic narrative reports to existing insider threat frameworks," in *Proc. 2nd Int. Conf. HCI Cybersecur., Privacy Trust (HCI-CPT), Held Part 22nd HCI Int. Conf. HCII*, Copenhagen, Denmark, Jul. 2020, pp. 619–636.

[4] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *Proc. IEEE Secur. Privacy Workshops*, May 2013, pp. 98–104.

[5] R. Chinchani, A. Iyer, H. Q. Ngo, and S. Upadhyaya, "Towards a theory of insider threat assessment," in *Proc. Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2005, pp. 108–117.

[6] Securonix Securonix. (Apr. 2020). *2020 Insider Threat Report Securonix*. [Online]. Available: https://www.securonix.com/resources/2020-insider-threat-report/

[7] C. Insiders. (Apr. 2020). *2020 Insider Threat Report Cybersecurity Insiders*. [Online]. Available: https://www.cybersecurity-insiders.com/portfolio/2020-insider-threat-report-darktrace/

[8] T. E. Senator, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow, and I. Essa, "Senator TE detecting insider threats in a real corporate database of computer usage activity," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, pp. 1393–1401 (2013), doi: 10.1145/2487575.2488213.

[9] A. Kim, J. Oh, J. Ryu, and K. Lee, "A review of insider threat detection approaches with IoT perspective," *IEEE Access*, vol. 8, pp. 78847–78867, 2020.

[10] T. D. Schneider, "Information content of individual genetic sequences," *J. Theor. Biol.*, vol. 189, no. 4, pp. 427–441, Dec. 1997.

[11] R. A. Alsowail and T. Al-Shehari, "Empirical detection techniques of insider threat incidents," *IEEE Access*, vol. 8, pp. 78385–78402, 2020.

[12] R. E. Deakin and D. G. Kildea, "A note on standard deviation and RMS," *Austral. Surveyor*, vol. 44, no. 1, pp. 74–79, Jun. 1999.

[13] J. E. Tapiador and J. A. Clark, "Masquerade mimicry attack detection: A randomised approach," *Comput. Secur.*, vol. 30, no. 5, pp. 297–310, Jul. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404811000654

[14] F. L. Greitzer, L. J. Kangas, C. F. Noonan, A. C. Dalton, and R. E. Hohimer, "Identifying at-risk employees: Modeling psychosocial precursors of potential insider threats," in *Proc. 45th Hawaii Int. Conf. Syst. Sci.*, Jan. 2012, pp. 2392–2401.

[15] Y. Hashem, H. Takabi, M. Ghasemigol, and R. Dantu, "Towards insider threat detection using psychophysiological signals," in *Proc. 7th ACM CCS Int. Workshop Manag. Insider Secur. Threats*, Oct. 2015, pp. 71–74, doi: 10.1145/2808783.2808792.

[16] O. Brdiczka, J. Liu, B. Price, J. Shen, A. Patil, R. Chow, E. Bart, and N. Ducheneaut, "Proactive insider threat detection through graph learning and psychological context," in *Proc. IEEE Symp. Secur. Privacy Workshops*, May 2012, pp. 142–149.

[17] Y. Hashem, H. Takabi, R. Dantu, and R. Nielsen, "A multi-modal neuro-physiological study of malicious insider threats," in *Proc. Int. Workshop Manag. Insider Secur. Threats*, Oct. 2017, pp. 33–44, doi: 10.1145/3139923.3139930.

[18] J. Jiang, J. Chen, T. Gu, K.-K.-R. Choo, C. Liu, M. Yu, W. Huang, and P. Mohapatra, "Anomaly detection with graph convolutional networks for insider threat and fraud detection," in *Proc. MILCOM IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2019, pp. 109–114.

[19] J. Yuill, M. Zappe, D. Denning, and F. Feer, "Honeyfiles: Deceptive files for intrusion detection," in *Proc. From 5th Annu. IEEE SMC Inf. Assurance Workshop*, Jun. 2004, pp. 116–122.

[20] B. Bowen, S. Hershkop, A. S. Keromytis, and S. J. Stolfo, "Baiting inside attackers using decoy documents," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, 2009, pp. 51–70.

[21] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis, "Towards systematic honeytoken fingerprinting," in *Proc. 13th Int. Conf. Secur. Inf. Netw.*, Nov. 2020, pp. 1–5, doi: 10.1145/3433174.3433599.

[22] K. S. M. Moe and T. Win, "Enhanced honey encryption algorithm for increasing message space against brute force attack," in *Proc. 15th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, Jul. 2018, pp. 86–89, doi: 10.1109/ECTICon.2018.8620050.

[23] P. Kaghazgaran and H. Takabi, "Toward an insider threat detection framework using honey permissions," *J. Internet Serv. Inf. Secur.*, vol. 5, pp. 19–36 Aug. 2015.

[24] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 145–160, doi: 10.1145/2508859.2516671.

[25] N. Bhagat and B. Arora, "Intrusion detection using honeypots," in *Proc. 5th Int. Conf. Parallel, Distrib. Grid Comput. (PDGC)*, Dec. 2018, pp. 412–417.

[26] M. Bercovitch, M. Renford, L. Hasson, A. Shabtai, L. Rokach, and Y. Elovici, "HoneyGen: An automated honeytokens generator," in *Proc. IEEE Int. Conf. Intell. Secur. Informat.*, Jul. 2011, pp. 131–136.

[27] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Automated insider threat detection system using user and role-based profile assessment," *IEEE Syst. J.*, vol. 11, no. 2, pp. 503–512, Jun. 2015.

[28] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, 1998.

[29] (2021). *Teacher Credentialing Assignment Unit, C. Administrator's Assignment Manual*. [Online]. Available: https://www.ctc.ca.gov/docs/default-source/credentials/manuals-handbooks/administrator-assignment-manual.pdf

[30] (2020). *Education, C. CALPADS Data Guide: A Guide for Program Staff*. [Online]. Available: https://documentation.calpads.org/Support/docs/CALPADSDataGuide.docx

[31] (Nov. 2018). *Education, C. Staff Assignment and Course Data*. [Online]. Available: https://www.cde.ca.gov/ds/ad/filesassign.asp

[32] (Nov. 1976). *Education, C. Education Code Privacy of Pupil Records*. [Online]. Available: https://leginfo.legislature.ca.gov/

[33] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992. [Online]. Available: https://www.jstor.org/stable/24939139?seq=1#metadata_info_tab_contents

[34] J. Pennington, R. C. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[35] E. Grave, P. Bojanowski, P. Gupta, A. T. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proc. Int. Conf. Lang. Resour. Eval. (LREC)*, 2018, pp. 1–5.

[36] *Word2Vec*. (2022). Accessed: Mar. 26, 2022. [Online]. Available: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/word2vec.html

[37] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[38] G. Ancora, G. Palli, and C. Melchiorri, "A hybrid genetic algorithm for pallet loading in real-world applications," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 10006–10010, 2020.

[39] E. Semenkin and M. Semenkina, "Self-configuring genetic programming algorithm with modified uniform crossover," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–6.

[40] X. B. Hu and E. D. Paolo, "An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 55–62.

[41] M. T. Ahvanooey, Q. Li, M. Wu, and S. Wang, "A survey of genetic programming and its applications," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 4, pp. 1765–1793, Apr. 2019.

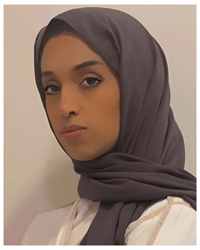
[42] J. Devlin, M. Chang, K. K. Lee, and T. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*. 2019, pp. 1–16.

[43] *Tokenization*. Accessed: Apr. 4, 2022. [Online]. Available: https://www.techopedia.com/definition/13698/tokenization

[44] H. Li, Y. Guo, S. Huo, H. Hu, and P. Sun, "Defensive deception framework against reconnaissance attacks in the cloud with deep reinforcement learning," *Sci. China Inf. Sci.*, vol. 65, no. 7, pp. 1–19, Jul. 2022.

[45] S. Jajodia, A. Ghosh, V. Swarup, C. X. Wang, and X. S. Wang, *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Cham, Switzerland: Springer, 2011.

[46] Y. Hashem, H. Takabi, M. R. GhasemiGol, and R. Dantu, "Inside the mind of the insider: Towards insider threat detection using psychophysiological signals," *J. Internet Services Inf. Secur. (JISIS)*, vol. 6, pp. 20–36, Feb. 2016.



MANAR ALOHALY received the Ph.D. degree from the University of North Texas, Denton, TX, USA, in 2020. She is currently an Assistant Professor in cybersecurity at Princess Nourah Bint Abdulrahman University (PNU), Riyadh, Saudi Arabia, where she is also the Director of the Innovation Center, College of Computer and Information Sciences. Her research interests include access control, usable privacy and security, cyber deception, insider threat detection, applied machine learning, and natural language processing on security-related topics. She is a member of the technical program committee for several international conferences. She is also a member of the Information Security Association-Hemaya.

OLUSESI BALOGUN (Student Member, IEEE) received the B.S. degree in computer engineering from Obafemi Awolowo University, Nigeria, in 2014. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Georgia State University, Atlanta, Georgia, USA. He works as a Research Assistant under the supervision of Dr. Daniel Takabi at the Information Security and Privacy: Interdisciplinary Research and Education Center, which is designated as the National Center of Academic Excellence in Cyber Defense Research. His research interests include insider threat detection, moving target defense, access control models, and security and privacy in cyber-physical systems (CPS). He also works as a Teaching Fellow at the Department of Computer Science, Georgia State University, Atlanta, Georgia, USA. He is a Student Member of ACM. He received the National Diploma (ND) Certificate in computer engineering from The Federal Polytechnic, Ilaro, Nigeria, in 2008.

DANIEL TAKABI (Member, IEEE) received the Ph.D. degree from the University of Pittsburgh, Pittsburgh, PA, USA, in 2013. He is currently an Associate Professor in computer science and the Next Generation Scholar with Georgia State University, Atlanta, GA, USA. He is also the Founding Director of the Information Security and Privacy: Interdisciplinary Research and Education Center, which is designated as the National Center of Academic Excellence in Cyber Defense Research. His research interests include various aspects of cybersecurity and privacy, including privacy-preserving machine learning, adversarial machine learning, advanced access control models, insider threats, and usable security and privacy.