Mitigating Privilege Misuse in Access Control through Anomaly Detection

Mitigating Privilege Misuse in Access Control through Anomaly Detection

Gelareh Hasel Mehri
Eindhoven University of Technology
Eindhoven, The Netherlands
g.hasel.mehri@tue.nl

Inez L. Wester
Eindhoven University of Technology
Eindhoven, The Netherlands
i.l.wester@student.tue.nl

Federica Paci
University of Verona
Verona, Italy
n.zannone@tue.nl

Nicola Zannone
Eindhoven University of Technology
Eindhoven, The Netherlands
n.zannone@tue.nl

ABSTRACT

Access control is a fundamental component of IT systems to guarantee the confidentiality and integrity of sensitive resources. However, access control systems have inherent limitations: once permissions have been assigned to users, access control systems do not provide any means to prevent users from misusing such permissions. The problem of privilege misuse is typically addressed by employing auditing mechanisms, which verify users' activities a posteriori. However, auditing does not allow for the timely detection and mitigation of privilege misuse. In this work, we propose a framework that complements access control with anomaly detection for the run-time monitoring of access requests and raises an alert when a user diverges from her normal access behavior. To detect anomalous access requests, we propose a novel approach to build user profiles by eliciting patterns of typical access behavior from historical access data. We evaluated our framework using the access log of a hospital. The results show that our framework has very few false positives and can detect several attack scenarios.

CCS CONCEPTS

• **Security and privacy → Access control**; • **Computing methodologies → Anomaly detection**.

KEYWORDS

Privilege misuse; User profiling; Behavioral patterns.

ACM Reference Format:
Gelareh Hasel Mehri, Inez L. Wester, Federica Paci, and Nicola Zannone. 2023. Mitigating Privilege Misuse in Access Control through Anomaly Detection. In *The 18th International Conference on Availability, Reliability and Security (ARES 2023), August 29–September 01, 2023, Benevento, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3600160.3604988

ARES 2023, August 29–September 01, 2023, Benevento, Italy
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0772-8/23/08.
https://doi.org/10.1145/3600160.3604988

1 INTRODUCTION

Access control systems are typically employed as the first line of defense to guarantee the confidentiality and integrity of sensitive resources and data. In particular, access control systems ensure that only authorized users can access resources based on predefined access control policies specifying users' permissions.

The level of security offered by an access control system mainly depends on the correctness of the employed access control policies. To this end, a large body of research has proposed principles to guide the specification of access control policies (e.g., least privilege and separation of duties) [12] as well as tools for their verification [32]. Despite these research efforts, once access privileges are assigned to a user, there is no guarantee that the user will not misuse them.

Privilege misuse is one of the foremost causes of data breaches [34], where the privileges associated with a particular user are used inappropriately or fraudulently to steal sensitive information or sabotage the system. Privilege misuse is typically carried out by insider threats, i.e. legitimate users of the system that exploit the assigned permissions, or by cyber attackers obtaining the credentials of a privileged user. Another example of privilege misuse comes from the medical domain, where most hospitals use the Break-The-Glass (BTG) protocol to handle access to patient data in emergencies. This protocol allows a user (e.g., a doctor) to override the access control policies in force and gain access to the requested data [2]. However, the unrestricted access granted by the BTG protocol leaves the system vulnerable to data breaches because users can exploit the protocol to obtain access to patient data that they normally cannot access.

The detection of privilege misuse is typically addressed using auditing methods for the a-posteriori analysis of system access logs to identify data accesses corresponding to privilege misuse [3, 14, 22, 26, 27]. Unfortunately, these methods do not proactively protect against misuse but only help identify one after the fact. A few works propose approaches for identifying unsafe data accesses at run-time based on a risk score [4, 28] or by eliciting ad-hoc constraints from access logs [3]. However, these approaches merely block potentially unsafe requests, which is undesirable in mission-critical systems such as hospitals where lives are at stake, or introduce a significant burden in policy administration.

To provide a proactive solution for detecting privilege misuse, in this work, we propose a novel framework that complements access control with anomaly detection for run-time monitoring and timely detection of privilege misuse. Our anomaly detection system checks if access requests deviate from the requester's typical behavior and, if so, triggers an alert indicating potential misuse of privileges. To this end, we propose a new method to build user profiles that encompass the usual access behaviors of individual users. In particular, we extract access patterns by applying clustering to access events recorded in access logs. We evaluated the effectiveness of our framework using a real-life dataset recording data accesses at a hospital.

The main contribution of our work can be summarized as follows:

- Our framework provides an additional layer of protection by leveraging anomaly detection to determine at run-time whether access requests authorized by the access control system pose risks of privilege misuse. Contrary to approaches based on auditing, our framework allows for the timely detection and mitigation of privilege misuse. Moreover, by flagging anomalous requests, rather than denying them altogether, our framework allows the employment of different mitigation strategies based on the context of use and application domain (e.g., requiring the requester's supervisor to confirm the request's legitimacy).
- Our approach for user profiling allows identifying behavioral patterns able to discriminate users (even with the same role) based on their access behaviors, thus providing fine-grained detection capabilities. Moreover, our approach can construct a profile also for those users for whom historical data are scarce.
- Our evaluation shows that our framework returns very few false positives (false positive rate <0.001) and can detect many attack scenarios representative of privilege misuse.

The remainder of the paper is structured as follows. Section 2 discusses related work. Section 3 presents our framework for user profiling and run-time monitoring of access requests. Section 4 describes a prototype implementation of our framework based on a real access log of a hospital, and Section 5 presents its evaluation. Section 6 discusses our findings, and Section 7 concludes the paper.

2 RELATED WORK

This work proposes an anomaly detection system that leverages machine learning (ML) and, specifically, clustering to build user profiles enabling the timely detection of privilege misuse. Next, we discuss related research on the use of ML techniques and anomaly detection in the context of access control.

ML-based access control. An extensive body of research has applied ML for the mining and engineering of attribute-based access control (ABAC) policies [1, 6, 7, 19, 20]. These works extract ABAC policy rules from access logs using different types of ML algorithms: association rule mining [1], support vector machines (SVM) [7], decision trees [6, 7], and k-modes clustering [20]. However, access logs are typically incomplete [33] and, thus, the mined policies might not fully capture the access constraints in place. Other works leverage ML to manage changes to access control policies [17, 35]. Xiang et al. propose P-DIFF [35], a tool that helps system administrators detect unintended changes to access control policies. This tool uses decision trees to infer access control policies from access logs. Then, when an access request that was denied based on the policy inferred from the logs is now granted by the enforced policy, P-DIFF notifies the system administrator, who can validate the policy change. Gumma et al. propose PAMMELA [17], a methodology that leverages supervised learning to augment an existing access control policy with access rules that accommodate organizational changes. The last body of research in this area proposes to replace policy-based evaluation engines with an ML model [10, 24, 25]. Chang et al. [10] propose a time-constrained access control model based on SVM. Liu et al. [24] propose EPDE-ML, an ABAC engine based on Random Forest. Nobi et al. [25], instead, present an access decision-making engine based on neural networks. However, ML models often generate false positives and false negatives, which can result in data breaches or affect business continuity.

Anomalous user behavior in access control. Some works have investigated methods for identifying possible unsafe data access, especially for uncovering privilege misuses [4, 28]. For instance, Srivastava et al. [28] present a risk-adaptive role-based access control model for hospital management systems, which uses Random Forest to determine if a request is legitimate or not. Differently from other works relying on ML for access decision-making [10, 24, 25], the decision is based on a risk score computed using contextual features extracted from access logs, such as time of access, location, type of request (emergency or normal), and previous history (number of requests the user made for the same data in the past), and the relevance of the requested data to the requester. However, this approach generates a very high number of false negatives (19%, as reported in [28]), making it unsuitable in practical situations. Similarly, Baracaldo et al. [4] present a framework that extends RBAC to account for the risk of permissions being misused by a role and the trust the system has in its users based on their past behavior. When a user's trust falls below a certain threshold, the framework removes her privileges. While this approach has the potential to prevent privilege misuse, it lacks the flexibility necessary to handle misclassifications. Another line of research investigates solutions based on behavioral models and anomaly detection techniques [2, 3, 30]. For instance, Alizadeh et al. [2] and Tasali et al. [30] propose approaches to detect anomalous access requests related to the use of the Break-The-Glass (BTG) protocol. However, these works only support the auditing of access requests in an a-posteriori fashion to identify misuses of the BTG protocol. On the other hand, Argento et al. [3] propose to monitor user behavior at run-time to learn behavioral profiles of users accessing resources. However, this work focuses on machine-assisted policy administration, where behavioral profiles are used to refine the access control rules in place. Yet, misclassification can result in too stringent constraints, which can affect the correct functioning of the system.

Contribution. Compared to previous works, our framework does not aim to replace traditional access control systems with an ML-based system, which might result in data breaches due to false positives, or to mine ad-hoc constraints to handle behavioral aspects, which might be too inflexible in critical-mission systems and introduce burden in policy administration. Rather, our framework aims to enhance existing access control systems with a monitoring component that raises an alert when an anomalous access request is detected, thus effectively providing an additional layer of protection. Conversely to auditing methods that rely on a posteriori analysis of access logs, our framework enables the detection of anomalous requests at run-time, allowing the access control system to promptly react to privilege misuses by applying strategies to prevent or mitigate the risk of abuse.

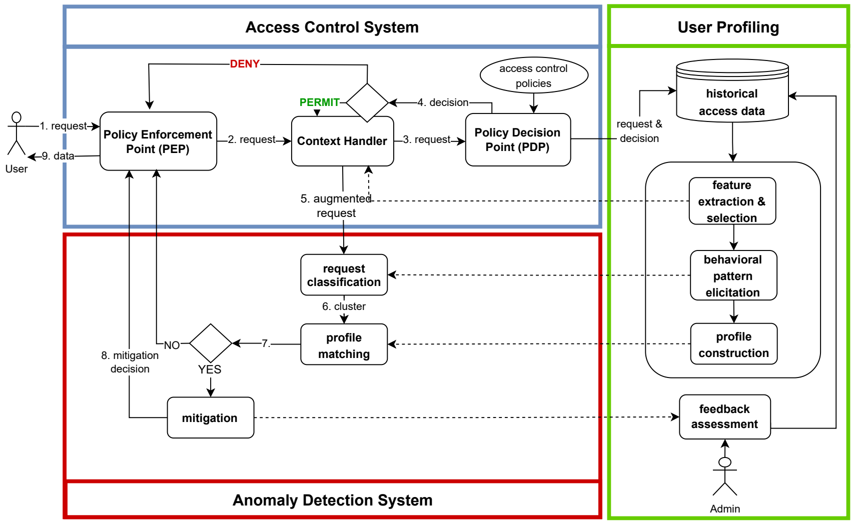


Figure 1: System overview

3 OUR APPROACH

This section presents our framework, which augments a traditional access control system with an anomaly detection system to monitor access requests at run-time and provide timely feedback on anomalous requests. Next, we introduce the underlying threat model; we then present the system architecture and its main components.

3.1 Threat Model

Our framework aims to detect situations in which the permissions associated with a user are exploited to perform malicious actions. We distinguish between two categories of attackers: *insider threats* abusing the assigned permissions to exfiltrate confidential or sensitive information and *external attackers* who have stolen the credentials (username and password) of a legitimate user (e.g., via a phishing attack) and impersonate the user to steal sensitive data. Contrarily to approaches based on auditing [2, 30], our framework allows for the timely detection of anomalous requests representing the abuse of privileges from insider threats or external attackers.

3.2 System Overview

The proposed system consists of three main components depicted in Fig. 1. The blue box represents the access control component, which follows the traditional functioning of access control systems. When a user requests access to sensitive information, the request is intercepted by the policy enforcement point (PEP), which forwards it to Context Handler (CH). The CH augments the request with attributes of the user, the resource, the action, or the environment, which might be required for policy evaluation. The request is then forwarded to the policy decision point (PDP), which evaluates it against the security policies to determine whether access should be granted.

In a traditional access control system, the decision (*permit* or *deny*) is sent to the PEP for its enforcement. If the request is evaluated to *permit* (or a *deny* decision is overwritten, for instance, using the BTG protocol in hospitals), access to the requested information is granted to the user. However, this approach does not account for potential risks where a user has abused his privileges or where an external attacker has impersonated a legitimate user of the system.

To mitigate these risks, we complement the access control system with an anomaly detection system (red box in Fig. 1). In particular, the anomaly detection system evaluates the access request at runtime by checking whether the request matches the typical access behavior of the user. If the request is deemed anomalous for the user, access can be denied altogether, or additional controls can be employed to mitigate the risk of misuse (e.g., requiring the user's supervisor to confirm the request's legitimacy). Although an investigation and definition of possible mitigation strategies are outside the scope of this work, we will discuss them in Section 6.

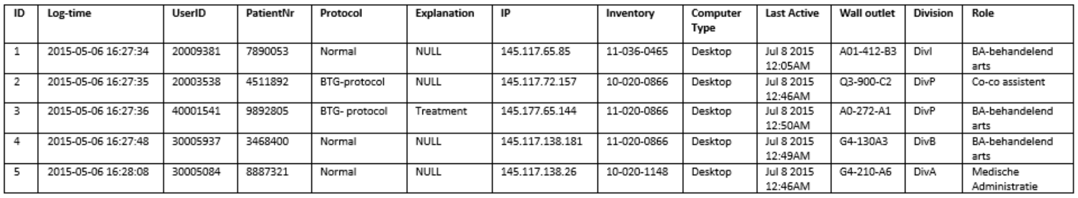
The anomaly detection system relies on *user profiles* representing the typical access behavior of users within the system. These profiles are constructed from historical access data by identifying patterns of access behavior, hereafter referred to as *behavioral patterns* (green box in Fig. 1). We assume behavioral patterns and user profiles are regularly updated to capture changes in user behavior over time, e.g. when a user is assigned to a new job function.

In the remainder of the section, we discuss the approach employed for building user profiles and detecting anomalous requests.

3.3 User Profiling

Our framework relies on user profiles representing the users' normal access behavior to detect anomalous requests. In this section, we describe the approach employed for user profiling, which encompasses selecting and extracting relevant features, eliciting behavioral patterns from historical data, and constructing user profiles from the extracted behavioral patterns.

3.3.1 Feature Selection and Extraction. Anomaly detection systems aim to discover behaviors that "stand out of the ordinary". To this end, an essential step is to identify the features that allow capturing 'interesting' user behaviors. In our context, a *feature* is an individual measurable property or a characteristic of access events. The choice of relevant features mainly depends on the data at hand and the behaviors to be captured. In this work, we are interested in *contextual*



ID	Log-time	UserID	PatientNr	Protocol	Explanation	IP	Inventory	Computer Type	Last Active	Wall outlet	Division	Role
1	2015-05-06 16:27:34	20009381	7890053	Normal	NULL	145.117.65.85	11-036-0465	Desktop	Jul 8 2015 12:05AM	A01-412-B3	DivI	BA-behandelend arts
2	2015-05-06 16:27:35	20003538	4511892	BTG-protocol	NULL	145.117.72.157	10-020-0866	Desktop	Jul 8 2015 12:46AM	Q3-900-C2	DivP	Co-co assistent
3	2015-05-06 16:27:36	40001541	9892805	BTG- protocol	Treatment	145.177.65.144	11-020-0866	Desktop	Jul 8 2015 12:50AM	A0-272-A1	DivP	BA-behandelend arts
4	2015-05-06 16:27:48	30005937	3468400	Normal	NULL	145.117.138.181	11-020-0866	Desktop	Jul 8 2015 12:49AM	G4-130A3	DivB	BA-behandelend arts
5	2015-05-06 16:28:08	30005084	8887321	Normal	NULL	145.117.138.26	10-020-1148	Desktop	Jul 8 2015 12:46AM	G4-210-A6	DivA	Medische Administratie

Figure 2: Excerpt of raw data logs

features that capture the behaviors exhibited by users accessing resources within the systems and might signal privilege misuse.

A feature can be an attribute recorded in the historical data or can be extracted from historical data to characterize relevant user behaviors. Examples of contextual features that can be derived directly from historical data could be the time of the access, the location from which it was made, or the type of access. The location from which the request originated can be used, for instance, to detect an external attacker impersonating a legitimate user who does not have physical access to the organization's buildings. It is worth noting that these contextual features are often recorded in access logs. As an example, Fig. 2 shows the excerpt of a real-life log recording accesses to patient data at a hospital, which provides insights into when a request was made (*log-time*) and the location from where it originated (*IP* and *wall-outlet*) (we refer to Section 4.1 for details on the log).

Other contextual features, such as the number of accesses and amount of accessed data, can be used to detect privilege misuse [3]. For instance, access to abnormal amounts of data can indicate an attempt at data exfiltration. The definition of these features is often based on common sense and domain knowledge and typically depends on the information available in the raw data. We discuss the feature extraction process on a real-life log in Section 4.2.

Although several features can be conceived for anomaly detection, employing a large feature set does not necessarily imply better results. High-dimensional data typically affect the performance of machine learning algorithms and create extremely complex models with often lower accuracy [9]. We discuss the selection of features that allow eliciting reliable behavioral patterns in Section 4.2.

3.3.2 Behavioral Pattern Elicitation. The next step is to extract patterns of typical access behavior from historical access data based on the identified features. As historical data are typically unlabeled, i.e., the behavioral patterns of interest are unknown a priori, we rely on an unsupervised learning approach and, particularly, on clustering.

One of the most used clustering algorithms is K-means [23]. K-means is a relatively generic algorithm that is easily adaptable to new features. However, K-means only supports numerical features. As the features of interests can be both numerical and categorical (see Section 4.2), we employ K-prototypes [13] to extract behavioral patterns. This clustering algorithm extends K-modes [18], a variant of K-means tailored to categorical data, to handle both categorical and numerical features.[1] Every data point is assigned to the cluster where its distance to the mean of that cluster is the smallest, for a given distance metric. K-prototypes uses the Gower distance to determine a numerical value for categorical data by comparing the categorical values to the numerical data already in the dataset [16]. Similarly to K-means, K-prototypes requires the number of clusters as an input parameter. Following best practices [29], we employ the elbow method to determine the optimal number of clusters.

We also considered the mean-shift algorithm [11] and the Density-Based Spatial Clustering of Application with Noise (DBSCAN) algorithm [31] in our study. The mean-shift algorithm shifts the clusters to higher-density clusters iteratively; however, it does not work well with high dimensional data [8]. On the other hand, DBSCAN requires a density drop between clusters [21], which a preliminary analysis of our dataset shows could not be guaranteed.

3.3.3 Profile Construction. The obtained behavioral patterns are used to build user profiles. Intuitively, a user profile represents the typical access behaviors of a user. To determine whether a behavioral pattern belongs to a user's profile, we consider all data accesses made by the user and check if any of them occur in the cluster representing the behavioral pattern. A user profile consists of all behavioral patterns exhibited by the user. Formally: Let $\mathcal{R}$ be the set of access events in the historical data and $B_1, \ldots, B_n \subset \mathcal{R}$ the behavioral pattern extracted from $\mathcal{R}$. Given a user u with historical data $\mathcal{R}_u \subset \mathcal{R}$, the profile of u, denoted as P_u, is defined as:

$$P_u = \{B_i \mid \exists r \in \mathcal{R}_u \cap B_i\}$$

The definition above assumes that a behavioral pattern is part of a user profile if the user made at least one access that matches the behavioral pattern. However, our approach can be generalized and allows one to tune the sensitivity of the anomaly detection system, e.g., by including behavioral patterns in a profile only if at least a certain percentage of the user's accesses are in the respective cluster.

It is worth noting that our approach builds profiles at the level of individual users instead of at the level of roles as done in prior work [26]. This way, the obtained user profiles can better characterize the differences in behaviors of users in the same job position. At the same time, our approach builds user profiles from behavioral patterns extracted from the access events of all users rather than only from the events of each user individually. As shown in previous studies, the latter can be problematic when the historical data available for some users is scarce [2], while our approach can construct a profile for a user even if she performed few data access in the past.

3.4 Anomaly Detection System

The anomaly detection system aims to determine whether an access request is *normal* or *anomalous* for the requester based on her user profile. To this end, the anomaly detection system should determine

[1] As an alternative to K-prototypes, we tested K-means using a one-hot encoding, i.e., each value of categorical features is encoded as a binary feature. This approach, however, is not only memory intensive but also requires adjusting the weighing of the introduced features. Indeed, splitting categorical features into multiple binary features would skew the results toward features with a higher number of values. For instance, in our dataset (Section 4), the feature *wall-outlet* has 2667 unique values. Thus, the clustering is extremely dependent on *wall-outlet*, disregarding nearly all other features.

to which behavioral pattern an incoming request belongs and verify whether that behavioral pattern belongs to the requester's profile. This can be seen as a classification problem. The choice of classification model to be used depends on the characteristics of the data, their dimensionality, and the algorithm employed for clustering. We selected the classification model for our prototype based on an experimental evaluation of widely used classification models (Section 4.2). We evaluated existing classification models in terms of both classification accuracy and computational performance. The first metric is used to assess their efficacy, whereas the latter determines the feasibility of the approach in a run-time setting.

It is worth noting that the contextual features used for the classification of requests should be the same as the ones used to construct the behavioral patterns (cf. Section 3.3.1). To this end, we assume that the Context Handler enriches the request with the selected features before passing it to the anomaly detection system (represented by the arrow labeled 'permit' to the Context Handler in Fig. 1).

4 PROTOTYPE IMPLEMENTATION

To evaluate our framework, we developed a prototype based on the real-life access log recorded by a hospital in the Netherlands [2].

4.1 Dataset

The dataset contains events recording the accesses to patient information made by the hospital staff over one month. Each event records either a "regular" access request (along with the corresponding decision) or an invocation of the BTG protocol. When a user requests access to patient data, the hospital IT system evaluates the request against the security policies in place. If the policies authorize the request, access is granted. Otherwise, the system prompts the user to invoke the BTG protocol, in which case the user has to indicate the reason for which she wants to access the data; if the user decides to use the BTG protocol, access to the data is granted to the user. For our evaluation, we considered the access requests and invocations of the BTG protocol made by users with the role "Behandelend Arts" (i.e., doctor), for a total of 447,877 events. Detailed statistics of the dataset used for the evaluation are reported in Table 1.

The dataset comprises 12 features, as shown in Fig. 2. Feature *log-time* records the time of the request; *userID* is the unique identifier of the requester, and *parientNr* is the unique identifier of the patient for which the data were requested; *protocol* indicates whether the access request is regular along with the access decision (permit or deny) or whether the BTG protocol was invoked, in which case *explanation* reports the reason for invoking the BTG protocol (e.g., New Patient, Treatment, Research); *explanation* is `NULL` for regular accesses. The log also contains information about the machine from which the access request was made, namely features *IP-address*, *inventory number* of the computer, and its *type* (desktop or laptop). Feature *last active* indicates the last time the user was active, which in most cases is very close to the time when the log was retrieved. Feature *wall-outlet* represents the identifier of the network socket (e.g., ID H2-259-A5 denotes the building (H), the floor number (2), the room number (259), and the room socket number (A5)), which uniquely identifies the location of the computer from which the request was made in the hospital, and *division* indicates from which division of the hospital the data were requested; finally, feature *role* denotes the job position of the user at the hospital (in our case "Behandelend Arts").

Table 1: Data used in the experiment

nr. features	12
total nr. requests	447877
nr. granted requests	425404
nr. BTG invokations	15572
nr. unique users	1522
nr. unique users who invoked the BTG protocol	525

4.2 Implementation Details

To build the anomaly detection system, we used the access requests that were permitted by the hospital IT system (hereafter referred to as *granted requests*). In particular, we used 95% of the granted requests made by users with the role "Behandelend Arts" to build the user profiles; the other 5% was used for validation (see Section 5.1).

Feature Extraction and Selection. We analyzed the features in our dataset (cf. Section 4.1) to identify relevant contextual features for the construction of user profiles. Features *userID*, *role*, and *protocol* were used to determine which requests should be considered for the construction of user profiles. In particular, *role* and *protocol* were employed to identify the requests used for the elicitation of the behavioral patterns (i.e., granted requests made by users with the role "Behandelend Arts"), whereas *userID* was used to associate the behavioral patterns belong to user profiles. We did not consider feature *explanation* as this feature is only meaningful for BTG invocations.

For features *division* and *log-time*, we leveraged the analysis of the access log presented in [2]. This analysis shows that many more data accesses were made during working days compared to non-working days (i.e., weekends and holidays). A similar pattern emerges by comparing accesses made during working and non-working hours. Based on these observations, we extracted two features from *log-time*: feature *shift*, which denotes whether the data were requested during the *day* shift (between 7:00 and 19:00) or the night shift (between 19:00 and 7:00), and work *schedule*, which indicates whether the requests was made during working days or non-working days. The analysis in [2] also reveals notable differences in access behaviors based on the *division* from which patient data were accessed. Therefore, we included this feature in our analysis.

We additionally considered the other features in the access log, which were not investigated in [2]. An analysis of *inventory*, *IP*, and *wall-outlet* shows that these features are strongly intercorrelated. In particular, their Cramér's V, which measures the association between two categorical variables, is nearly 1, indicating a strong association between these variables. Thus, we only used *wall-outlet* for clustering. We performed Principal Component Analysis (PCA) to evaluate the impact of *patientNr* and *computer type* on clustering and observed that their use negatively affects clustering, resulting in less well-marked behavioral patterns. In particular, *patientNr* is an identifier that carries no contextual information about the type of patient (e.g., the division in which she is treated or the disease she is affected) or the type of data accessed. Feature *last active* reports the last time the user was active within the system, which in our dataset is close to the date on which the dataset was retrieved from

Table 2: Cluster analysis

clusterID	**division**	**weekly**	**daily**	**hourly**	**wall-outlet**	**shift**	**schedule**	**nr. requests**	**nr. users**
1	DIVA	179	21	16	G3-155-A8	day	weekend	17974	455
2	DIVB	1928	285	103	A01-127-A3	day	working day	8666	2
3	DIVB	1918	287	74	A01-127-A6	night	working day	5875	2
4	DIVB	194	37	10	F3-251-A2	night	working day	23508	569
5	DIVA	290	68	32	G6-255-A5	day	working day	106717	262
6	DIVB	184	24	9	F3-251-A2	night	weekend	7232	277
7	DIVA	112	27	11	F3-251-A2	day	working day	234161	1504
total	DIVA	233	47	19	A01-127-A3	day	working day	404133	1522

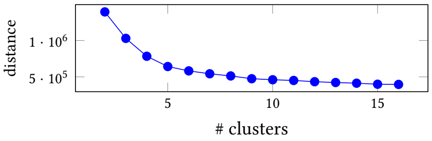


Figure 3: K-Prototypes clustering costs for different k

the IT system for all requests. Accordingly, it is not representative of user access behavior and, thus, discarded from the analysis.

To capture behaviors that can potentially signal the occurrence of privilege misuse, we also extracted contextual features representing the number of requests the requester made per hour (*hourly*), per day (*daily*), and per week (*weekly*). These features allow us, for instance, to flag cases where a user is attempting to exfiltrate abnormal amounts of patient data. To compute these features, we leveraged the historical access data. However, their computation varies for the requests used to elicit the behavioral patterns and for the requests used for validation. For the training data, we looked up in the historical data how many requests the user made at each hour (day and week, respectively) based on the request's *log-time* and computed the number of requests in the given period for all those requests. This allows us to build behavioral patterns accounting for the amount of data requested by users (measured in terms of the number of requests). The computation of the features for the requests in the validation dataset is similar, but the feature values are computed on a request basis. In particular, for each request, we compute the number of requests the user made in the previous hour, on that day, and in that week, respectively. This allows us to simulate the run-time functioning of the system.

Overall, we selected seven contextual features from the considered features: four are categorical (*shift*, *schedule*, *division*, *wall-outlet*), and the others are numerical (*hourly*, *daily*, and *weekly*).

Behavioral Pattern Elicitation. As discussed in Section 3.3.2, we employed K-prototypes to elicit behavioral patterns from the dataset of granted requests. Fig. 3 shows the application of the elbow method from 2 to 16 clusters to our dataset, concluding that the best number of clusters is 7. An overview of the seven clusters obtained using K-prototypes is presented in Table 2. The first column reports the clusterID, and the next seven columns report the most frequent feature value in the cluster for categorical features and the average feature value for numerical features. The last two columns show the number of requests and unique users in each cluster, respectively.

We can observe that some clusters stand out. Clusters 2 and 3 consist of requests made only by two users, which indicates that these two users exhibit distinct behavioral patterns. These users turned out to be two bots (registered within the system with the role "Behandelend Arts"), which make many requests around the same time every day as demonstrated by the extremely high count of *weekly* requests. The main difference between these two clusters is that one comprises requests made during the day shift and the other requests made during the night shift (we refer to Fig. 7 for an in-depth analysis of feature *shift*). On the other hand, cluster 7 comprises the access requests made by 1504 unique users, with only 18 users not having any requests in this cluster. Therefore, we can conclude that this cluster captures a general behavioral pattern common to almost all doctors in the hospital. This observation is also supported by the number of requests in the cluster, which is more than half of all requests in the dataset.

When looking at the clusters' features, we can observe that DIVA and DIVB are the most dominant divisions in all clusters. This can be explained by the fact that these two divisions are the most frequent in the dataset, with DIVA occurring in 23.4% of the access events in the dataset and DIVB in 16.4%. The distribution of divisions per cluster in Fig. 4 shows that, while these feature values are not discriminant for the obtained behavioral patterns, less frequent divisions are more common in certain clusters.

User Profiles Construction. We constructed the user profiles by considering, for each user, the granted requests that the user made and to which cluster(s) these requests belong (cf. Section 3.3.3). The profile of a user, thus, encompasses all behavioral patterns exhibited by the user. Fig. 5 shows the size of the obtained user profiles, and Fig. 6 shows how frequently clusters occur together. For instance, the first line of Fig. 6 shows that, among the 455 users exhibiting the first behavioral pattern, 319 users also exhibit the fourth behavioral pattern and 124 users the fifth behavioral pattern.

We can observe that the profile of all users comprises from 1 to 5 clusters of requests and that the behavioral patterns are distributed across the board, indicating that the extracted behavioral patterns can discriminate the behavior of different users. It is worth noting that the profile of 57 users consists of 5 clusters; these clusters represent all behavioral patterns exhibited by humans (recall that two clusters comprise the requests made by bots).

Request Classifier. To determine whether an access request is anomalous for a given user, the anomalous detection system should determine whether the request falls in any behavioral pattern associated with that user. As discussed in Section 3.4, we do this by employing a classifier that, given an access request, classifies it into one of the extracted behavioral patterns. We tested several classification models to determine which one is more suitable for our purpose. For our assessment, we used the portion of the dataset used to extract the behavioral patterns, where 85% of the data was used to train the classifier, and the remaining 15% was used for validation; the cluster

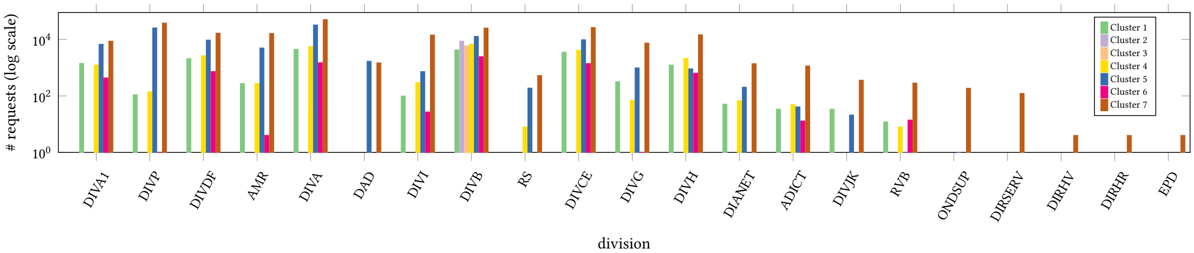


Figure 4: Division distribution per cluster

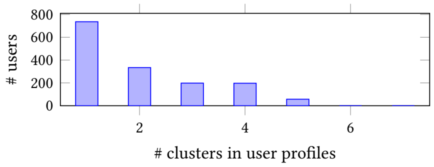


Figure 5: Size of the obtained user profiles

c	1	2	3	4	5	6	7
1	455	0	0	319	124	249	445
2	0	2	2	0	0	0	0
3	0	2	2	0	0	0	0
4	319	0	0	569	147	238	561
5	124	0	0	147	262	67	256
6	249	0	0	238	67	277	272
7	445	0	0	561	256	272	1,504

Figure 6: Cluster confusion matrix

Table 3: Classifier evaluation

Algorithm	Accuracy
Multi Layer Perceptron (MLP)	99.93%
K-nearest neighbor (KNN)	99.93%
Random Forest	99.92%
Decision Tree	99.83%
Support Vector Classifier (SVC)	99.74%
Naive Bayes	90.79%
AdaBoost Classifier	93.25%
Quadratic Discriminant Analysis	22.07%

IDs were used as classification labels. A summary of the tested classification models along with their accuracy is reported in Table 3.

We can observe that SVC, K-Nearest Neighbors, Decision Tree, Random Forester, and MLP all perform well in terms of classification accuracy. As their accuracy is nearly identical, we selected the classifier model based on their computational performance to ensure a fast evaluation process at run-time. Our investigation showed that SVC has a hefty evaluation time (0.67 ms per request on average) compared to the other algorithms (less than 0.1 ms per request); thus, we discarded this classification model. Among the remaining algorithms, we chose K-Nearest Neighbors because the cluster relations made by K-means are easier to capture by this algorithm.

The K-Nearest Neighbor algorithm takes k of nearest neighbors to the request to be classified and chooses the most dominant cluster among its nearest neighbors. We tested the algorithm for various values of k and observed that the best performances are obtained when k is equal to 3, as shown in Table 4.

5 EVALUATION

To assess the performance of our framework and its practical feasibility, we conducted two experiments. In the first experiment, we generated synthetic anomalous behaviors based on predefined attack scenarios and evaluated the ability of the framework to detect such behaviors. In the second experiment, we used the prototype to evaluate the invocations of the Break-The-Glass (BTG) protocol at the hospital to assess our framework in a real setting.

Table 4: Number of neighbors (k) for K-Nearest Neighbors

nr neighbors	Accuracy
1	99.89%
2	99.91%
3	99.93%
4	99.91%
5	99.87%
6	99.85%
8	99.83%
10	99.78%
15	99.61%

5.1 Experiment 1: Attack Scenarios

5.1.1 Settings. This experiment aims to evaluate the effectiveness of our framework in discriminating anomalous behaviors from legitimate behaviors. To this end, we used the 5% of the granted requests that were not used to build user profiles to evaluate its ability in recognizing legitimate behaviors. For our evaluation, we assume these requests to be legitimate. To evaluate the ability of the approach to detect anomalous behaviors, we artificially generated anomalous access events based on predefined attack scenarios.

In total, we defined five attack scenarios (Table 5), each representing how a user of the system (or an external attacker impersonating a legitimate user of the system) can deviate from her normal behavior. Based on these attack scenarios, we generated anomalous access events semi-randomly. The users of these requests were chosen manually to ensure they do not exhibit behaviors in line with the attack scenario. The feature parametrization of the generated anomalous access events was derived from the analysis of behavioral patterns and user profiles presented in Section 4.2; request features that are not relevant to the attack scenario were generated randomly.

We assessed the results in terms of *true positives* (TP), *false positives* (FP), *true negatives* (TN), and *false negatives* (FN), where the positives indicate the number of requests flagged by the system as anomalous, and the negatives indicate the number of requests flagged by the system as legitimate. To better assess how well the system performs, we also compute the *accuracy* (i.e., the proportion of correct predictions – both true positives and true negatives – among the total number of cases examined), *precision* (i.e., number of true positives divided by all positives), *recall* (i.e., number of

Table 5: Attack scenarios

1	**Unusual number of requests**: The number of requests made by the user exceeds his average number of weekly and daily requests.
2	**Unusual working shift**: A user requests access to patient data during a night shift when he does not normally do it.
3	**Unusual working schedule**: A user requests access during the weekend when he normally does it during non-working days.
4	**Unusual working days**: A user requests patient data during working days in which he has never requested before.
5	**Unusual working locations**: A user requests patient data from locations from where he has never requested before.

Table 6: Attack scenario evaluation

	TP	FP	TN	FN	Accuracy	Precision	Recall	F1-score
no scenario	–	6	21260	–	0.999	–	–	–
scenario 1	4486	6	21260	1505	0.945	0.999	0.749	0.856
scenario 2	317	6	21260	0	0.999	0.981	1.000	0.990
scenario 3	241	6	21260	0	0.999	0.976	1.000	0.904
scenario 4	0	6	21260	78	0.998	0.000	0.000	0.000
scenario 5	80	6	21260	385	0.982	0.930	0.183	0.306
total	5124	6	21260	1968	0.930	0.999	0.723	0.839

true positives divided by the true positives and false negatives) and *f1-score* (i.e., the harmonic mean of the precision and recall).

5.1.2 Results. Table 6 reports the results of the first experiment, where **no scenario** refers to the case where only legitimate requests are considered and **scenario X** to the cases where the attack instances of the corresponding scenario are considered along with the legitimate requests; in **total**, the results for legitimate requests are counted only once. We can observe that the system has an overall accuracy of 93% and a recall of 72.3%. In particular, for legitimate requests (no scenario in Table 6), it has a very low false positive rate, with only 6 requests out of the 21266 legitimate requests flagged as anomalous (corresponding to an accuracy of 99.9%). On the other hand, the system was able to detect 72.3% of the simulated attacks. However, we note a large variation per attack scenario. Next, we discuss the results for each attack scenario individually.

Attack Scenario 1: Unusual number of requests. As shown in Table 6, the system was able to detect 74.9% of the attack instances (recall) for the first attack scenario, while a quarter of the instances were labeled as normal. This result is expected as an alert is raised only when the cumulative number of weekly and daily requests exceeds what is typically requested by the user. Recall that, in our experiment, we simulated the actual functioning of the system and computed the cumulative number of requests on a request basis. Accordingly, the first few requests of the user were in line with their behavioral patterns and, thus, considered legitimated by the anomaly detection system. For example, consider a doctor who usually accesses the data of 50 patients per day; if one day he suddenly accesses the data of hundreds of patients, the first 50 requests would be recognized as legitimate by the anomaly detection system and, only when the amount of requested data requested exceeds normal usage, the anomaly detection system raises an alert. The results show that the detection system can detect this attack when discrepancies with users' behavioral patterns emerge.

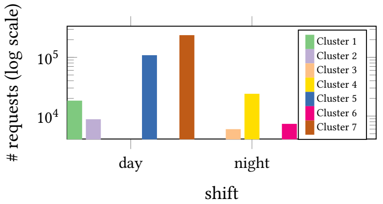


Figure 7: Night and day shift distribution per cluster

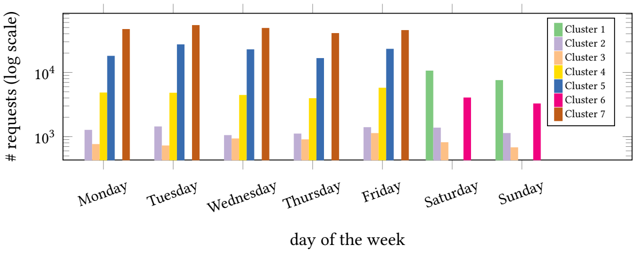


Figure 8: Schedule distribution per cluster

Attack Scenario 2: Unusual working shift. This attack scenario covers situations in which users who usually do not request patient data during the night shifts suddenly do. For example, an employee working in the administration requests access to patient data during the night when he has previously requested data only during the day. The results for the second attack scenario show a recall of 100%, indicating that all cases are flagged as anomalous by the system. This highlights that the shift is a strong indicator for discriminating user behaviors, as also illustrated by the shift distribution per cluster in Fig. 7, which shows that all access events in each cluster occurred either during the day shift or during the night shift.

Attack Scenario 3: Unusual working schedule. This attack scenario represents situations in which users who usually do not access patient data during non-working days suddenly request patient data during those days. For example, a doctor requests access to patient data during the weekend, when he usually places access requests only during working days. The results show that the system can detect all attack instances of this type (100% recall). Again, this is due to the ability of the clustering algorithm to distinguish between requests made on working days and requests made during non-working days, as shown in Fig. 8. From the figure, we can observe that only the behavioral patterns exhibited by the bots (clusters 2 and 3) comprise requests made during both working and non-working days; all other clusters consist of requests made either during working days or during the weekend.

Attack Scenario 4: Unusual working day. This attack scenario represents situations where an attacker requests access to data on different working days from when they usually do. For example, a doctor requests access to patient data on Friday when he typically does only on Tuesday and Thursday. The results show that the system is not able to detect these cases (0% recall). This result is not surprising when looking at the day distribution of requests per the cluster in Fig. 8. We can observe that all clusters comprising requests made during working days include requests made during every working day. It is worth noting that we used a Boolean feature (working days vs. non-working days) to capture day-related information in the construction of behavioral patterns (cf. Section 4.2). Therefore, we argue that the detection system was not trained to distinguish requests made during different working days. We did additional experiments in which we clustered access events on individual weekdays. However, we obtained virtually identical behavioral patterns, with no clear distinction between requests over different working days.

Attack Scenario 5: Unusual Working Location. This attack scenario covers situations in which, for example, a doctor places requests to access patient data from a division different from the one from which he typically works. The results in Table 6 show that the system can only partially detect requests made from unusual locations. In particular, the system did not flag as anomalous any request made from a location from which the user never requested before and only detected some attack instances when the user has requested patient data from different locations. These results are partially due to the low discriminant power of the features representing location information (*wall-outlet* and *division*). An inspection of the distribution of *division* (cf. Fig. 4) and *wall-outlet* (not reported here for the lack of space) per cluster shows that the values used to generate the attack instances were also present in some behavioral patterns associated with the user. Overall, a realistic location switch was difficult to simulate due to the lack of context information when generating the attack instances for this scenario.

5.2 Experiment 2: BTG Requests

5.2.1 Settings. The second experiment aims to evaluate the practical applicability of our framework in a real-life setting. In particular, we used the prototype implementation trained using the granted requests, as described in Section 4, to evaluate the 15572 invocations of the BGT protocol at the hospital (cf. Table 1). This allows us to assess the framework in the context of actual data accesses performed by the hospital staff and, thus, to gain insights into its practicality in real-life situations. As the legitimacy of the BTG invocations is unknown, we only provide a qualitative evaluation. In particular, we assess how frequently an anomalous request is detected and, thus, how often the system should take action to mitigate the potential risks of misuse of the BTG protocol.

5.2.2 Results. Our analysis shows that, out of the 15572 BTG invocations in our dataset, 15533 were classified as normal and 23 as anomalous; the remaining 16 BTG invocations are from 8 users who do not have a user profile (i.e., they never made regular requests and only accessed patient data using the BTG protocol within the data collection period). As our system returns very few false alerts (cf. Section 5.1.2), these requests are likely to represent anomalous behaviors within the hospital. An inspection of these access events shows that 19 of the 23 flagged BTG invocations were made at night, either to add a new patient or to consult on existing patients. On the other hand, for the four requests made during the day, 'research' was provided as an explanation for using the BTG protocol.

The 23 anomalous BTG invocations were made by 14 users, out of the 525 users who invoked the BTG protocol (cf. Table 1); their distribution per user is reported in Fig. 9. We can observe that the number of anomalous requests per user ranges from one request (nine users) to four requests (one user). On the other hand, none of the users who do not have a user profile did more than three BTG invocations. All their requests were made during the day and have as an explanation either 'consulting with a colleague' or 'adding a new patient', with only two cases where the explanation was 'research'.

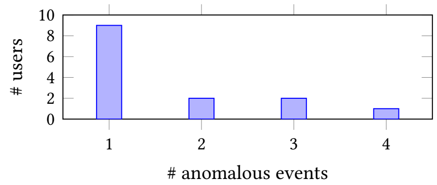


Figure 9: Anomalous BTG invocation per user

6 DISCUSSION

In this section, we elaborate upon some key observations obtained from our experiments and discuss the limitations of our approach.

Adversarial Attacks. A key aspect of our system is constructing user profiles representing patterns of typical access behavior. Since this process relies on clustering, it may be subject to attacks that could lead to altering the clustering output. For example, user profiling could be subject to *poisoning* and *obfuscation* attacks [5]. In a poisoning attack, an adversary tries to poison the dataset used for clustering by injecting carefully crafted samples to compromise the whole clustering process. In our system, an attacker can inject a large number of access requests that lead to generating clusters that no longer discriminate users' behavior. On the other hand, in obfuscation attacks, the adversary's goal is to hide a given set of initial attack samples within some existing clusters, possibly without altering the clustering results for the other samples. An example of this attack is an insider threat that generates access requests to be included in one of the clusters representing normal user behavior. The effectiveness of such attacks is determined mainly by the information available to the adversary about the attacked system. The attacker should have full knowledge of the attacked system, meaning he knows the behavior of legitimate users, the clustering algorithm, and the features used for clustering. This might be a realistic assumption for an insider threat who can gain full knowledge of the system by colluding with other system users, e.g., a system administrator. On the other hand, an external attacker can be considered a weaker adversary because it might be more difficult for such an attacker to acquire knowledge about the behavior of the users of the system as well as about the technical details of the anomaly detection system.

Sensitivity of the anomalous detection system. Our experiments show that our framework returns very few false positives (cf. **no scenario** in Section 5.1). Although the discriminant power of some features in our dataset (e.g., *shift* and *schedule*) and the clustering algorithm could have positively contributed to these results, the low false positive rate is also due to our approach for building user profiles. We employed a lenient approach where a behavioral pattern is added to the profile of a user if the user made at least a similar request in the past (cf. Section 3.3.3). Nonetheless, our approach is flexible and can be easily customized to restrict the behavioral patterns forming user profiles, for example, by requiring a minimum percentage of requests in a given cluster or by employing outlier detection approaches to remove unusual access events from the data used to build user profiles. Further research is needed to investigate how to tune the sensitivity of the anomaly detection system.

Handling new users. Our approach requires that a profile is associated with each user to determine whether their requests are anomalous. Nonetheless, the evaluation of BTG invocations at the hospital in Section 5.2 shows this might not be the case in real-life settings, where we encountered eight users for which it was impossible to define a user profile due to the lack of data. While, in our experiments, this is mainly due to the restricted data collection period (one month), this issue can generally be encountered with new users. In these cases, a 'default' profile should be assigned to those users. However, the definition of such a profile can introduce risks because it would associate behaviors not typically exhibited by the user. An analysis of the elicited behavioral patterns can assist in the definition of a 'default' profile. For instance, the analysis in Section 4.2 shows that the behavioral patterns corresponding to cluster 7 occur in the profile of 1504 out of the 1522 users with the role "Behandelend Arts". Therefore, one could safely assume that this behavioral pattern represents a typical behavior of users with that role and can be leveraged to define a default profile for new doctors.

No one-size-fits-all mitigation strategy. A common issue of all anomaly detection systems is false positives [15]. Therefore, anomalous access requests cannot be merely denied, especially in mission-critical systems such as hospitals. Mitigation strategies can be employed to mitigate the risks of privilege misuse while guaranteeing the correct functioning of the system. The choice of the strategy to be used might depend on several factors, e.g., the sensitivity of the requested information and its context of usage. For instance, anomalous requests for patient data for research purposes (cf. Section 5.2) might undergo the approval of the requester's supervisor, before they can be granted. An investigation of mitigation strategies and the definition of automated methods for their selection based on contextual information are interesting directions for future work.

7 CONCLUSION

In this work, we presented a framework that complements a traditional access control system with anomaly detection to identify privilege misuses. To this end, we proposed a novel approach to building user profiles by eliciting patterns of typical access behavior from historical data. Our framework relies on the constructed user profiles for run-time monitoring of access requests and raises an alert when a user's request diverges from her profile. We evaluated our framework using a dataset recording access to patient data in a hospital. The results show that it raises a low number of false positives (99.9% accuracy) and can detect several attack scenarios. We also applied our framework to evaluate BTG invocations at the hospital, thus demonstrating its practical applicability in real-life settings.

Our analysis only focused on requests of users with the role "Behandelend Arts". We plan to extend our prototype to consider other roles within the hospital. We also plan to investigate mitigation strategies to reduce the risk of privilege misuse and automated methods for their selection based on contextual information.

ACKNOWLEDGMENTS

This work is funded by the EDA project WINLAS.

REFERENCES

[1] A. Abu Jabal, E. Bertino, J. Lobo, M. Law, A. Russo, S. Calo, and D. Verma. 2020. Polisma - A Framework for Learning Attribute-Based Access Control Policies. In *ESORICS*. Springer, 523–544.

[2] M. Alizadeh, S. Peters, S. Etalle, and N. Zannone. 2018. Behavior analysis in the medical sector: theory and practice. In *SAC*. ACM, 1637–1646.

[3] L. Argento, A. Margheri, F. Paci, V. Sassone, and N. Zannone. 2018. Towards Adaptive Access Control. In *DBSec*. Springer, 99–109.

[4] N. Baracaldo and J. Joshi. 2013. An adaptive risk management and access control framework to mitigate insider threats. *Computers & Security* 39 (2013), 237–254.

[5] B. Biggio, I. Pillai, S. Rota Bulò, D. Ariu, M. Pelillo, and F. Roli. 2013. Is Data Clustering in Adversarial Settings Secure?. In *AISec*. ACM, 87–98.

[6] T. Bui and S. Stoller. 2020. A Decision Tree Learning Approach for Mining Relationship-Based Access Control Policies. In *SACMAT*. ACM, 167–178.

[7] L. Cappelletti, S. Valtolina, G. Valentini, M. Mesiti, and E. Bertino. 2019. On the Quality of Classification Models for Inferring ABAC Policies from Access Logs. In *Big Data*. IEEE, 4000–4007.

[8] S. Chakraborty, D. Paul, and S. Das. 2021. Automated Clustering of High-dimensional Data with a Feature Weighted Mean Shift Algorithm. In *Conference on Artificial Intelligence*. AAAI Press, 6930–6938.

[9] G. Chandrashekar and F. Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering* 40, 1 (2014), 16–28.

[10] C. Chin-Chen, L. Iuon-Chang, and L. Chia-Te. 2006. An Access Control System with Time-constraint Using Support Vector Machines. *Int. J. Secur. Netw.* 2 (2006).

[11] D Comaniciu and P Meer. 2002. Mean shift: a robust approach toward feature space analysis. *TPAMI* 24, 5 (2002), 603–619.

[12] S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. 2003. Access control: principles and solutions. *Software: Practice and Experience* 33, 5 (2003), 397–421.

[13] NJ de Vos. 2015–2021. kmodes categorical clustering library. https://github.com/nicodv/kmodes.

[14] P Duessel, S Luo, U Flegel, S Dietrich, and M Meier. 2020. Tracing Privilege Misuse Through Behavioral Anomaly Detection in Geometric Spaces. In *SADFE*.

[15] G. Fernandes, J. Rodrigues, L. Carvalho, J. Al-Muhtadi, and M. Proença. 2019. A comprehensive survey on network anomaly detection. *Telecommun. Syst.* 70, 3 (2019), 447–489.

[16] John C Gower. 1971. A general coefficient of similarity and some of its properties. *Biometrics* 27, 4 (1971), 857–871.

[17] V. Gumma, B. Mitra, S. Dey, P. Patel, S. Suman, and S. Das. 2021. PAMMELA: Policy Administration Methodology using Machine Learning. *CoRR* (2021).

[18] J Ji, T Bai, C Zhou, C Ma, and Z Wang. 2013. An improved k-prototypes clustering algorithm for mixed numeric and categorical data. *Neurocomputing* 120 (2013).

[19] L. Karimi, M. Aldairi, J. Joshi, and M. Abdelhakim. 2022. An Automatic Attribute-Based Access Control Policy Extraction From Access Logs. *TDSC* 19, 4 (2022).

[20] L Karimi and J Joshi. 2018. An Unsupervised Learning Based Approach for Mining Attribute Based Access Control Policies. In *Big Data*. IEEE, 1427–1436.

[21] K Khan, SU Rehman, K Aziz, S Fong, and S Sarasvady. 2014. DBSCAN: Past, present and future. In *ICADIWT*. IEEE, 232–238.

[22] P. Legg, O. Buckley, M. Goldsmith, and S. Creese. 2015. Caught in the act of an insider attack: Detection and assessment of insider threat. In *HST*. IEEE, 1–6.

[23] A. Likas, N. Vlassis, and J. Verbeek. 2003. The global k-means clustering algorithm. *Pattern Recognition* 36, 2 (2003), 451–461. Biometrics.

[24] P Nicopolitidis, A Liu, X Du, and N Wang. 2021. Efficient Access Control Permission Decision Engine Based on Machine Learning. *Secur Comm Netw* (2021).

[25] M Nobi, R Krishnan, Y Huang, M Shakarami, and R Sandhu. 2022. Toward Deep Learning Based Access Control. In *CODASPY*. ACM, 143—-154.

[26] J Park and J Giordano. 2006. Role-based profile analysis for scalable and accurate insider-anomaly detection. In *IPCCC*. IEEE, 463–470.

[27] T Rashid, I Agrafiotis, and J Nurse. 2016. A New Take on Detecting Insider Threats: Exploring the Use of Hidden Markov Models. In *MIST*. 47–56.

[28] K Srivastava and N Shekokar. 2020. Machine Learning Based Risk-Adaptive Access Control System to Identify Genuineness of the Requester. In *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough*. Springer.

[29] MA Syakur, BK Khotimah, EMS Rochman, and BD Satoto. 2018. Integration k-means clustering method and elbow method for identification of the best customer profile cluster. In *ICVEE*, Vol. 336. IOP Publishing.

[30] Q Tasali, N Gyawali, and E Vasserman. 2020. Time Series Anomaly Detection in Medical Break-the-Glass. In *HotSoS*. ACM.

[31] T M Thang and J Kim. 2011. The Anomaly Detection by Using DBSCAN Clustering with Multiple Parameters. In *ICISA*. IEEE, 1–5.

[32] Fatih Turkmen, Jerry den Hartog, Silvio Ranise, and Nicola Zannone. 2017. Formal analysis of XACML policies using SMT. *Computers & Security* 66 (2017), 185–203.

[33] S Vavilis, A Ionut Egner, M Petkovic, and N Zannone. 2016. Role Mining with Missing Values. In *ARES*. IEEE, 167–176.

[34] Verizon. 2021. Data Breach Investigations Report.

[35] C. Xiang, Y. Wu, B. Shen, M. Shen, H. Huang, T. Xu, Y. Zhou, X. Moore, C.and Jin, and T. Sheng. 2019. Towards Continuous Access Control Validation and Forensics. In *CCS*. ACM, 113–129.