UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

# Profiling Users by Access Behaviour Using Data Available to a Security Operations Center

**J.J. Sonneveld**

**M.Sc. Thesis**
**January 2023**

**Supervisors:**
dr. R. Holz
dr. ir. A. Continella
ir. J.S. Spenkelink (Northwave)

Design and Analysis of Communication Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

# Abstract

Businesses face constant threats regarding cyber security, from both in- and outside their organisation. A Security Operations Center (SOC) monitors a company's digital infrastructure to protect against these threats by detecting suspicious events and taking mitigating action. Adversaries commonly need to access resources illegitimately to achieve action upon their objectives, and do so via existing user accounts. We develop a method to identify suspicious access behaviour using non-intrusive data available to a SOC. For every user, a feature vector is constructed describing their access behaviour. This vector contains statistics over a predefined period of time on *what resources* a user accessed at *what time* and in *what manner*. We survey different Machine Learning profiling possibilities and select the general-purpose K-means clustering to group users with similar behavioural characteristics.

We apply this method to the Insider Threat logs from the Carnegie Mellon University Software Engineering Institute, a synthetic dataset which is a benchmark for insider threat detection research. Through access behaviour profiling, we were able to identify all users with the ITAdmin role. The consistency of clusters for benign users over time was compared and quantified through the Adjusted Rand metric. Comparing consecutive months, this resulted in an average consistency score of 0.87. We were able to detect 80% of insider threats with the ITAdmin role by monitoring for changes in their cluster over time.

Applying the same methodology to access data of real-world organisations allows for significantly less consistent clustering, with average consecutive month consistency scores of 0.41. We observe a slight similarity between our clusters and the groupings within the organisation's Active Directory. The clusters also show slight similarity with the groups inferred by Microsoft User Entity Behaviour and Analytics via Machine Learning. We show how the granularity of the most relevant features of the synthetic dataset differs from that of real-world data, which is suspected to cause this decrease in clustering consistency.

To be able to detect suspicious changes in user access behaviour a consistent profiling method is required. Our privacy-preserving, general-purpose profiling methodology enabled consistent results on the synthetic dataset. However, we have strong indication that in a production environment this data alone is not able to produce results usable for detecting suspicious access behaviour.

# Contents

Chapter 1

# Introduction

Cyber threats pose a significant danger to organisations, and using account privileges to access digital resources illegitimately is a part of almost every successful cyber attack. This research creates a method to profile users based on their access behaviour using the data from a Security Operations Center (SOC). Having accurate user access profiles allows for detecting suspicious deviations in this behaviour, which can be used to alert a SOC analyst to take a closer look at an account's activity and determine whether the accesses could be malicious.

One of the good practices regarding cyber security is the least privilege principle. If a user does not need to access a particular resource for their regular responsibilities, their account should not have access to this resource. This is an example of *access control*, which takes effort to keep track of and maintain over time. For example, if someone leaves a the company, their accesses should be revoked. In reality, this is often not done correctly [1], causing accounts to have more privileges than required. This becomes a problem when an adversary manages to gain control of this account, known as account compromise, or when a user turns against their employer, called an insider threat. If a domain admin account gets compromised with all the access rights they possess, it will pose a security risk even when adhering to the least privilege principle. Therefore it is crucial to not only survey the access control policies but also consider the actual accesses an account performs to detect suspicious activity, which is where a SOC comes into play.

The goal of a SOC is to detect suspicious events within an organisation's digital infrastructure. Multiple data sources can be used for detection, such as sensors within the network to capture traffic, logs of servers and logs of end-point devices. A Security Information and Event Management system (SIEM) is used to combine and correlate this data. Using this SIEM, a SOC looks for suspicious events such as an account being granted admin privileges or a user visiting a possible phishing website. Not every detection implies a security risk. Actions labelled as suspicious can have been performed legitimately, or the detection rule can be faulty. Because every detection causes manual labour for an analyst to verify the detection's legitimacy, it is a SOC's goal to finetune its detection rules. The detections should trigger on as many security risks as possible, this is called a True Positive. The other possibilities include events without actual risk, called False Positives, from which we want as few as possible. Additionally, the rate of actual security threats going undetected, called False Negatives, should naturally be as close to zero as possible.

The digital infrastructure of an organisation usually contains 'crown jewels', which are their most valuable assets. These can be on-premise servers such as file servers, Domain Controllers (DCs) and mail exchange servers. DCs host a domain's Active Directory (AD), which imposes user and device policies. A compromise of the DC can lead to malicious policies being executed on all of an organisation's domain-joined devices, such as employee laptops. A file server usually contains much of an organisation's confidential data, which should only be accessed by those with a legitimate reason. Because we use access as an indicator for potentially malicious behaviour, these resource types are our main points of focus. The input data in this research is limited to access events within the Windows Event log to remain as little invasive on user's behavioural data as possible, whilst being able to make inferences on suspicious accounts. When behavioural monitoring raises an alert for the SOC, the alert needs to clarify *why* a certain user account was marked as suspicious to assist the analyst in their further investigation. This reasoning is also a requirement from ethical perspective, as it is impermissible to have an opaque algorithm labelling users for suspicion. This process needs to be as transparent and explainable as possible.

Detection rules within a SIEM describe when to raise an alert, based on log entries. An alert is a suspicious activity that a SOC analyst looks into to determine its legitimacy and takes action upon if it is suspected to be malicious. Frequently, such rules are static in determining suspicious events. However, this poses a problem when creating a profile based on behaviour, as this is a dynamic characteristic. Therefore, this research aims to develop a novel methodology using existing methods to create user behavioural profiles from access logs and be able to detect suspicious deviations from this profile over time, by using non-invasive data available to a SOC.

This research is conducted at the Northwave[1] SOC, which has a number of customers to which they provide Detection and Response services. Our novel profiling methodology constructed through literature research is first tested on a synthetic dataset. The dataset, called 'Insider Threat v4.2' [2], contains computer logs on benign and malicious users and was created by the Computer Emergency Response Team (CERT) division of Carnegie Mellon University. We use this dataset because

[1]https://northwave-security.com/

related literature often benchmarks their anomaly detection algorithms on it to quantify their performance. It also features data similar to that of available to a SOC. We use the access logs, a subset of this dataset, to validate our algorithm. If our clusters stay consistent for benign users, users switching clusters could be seen as deviating behaviour, which is potentially malicious. Consequently, we apply our method to the data of real-world organisations to measure how it performs in practice. Finally, we compare our groupings to the groups within the Active Directory, and those inferred by Microsoft's UEBA (User Entity Behaviour and Analytics). We measure the similarity between our clusters and the groups of those external sources.

The main question this research aims to answer is:

- **Is it possible to profile users consistently using explainable clustering and non-invasive data sources available to a Security Operations Center?**

To answer this question and guide this research, we need to answer the following sub-questions:

1. **Which non-invasive features readily available to a SOC are relevant in describing user access behaviour?**
   Using both related literature and threat intelligence, we infer relevant features to keep track of regarding access behaviour.

2. **Which explainable machine learning method can be used to profile users?**
   Of the available Machine Learning methods, we choose one which fits our purpose as explainable, simple and genericly applicable approach.

3. **How does our profiling methodology perform within the CERT v4.2 synthetic dataset on consistency and extracting user groups?**
   The CERT Insider Threat dataset contains data on every user's role within the artificial company and spans a large period. Therefore we can compare our clusters to these role groups and measure how stable the clusters are for benign users over time.

4. **How does this profiling methodology translate to real-world data in terms of consistency and extracted user groups?**
   We profile users by applying our methodology to login data available to a SOC and measure how stable the clustering is over time. We compare the extracted user groups to the Active Directory and to the UEBA-infered groups.

The structure of this report is as follows: The background in Chapter 2 describes technical and theoretical information required to understand the setup and machine learning used in this research. Chapter 3 surveys the literature for similar research and how user access behaviour can be adequately profiled. In Chapter 4 we describe the methodology on how our profiling is performed for both the synthetic CERT Insider Threat dataset and real-world data, and how we analyse its output. Chapter 5 contains the results of applying the algorithm to both datasets and its further analysis. In Chapter 6, we discuss the implications and shortcomings of our research and how this research could be extended as future work. Finally, in Chapter 7, we draw conclusions for every research question.

Chapter 2

# Background

This research is performed inside a Security Operations Center (SOC). First, we describe the data available to a SOC which can be used to reach our objective. Second, the Security Information and Event Management (SIEM) processes and possibilities are explained. Finally, we highlight the Data Science fundamentals required to understand the methodology of this research.

## 2.1 SOC Access Data

A SOC has several methods of monitoring digital infrastructure. By inserting sensors into the network, traffic can be captured to detect which hosts interact with each other and possibly inspect the payload. Server logs are very useful, as these are central parts of the infrastructure with which many users directly or indirectly interact. Organisations generally use a domain to manage their devices. A domain is a group of computers within a network administered by a central server, the Domain Controller (DC). Accounts within the domain can log onto every device they are authorised to, using their domain account and password. Domain accounts authenticate against the DC to log onto such devices and resources. There are multiple types of authentication, which are some of the events the SOC can monitor to determine a user's access behaviour. In addition to servers and networks, endpoint devices such as employee laptops or phones can be monitored, as these types of devices can provide access to company resources.

Each of the above-mentioned devices generates logs. To correlate these different data sources and analyse them, logs are sent to a Security Information and Event Management (SIEM) system. The SIEM solution used for the real-world scenario part of this research is Sentinel [1], a cloud service by Microsoft. Onboarded machines

[1]Microsoft Sentinel: `https://azure.microsoft.com/en-us/services/microsoft-sentinel/`

send a subset of their Windows Events logs [2] [3] to this SIEM for analysis.

To access a resource, a user must first authenticate to prove they are authorised. In the Windows domain this is commonly performed via the default protocol Kerberos [3], which uses tickets distributed by the DC, or the fall-back NTLM [4], a challenge-response mechanism. When a user accesses a resource after valid authentication, Windows Event 4624 is logged: *An account was successfully logged on*. This event contains, amongst others, information on the time, user, target host, authentication type, connection method and source IP. If a user logs off, this generates events 4634 or 4647, which include similar information.

## 2.2 Microsoft Sentinel

The real-world data part of this research is performed using the Microsoft Sentinel SIEM [5]. Sentinel is a cloud solution that, among others, can be used to ingest logs from endpoint devices. The data is saved in tables and queried using the Kusto Query Language (KQL). The tables used for this research include the `SecurityEvent` table, which contains the ingested Windows Events. Azure Active Directory (AAD) is an identity and access management tool organisations can use to sign into apps using Single Sign-On (SSO). Suppose the on-premise AD is synchronised with the Azure Active Directory. In that case, it is possible to correlate the on-premise accounts with cloud accounts and perform further analyses using User and Entity Behaviour Analytics (UEBA) [6]. This produces two additional tables: `IdentityInfo` and `UserPeerAnalytics`. IdentityInfo displays a combination of on-premise and cloud properties of an account. UserPeerAnalytics provides a list of up to 20 ranked peers for any AAD account, generated using Microsoft's machine-learning algorithm applied to all of its connected data sources. Since the peer analytics table is based on cloud accounts and our analysis is performed by monitoring on-premise accesses, the accounts need to be correlated using the IdentityInfo table as shown in Figure 2.1.

[2]Windows Event list: `https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/`

[3]Windows Events sent to a SIEM: `https://docs.microsoft.com/en-us/azure/sentinel/windows-security-event-id-reference`

[4]NTLM: `https://docs.microsoft.com/en-gb/windows/win32/secauthn/microsoft-ntlm`

[5]Microsoft Sentinel: `https://learn.microsoft.com/en-us/azure/sentinel/overview`

[6]Sentinel UEBA: `https://docs.microsoft.com/en-us/azure/sentinel/ueba-reference`

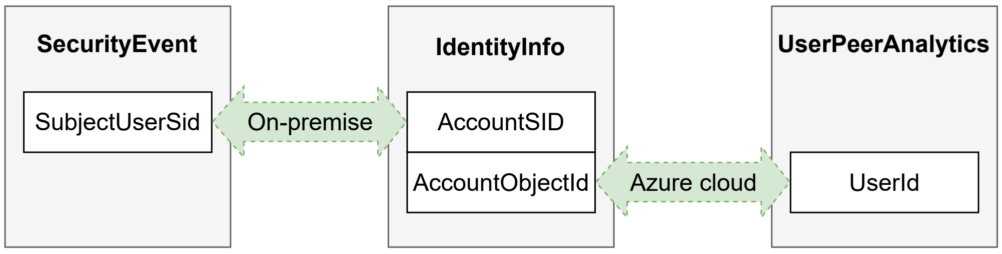


**Figure 2.1:** The fields within the SecurityEvent, IdentityInfo and UserPeerAnalytics Sentinel tables used to correlate on-premise to AAD accounts.

## 2.3 Data Science

We elaborate on several data science concepts used in our research. The way we apply these methods are further discussed in the methodology section.

### K-means clustering

This section describes the clustering approach, a choice which is further motivated in Section 3.5. There are different possibilities for clustering points in a multi-dimensional area. In our case of user clustering, every point represents a user and a feature vector is a list of numerical properties describing the user's behaviour. Every feature within a user's feature vector is translated to a dimension within the multidimensional area. K-Means aims to partition objects in this multi-dimensional space into *n* clusters. This is achieved by minimising the 'Within-Cluster Sum of Squares' and maximising the 'Between-Cluster Sum of Squares'.The naive version of the algorithm uses the following steps, which are further illustrated by Figure 2.2:

- **Step 1:** *Initialisation* select *n* random points in multi-dimensional space to be the initial 'centroid'
- **Step 2:** *Initialisation* Associate each point to its closest 'centroid'
- **Step 3:** Compute the gravitational centre, the centroid, for each label
- **Step 4:** Associate each point with their closest centroid
- **Step 5:** Repeat from step 3 until convergence, or halt if the centroids do not move anymore.

The parameter provided to the K-Means clustering algorithm is the number of clusters, which influences every of the clustering steps. An extra centroid means that the results of steps 3 and 4 of the algorithm will be different. Additionally, there is a factor of chance in the algorithm. Picking the initial centroid points is performed at random, so selecting different points could cause the centroids to travel in a different direction. Hence it is essential to reproducibility to seed the randomness so that, given the same data, the algorithm will result in the same clusters when applied

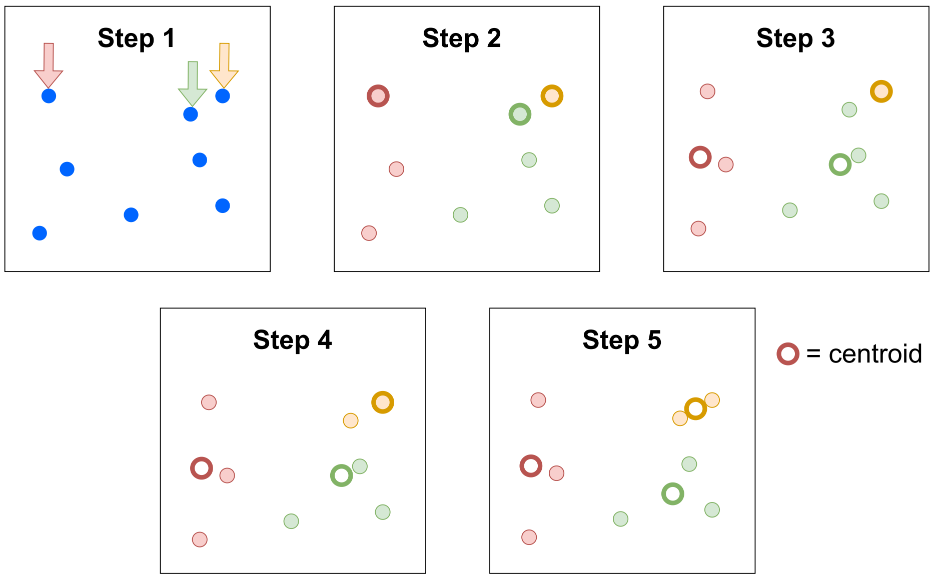


**Figure 2.2:** A step-by-step simulation of the K-means algorithm for K=3.

multiple times.

## (Adjusted) Rand index

A metric is required to measure how similar the clusters from different periods in time are. For this, we compare how users are divided amongst the clusters for different iterations of the algorithm. This is not as trivial as counting the number of users classified into different clusters, as the number of users designated to a cluster can change. For this, we apply the adjusted Rand index[7], a function measuring the similarity of two separate cluster divisions. This metric was initially developed by Rand [4], but because of the underlying formula, it often returns a value close to 1 when comparing different sets of labels. The Rand index method was later optimised by Hubert et al. [5] to compensate for this factor of chance. We therefore apply the adjusted Rand score in this research.

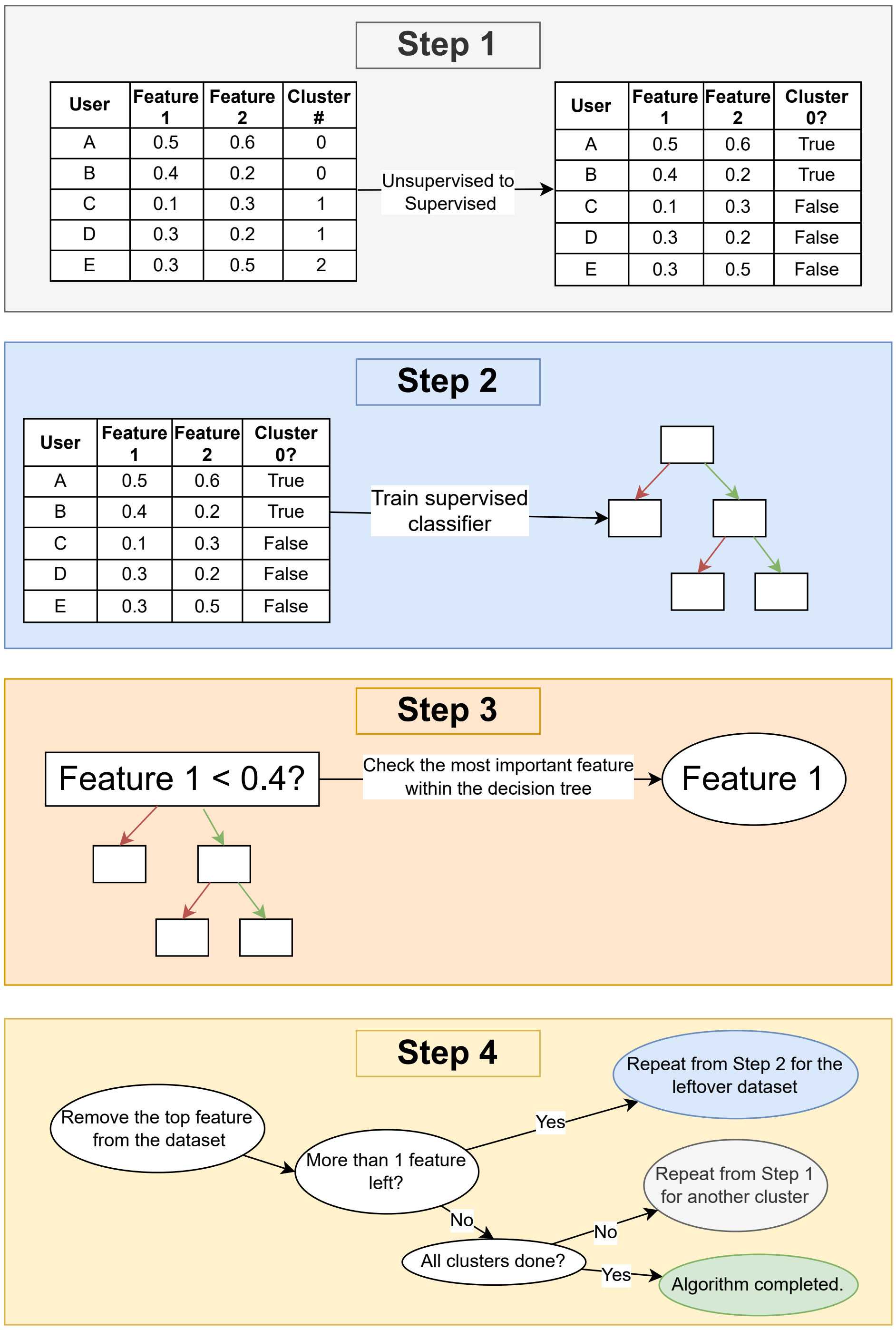
## Quantify feature importance

To investigate what the most defining features are that caused a user to be classified into a certain cluster, we apply the 'Unsupervised to Supervised' method, using an external library[8]. We selected this method as opposed to alternative options because of its easy steps and explainability. This approach converts an unsupervised

[7]Rand index: `https://scikit-learn.org/stable/modules/clustering.html#rand-index`

[8]Unsupervised to Supervised feature importances: `https://towardsdatascience.com/interpretable-k-means-clusters-feature-importances-7e516eeb8d3c?gi=a39843395880`

classification problem such as clustering into a supervised classification problem. For the supervised classification we use a model which is easily interpretable: the RFC (Random Forest Classifier). The steps taken are illustrated by Figure 2.3. 'One-vs-all' means that for every iteration, all users classified into a certain cluster are set as 'True', and all other cluster's users are set as 'False'. Then, the RFC is trained to distinguish between this cluster and all other clusters, using all features. The RFC produces a decision tree in which the top node, the first deciding factor, contains the most distinguishing feature. After this is found, the feature is removed, the RFC is trained on the reduced feature vectors, and another feature will come out on top. These iterative steps continue until there are no more features left. Then, we repeat all steps from the start for every other cluster.


Step 1
Unsupervised to Supervised
Step 2
Train supervised classifier
Step 3
Feature 1 < 0.4?
Check the most important feature within the decision tree
Feature 1
Step 4
Remove the top feature from the dataset
More than 1 feature left?
Yes
Repeat from Step 2 for the leftover dataset
No
All clusters done?
No
Repeat from Step 1 for another cluster
Yes
Algorithm completed.

| User | Feature 1 | Feature 2 | Cluster # |
|---|---|---|---|
| A | 0.5 | 0.6 | 0 |
| B | 0.4 | 0.2 | 0 |
| C | 0.1 | 0.3 | 1 |
| D | 0.3 | 0.2 | 1 |
| E | 0.3 | 0.5 | 2 |

| User | Feature 1 | Feature 2 | Cluster 0? |
|---|---|---|---|
| A | 0.5 | 0.6 | True |
| B | 0.4 | 0.2 | True |
| C | 0.1 | 0.3 | False |
| D | 0.3 | 0.2 | False |
| E | 0.3 | 0.5 | False |

| User | Feature 1 | Feature 2 | Cluster 0? |
|---|---|---|---|
| A | 0.5 | 0.6 | True |
| B | 0.4 | 0.2 | True |
| C | 0.1 | 0.3 | False |
| D | 0.3 | 0.2 | False |
| E | 0.3 | 0.5 | False |

**Figure 2.3:** The steps of the Feature Importance computation algorithm via the Unsupervised to Supervised method.

Chapter 3

# Related Work

This section reviews the existing literature about user access profiling. First, we survey the literature about the threats of improper user access control and detection possibilities. Then, we look at anomaly detection mechanisms based on access logs. Finally, we visit the work benchmarking their anomaly detection algorithms on the synthetic CERT Insider Threat dataset.

## 3.1 Access Control

For a user account to access a resource, they require the appropriate permissions. Within an organisation's digital infrastructure, this is usually arranged via Active Directory groups. A group can be granted read and write rights to a resource. If a user is a member of this group, they inherit all these rights. Managing such accesses for users and groups is called *Access control*. The Principle of Least Privilege is a concept within information security which describes that users should only have access to the resources they need. Not adhering to this principle forms a security risk if an adversary manages to control an account, such as in the case of account compromise or insider threats.

Research has tried to detect misconfigurations in such policies using different approaches. Das et al. [6] created Baaz: a tool that represents access control policies as a matrix, which for every user (column) and every resource (row) states whether access is granted. If a reference dataset exists on which resources every role should be able to access, users can be mapped to a role group with similar rights. The resources that share the same policies are clustered if no reference matrix is available. From this, clusters of users with roughly the same access permissions can be produced. After the clustering step, users and resources which do not fit into any cluster are marked as outliers. In this case, an *Outlier* can imply users with too many permissions, a security issue, and users who have too few permissions, an accessibility issue. The detection of accessibility issues is, to us, not a goal in itself, but repeated access requests through misconfigurations could lead to administrators becoming too generous in granting access permissions, hence leading to security issues again. In reality, such well-documented access control matrices or even a reference dataset on roles and resources are often not available. Therefore is more realistic to perform analyses on logs. Bauer et al. [7] performed log analysis to analyse the access attempts for *accessibility* problems. They used association rule mining [8], a type of inference that tries to predict future accesses through previous ones. Because such rules can repeatedly produce the same false association, a feedback mechanism was introduced to prevent these mistakes from reoccurring. However, since such labelling is performed manually, when applied on a larger scale it becomes labour-intensive, which is undesirable to a SOC.

## 3.2 Relational database user profiling through access logs

One of the goals of profiling users through access logs is to detect deviations from the norm. This is a use case for *anomaly detection*, for example, when someone suddenly changes behaviour to access more resources than usual or logs in outside of office hours.

The research area of user profiling and anomaly detection using logs is more mature within the field of relational databases. Even though this is a specific type of resource access, their methodologies are still relevant when considering broader resource types. Kamra et al. [9] mined SQL queries from the database audit logs, creating profiles to model normal database access behaviour. These profiles consist of *quiplets*, extracting different fields from the SQL query to determine what data is queried, such as tables, columns and constraints. To train their model, outliers were removed, under the assumption that the leftover queries were benign. Similar to Bauer [7], they used two scenarios: role-based and unsupervised. For the former, the role was available for every user executing query, and the Naive Bayes Classifier[1] was used to produce the probability that a user with this role would indeed execute such a query. The unsupervised scenario is approached as a clustering problem. Every user is mapped onto a cluster, and whenever a new query is issued, their model tests whether it is likely that the authenticating account issued this query. Both the Naive Bayes method and a statistical test are used to determine outliers. Bertino et al. [10] converted SQL queries into triplets with different levels of granular-

[1]Naive Bayes Classifier explanation: `https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c`

ity in a similar manner to the abovementioned quiplets. They used the Naive Bayes Classifier, trained through supervised learning using synthetic and real-world data. Every query included role data, and the classifier computed the probability that this role would execute a query with these properties. The testing was performed by altering the role of the query and verifying whether it was marked as an anomaly. In addition to verifying the precision and recall of their triplets with different amounts of training tuples, the researchers calculated the influence of the number of roles and database attributes for this classification. Discovering the proper number of roles for access control within an organisation is very useful to identify accessibility and security issues, such as Bauer [7] aimed to identify. A query within a relational database shares similarities to logon actions within an organisational domain because it is a user action connecting to a resource through characterizing features. The target table, parameters, and query conditions make up the access properties for SQL queries. A user connecting to a resource also has a destination host, and properties could be the connection or authentication method. Converting such information into a quiplet is essentially a feature vector fed into the ML (Machine Learning) algorithm. As they were able to cluster users based on such information and verify whether every new query falls within the cluster of the authenticated user is a favourable outcome for our proposed method.

## 3.3 Anomaly detection through user action monitoring

This literature research aims to find the best method for detecting access (attempts) that deviate from normal behaviour since this can indicate an insider threat or compromised account. Therefore we are looking at *anomalies* which deviate from the regular model. Furthermore, we desire a method that is as generic as possible to avoid significant manual labour before and during deployment. We look into different approaches applied in research with similar goals.

Pannell et al. [11] approached the *compromised account* insider threat similar to network monitoring through Intrusion Detection Systems (IDS). Their algorithm uses information from running applications, visited websites, open windows and keystrokes to profile users. These properties are converted into numerical values, such as the number of running processes every 30 seconds and the delays between pressed keyboard keys. Each element is statistically analysed for deviation from the user's historical profile. All elements have their own weight points, and when an element is regarded as anomalous, its points get added to the total score of that interval. The entire behaviour is marked as anomalous if the cumulative weight threshold is crossed. As their model has a learning phase, it requires historical data, which is initially full of anomalous behaviours. While the system learns the user's behaviour, it reaches an equilibrium where rarely any new anomalies are detected after a while. The research compares the different monitoring elements by placing a different person than the authenticated account in front of the computer, having them perform exercises and testing how long it takes for every element to mark the user's behaviour as anomalous. The keystrokes and viewed sites features recognised anomalous behaviour the quickest, but a weighted combination of all features performed the best. For access behaviour, a similar approach could be used by converting every feature into a numerical value, giving weight to every feature, and testing against the historical profile whether there is a statistical difference. If there is a difference, its weight score gets added to the cumulative anomaly score every interval. The session is marked as anomalous if the anomaly score crosses a threshold.

Wang et al. [12] analysed user behaviour through characteristics similar to those used by Pannell [11]. As they only test on labelled datasets, a supervised ML approach is possible through the use of Support Vector Machines. Their dataset includes biometric information, and they found that this characteristic worked well in profiling the correct individual and identifying whether the authenticated account belongs to the user behind the computer interactions. However, their testing setup allows for making assumptions on legitimate training data without anomalies and requires detailed datasets on user-computer interactions. Thus, the nature of this data and training assumptions make it infeasible to apply their method within a SOC.

When teaching an algorithm to distinguish between different users in audit sessions, if users share many behavioural traits, this causes many false positives. This issue was recognised by Garchery et al. [13], who went on to group similar individual users into clusters and found this led to better overall performance. Their approach to user representation and clustering is confusion matrices with actual and predicted users on the different axes. The main difference with regular unsupervised anomaly detection is that they start with a supervised dataset with identified user sessions. It is then converted into an unsupervised method through the clustering of similar-behaving users. Their identification of users is performed through similar information per session as other IDS mechanisms like Pannell's research [11] did, e.g. authentications, IP address- and temporal features. This methodology worked well for masquerade detection: when a user logs in via someone else's credentials. Therefore, it could possibly also detect when a regular user suddenly acts like an administrator. The features used are not solely access-focused, but they use features available to a SOC, such as destination host and temporal login features. Our research will determine whether such a subset of their features is also sufficient to classify users

consistently into their clusters.
Tabash et al. [14] converted the behaviour of employees into ten boolean variables to turn into a feature vector. They used Gaussian Mixture Models (GMMs) to detect anomalies with the ability to involve non-technical indicators in the form of sensitivity profiles: certain situations can cause parameters to become more significant, such as financial problems for an individual causing to become more sensitive to financial fraud. GMMs allow for a mixture of probability distributions, which is helpful for monitoring behaviour as this can change slowly over time. Their solution has a feedback loop through analysts to improve detection, as the analysts' input is a labelled observation. A GMM per employee is created, and the decision-making process is visually represented to get insight into which factors lead to the detection of the anomaly being triggered. Role information can also be included, as different roles may have different suspicious behaviour, such as the sales department mailing many files to people outside the company. Although quite useful, their approach with the sensitivity profiles is very intrusive, so generally not applicable within a SOC. The analyst feedback loop could work well in practice to reduce false positives, although this would require significant manual labour. Their ML is unsupervised and can include role information, so it could be helpful for comparing the user themselves to the rest of their role group.
In addition to ML, statistical analyses can be used for anomalous insider detection. Chen et al. [15] developed a methodology called Community Anomaly Detection System (CADS). They take the set of User-Resource pairings and form communities through this, visualised in a 2D grid with one node per user. Through several formulas, the distance of every user to the nearest community is calculated. If this distance crosses a threshold, the access behaviour is considered anomalous. The domain of CADS is Electronic Health Registries (EHR). As there are likely to be more unique EHR patient files than users in the system, this type of comparison likely does not hold within an organisation's digital infrastructure, as there will be fewer servers than users accessing them, reducing the granularity of user profiles. Chen et al. also introduce MetaCADS, which additionally considers the subject of the accessed files, such as the medical department. This improved the communities and detected outliers, but requires additional domain knowledge beforehand to categorise resources, so its SOC applicability would depend on the availability of information regarding groups of resources.
Gates et al. [16] focused on file accesses and detected the problem that when new employees have been onboarded, they do not yet have a regular access history profile. Many existing methods use a user's access history and analyse every new access, which is easily seen as an anomaly resulting in high false-positive rates. Instead, they created a scoring system using the user's history and current file ac-

cessed, considering the file hierarchy and the access history of other users. Those 'distance scores' are used to create a feature vector per user, and those features can be used to identify users and detect anomalies if a threshold is crossed. This scoring system is the core of the entire anomaly detection, but Gates et al. claim that this methodology also applies to domains outside the file system hierarchy. Another example is the shortest path between two URLs: how many hyperlinks it takes to get from one internet page to the other. The core of this detection mechanism is that there needs to be a numerical distance to express similarity between two accesses, which is hard to apply within our access type without significant domain knowledge of the network and occupations. This approach is likely not suitable for our research.

We have seen so far that related work can perform proper detection of malicious activity on user accounts, but mainly by profiling the user interaction with the computer itself, contrary to just the access properties. This leads to substantial feature vectors per person fed into an ML algorithm, which can perform well. Such experiments were, however, mostly performed in an experimental setting with intensively monitored test subjects. This, in practice, is undesirable, if not impossible, for a SOC. Additionally, as every experiment was conducted in a different setting, it is hard to compare the results to conclude which approach works best. This problem is solved by having a standardised test dataset, which we highlight next.

## 3.4 CERT Insider Threat dataset

The CERT Insider Threat Dataset [17] was designed by B. Lindauer from Carnegie Mellon University, containing multiple synthetic log files and user properties of an imaginary organisation. The dataset consists of files regarding different aspects of a user's digital behaviour, as well as organisational data such as users' roles and teams and their personality traits. This dataset spans two years and contains different types of insider threat scenarios. A subset of users performs malicious actions somewhere in the course of the two years, in which actions are properly labelled. This makes it very useful for benchmarking insider threat detection algorithms, which it is often used for in related work.

An unsupervised user behaviour modelling approach for anomaly detection was developed by Sharma et al. [18]. They used Long Short-Term Memory (LSTM) [19] auto-encoders to model user behaviour based on session activities. An auto-encoder is a neural network model which tries to recreate its input in a compressed manner. In the LSTM approach, the goal is to represent the entire recurrence of events as a fixed-length vector. This first requires a non-anomalous dataset to compute the reconstruction error used for thresholds, after which the model can identify outliers. The input data consists of feature vectors describing user actions. These actions are kept track of within both static time windows and user sessions to compare performance. They used the CERT v4.2 dataset to benchmark their approach. Sharma recognises that it is harder to find a trade-off between false positives and missing anomalies when using fixed time windows to extract features. It is hard to detect an anomaly for short windows, whilst for large windows, the behaviour could become too generic. Therefore, the model takes all events within one session as feature vector input, which comprises 44 features. These events include numerical features such as to how many devices a connection was made or how many job searching websites were visited and categorical features such as the role or functional unit. All numbers are normalised to a range of (0,1). Sharma et al. found that off-hour activity is critical information, just as whether usage is from a user's regular pc. Their feature vectors are chronologically divided into training, validation and testing. Unsupervised LSTM Recurrent Neural Networks [20] are used to prepare the model as this allows for temporal analysis when one action depends on the other. In this LSTM network, the session is anomalous if a reconstruction error is larger than the threshold computer during the learning phase. After 95 sessions per user, they could predict with over 90% accuracy whether a session was anomalous. Their type of data used is similar to SOC access data, so LSTM auto-encoders have the potential for discovering insider threats from a SOC standpoint.

Tuor et al. applied deep learning to identify insider threats in an unsupervised manner [21], through, amongst others, LSTM Recurrent Neural Networks. Using both categorical (e.g. role, supervisor) user attributes and continuous count features tracking 408 different activities in a 24h time window, they mark a user's day as regular or anomalous. They found that the LSTM model is quite capable of generalising. Moreover, keeping track of the sequence of actions in its hidden memory saves the analyst's time for creating aggregate count-style features that related research has attempted and can be applied to single log entries.

Le et al. performed multiple types of research on the CERT dataset. In one paper [22], they recognise the problem of scarcity of labelled data to make decisions on. They developed a user-centred insider threat detection system, focusing on different data granularity levels. Their feature vectors consist of the user actions from a week, day, session or sub-session, generating vectors with 200 to 1100 features, and the number of vectors is respectively 4.000.000 to 100.000. Focusing on the security analyst's perspective to provide them with clear information makes it similar to our SOC-based approach. They applied several supervised ML algorithms: Logistic Regression (LR), Neural Network (NN), Random Forest (RF) and XGBoost (XG).

As the CERT dataset includes different malicious scenarios, Le also tested which approaches work best for what type of scenario. They performed classification of both instances (one log entry or feature vector) and users (whether an employee turned malicious at some point). From these two, the user-based detection resulted in the highest precision rates. Additionally, for the RF classifier, they performed feature analysis based on Gini impurity. This is used to infer how important every feature within the vector is for the ML algorithm. Realising this effectivity per type of threat scenario and the importance of different features is very useful to realise what input data is relevant and what types of threats still go undetected. After applying supervised ML approaches, the same authors performed an unsupervised learning-based anomaly detection approach for insider threat detection [23]. They applied four methods: Auto-encoders, Isolation Forest, Lightweight On-line detection of anomalies, and Local Outlier Factor. Additionally, they varied the representation of temporal information within data, optimising how behavioural changes over time are detected. Variations include concatenating feature vectors from a specific timeframe, explicitly adding the percentile, or the difference between a number and its mean or median, to the feature vector. Every CERT version dataset has a different organisational structure and scenarios. They looked into their model's generic applicability and robustness for different datasets by training on one CERT dataset version and applying it to another. Finally, applying multiple algorithms allows voting: An event is only marked as anomalous if the number of algorithms marking it anomalous crosses a threshold. Combining the knowledge this way improved their approach's detection ability and robustness. The auto-encoder worked best of the individual ML methods, using the percentile temporal data representation, possibly better than LSTM.

## 3.5 Literature inferences

After researching related literature, we formulate an answer to the first research question: *Which non-invasive features readily available to a SOC are relevant in describing user access behaviour?*. First, we look at how similar research converted logs into profiles. Then, the selected ML method is described in detail. Finally, we find the corresponding Windows Events to use as input data for our case.

### User profiling

The general approach from related research is to convert user behaviour into a vector with numerical values describing the user's behaviour. This requires a method to convert (access) logs to such vectors, which is approached differently in the literature. Pannell et al. keep track of a user by counting specific actions, also referred to as *frequency feature*, over a session of 30 seconds, such as keystroke analysis and the amount of running applications. For access behaviour, other elements could be kept track of and counted in a time interval, such as the different destination hosts, authentication methods and logon types. Sharma et al. used a session-based approach instead of a predefined timeframe. They kept track of, amongst others, whether a logon was during working hours, the day of the week and the logon until logoff session duration. User activities were tracked by Tuor et al. by counting them in a 24-hour time window. For every action taken by the user a value was increased, based on a decision tree, such as a separate counter for a logon action between 12 am and 6 am. Le et al. found that for frequency features, the session-based and 24-hour time windows data worked best. In addition to frequency features, they used statistical features: descriptive statistics such as the mean, median and standard deviation of properties related to user actions. From this, as they kept track of all activity within the session, too, we infer that the daily frequency features will suit our approach best. Within the CERT dataset, linking the temporal dependencies is essential. When someone logs in on a deviating hour, opens a cloud storage website, plugs in a USB and starts uploading, the sequence of events makes it an apparent anomaly of, for example, espionage attempting to exfiltrate company secrets. Single log entries are likely not enough to detect deviating behaviour. It needs to be seen in context, hence we also need to add features using data from previous intervals to track changes over time.

In addition to considering what worked well for similar research, we use threat intelligence to infer what behavioural change relates to suspicious behaviour. One of the steps within the Cyber Kill Chain, a framework describing the steps an adversary takes before reaching their goal[2], is lateral movement. This consists of searching around the network for interesting resources. In this process, we expect an adversary to access more resources than the user account normally does. Therefore accesses performed for the first time, *novel accesses*, is a factor to keep into account, and a higher value than average could indicate lateral movement. It is also relevant whether an access was performed during or outside office hours. Malicious adversaries tend to operate outside of office hours, either during the night or on the weekends. This tactic was also mentioned by ransomware group Conti, whose playbooks were leaked in February 2022 [3]. They name at one step using

[2]Cyber Kill Chain: `https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html`

[3]Conti playbook leaks: `https://www.bleepingcomputer.com/news/security/conti-ransomwares-internal-chats-leaked-after-siding-with-russia/`

Remote Desktop Protocol to log onto a victim machine, but only at night. This is likely because otherwise, it might interfere with the actual user's session, who will get alerted. Additionally, operating outside of office hours is likely to delay a reaction from the victim's side.

The profiling of user behaviour and detection of its deviations is tackled in different ways in the literature. The two main ways of processing the feature vectors to profile user behaviour are via statistical tests or ML. Statistical tests allow for a simple method of giving different weights to vectors. On the other hand, ML is more widely used by related work and has yielded better results. Most often, a type of anomaly detection was used to detect insider threats or compromised accounts, ranging from, e.g. LSTM Auto-encoders to Gaussian Mixed Models. Applying such a mechanism is out of scope for this research, as it would require further research on which types of anomaly detection are suitable for CERT and production data and how they work under the hood. Anomaly detection is possible in both a supervised and unsupervised manner. However, the supervised version is generally only suitable when ground truth is available on, for example, insiders or user roles. Before able to detect anomalies, we need to be able to profile users meaningfully and consistently over time. For that, we focus on clustering to develop this profiling methodology.

## K-means clustering

There are several methods readily available to use in python libraries, such as sklearn [4]. We do not survey all of them, but choose an explainable one to test our research on. The explainability is important because of its intended use for real-world data involving humans, it is impermissable for an algorithm to mark a user as suspicious without a transparent reasoning behind it. The reason for choosing K-means as a clustering method is threefold. First, different clustering mechanisms are suitable for different dataset characteristics, such as whether the groups are even in size or what order of magnitude the number of clusters is. Because our approach should be as generally applicable as possible without requiring domain knowledge on the input user base, we cannot describe the dataset beforehand. Second, clustering algorithms require input parameters describing, e.g. the dataset or the desired amount of clusters. Similar to the previous option, there are very few parameters we can automatically generate based on the input user dataset. Finally, this research is a proof-of-concept on what inferences we can make by clustering users based on access behaviour and be able to explain the outcome, not to find the

[4]sklearn library clustering methods: `https://scikit-learn.org/stable/modules/clustering.html`

most optimal clustering algorithm fitted for a given dataset. Therefore we decided to use the general-purpose, straightforward K-means clustering algorithm, which solely requires the desired amount of clusters as input parameters, which can be generated using the Elbow heuristic. The main drawback of this algorithm is that it does not perform very well on clusters of different sizes. This could, in future research, be solved by profiling the feature vectors before clustering and selecting the clustering approach based on this, but discovering the perfect clustering algorithm falls outside of the scope of this research.

### Access log data used

There are multiple stages where it is possible to monitor user access behaviour. During authentication via Kerberos, when a user authenticates against the DC a Windows Event describes what type of service a TGS (Ticket Granting Server) had been requested for. Then a user logs onto the host and performs actions on this device. Our goal is to detect it when a user suspiciously changes access behaviour from their regular profile. We want this detection to be as generally applicable as possible. Logged authentication events are hard to correlate because TGSs are commonly not sent to the SIEM and authentication tokens are valid for en extended period of time, making monitoring difficult for, e.g. outside office hour activity. We also need to decide which devices we can use the logs from. Logs from endpoint devices such as employee laptops generate many events. Since there are many such devices, a SIEM would be flooded with data if they were all onboarded to the SIEM. Therefore, these endpoint devices are commonly not onboarded, making them hard to monitor. As previously mentioned, servers are among the most valuable assets within an organisation, so we will limit our monitoring to those servers. Additionally, many employee devices will need to connect to such servers to access domain-wide resources. Therefore we limit our focus to the actual logon event (4624) onto servers, as this allows us to observe *which account* connected to *which host* at *which time* and *how*. Logoff events can be included to determine session duration.

### Feature construction

Using related literature and threat intelligence, we formed an approach to create feature vectors from access logs. Within time windows of 24 hours, several access properties are counted per user, such as destination host and whether the access was performed outside of office hours. These daily features are aggregated by computing the mean, median and standard deviation to describe a user's behaviour over a period of time. This results in a single feature vector per user, which can be used as input for the K-means clustering algorithm.

Chapter 4

# Methodology

This section describing the research methodology is threefold. First, we elaborate the general methodology of using login log data to profile users and visualise the result. Then, we show how these results can be quantified in terms of performance. Finally, we show how both methods can be applied to Sentinel data to be able to incorporate data from a real-world SOC.

## 4.1 CERT Insider Threat dataset

Multiple versions of this dataset exist [2], where v4.2 is used most often. This is because it is a 'dense needles' dataset, meaning it has an unrealistically high amount of users performing malicious behaviour. The user's digital behavioural in this dataset includes logon information, network traffic, mail conversations and mobile data carrier usage. Of the available files, we solely use `logon.csv` as this resembles Windows logon Events the most.

### Logon logfile characteristics

The logon data from the CERT Insider Threat dataset is described as follows:

- *1k users, each with an assigned PC*
- *100 shared machines used by some of the users in addition to their assigned PC. These are shared in the sense of a computer lab, not in the sense of a Unix server or Windows Terminal Server.*
- *Systems administrators with global access privileges are identified by job role "ITAdmin".*
- *Some users log into another user's dedicated machine from time to time.*
- *A small number of daily logons are intentionally not recorded to simulate dirty data.*

Every log entry contains data in the format '`id;date;user;pc;activity`', for example:

```
                        id                 date     user       pc activity
{S2U3-D7TD98CV-4593RNVD} 2010-05-01 01:42:00  BWP0202  PC-7445    Logon
{C1G9-N7GE88JL-2827CYHQ} 2010-05-01 01:45:00  RHO0732  PC-6232   Logoff
{V3F7-X7BB51OQ-7565PUAK} 2010-05-01 02:09:00  DMK0257  PC-1279   Logoff
{W7Y7-U0CP28MF-3057ELAF} 2010-05-01 02:16:00  RTO0313  PC-3311    Logon
{L1X1-E8VJ74TW-2769UGXY} 2010-05-01 02:21:00  RTO0313  PC-3311   Logoff
```

The most relevant information we can take from this dataset is *which user* logged onto/off of *which PC* at *which timestamp*. Therefore apart from knowing the device a user accessed, the temporal information is an important data point used to profile users.

## Profiling steps

### 1) One-hot encoding

The first step in converting the log file to a format which can be used as feature vectors is by a variation on *One-hot encoding*. This is a method used to encode a categorical feature without logical order. Using this method, every categorical feature with *n* categories is transformed into *n* binary features [1]. In our case, for the categorical feature *PC* which is transformed into about a thousand binary features: every user is a row and every PC is a column, and for every time a user accessed a PC in the selected logs the value is *incremented by 1*, if they did not access the PC at all the value is a *0*. Using this as a user's feature vector causes users who access the same groups of resources to be clustered together. The resulting table looks as follows:

| user | PC-7445 | PC-6232 | PC-1279 | PC-3311 | .. | PC-9324 |
|---|---|---|---|---|---|---|
| BWP0202 | 1 | 0 | 0 | 0 | .. | 0 |
| RHO0732 | 0 | 6 | 0 | 0 | .. | 0 |
| DMK0257 | 0 | 0 | 3 | 0 | .. | 0 |
| .. | .. | .. | .. | .. | .. | .. |
| BIS0247 | 0 | 0 | 0 | 0 | .. | 2 |

This example already highlights the pattern of the dataset: most users only access very few PC's. This is in accordance with the dataset description, where most users

[1]One-hot encoding information: `https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9`

have their own dedicated PC. There are only 100 officially shared machines and only the ITAdmins are able to access all PC's. This is not in line with what we expect from the real-world scenario with likely many fileservers and printers as accessed devices, and makes it harder to cluster based on user's overlap regarding which PC's they access. Therefore for this dataset we focus on the temporal aspects of the accesses.

### 2) Feature engineering

To create feature vectors we need to look at more than just how many times a user accessed a certain PC, especially because for this dataset there are only few resources shared amongst many people. Section 3.5 describes why these features are relevant for monitoring access behaviour. The features are elaborated below and the overview is shown later in Table 4.1

**Unique accesses** The `uniq_accesses` parameter counts how many different PC's a user has accessed. Using this, we can filter out users who have only accessed $< N$ machines from our analysis. Too many users with identical access behaviour skews the K-means clustering method, as there will be large number of users with the same feature vector. This causes a huge 'gravitational center', meaning most centroids will move towards this user group. This lowers the granularity to which other users can be classified into clusters. Additionally, such users are less suspicious when ignoring the significance of certain machines. They have too few data points for our profiling methodology, as we can only incorporate temporal behaviour and nothing with regards to *which* resources were accessed to match their behaviour with that of their peers.

**Timeframe parameter** We introduce the *timeframe* parameter, used to define the granularity of features. Using this, we divide the total duration of the log data into these shorter timeframe lengths and count how many accesses (logons) a user performed within that timeframe. If the timeframe is increased to one week instead of one day, there will be less and more generic feature vectors per user. If we decrease the timeframe to 1 hour, every user will have many more statistical feature vectors per timeframe, which values deviate a lot. This value is for now is set to 24 hours, based on similar research [21] [22]. From these `access_amt_in_timeframe_yyyymmdd` columns we calculate the mean, median and standard deviation, forming *statistical features*. Such descriptive statistics were also used by Le et al. [22] in their research on data granularity levels for such profiling.

**Office Hours** Another important characteristic within logon data is whether a logon took place during or outside of office hours [18]. A user accessing more resources outside of office hours than regular could be indication for an account compromise. Therefore for every timeframe, we calculate how many of the logins were outside of office hours. For weekdays this is defined as before 7:00 or after 21:00, as based on 5.1. A weekend access is always outside of office hours. From these features per timeframe we also calculate the mean, median and standard deviation as statistical features. The same features per timeframe and corresponding statistical features are computed for accesses *within* office hours.

**Novel accesses** Per timeframe, we count how many of the resources were accessed for the first time: `newly_accessed_ resource_amt_in_timeframe_yyyymmdd`. This is important to profile an account used for network discovery or propagation. Such behaviour will make many accesses for the first time, ones which the legitimate user or the person whilst they were still acting benign had the rights to but never previously accessed. As with the above features, the mean, median and standard deviation statistical features are computed.

**Cumulative difference from the mean** To find patterns in changing behaviour over time, we introduce a new feature: the *cumulative difference from mean accesses until timeframe X*. This feature is used to keep track of trends over time, such as a user continuing to access more resources over time. This feature can be compared to the Cumulative Sum control chart (CUSUM). The CUSUM is a method to monitor change: A temporal sequence of values is standardised and compared to the expected value. If the difference is larger than a pre-defined parameter (e.g., half a standard deviation), it gets added to the running cumulative deviation sum. There are separate running sums for both positive or negative changes in value. This is most likely not applicable to our case with daily accesses. Here our 'expected value' is the average value up until a given timeframe, and any timeframe where a person does not access any resources (e.g., weekends or vacation days) would cause a significant negative deviation. This lowers the mean and causes the next working day to be a significant positive deviation et cetera. In our implementation, positive and negative deviations can cancel each other out, so the only way for this cumulative difference value to increase significantly over time is if on average, a user keeps accessing more resources than the time period before. It is also possible to monitor for especially *accesses outside of office hours* or *novel accesses* by simply using that feature instead of *number of accesses per day*. An example of how this feature is constructed is provided in Figure 4.1.

1. We start off with a user's *number of accesses per timeframe*.

2. For every timeframe, we construct *the mean number of accesses until that timeframe*.
3. Subsequently, we compute the *difference between the mean number of accesses until timeframe X and the actual number of accesses on timeframe X*.
4. Finally, we add this number to the *cumulative difference from mean accesses until timeframe X*

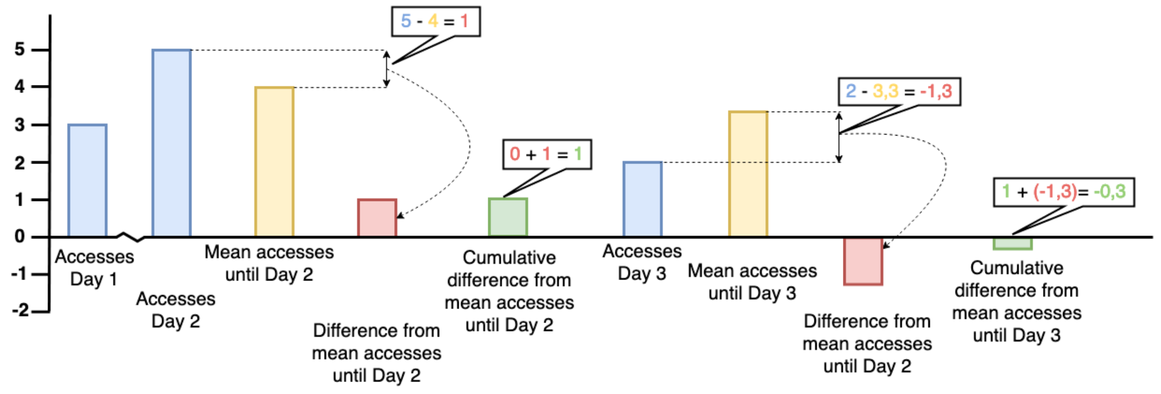

5 - 4 = 1
0 + 1 = 1
2 - 3,3 = -1,3
1 + (-1,3)= -0,3
5
4
3
2
1
0
-1
-2
Accesses Day 1
Accesses Day 2
Mean accesses until Day 2
Difference from mean accesses until Day 2
Cumulative difference from mean accesses until Day 2
Accesses Day 3
Mean accesses until Day 3
Difference from mean accesses until Day 2
Cumulative difference from mean accesses until Day 3

**Figure 4.1:** A bar chart showing how the cumulative deviation from the mean accesses is calculated over time.

### 3) Feature standardisation

Before the features can be used for clustering they need to be *standardised*[2]. The purpose of this is to be able to treat features of different units equally, but still preserve characteristics such as outliers to avoid signal loss. This way, a feature can still be used to compare one user to others. This is especially important for Machine Learning methods which calculate the distance between points in the multidimensional spectrum. Standardisation is performed by removing the feature's mean of and dividing by the standard deviation, according to the formula $z = (sample - mean)/stdev$.

### 4) Optimal cluster amount

For K-means clustering only one input parameter is required: the number of clusters. A widely used heuristic for this is the Elbow method. This method plots the explained variation of the number of clusters: per hypothetical number of clusters, how much the multidimensional points appointed to that cluster are spread apart

[2]sklearn library standardisation function: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

| **Feature** | **Particularities** |
|---|---|
| Total logons during weekend | |
| Total logons outside of office hours | |
| Total logons | |
| Percentage of accesses outside of office hours | |
| Percentage of accesses during the weekend | |
| Total unique accesses | |
| Access amount per timeframe | |
| Novel accesses per timeframe | |
| Accesses outside of office hours per timeframe | Office hours are infered from the dataset, weekend days are fully outside of office hours. |
| Accesses during office hours per timeframe | Office hours are infered from the dataset, weekend days are fully outside of office hours. |
| Session length per timeframe | |
| Cumulative deviation from the running average number of accesses per timeframe | |
| Every destination host | The total number of times the user accessed this resource, for every resource. |

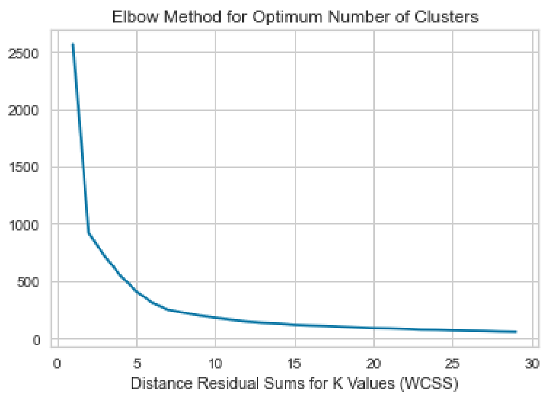
**Table 4.1:** The features engineered to be available for the CERT dataset. The default timeframe is set to 24h and the mean, median and standard deviation are taken over the entire period.

form the cluster average. It is called the 'elbow method' because the plot looks like an logarithmic curve. The optimal number of clusters is the 'elbow', the value where an increase in computational cost is no longer worth the diminishing returns. We use the elbow method as implemented in the `yellowbrick` library[3]. An example plot of this is given in Figure 4.2.

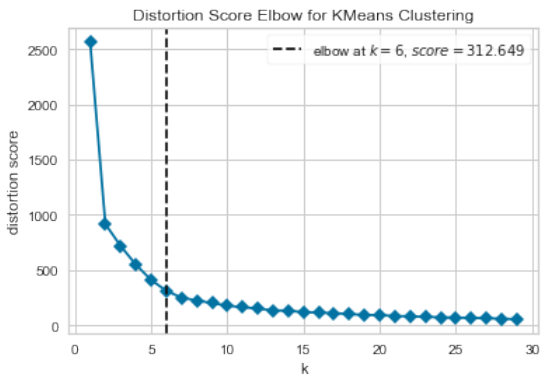
### 5) K-Means clustering

Using the features as computed in step 3 and the K-value from step 4, the users are classified into a cluster. The randomness for the initial centroids is seeded for reproducability.

[3]Yellowbrick python library: `https://www.scikit-yb.org/en/latest/api/cluster/elbow.html`



(a)



(b)

**Figure 4.2:** An example of the elbow heuristic applied to the feature vectors created from the CERT v4.2 dataset.

## Cluster analysis steps

### 6) Visualisation via Principal Component Analysis

After the clusters have been computed, we visualise the results to comprehend them better. Principal Component Analysis (PCA) is a method to reduce the dimensions of points in multidimensional space, which is in our case the feature vector per user. This is performed whilst retaining as much information as possible. We aim for two resulting components, to be able to display the results in a 2D plot. It is important to note that PCA is not the same concept as computing feature importance and removing the least valuable feature. The latter inevitably leads to information loss. PCA tries to express a combination of features in a new *component*, by inputting a certain percentage of every feature. An example of this for the first PCA dimension (PCA 1) and the second PCA dimension (PCA 2) is shown in Figure 4.3.

Since variance is information, the first coordinate (principal component) captures the greatest variance, the second coordinate captures the second greatest variance et cetera. A scree plot shows how much variation every component captures from the feature vectors. From the scree plot in Figure 4.4 we can see that in this example 80% of variance can be explained via just two principal components, which makes it suitable to display in 2D using PCA 1 and PCA 2.

### 7) Compute feature importance per cluster

We apply the 'unsupervised to supervised' method as described in Section 2.3 to produce an ranked list of features based on influence, per cluster. The highest ranked feature is the one which is most distinctive for that cluster and had the largest effect on users to be clustered together. Knowing the most important features, we

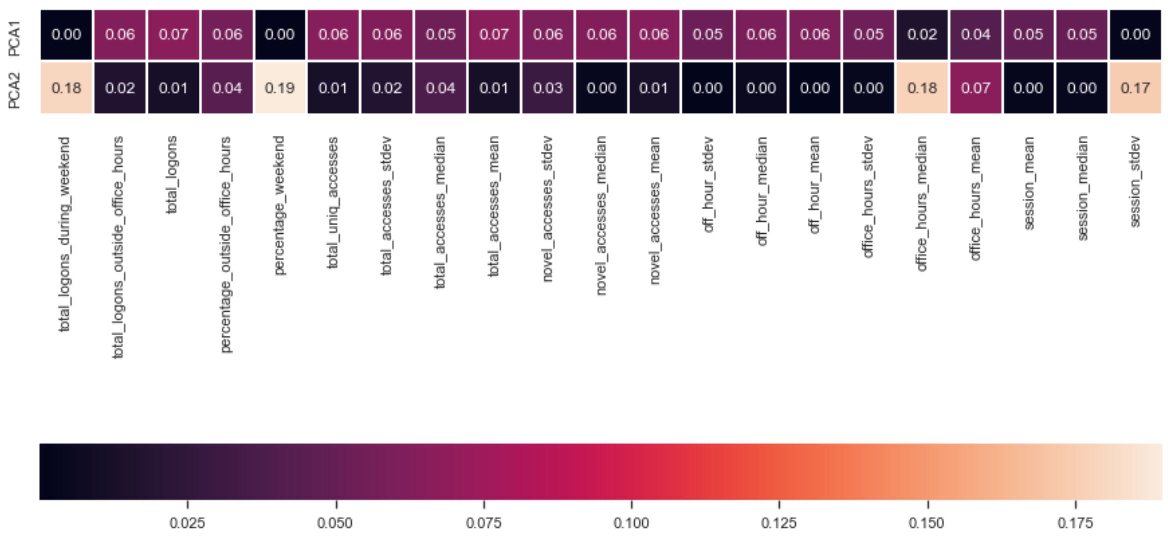


**Figure 4.3:** A heatmap which shows how much variance of every feature contributes to each PCA dimension. In this example we see that the second PCA is made up of almost entirely of the 'total_logons_during_weekend', 'percentage_weekend', 'office_hours_median' and 'session_stdev' features.

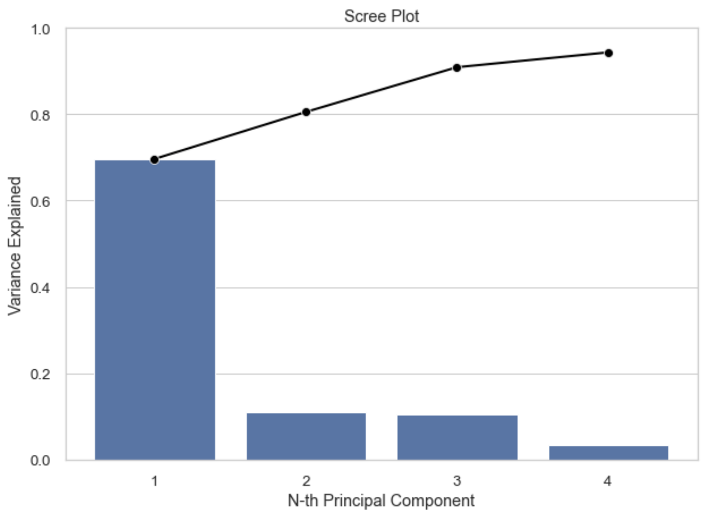


**Figure 4.4:** An example of a scree plot of PCA variance per feature, showing that a 2D plot captures 80% of the multidimensional variance.

can look up its average value among all users within the cluster to describe them. For example, IT Administrators could be differentiated to the rest of the users by their accesses to a wide variety of servers and high number of accesses outside of office hours for maintenance.

**8) Compute stats per cluster/role group**

In addition to looking what features are important, we need insight in the values of those features per cluster and per role group. Therefore, per cluster, we compute the mean, median and standard deviation of every feature. Using the Role information from the CERT dataset, we compute the same values for every role group for every feature.

**9) Quantifying performance**

There are multiple ways to quantify clustering performance, depending on the motivation. We quantify performance in two different ways. First, to see if clusters can be user to group people of the same role, we compare the clusters via the role information as provided in the CERT dataset. This can be done either visually or using the Rand index[4]. The Rand index is earlier elaborated upon in Section 2.3, and gives a score between 0 and 1 on how similar two cluster divisions are. We use the version which adjusts for chance, to compensate for 'lucky guesses' by our clustering algorithm.

Another method to quantify clustering performance is to see how stable they are over time. This is achieved by applying the clustering algorithm based on the data of e.g. one month, and doing this for several months during a year. Then, we compare each month pair-wise via the abovementioned Rand index, using just the data from users which were active in both periods and filtering out the malicious actors. This will result per pair of months, a score on how similar the clusterings are. There is to our knowledge no literature performing such comparison over time hence we have no baseline of what values would be acceptable or good. Therefore, we assume that benign user's behaviour does not change between months and aim for the values closest to 1.0 as possible, as this would mean an identical clustering.

Having stable clusters means a SOC can take a user's access behaviour over a certain time period, compare it to the behaviour of all other users and determine whose behaviour is alike. If we assume that the majority of users within a cluster stay benign, a user leaving this cluster over time can be regarded as a suspicious

[4]Rand index adjusted for chance: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html`

access behaviour change. A SOC analyst would take a closer look at the historical behaviour of the user and gather contextual data to determine whether the behaviour is potentially malicious. If too many users change clusters two problems arise when comparing different time periods. First, as a SOC analyst looks at all users changing behaviour to determine its legitimacy, more users being marked as having suspicious behaviour leads to an increase in manual workload. Second, if users change behaviour this will affect the location of their cluster's centroid. The dynamic element of changing human behaviour is part of the reason why we expected ML to be more suitable for suspicious change detection rather than static thresholds. However, if many users change behaviour and thereby possible switch clusters, this affects the locations of all centroids. Then, it is not trivial anymore to determine which centroid's cluster belonged to which group of users. Determining this is necessary to detect which users changed behaviour more than the rest of their cluster. Therefore, this approach would only work if the number of distinguishable user groups stays the same, and that the majority of the users stay in their own cluster over time. Huge behavioural shifts could be avoided by using smaller time periods, however this increases the dataset skew as relative changes have a greater influence so might work counterproductive. We go further into this in Section 6.3.

## 4.2 Sentinel

For the analysis within Sentinel the analysis methodology is comparable to CERT's Section 4.1, only the data types and performance quantification possibilities are different. Section 2.2 elaborates further on the functioning of Sentinel, including the data and analysis possibilities. The analysis is performed within a Sentinel Notebook, which is a Jupyter Notebook whose kernel runs on an Azure virtual machine. The Sentinel Notebook supports Python, ML libraries and supports integration with logs stored in Sentinel.

### 1) Ethical caveat

Because this research incorporates data generated by humans, it requires approval by the ethics committee of the University of Twente. To gain this approval, we had to set strict limits on what our data and inferences can and cannot be used for.
Because the dataset concerns human behaviour, a skew towards users behaving in a particular manner is certainly possible. We need to keep this skew for this approach to function in the real world. This type of analysis and user profiling is also already commercially available via products such as Microsoft's UEBA. One of the safeguards to avoid profiling discrimination is that if input from this research

would be used in a SOC, there is always a human in the loop. No decision to follow up on an alert is made by an algorithm. The only actions that would happen if a user were to be flagged as displaying suspicious access behaviour, is that a SOC analyst would gather contextual data to determine whether the behaviour is actually suspicious. If and only if this is the case, a customer could be alerted following agreed upon procedures with the SOC to determine the legitimacy of the access behaviour. It is the responsibility of the SOC analyst to be this human in the loop between a SOC alarm and possible consequences for the involved users. The European Union has proposed a law on artificial intelligence[5], which determines for different types of AI which risk they entail. Our proposed solution falls under their 'minimal or no risk' category, which is permitted with no restrictions. We accept that a part of ML is black box, but at every step we get insight into the profiling process to make it as explainable as possible. Any data or inferences stay within the scope of this research. When during the research any potentially malicious behaviour is encountered, this is escalated to the Northwave SOC via the regular procedures. To avoid misconceptions about what our algorithm can and cannot do, in Section 6.2 we reason on the uses and limitations of our methodology. Any ML model that is trained during this research gets discarded after its use, only the methodology remains.
The goal of this research is to be as little invasive as possible, whilst staying usable to a SOC with regards to suspicious access behaviour detection.

### 2) Gathering data

We query the SecurityEvent table for the Windows EventID's 4624 (logon), 4634 and 4647 (both logoff) for a pre-defined period. From this we extract the `Timestamp;User;EventID;Computer;Logon type;Authentication type;Source IP`. We convert the EventID into either "Logon" or "Logoff".

### 3) Feature creation, standardisation and elbow heuristic

First, we plot the hourly activity in a bar plot to interpret office hours parameters. Sentinel logs contain more information regarding accesses than the CERT dataset. This enabled us to create more features describing user behaviour. We include per access the *logon type*, *authentication method* and *source IP*. The most common types of logons are *Interactive* (physically via a keyboard), *Network* (as with mounting a Fileserver to a host) and *Remote Desktop*. As mentioned in Section 3.5, an account performing an RDP connection to a host where they usually do not connect

[5]EU AI Act: `https://artificialintelligenceact.eu/`

to via this method is suspicious. Additionally, the source can reveal potential account compromise in the case that a user logs in from different locations than they used to. Via feature importance metrics we determine whether features actually contribute to a granular and consistent clustering. For authentication, the most common methods are *NTLM* and *Kerberos*. Additionally, the source IP where the connection originated from is logged. Table 4.2 shows the features additionally available to the Sentinel data compared to the CERT dataset.

| Feature | Particularities |
|---|---|
| Access per timeframe with logontype X | X = all different logontypes, e.g. Network |
| Accesses per timeframe using authentication method Y | Y = all different authentication methods, e.g. NTLM |
| Accesses per timeframe from source Z | Z = the IP where the access originated from, or which subnet, e.g. 130.0.0.0/8 |

**Table 4.2:** The features engineered to be available for the Sentinel dataset, in addition to the those of the CERT dataset. The timeframe parameter is set to 24h, of which the mean, median and standard deviation taken over the entire period.

Not every account causing logon and logoff events are human users, Windows also contains *service accounts*. Such accounts exist to perform task in the context of the operating system. Examples are Machine accounts, created for on-premise computers and recognisable by the hostname followed by a '$' sign, or NT_AUTHORITY accounts, accounts that act on behalf of the user. Every account has a Security Identifier[6], which contains information on what type of account it belongs to. Accounts owned by users can be recognised by the 'S-1-5-21' prefix in their, which identifies user accounts within a custom domain[7]. We also filter other accounts which based on their username reveal they are automated, such as Healthmailbox accounts[8] which are automated to check for exchange server connectivity. When applying clustering, these types of accounts were consistently placed in groups of their own. This is undesirable as these account types are not our point of focus and reduce the granularity among the other clusters because of their influence in the standardisation step and every iterative movement of the centroids during k-

[6]Windows SID: `https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-identifiers`

[7]Account SID prefixes: `https://renenyffenegger.ch/notes/Windows/security/SID/index`

[8]Healthmailbox accounts: `https://techcommunity.microsoft.com/t5/exchange-team-blog/exchange-2013-2016-monitoring-mailboxes/ba-p/611004`

means clustering. Therefore the abovementioned accounts are removed before the standardisation step. After this, the features are standardised and the elbow method is applied to deduce the optimal number of clusters.

### 4) Clustering

Using the optimal number of clusters, we apply K-means clustering. The scree plot shows how much information is lost by displaying the clusters in two dimensions, and the weight of every feature for each Principal Component is outputted. The cluster plot only feature username since role and insider information is not or not as clearly retrievable as with the CERT dataset.

### 5) Quantifying clustering performance

After every user has been classified into a cluster for their access behaviour over a given period, we again need to quantify the quality of this clustering. The two different ways for this are to see how our clustering progresses over time, and comparing our own clustering to the groupings created by another party.

**Clustering over time** Similar to the CERT dataset, we can compare the formed clusters for different time periods. Because of the data retention policies within Sentinel it is only possible to query data as far as 90 days back in time, limiting the temporal comparison possibilities. Additionally, we have no knowledge on potential insiders, so have to assume all profiled users are benign. Therefore, the number of periods we can compare will depend on how much data is required to form a baseline profile for users. As with the CERT analysis, we compare every period pair-wise via the Adjusted Rand index.

**Comparing clusters to the IdentityInfo table** Sentinel allows for multiple approaches to compare our own clustering. This requires combining the information of multiple Sentinel tables, which are elaborated upon in Section 2.2. From the `SecurityEvent` login event we extract the *SubjectUserSid*, with which we can lookup user information within the `IdentityInfo` table by querying this value as *AccountSID*. The latter table includes several values describing a user's affiliations:

- **AssignedRoles**: A list, commonly describes whether someone is (which type of) admin. This value concerns the roles a user has within the Azure cloud
- **Department**: A single string
- **GroupMembership**: A list of all the groups the user is a part of, this is equivalent for on-premise AD infrastructure and Azure cloud

- **JobTitle**: A single string

The access events we monitor for are related to the on-premise machines, so the AssignedRoles list does not guarantee correlation with the on-premise groups, hence this variable cannot be used for our analysis. Furthermore, the Department and Jobtitle fields would be suitable as they place the users in distinct groups, however in practice these are not filled out for a large portion of an organisation's users, harming the analysis possibilities. We are left with the GroupMembership list. As these can consist of multiple groups overlapping for different users, we cannot compare it to the clustering as with other distinct groups. Therefore we need a different method to compare our clustering to these group memberships. The method for this is described in Algorithm 1. This results in a score between 0 and 1 per user, which is the average similarity per user. This is created via pairwise comparing a user's group memberships to the group memberships of every other user in their cluster.

**Algorithm 1:** Method to quantify how similar GroupMemberships are to the cluster division

```
input : The matrix of users dataset, including every user's cluster and
        group memberships
output: For every user the score: the percentage of overlap between the
        user's own group memberships and that of the other users within
        their cluster
1  for $user \in dataset$ do
2      $cluster = user.cluster$
3      $Score\_list \leftarrow [\,]$
4      for $other\_user \in cluster$ do
5          if $other\_user \neq user$ & $other\_user.GroupMembership \neq empty$ then
6              $overlapping\_groups \leftarrow$
                 $user.GroupMembership \cap other\_user.GroupMembership$
7              $score \leftarrow$
                 $overlapping\_groups.length / user.GroupMembership.length$
8              $Score\_list \leftarrow Score\_list + score$
9          end
10     end
11     $user.score \leftarrow Score\_list.average$
12 end
```

**Compare clustering to the UserPeerAnalytics table** We further compare our clustering to groupings made by Microsoft's UEBA, introduced in Section 2.2. The

*AccountObjectId* field from a user within the `IdentifyInfo` table can be used as *UserId* within the `UserPeerAnalytics` table. This returns an ordered list of up to twenty users who according to UEBA depict the same behaviour. The order is defined via the *Rank* value, which cannot be shared among multiple users at one moment and is a variable between 1 and 20. The information from `UserPeerAnalytics` does not feature distinct clusters: Any user who features in the top twenty of someone's peers can also appear in another person's top peer list. Therefore we apply a different method than for GroupMemerships to quantify how alike the clusters are with the peer groups. This method is described in Algorithm 2. The output of this is a score per user, a value between 0 and 1, on the fraction of overlap between their cluster members and their group of peers. To construct this ranked list of peers Microsoft uses *"the user's Azure AD security group membership, mailing list, et cetera"* [9] as input data for their ML algorithms. Because their steps from data to inferences are black box, we cannot assume it to be the most accurate grouping of people based on behaviour, nor can we tell how their algorithm came to this conclusion. Therefore, we cannot assume it to be the use this as ground truth. We merely try to discover whether we are able to construct the same types of subgroups using solely access data. Our methodology and data used is fully known, hence more explainable than UEBA's approach. If the groups are similar according to our metric, then we have uncovered a transparent method to reach to the same conclusion.

[9]UserPeerAnalytics `https://learn.microsoft.com/en-us/azure/sentinel/identify-threats-with-entity-behavior-analytics#user-peers-metadata---table-and-notebook`

**Algorithm 2:** Method to quantify how similar a user's peer group are to the cluster division.

```
input : The matrix of users dataset, including every user's cluster and up to
        20 ranked peers
output: The score for every user: the percentage of peers in the same
        cluster as the user, relative to all the user's peers
1  for $user \in dataset$ do
       /* Subtract by 1 to remove the user themselves from the number
          of cluster members                                          */
2      $cluster\_size \leftarrow user.cluster.size - 1$
3      $peer\_group \leftarrow [\,]$
4      for $peer \in user.peers$ do
5          if $peer.Rank \leq cluster\_size$ then
6              $peer\_group \leftarrow peer\_group \cup peer$
7          end
8      end
9      $overlap \leftarrow peer\_group \cap user.cluster$
10     $user.score \leftarrow overlaps.length/min(cluster\_size, peer\_group.size)$
11 end
```

## 4.3 Sentinel dataset additional steps taken

Initial results of the Sentinel data produced inconsistent clustering over time and only mediocre similarity to the AD and UEBA groups. This section describes how we attempted to improve the clustering consistency scores and verify the additional value of our methodology.

### 4.3.1 Improving clustering consistency

#### Different input features

Using the feature importance metric, we can filter out features less relevant to the clustering. Additionally, we can mimic the CERT dataset features to achieve the consistent clustering we observed for that dataset.

#### Different standardisation method

In the regular methodology, standardisation is performed by applying the sklearn StandardScaler. To magnify or minimise difference between users' features, other

standardisation methods are applied. The MinMaxScaler[10] scales each feature value back to a given range. This equalises the importance of all individual features, as they all have the same range, limiting the influence of outliers. The relative difference between a feature's value between different users stays the same.
Another method of standardisation is achieved through QuantileTransformers[11]. This method transforms the feature values to follow a uniform or normal distribution. Through this, the relative distance to outliers are reduced which limits their impact. Additionally, values which are close together are spread out through this type of transformation.

### Different clustering method

As the K-means clustering approach simply classifies the users closest to a centroid into one cluster, this could blur the boundaries between clusters and is heavily influenced by the randomised initial position of the centroids. In addition, it does not recognise clusters of different sizes well, whereas we could expect organisations to not have evenly distributed groups of users with similar access behaviour. Therefore, we apply the DBSCAN clustering approach[12] to get an indication whether this could outperform K-means clustering with regards to clustering consistency or comparison to peers and AD groups. DBSCAN clustering recognises densely populated areas in a multidimensional space and groups the users within such areas together in a cluster. If an area is not densely populated, then the users within this area are not put into a cluster at all. A single parameter is required for this function: the *epsilon*. This parameter describes the maximum distance between two points for them to be considered in each others neighbourhood. A higher epsilon value will lead to less dense points being included in the cluster.

### Longer period of time for feature vector creation

Whilst requiring significantly more resources in computing power and time, the data of multiple weeks could be aggregated together to take the data of an entire month into account when creating the feature vectors. This reduces the influence that minor behavioural changes such as a vacation day or tasks which are not executed on a weekly basis could have on the clustering classification. Hence, we would expect the clusters to become more stable over time.

[10]sklearn MinMaxScaler: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`
[11]sklearn QuantileTransformers `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html`
[12]sklearn DBSCAN clustering `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html`

### 4.3.2 Comparing groupings to random clusters

This is the first research to apply such methodology to sole access data. Therefore we do not expect the clusterings resulting from this proof of concept to be perfect. Therefore we need to measure whether there is any information that can be retrieved from the access data with regard to consistent user classification, the AD groups or the peer groups. This shows that there is some signal in the access data with regards to the before-mentioned groups. To measure this, we compare the different scores of our clusters to that of random chance. There are two ways to go about this: either we extract the user amount per cluster and randomly re-distribute all users according to this distribution, or we fully randomise the classification given the number of clusters. The former likely results in clusters of uneven size, the latter in evenly sized clusters.

Chapter 5

# Results

This section contains the results of applying our clustering methodology both to the CERT dataset and within the real-world Sentinel environment. We analyse the extracted clusters and quantify the performance in profiling users based on access behaviour.

## 5.1 CERT Insider Threat dataset analysis

For these results we used the *logon.csv* file as input from version 4.2 of the CERT Insider Threat dataset, combined with the LDAP role information from *2010-05.csv*, and all insider threat scenario files from the *answers* directory.

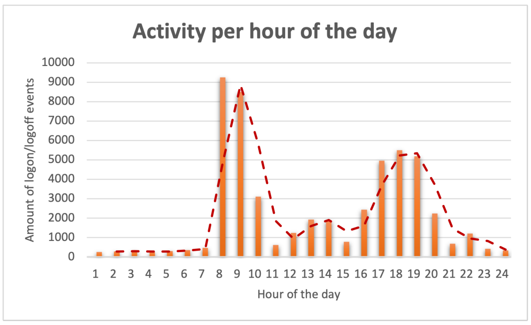
### Variables

Whilst applying the before-mentioned methodology, we used the following filters and parameters before clustering:

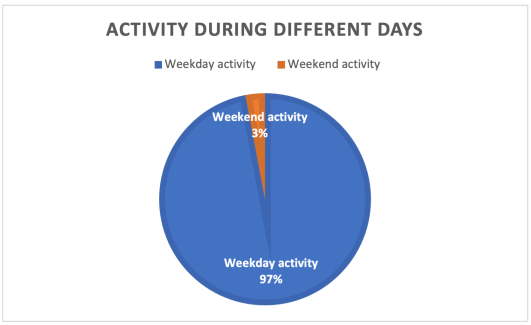
- User with less than three unique PC's accesses were filtered out
- The timeperiod duration was one month
- Only months from 2010 were used, every figure hereafter mentions which exact month
- Office hours are set from 7:00 until 21:00, following Figure 5.1
- Timeframe parameter is set to 24 hours
- Number of clusters: two or five, dependant on the type of analysis

The features used are listed in Appendix A.6, which shows that two features remain unused. The destination hosts feature was not used for clustering because of the resource principles within the dataset. There are only few users accessing more than three resources, which creates very little overlap between users and the few resources they access. Since there is no known relations between the resources, this cannot be used to group users based on the resources they access. The cumulative difference from the mean feature was not used for clustering because of the skew it has dependant on the first day of the dataset. The access amount on the first day of the dataset sets the mean, and it takes a while for this skew to average out and for the mean number of daily accesses to reach an equilibrium. It also did not contribute to identifying the users labelled insiders. This is most likely since the malicious actions in the CERT dataset are not preceded by share discovery or lateral movement. Because of this arbitrariness, and the unexplainability on its influence in generating logical clusters, we did not include this deviation feature in the clustering.

Figure 5.1 shows the logon and logoff activity during different hours and days of the week. This looks as expected: Office hours from 7:00 to roughly 21:00 and 33x as much activity during weekdays than in the weekend, or 13x as much on a daily basis.



*(a)* Activity per hour of the day



*(b)* Activity for different parts of the week

**Figure 5.1:** Characteristics of the CERT v4.2 synthetic dataset for the period 2010-05-01 until 2010-06-01.

## Number of clusters

Applying the Elbow method for selecting the optimal amount of clusters results in similar plot for nearly every month in 2010 of the CERT dataset. The elbow plot for the period 09-01 until 10-01 is shown in Figure 5.2 and results in an optimal K of 5. This is also the average number of optimal clusters amongst every month of 2010.

Figure 5.3 shows the clusters for K=5, but there is also a clear division between the positive and negative groups on the PC_1 axis. We took a closer look at this phenomenon by displaying the user roles for every cluster. Apart from ITAdmins, the roles within the clusters do not seem correlated to which cluster the user is classified as. We verify this by applying the adjusted Rand index to compare the cluster classification with the role of every user, which for this month of data resulted in a value of just **0.25**, supporting our observation.

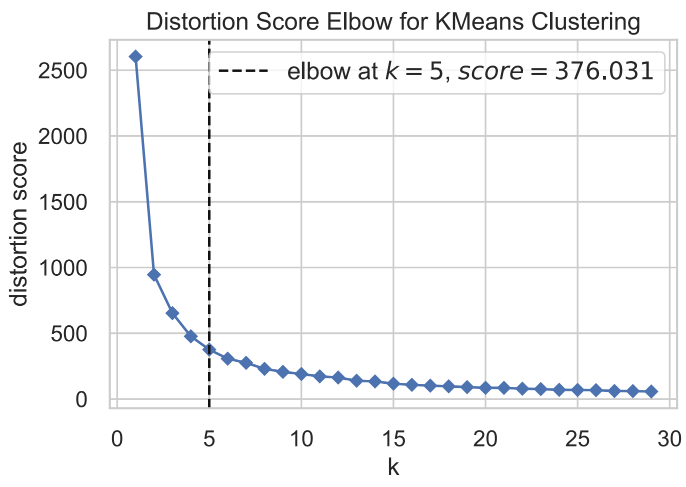
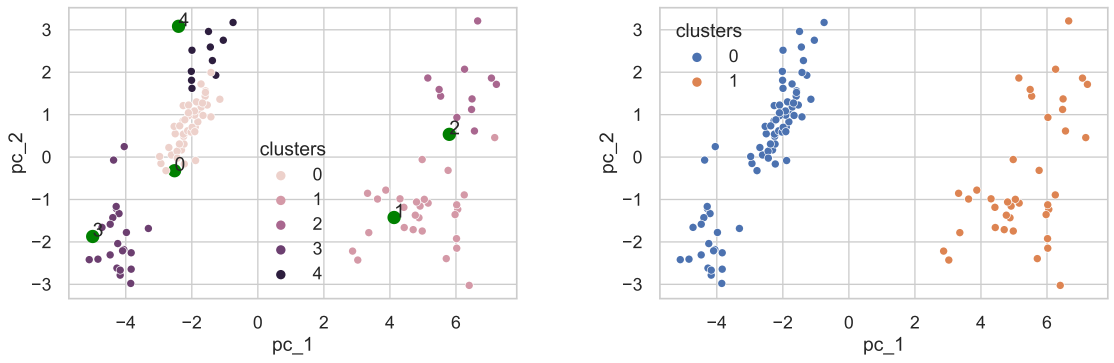


**Figure 5.2:** The elbow plot for the CERT dataset of the month 2010-09, using the features listed in Appendix A.6.



*(a)* The clustering for K=5.

*(b)* The clustering for K=2.

**Figure 5.3:** The clustering for the period 2010-04-01 until 2020-05-01 from the CERT dataset. The green dots show the centroids of each label. Since the actual plot is multidimensional, displaying it in 2D may seem off at a few borderline cases.

Applying the same methodology using K=2 to other months resulted in the same outcome: ITAdmins and non-ITAdmins are consistently classified in a separate cluster. There are some exceptions, which are highlighted at the end of this section. The scree plot returns that all the variance can be shown using solely PC (Principal Component) 1, which is also visible in Figure 5.3, as the PC_1 axis alone separates the two user groups. PC 1 is constructed through a combination of almost all features, the division of which is shown in Figure 5.4.

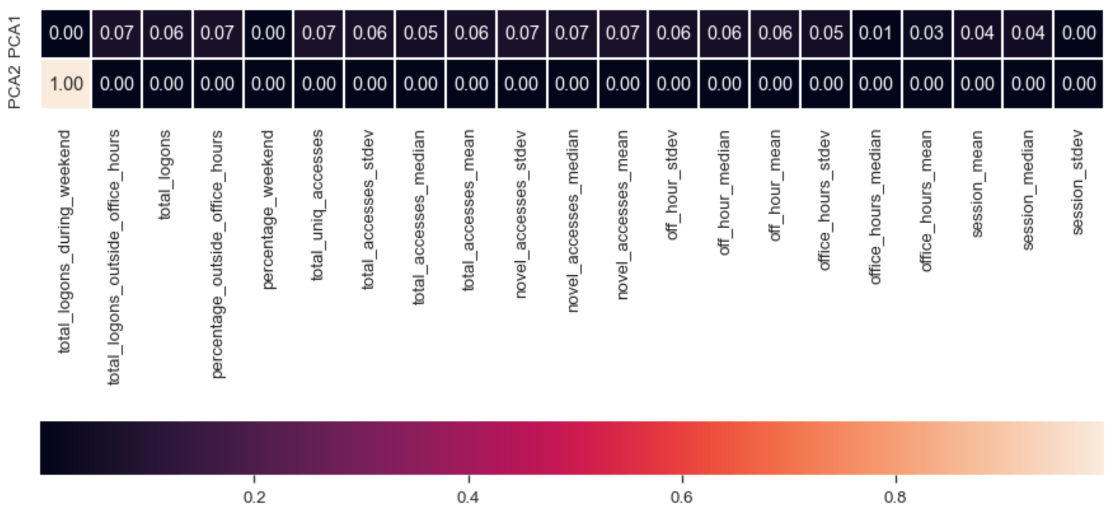


**Figure 5.4:** Heatmap on how much every feature is used to display the two Principal Components for the CERT dataset period 2010-04-01 until 2020-05-01.

**Feature importances** We can look at what features caused the ITAdmins to be classified in a different cluster than all other roles via the Feature Importance metrics. There are just two clusters, so the feature importances for cluster 0 are identical to the feature importances for cluster 1, since the most influential feature causing a user to be classified into cluster 0 is also the most influental feature causing a user to not be classified into cluster 1. Figure 5.5 shows that the most influential features are the average number of daily accesses, the median session length and the total number of logons. The features differentiating the least between admins and other users are the weekends logon, the percentage thereof, and the number of office hour accesses.

Now that we know the feature importances we can look at the feature statistics of the clusters, compare it with feature statistics on roles, and verify whether the clustering is explainable. Figure 5.6 shows for the top three most influential features what the average, median and standard deviation is amongst the users of the two different clusters. The same statistics about every feature is provided per *role* in Appendix A.1, due to the size of these figures. For these three features the mean

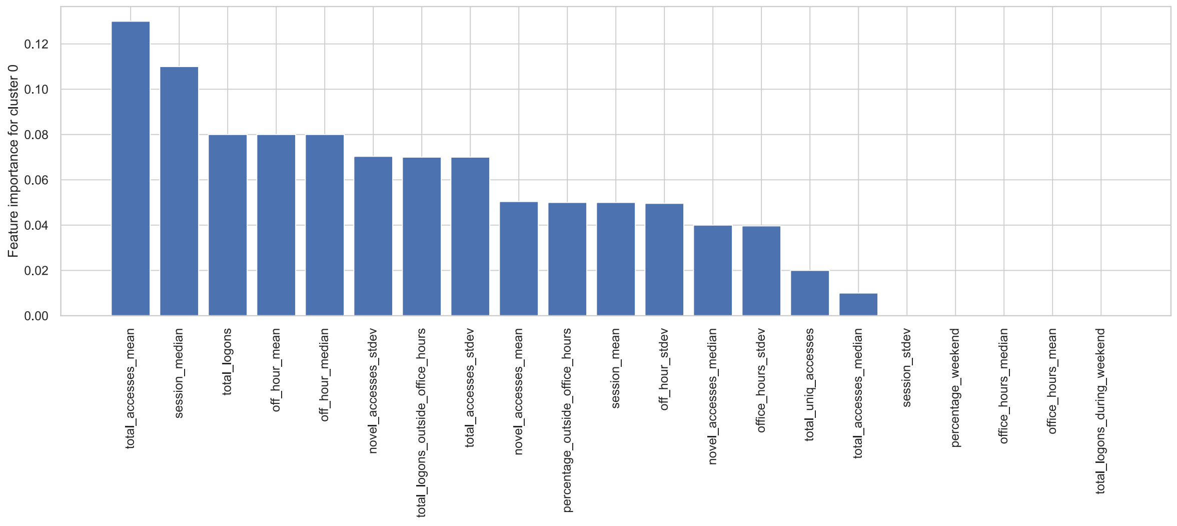
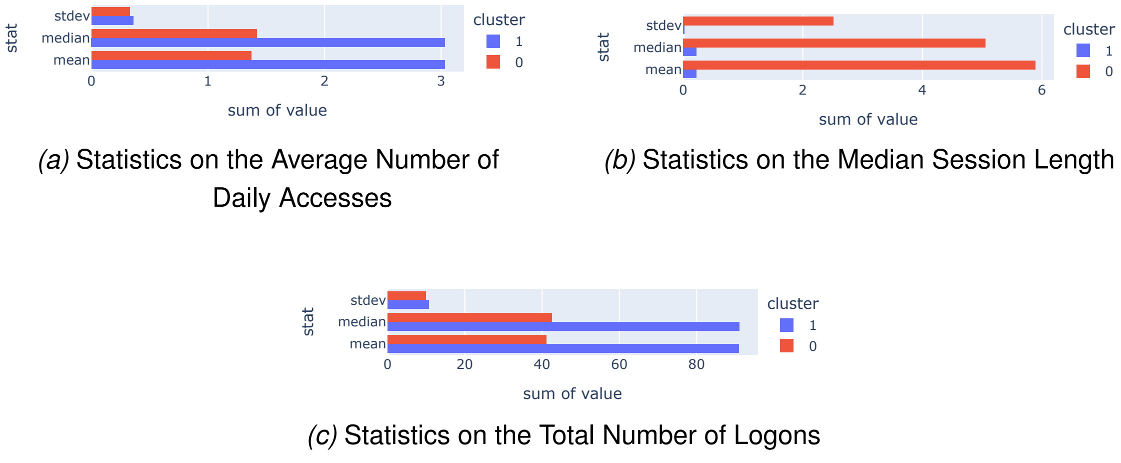


**Figure 5.5:** Feature importances for cluster 0 for the period 2010-04-01 until 2010-05-01. This plot is identical to the feature importances of cluster 1.

and median statistics per cluster differ significantly, and users with the ITAdmin role indeed have higher Total Daily Access Average and Total Logons values, and lower Median Session Lengths. This shows that the features listed as most differentiating are similar for the ITAdmin role and Cluster 1, and differ significantly from other roles.



*(a)* Statistics on the Average Number of Daily Accesses

*(b)* Statistics on the Median Session Length

*(c)* Statistics on the Total Number of Logons

**Figure 5.6:** The mean, median and standard deviation of the top three features amongst the users of the listed cluster.

**Insider analysis** When applying the clustering methodology to different months, occasionally an ITAdmin got placed in the non-ITAdmin cluster. Comparing these to the known periods that Insider Threats were active revealed that if an ITAdmin was an insider in a certain month, they were likely to be placed in the non-ITAdmin cluster. For every month of 2010, we therefore verified whether a user was an ITAdmin, an insider and/or classified into the ITAdmin cluster. Table 5.1 shows that 8 out of 10 ITAdmin insiders get clustered differently than their role, two do not get 'detected' and none of the insiders with a role besides ITAdmin were possible to spot through this method. This is likely caused by insider ITAdmins performing little activity after their malicious actions, causing them to access less resources than a regular ITAdmin. This phenomenon is further discussed in the Discussion chapter.

| | **ITAdmin** | **Other role** |
|---|---|---|
| **Classified differently during insider period (TP)** | 8 | 0 |
| **Classified differently whilst not an insider (FP)** | 2 | 0 |
| **Not classified differently during insider period (FN)** | 2 | 7 |
| **Not classified differently whilst not an insider (TN)** | 35 | 158 |

**Table 5.1:** The detection rate of insider ITAdmins and other roles, for every period of the year 2010 out of the CERT Insider Threat dataset.

## Five clusters

We continue with the analysis for K=5, the optimal number of clusters according to the Elbow heuristic, such as Figure 5.3. We computed the feature importance scores, cumulative for all clusters, per month. This cannot be used to statistically prove the importance of one feature above the other, as it is not a weighted average for clusters of different sized. It can, however, give an indication of which features are most influential, and did correspond to the feature importance metrics of most clusters. The top three for every month are displayed in Table 5.2. This shows that for K=5 the most influential features are the average daily accesses, the total number of logons and the standard deviation of accesses during office hours.

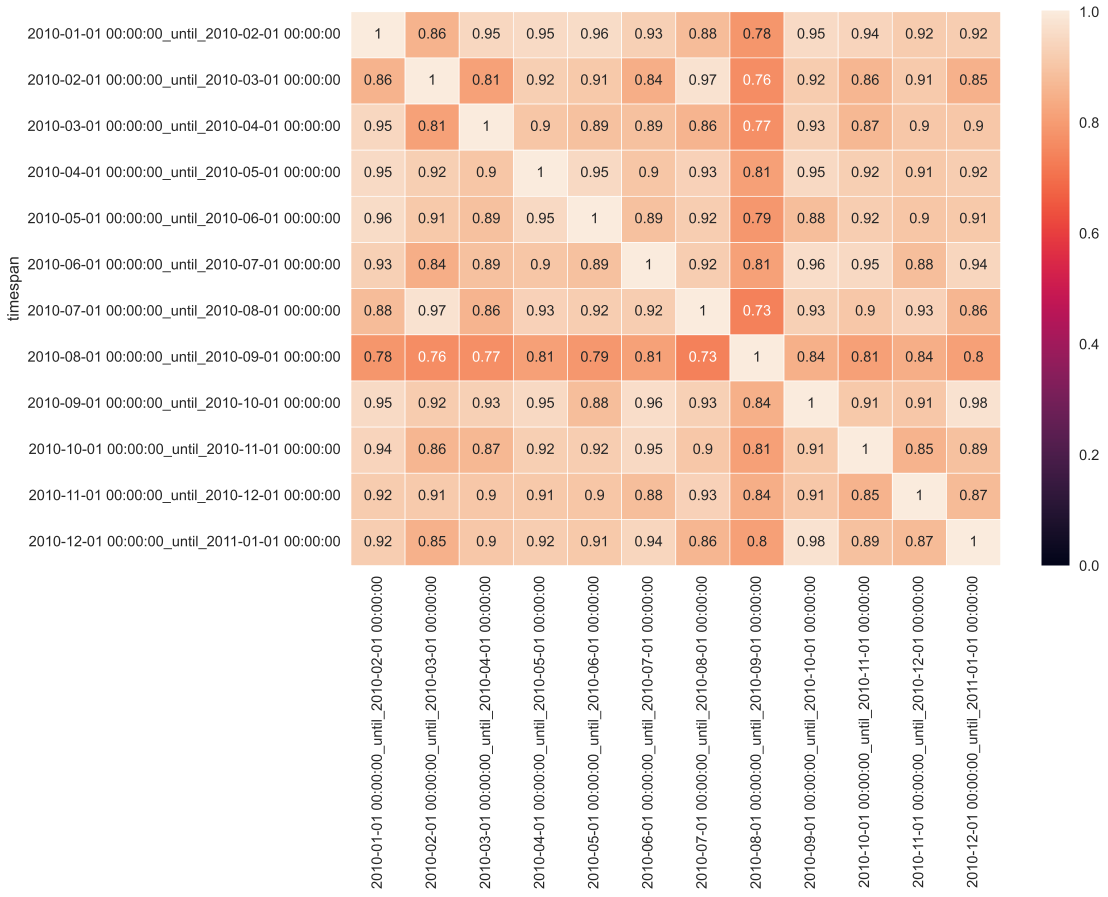
**User role groups** Apart from the possibility to separate the group of ITAdmins from non-ITAdmins, no significant intersection between the role groups and cluster groups were found through applying the adjusted Rand index to the different user groups, nor by manual inspection.

**Quantifying cluster stability over time** One of the possible use cases for a SOC through this type of profiling is to monitor for users with sudden deviation in behaviour. This could be detected by a user changing cluster. Therefore it is important to have a stable clustering for benign users. We measure how consistent its profiling of users is, by comparing the cluster division of every month of 2010 pair-wise via

| | #1 feature | #2 feature | #3 feature |
|---|---|---|---|
| **Jan** | total_accesses_mean | total_logons | total_accesses_stdev |
| **Feb** | office_hours_stdev | office_hours_mean | total_accesses_mean |
| **Mar** | total_accesses_mean | total_logons | total_accesses_stdev |
| **April** | total_accesses_mean | office_hours_mean | office_hours_stdev |
| **May** | total_accesses_mean | total_logons | total_accesses_stdev |
| **June** | total_accesses_mean | total_logons | office_hours_stdev |
| **July** | total_logons | total_accesses_mean | office_hours_stdev |
| **Aug** | total_accesses_mean | total_logons | off_hour_stdev |
| **Sept** | office_hours_stdev | total_accesses_mean | total_logons |
| **Oct** | office_hours_mean | office_hours_stdev | total_accesses_mean |
| **Nov** | total_accesses_mean | total_logons | office_hours_stdev |
| **Dec** | total_accesses_mean | total_logons | office_hours_stdev |

**Table 5.2:** The top three most influential features per period within 2010, computed by summing the importance metric per feature for every cluster.

the Rand index adjusted for chance. For this comparison, all malicious actors which were active during 2010 were excluded from this dataset, as they are expected to have changing behaviour. The similarity scores of every comparison is displayed in Figure 5.7. All values are in the 0.73-0.96 range, with the average value of months following each other (Jan - Feb, Feb - March etc) at 0.87.



| timespan | 2010-01-01 00:00:00_until_2010-02-01 00:00:00 | 2010-02-01 00:00:00_until_2010-03-01 00:00:00 | 2010-03-01 00:00:00_until_2010-04-01 00:00:00 | 2010-04-01 00:00:00_until_2010-05-01 00:00:00 | 2010-05-01 00:00:00_until_2010-06-01 00:00:00 | 2010-06-01 00:00:00_until_2010-07-01 00:00:00 | 2010-07-01 00:00:00_until_2010-08-01 00:00:00 | 2010-08-01 00:00:00_until_2010-09-01 00:00:00 | 2010-09-01 00:00:00_until_2010-10-01 00:00:00 | 2010-10-01 00:00:00_until_2010-11-01 00:00:00 | 2010-11-01 00:00:00_until_2010-12-01 00:00:00 | 2010-12-01 00:00:00_until_2011-01-01 00:00:00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2010-01-01 00:00:00_until_2010-02-01 00:00:00 | 1 | 0.86 | 0.95 | 0.95 | 0.96 | 0.93 | 0.88 | 0.78 | 0.95 | 0.94 | 0.92 | 0.92 |
| 2010-02-01 00:00:00_until_2010-03-01 00:00:00 | 0.86 | 1 | 0.81 | 0.92 | 0.91 | 0.84 | 0.97 | 0.76 | 0.92 | 0.86 | 0.91 | 0.85 |
| 2010-03-01 00:00:00_until_2010-04-01 00:00:00 | 0.95 | 0.81 | 1 | 0.9 | 0.89 | 0.89 | 0.86 | 0.77 | 0.93 | 0.87 | 0.9 | 0.9 |
| 2010-04-01 00:00:00_until_2010-05-01 00:00:00 | 0.95 | 0.92 | 0.9 | 1 | 0.95 | 0.9 | 0.93 | 0.81 | 0.95 | 0.92 | 0.91 | 0.92 |
| 2010-05-01 00:00:00_until_2010-06-01 00:00:00 | 0.96 | 0.91 | 0.89 | 0.95 | 1 | 0.89 | 0.92 | 0.79 | 0.88 | 0.92 | 0.9 | 0.91 |
| 2010-06-01 00:00:00_until_2010-07-01 00:00:00 | 0.93 | 0.84 | 0.89 | 0.9 | 0.89 | 1 | 0.92 | 0.81 | 0.96 | 0.95 | 0.88 | 0.94 |
| 2010-07-01 00:00:00_until_2010-08-01 00:00:00 | 0.88 | 0.97 | 0.86 | 0.93 | 0.92 | 0.92 | 1 | 0.73 | 0.93 | 0.9 | 0.93 | 0.86 |
| 2010-08-01 00:00:00_until_2010-09-01 00:00:00 | 0.78 | 0.76 | 0.77 | 0.81 | 0.79 | 0.81 | 0.73 | 1 | 0.84 | 0.81 | 0.84 | 0.8 |
| 2010-09-01 00:00:00_until_2010-10-01 00:00:00 | 0.95 | 0.92 | 0.93 | 0.95 | 0.88 | 0.96 | 0.93 | 0.84 | 1 | 0.91 | 0.91 | 0.98 |
| 2010-10-01 00:00:00_until_2010-11-01 00:00:00 | 0.94 | 0.86 | 0.87 | 0.92 | 0.92 | 0.95 | 0.9 | 0.81 | 0.91 | 1 | 0.85 | 0.89 |
| 2010-11-01 00:00:00_until_2010-12-01 00:00:00 | 0.92 | 0.91 | 0.9 | 0.91 | 0.9 | 0.88 | 0.93 | 0.84 | 0.91 | 0.85 | 1 | 0.87 |
| 2010-12-01 00:00:00_until_2011-01-01 00:00:00 | 0.92 | 0.85 | 0.9 | 0.92 | 0.91 | 0.94 | 0.86 | 0.8 | 0.98 | 0.89 | 0.87 | 1 |

**Figure 5.7:** Cluster comparison heatmap of every period in 2010 for the CERT dataset, without taking into account malicious insiders and using the adjusted Rand function.

## 5.2 Sentinel

This part of the research was conducted in the Microsoft Azure Machine Learning Studio[1], a cloud Jupyter notebook. We perform this analysis in the Sentinel environment using Northwave SOC customer data. *Note: Because this concerns privacy-sensitive data, any information on user accounts and infrastructure has been anonymised.*

### Preprocessing steps

Within Sentinel Notebook, the results of executing a query are saved in a dataframe. This process has a limit of 500.000 rows, which poses a problem when processing data for larger organisations as they generate too many logon- and logoff Windows events. This is because one user action is not necessarily correlated with one logon event. Under the hood Windows can keep connecting and disconnecting to for example a remote file server, causing a lot of events. This can lead to over 500.000 events on a single day alone. Therefore we reduce the granularity of the access times, by summarizing all accesses with the same properties into one event per hour[2]. This reduces the data more than tenfold. Still, with this reduce of granularity the largest timespan which can be queried at once is a week, instead of a month like for the CERT dataset.

In addition to removing accounts based on SID and account name, we attempted to cluster the dataset and removed outlier users which were placed in a cluster on their own. The purpose of this is to improve the clustering granularity for the rest of the user-base. These accounts are consistently outliers amongst different weeks. It could not always be infered by the account name or description that they were automated, so in reality would be worth an investigation from an analyst to identify why their access behaviour differs so significantly from the rest. More often than not, such outliers were accounts with administrator privileges as can be seen in Figure 5.8, or were developer accounts. These were also commonly put in the same cluster in the iterative clustering phases, as seen in Appendix A.3. Whilst most users with administrative rights were grouped together in one or two clusters, there were also a lot of non-administrative users part of these clusters. In addition, there are different degrees of administrative rights within a domain, which we were unable to identify. Therefore we could not make further inferences on this.

[1]Azure Machine Learning environment: `https://azure.microsoft.com/en-us/products/machine-learning/`

[2]KQL summarize operator: `https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/summarizeoperator`

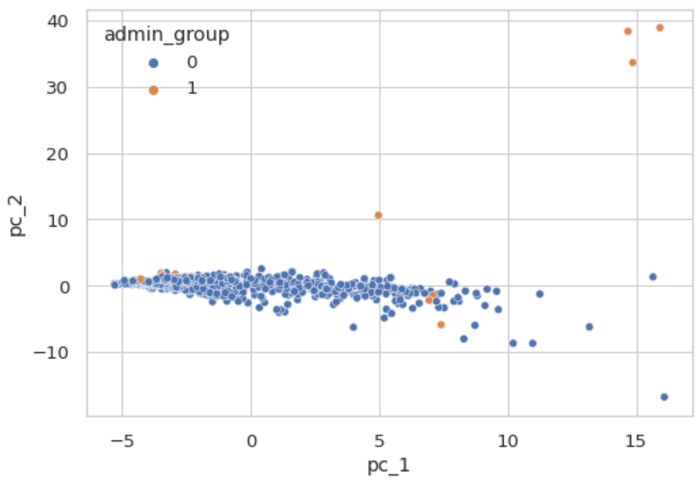


**Figure 5.8:** One of the pre-final clustering phases within the Sentinel dataset, showing how some administrative users were outliers.

## Dataset properties

To avoid the potential data skew of one organisation, all tests were performed at a few different organisations. We observed no significant difference in results between organisations. This hints towards similar data across organisations after the feature creation phase. This section, therefore, displays the results of one organisation, but the conclusions are drawn upon the aggregated results.

## Clustering properties

The properties used for the majority of the clustering are listed below.

- Service- and automated accounts were removed before the standardisation step
- The timeperiod duration was set to one week
- Timeframe parameter is set to 24 hours
- The working hours are between 4 and 17 (GMT), as based on Figure 5.9. We see a trend in logon and logoff activity similar to the CERT dataset with peaks at the start and end of office hours.
- Number of clusters: Six, as per the elbow plot of Figure 5.10. The elbow was consistently around this value for the different weeks.

The table in Appendix A.7 shows for all available features whether they were used. The *session duration* parameter cannot be used in this scenario, as logon and logoff events are mostly attributed to the same timestamp. This is due to the previously explained background logons performed by Windows. Looking at the feature vector importances, all different logontypes and source IP features were the

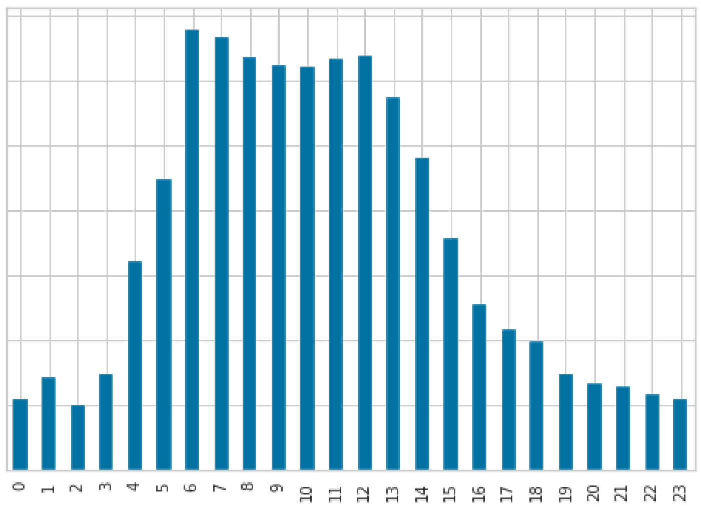


**Figure 5.9:** Logon and logoff activity per hour, in GMT, for one of the organisations.

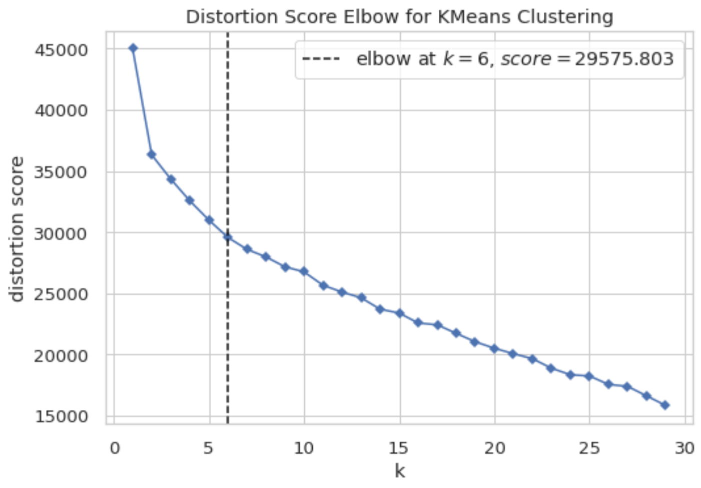


**Figure 5.10:** Elbow plot to compute the optimal k-value, applied to the production data of one week.

least influential by default. We observed that the logontype 3 occurred most often, which are the background logons. Logontypes 8 and 9, NetworkClearText and NewCredentials were only performed by accounts which were later deemed to be service accounts. For source IP we tried different methods of including IP addresses as a feature, through solely using the first 8, 16 or all bits. The latter choice was made as there were too many unique IP addresses, and through this we attempted to profile subnets. However, likely due to the way public IP addresses are distributed, and the difference between private (e.g. 10.0.0.0/8) and public address space, these features were also not deemed relevant by the feature importance heuristic and did not contribute to more consistent clustering. On average, the values distribution of the 'cumulative deviation from the running average number of accesses per timeframe' feature looked like Figure 5.11. We observed that users who were placed far outside the other clusters were commonly also the ones with extreme cumulative difference from mean values. For Sentinel data, this feature experienced an even larger skew than the CERT dataset, likely because we could only incorporate a week's worth of data in the feature. Therefore, this feature was also not used for clustering production data. Further inferences and potential use cases for this feature are further discussed in Section 6.2.

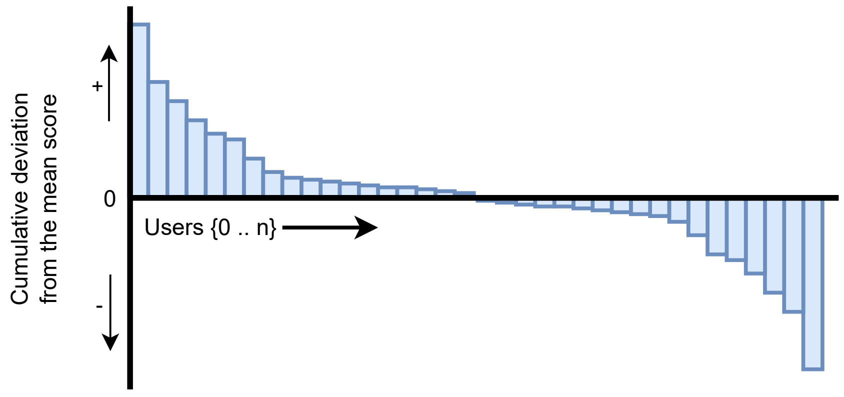


**Figure 5.11:** A plot showing the outline of how the 'cumulative deviation from the mean' score was distributed, over the entire userbase, on the last day of the selected time period, for most of the tested datasets.

## Clustering and feature vector results

The results of the clustering using a week worth of data is shown in Figure 5.12, of which the scree plot is shown in Figure 5.13. The first two principal components, the X- and Y-axis of the plot, were mainly constructed through the destination host features. For the smaller clusters, a certain destination host was usually the most important feature. The logontype was no feature of significance to any of the clusters. Most clusters used solely type 3 (Network) and type 10 (Remote Desktop Protocol). Any accounts with logontypes 8 or 9 (NetworkCleartext and NewCredentials) are usually automated hence were not taken into account for the clustering.

The most relevant feature overall was the total amount of logons, followed by the novel accesses mean, a few destination hosts and off-hour activity.

## Quantifying performance

To quantify the performance of the clustering, we measured the clustering stability over time, compared it with Microsoft's UEBA UserPeerAnalytics table and with the

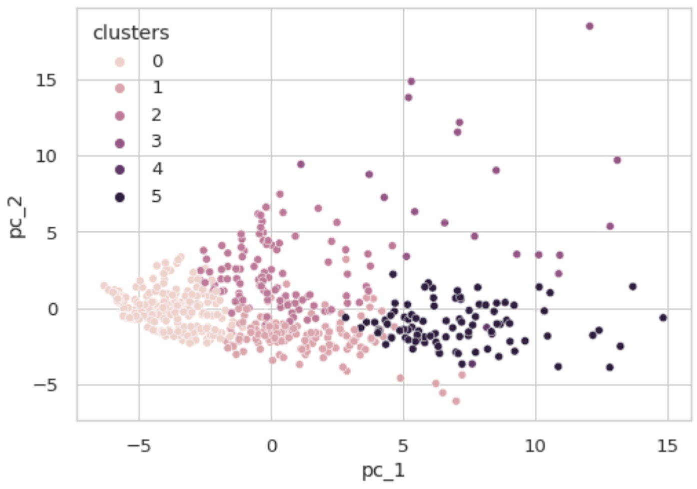


**Figure 5.12:** The cluster plot for production data of one week, using the features listed in Appendix A.7.

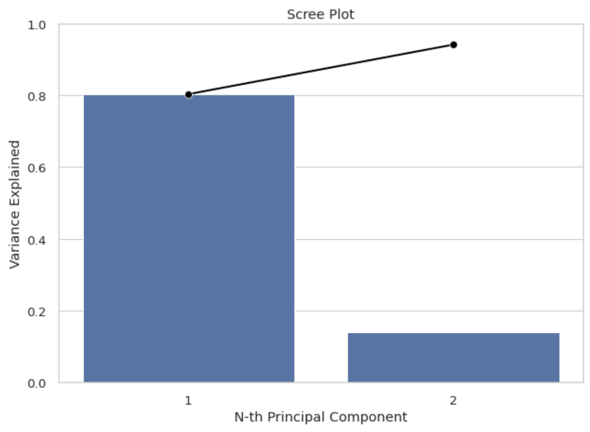


**Figure 5.13:** The scree plot corresponding to Figure 5.12.

on-premise Active Directory groups. Additionally, we performed a qualitative analysis on the correlation between our clustering and administrative users.

**Clustering stability over time** To compute the stability for the formed clusters over time, the clustering division of four weeks was compared via the adjusted Rand index. The results are shown in Figure 5.14, and for another organisation in Appendix A.4. Both ranged between 0.3 and 0.4 in similarity score, with an outlier to 0.49.

**Comparison with AD Group Memberships** The comparison with AD group memberships was done in twofold. First, the quantitative analysis was performed by applying Algorithm 1. This resulted in the value of 0.45 averaged amongst all users. We plotted every user's average comparison score against their amount of group

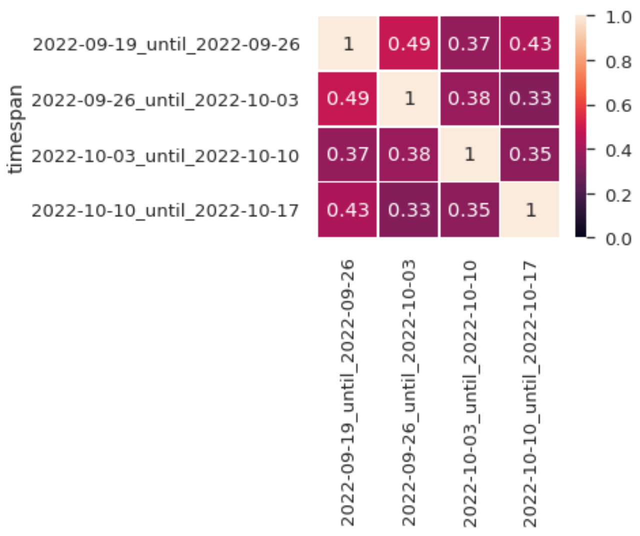


**Figure 5.14:** A heatmap showing the scores of comparing the cluster similarity amongst different weeks of Sentinel data.

memberships in Figure 5.15. On average, a user having more groups memberships is correlated with a lower comparison score.

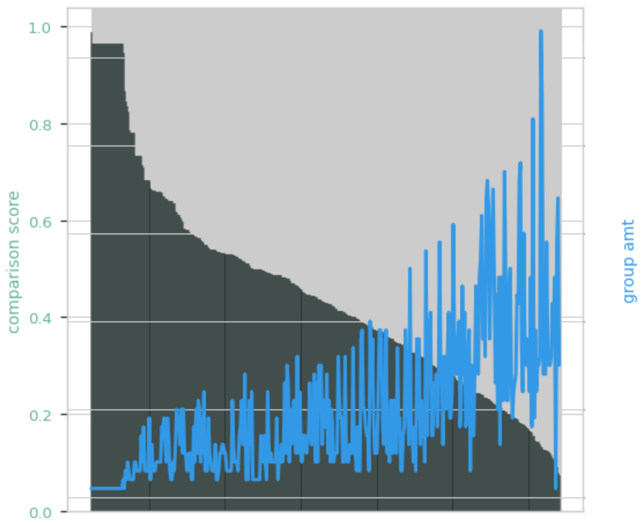


**Figure 5.15:** The comparison score according to Algorithm 1 for every user, plotted in dark green. Per user, their relative number of AD group memberships (referred to as ’group amt’) is shown as the blue line.

## Qualitative analysis

For the qualitative analysis we analysed which clusters all the AD groups were made up of. This organisation contained multiple AD groups with administrator rights. We observed that these groups were predominantly populated by two clusters. We identified users which were part of admin groups but not member of a different cluster than the two predominant ones. Further inspection revealed that these user accounts had different properties than most other admin privileges accounts, such as a different department. Figure 5.16 shows a clustering plot where users with admin privileges are highlighted, which shows that apart from the outliers from Figure 5.8 the admin group users are not grouped together. Three features separated the two clusters with most admin privileged users to the rest, such as the mean number of accesses in Figure 5.17. Other features are shown in Appendix A.2. Based on this analysis, we could not find a usable correlation between the users in AD admin groups and the clusters they were placed in.

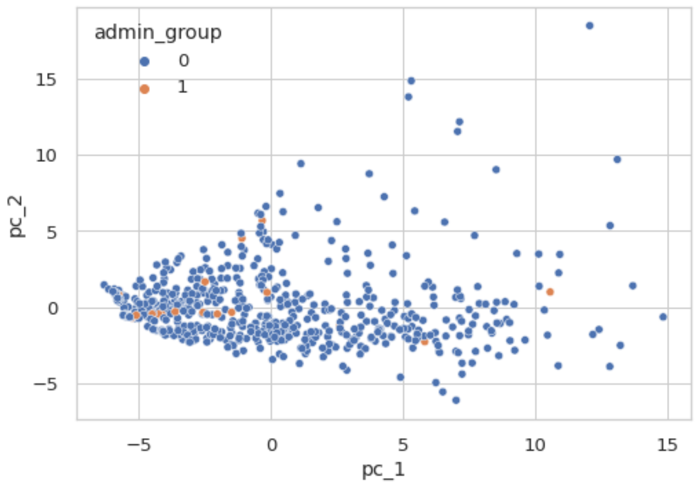


**Figure 5.16:** The same clustering plot as Figure 5.12, with users with administrative rights highlighted in orange.

**Comparison with UserPeerAnalytics** Finally, we compared the formed clusters with the UEBA UserPeerAnalytics table following Algorithm 2. For this comparison, the user identifiers for the on-premise digital infrastructure had to be correlated with the users' Azure cloud identifiers. This resulted in about 1/4th of the total number of users being excluded from this analysis, Additionally, not all listed peers were actually clustered, so these were also excluded from the comparison as they did not have the opportunity to be part of any cluster. This was the case for 1/5th of all leftover peers. The resulting score distribution is shown, in descending order, in Figure 5.18. The value averaged over all users is 0.36.

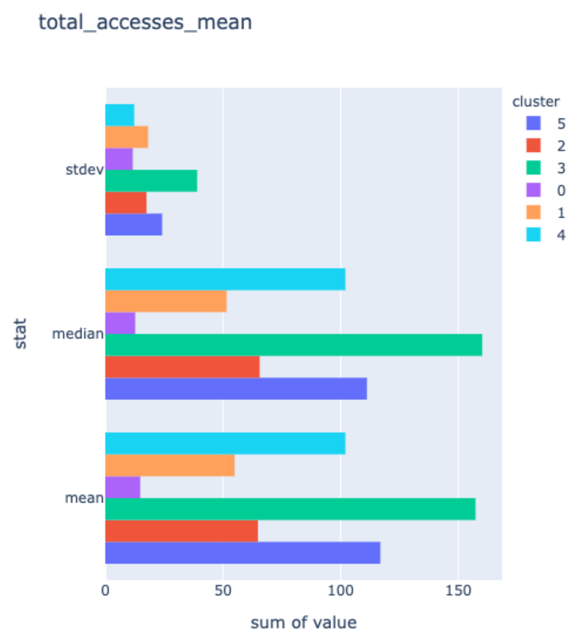


**Figure 5.17:** The properties of the 'total_accesses_mean' feature, showing relative low values for clusters 0 and 2, which were the two clusters with most admin users.

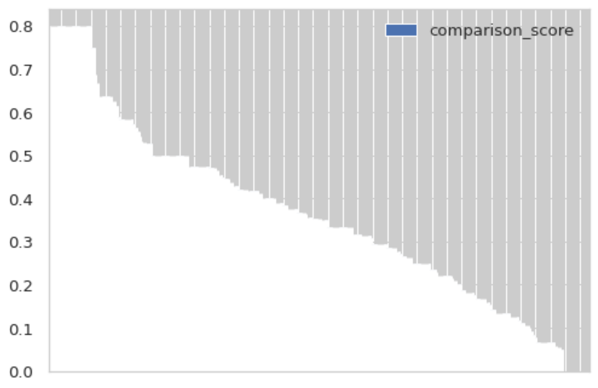


**Figure 5.18:** A UserPeerAnalytics comparison score bar chart, where every bar on the X-axis represents one user's score. The chart is ordered from highest to lowest score, with the best score being 0.8 and the lowest 0.0.

## 5.3 Clustering consistency improvement efforts

This section lists the different efforts put into increasing the clustering consistency and similarity to UserPeerAnalytics and AD groups.

In an attempt to improve the stability of the clusters, we aggregated data of multiple weeks, circumvening the earlier query limit. This lead to a slight improvement in clustering stability score, as seen in Figure 5.19, but is still far from a stable clustering. Additionally, the comparison to AD groups score increased to an average of 0.47 amongst users, and the average UserPeer score increased to 0.30.

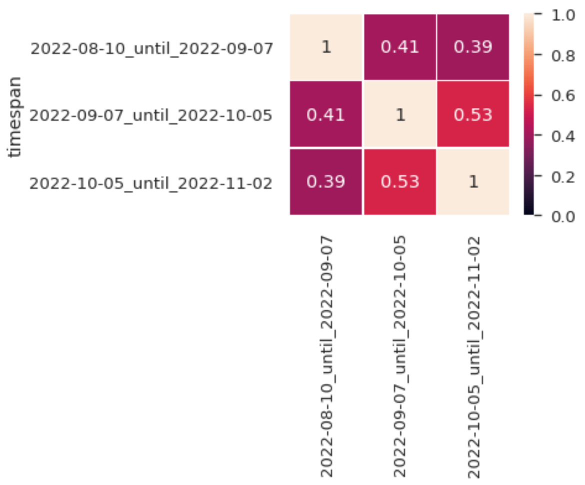
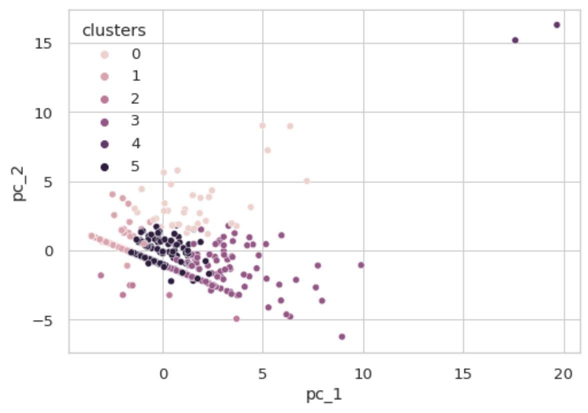
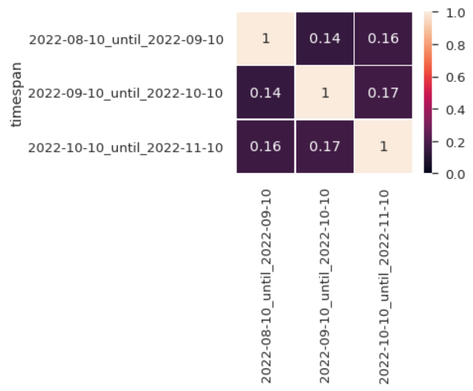


**Figure 5.19:** Cluster similarity score in production data through comparing different months.

We tested a variety of input feature compositions, such as including matching the CERT dataset features, or only incorporating destination hosts as features, but these only decreased the consistency scores. Of the different logontypes, the network logon was observed most. This is because of the way Windows operates in the background when connecting to another device, as explained in Section 4.2. Therefore, not every access logged correlates to a user interaction. To try and more closely match the CERT dataset's structure, we filtered on the LogonType 10: RDP (Remote Desktop). With this type of logon, one user action does correlate to a single logon event. In addition, there are significantly fewer events hence we could incorporate the data of an entire month and include the session duration feature. RDP is also often used by threat actors to use a compromised account for lateral movement. Therefore this behaviour could visibly change compared to a user's regular actions. Unfortunately, filtering on RDP only decreased the cluster stability when comparing consecutive months, as shown in Figure 5.20. One of the causes that this could have, is that whilst RDP access events to represent user interaction better, it is used in a different manner than any of the other logons. RDP connections are much less common, so it is hard to create a baseline for a user's behaviour describing events outside someone's daily or weekly routine.



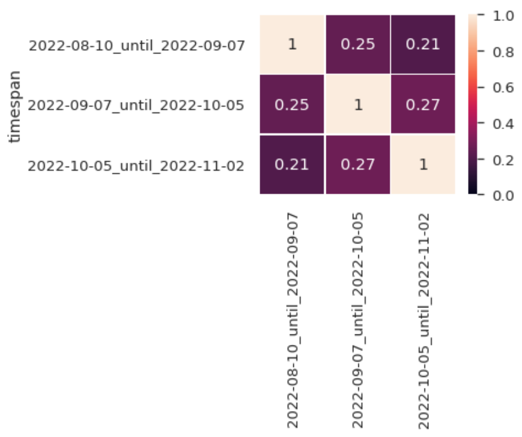
*(a)* The clustering based on a month of data.



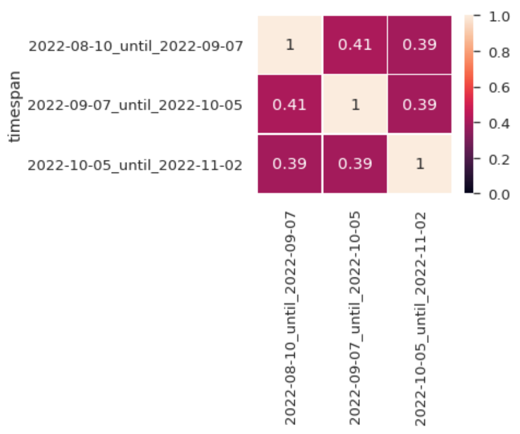
*(b)* The cluster similarity scores of three separate months.

**Figure 5.20:** The clustering properties of the Sentinel dataset, filtered on just RDP accesses.

Figure 5.21 shows the consistency results when applying different feature vector standardisation methods, to either minimise of enlarge the differences between users' feature values. This was performed on the aggregated data of multiple weeks, so can directly be compared to Figure 5.19. Both the QuantileTransformer and the MinMax standardisation returned lesser scores than the StandardScaler.



*(a)* Standardisation method: MinMaxScaler



*(b)* Standardisation method: QuantileTransformer

**Figure 5.21:** The cluster similarity scores amongst three separate months, using the same features as listed in Appendix A.7, but a different standardisation method.

We also applied the DBSCAN clustering algorithm as replacement for K-means, to verify whether the way K-means computes clusters forms a bottleneck to improving the stability and similarity to AD/UEBA. DBSCAN is better at identifying the existence of groups within a multidimensional space, and only clusters points if they are regarded part of such a group. Using an epsilon parameter value of 0.3, figure 5.22 shows that very few users actually get clustered, meaning 80% of users get regarded as 'outlier'. Increasing the epsilon parameter meant that more users got included in the clustering, but the number of clusters also decreased. This means that with an epsilon value high enough to include the majority of users, all those users would be classified into the same cluster.

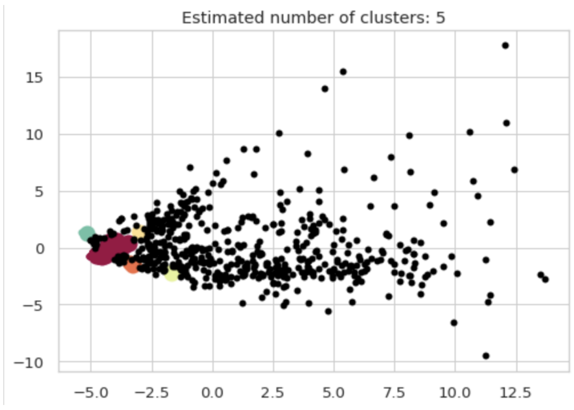
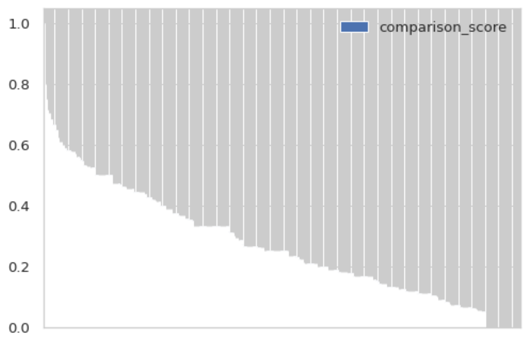


**Figure 5.22:** The cluster plot for a month of Sentinel data when using the DBSCAN clustering method with an epsilon value of 0.3. The coloured dots represent groups of clustered users. The black dots are users which are not clustered.

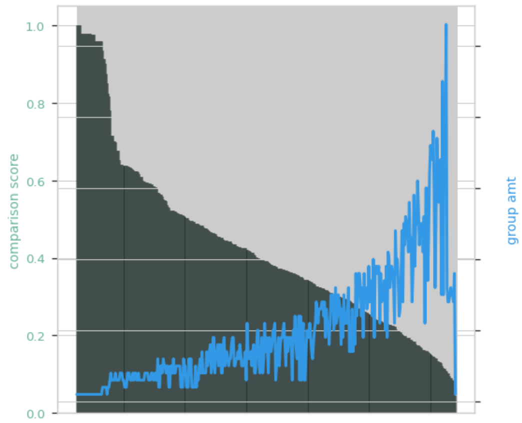
## 5.4 Comparing clusters to random groupings

Figure 5.23 shows the UserPeerAnalytics comparison score bar chart, where the users were randomly classified into clusters. Every of the six clusters contained the same number of users as with the clustering used for Figure 5.18. The average score of the comparison of UserPeerAnalytics and randomly classified users was 0.27, which is lower than the for that of Figure 5.18. We also performed this methodology but by randomly distributing users over any cluster, resulting in evenly divided clusters, but this yielded similar or slightly lower scoring results.

For the AD groups, Figure 5.23 shows the similarity scores comparing them to the clusters, with the size of every cluster based on previous runs, such as for Figure 5.15. The average score for this figure is 0.43. We see that as on average, the more AD groups a user is a member of, the lower their comparison score, more strongly than in Figure 5.15. This makes sense for a randomised cluster allocation, since a user with many AD groups has a lesser chance for them all to overlap with the other users in their cluster.



*(a)* The UserPeerAnalytics score for randomly classified users.



*(b)* The AD group comparison score for randomly classified users.

**Figure 5.23:** Bar charts with users on the X-axis displaying scores for the two comparison metrics. The users are randomly distributed over the clusters using the same user amount per cluster and user data as used in Figures 5.18 and 5.15, hence can be directly compared to those plots.

Chapter 6

# Discussion and Recommended Research Directions

This section discusses the outcomes of the previous section on our research questions. We evaluate shortcomings within the current methodology and suggest how this can be improved in the future.

## 6.1 Synthetic Insider Threat dataset clustering

### Profiling consistency over time

The CERT dataset features very few shared resources, so counting per resource how many times it is accessed could not be used as a feature to group users. Therefore most features were constructed using temporal analysis, such as whether a login was within office hours and the session duration between logon and logoff. Other than the ITAdmin role, we could not find any correlation between user role and cluster through this.
The clusters over time for benign users are very consistent, although using 'user switch cluster over time' as suspicious indicator would still return too many false positives to be usable by a SOC. This approach is therefore promising, but would require further improvement prior to production usage.
The usage of the 'cumulative difference from the mean number of daily accesses' feature did not assist in identifying malicious actors for the CERT dataset, as their actions were not preceded by network discovering or lateral moving actions which involved contacting significantly more machines than before.

### ITAdmins versus other users

Our clustering approach can separate users with the ITAdmin role from regular users with ease. The main defining features for this were the average daily accesses and session length, as Appendix A.1 reveals that ITAdmins perform more and shorter sessions. This is an inference that we could not have gotten by reading the dataset description, which only decribes ITAdmins having global access privileges.
As shown in Table 5.1, our clustering approach works well in detecting admin insiders as they switch clusters in the period of their insider actions.
Investigation of the logon actions of ITAdmin users revealed that they perform no further accesses soon after their last insider actions are performed. This decreases the ITAdmin's number of accesses for the month they performed malicious actions in, which skews their behaviour from an ITAdmin user to a regular user. Within the scenario where the CERT Insider Threat dataset is created this implies that after someone has performed their malicious actions they leave the company straight away. This type of detection limits itself to a certain type of insider, and could also trigger when a user is less productive than regular. The latter is not a security risk and quite invasive, and would be regarded as a False Positive for our algorithm. Therefore, in this case, our algorithm does not detect a suspicious change in behaviour through a user accessing *more* resources than usual, but by a user accessing *less* resources than usual. This has overlap with the research topic of Quitting Detection. Quitting detection is a subject within Insider Threat detection research which focuses on predicting that a user will turn against their employer. This is used in the insider threat detection research by Gavai et al. [24], who use quitting detection as a proxy for insider threat detection. Therefore, our clustering approach works well in quitting detection for ITAdmins, but only after the user has actually quit.

## 6.2 Real-world production environment clustering

### Observations

Due to the problems with query limits as described in Section 5.2, we performed the majority of the clustering using one week of data. Since this is relatively short, the feature vector values are unlikely to converge (yet) for every user, as the mean, median and standard deviation shift with every added day of data. The feature most influenced by this is the 'deviation from the average number of daily accesses value'. Weekend or vacation days therefore have significant influence on this value, which is likely to lead to the distribution as seen in Figure 5.11. Manual inspection only revealed that users with a high value of this feature were likely to be outliers in the clustering plot. Such users usually had more accesses than other users. This is, however, also incorporated in other features. Additionally, as we had no data on actual lateral movement or insider threats we removed this feature for the remainder of research using production data, as it did not contribute to explainable clustering.

Outlier user account, or users alone in a cluster, usually proved to be service accounts or admin accounts. Such clusters had a few specific hosts as most important feature as they were the only one to log onto that resource, and logged out outside of office hours more often than other users. This fits the ITAdmin feature values obtained from the CERT dataset, with admin users operating outside of regular office hours more often. Additionally, automated user accounts can run in an automated manner, ignoring office hours. Therefore, being an outlier can be indicative of an automated account or administrator behaviour.

### Inferences on formed clusters

The UserPeerAnalytics and AD group comparison scores, computed through our custom metric, did not come close to the theoretical maximum of 1.0. Therefore we compared our clustering to that of a randomly clustered users, both with the same and a random distribution among clusters. Our clustering consistently outperformed random labels by on average 0.1 for the UserPeerAnalytics and 0.03 for the AD groups. This implies that there is still some signal in the access data related to AD groups, and related to the way Microsoft's UEBA computes their user peer groups. As the metrics to compare our clusters to AD groups and UserPeerAnalytics are niche, we have no way of comparing them to existing literature. We do have to place a note regarding the AD group scoring metric. No scores outside of the 0.40 - 0.47 range were observed for the entirety of the research, and most were in the 0.42 - 0.45 range. This suggests that the eventual score is influenced by the amount of AD groups and their sizes more than it is by our actual clustering. We did, however, in all cases get a higher score for our clustering than for a randomised user labelling using the same distribution. In general, fewer clusters will decrease the AD group comparison score through the user being compared to more other users, who likely differ more in AD groups. If an organisation has many large AD groups, such as 'All users in the Netherlands' for an international organisation, this increases the AD comparison score. Fewer clusters increase the likeliness that more of a user's peers are in the same cluster, increasing the peer score. Therefore gaining information from these scores requires knowledge on the organisation's groups. Additionally, a possible weighted combination of the two scores could be produced, as to avoid the number of clusters heavily skewing the outcome. Either way, prior to using these scores as an indicator of clustering quality one should therefore research the impact the number of clusters has on either of them.

The AD groups featured several groups for users with certain administrative privileges. Further inspection revealed that two clusters encompassed all global domain admins, and almost all users within admin groups. These clusters, however, also contained a lot of non-admin users. Hence, if a user were to move to this cluster, it would not necessarily mean they displayed admin access behaviour. The few accounts which were part of admin AD groups but not in one of the beforementioned two clusters had different characteristics to the rest, such as Job Title or Department. This could point to a misconfiguration, however deeper analysis of the situation requires domain knowledge of the organisation as well as their privilege and AD group overview. This type of analysis is closer to the Baaz [6] tool, and would not require access data as input. There was one cluster with only a handful of members, where the cluster average on the 'outside office hours accesses' feature was significantly higher than any other cluster. However, we did not find a common denominator in e.g. AD groups or IdentityInfo for this cluster. Therefore, we found no correlation usable in practice between our access behaviour clusters and admin privileged users.

## Clustering consistency

The clustering consistency expressed in adjusted Rand value drops significantly compared to the CERT dataset results. When the goal is to detect suspicious user access behaviour through a user switching clusters over time, we require the majority benign users to stay in their original cluster. This is currently not the case, assuming most users in our dataset are benign. Visual PCA inspection shows that whilst there is a clear distinction between the cluster in the CERT dataset in Figure 5.3, for the production data clustering in Figure 5.12 users are gathered together in one large group. For the latter, users are clustered together with a larger factor of randomness, and labels are dependant on where the initial centroids are created. In an effort to consistently 'recognise' groups within the multidimensional space we applied the DBSCAN clustering method, to determine whether it could extract clusters from the data in a less random manner than K-means. Using the same features as K-means, DBscan regarded 80% of users to be outliers, hence not placed in a cluster. Therefore any consistency would only apply to those 20% of labelled users. This would ignore a significant portion of the userbase, and cannot be used to detect suspicious access behaviour by comparing different clusterings over time. As an increase in epsilon value means a decrease in number of clusters for our dataset, this reduces the granularity in user groups. Less user groups leads to coarser user profiles, which diminishes the usefulness in identifying users changing behaviour.

As expected, taking into account the data from an entire month instead of a week increased the consistency scores. This is likely due to the user's feature vectors getting closer to an equilibrium than with weekly data. In addition, it increased the similarity scores to UserPeers as well as to AD groups. Therefore it looks like longer periods of data allow for better inferences based on access behaviour. However, creating a user profile through a longer period of time also has downsides. If malicious behaviour is only shown on e.g. one day of the entire incorporated month, it is likely to have minimal influence on the user's feature vector. Even if the malicious behaviour is shown for a longer period of time, this behavioural change would only be shown after the entire clustering period has passed. Taking mitigating actions against this user might then already be too late. Shorter periods would allow quicker response times, but contain the abovementioned skew and therefore inaccuracy. There exists a potential solution to reducing the slow response time when incorporating the data over a large period of time. In this research, we compared the clustering within different periods one after the other. It is also possible to e.g. compare the clustering which incorporates the data from the previous month on a daily basis: comparing 2022-08-10 until 2022-09-07 to 2022-08-11 until 2022-09-08 and so on. This does however require making the same decision type on how far the lookback window in incorporated data goes, and what is the best interval to create new clusters to compare to the previous ones. However, in the longer term, the same clustering difference would still be observed, as after a month still the same amount of people are on another cluster than they were in the month prior. Additionally, it would require at least as much manual effort from analysts as users could be switching back and forth between clusters. Therefore it is worth exploring whether this gives more insight into why users are placed in a different cluster, but is not likely to solve the consistency problems on its own.

We altered different parts within our methodology in an effort to increase the stability of clusters over time. We removed features which were of no importance to any cluster. Examples include the LogonType feature. This was consistently ranked the least influential feature as the majority of logons are network-based. Including the number of accesses per destination host as feature, or clustering on destination hosts alone also did not lead to the desired result. In the standardisation phase we applied the MinMaxScaler and the QuantileTransformer to create either larger or smaller differences respectively between users' feature values, but this also did not increase cluster stability. To try and closer match the CERT dataset we filtered on just RDP connections, as these logon and logoff events are caused by human interactions instead of continuous background connections. This allowed re-introducing the session-related features. The clustering plots changed, showing a clear diagonal line through the PCA plot. We did not perform further analysis on what this shape implies on the underlying data, but the key inference is that most users are clustered closely together. This likely causes K-means to classify many users differently over time, leading to low similarity scores.

As elaborated in the Methodology section, one of the reasons we employ ML for our feature vectors instead of static thresholds is to account for the dynamic behaviour users have. A user gets placed in a cluster relative to all other users. In the case all other users change behaviour often according to our feature vectors, users will be classified into different clusters often. This is the intended behaviour for our algorithm, but it leads to more workload for a SOC due to an increase in false positives and is, therefore, no added value to them. Besides this, if many users exhibit the same access behaviour, the random factor in K-means clustering can classify users bordering clusters differently every time. As previously discussed, the DBSCAN clustering method is not a viable alternative clustering option when aiming to monitor all users for suspicious access behaviour. We leave the option open that an alternative ML approach exists which can consistently and explainably cluster users based on this data, but have not been able to identify such method during our research.

### Dataset complexities

We were unable to match the clustering stability of the CERT dataset using production data. Our hypothesis for this is that the features from the CERT dataset already show a kind of segmentation between users, whereas for production data most users share a similar value. To prove this we looked at the most important features of both datasets, for the CERT dataset this is the average daily number of accesses, production data has the average off hour number of accesses as more relevant feature. In Figures 6.1 and 6.2 we plotted the standardised feature values per user in a bar chart, with the bars coloured corresponding to the user's cluster. We observe that for the CERT dataset there is a clear distinction in values, and the feature value has a clear correlation with the final cluster a user gets designated to. This is contrary to the production dataset, where apart from the highest few values, the cluster designations are seemingly randomly distributed throughout all feature values. This observation also goes for other relevant features of both datasets.

Because production data lacks this segmentation between user's feature values, it seems there is no clear correlation between this value and the actual cluster a user gets classified as. Therefore, using our method, it is currently unfeasible to cluster users consistently in a production environment.

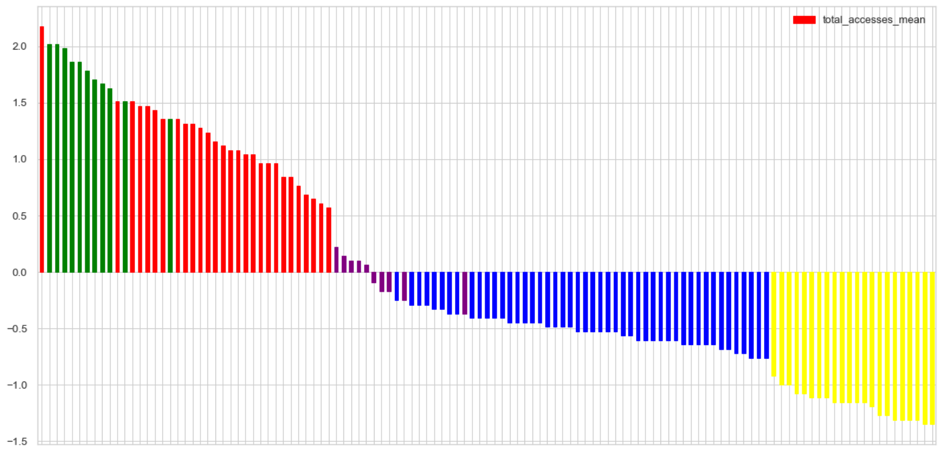


**Figure 6.1:** A bar chart with the standardised values for all users clustered in the CERT dataset for the 'total accesses mean' feature, which is one of the most influential features for the CERT clusters.

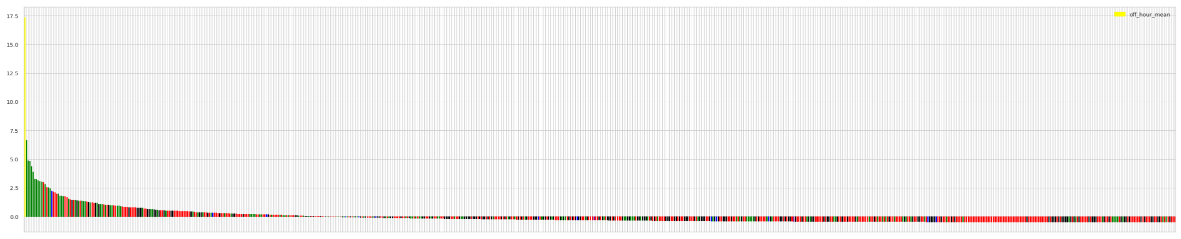


**Figure 6.2:** A bar chart with the standardised values for all users clustered in the Sentinel dataset for the 'off hour mean' feature, which is one of the most influential features for the Sentinel clusters.

## Boundaries of research methodology

Performing research at the University of Twente requires adherence to their ethical standards. As this research involves data on human participants, these standards influence what type of data can be processed and the inferences that could be made on this data. This data gathered would need to be proportional to the goal of the research, profiling users and detecting suspicious behaviour. Combining this with the range of data available to a SOC, we deemed access data to be the the most suitable, motivated by successful results by related work. The access data consisted of logon events of devices onboarded to Sentinel, which commonly consist of important servers such as DC's and file servers. We accepted any possible bias in the data with regards to for example demographic and user behaviour, as removing any skew from the dataset would make it less applicable to real-world data. The ML applied would need to be as explainable and comprehensible, which limited the options and contributed to choosing the primitive general-purpose K-means clustering method. Our methodology on feature creation and the clustering thereof was tested and validated the CERT dataset, which yielded positive results with regards to consistency. This looked promising for real-world data inferences. However, in reality, the values of the features created are too close together to create meaningful clusters or create consistent user profile groups. Creating more granularity between users would require either different features from the current input data. More likely, however, it requires different input data to be able to consistently differ between users, as this was used by related work which achieved more successful profiling and detection scores.

## Comparison with similar research

Our research applied a primitive clustering algorithm to access data to profile users consistently. If consistent profiling was acquired, it would be possible to detect users which deviate from their previous known behaviour, or are an outlier to the rest of the organisation. This approach was validated using the synthetic CERT dataset, where users were profiled consistently using mainly time and resource-based features. In production, using the same features, we could not achieve the same consistency in clustering due to the similarity in most users' access behaviour. If we compare our research to other work in this field, they often use a more sophisticated ML approach which is more tailored to the data, but outperform our ability to identify suspicious user behaviour by far. In comparison with the majority of related work, we only use a fraction of their input data for creating feature vectors. Research with the same objective as ours used the CERT dataset to benchmark their algorithm, but in addition to logon data incorporated amongst others HTTP traffic and email statistics to get a user profile. If a user suddenly visits more job searching websites, changes from their regular number of words per email or attaches more files to mails than usual, this can raise an alert. We suspect that one of the main differences between our work and others with the same objective, is that we attempted to detect the irregularity in user behaviour by the user accessing a resource which is not common for their account. We tried to capture the malicious behaviour itself, by showing signs of lateral movement or network discovery. Other works most likely detect the behaviour that employees show *prior* to their insider threat actions, such as looking for different jobs or changing their mailing behaviour. They use a lot more indicators than just resources accessed, which enabled them to get a very broad, possibly invasive, perspective on how every user usually behaves. Where our work tried to capture most types of malicious behaviour using a legitimate user account, others captured the user's digital footprint and looked for deviations in this. If the model gets trained well enough to detect the behaviour of users which are ready to quit with their job, this flags potential insiders. As mentioned in the previous section, this is quitting detection: using quitting behaviour as a proxy to indicate potential insider threats. In the literature we researched, the creation of such models good enough to deduce this behaviour uses too invasive data on user behaviour to be applicable in practice. Therefore in any future research on this topic it is important to note the difference between what is possible to infer using synthetic datasets and what data types is actually possible to be incorporated in reality. This invasive data is possible to generate for synthetic datasets and achieve high insider threat detection scores, but any setup incorporating such data would never reasonably be permitted in the real world.

## 6.3 Clustering in production environment evaluation and future opportunities

This work gives a strong indication that it is not possible to use a primitive ML algorithm in a production environment to profile users in order to detect suspicious access behaviour. Our approach was tested and validated against the data of different medium to large organisations. For all organisations, similar results were seen observed regarding the data and clustering metrics. Each of them had little difference in feature values between users, tight clusters and inconsistent clustering.

Our focus was on illegitimate resource accesses as an indicator of malicious behaviour, but other research indicates a more effective approach could be to detect users who are about to quit their job. Quitting detection in production would be significantly more invasive than detecting the suspicious actions themselves, both in terms of data used and type of information inferred. The validation we performed on the CERT dataset in terms of insider threat detection was also related to a user quitting their job, but only after the user had performed their malicious actions. Our approach could provide a functional product using synthetic data, due to its clustering stability for benign users. For production data however, the features used to profile user access behaviour are too fluid and do not generate enough distinction between users to classify them consistently.

If parts of our approach are used in the future, it is recommended to change the clustering algorithm. The element of randomness in K-means clustering can cause users in bordering clusters to switch continuously over time. Additionally, the gravitational centers (centroids) tend towards larger groups. Many people sharing the same behaviour will therefore skew the rest of the groups. In our research, every resource had as much influence as any other as feature in the multidimensional area, whilst in reality some resources could be more relevant for clustering than others. This can be done according to the resource importance, or type such as file server or domain controller. That way, it is likely that one of the clusters will contain users accessing the crown jewels amongst resources, which are likely to be administrators. Giving weight to resources enables detecting users suddenly connecting to vital infrastructure whereas usually they did not. Combining this with other properties of an access event, such as using RDP for the first time to connect to a DC[1], is significant data when keeping track of an 'anomaly score', such as our cumulative difference from the mean number of accesses. This also requires mapping adversarial actions to potential indicators in the data, the RDP connection to a DC in the beforementioned case. In addition to resources, of the weight of any feature could of course be increased relative to less important features. Whilst we had statistics per cluster on which resources were accessed most often, this did not enable us to draw a picture on what type of users populated a cluster. As the resources themselves were a significant feature for the clustering, classifying the resources could show a 'bigger picture' of what types of resources users access, providing a better picture of a cluster's regular behaviour. Therefore, in future research we would also recommend categorizing the resources and using the categories as features.

In the Sentinel data gathering phase our approach still required a lot of manual work in removing outliers which were deemed automated accounts. Additionally, because of the amount of events we had to summarise the logon data per hours, and for monthly clustering we manually aggregated data of several weeks because of the 500.000 row limit on KQL queries. Removing redundant 'noisy' accounts in an earlier phase preserves more of the 500.000 row limit to the accounts which are actually used for the clustering. This could be computed and filtered by KQL in the query itself already. Furthermore, KQL is able to count events and aggregate data, so is also able to compute the full feature vectors per user from the raw data. Any future research incorporating this data is therefore recommended to perform as much computation as possible within the KQL query itself.

[1]Adversarial indicators: `https://www.microsoft.com/security/blog/2022/10/18/defenders-beware-a-case-for-post-ransomware-investigations/`

In our research, we expected the clusters to enable identifying roles within the organisation. Apart from the ITAdmin role in the synthetic dataset we did not find any correlation between clusters and roles. A possibility to we did not explore is to take the Microsoft UEBA IdentityInfo table and extract every user's role-data. For every user, a feature vector on access behaviour could be created and used in a supervised ML approach. The label for every user then represents their role. This way, a classifier could be trained on which access behaviour belongs to which roles. If this is accurate, anomaly detection could be applied to detect whether someone stays within the cluster of their own role.

Chapter 7

# Conclusion

This section wraps up our research by answering the research questions and listing our contributions to the field.

## Which non-invasive features readily available to a SOC are relevant in describing user access behaviour?

Logon events are suitable to keep track of *which user* accessed *which resource* at *which time* and in *which manner*, which in Windows infrastructure translates to Events 4624 (logon), 4634 and 4647 (both logoff). This enables profiling user's regular logon and logoff times, session duration, which and how many different times resources are accessed, the method of authentication and the connection type. These results are aggregated over the timespan of a day, and over a larger period of time the mean, median and standard deviation of these values are computed.
In the synthetical dataset, the mean and standard deviation of daily accesses and accesses during office hours are the most relevant for clustering. For production data, the mean value of daily accesses outside of office hours, the mean value of daily accesses within office hours, and total amount of accesses to a selection of destination hosts are the most important features.

## Which explainable machine learning method can be used to profile users?

We selected K-means clustering because of its general-purpose application, its straightforwardness and requiring just a single input parameter which can be computed in an automated manner. K-means clustering enables consistent clustering when there is a proper segmentation in the values of features within the dataset. Then it is possible to explain the clustering by computing feature importance and the average feature values for clusters. However, if there is not a large distinction between many user's values, then small deviations can lead to significantly different clusterings. This is due to the the primitive method K-means works, combined with the randomness of the initialisation phase.

## How does our profiling methodology perform within the CERT v4.2 synthetic dataset on consistency and extracting user groups?

The profiling methodology creates a clear distinction between users with the ITAdmin role and users without. The distinctive feature values of ITAdmins show that they perform more accesses, have shorter session durations and more accesses outside of office hours compared to other users. Other than the ITAdmins, no clear role distinction could be made. For the K=5 value the clusters are consistent with an average adjusted Rand score of 0.87 for consecutive months.

## How does this profiling methodology translate to real-world data in terms of consistency and extracted user groups?

For production data our features show less separation between the users, so slight behavioural changes causes users to switch clusters often. Additionally, the K-means initialisation phase is random so users close to bordering clusters can randomly switch clusters. Increasing the timespan over which the features are created has a slight positive effect on the consistency, but stays at an adjusted Rand score of 0.41 for consecutive periods. Hence, the consistency does not come close to the one achieved for the synthetic CERT dataset and cannot be used in practice for identifying suspicious access behaviour.
There is a mediocre similarity between our extracted clusters, AD groups and Microsoft UEBA's UserPeerAnalytics. The scores for our custom performance metrics do not come close to the hypothetical maximum of 1.0, but are consistently better compared to users being classified into clusters at random. This implies there is a signal in the data, but via our methodology does not lead to usable results.

## Is it possible to profile users consistently using explainable clustering and non-invasive data sources available to a Security Operations Center?

We used logon events to create privacy-preserving feature vectors for every user, based on literature research and threat intelligence. For the synthetic dataset, we

were able to detect 80% of ITAdmin insiders through them changing cluster in the duration of their insider threat actions. This is likely due quitting after the malicious actions have taken place.
For real-world data, we conclude that there is not enough granularity between these features to consistently profile users. Through this methodology, it is not possible to determine deviating behaviour accurately using solely logon data. Profiling users consistently using explainable clustering requires at least different pre-processing to bring more granularity to the available data. The exact insider threat or account compromise actions should be determined beforehand to tailor to an adversarial indicator. More invasive data is likely required to differ between users, as was used by similar research to achieve higher insider threat and account compromise detection scores.

To summarise, we converted non-invasive access logs into a feature vector per user. We applied K-means clustering to these feature vectors and compared the formed clusters with known user groups. We also compared clusters of different periods in time to each other. Whilst for a synthetical dataset this yielded promising results, in production data we could not get useful inferences from the resulting clusters. It was also not possible to construct consistent user profiles over time, since we were unable to separate users clearly on their access behaviour. Therefore it could not be used by a SOC to detect suspicious deviations in user access behaviour.
Because of these findings, the Northwave SOC is looking into how they can include other types of data to achieve more granularity between users. Since we were unable to use the clusters in a meaningful manner in a real-world setting, another Machine Learning method will be applied that is targeted solely at anomaly detection.

# Bibliography

[1] "Identity and Access Management: The Stakeholder Perspective — Identity Defined Security Alliance." [Online]. Available: https://www.idsalliance.org/white-paper/identity-and-access-management-the-stakeholder-perspective/

[2] B. Lindauer, "Insider Threat Test Dataset," 3 2020. [Online]. Available: https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247

[3] B. C. Neuman and T. Ts'o, "Kerberos: an authentication service for computer networks," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 33–38, 1994.

[4] W. M. Rand, "Objective Criteria for the Evaluation of Clustering Methods," *Source: Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[5] L. Hubert and P. Arabic, "Comparing Partitions," *Journal of Classification*, vol. 2, pp. 193–218, 1985.

[6] T. Das, R. Bhagwan, and P. Naldurg, "Baaz: A system for detecting access control misconfigurations," in *Proceedings of the 19th USENIX Security Symposium*, 2010, pp. 161–176.

[7] L. Bauer, S. Garriss, and M. K. Reiter, "Detecting and resolving policy misconfigurations in access-control systems," in *ACM Transactions on Information and System Security*, vol. 14, no. 1, 2011.

[8] R. Agrawal and R. Srikant, "Fast Algorithms For Mining Association Rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[9] A. Kamra, E. Terzi, and E. Bertino, "Detecting anomalous access patterns in relational databases," *VLDB Journal*, vol. 17, no. 5, pp. 1063–1077, 2008.

[10] E. Bertino, A. Kamra, E. Terzi, and A. Vakali, "Intrusion detection in RBAC-administered databases," in *Proceedings - Annual Computer Security Applications Conference, ACSAC*, vol. 2005, 2005, pp. 170–179.

[11] G. Pannell and H. Ashman, “Anomaly detection over user profiles for intrusion detection,” in *Proceedings of the 8th Australian Information Security Management Conference*, 2010, pp. 81–94.

[12] X. Wang, Q. Tan, J. Shi, S. Su, and M. Wang, “Insider threat detection using characterizing user behavior,” in *Proceedings - 2018 IEEE 3rd International Conference on Data Science in Cyberspace, DSC 2018*, 2018, pp. 476–482.

[13] M. Garchery and M. Granitzer, “Identifying and Clustering Users for Unsupervised Intrusion Detection in Corporate Audit Sessions,” in *Proceedings - 2019 IEEE International Conference on Cognitive Computing, ICCC 2019 - Part of the 2019 IEEE World Congress on Services*, 2019, pp. 19–27.

[14] K. A. tabash and J. Happa, “Insider-threat detection using Gaussian Mixture Models and Sensitivity Profiles,” *Computers and Security*, vol. 77, pp. 838–859, 2018.

[15] Y. Chen, S. Nyemba, and B. Malin, “Detecting anomalous insiders in collaborative information systems,” *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 3, pp. 332–344, 2012.

[16] C. Gates, N. Li, Z. Xu, S. N. Chari, I. Molloy, and Y. Park, “Detecting insider information theft using features from file access logs,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8713 LNCS, no. PART 2, 2014, pp. 383–400.

[17] J. Glasser and B. Lindauer, “Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data,” in *2013 IEEE Security and Privacy Workshops*, 2013, pp. 98–104.

[18] B. Sharma, P. Pokharel, and B. Joshi, “User Behavior Analytics for Anomaly Detection Using LSTM Autoencoder-Insider Threat Detection,” in *ACM International Conference Proceeding Series*, 2020.

[19] S. Hochreiter and J. J. Urgen Schmidhuber, “Long Short-Term Memory,” Tech. Rep. 8, 1997.

[20] A. H. Mirza and S. Cosan, “Computer Network Intrusion Detection Using Sequential LSTM Neural Networks Autoencoders.”

[21] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, “Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams.”

[22] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, “Analyzing Data Granularity Levels for Insider Threat Detection Using Machine Learning,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 30–44, 3 2020.

[23] D. C. Le and N. Zincir-Heywood, “Anomaly Detection for Insider Threats Using Unsupervised Ensembles,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1152–1164, 6 2021.

[24] G. Gavai, K. Sricharan, D. Gunning, J. Hanley, M. Singhal, and R. Rolleston, “Supervised and Unsupervised methods to detect Insider Threat from Enterprise Social and Online Activity Data.”

Appendix A

# Appendix

## A.1 Statistics on the most influential features amongst all different roles

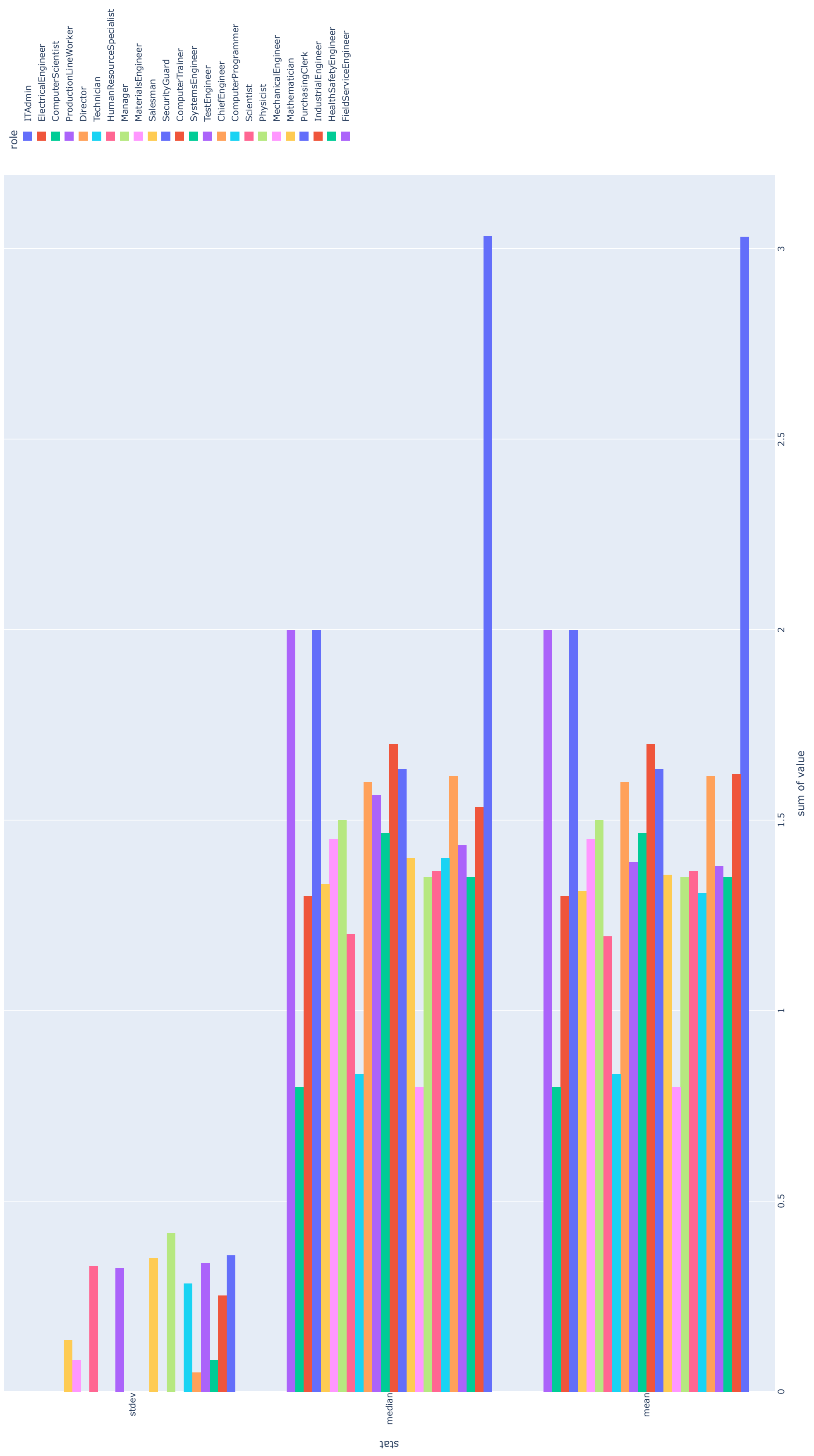


**Figure A.1:** The mean, median and standard deviation of the Average Total Daily Accesses feature amongst all roles. The ITAdmin value is the last purple bar per statistic.

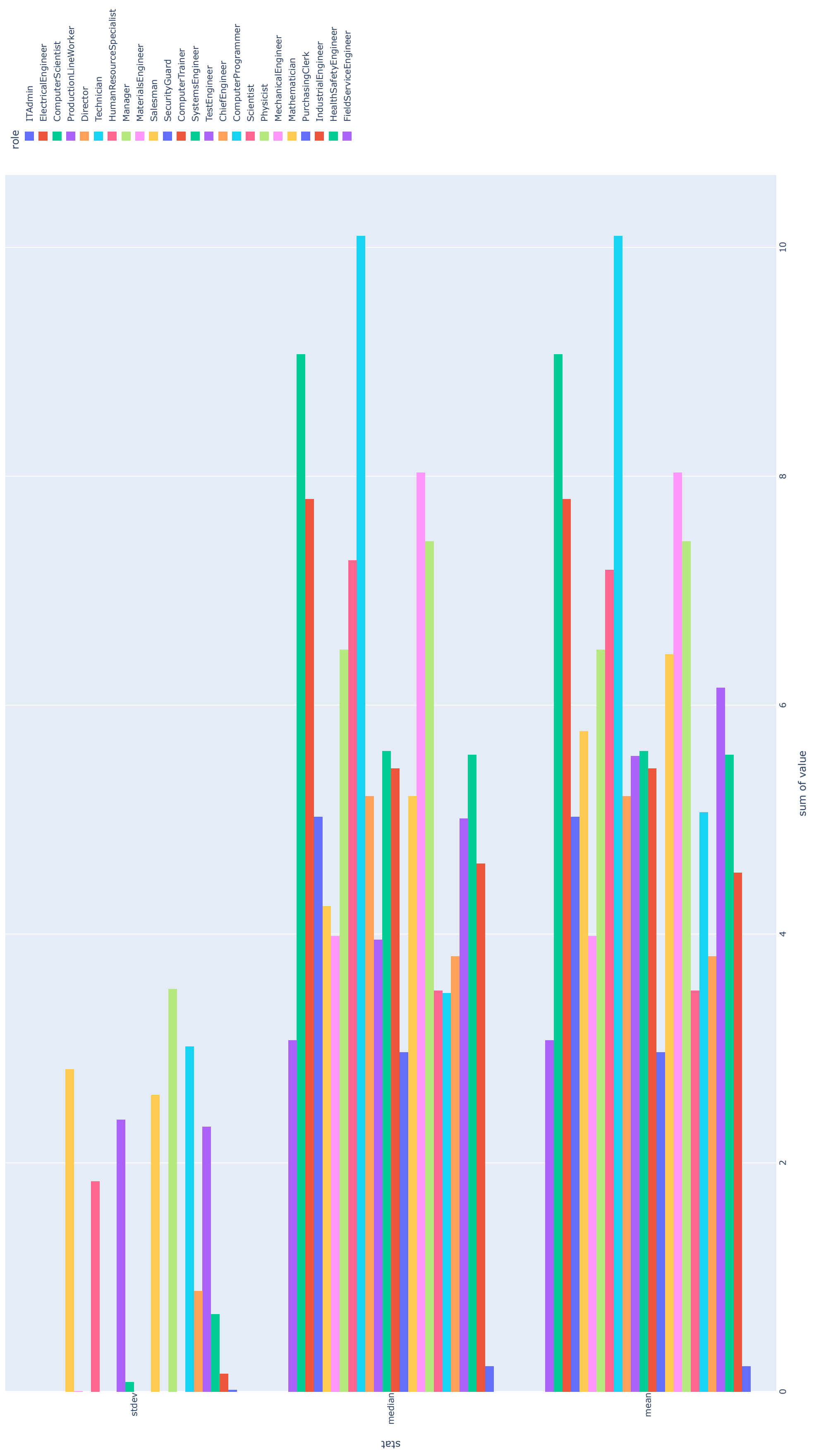


**Figure A.2:** The mean, median and standard deviation of the Median Session Length feature amongst all roles. The ITAdmin value is the last purple bar per statistic.

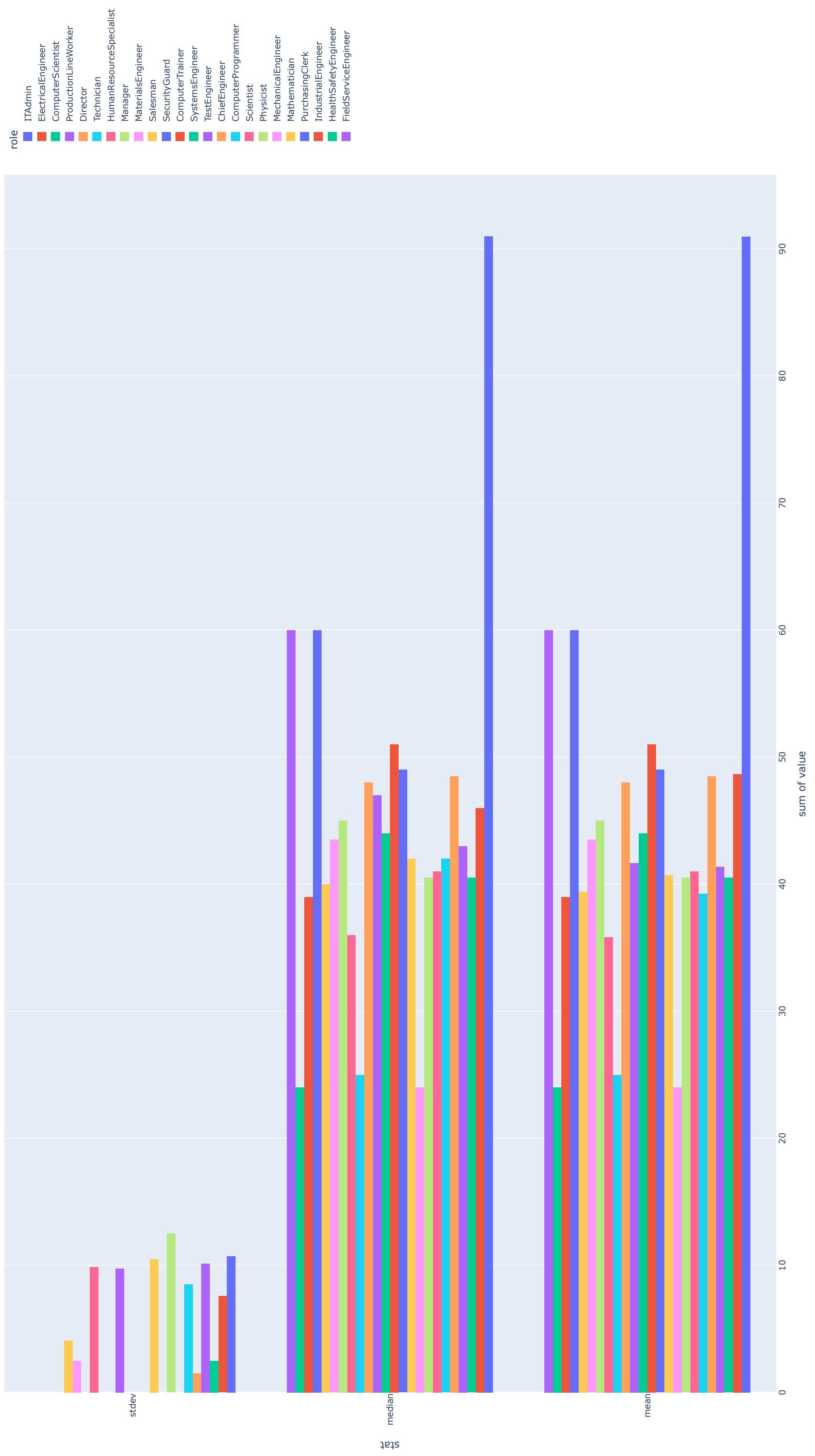
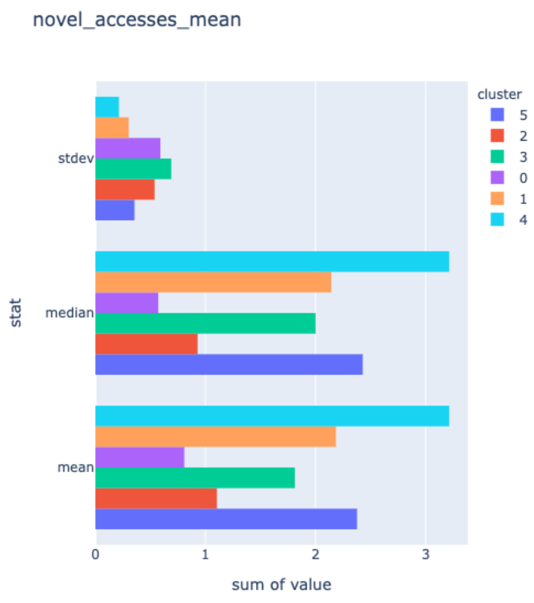
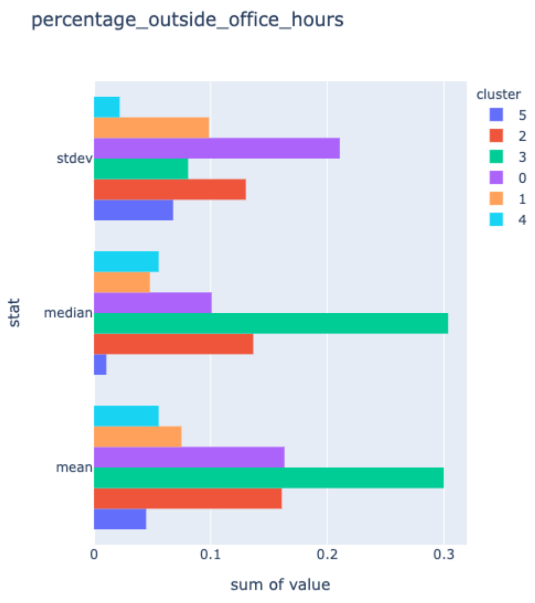


**Figure A.3:** The mean, median and standard deviation of the Total Logons feature amongst all roles. The ITAdmin value is the last purple bar per statistic.

## A.2 Production data feature properties



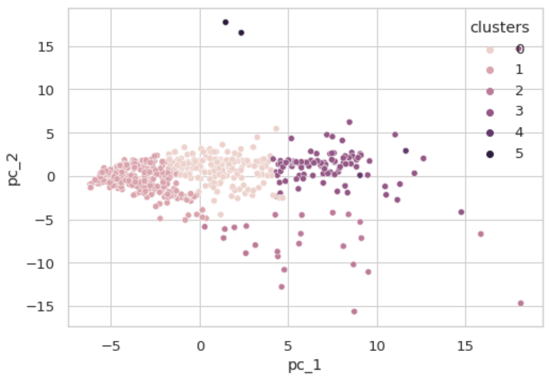
*(a)* Statistics per cluster for the 'novel accesses mean' feature.



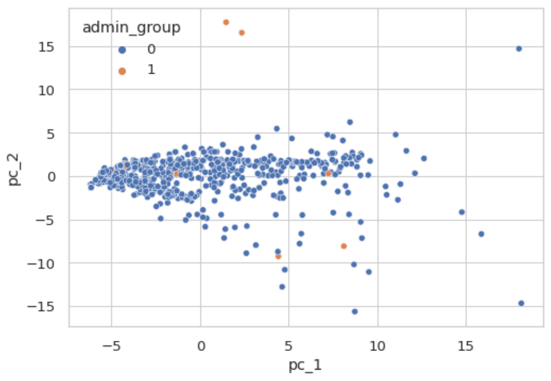
*(b)* Statistics per cluster for the 'percentage outside office hours' feature.

**Figure A.4:** The mean, median and standard deviation of two influential feature vectors, computed per cluster. Two clusters (0 and 2) contain the majority of users with admin rights and have values divergent from the other clusters.

## A.3 Sentinel data clustering plot comparison to administrator labelled plot.



*(a)*



*(b)*

**Figure A.5:** Two plots of the same clustering of Sentinel data. *(a)* is coloured based on cluster, *(b)* has administrator users coloured in orange. We observe that one of the clusters consists of solely administrator users.

## A.4 Clustering stability over time

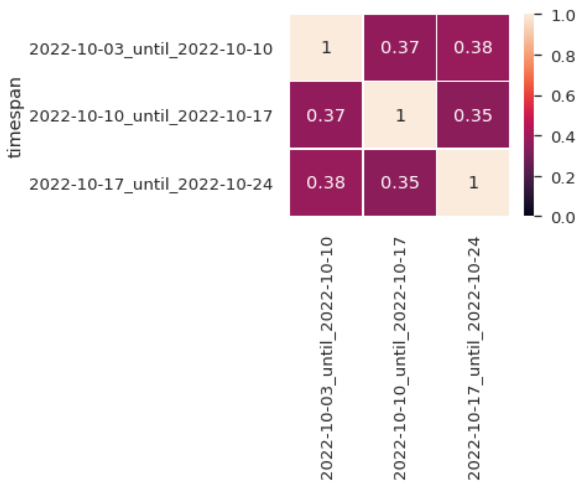


**Figure A.6:** Clustering stability comparison over different weeks, for an organisation different to the one shown in Figure 5.14, but using the same methodology.

## A.5 PCA composition for Sentinel data filtered on RDP events

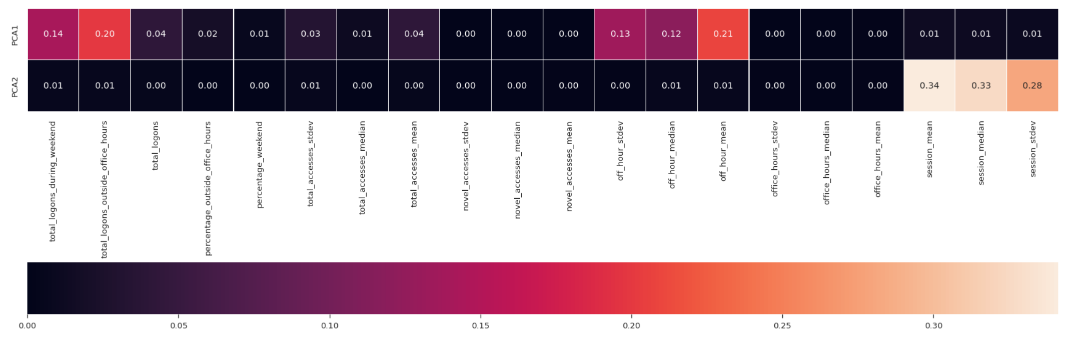


**Figure A.7:** The PCA composition a for Sentinel clustering plot where the data was filtered on logontype=10 (RDP) events.

## A.6 Overview of features used in CERT Insider Threat dataset clustering

| **Available feature** | **Used?** |
|---|---|
| Total logons during weekend | Yes |
| Total logons outside of office hours | Yes |
| Total logons | Yes |
| Percentage of accesses outside of office hours | Yes |
| Percentage of accesses during the weekend | Yes |
| Total unique accesses | Yes |
| Access amount per timeframe | Yes |
| Novel accesses per timeframe | Yes |
| Accesses outside of office hours per timeframe | Yes |
| Accesses during office hours per timeframe | Yes |
| Session length per timeframe | Yes |
| Cumulative deviation from the running average number of accesses per timeframe | No |
| Every destination host | No |

**Table A.1:** All features available the CERT dataset and whether they were used for the clustering.

## A.7 Overview of features used in Sentinel dataset clustering

| Available feature | Used? |
|---|---|
| Total logons during weekend | Yes |
| Total logons outside of office hours | Yes |
| Total logons | Yes |
| Percentage of accesses outside of office hours | Yes |
| Percentage of accesses during the weekend | Yes |
| Total unique accesses | Yes |
| Access amount per timeframe | Yes |
| Novel accesses per timeframe | Yes |
| Accesses outside of office hours per timeframe | Yes |
| Accesses during office hours per timeframe | Yes |
| Session length per timeframe | No |
| Cumulative deviation from the running average number of accesses per timeframe | No |
| Every destination host | Yes |
| Every method of authentication | Yes |
| Every type of logon | No |
| Every source of the access | No |

**Table A.2:** All features available the Sentinel dataset and whether they were used for the clustering.