UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Detection and Classification of Cybersecurity User/Entity Behaviour Anomalies on Web events

João Pedro da Silva Neves

Mestrado em Ciência de Dados

Trabalho de Projeto orientado por:
Prof. Doutor Vinicius Vielmo Cogo

2024

Acknowledgments

Completing this thesis was a challenging yet significant and enriching journey, one that would not have been possible without the unending support of those who guided me along the way.

I would first like to express my sincerest thanks to Professor Vinicius Cogo for his unwavering support and insightful feedback, which was instrumental in shaping this research throughout the year-long process.

I am also very grateful for the ever-present help, experience, and invaluable knowledge that Ricardo Ramalho provided during the development of this project. Likewise, I am profoundly thankful for the amazing support, readiness, and crucial insights that Madalena Guerra offered, without which I would not have been able to overcome certain obstacles in this thesis.

I would also like to thank the DCY team for their tireless efforts, always being available to help when needed, and for providing such a great opportunity and environment to develop my project.

I am also filled with the utmost gratitude from all of my closest friends for all of the constant moral boosting, experience sharing and for making this year much more enjoyable.

Finally, I want to extend my most special thanks to my family, especially to my parents, Sandra and José, for their unending love, sacrifices, and support throughout my entire academic journey, which were the key factor in all my achievements and success over the years.

Dedicated to my closest family and friends

Resumo

A evolução constante dos processos tecnológicos aliado do incessante aumento e dependência de dados de todos os tipos têm vindo lentamente a alterar a visão e metodologias que as empresas tomam para se defenderem dos vários perigos existentes. Prevenir ataques externos deixou de ocupar a maior parte da atenção, passando a existir uma crescente preocupação com ataques internos. Milhares de empresas e organizações hoje sofrem ataques internos que podem chegar a custar milhões de dólares cada, dependendo dos estragos que estes provocam, por exemplo, exfiltração de dados e interrompimento de sistemas. A frequência destes ataques tem levado a investigação constante de novos métodos e técnicas que consigam prevenir ou rapidamente detetar as intrusões internas.

O projeto apresentado nesta dissertação foi realizado em parceria com o Departamento de Cibersegurança (DCY) da Altice Portugal e aplica uma das tecnologias emergentes que mais se mostra capaz de prevenir os diversos ataques internos que podem ocorrer. O nome popularizado da técnica é *User and Entity Behavior Analytics* (UEBA) e passa por utilizar metodologias diversificadas de forma a detetar comportamentos suspeitos das entidades consideradas. Este conjunto complexo de técnicas é um sistema de gerenciamento de riscos que utiliza análise de dados em tempo real para monitorizar utilizadores e/ou entidades, por exemplo IPs ou máquinas. Para tal, os dados são processados e avaliados com técnicas estatísticas e de Aprendizagem Automática.

Uma das grandes vantagens desta metodologia é a sua capacidade de generalização, tanto no tempo como no tipo de ataque, que vem possibilitar a deteção de ataques de dia zero. Um ataque de dia zero é uma vulnerabilidade de um software desconhecida que é explorada pelo atacante. Por outras palavras, é um tipo de ataque que não foi antes reconhecido pelos analistas. Previamente ao UEBA, os métodos mais comuns para deteção de ataques internos eram os métodos que se baseavam nas assinaturas específicas de cada ataque (*Signature-based*). Estes sistemas eram construídos sobre a informação de ataques já conhecidos, dependendo de regras escritas à mão. Apesar de simples, eles são apenas capazes de detetar ataques já encontrados anteriormente, pelo que o UEBA é um método que traz numerosas vantagens.

Esta tese tem o objetivo de desenvolver um sistema capaz de detetar possíveis ataques internos ou máquinas possivelmente infetadas que pousem perigo à Altice Portugal. Para esse efeito, vão utilizar-se dados no formato *log* referentes a acessos web por parte dos colaboradores da empresa que foram sinalizados por um serviço *cloud* de web proxy, possivelmente por apresentarem alguma ameaça. Cada *log* carateriza totalmente a razão da sinalização do acesso, incluindo informação referente ao URL, classificação de ameaças, IPs, solicitações HTTP, etc.

Como já mencionado, a investigação nesta área tem vindo a crescer ao longo dos anos. No entanto, esta pesquisa complica-se por vários fatores diferentes, como o comportamento dinâmico das entidades no tempo e a alteração do significado de anomalia de domínio para domínio, não existindo um consenso geral na melhor metodologia a tomar para estes sistemas, significando que cada caso é um caso. Para além disto, a maioria da investigação feita não utiliza dados reais, recorrendo antes a bases de dados sintéticas que tentam reproduzir ataques reais, pelo que não existem muitas metodologias aplicadas em casos reais de empresas. Adicionalmente, a grande maioria dos trabalhos que existem apenas exploram a dimensão do utilizador, havendo pouco investimento em como traduzir comportamento anómalo de outro tipo de entidades, como máquinas, que são, também, objeto de estudo nesta tese.

Visando estas dificuldades, esta dissertação descreve a construção de dois detetores diferentes: um que se especializa em detetar utilizadores que consistentemente apresentam um comportamento suspeito e um outro que se especializa em detetar máquinas que aparentam possuir um comportamento infetado, cada um considerando uma quantidade de dados adequada ao seu propósito. Este sistema foi construído em *Python*, utilizando *Jupyter Notebooks*.

O sistema passa inicialmente por uma fase de processamento, integração e transformação de dados que, tendo estes num índice de *ElasticSearch*, os extrai, filtra e trata com ferramentas como a biblioteca *pandas*. Os dados são continuamente enriquecidos e alterados até chegarem a um estado semelhante a um perfil da entidade em estudo, seja de atividade diária ou do comportamento geral (dois datasets diferentes), sendo este último caraterizado por um conjunto vasto de campos estatísticos que descrevem toda a atividade da entidade na janela de tempo considerada. Este processo tinha como objetivo uma transformação que conseguisse preservar um equilíbrio ótimo entre custo computacional e qualidade de informação que foi preservada no processo. O resultado final mostrou-se capaz disto, conseguindo filtrar e preservar informação suficiente sobre cada entidade que permitisse uma deteção de anomalias facilitada.

No que toca aos perfis de utilizadores, o sistema introduz a seguir um módulo de deteção que se divide em 3 partes. A primeira parte consiste na deteção de *outliers* puramente estatísticos de forma a tentar descobrir utilizadores que, univariadamente, saltem imediatamente à atenção. Seguidamente, os utilizadores são agrupados utilizando algoritmos de *Clustering*. Observou-se que a separação ótima de utilizadores era em 2 grupos, sendo um deles caraterizado por utilizadores que não apresentavam perigo à empresa (*baseline*) e outro por utilizadores que apresentavam indícios de comportamento suspeito (grupo anómalo). Cada grupo teve, então, um tratamento diferente. Para o grupo da *baseline*, como este não apresenta ameaça imediata, o sistema utiliza a atividade diária de cada utilizador para criar um perfil para cada utilizador e compara depois estes perfis a novos dados que entram diariamente de forma a investigar se cada utilizador se está a tornar um perigo. Já para o grupo anómalo criou-se um *ensemble* de modelos especializados em deteção de anomalias para se detetar os utilizadores que, dentro do grupo, mais se destacam por serem consistentemente mal comportados. No entanto, sendo estes dados reais, não existe um campo com informação do que é anómalo ou não. Por outras palavras, não é possível treinar os modelos.

Para ultrapassar este desafio, injetaram-se anomalias sintéticas criadas a partir dos dados originais que possibilitaram a criação de uma verdade-base. Esta metodologia também permitiu avaliar o desempenho dos modelos e as deteções feitas.

Para detetar máquinas possivelmente infetadas, o sistema percorre outro processo. Este processo foi extremamente simplificado graças à criação de campos que traduzem bem a infeção das máquinas ao longo de um dia e ao longo de todo o período de tempo. Tendo esta informação disponível, o sistema cria um conjunto de dados referentes a um período mais recente da janela de tempo (dataset recente) e o resto dos dados são utilizados para criar um dataset semelhante (dataset histórico). O detetor passa depois por comparar estes dois datasets, bem como comparar as máquinas entre si, de forma a detetar possíveis infeções.

Cada uma das deteções que os dois detetores fizeram foram caraterizadas e justificadas a partir de um relatório em Excel com informações adicionais que auxiliam o analista a compreender e diagnosticar cada anomalia detetada, facilitando o seu trabalho. Os relatórios gerados incluem ainda um link para um *dashboard* na plataforma de visualização de dados Kibana, onde se exibem vários gráficos que tentam explicar a atividade que gerou a anomalia.

O sistema que foi desenvolvido no âmbito desta tese mostrou resultados promissores, sendo capaz de se adaptar ao comportamento dinâmico de utilizadores, detetando com sucesso colaboradores que mostravam um comportamento consistentemente mau, bem como manter registro de possíveis alterações súbitas que houvesse em utilizadores não considerados perigosos. Para além disto, o detetor de máquinas infetadas também conseguiu atingir as suas metas, encontrando máquinas com alta possibilidade de infeção. Os relatórios apresentados à equipa de analistas mostraram-se suficientes para esta fase inicial, conseguindo indicar claramente a natureza da anomalia, permitindo assim um aumento da velocidade de resposta da empresa perante casos comportamentais inesperados.

Palavras-chave: Análise de comportamento de usuários e entidades (UEBA), Ameaças de Segurança Interna, Aprendizagem Automática, Deteção de Anomalias, Cibersegurança

Abstract

For the past decades, companies have been dealing with insider attacks that can cost them up to millions of dollars each. Finding a way to prevent or quickly detect these attacks has become of utmost priority to the enterprises, leading the industry to develop techniques that can do so. In the past years, User and Entity Behaviour Analytics (UEBA) have taken the spotlight as the leading algorithms to detect and prevent insider attacks due to their power of dealing with 0-day attacks and real-time detection capabilities.

This thesis, conducted at the cybersecurity department (DCY) of Altice Portugal, has the goal of detecting possible insider threats or machine infections that could put the company in danger by using the collaborators' flagged web accesses through the analysis of a web proxy's web insight logs.

To achieve this, a UEBA detecting system was developed to find both suspicious users and infected machines. This system includes a data processing phase, which extracts, integrates and transforms the data into a user/machine profile-like state.

The user profiles pass through a complex pipeline that manages to deal detect consistently misbehaved users effectively by using Machine Learning techniques which specialize in outlier detection, trained on injected anomalies which formed a synthetic ground truth. It also includes a parallel pipeline for users who have been determined not to be dangerous which detects dangerous daily variations which might indicate that the user is becoming compromised.

The machine profiles undergo a different pipeline which effectively compares different periods of times and machines to find which machines present the most infected behaviors. Both detectors, in the end, present descriptive reports and analytical dashboards for each found anomaly.

The injected anomalies and several rounds of results have indicated that both detectors appear to be successful in each of their respective task.

Keywords: User and Entity Behavior Analytics (UEBA), Insider Threat Detection, Machine Learning, Anomaly Detection, Cybersecurity

Contents

List of Figures

List of Tables

Abbreviations

DCY Department of Cybersecurity.

DL Deep Learning.

EDR Endpoint Detection and Response.

ETL Extract, Transform, Load.

GMM Gaussian Mixture Model.

iForests Isolation Forests.

IMD Infected Machine Detector.

IQR Interquartile Range.

kNN k-Nearest Neighbor.

LODA Lightweight Online Detector of Anomalies.

LOF Local Outlier Factor.

MCD Minimum Covariance Determinant.

ML Machine Learning.

OCSVM One-class Support Vector Machine.

OS Operating System.

PC Principal Component.

PCA Principal Component Analysis.

SOC Security Operations Center.

SQL Structured Query Language.

SVD Singular Value Decomposition.

t-SNE t-Distributed Stochastic Neighbor Embedding.

ts Timestamp.

TSC ThreatSuperCategory.

UBA User Behavior Analytics.

UD User Detector.

UEBA User and Entity Behavior Analytics.

XDR Extended Detection and Response.

Chapter 1

Introduction

1.1 Motivation

The evolution of technology and available information on the Internet has brought about countless forms for attacks to occur and for insiders to act [4,53]. To counteract this evolution, the existing defending mechanisms had to follow the rise in threats. As a result, a bigger monitoring on the activity of internal systems began so that user accounts that are compromised or ill-intentioned insiders were detected within the smallest time window possible [6].

Verizon's Data Breach Investigations Reports [51] showed that 74% of all breaches include the human element and 83% of breaches involved external actors, mostly motivated by financial reasons. According to the IBM Cost of a Data Breach Report [20], the global average cost of a data breach in 2023 was $4.45 million, 15% more than 2020. Additionally, in most of these breaches, a third party detected it. For example, we have law enforcements filing in for complaints for TJX Companies in 2006 or Yahoo's partner warning them of a breach in 2014 [11,42,52]. It can also happen that a client may detect the breach, as it happened to LinkedIn back in 2016 [42].

These trends bring to light the critical need for enhanced and modern cybersecurity measures that can swiftly detect and mitigate both insider and external threats to safeguard organizational assets [3,7].

1.2 Problem Statement

A company's usual goal is to prevent outside attacks, leaving little precautions for insider attacks. Legitimate users usually have more authorizations than any non-legitimate user, which leads to the thought of how to act under compromised legitimate users [23]. Raut *et al.* [38] define three types of insiders: the Traitors, who betray the company for their own gain, the Masqueraders, who pretend to be employees, and the Unintentional perpetrators, who accidentally put the company at risk as a result of poor security practices. Hence, these insiders pose a great danger for companies. With the right accesses, they are able to wreck havoc in a matter of minutes. They can perform data exfiltration, disrupt systems, etc., resulting in loss of capital, reputation and customers [2,23,42].

Moreover, this is not the only problem one can face. Not all threats are internal and can be, for

example, a machine which was infected with malware originated from external attackers that is just loading the system with lots of unwanted traffic. Malware infected machines can form botnets that launch coordinated attacks, such as flooding targets with packets or searching for personal and financial information or even, in this case, confidential information of the enterprise [41].

To address this challenge, the company's systems must quickly spot any unusual activity within the company itself while ensuring it poses a genuine threat. The solutions should leverage modern security approaches such as Signature-based methods or User/Entity Behaviour Analytics (UEBA). While the first method relies on blacklist signatures of previously detected attacks [12], the second method employs Machine Learning and statistical methods to detect anomalies in user or entity behavior swiftly and effectively [23, 24], making it able to detect previously unseen attacks in real-time, and, therefore, more suitable for the landscape of today's cybersecurity internal threats.

1.3 Goals

The main objective of this thesis is to build an automated system whose main functionality is to detect anomalous behaviour on flagged web events. To do so, the system will create a modular detection solution capable of identifying users with anomalous behaviours and possibly infected machines, by creating two different specialized detection pipelines. In one hand, one of the processes will focus on finding anomalous users, who present suspicious behavior compared to their peers. On the other hand, the second process will center on finding possibly infected machines that pose some form of threat to the company. Furtherly, each of these pipelines has the specific goals of:

1. Creating an optimized module for extracting, integrating, and processing data for the process. This module should be able to take raw data and convert it into a format that can be used in the model pipeline.

2. Developing an appropriate model pipeline for the respective detection process, taking into consideration each of their main focus. It should effectively receive the processed data and identify anomalies while focusing on minimizing false alarms, ensuring analysts are not overwhelmed with unnecessary workload.

3. Creating a characterization module, which consists of generating automatic reports that try to provide valuable insights for the reason of each detection, in order to help security analysts in their evaluation.

The present project will be developed at Altice Portugal's Department of Cybersecurity (DCY) and it will focus on the detection and characterization of anomalies in a flagged web accesses dataset by resorting to Machine Learning methods. This project will be implemented various

technologies, including Jupyter Notebooks[1], ElasticSearch[2] and Python libraries.

This thesis aims to contribute to Altice Portugal's set of tools for counteracting possible internal attacks or harmful user behavior by providing a new system capable of detecting such behaviors on flagged web accesses. Upon detection, the system alerts the appropriate teams so that quick action can be taken to avoid or reduce financial losses.

1.4 Document Structure

The remainder of this document proceeds as follows. Chapter 2 presents a further description of the problem at hands and explains the several tools and algorithms utilized in this thesis, also including a brief summary on various related works in this area. The requirements and the global view of the system which was implemented are outlined in Chapter 3. Chapter 4 thoroughly describes the dataset and the several exploratory analysis done. Chapter 5 extensively explains the full system that was implemented during this thesis. In Chapter 6, the ensuing results from the previously described system are shown and discussed. Finally, Chapter 7 is composed of the conclusion and possible future work to be done after this thesis.

[1] https://jupyter.org/
[2] https://www.elastic.co/elasticsearch

Chapter 2

Background and Related Work

This chapter provides a comprehensive overview on the foundational concepts used for this project, contextualizing Altice Portugal's active cybersecurity doctrine and anomaly detection. It also presents the description of the many different technologies and methodologies that defined this project. Finally, this chapter reviews previous work done related to this field, providing insights on the different techniques that have been studied and their limitations.

2.1 Active Cybersecurity Doctrine

The cybersecurity strategy that Altice Portugal's Department of Cybersecurity (DCY) follows is represented in Figure 2.1, which is centered on a 360º pro-Active Doctrine, with 5 complementary dimensions, focused on continuous cyber defense and improvement: Active Governance, Active Prevention, Active Protection, Detection & Active Response and Active Recoverability, each one serving its purpose.

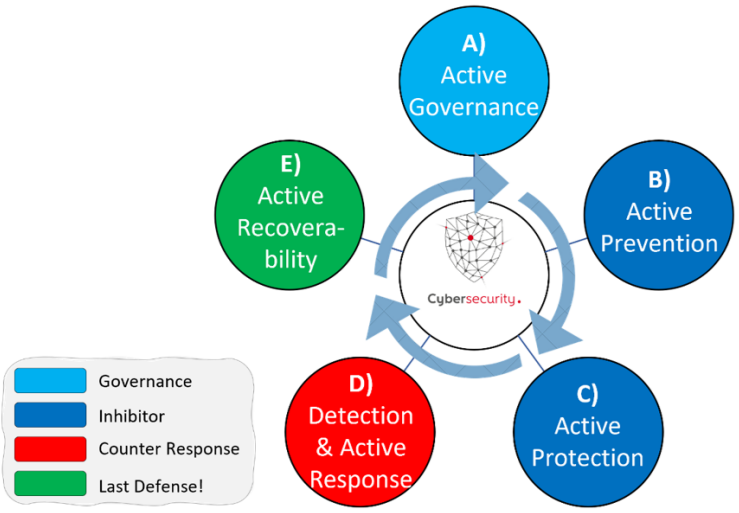


Figure 2.1: Cybersecurity Doctrine in Altice Portugal's Department of Cybersecurity.

Each pillar plays a crucial role in defending the company from attacks:

1. Active Governance: actively and coordinately managing the remaining pillars, ensuring

the necessary support and resources from the Executive Administration and the required compliance level with all Governmental, Regulatory, and Legal Entities.

2. Active Prevention: preventing cyber incidents. It is an end-to-end monitoring program and continuous improvement of the organization's cyber hygiene, while continuously discovering and eliminating vulnerabilities. It must cover all four attack surfaces in an organization: External or Internet-facing, Internal systems, third Party Dependencies, and Users/People.

3. Active Protection: attempts at protecting from cyber incidents materializing—technologies and security protection processes in-depth (DDoS mitigation, Next-Generation Firewalls, etc.).

4. Detection & Active Response: swiftly detecting and responding to incidents that have materialized. With the current state of technology, this is only viable with integrated state-of-the-art solutions for Extended Detection and Response (XDR) / Endpoint Detection and Response (EDR), which are a threat prevention, detection and response platform and an endpoint user monitoring detector, respectively, and User and Entity Behavioral Analytics (UEBA) to detect suspicious behaviors

5. Active Recoverability: in case the attack cannot be evaded, implement active measures to quickly recover and mitigate the impact of the incident, ensuring a prompt restoration of normal operations and minimizing potential damage.

This project falls under the 4th pillar, Detection & Active Response, where we will focus more on the detection part, working on anomaly detection in blocked web accesses of users by using UEBA techniques.

2.2 Anomaly Detection

The definition of anomaly or outlier has been the target of many changes throughout the past decades and the approaches taken by researchers on detecting these outliers has evolved as well, thus extending the definition. Ayadi *et al.* [5] presented twelve different prevalent definitions of outliers proposed by several different authors between the years 1960 to 2011, illustrating the lack of an universal definition of outliers. Given this ambiguity, other researchers can still define an anomaly as a data point that does not conform to a well defined notion of normal behavior, appearing to be inconsistent with the rest of the data [10, 18].

There are many factors and reasons that may lead to the appearance of these data points in cybersecurity, including credit card fraud, cyber intrusion, and malicious activities. These and many other examples may create data that completely leaves the notion of normal behaviour, thus generating an anomaly.

The goal for an anomaly detection algorithm is, in its essence, to determine a region that defines what we can consider to be normal behaviour and anything that exists outside of the region

is regarded as an outlier. However, this task brings many hardships. Among these we include the definition of an often imprecise normal behaviour region (an anomaly lying close to the border can be considered as normal behaviour); malicious users being able to mask themselves by adapting their behaviour into one that resembles normal behaviour; the notion of normal behaviour changing with time; the existence of noise in the data that is being used tending to behave as anomalies, making them difficult to detect and remove; the lack of ground truth in most problems (not knowing which are and which are not true anomalies); rarity and class imbalance; change of meaning of an anomaly from domain to domain; high dimensionality of datasets; and different types of anomalies [10,16,31,36].

Before addressing the problem at hands, one must understand their data and what they are looking for. To know the nature, type of data of each column (e.g., categorical, numerical) and whether there is any ground truth for the data is fundamental. This information is crucial on deciding the appropriate machine learning path to take. Machine learning (ML) is a subfield of artificial intelligence that focuses on developing algorithms and models that allow computers to learn from and make predictions or decisions based on the data that is fed to said algorithms [46]. In cybersecurity, ML's usage has been in constant growth, providing a key to different cyber-attack detection problems, e.g., intrusion or malware detection, and security issues related to critical infrastructures, such as power system security and industrial control systems [17].

In the context of ML, if ground truth exists, then one can use regular supervised machine learning methods (among other possible methods) and it is possible to extract relevant metrics, such as the detection rate and false alarm rate. The detection rate can be defined as the proportion of the detected anomalies accounting for all the anomalies and the false alarm rate is known as the ratio between the number of normal events wrongly categorized as anomalies and the total number of actual normal events [54].

In case there is no ground truth, however, one must rely on unsupervised machine learning (again, among other methods). User/Entity Behaviour Analytics (UEBA) techniques are more common for these models, employing them to group users or entities. Due to the lack of ground truth, the aforementioned metrics cannot be used, leaving us only with indicators that explain how well groups are being formed, e.g., the Silhouette score and Calinski-Harabasz score.

Also, one must take into consideration that different types of anomalies exist. Most commonly, one wants to detect individual instances that are anomalous, known as *point anomalies*. Nevertheless, a point can be considered an anomaly only on a given context e.g., a user accessing sensitive data from an unusual place. These are known as *contextual* or *conditional anomalies*. Finally, there are *collective anomalies*, which are a subset of events that are anomalous as a whole, where if we take an individual event it would not be considered an anomaly. An example of this type of anomalies is the dense subgraphs of fake accounts in social networks [36].

2.2.1 User/Entity Behaviour Analytics (UEBA)

User/Entity Behaviour Analytics (UEBA) is a risk management system to detect suspicious users or insider activities in an organization to avoid significant damage to the organization [23,24,27].

When the idea of giving a name to this set of techniques first surged in 2015, they were initially only called User Behaviour Analytics (UBA) [12]. Later that year, the term "entity" was introduced to demonstrate the capability of vendors to monitor and analyse entities besides users [12, 42].

The analyses consist of a plethora of statistical and machine learning methods to perform anomaly detections on the consistent monitoring of users and other entities simultaneously. They range from dimensional reduction methods to unsupervised machine learning and offer a way to correlate multiple anomalous events that could be related to a single security incident. To do so, they rely mainly on log data from multiple sources to enrich the analyses. These datasets are usually raw big data created from high velocity, heterogeneous streams that are originated from different possible entities (e.g., users, machines, IP addresses) [12, 48, 50].

These techniques came to dethrone the previous most popular technique—Signature-based models (misuse detection)—where the model is built based only on already known attacks and any action that shows the existence of registered attack signatures. In other words, they rely on blacklist signatures or handwritten rules [12, 54]. Although this model appears effective on detecting anomalies, since a detection by the system can be considered highly reliable, it fails on an important aspect: it cannot detect anything beyond what has already been detected or defined in the past, meaning they cannot detect signature-less attacks. In contrast, due to their user profiling properties, UEBA methods manage to see past this problem, being able to detect zero-day attacks (attacks that have not been observed before) and attacks that take a longer time to take effect.

UEBA methods also allow for real-time data analytics (use of data as soon as it enters the system) and threat detection [42]. Depending on the goal of the detector, it is possible to build systems where only a relatively small time window is needed to detect anomalous behavior, instead of resorting to offline algorithms.

Apart from real-time detections, UEBA allows deploying detectors specialized on entities with anomalous behaviour on a longer time windows. Additionally, less human intervention is required to detect threats [35], as human involvement is primarily needed only to confirm whether the system's detection is a genuine threat, thus leading to UEBA's biggest drawback—high false positive rates [24]. Having this in mind, the goal is to continuously develop a system that minimizes false positives.

2.3 Unsupervised Anomaly Detection Algorithms

2.3.1 One-Class Support Vector Machines (OCSVM)

The One-Class Support Vector Machine (OCSVM) [43] emerged as a type of anomaly detection method. They are an unsupervised learning method that, instead of focusing on dividing 2 classes with a hyperplane (regular SVM), tries to identify a region in the feature space that contains the majority of the data (normal instances). Anything outside this region is then considered an anomaly. It also has the ability to use the "kernel trick", where it projects the data in a higher dimension in order, allowing a non-linear decision boundary in the original feature space.

This algorithm also has overfitting measures in place. By using a parameter which represents

an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors, the algorithm controls the trade-off between a tight decision boundary and allowing some points to be outliers. The algorithm is also computationally efficient in terms of scalability. The model, however, has the limitation of the correct choice of kernel parameters, complicating its optimization.

2.3.2 k-Nearest Neighbors (kNN)

Fix and Hodges [15] first introduced a non-parametric method for pattern classification back in 1951. Later, in the year 2000, Ramaswamy *et al.* [37] expanded on this concept and proposed the method that is today known as k-Nearest Neighbor (kNN). This method is a highly efficient partition and distance-based unsupervised algorithm that assigns an anomaly score to each point by ranking each point on the basis of its distance to its k^{th} neighboring point. Hence, the top n points in the ranking will be considered outliers. It does so by partitioning the input data into disjoint subsets, and then pruning any subset which has been determined not to have any outliers, thus making it very efficient.

The algorithm heavily relies on a distance or similarity measure method that scales well with the computational cost of the algorithm. In spite of this, it has some limitations when it comes to its variables, mainly on being dependent of the number of neighbors, k, and the distance threshold, d, for which it classifies a point as an outlier.

2.3.3 Local Outlier Factor (LOF)

The Local Outlier Factor (LOF) [9] is a density-based outlier detection algorithm. LOF calculates the local density of each data point, considering the density of its neighbors. Thus, a data point is considered an anomaly not as a binary property, but as the degree to which the local density significantly differs from its neighbors. To do so, and assuming that anomalies are distant from its neighbors, it uses the k-Nearest Neighbors (kNN) to determine the point's neighbor points and calculates its LOF score. The LOF score consists in the average of the ratio of the local reachability density of a sample and those of its k-nearest neighbors. Thus, a LOF score significantly higher than 1 means that we will have an outlier.

It is shown that LOF is effective in identifying local outliers, making it sensitive to the local structure of the data, while not assuming cluster shapes. This makes it particularly useful for detecting local outliers in datasets with varying densities and shapes of clusters. This algorithm only has one parameter, which is the number of neighbors, k. Thus, its weakness lies in the optimal choice of k.

2.3.4 Isolation Forests (iForests)

Isolation Forests (iForests) [30] are an anomaly isolation algorithm which proves to be highly effective in anomaly detection. Its key idea is that anomalies have a tendency to have less connections to the rest of the data and are, therefore, more prone to being isolated. The method builds

an ensemble of iTrees, where each tree is constructed by randomly selecting a feature and a split point. Having built the trees, the algorithm calculates the anomaly score based on the data point's distance to the root of the tree. If it is closer to the root, it means that the data point was isolated faster, thus having a bigger anomaly score. The overall anomaly score for a data point is derived by aggregating the scores from all trees.

The author showed that this algorithm has near-linear time complexity and outperforms algorithms like LOF in large datasets, converging quickly for a small ensemble size. They are also ideal for high dimensional problems with a big number of irrelevant features. In its nature, the algorithm only possesses 2 parameters: the number of trees and the sub-sampling (partition) size.

2.3.5 Mahalanobis distance

The Mahalanobis distance is "a multi-dimensional generalization of the z-score" [45]. It is given by the following expression:

$$D = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{2.1}$$

where $\mathbf{x}$ is an input, $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix. The Mahalanobis distance shows how many standard deviations one data point is from the mean value of all the instances, accounting for the covariance structure of the data.

Calculating this distance for each point and mapping it to a score allows for anomaly detection (the bigger the score, the more probable it is that the point is an anomaly). To do so, we would only need to use a threshold that defines the existence of the anomaly.

This method has, however, some problems for the purpose of anomaly detection. The classical Mahalanobis distance is based on the sample mean and the sample covariance, which are sensitive to outliers [28]. Hence, Mahalanobis systems which are robust to outliers are proposed by using different estimators to estimate the mean and covariance, such as S estimators [39], the Donoho estimator [13], etc. In this work, we will use the most popular estimator—the Minimum Covariance Determinant (MCD) estimator [40]. This method works by calculating the covariance matrix of various different subsamples of the sample data. The most robust covariance matrix will be the one with the minimum determinant and will, then, be the one used for the Mahalanobis distance, along with the subsample mean [19]. One limitation of this method is, as is with k-Nearest Neighbors, the distance threshold, d, that we consider a point to be anomalous.

2.4 Clustering

Clustering is one of the most common techniques for statistical data analysis and can be considered one of the most important unsupervised learning processes. It deals with discovering the natural groupings of a set of objects considering a cluster as a collection of similar objects, which are dissimilar to objects in other clusters, according to some defined distance measure [32, 33].

These distance measures originate various different types of clustering methods, which include Hierarchical Clustering, Partitional Clustering, Density-Based Clustering, among others [32].

2.4.1 k-Means

k-Means [33] is a partitional (or centroid-based) clustering method which focuses on assigning a point to its closest centroid. Its process consists of 3 simple steps. Firstly, we initiate randomly k centroids and, based on the distance metric, assign each point to the closest centroid, thus forming k clusters. Then, we compute the new centroids as the mean point of each group. Having the new centroids, we reassign every point to its closest (new) centroid and repeat the last 2 steps until we have converged to a stable cluster membership [21].

The determination of an appropriate k is difficult, though. It needs domain knowledge as well as other techniques, such as the elbow technique and cluster evaluation methods such as the Silhouette score and Calinski-Harabsz Index. Besides this, this method usually is used with the Euclidean distance, generating mostly ball-shaped clusters and equally sized (in terms of area), making it weak against irregular shaped clusters and sensitive to outliers.

2.4.2 Spectral Clustering

Spectral Clustering [34] is a method that uses concepts from spectral graph theory to group similar data points together. It starts by building an affinity matrix, which captures how each pair of data points relate to each other, often using techniques like nearest neighbors to do so. This matrix is then broken down into a smaller dimension using its eigenvalues and eigenvectors (spectrum). Using the resulting segmented matrix, the algorithm creates subgraphs (clusters) based on its similarities by using k-Means.

This method is a very strong method which, not only is able to create clusters in irregular shapes, but also is not sensitive to outliers. However, just like k-Means, it is dependent on the number of clusters, k. Therefore, it needs the same treatment as k-Means to find an optimal k.

2.5 Visualization

2.5.1 Visualization Tools

In order to efficiently analyse millions of logs, we must use a data visualization tool that allows us to dynamically visualize, explore, and monitor the data in useful time. Besides this, the abbility to design charts and graphics at-will within a user-friendly interface and customizable dashboard is something that the user must search for in a visualization tool. Among the few platforms that offer these features successfully, Kibana[1], which fulfills all the characteristics that were just described, is the visualization tool that is employed by Altice Portugal and is the one that will be used for this project.

[1] https://www.elastic.co/kibana

Kibana is a powerful open-source dashboard for data visualization, analysis and exploration which integrates very well with ElasticSearch, leveraging its indexing and search capabilities. In this platform, the user is able to create bar, line and scatter plots, and use many other useful analysis tools. Besides this, the user can also dynamically access and query the stored data. An example of a Kibana Dashboard can be seen in Figure 2.2.

Besides Kibana, there are many other existent platforms. Grafana, for example, is a popular open-source platform for monitoring and observability supporting various data sources, including Elasticsearch. One other popular platform is Splunk, which is widely used platform for log management and analysis.

Along with Kibana, Python's *Plotly*[2] and *Matplotlib*[3] libraries were utilized throughout the whole project to plot several graphics.

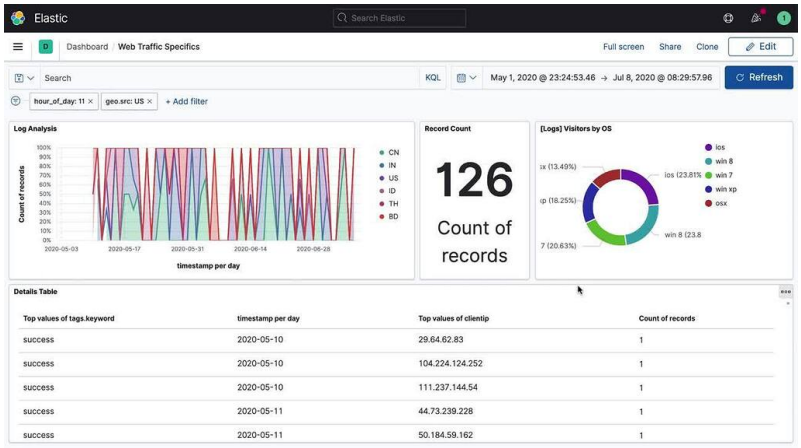


Figure 2.2: Example of a Kibana Dashboard.

2.5.2 Dimensionality Reduction for Data and Cluster Observation

In addition to directly analyzing the data with dashboards, there is often the need to visualize the processed data in a 2D plot (e.g., clusters of user profiles). However, since data typically has hundreds of dimensions, it is impossible to interpret and visualize it, reducing the problem to using dimensionality reduction techniques to make the data easier to visualize.

Principle Component Analysis (PCA)

Principal Component Analysis (PCA) is the most widely used dimensionality reduction technique due to its simplicity. This method works by reducing the number of dimensions while conserving the most information (variance) of the data as possible. To do so, it finds new orthogonal and uncorrelated axes which are linear combinations of the original features called Principal Components (PCs) that consecutively have maximum variance for the data [22].

[2] https://plotly.com/python/
[3] https://matplotlib.org/

For the purpose of observing the data points, each PC is very well characterized by what features made it carry information. The first PC (most informative) should be characterized by the features with most variance. Hence, outliers are most likely to be along this axis. However, in spite of being a possible indicator of an anomaly, this algorithm is not an anomaly detection method, serving mostly to observe the data in 2D for this work.

t-Distributed Stochastic Neighbouring Entities t-SNE)

t-Distributed Stochastic Neighbouring Entities (t-SNE) [49] is a nonlinear dimensionality reduction technique specialized in high-dimensional data visualization in a two or three-dimensional map. To do so, it computes a similarity matrix, whose components are the probability that a point x_i is a neighbor to x_j, using all N dimensions, $p_{i|j}$, as well as the same matrix for its lower-dimensional counterpart with n dimensions, $q_{i|j}$. Theoretically, the probabilities $p_{i|j}$ and $q_{i|j}$ should be the same between two points. Hence, the goal is to find a lower-dimensional data representation that minimizes the mismatch between $p_{i|j}$ and $q_{i|j}$.

It has been shown that this method works well on conserving local (intra-cluster) and global structure (inter-cluster) visualization. The author also shows that the method better interprets the data against seven different techniques. However, this method lacks in two aspects: it can solely be used for visualization since the projected coordinates carry no meaning and it may also require the previous application of a different dimensionality reduction technique.

2.6 Related Work

Anomaly detection related techniques have become a main pillar of cybersecurity systems in recent years due to the variety of attacks that a company can suffer and the sheer power it adds to the enterprise's defense. But, developing such a system is not an easy task—there are multiple ways to develop one and each problem is unique. Consequently, stating that a state-of-the-art method exists would be inaccurate. Nevertheless, with the recent evolution of deep learning, some algorithms have proven to achieve better results than previous ones. However, we are still far from assuming that deep learning can surpass other more traditional methods (e.g., Isolation Forests or One-Class Support Vector Machines). From here on, different methods and techniques of this line of work will be described. For a short summary of the contents of each paper presented in this section, refer to Table 2.1.

2.6.1 UEBA

Many different ways exist to tackle the problem of anomaly detection. One of the most common ways to divide these works is on the usage of user/entity profiles—UEBA . Many authors did not use this approach and just opted for a more general detection system [14, 25, 29, 47, 57]. These methods detect pure anomalies on the data instead of grouping the data by user/entity and detecting anomalies in those spaces. Nowadays, UEBA methods are being more and more used, which leads

to many authors taking these approaches [2,12,16,24,26,27,45,48,50,54]. These works focus on creating a specialized profile for each entity, differing mostly only on the size of the time window which was considered for each profile. The posterior treatment for anomaly detection, however, varies a lot from work to work, and the different paths taken are described in the following sections. Based on these studies, one can conclude that not using UEBA is simpler and has a broader scope on the problem, but it suffers from limited context that can lead to more false alarms and also is worse at insider detection. Using UEBA gives us a better understanding of the anomalies and allows for insider detection; however, it has a more complex implementation, thus being more computationally challenging. The advantages that these techniques bring, although, outweigh the ones from the first approach.

2.6.2 Supervised Learning

There have been several different procedures taken in anomaly detection. Depending on the dataset, one can take either a supervised or an unsupervised approach. The former can only be used if the data has ground truth information, which is very uncommon in a real world dataset. Most of the works that use supervised methods trained their methods on synthetic data (e.g., Lin *et al.* [29]). However, one can dodge the synthetic data if the system itself creates the ground truth. Veeramachaneni *et al.* [50] created a system that used both unsupervised and supervised learning. The system used the unsupervised module to detect anomalies and then communicated the results to security experts which would then provide feedback on these results. The feedback would then be used to train a supervised model which would be able to predict anomalies on the following day.

2.6.3 Dimensionality Reduction

In order to detect the anomalies, there has been great investigation into the various different procedures one can take when there is a lack of ground truth, which are all based on unsupervised methods. There are four main paths one can take: Dimensionality Reduction, Distance methods, and Unsupervised Learning with and without Deep Learning. Within Dimensionality Reduction methods [2, 14, 16, 27, 29, 45, 47, 48, 50, 57], one can either just use it plainly for reducing the dimensionality of the dataset (most cases need this since the data suffers from the curse of dimensionality) or it can be used as an anomaly detection tool in itself. One notable instance of using dimensionality reduction as an anomaly detection tool is the system developed by Shyu *et al.* [47]. The authors developed a PCA based model that uses two different functions to detect anomalies, each with their own objective. The first function utilizes the principal components that account for 50% of the total data variance and the associated variable projections. Its primary aim is to detect regular anomalies within the dataset. Conversely, the second used only the least relevant principal components and the projections as well. This aspect aims to detect anomalies based on the correlation structure of the data. Even though good results were achieved, the usage of dimensionality reduction as a detection tool leads to loss of information, which can make difficult to characterize

anomalies.

2.6.4 Distance Methods

When it comes to Distance methods [24,25,27,29,45,48,57], the techniques always revolve around calculating distances between objects (e.g., a point and a centroid) and in case a data point is over a certain threshold then it should be an anomaly. Pure distance methods are usually based on entity profiling and then calculating an event's distance to the profile. Shashanka *et al.* [45] utilized an adapted Mahalanobis distance algorithm where entities would be profiled using the Singular Value Decomposition (SVD) algorithm and would then have events scored against their base profile. Similarly, Khan *et al.* [24] turns the problem into a time series analysis problem by profiling users' profiles using the STOMP algorithm [56] and the z-normalized Euclidean distance. With this, we would have a time series characterized by magnitude peaks for each user profile and a change in the magnitudes would indicate the existence of anomalies. Lin [27] created a sequential system for user profiling anomaly detection that detected anomalies in three phases: direct detection from previously known attacks, anomaly parameters of subsets of features that surpass a certain threshold, and the deviation values of statistical quantities, e.g., standard deviation. Phases 2 and 3 are distance based methods around user profiles. Even though distance based methods are straightforward, robust and can be computationally efficient, they pose challenges when confronted with different types of data and are quite sensitive to dimensionality, since the notion of distance becomes less meaningful at higher dimensions [1], making it advisable to complement these methods with dimensionality reduction techniques for improved performance.

2.6.5 Unsupervised Learning

Unsupervised learning is the main tool when developing an anomaly detection system. It is composed of a variety of methods—density-based, cluster-based, anomaly based, deep learning, etc. Authors that opted not to use deep learning [2,12,14,16,25,26,54,57] have achieved very good results nevertheless. Lazarevic *et al.* [25] built a detection system while comparing four different methods: Nearest Neighbor, Mahalanobis distance, LOF, and OCSVM. It was observed that the LOF system obtained the best results. Aldairi *et al.* [2] proposed two approaches using Isolation Forests and OCSVMs: a simple model and a trust aware model, which uses the anomaly scores as a feature for the next period. The authors showed that the trust aware model provided better results than the simple model. Zhang [54] created a multimodel system where three different models—Statistics Distribution Model, Hidden Markov Model, and OCSVM—acted on different disjoint subsets of features. The overall result is anomalous if one of these models detects an anomaly. It was found that, individually, each model did not get good results but the joint usage (ensemble) of the three models obtained very good results in terms of the detection and false alarm rates. This means that, even though individual models might not behave well or might just specialize on a type of anomalies, not generalizing well, the ensemble might be capable of obtaining much better results. These methods are very popular in anomaly detection systems due to their power in dis-

covering inherent patterns and wide applicability, but can be computationally intensive when the dataset is very big, which is usually the case with log datasets.

2.6.6 Deep Learning

Along with these models, one can also use unsupervised deep learning [14, 26, 48, 50, 57] to aid on anomaly detection. Deep Learning, or DL, is many times the model of choice or a tool to use along with other techniques. Le *et al.* [26] built an ensemble of an Autoencoder (DL model), an Isolation Forest, a LODA (Lightweight on-line detector of anomalies), and a LOF for various different temporal representations of the data and would combine the anomaly scores to create the ensemble. This proved to improve the detection rate of anomalies. Erfani *et al.* [14] created a system specialized in high dimensional data which would use Deep Belief Networks to effectively reduce the dimensionality and applied an OCSVM on the resulting dataset, which proved to have better performance than other simpler models. Zong *et al.* [57] created an end-to-end system of a Deep Autoencoding Gaussian Mixture Model which used two simultaneous components: one that projects the data into a small dimension with an Autoencoder and another that evaluates the probability of the sample being an anomaly with a Gaussian Mixture Model (GMM). Like this, the system minimizes the Autoencoder input reconstruction error and the sample energy. This proved to obtain better results than considering the problem in a dimensionality reduction combined to anomaly prediction separation. Tuor *et al.* [48] compared both Deep Neural Networks and Long short-term memory models to a baseline of three normal models, while testing whether a probability reconstruction of the input (same time step) or the prediction of the following input would be a better approach. It was shown that the same time step approach was better and that the deep learning models outperformed the baseline models. Within the baseline models, the Isolation Forest gave the best results. Deep Learning clearly shows very good promise. However, besides being computationally very intensive, they often require large amounts of data for training. Having a scarcity of data, along with the rarity of anomalies (class imbalance) can lead to challenges in effectively training deep learning models to recognize anomalies. Besides this, they are "black boxes", meaning that one does not know what happens during the training phase and interpreting the decisions made by these models can be challenging.

2.6.7 Real Datasets

Regardless of the good results that the authors present, the vast majority of the investigative work done nowadays utilizes synthetic datasets with ground truth information to check the results, with only a small percentage using real world datasets to test the models [12, 14, 16, 25, 45, 50, 54]. While showing good results, one must take into consideration that the dataset of the present project is a real dataset with no ground truth information. The lack of investigation with real datasets that focuses on high dimensionality leads to an unclear answer on what should be the best approach and algorithms for addressing the problem. Still, some works have used real datasets. Datta *et al.* [12] compared four different models on clickstream data and used unsupervised models metrics, e.g.,

Silhouette Score, to identify which models behaved best. The two best models would then be used in an ensemble to improve the results. Guerra [16] created an anomaly detection system on an Altice Portugal's dataset regarding database accesses. The author created features that included the whole time window and time windows within the full time window in order to investigate anomalies both on the user spectrum and user's historical spectrum. Besides this, the author used unsupervised learning to create an ensemble of models to detect anomalies, training them on injected anomalies, creating a very effective system. However, not all detected anomalies by the system were insiders, being unable to distinguish between malicious and legitimate users. The present works differs this one in terms of both the type of anomaly that is being detected and in the detection pipeline itself, although sharing some algorithms and techniques.

This study aims to leverage the maximum knowledge derived from the described works to create an insider detection system. Given the present real dataset, the system will base itself on UEBA techniques by using unsupervised learning and distance-based models in ensembles, as these methods have demonstrated superior performance compared to single model pipelines, trained on injected anomalies to solve the problem of lack of ground truth presented by works who used real datasets to detect suspicious users. The infected machine detector will not use the ensembles, as the pipeline the system takes allows it to be independent from unsupervised modeling.

Table 2.1: Overview on each of the papers presented.

Ref.	Year	Dataset	Methods	Main Focus
[2]	2019	CERT R4.2		ITD
[12]	2021	Clickstream for online shop		ITD
[14]	2016	Various (6 real and 2 synthetic)		High-Dimension Anomaly Detection
[16]	2020	Altice Portugal's Database Accesses		ITD
[24]	2022	CMU's Insider Threat Dataset		ITD
[25]	2003	1998 DARPA Intrusion Detection Evaluation Dataset		Comparison of Methods in ITD
[26]	2021	CERT R4.2, CERT R6.2		Effect of Temporal Representations on UML
[27]	2015	CMU-CERT		ITD
[29]	2015	KDD-Cup 1999		Feature Representation for Anomaly Detection
[45]	2016	Niara Security's internal network traffic		Platform building for incident investigation
[47]	2003	KDD-Cup 1999		ITD
[48]	2017	CERT R6.2		ITD
[50]	2016	Enterprise's Dataset with reported attacks		Tool Building for ITD
[54]	2017	Wikipedia Accesses, FuzzDB		Model Ensemble for ITD
[57]	2018	KDD-Cup 1999, KDD-Cup Rev, Thyroid, Arrythmia		New UML Method for Anomaly Detection

Methods:  UEBA,  Dimensionality Reduction,  Deep Learning,  Distances,  Unsupervised ML,  Supervised ML,  Real Dataset

Main Focus: ITD - Insider Threat Detection, UML - Unsupervised Machine Learning

Chapter 3

Solution Design

This chapter aims at explaining the global pipeline created to tackle the problem this project presents. Along with the several different requisites that an anomaly detector requires, the path that the data took to detect anomalies will be described.

3.1 System Requirements

This project is integrated in the Cybersecurity Department (DCY) at Altice Portugal and aims at building an anomaly detection system on a dataset regarding the flagged web accesses of the company's employees. Several different technologies were utilized to complete this project, including a Python integrated development environment, such as Jupyter Notebooks or PyCharm, distributed analytics engines for data storage, e.g., ElasticSearch or Splunk, and data visualization and analytics tools, like Kibana or Grafana. The whole process was also assisted by security experts, SOC analysts and data scientists. With their aid, several different key points that have to be carefully addressed in order to develop the system were determined for this project. Some of these include:

- The system must be flexible and robust to changes in data. For example, some columns in the dataset are always subject to additions of new custom made values, for which the system must be able to handle.
- The system must be adaptable in time, meaning that it has to be able to adapt itself to the ever-changing behavior of the considered entity of the UEBA process. In other words, the system must be capable of adapting itself to two different cases—an entity that used to present itself as normally behaved is now anomalous and a previously anomalous entity is no longer anomalous. By consequence, the system must prioritize only entities who are anomalous in a recent time frame.
- The system must be impervious to different types of anomalies, that is, it must be able to deal with point anomalies, contextual anomalies and collective anomalies (refer to Section 2.2).
- When it comes to the User Detector, the system must be insensitive to peaks. In this work, we are interested only in building a system that can detect users who were consistently

anomalous throughout a relatively long period of time.

- The system must be fine-tuned to filter the most False Positives as possible, since, according to the security experts, it is better to let some anomalies pass than to report every single one due to the small amount of reports that the security team can effectively analyse.
- The system must share insights on why the detection occurred, illustrating the reason of detection and aiding in location and visualization to the analyst who will further investigate the anomaly.

All of these points are of utmost importance to the detectors. Taking into consideration all of these key issues, an efficient design that can take care of the various limitations that an anomaly detection system has is not an easy task. This dissertation aims at developing the best system possible that can solve each limitation presented. In this following section, the global conceptual pipeline for the system is presented.

3.2 System Architecture

The global architecture of this project from data retrieval to results analysis can be observed in Figure 3.1. Both the User Detector (UD) and the Infected Machine Detector (IMD) follow this system. This pipeline is composed of eight different components; however, only the components inside the Anomaly Detection System are within the scope of the project.

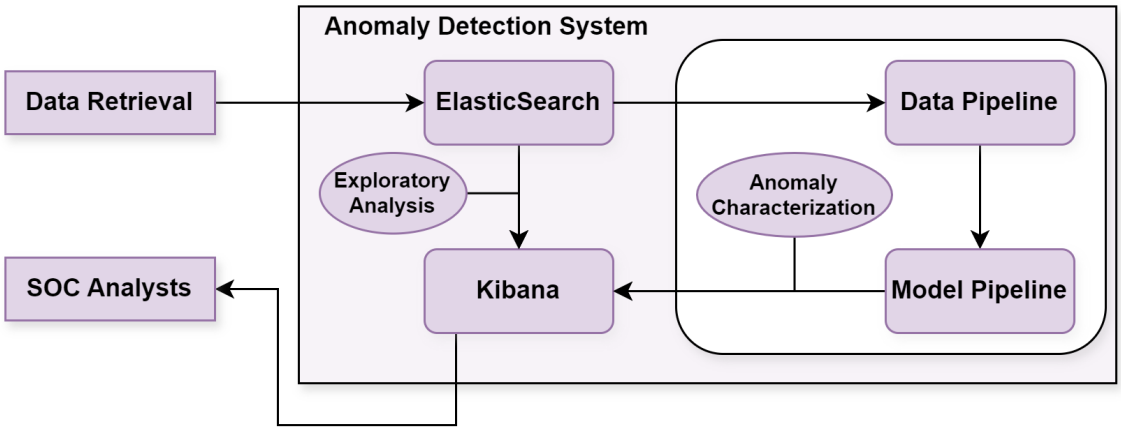


Figure 3.1: Global pipeline for this project, representing both the UD and IMD. The Data Retrieval and SOC Analysts phases are out of the scope of this thesis.

3.2.1 Data Retrieval

The system first starts with the data retrieval. This section outlines the step-by-step process that data undergoes from user web access to its integration into the data storage platform. The system is composed of an arbitrary number of internal collaborators using internal machines (desktops, notebooks, mobile devices) attempting to access any website on the internet (i.e., external servers).

A finite set of monitoring machines and a Proxy web monitor flag suspicious accesses. Following the flagging, Zscaler[1] , a cloud proxy-based service, records the access data associated with the marked attempt. This data is then stored on designated storage servers. From this point on, the data enters a pre-treatment phase which consists in the application of some filters to the data, enhancing the overall quality and relevance of the data. Due to security reasons, these filters cannot be publicly revealed. To finalize the process, the data is injected into the data storage platform, which can be accessed within the data visualization tool dashboard.

3.2.2 Exploratory Analysis

Having the data accessible in the visualization tool allows for Exploratory Analysis. This analysis, along with the aid of security experts, helps on deciding the best features for the project, along with what treatment they must have so they can be correctly transformed in the following phase.

3.2.3 Data Pipeline

Following the data exploration, it will then be extracted, integrated, processed and fully transformed into a format that the Model Pipeline can then read. The two detectors start to differ here. The main difference is that the UD uses Users while the IMD connects the user to a machine and uses that machine for the analysis instead. Also, new metrics that translate the degree of infection are defined and created for the IMD. The data is read and transformed into a daily activity summary which is then transformed into an entity profile, by grouping a set of days together. Each of these profiles, be it user or machine, is characterized by various statistical features that fully describe the behavior of the entity during the considered time window. The IMD's profiles, however, group the daily data both based on historical data (older section of time window) and recent data (the remaining part of the time window), while the UD does only a single data grouping considering all of the time window.

3.2.4 Model Pipeline

After transforming the data, the UD and IMD each follow its own model pipeline, shown in Figures 5.8 and 5.11, respectively. On the one hand, the UD takes advantage of statistical analysis, cluster techniques and model ensembles in order to detect the most anomalous users among the dataset, while keeping lesser dangerous users under surveillance with daily outlier detections. On the other hand, the IMD focuses on the metrics specialized on translating the infection and the comparison between the historical profile and recent profile.

It is important to notice that, in order to solve the problem of having users changing their behavior over time, this system will be implemented over a sliding window configuration. This means that, for example, if we used three months of data for the UD and we detected anomalies at the start of every month, then, given a detection at the beginning of September, we would only

[1] https://www.zscaler.com/

use data up to the beginning of June. A following detection window would be July - October. This allows the system to adapt to the dynamic behavior of users in time.

3.2.5 Anomaly Characterization

Finally, each detection must be justified so that the SOC analysts can understand why the detector flagged the entity as anomalous. Therefore, each detection is fully characterized with a detection score and other metrics that attempt to explain the reason of detection. These results are visualized through a report that includes a link for a data analytics dashboard which includes various tables and time series characterizing the entity. The reports are then sent to SOC Analysts for further analysis on the severity of the detection and whether it is a False Positive or not (although the communication pipeline has not yet been implemented).

Chapter 4

Data Analysis

Before presenting the proposed solution to the project, one must try to understand the data and its intricacies so the present problem does not get blindly tackled. Therefore, an initial phase of a comprehensive data exploration and analysis to the data which will be used must be done.

In this chapter, the aforementioned exploration and analysis which were done will be described. This part consists of the characterization, description and initial remarks made on a previously unstudied dataset which were taken from a thorough analysis on the data through several different plots, tables, etc., on the various existing features. These analyses were created by taking advantage of Kibana's interactive dashboard using data stored in an ElasticSearch Index. This analysis was made using only three months of data. Although, when the time window was increased, the inner data relations were observed to be maintained. This stage is crucial for the development of the system for the reason that it makes possible the detection of several problems within the data and the simplification of pre-processing steps.

4.1 Data Characterization

To develop the detection system at hands, the web insight logs[1] dataset was used. This dataset is comprised of a log-based data belonging to a real enterprise that is offered by Zscaler's security services and is updated in real-time, getting new logs every 15 minutes. The data consists of the flagged web accesses of the company's internal collaborators and is characterized by a set of 37 features, from which 18 were considered to be relevant (with the aid of security experts) for the system or for the exploratory analysis. The fields fully describe the reason for why the access was flagged, including the policy that took action, the URL information, the HTTP request information, IPs and threat classification. The full set of relevant features along with a brief description of each one of them is presented in Table 4.1.

There were some features which were ultimately put aside. Some columns were referring to ElasticSearch's network—mainly identifiers. Other columns referred to the transaction times of the clients and servers, which, according to security experts, are not relevant for this work. There are also many other fields regarding the time of the event, but we only need the field *ts* (timestamp)

[1] https://help.zscaler.com/zia/about-insights-logs

for that purpose. In addition to these, there were 2 columns—ReceivedBytes and SentBytes, which represent the size of the HTTP request requested and sent to the destination server, respectively—which could not be used due to an inner problem with the collection of the data. Among the fields presented in the table, some ended up not being used too, e.g., Location; however, they were still important for the exploratory analysis.

When it comes to the amount of data that was extracted, depending on the phase of this project, different sizes were considered. For the initial Exploratory Analysis, only three months were

Table 4.1: Set of relevant features for the analysis with respective brief description. Some fields are not shown because they are not relevant for this work.

Field	**Description**
Agent	String providing server information, including application and operating system details, to optimize content delivery.
ClientExternalIP	Public IP used for internet connectivity, acting as a gateway connection.
ClientIP	Unique device IP within the local network, assigned by the router.
FileName	Name of the downloaded or uploaded file, applicable in file transfer scenarios.
Location	Origin of the transaction; marked as "remote user" if the service cannot define the location.
PolicyAction	Outcome of the service action: allow, block, or user warning.
ReceivedBytes	Amount of data, in bytes, returned by the destination web server for each HTTP Request.
ReferrerURL	The website from which the user clicked a link to access the specified URL.
Request	Type of request made, such as GET, POST, CONNECT, etc.
Response	Server's response status, indicating success (200 OK), error (404 Not Found), etc.
SentBytes	Size of the HTTP Request sent to the destination web server, in bytes.
ServerIP	IP address of the destination server.
ThreatName	Name assigned to the detected threat, if applicable.
ThreatSuperCategory	Higher-level category of the detected threat, if applicable.
ts	Timestamp in date format, equivalent to the Event Time.
URL	The web address of the site involved in the transaction.
URLCategory	Categorization of the URL, such as AI, Auctions, Music, etc.
User	Username or identifier of the user who accessed the site.

considered, since, at the time, there was not any more data available. For the Modeling phase, about four and a half months of data were mostly used, totalling in over 48 million logs, having been used to train the models and tune the results. However, recall that this system is based on a sliding window configuration; thus, there is no fixed dataset that is being used.

4.2 Exploratory Analysis

Within Altice Portugal, prior to starting the project, the web insight logs had yet to be studied, meaning that no analyst had even looked at the data before. Having this lack of a previous data exploration, this initial section had to be detailed and thorough. This work started around September 2023, in which only three months of useful data were available, up to June 27th 2023, in view of the fact that earlier than that day there was a large time window with no data (and our analysis must be based off recent data that represents the current reality). Therefore, about three months of data were used for the Exploratory Analysis.

The Exploratory Analysis of a dataset consists of the several preliminary analysis made to the data in order to answer to a set of questions which will aid on the data processing in the following phase. For this project, 5 research questions had to be answered initially:

1. What was the quality of the data in terms of missing values and duplicated lines? Also, what are the possible values of each field and their data type? (Section 4.2.1)
2. How does the data behave in terms of univariate behaviour and correlated variables? (Section 4.2.2)
3. What features are most appropriate for the project? And within an important feature, are all of its values relevant? (Section 4.2.3)
4. What problems inherent to the data must be taken into account? (Section 4.2.4)
5. What could be considered suspicious behavior in terms of users and machines? In other words, what is the expected normal behavior? (Section 4.2.5)

In order to answer to each of these questions, Kibana and Python Jupyter Notebooks were used. Using Kibana, each feature was analysed in terms of missing values, unique values and correlation with other fields. With Python, more complex analyses were performed in terms of grouping data, transforming, etc.

4.2.1 Data Quality

For a simple first step, a superficial overview of the data and each field was performed. It was observed that the data possessed no missing values, having its own custom characterizations already whenever there was no information. Mainly, whenever there was a missing value, the data processing prior to its insertion in ElasticSearch just filled it with a None string.

Table 4.2: Number of unique values and respective data type of each field of the dataset. The ts field's Unique Count was excluded since it carries no meaning. The Unique Count is not a fixed value for most of the fields.

Field	Unique Count	Data Type
Agent	378	String
ClientExternalIP	2718	String
ClientIP	19791	String
FileName	1530	String
Location	20	Categorical
PolicyAction	20	Categorical
ReceivedBytes	706648	String
ReferrerURL	2821	String
Request	7	Categorical
Response	13	Categorical
SentBytes	28850	String
ServerIP	2260	String
ThreatName	298	String
ThreatSuperCategory	4	Categorical
ts	-	Time
URL	261069	String
URLCategory	48	Categorical
User	5826	String

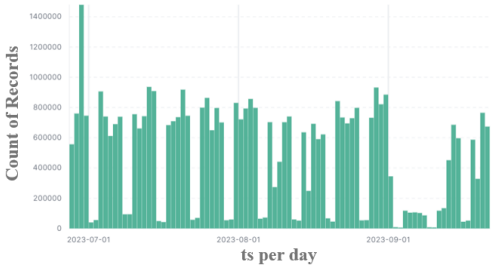
The number of unique values and type of data of each one of the fields is available in Table 4.2. Note that the field *ts*'s Unique Count is absent since it carries no meaning. As one can see, most fields are in *string* format, including the Byte features, which complicated any further analysis made to them. Besides this, the Unique Count for most of the fields is not a fixed value. The value which is presented only refers to the 3 months of data considered. To evaluate these fields, some data had to be extracted into a Python environment and turned into *int* format. Still, a request to improve the data types in the ETL (Extract, Transform, Load) was made in order to correct the wrong data types.

Regarding duplicated lines in the data, although there were no direct duplicates, there were 2 problems. One was with the insertion of data—during some weeks, the data was wrongly inserted, leading to some index duplicates and a great increase of the number of logs. This problem was later solved and is now no longer present in more recent data. The second problem deals with the

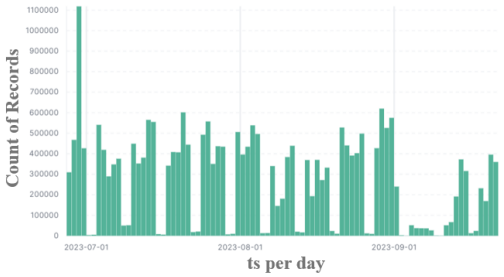
way the data is collected and will be further explained in Question 4.

4.2.2 Univariate behavior and correlated features

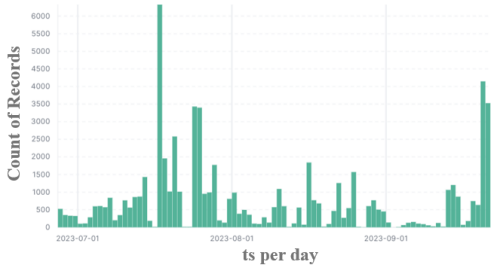
The next topic of study is relevant distributions of variables in time. In Figure 4.1, three of the most relevant time series are shown. Figure 4.1a is the unfiltered version of the data through time, showing the overall data distribution. Figure 4.1b shows a very similar distribution; however, the data was filtered only to include logs whose PolicyAction was not 'Allowed'. We see a decrease in volume, but not as big as expected. This shows that most logs refer to blocked accesses. Finally, Figure 4.1c shows a filter regarding ThreatSuperCategory (TSC), where we only consider it to be different than 'None'. We see a great decrease in volume, meaning that most logs have their ThreatSuperCategory as None.



(a) Web insight log records per day without any data filter.



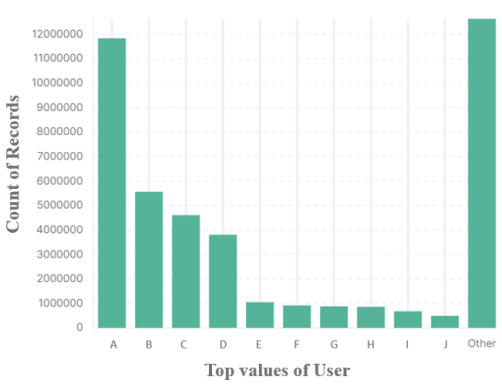
(b) Web insight log records per day without logs with PolicyAction as Allowed.



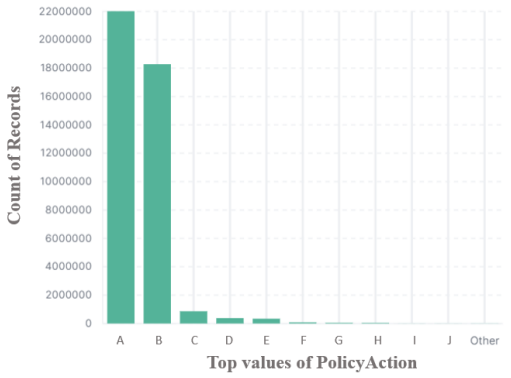
(c) Web insight log records per day only considering logs with their ThreatSuperCategory different than None.

Figure 4.1: Several time series representing the logs throughout the three months which were considered.

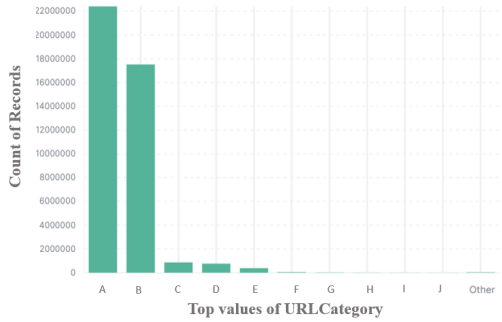
Now, refer to Figure 4.2, which shows four different plots regarding the number of logs per User, TSC, URLCategory and PolicyAction. Due to privacy reasons, the real names of the values will be kept private. In Figure 4.2a, one can see a big unbalance. To understand the unbalance, one must know first that the users are all in an e-mail format. However, sometimes, under some conditions, the log just registers the Location as the User, which is a VPN, creating this unbalance. Moving to the other three figures, it is observable that there is always one or two values that dominate the distributions. However, they might not be relevant to the analysis, which will be further analyzed in Question 3. This relevance will highly affect the volume of data which will be utilized during the pre-processing.



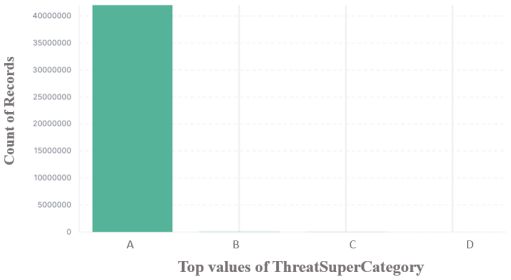
(a) Distribution of logs per Users.



(b) Distribution of logs per PolicyAction.



(c) Distribution of logs per URLCategory.



(d) Distribution of logs per ThreatSuperCategory.

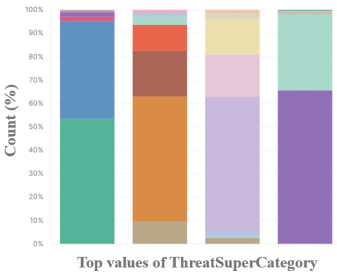
Figure 4.2: Log Distributions per User, PolicyAction, URLCategory and ThreatSuperCategory.

After studying the features individually, several correlations between variables were examined. For this thesis, only the most important correlations found will be presented. In Figure 4.3, three different plots correlating the ThreatSuperCategory, URLCategory and PolicyAction can be seen. The plots show stacked columns with the `%count` that a certain value contributes to that feature, depending on which feature the original field is being broken in. In other words, e.g., Figure 4.3a shows how much each value of PolicyAction, contributed to a certain value of ThreatSuperCategory (each TSC value is a column). This figure reveals 2 values of the TSC for which there is only one corresponding PolicyAction value, with one of the TSC's values corresponding to most of the other possible PolicyAction values. Figure 4.3b shows some more correspondence between the threats and URLCategory, although it seems that each possible category value corresponds only to one TSC value. These 2 figures show that the ThreatSuperCategory was almost disjoint from the PolicyAction and URLCategory, demonstrating a low correlation. Finally, Figure 4.3c shows that there are many PolicyAction values with only one corresponding URLCategory's value, meaning that these fields are highly correlated.

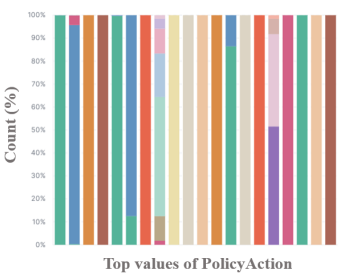
4.2.3 Relevant features and respective values

One of the most important parts of any Data Science project is to clear out any noise the data has, which is irrelevant data for the analyst, acting as a hindrance to the analysis [10], that would

(a) ThreatSuperCategory in function of PolicyAction.



(b) ThreatSuperCategory in function of URLCategory.



(c) PolicyAction in function of URLCategory.

Figure 4.3: Several relationships between the ThreatSuperCategory, URLCategory and PolicyAction shown on a percentage count stacked vertical column configuration—shows how much each value of, e.g., PolicyAction contributed to each value of ThreatSuperCategory.

negatively affect the following phases. Although not unique to this section, a good Exploratory Analysis should be able to give insights on which columns would be noisy on the dataset, or not useful at all. Besides this, given a relevant categorical feature, it is highly possible that not all of its possible values are relevant for this project. Hence, a good set of features must be decided on and studied to find if all of their possible values are relevant for this project.

Starting with the relevant fields, considering the features presented in Table 4.1, many were excluded. Firstly, the Location field is not necessary since it is, technically, only an extra VPN location information. As mentioned before, the ReceivedBytes and SentBytes fields were also excluded due to data collection issues. Hence, the Request field was also dropped, since it would be more useful to use it to correlate the Byte fields with a download or upload. Furthermore, retaining the Request would generate noisy data. For example, a 'Get' value without the associated number of received bytes would lack context, making it unclear whether it represents a simple request or a file download by the user. Furthermore, the Response field was not considered since it was concluded that it would not be relevant considering an insider detection context and having information on the response of the connection would only create noise. Also, the ThreatName field was ultimately not considered, since ThreatSuperCategory presents a more compact option to evaluate threats. Finally, the ReferrerURL and ServerIP fields were both dropped as suggestions of security experts, due to irrelevance for the provided context.

Moving to relevant values for the features, with the aid of Altice Portugal's security experts, many filters were made possible. The filters which were created concern the PolicyAction[2], URLCategory[3] and User features. The first two are both categorical features with many possible values. However, in an insider detection context, not all of these values are relevant. According to the experts, the values of interest are only those who translate a posing threat to the company. So, for example, if the URLCategory of a log is 'Travel', it would not be of interest. Nevertheless, other categories, e.g., 'Malicious Content', are relevant for the project. Regarding the User field, our focus is solely on users identified by their email addresses, not on VPNs. VPNs encompass a

[2] https://help.zscaler.com/zia/policy-reasons
[3] https://help.zscaler.com/zia/about-url-categories

diverse group of users who cannot be individually identified. For effective entity-based analysis, it is essential to concentrate on identifiable users. In reality, the biggest bars in Figure 4.2a are mostly VPN related. For the presented project context, we are more interested in human or infected behavior. Therefore, VPN user logs are not as relevant and must be removed from the data.

4.2.4 Inherent data problems

A very important part of exploring the data is to understand the data and know its limitations. Having knowledge on the inner problems and challenges in a dataset is a crucial step in determining an appropriate data transformation pipeline. One of these problems was, as mentioned before, the Byte columns seeming to have data collected incorrectly.

Besides this, one study done on the data, in particular, stood out as it revealed a strange property. The study started by extracting a smaller dataset of around 1 week into a Jupyter Notebook. On it, the average time between flagged accesses for each user was investigated, taking into account users that only made 1 access and the number of different URLs inside this average. Some results can be found in Table 4.3. The results show something that generates immediate questions: there are multiple users with multiple accesses whose mean times are equal to 0 seconds (with multiple different URLs), which is humanly impossible. The reason was found to be that during attempts to access certain websites, encountering access restrictions prompted the website to make multiple connection attempts, either by reaching out to different servers or persistently trying to display its contents. Also, dynamic pages can generate multiple HTTP requests in a single page. These behaviors generate numerous access attempts. Depending on the website, different URLs could be attempted, thus explaining the different number of unique URLs. This meant that a careful treatment to the data had to be done in order to correctly estimate the real number of accesses that the user made in a certain time span.

Table 4.3: Some examples on the mean interval between consecutive flagged web accesses. In the table, it is also considered the number of accesses made and the number of unique URLs within those accesses.

User	Mean (sec)	#Accesses	Unique URLs
User A	0	15	3
User B	0	40	8
...	...	...	...
User M	58.98	65	2

4.2.5 Suspicious behavior vs Normal behavior

After completing the comprehensive Exploratory Analysis, we were able to establish criteria for identifying anomalous behavior. In consultation with security experts, one primary type of suspi-

cious activity for initial investigation was identified—highly volumetric behavior. This possibility stems from previous works which also considered this use case, such as Guerra's work [16]. This behavior can be viewed in two different ways, given the amount of accesses that the user possesses:

- Moderately High Volume: this could indicate a user who is generating more traffic than expected, suggesting that the user needs additional training or could be a potential insider.
- Impossibly High Volume: this amount of accesses is highly unlikely to be generated by a human, which might indicate that there is a machine that might be infected with malware.

Finding direct examples of anomalous behavior in the dataset through simple analyses is very hard, considering all the different filters that must be taken into consideration; therefore only this possibility is considered for the meantime.

4.3 Discussion

During the Exploratory Analysis phase of this project, several important points were taken that were crucial for the development of the pipelines regarding the pre-processing and modeling of the data. Each question helped on building a global view of the quality and state of the web insight logs. The essential ideas taken from each question were:

1. The data had no missing values nor duplicated lines. Also, most fields were strings or categorical;
2. A good portion of logs comes from VPN-like usernames. Also, the URLCategory, PolicyAction and ThreatSuperCategory fields share very clear correlations. While the URLCategory and PolicyAction are very correlated, the TSC is almost disjoint from these two;
3. In order to remove noisy data, most fields were not considered for not being relevant. Also, both the PolicyAction and URLCategory fields had their values filtered to only consider values considered to be indicative of danger. For the User field, only users who are not VPNs were kept;
4. The data had inherent collection problems on the Byte fields as well as the volume of logs itself, since there are multiple users with multiple accesses whose mean times between consecutive accesses are equal to 0 seconds;
5. The expected anomalous behaviors are those where a user has either a moderately or impossibly high amount of logs, where most of the logs are referent to relevant values of the PolicyAction and URLCategory fields.

All the different insights taken from this step were taken into account when building the detection system for the suspicious users and the infected machines, enriching the pipelines, greatly optimizing the quality of the data and helping in finding the best balance between information refining and computational cost. This process is described in detail in the next chapter.

Chapter 5

Implementation

In this chapter, the building process of both the User Detector (UD) and Infected Machine Detector (IMD) will be explained based on the architecture presented in Chapter 3. The process behind each component will be thoroughly described, as well as what led to their creation. As mentioned before, Python was the main tool used to build the systems, along with its various libraries: *elasticsearch* and *mysql* for data extraction; *numpy*[1], *pandas*[2], *scipy*[3], among others, for data processing; *sklearn*[4] and *pyod*[5] for anomaly detection models; *pickle*[6] for model serialization and saving; and *plotly*[7] and *matplotlib*[8] for data visualization. Kibana, a powerful visualization tool, was also used throughout the whole project, not only for the initial data analysis but also to evaluate models' results.

5.1 Data Pipeline

After the Exploratory Analysis, and following Figure 3.1, the process starts with the Data Pipeline. This section encompasses all the transformations, integration and processing steps that the data went through in order to become readable by the machine learning pipelines used later on. The goal is to turn the data, which consists of string and categorical fields, into a group of numerical features that fully characterize a user or a machine during the whole time window. The Data Pipeline was created taking into consideration all the answers found in the Exploratory Analysis and can be seen in Figure 5.1. As showed in the diagram, the data goes through four different phases: the extraction, the enrichment, the correction, and the feature extraction phase. Firstly, the pipeline will be explained with respect to the UD. Subsequently, the IMD branches will be discussed.

The process starts by the data being extracted from the ElasticSearch index into a Python's

[1] https://numpy.org/
[2] https://pandas.pydata.org/
[3] https://scipy.org/
[4] https://scikit-learn.org/stable/
[5] https://pyod.readthedocs.io/en/latest/
[6] https://docs.python.org/3/library/pickle.html
[7] https://plotly.com/python/
[8] https://matplotlib.org/

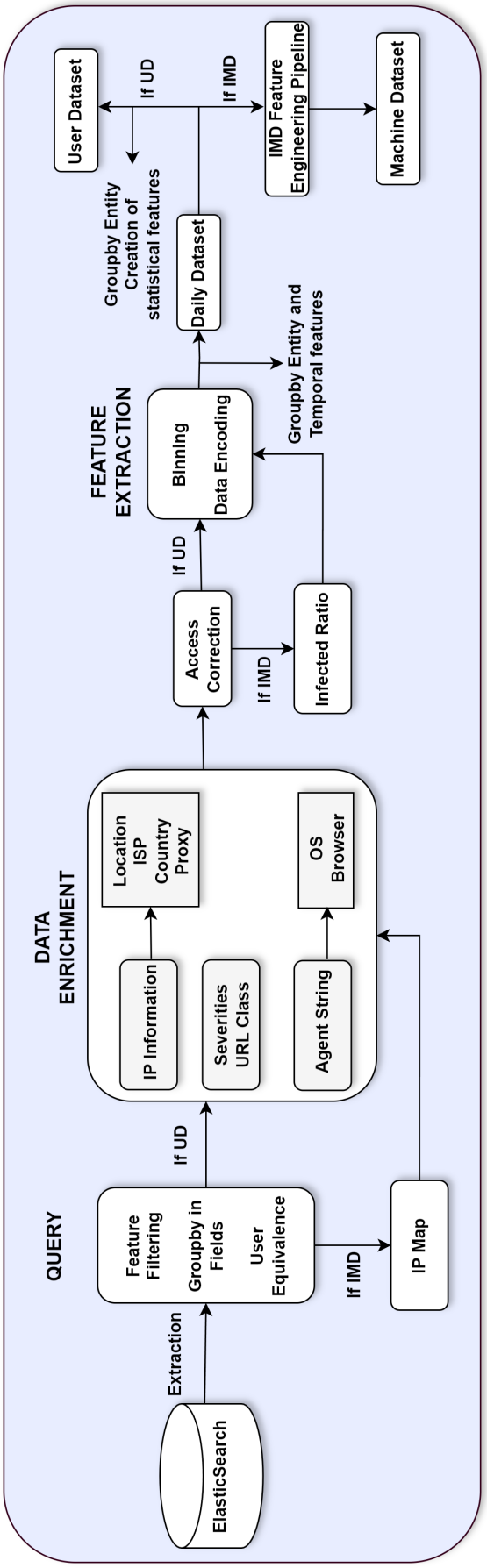


Figure 5.1: Data Pipeline Diagram.

Jupyter Notebook environment by using the *elasticsearch* library through ElasticSearch structured query language (SQL). The query that extracts the data consists of three different functions: Feature Filtering, Group by operations and User Equivalence. Firstly, due to the sheer volume of the data, one must be expecting the existence of noise; therefore, the data must be filtered in order to clear some noise and to ease on query time. Having the information from Question 3 available, the following filters were applied:

- ThreatSuperCategory exists;
- PolicyAction is relevant for analysis;
- URLCategory is relevant for analysis;
- User is not VPN-based.

This query was optimized by using the insight from Figure 4.3. Another measure used to deal with the big volume of data was to use group by operations. Grouping by the fields of biggest interest allows for faster querying and a more compact dataset. The fields which the data was grouped by were the User, a date truncation by the second that will be explained further ahead, URL, ThreatSuperCategory, existence of Files, PolicyAction, URLCategory, ClientExternalIP, ClientIP, and the Agent string. Finally, we have the user equivalence section, which concerns the fact that there are different user e-mails that referred to the same person. So, the e-mails that belong to the same person were determined using an available user database and the information was then joined in the query to correctly assign the accesses to the users.

After the data extraction, the data was enriched by extracting information from existing features. Using internal databases, both the ClientIP and ClientExternalIP are used to determine the Location, ISP, Country and Proxy of the user. From the agent string, the OS and the Browser that were used for the access were extracted using the *user-agents*[9] library. Also, the severities for both the PolicyAction and URLCategory were merged in the data. The severities are either 0 if not critical, 1 if moderate, or 2 if highly critical. This categorization was validated by the security experts. Finally, in order to ease the feature extraction step that would further ahead be done, the URL category class, whose information was taken directly from the web proxy's online documentation, was added to the dataset.

Following the enrichment, the problem regarding the number of accesses had to be dealt with. The data, as shown in the Exploratory Analysis, had issues with the amount of times a website tried to connect to the server, thus generating multiple events in the data when in reality we had only one real access. To solve this, a 2-part method was developed, which is illustrated in Figure 5.2. The first part was, as mentioned before, bucketing the data when grouping by in the query, using truncations by the second, meaning that any equal access within a second is considered the same access. The second part consisted in a URL treatment and further time bucketing scheme. The URLs presented some challenges that had to be cared for. For example, some URLs only

[9] https://pypi.org/project/user-agents/

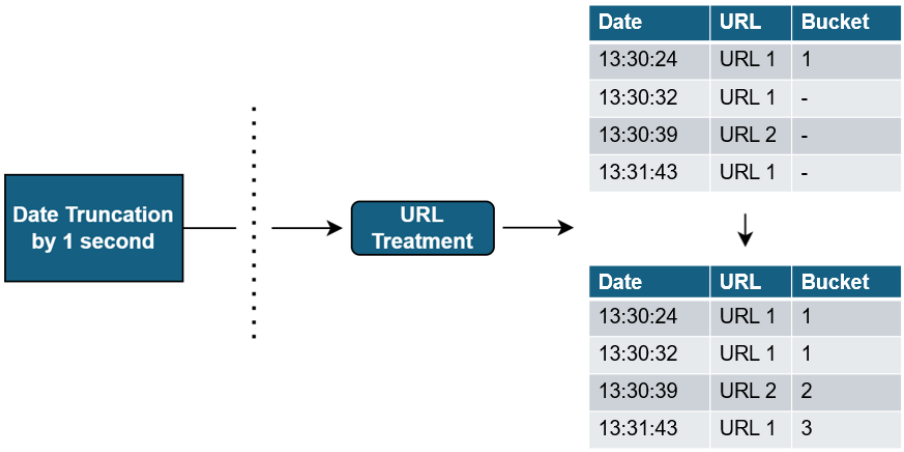


Figure 5.2: Access Correction pipeline. After truncating the data by the second during the extraction, the URLs were treated and then buckets of 10 seconds were created—given a user, any access within 10 seconds of each other with the same URL is considered to be the same access.

added a port number or a subdomain change. Following the URL treatment, the logs were put into 10 second buckets. In other words, given an instance, if unbucketed then a new bucket is created. It then checks any other logs with the same URL within the considered time window and puts them in the same group. This process is repeated until all logs have been grouped into a time bucket. Each time bucket, thus, represents a real access by a user. Of course, this method is not bulletproof; however, it is adequate, as the results of the detectors showed so, so that we have an approximate reality of how many accesses each user had.

Following the time bucket creation, the next challenge was to deal with the multi-categorical features. Both data binning and one-hot encoding (converting into a binary format) operations were applied to the features in order to turn them into a numerical format:

- The PolicyAction field was binned into Passed, Blocked (relevant action) and Blocked (irrelevant action). The relevance of the action is determined if the action had its severity different than 0.
- The IP Proxy information was binned into Safe Proxy and Dangerous Proxy, due to the existence of some proxies presenting danger to the company. For security reasons, these will not be disclosed.
- The ThreatSuperCategory, PolicyAction Severity, URLCategory Severity, URLCategory's Class, OS and Browser fields were one-hot encoded.

The last step was to transform the data into a fully numerical format. To determine how to do this, we had to keep in mind that the system must be able to capture a characterization of the entity in a clear way and in a way that it is possible to actually distinguish an anomalous entity from a normal entity, while maintaining a small computational complexity. It was decided that, since this project aims at finding consistently anomalous entities, then, given the time series nature of the data, the best way to approach the problem would be to group the data by a time-like granularity (hour, day, month...). Keeping the computational complexity of this problem in mind, the granularity

which was used, after discussing with security experts, was a daily granularity. Considering this factor, two datasets were created: the Daily dataset and the Entity (User or Machine) dataset. The first one describes the daily action of each entity and the second one fully characterizes the entity throughout the whole time window that is considered for the detection. The process for the User Detector is illustrated in Figure 5.3.

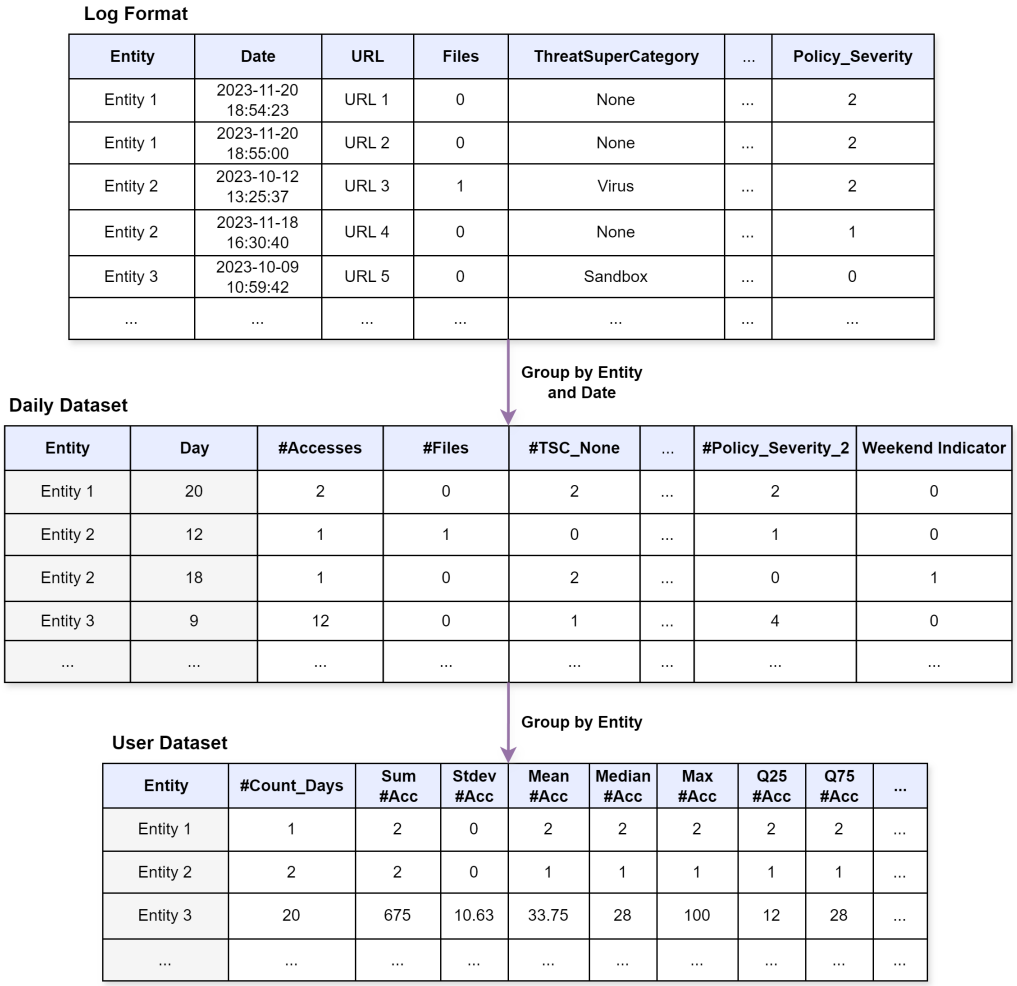


Figure 5.3: Data transformation process from Log format to Daily Dataset to User Dataset. The year and month in the Daily Dataset were hidden for simplicity.

To obtain the first dataset (daily dataset), all that has to be done is to group the data by entity and day (year, month and day), aggregating the features by a sum, creating a frequency-based dataset. An indicator of weekend was also created for this dataset. For the features that were not one-hot encoded, the unique sum was taken while for the other features the regular sum was used. For the Entity Dataset, both the UD and IMD followed different paths due to the different considerations taken to build each detector. However, since we want detect consistently anomalous entities, both detectors performed the following filter: only keep the entities which had 33% of their days with more accesses than the average number of accesses (considering all days of all entities). Also, due to later studies, features regarding weekends and the agent string (OS and

Browser) were dropped since they introduced too much noise.

As mentioned in the beginning of this section, the UD is described first. For the User dataset, finding the most anomalous users was the priority, thus the Daily Dataset was grouped by entity. The features, though, had to describe the behaviour of the entity in this time frame. To do so, for each feature of the Daily Dataset, seven statistical features were extracted by aggregating operations. These were the sum, standard deviation (*std*), mean, median, maximum (*max*), first and third quartiles (*Q25* and *Q75*). The minimum was not considered, since minimum values are not relevant for anomaly detection. In addition, the number of distinct days that a user had entries was also added.

5.1.1 Differences for the IMD

As mentioned before, and as shown in Figure 5.1, the IMD branches out of the regular pipeline three times, needing three extra pre-processing steps to build the appropriate dataset for detection. The first branch deals with a User-Machine connection and transformation and the second branch concerns the creation of an appropriate field that translates the rhythm of infection of the machine.

IP Map

Starting with the first branch, this step stems from the dataset only possessing a User field for entity identifying. However, for the IMD, the required entity is the machine being used during the access; therefore, a connection between the User and the machine at the time the access was registered must be made. To do so, an intermediary field was used—ClientIP. By using a complementary database with information on actions done using the machines, e.g., login, logout and read (we are only interested in logins to know when an IP would be attributed to a machine), the idea would be to create an IP map as is represented in Figure 5.4, which consists of 5 fields: the IP address of the actor; the device's MAC address; the device's Name; and two timestamps, taken from the ts field of the database. This map translates the time window ($ts_i \longrightarrow ts_f$) during which a specific IP address was utilized by a machine, identifiable either by its MAC address or by its name. Therefore, using this map, taking the ClientIP and the timestamp from a log is all that is needed to identify the

IP	Device MAC	Name	ts_i	ts_f
IP1	MAC 1	Name 1	2023-07-01 13:59:32	2023-07-02 17:04:49
IP1	MAC 2	Name 2	2023-07-02 17:04:50	2023-07-04 06:52:29
IP1	MAC 1	Name 1	2023-07-04 06:52:30	2023-11-21 23:59:59
IP2	MAC 3	Name 3	2023-11-09 09:21:22	2023-11-11 06:52:30
IP2	MAC 4	Name 4	2023-11-11 06:52:31	2023-11-21 23:59:59
...	...	...	...	...

Figure 5.4: Representation of the IP Map created to connect an IP to a Machine.

machine used for the access.

The creation of the map, however, was not so straightforward. The database contains hundreds of millions of logs, rendering the process to be slow. Therefore, the making of the map was divided into two parts. For both parts, only the logs with "action" = 'login' that went through ("result" = 'ok') were considered. These logs' timestamps would, consequently, be the main field to build the map. The key point was to first find which ClientIPs from the web proxy dataset corresponded to a VPN object (field of the database) and which did not. In reality, the logs with the condition "object" = 'vpn' had a much smaller volume than the rest, so the process would be much faster due to the smaller amount of data, which is in the order of the hundreds of thousands. For the second part, all objects apart from 'vpn' were considered, but only considering the remaining IPs that were not mapped in the previous phase. Due to the volume of data, iterative extractions by month were performed and transformed into an IP map which were then correctly joined. Joining the maps from both parts results in the full map of IPs. Note that the final timestamp for each IP was chosen to be the end of the time frame of the dataset for anomaly detection.

To finalize, there is one problem that has not been mentioned yet. Due to lack of information in the complementary database for some ClientIPs, meaning that they just did not show at all, some information was ultimately lost. However, it was observed that the data loss was not significant for the system ($1\% - 5\%$) and, thus, deemed not to be relevant.

Having the map, the following step was to correctly connect each ClientIP from the original dataset to its correct machine, considering the timestamp of the flagged web log. This was done using pandas' *merge_asof* function, which, given a ClientIP and timestamp, is able to find the machine used from the map. Using the *merge* function was also possible, but it was observed that the system's memory was not sufficient due to the way the function operates on data.

Finally, the correct identifier—MAC address or Device name— had to be standardized for each machine, since only one identifier can be used per machine. With the security experts aid, it was decided the following: if the device's name existed within a database regarding the inventory of machines in the company, then the machine is internal and its name is used. Otherwise, the machine would be identified by the MAC address. This method was developed because some device names had more than one corresponding MAC address and vice-versa. So, to avoid not considering the user's machine itself, we use the device's name if possible.

Infection Risk Factor

Moving to the second IMD branch, this section covers the creation of a feature that can translate how infected a machine was during a day. For this project, an infection was considered to be a highly volumetric behaviour that had multiple accesses throughout the whole time the machine was on. No similar works like this were found on the Internet, which meant the feature had to be created from scratch.

The first version of the feature, whose name is Infection Risk Factor, R, was:

$$R = \frac{\bar{t}}{d}$$

, where $\bar{t}$ is the average time between consecutive flagged accesses and d is the duration of the day (distance between the first and last access in the day), both in seconds. This feature is comprised in $[0, 1]$. In order to study it, the limit of $R \longrightarrow 0$ and the meaning of $R = 1$ were studied:

1. If $R \longrightarrow 0$, then $d \gg \bar{t}$, which means that either $\bar{t}$ is small and there were many consecutive accesses during the day or there were not many consecutive accesses during the day, but d is not big. In other words, $\bar{t}$ is small in both cases but no real conclusion on d can be taken.

2. If $R = 1$, then $\bar{t} = d$. This is a special case where the user only had 2 accesses in a day, and, therefore, does not translate to any immediate threat.

In lesser words, the presented expression had ambiguities on small values and does not express infection for $R = 1$. Therefore, a new expression was considered:

$$R = \frac{\bar{t} \times d}{\sigma_t^2} \longrightarrow \frac{\bar{t} \times d}{\sigma_t^2 + \frac{\alpha}{A}}$$

, where σ_t^2 is the variance of the time between consecutive accesses, α is an attenuation factor and A is the total number of accesses during the day. Initially, the term $\frac{\alpha}{A}$ had not been inserted; however, in case we had $\sigma_t^2 = 0$, the Infection Risk Factor would become undefined. The number α was divided by A in order to give more weight to days with a bigger number of accesses. The variance was included since a smaller variance means that the accesses were more equally spaced, which represents a more infected behavior. In other words, if the accesses throughout the day were always equally spaced in time, then it is much more probable that the machine is infected.

The Infection Risk Factor was calculated using this expression. During this project, a candidate to infected machine had been found. Using this candidate, this feature was tested and it was observed that this was not the best approach. Therefore, a new expression was derived and tuned taking the infected machine candidate into consideration. The final expression for the Infection Risk Factor was determined to be:

$$R = \frac{d^2 \times A^2}{\bar{t}(\sigma_t^2 + \alpha A + \beta d + \gamma \bar{t})} \tag{5.1}$$

, where α, β and γ are attenuation factors. Although a more complex expression, this equation proved later to be very effective on indicating infection. This feature is positively unbounded and the higher the value, the bigger the infection. In order to demonstrate the role each term plays in the expression, various plots to study the effect of each variable were made, which can be seen in Figures 5.5 and 5.6.

From Figure 5.5, we can justify that:

- The term $\frac{1}{\bar{t}}$ weighs in on giving a bigger R value when average time between consecutive accesses is small. As we can see, the bigger the $\bar{t}$, the smaller the Infection Risk Factor.

- Multiplying by d and A (squared to give these factors a bigger weight) gives emphasis that an infected machine will have a bigger frequency of accesses along the whole day. We see, that, the bigger the A or d, the bigger the R.

- A bigger variance decreased the Infection Risk Factor, as expected. The bigger the variance, the less regular is the time between consecutive accesses, which is less indicative of infected behavior.

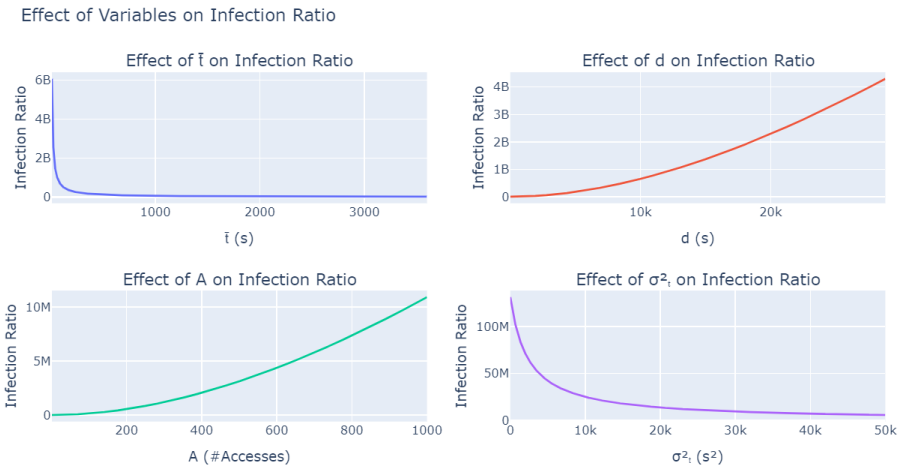
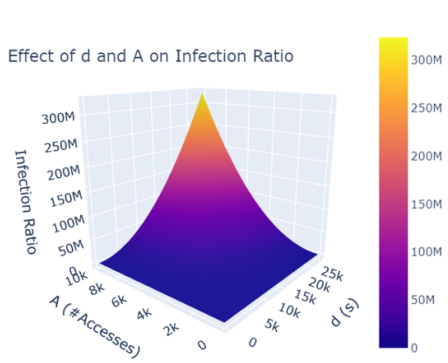
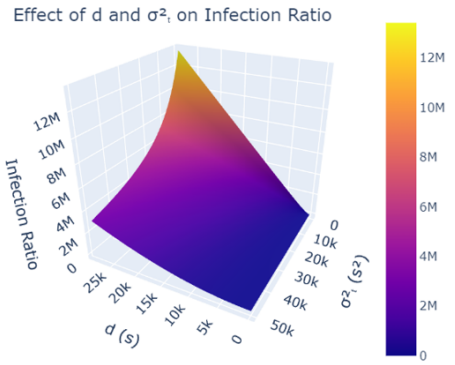


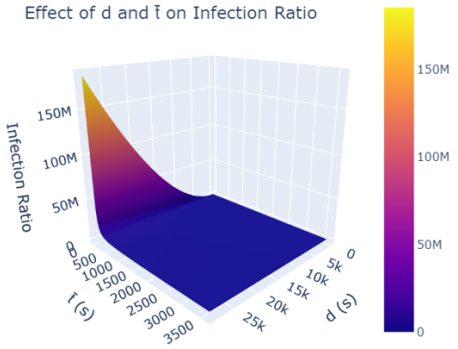
Figure 5.5: Effect of each variable on the value of R. For each variable, all other variables were locked.



(a) Infection Risk Factor in function of A and d.



(b) Infection Risk Factor in function of d and σ_t^2.



(c) Infection Risk Factor in function of $\bar{t}$ and d.

Figure 5.6: Several 3D plots representing various relationships between 2 variables and their joint effect on the Infection Risk Factor.

Finally, from Figure 5.6a one can see that both variables have a similar weight on the Infection Risk Factor. Figure 5.6b shows a bigger effect on the change of σ_t^2 than d. Observing Figure 5.6c one can observe that $\bar{t}$ highly outweighs d and, thus, the Infection Risk Factor gives more relevance to machines with small times between accesses than longer days, which was expected, since a longer day might just mean the machine has one flagged access in the beginning of the day and another at the end, while a small $\bar{t}$, although it may be a small number of accesses, still represents better an infection.

To finalize this section, one must know what values to attribute the attenuation factors. To study this, similar plots to the ones in Figure 5.5 were made, but considering every variable constant, except the attenuation constant at study. The results are shown in Figure 5.7.

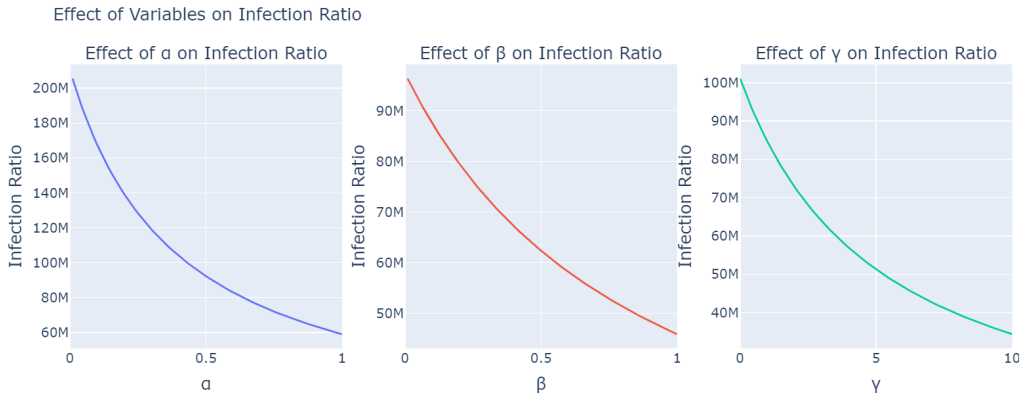


Figure 5.7: Effect of each constant on the value of R. For each constant, all other variables were locked.

The figure shows the order of magnitude that each constant must have to be able to affect the values of R significantly, which are $\sim 10^{-1}$ for α, $\sim 10^{-1}$ for β, and $\sim 10^{0}$ for γ. However, given that an infected machine is better characterized with a smaller $\bar{t}$, a smaller γ value should be considered. Also, given these plots, we can conclude that the attenuation terms of the denominator work as intended, decreasing the value of R and providing different weighting systems to each different variable.

Having the expression derived, the Infection Risk Factor was calculated for each day of each machine and, later, correctly joined on the Daily dataset.

Feature Engineering Pipeline

The final branch corresponds to the pipeline that the daily machine dataset went through in order to form the Machine Dataset. As mentioned in Chapter 3, the detectors must be able to adapt in time. The two detectors will work with different sized time windows. The UD works with a bigger time frame—3 months—in order to better capture the behavior of the user. The IMD, however, works with a relatively smaller time frame—4 weeks—because we are more interested in immediate detections of machines who are very recently anomalous. Hence, different treatments were planned with this in mind. Therefore, there are two main factors that must be reflected in the

machine dataset:

1. An infected machine must be anomalous among all machines;

2. An infected machine must be anomalous according to its history, if applicable (the history might also be anomalous).

In these circumstances, a dataset similar to the one in the UD was created. The main difference was that a separation between the history data and recent data was created. For this dataset, the recent data was considered to be the most recent week and the other three weeks were the historical data. On each of these subsets, the methodology exerted for the user dataset was applied and the resulting datasets were joined, creating a dataset comprised of older statistics and newer statistics. Like this, we have a register on past and recent activity, which allows for effective recent anomaly detection.

Aside from this, recall that for features that do not make sense to sum along all the time frame, the unique amount of values was used instead. However, for the Infection Risk Factor, none of these seemed reasonable, since we wanted a value that could characterize the infected behaviour of the machine in the time window. Therefore, the Infection Quality Value, Q, was created:

$$Q = \left(\frac{I}{D} \times I\right) \times L \tag{5.2}$$

, where I is the number of days the machine had its Infection Risk Factor above the $\mathrm{Q}_3 + 1.5 \times$ IQR (Interquartile Range) threshold, D is the number of distinct days the machine had accesses associated to it and L is the maximum number of consecutive days that the machine had its R above said threshold. Just like the Infection Risk Factor, this expression is positively unbounded and, the higher the value, the more likely is the machine to be infected. This expression has three terms:

1. The first term, $\frac{I}{D}$, acts as a weight for I, translating into the ratio of highly infected days. This expression alone was insufficient, however, since a machine might only have one day of accesses with an infected like behavior ($\frac{I}{D} = 1$, because $I = D$), but not be infected at all;

2. Multiplying the weight by I allows for better discernment between machines with a smaller or bigger D. However, $\frac{I}{D} \times I$ has no information on consecutive infection days. Considering a high value for this term with $D = 4$, it is impossible to determine whether these 4 infected days were consecutive or spread out, such as one per week. If the machine was logged on every day, one day per week would not suggest infected behavior. Hence, this term alone was insufficient for accurate identification.

3. Multiplying this term by the maximum number of consecutive infected days, L, offered insight on the lacking points, thus easing the problem.

To further understand the meaning of Q's different values, a simple discretization is displayed in Table 5.1.

Table 5.1: Value discretization for the Infection Quality Value, Q

Q	Value	Class
	$0 \leq Q < 5$	Low
	$5 \leq Q < 10$	Intermediate
	$Q \geq 10$	High

With this quantity determined, the Machine Dataset can be fully completed, providing a comprehensive understanding of the machine's recent and historical behavior.

5.2 User Detector

In this section, the Model pipeline will be described and justified. In Figure 5.8, the full User Detector pipeline is represented.

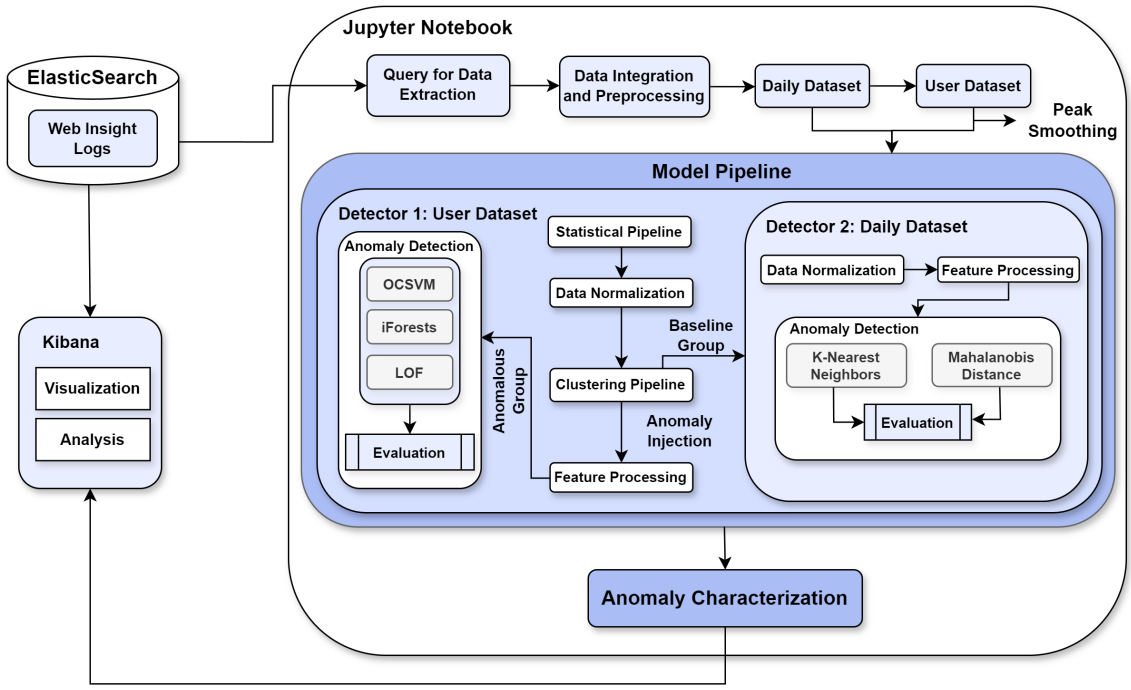


Figure 5.8: User Detector Representative Diagram.

As shown is Section 3.2, the pipeline consists of an Exploratory Analysis phase with Kibana and then moves on to its extraction and pre-processing to obtain the User Dataset, as described in the previous section.

Before moving on to the Model Pipeline, one last change to the User Dataset was applied. Since the system should prioritize users who consistently exhibited anomalous behavior over time,

rather than being overly sensitive to occasional data spikes, a peak smoothing was performed. Users who did not meet a threshold for the number of days with accesses had all their columns scaled down by a small factor, according to the statistic each column represented. This approach was taken since users who survived the initial filter could be users with a few days of high access peaks. This made it so median and quartile columns always took high values, creating noise for the detection. So, for example, the mean columns could be multiplied by 0.5 and the medians by 0 (eliminating them) to fully eliminate these peaks. This method effectively smoothed out the introduced peaks and reduced noise.

Having done the peak smoothing, the Model Pipeline could be applied. This phase consists of three distinct detection phases: Statistical Pipeline, Clustering Pipeline, and Group Pipeline. The two first phases utilize the User Dataset and the last phase utilizes both the Daily and User dataset, depending on the path the data took.

The Model Pipeline starts by detecting purely statistical univariate anomalies as an attempt to try to find evident anomalies. It does so by considering outliers as the points that are beyond the $Q_3 + 1.5 \times$ IQR mark. For this first phase, only the columns that represent the sum of accesses for a basic volumetric analysis were considered. The aforementioned condition was checked for each of these columns for each user. Having the columns that a user is anomalous in, a global statistical outlier score, E, was assigned which is given by:

$$E = \sum_{\text{col} \in \text{outlier}} \left(\frac{\text{col(user)}}{\max(\text{col})} \times (1 - \overline{r_{\text{col}}}) \times w_{col} \right) \tag{5.3}$$

, where outlier is the set of columns that the user is an outlier in, col($user$) is the user's column value, $\overline{r_{\text{col}}}$ is the average Spearman's Rank correlation for that column, and w_{col} is a manually assigned weight to the column. The Spearman Rank was used, since it evaluates the correlation of a column without assuming linear relations. Additionally, the term $(1 - \overline{r_{\text{col}}})$ was introduced, because, when columns are strongly correlated, being an outlier in one column often means being an outlier in another related column. Therefore, we want to give these columns smaller weights. Finally, the weight term was added to give bigger values to columns which were considered more alarming for detection (decided with the support of experts in the area). The final score was, then, mapped into a value between 0 and 1 by using a sigmoid function which was fitted taking into consideration several past result evaluations.

After the Statistical Pipeline, the data was normalized in order to avoid clustering hardships and model requirements. The data was normalized using *scikit*'s implementation of the RobustScaler[10], which is robust to outliers. Instead of removing the mean and dividing by the standard deviation, it instead removes the median and divides by the interquartile range (IQR). This helps to keep the existence of outliers, since the regular StandardScaler reduces the relative distance between outliers and other data points.

Following the normalization, the data was grouped using clustering algorithms. This step stems from the fact that there might be an anomalous group, characterized by users who were con-

[10] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

sistently misbehaved. Therefore, this step deals with the aforementioned group anomalies, where a single point is not anomalous, but the whole group of points is anomalous. For this phase, both k-Means and Spectral Clustering were used in order to find the best possible clustering method. For each model, an exhaustive grid-search hyperparameter tuning was performed. Having no ground truth available, as mentioned before, highly hindered this endeavor. With no way to know which point belongs to which cluster, or even how many clusters there are, evaluating each combination of hyperparameters becomes very difficult. Therefore, initially, the method resorted to the use of intrinsic evaluation methods, which focus on assessing clustering quality. Two different metrics were used: the Silhouette coefficient, which evaluates the overall structure of the clusters, and the Calinski-Harabasz Index, which indicates how well clusters are separated and how compact the clusters are.

Along this project, the User Detector pipeline changed according to the results it presented in order to obtain the best results possible. For this version of the UD, due to the peak smoothing working well, the metrics ended up being obsolete. With the help of security experts, it was observed that the data always split well into two different clusters: an anomalous group, characterized by seemingly bad behavior, and a baseline group, described by users who do not display dangerous behavior; however, the metrics were still computed to study the quality of clustering. To determine which of the two groups was anomalous, a RandomForest [8] was used. This is a supervised ML model which we fit all the data with considering the cluster we want to study as the independent variable (1 if point is in cluster and 0 if not). Having the model fitted, the feature importance is directly extracted. If a cluster is characterized by features that carry some sort of danger for the company, e.g., Mean_#Policy_Severity_2, then it is considered anomalous.

From this point on, the pipeline divides itself in two paths, one for each group of users.

5.2.1 Anomalous Group

Beginning with the Anomalous Group, the primary objective was to perform a model ensemble of an OCSVM, iForest and LOF to identify the users who were the most anomalous within the group. These would be the final results which, combined with the statistical anomaly results, gave us the set of users who were consistently anomalous. However, there is no definite way to check if these detections are a False Positive or not, having to resort to the security experts. This process, however, can be very slow. Besides this, training models with no ground truth information is impossible to do. Therefore, some form of reference information had to be created to train the models and generate indicative metrics of how well each model behaved to obtain the best hyperparameters for each model.

In light of training the models on the Anomalous Group, several artificial anomalies were injected into the dataset. These anomalies were created from original users who had some of their columns boosted in order to create an anomalous user. Different types of anomalies were also created. Some emphasized columns regarding the severity of the PolicyAction and URLCategory, other focused the type of ThreatSuperCategory present and the existence of Files, while others

prioritized IP columns.

Having these manufactured users, a ground truth was built. Besides considering these new users as anomalous, users who had been detected in the statistical phase who were also in the Anomalous Group were considered as an anomaly for the ground truth as well. Using this baseline allows for model optimizing and ensuring that any detection from the system is highly likely to be a real anomaly that should be further studied by the analysts.

Having the reference information, the following step was to process the columns in order to aid on the detection. At this point there were hundreds of columns. This high dimensionality issue would not only hinder the model results, but increase the time of hyperparameter training. Therefore, two different processing steps were taken: Feature Weighting and Feature Selection.

For the Feature Weighting, the features were weighted to give more importance to the fields who were considered more important for detection. One important note is that all of these weights are given by hand to have full control of the weighting system. The weighting methodology which was created is described in Figure 5.9 and consists of dividing the feature set into disjoint sets of global features (groups) which regard the same type of information. A global feature is defined as any feature existing in the Daily Dataset. In other words, the statistical features extracted for the User Dataset are not considered for the meantime. So, for example, one group would be the features regarding the ThreatSuperCategory (None, Virus, etc). Each of these groups has an associated weight such that $\sum_G w_G = 1$, where w_G is the weight of the Group.

Within each group of features, each feature has its own weight so that $\sum_{f \in G} w_f = w_G$, where f is the global feature which belongs to the group G. This way, the mandatory condition for weighting, $\sum_f w_f = 1$ is maintained. Each feature is ranked on importance within each of the groups. The weights for each of these global features are calculated through a series of relationships depending on the needs of the system. Finally, for each global feature, there are 7 statistical features. Each of these statistical features receive the weight of the global feature divided by 7, so each one gets the same weight.

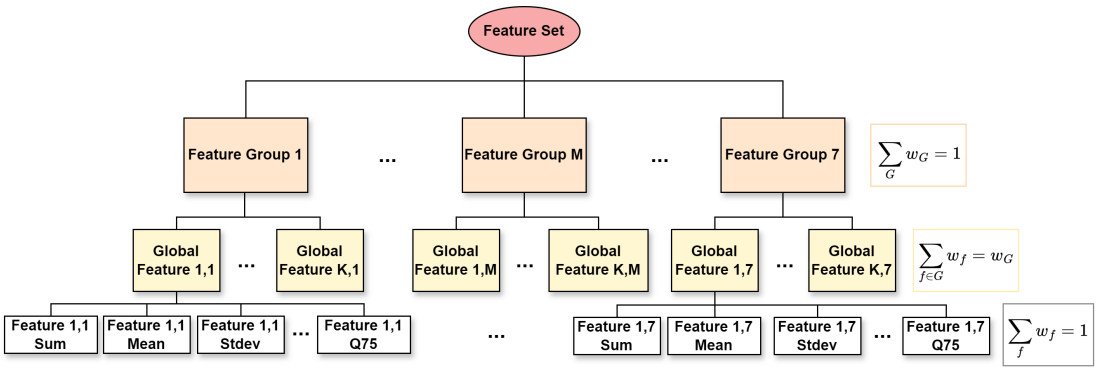


Figure 5.9: Weighting system distribution for the User Dataset's features.

For the Feature Selection, three simple procedures were done. The first was to remove repeated columns, which were generating noise. The second was to remove features which never changed

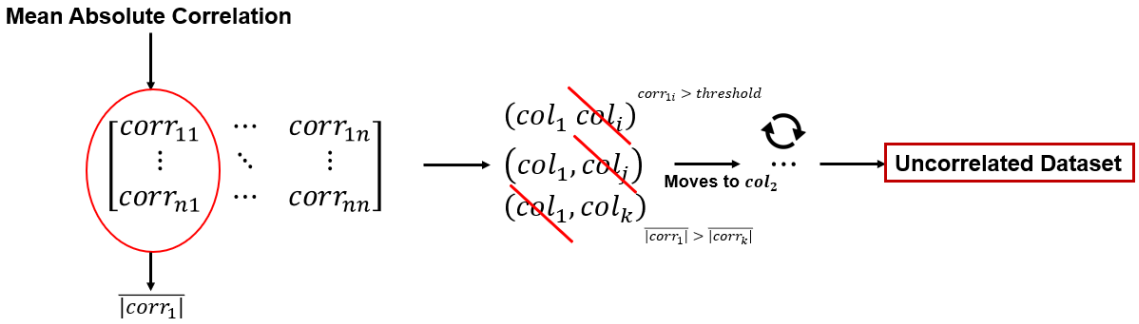


Figure 5.10: Process to create the Uncorrelated Dataset. The columns are iteratively removed based on their mean absolute correlation.

along the users. In other words, their variance was zero. To do so, *scikit*'s VarianceThreshold[11] was used. Finally, highly correlated features were removed to eliminate some noise as well. However, no definite method on removing correlated features actually exists. There is always the question of which of the two correlated features should be removed. Hence, the process illustrated in Figure 5.10 was taken: after calculating the correlation matrix, having the Spearman Rank as the correlation factor, the pairs of correlated features were determined in terms of the considered high correlation threshold. In parallel, the mean absolute correlation value of a feature with all the other features was calculated for all columns. Following this, the features with the highest mean correlation are iteratively removed from the pairs of highly correlated columns until no pairs of highly correlated features are left. With this method, we ensure that highly correlated features to the rest of the dataset are removed, and only the columns with the least noise possible are kept.

These two data processing modules are not separate as one might think. Weighting the features removes information on which columns are duplicated, if the column weights are different. Therefore, information on repeated columns must be saved before weighting the columns.

With the processed dataset, the models can be trained. All three models were implemented using *PyOD*'s implementation. *PyOD* [55] is a Python library specialized in scalable outlier detection for multivariate data, employing just-in-time compilation and parallelization if possible. Both the OCSVM and LOF had their optimal hyperparameters exhaustively searched using the GridSearchCV function. The iForest, however, was a much slower algorithm to train. To find its best hyperparameters, *skopt*'s BayesianCV was used instead. BayesianCV [44] has the goal of automating the process of finding the optimal configuration of a problem and maximizing the performance of an objective function that is expensive to evaluate, while balancing exploration and exploitation to efficiently search the parameter space. For the present case, this function accelerated noticeably the iForest optimizations. It would, however, slow down the optimizations of the remaining two models, hence the usage of GridSearchCV. The models were always optimized using the Recall metric, which measures the frequency with which a model correctly identifies positive instances. This approach was chosen to ensure that the model identifies the maximum number of injected anomalies. Additionally, a lower precision is also desired, as it indicates the

[11] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html

detection of new cases. One important thing to note is that the weights of the groups were randomized (with some conditions on which groups should have bigger weights) and the models were optimized for each weight combination.

The optimized models are then combined into an ensemble, which functions as a result aggregation tool. Essentially, any outlier detected by any of the models is considered a potential detection, thereby expanding the detection scope.

To finalize, the detections of interest are only those which are recent in the time frame. An older anomaly is not relevant to the analyst. Therefore, a final filter was applied to account for the recency of the anomaly. Given the set of anomalous users, a user is considered a recent anomaly if they have more than 75 accesses throughout 7 or more distinct days in the last three weeks. This filter works in three ways: It guarantees recency, it assures detection on reoccurring anomalies (such as a user displaying anomalous behavior, then experiencing a period of reduced activity, followed by data spikes near the present day), and retains users who were consistently anomalous within the time frame

Each detection was, then, thoroughly characterized by an Excel spreadsheet, which is automatically generated by using the *openpyxl* [12] library, with several characterizations to aid the analyst's job, including an illustrative Kibana dashboard of several fields, IP plots and other descriptive quantities. Some examples of this dashboard are shown in Figure 6.7

Considering the pipeline which was just described, this detector has two big strengths, which consist in its flexibility in detecting both anomalous groups (which would not be considered anomalies if we had directly applied the models in the groups) and anomalies inside the anomaly group, which are considered the users who are most likely to be dangerous for the company, whether for being an accident or not (insider). Nevertheless, this detector remains sensitive to clustering, necessitating extra caution when organizing the data. Additionally, it does not account for daily variations; however, the baseline detector addresses this concern.

5.2.2 Baseline Group

Moving to the Baseline Group, a second detector utilizing the Daily Dataset was created. The main goal was not itself to detect anomalous users, since the the users in the Baseline Group have been pre-determined not to be dangerous, but to detect any user who could start displaying strange behavior in the course of the time the UD is not ran again, detecting anomalies on a daily-user level. Therefore, the detector finds anomalous days among a daily-user profile. Technically speaking, this phase works as one detector per user.

After an initial filter for users with only at least 5% worth of days of all possible days (a user profile with so little days could not establish a trustful user baseline), the data was normalized with the RobustScaler and the feature processing was also similar to the anomalous group, only not removing correlated features, since the feature space was much smaller for this dataset.

For anomalous day detection, an ensemble of a kNN and Mahalanobis Distance (with the MCD

[12] https://openpyxl.readthedocs.io/en/stable/

measures included, as mentioned in Section 2.3.5) was created. Both models were implemented with *sklearn* due to its simplicity and the goals of detection. After processing the features, the detector enters a baseline creating phase, where it detects any possible anomalous day for each user's history profile using the ensemble, which works as a weighted average model combination of the normalized model result, and eliminates it from the data in order to form a pure baseline for each user. Finding the best model hyperparameters in this phase is not urgent, since only a simple baseline is wanted. Any possible anomaly should be eliminated.

Having the baselines, the system then matches each incoming new day against the baseline and evaluates if it is anomalous, attributing a score to it. If true, then it indicates the anomalous columns. If not, it adds the new day to the profile. To ensure the hyperparameters chosen for the user models are appropriate, a new dataset of daily anomalies was injected and several distance thresholds were considered for the models, combining them with several different feature weight combinations to find the best hyperparameter-weight combo for each user.

This detector is planned to run weekly. For each day of the week, it filters out the data to only consider the baseline users, and uses their profiles to check if the new day is anomalous. At the end of the week, an average daily score is calculated.

The biggest strength of this detector is the fact that it can detect immediate changes in behaviour (which would take weeks or months to show at the user level). Also, the updating baseline profile of each user allows the system to take dynamic user behaviour into consideration. However, daily variations can generate noise, leading to false alarms.

5.3 Infected Machine Detector

To finalize this chapter, a different pipeline was developed for the IMD due to its underlying nature. Having a different time frame size for analysis, different purpose and a new feature which efficiently describes the infection of the machine allowed for a simpler procedure to be taken. The process is illustrated in Figure 5.11.

The data is comprised of 4 weeks, so that the most recent week can be compared to the three remaining past weeks. As previously shown, the data has both history statistical features and recent statistical features. For the detection module, only the features regarding the Infection Risk Factor were considered, since we are mostly interested in the infection behavior. The rest of the features were used in a later phase.

Both the normalizing (RobustScaler) and feature weighting were done separately for each of the historical and recent features, leading to two different normalized and weighted datasets. Based on the curated datasets, a machine is considered potentially infected if it exhibits anomalies relative to other machines and its historical behavior. Keeping this in mind, two distinct anomaly candidates were examined:

- Anomalous History: A machine is an anomaly candidate if its history is anomalous to all the other machines' histories.

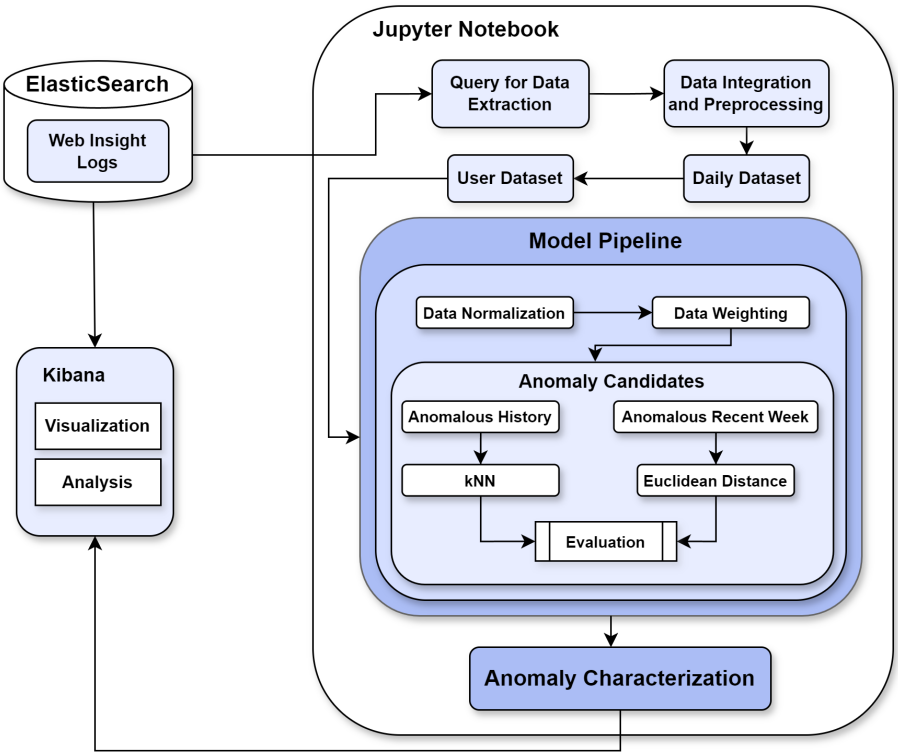


Figure 5.11: Machine Detector Representative Diagram.

- Anomalous Recent Week: A machine is an anomaly candidate if its most recent week is anomalous to its history.

For the first candidate case (Anomalous History), a simple k-Nearest Neighbors model was implemented to check the machines which had the most anomalous history, created with the *sklearn* library, identically to the Daily Detector. For the second case, the recent data was compared to the older data with a simple Euclidean Distance, implemented using *numpy*'s *linalg.norm* command. The thresholds for both candidates were tuned taking into account an existing identified infected machine.

Once the candidate infected machines were identified, an infection score was calculated based solely on infection-related features. Machines with scores above a certain threshold were flagged as possible detections. These detections were then evaluated against all other non-infection-related features to determine a criticality score, with higher scores indicating more critical detections. Finally, the infection and criticality scores were combined into a final weighted score, reflecting the overall infection level of each machine.

In the same vein as the UD, following a detection, a descriptive Excel spreadsheet is automatically created as an attempt of making the analyst's job easier and to aid on protecting the company. It includes data on the machine which was used, a link to the Kibana Dashboard with the data and other descriptive quantities to characterize the anomaly. An example of the report can be seen in Figure 6.16.

Chapter 6

Experimental Evaluation

In this chapter, the experimental setup and the results obtained under these conditions will be shown for each of the systems which were implemented. As mentioned, both detectors work under a sliding window configuration. For this thesis, one specific time window will be evaluated for the data processing. This time frame's results, along with other results from the sliding window, will be shown. However, for security reasons, some values will be hidden.

6.1 Experimental Setup

To employ the pipelines described in the previous chapter, various different Python libraries and systems were used. The main Python code was written in Jupyter Notebooks, set up in an Altice remote machine system, with 8 CPUs, 100GB of disk space and an available 32GB of RAM. All processes were always thought out considering these system limitations. For the sake of future reproducibility, in Table 6.1 the versions of the main programs utilized in this thesis are written.

Table 6.1: Versions of the main programs used to implement the project.

Kibana $\longrightarrow$ 7.17.11	*scipy* $\longrightarrow$ 1.7.2
Python $\longrightarrow$ 3.9.6	*sklearn* $\longrightarrow$ 1.4.2
elasticsearch $\longrightarrow$ 7.17.9	*pyod* $\longrightarrow$ 1.1.3
mysql $\longrightarrow$ 8.0.33	*plotly* $\longrightarrow$ 5.18.0
numpy $\longrightarrow$ 1.21.2	*matplotlib* $\longrightarrow$ 3.4.3
pandas $\longrightarrow$ 1.3.2	*pickle* $\longrightarrow$ 4.0

6.2 Data Pipeline Results

In this section, the data details of each processing phase will be closely described for both detectors. Since the time frame's size for each detector is different (3 months for the UD and 4 weeks for the IMD), the tables will be displayed side by side, one per detector. The results presented are only example results. As this dissertation stands, the sliding window configuration has already

been implemented. Therefore, one example time frame for each detector was chosen for the Data Pipeline. The detector results, however, will include other time window results. It is important to notice that not all results will be shown due to space limitations.

The time frames which were chosen for this section were:

- User Detector: 2024-02-01 : 2024-04-30
- Infected Machine Detector: 2024-04-15 : 2024-05-12

It is safe to assume that any value presented in the following sections has its order of magnitude maintained for different time frames of the same size.

6.2.1 Data Extraction Results

Tables 6.2 and 6.3 show the number of unique values of each extracted feature before and after filtering the features during extraction as explained in Section 5.1. No information is shown on the number of distinct machines before filtering, since the machine mapping is only introduced after the filtered extraction. One important detail is that the values shown for the IMD data refer to the data after removing IPs with no information, which consists of one extra filtering step.

Table 6.2: UD's number of distinct values pre and post feature filtering.

Feature	Pre-Filter	Post-Filter
Users	6802	3002
Agent	371	143
ClientExternalIP	4874	626
ClientIP	22127	5186
FileName	1610	2
PolicyAction	22	17
TSC	4	4
URL	254235	19091
URLCategory	55	50

Table 6.3: IMD's number of distinct values pre and post feature filtering.

Feature	Pre-Filter	Post-Filter
Machines	No info	1714
Agent	172	42
ClientExternalIP	2227	354
ClientIP	13779	2381
FileName	747	2
PolicyAction	18	12
TSC	4	3
URL	107052	5857
URLCategory	37	21

6.2.2 Data Integration and Preparation Results

In Tables 6.4 and 6.5, information on the amount of rows of the dataset during the data processing and integration is shown. The path that the number of accesses took is described by the first three lines and shows a clear decrease due to the feature filtering and access correction bucketing. Besides this, the size of the entity (user or machine) dataset represents the number of entities

that we are interested in for the outlier study. The pipeline shows a clear decrease in the original number of entities than the number shown in the previous tables, which proves to be a successful round of filtering for the system.

Table 6.4: Size of dataset during the data pipeline for the User Detector.

Phase	**#Rows**
Raw Data	10420366
Post-Filter	121957
Access Correction	86228
Daily Dataset	10122
After 33% Filter	2598
User Dataset	256

Table 6.5: Size of dataset during the data pipeline for the Infected Machine Detector.

Phase	**#Rows**
Raw Data	3148522
Post-Filter	25902
Access Correction	17547
Daily Dataset	2900
After 33% Filter	571
Machine Dataset	123

To end this section, the evolution of the number of the feature set is shown in Tables 6.6 and 6.7. The * is introduced to mean that features either were removed since they were purely indicators for coding details or became indexes.

Table 6.6: Size of feature set in each step of data processing for the User Detector. The * indicates feature removal or indexing.

Phase	**#Features**
Raw Data	31
Extraction	10
Data Enrichment	29
Access Correction	31
Feature Extraction	62
Daily Dataset	56*
Feature Noise Removal	39
User Dataset	239*

Table 6.7: Size of feature set in each step of data processing for the Infected Machine Detector. The * indicates feature removal or indexing.

Phase	**#Features**
Raw Data	31
Extraction	10
Data Enrichment	26
Access Correction	28
Feature Extraction	55
Daily Dataset	51*
Feature Noise Removal	36
Machine Dataset	418*

6.3 User Detector Results

The User Detector's results will be showed first. After creating the User Dataset, the first detection phase was to find purely statistical univariate outliers. The most alarming result is shown in Figure 6.1. This module not only attributes a score to each user, but is also capable of fully explaining

the score, showing the features which contributed the most for the final score. The presented example shows a user that had a very high volume of accesses considered to be dangerous.

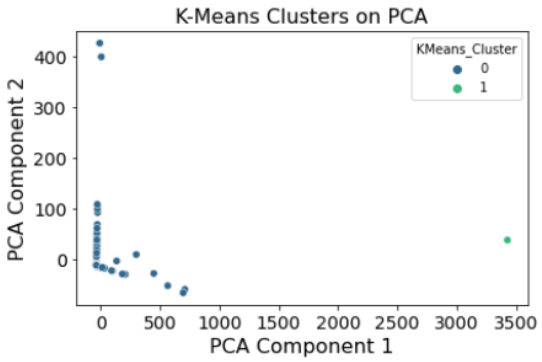
```
User: [redacted] ---> Score: 0.987885
Columns:
Sum_#URLCategory_Severity_2 ---> Weight: 0.007208 ---> User's Rank in Column: 1 (6357 Accesses)
Sum_#Policy_Severity_2 ---> Weight: 0.004940 ---> User's Rank in Column: 1 (6350 Accesses)
Sum_#Policy_Status_Blocked_Relevant ---> Weight: 0.002341 ---> User's Rank in Column: 1 (6385 Accesses)
Sum_#Accesses ---> Weight: 0.001156 ---> User's Rank in Column: 1 (6385 Accesses)
Sum_#Class_URLCategory_Advanced_Security_Risk ---> Weight: 0.000768 ---> User's Rank in Column: 1 (6385 Accesses)

Rank in Sum_#Accesses: 1 ----> 6385 Accesses
```

Figure 6.1: Example of a purely statistical anomaly detection.

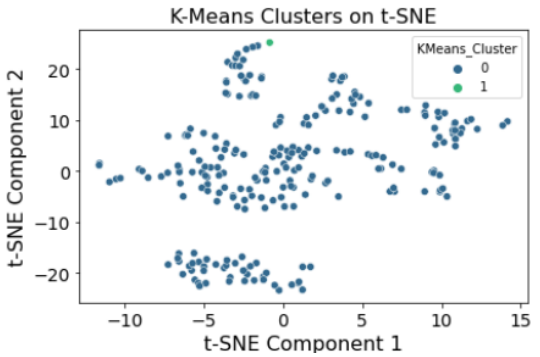
The data was then normalized and clustered into groups. For the Clustering, both k-Means and Spectral Clustering models were tested to detect anomalous clusters in the data. The tested parameters and optimal results are shown in Appendix A.2. To observe the resulting clusters, the data was projected into 2 dimensions using both a PCA and a t-SNE model. The optimal number of clusters, as mentioned in Chapter 5, was considered to be two with the aid of security experts, since it split the data between anomalous and baseline users. Additionally, to check if 2 clusters were in reality a good result, the Silhouette Coefficient and Calinski-Harabasz Index were calculated. The results for the clustering are shown in Figures 6.2 and 6.3 and the two clustering assessing metrics are described in Figures 6.4 and 6.5. To read the Silhouette plot, consider each cluster as a group of filled in horizontal bars, each bar being a user. Each bar has a corresponding Silhouette Score, where values closer to 1 indicate well-clustered points, values near 0 suggest overlapping clusters, and negative values denote a possible misclassification.

An assessment of the plots regarding the PCA and t-SNE representations lets us know how differently the clustering algorithms grouped the data. We see that the k-Means method failed in clustering data. This project demonstrates this method's weakness to irregular shaped clusters, being sensitive to outliers. The Spectral Clustering, however, showed great promise, being able to split the users into 2 groups. This separation is clearly seen with the t-SNE 2D representation.

When it comes to the PCA representation, it kept circa 97% of the data variance, which

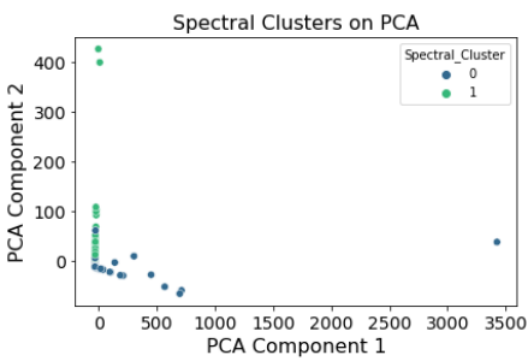


(a) PCA representation of the k-Means clustering method for the User Dataset with $k = 2$.

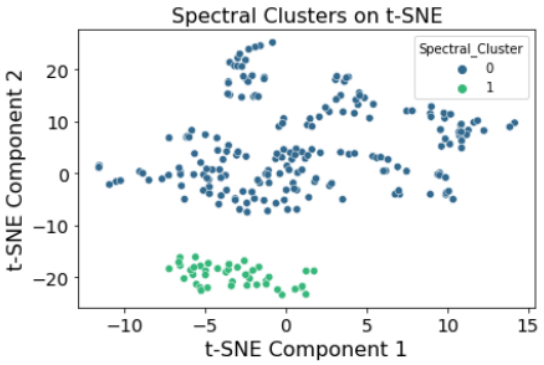


(b) t-SNE representation of the k-Means clustering method for the User Dataset with $k = 2$.

Figure 6.2: Clustering results for the k-Means algorithm projected into 2D using both PCA or t-SNE.

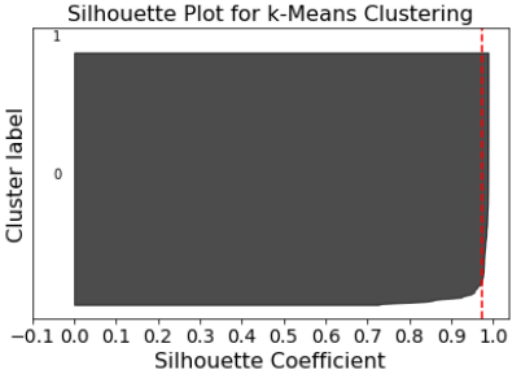


(a) PCA representation of the Spectral clustering method for the User Dataset with $k = 2$.

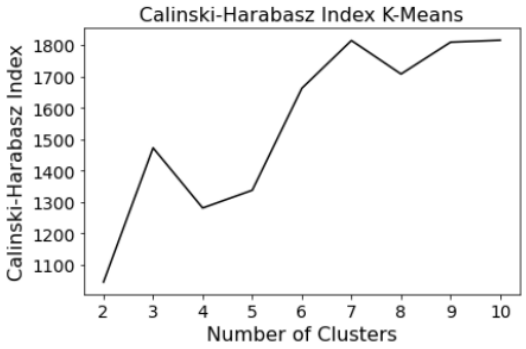


(b) t-SNE representation of the Spectral clustering method for the User Dataset with $k = 2$.

Figure 6.3: Clustering results for the Spectral Clustering algorithm projected into 2D using both PCA or t-SNE.

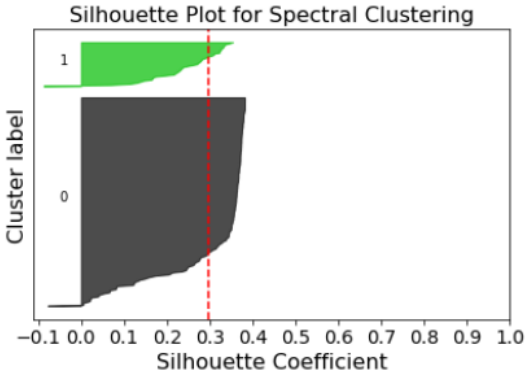


(a) Silhouette plot for the k-Means algorithm for $k = 2$. The red dashed line indicates the mean Silhouette plot.

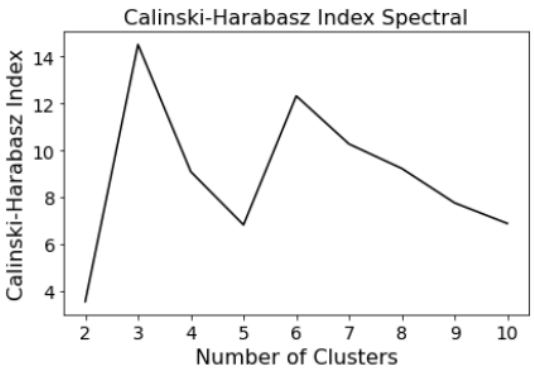


(b) Calinski-Harabasz Index plot for different values of number of clusters for the k-Means algorithm.

Figure 6.4: Various intrinsic metrics for the k-Means algorithm.



(a) Silhouette plot for the Spectral Clustering algorithm for $k = 2$. The red dashed line indicates the mean Silhouette plot.



(b) Calinski-Harabasz Index plot for different values of number of clusters for the Spectral Clustering algorithm.

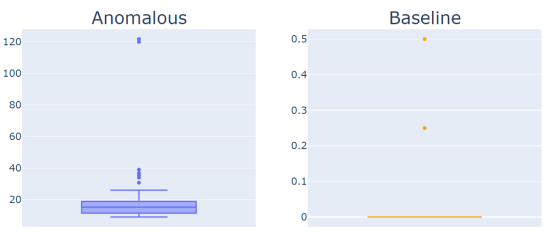
Figure 6.5: Various intrinsic metrics for the Spectral Clustering algorithm.

is very good. The first principal component, PC1, consists of 94% and the second, PC2, has around 3%. From Figure 6.3a, we see that most points from Cluster 1 are along PC2, which means that possible outliers in this direction are not, in principle, as anomalous as outliers in PC1. However, a quick check on the features which contributed the most to each component revealed that the features which characterize PC1 are all related to features which carry no danger, e.g., "Mean_#Policy_Severity_0". This is not a general case, as there have been other time windows in which Cluster 1 was located along the PC1.

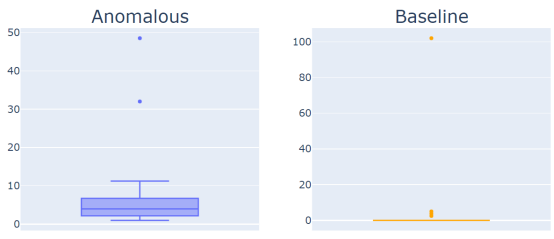
Furthermore, determining the feature importance for each cluster with a RandomForest lead to the conclusion that Cluster 1 is anomalous, since it was characterized by features which present a threat to the company. In a general case, the smaller cluster is always characterized by possibly anomalous users.

In parallel, the choice of the number of clusters had to be studied. As mentioned before, the number of clusters was ultimately chosen to be $k = 2$ with the aid of security experts. As we can see from Figures 6.4a and 6.5a, $k = 2$ leads to a good average silhouette score. The Calinski-Harabasz Index tells a different story, although, indicating that $k = 2$ is not the best value. Theoretically, $k = 3$ would be a better choice considering only the metrics. In spite of this, studying the third cluster showed that it was composed only of users who presented a "more anomalous" behavior compared to those in the baseline. Considering this, it was concluded that these users did not pose a threat to the company and leading with $k = 2$ would be more appropriate. These metrics are merely indicators of how well the clusters are formed which are used when there is no ground truth, leading to a bigger trust in the experts' insights. Therefore, considering all that has been said, the Spectral Clustering algorithm was chosen for clustering.

To better understand the sheer difference between the clusters, in Figure 6.6, the boxplots for the five most important features are shown for each cluster. From these plots, the effect of the peak smoothing can be observed. The variables referring to the Q25, median and Q75 variables were highly diminished for users with a low amount of distinct days with accesses, allowing for users who were consistently getting relevant blocked web accesses (high #Count_Days) to stand out.

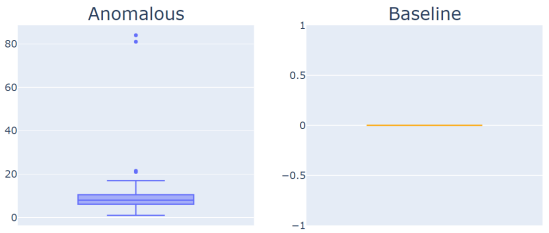


(a) Boxplot of anomalous and baseline groups for Q75_#Policy_Status_Blocked_Relevant.

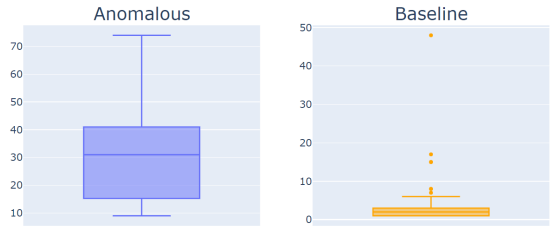


(b) Boxplot of anomalous and baseline groups for Q25_#Policy_Status_Blocked_Relevant.

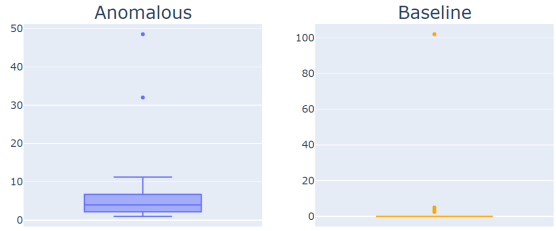
Figure 6.6: Boxplots comparing the anomalous group and baseline group for the 5 most important features for clustering (continues in next page).



(c) Boxplot of anomalous and baseline groups for Median_#Policy_Status_Blocked_Relevant.



(d) Boxplot of anomalous and baseline groups for #Count_Days



(e) Boxplot of anomalous and baseline groups for Q25_#Accesses.

Figure 6.6: (Continuation from previous page) Boxplots comparing the anomalous group and baseline group for the 5 most important features for clustering.

The two groups were then separated into their respective detection pipeline. The results of each pipeline are described in the following two sections.

6.3.1 Anomalous Users

In Table 6.8, the dataset size for the anomalous group pipeline is displayed. With the anomalies being injected, a ground truth could be created which allowed the outlier detection models to be trained. However, always keep in mind that this ground truth is mostly synthetic, since the anomalies were generated by hand, although created from real users.

Table 6.8: Dataset size for each phase of the Anomalous Group pipeline.

Phase	**#Rows**
User Dataset	256
Anomalous Group	43
Anomaly Injection	69

The following phase consists of creating an ensemble of an OCSVM, LOF and iForest. The training of the three models involved several different feature weighting and hyperparameter combinations. The data regarding every weight and hyperparameter combinations, as well as the optimal results for each model, is presented in Appendix A.3. Reiterating, these results are only for the time window being considered for the UD. A different time window would lead to completely different results, as we will see ahead.

Table 6.9 shows the number of features of the dataset as the data is additionally processed for the model training. Building models in a dataset whose number of features is far bigger than the number of rows may present dimensionality problems. Thus, a reduction must occur to avoid overfitting and other issues. The correlation threshold chosen for this phase was of 0.9, since it proved to be a good balance between information loss and dimensionality loss.

Table 6.9: Anomalous Group's number of features during the extra feature processing.

Phase	**#Features**
User Dataset	239
VarianceThreshold	183
Duplicated Features	145
Correlated Features	59

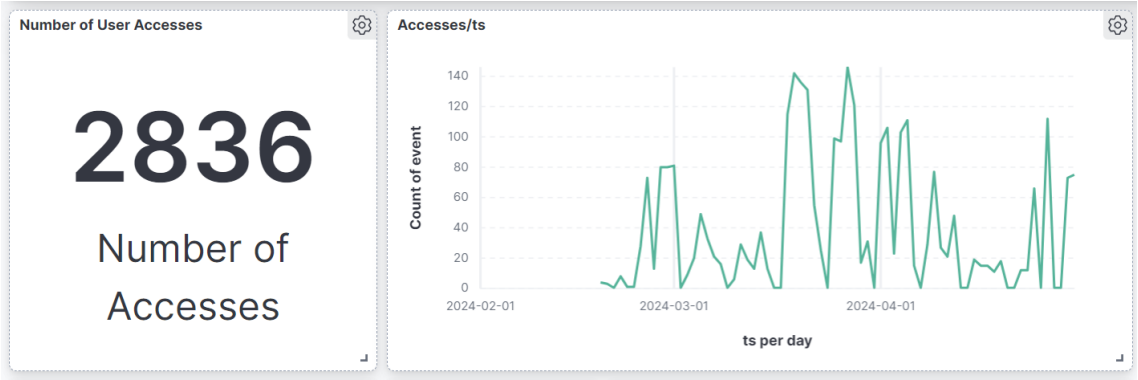
After training the 3 models, the ensemble was formed where any possible detected outlier is considered. Then the results were filtered for recent anomalies. After several tests considering previously found anomalies and help from the team, an anomaly was considered recent if in the last 21 days the user had 7 distinct days of accesses, summing to over 75 filtered accesses. From here on, several results will be shown. This time window managed to detect three different anomalous users; however, examples from other time frames will also be given.

We start by showing one of the detected outliers, illustrated in Figure 6.7. The report shows a user with a significant amount of severe accesses who are distributed fairly in time. This is, thus, a good example of a user who poses a threat for the company. From now on, only the visible rows of the Excel table will be shown, and the extra information and IP pie plots will be hidden.

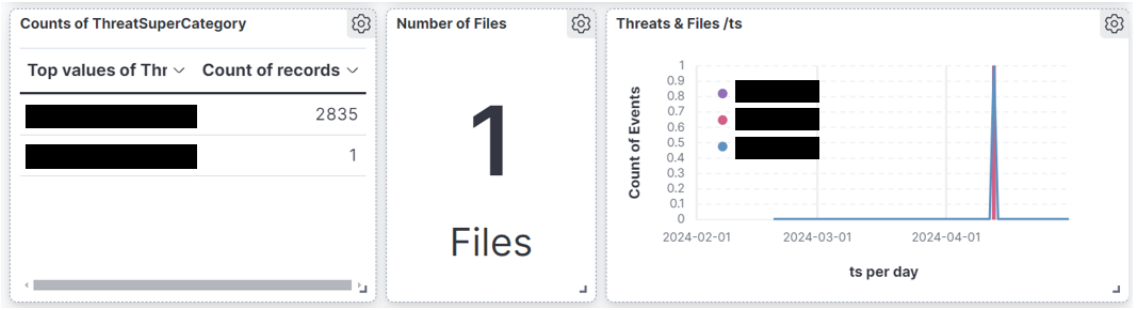
Employee's Account	[redacted]
E-mails used in analysis	[redacted]
Employee's Link	[redacted]
Employee's Name	[redacted]
Company/Management	[redacted]
Work Place	[redacted]
Total number of filtered accesses	1261
Distinct days with accesses	57
Flagged accesses with relevance	1260
Detector Score	0.999
Detection Period	2024-02-01 to 2024-04-30
Report Date	[redacted]

(a) Screenshot of the summary of the Excel report for the detected anomaly.

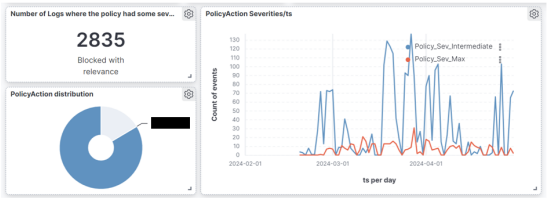
Figure 6.7: Example of a detected anomalous user (continues in next page).



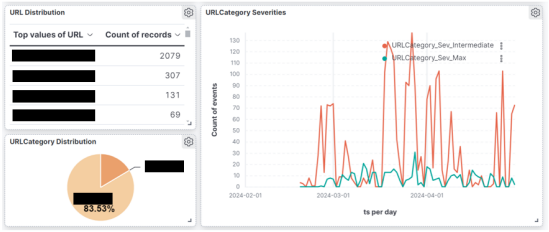
(b) Time series for the detected anomaly, taken from the Kibana Dashboard.



(c) Threat time series for the detected anomaly, taken from the Kibana Dashboard.



(d) Policy Action section of the Kibana Dashboard.



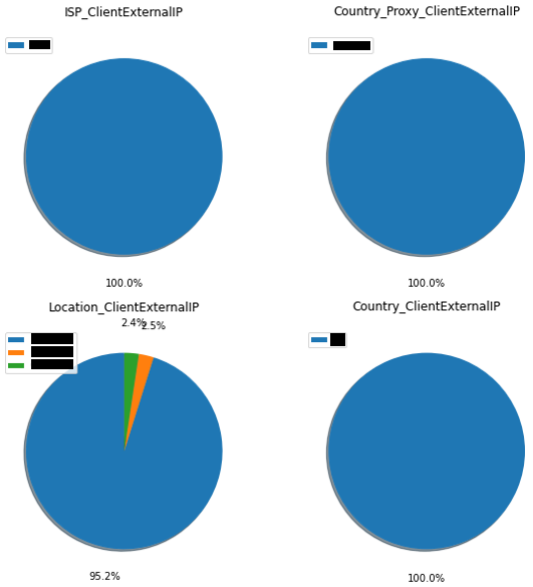
(e) URL section of the Kibana Dashboard.

Account	Detection Phases	Detection Date	Analysis Start
[redacted]	Modelling	[redacted]	2024-02-01 00:00:00

Analysis Ending	Score	Score_Volume	Score_Criticality
2024-05-01 00:00:00	0.999	0.996	1

Acesses_Week	Acesses_Month	Blocked_Frac_Week	Blocked_Frac_Month
143	528	1	0.998

(f) Screenshot of extra information given by the detector.



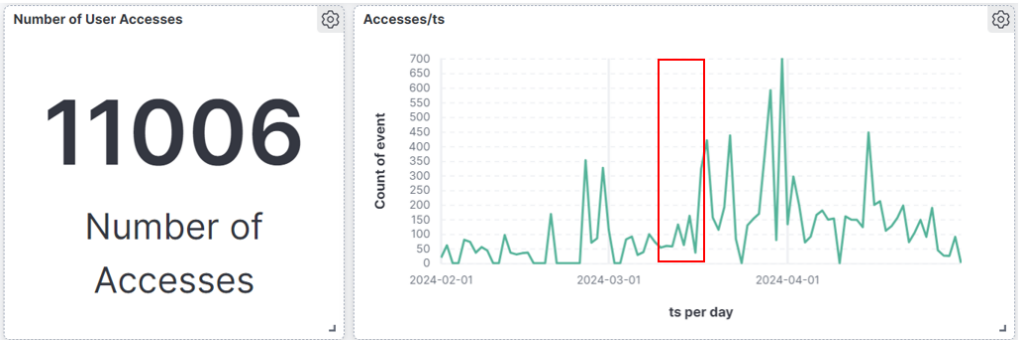
(g) ClientExternalIP pie plots. A similar plot is also available for the ClientIP.

Figure 6.7: (Continuation of previous page) Example of a detected anomalous user.

The preceding example illustrates a user who was consistently anomalous. The following two examples also show this behavior. The difference, however, is that the IMD also detected infected machine behavior on the computers these users own. In Figures 6.8 and 6.9, an excerpt of the report is shown. The red rectangles indicate the periods of possible machine infection detected by the IMD.

Total number of filtered accesses	6385
Distinct days with accesses	74
Flagged accesses with relevance	6385
Detector Score	1
Detection Period	2024-02-01 to 2024-04-30

(a) Screenshot of the summary of the Excel report for the detected anomaly.

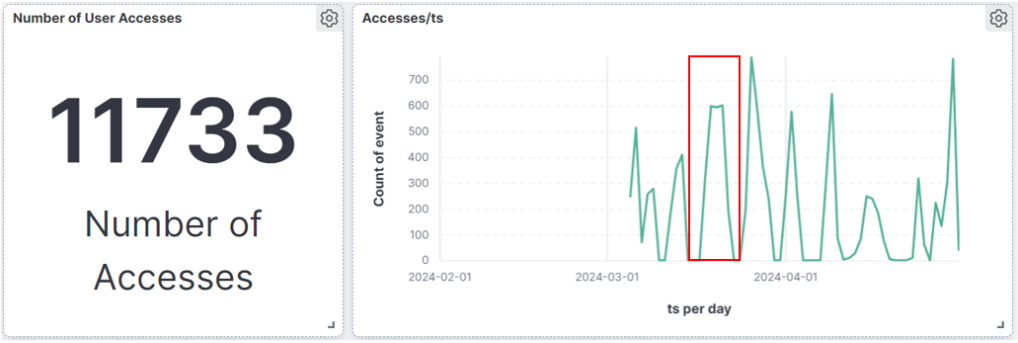


(b) Time series for the detected anomaly, taken from the Kibana Dashboard.

Figure 6.8: Example of a detected anomalous user. The red rectangle indicates a possible infected machine time period detected by the IMD.

Total number of filtered accesses	3801
Distinct days with accesses	41
Flagged accesses with relevance	3801
Detector Score	1
Detection Period	2024-02-01 to 2024-04-30

(a) Screenshot of the summary of the Excel report for the detected anomaly.



(b) Time series for the detected anomaly, taken from the Kibana Dashboard.

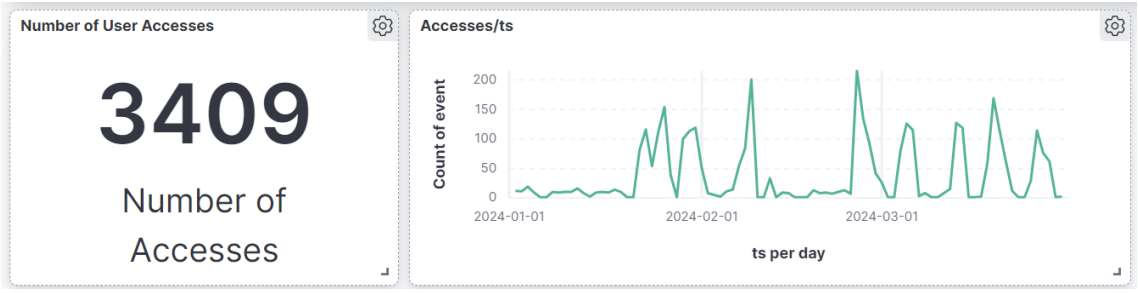
Figure 6.9: Example of a detected anomalous user. The red rectangle indicates a possible infected machine time period detected by the IMD.

Besides sharing detections, one detector does not exclude the other. As a matter of fact, one detector can actually reinforce the other on the existence of anomalous behavior, be it infected machine or misbehaving user. Also, the existence of both detectors is crucial, since each of them has their own focus and can detect anomalies that the other cannot. Additionally, these two cases show a big volume of accesses consistently throughout the whole time window—thus the detection by the UD.

Besides this time frame, anomalous users from other time windows were also captured by the UD. In Figure 6.10, one detection, which was not caught by the IMD, in a different time window is shown. Again, a consistently anomalous behavior is observed. Each dip from data peaks indicates weekends, where the user would not be at work. The user also had accesses from intermediate and maximum severities consistently.

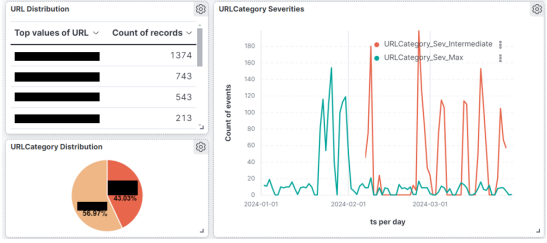
Total number of filtered accesses	1411
Distinct days with accesses	69
Flagged accesses with relevance	1411
Detector Score	0.996
Detection Period	2024-01-01 to 2024-03-31

(a) Screenshot of the summary of the Excel report for the detected anomaly.



(b) Time series for the detected anomaly, taken from the Kibana Dashboard.

(c) Policy Action section of the Kibana Dashboard.



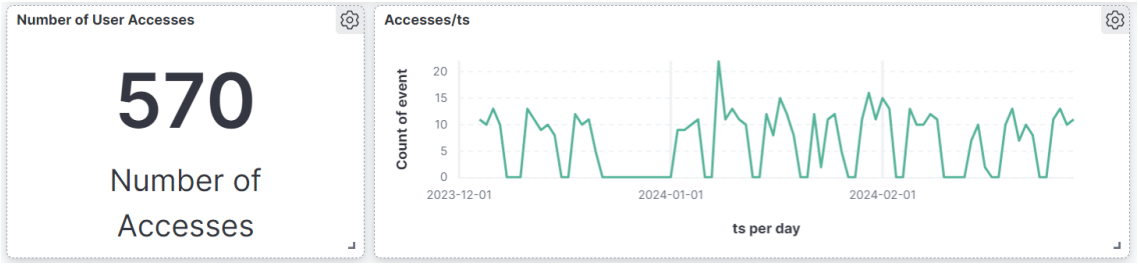
(d) URL section of the Kibana Dashboard.

Figure 6.10: Example of a detected anomalous user in a different time frame.

The final example chosen for this dissertation is one where the volume is not as high as the remaining presented cases, illustrated in Figure 6.11. This user presents a good example on how a moderate volume of accesses can still be detected by the UD. The user clearly shows consistent accesses, paused only by a possible Christmas time vacation in the December. Also, the user presented a peak in threat related columns, which can also make the detection a priority by the UD.

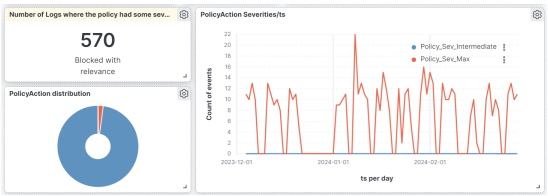
Total number of filtered accesses	570
Distinct days with accesses	54
Flagged accesses with relevance	570
Detector Score	1
Detection Period	2023-12-01 to 2024-02-29

(a) Screenshot of the summary of the Excel report for the detected anomaly.



(b) Time series for the detected anomaly, taken from the Kibana Dashboard.

(c) Threat time series for the detected anomaly, taken from the Kibana Dashboard.



(d) Policy Action section of the Kibana Dashboard.

(e) URL section of the Kibana Dashboard.

Figure 6.11: Example of a detected anomalous user in a different time frame. This user shows a smaller volume, but a great consistency in time.

6.3.2 Baseline Users

Concurrently, the users who are in the baseline go through the Baseline Pipeline. This phase creates an ensemble of a Mahalanobis Distance and kNN detectors for each of the users. In Tables 6.10 and 6.11, the number of rows and features for the Daily Dataset are shown, given the initial feature noise removal and user filter.

To present the results, a specific baseline user was chosen as an example of the process each user goes through to detect daily anomalies. Keep in mind that each user is different from each other, so the number of features and rows per profile is expected to be different between different users. Therefore, the average amount along all users will also be shown. In Table 6.12, the amount

Table 6.10: Size of Daily Dataset during initial steps.

Phase	#Rows
Daily Dataset	2598
Baseline Users	538
Post 5% Filter	187

Table 6.11: Size of feature set during initial steps.

Phase	#Features
Daily Dataset	56
Feature Noise Removal	35

of features of the user and the average among all users is displayed. When it comes to the average amount of days in a user profile, users typically had flagged accesses for 9.35 days.

Table 6.12: Number of features for the chosen user's profile and average number of features for all users' profiles in each step of data processing.

Phase	User #Features	Average #Features
Feature Noise Removal	35	35
VarianceThreshold	20	12.35
Duplicated Features	13	5.05

After weighting the data, a baseline for each user was created by detecting anomalies on each processed user dataset. As mentioned before, these outliers were removed in order to create a pure baseline profile for each user. In Figures 6.12 and 6.13, the outlier map for both detectors is shown, with the data represented both in 2D using t-SNE and PCA representations.

The PCA representation manages to show better the different outliers. It is important to notice that a perfectly appropriate distance threshold in this phase is not necessary, since we only want to

Figure 6.12: Detections made by the Mahalanobis detector on the considered user.

Figure 6.13: Detections made by the k-Nearest Neighbor detector on the considered user.

eliminate a point that could possibly go out of norm compared to the rest of the profile. Therefore, a distance threshold of 0.95 (distance percentile) for the individual models was chosen for this phase.

To explain the combination of the models, after investigating several different users, it was concluded that the Mahalanobis model prioritized outliers in the Principal Component 2, which does not capture the same amount of information as the Principal Component 1 (PC1), while the k-Nearest Neighbors managed to get more times the outliers along the PC1. One does not exclude the other, however. Although, in principle, we would be more interested in the outliers along the PC1. Therefore, the weights given to the kNN and Mahalanobis were 0.75 and 0.25, respectively. The threshold given for the ensemble was of 0.85, in order to eliminate the most outlying points as possible.

With the ensemble, outliers are fully detected, characterized and eliminated. The characterization includes information on the columns which are a statistical outliers and their rank value on the user's profile. Besides this, the characterization must tackle the problem of the reason of an outlier existing being because the day is anomalous due to low values and not high values. These outliers are not of interest. In case they exist, they do not get eliminated. In Figure 6.14, an example of two outliers' characterizations is shown. The (Boolean, Boolean, Integer) tuple indicates the following: (Low Outlier, High Outlier, Rank). The provided example shows that all columns were outliers because their value was high and they all ranked first in the profile.

With each user profile, the detector was ready to capture any new daily outlier with the weekly detector. As explained before, to get an optimal distance threshold for the ensemble, anomalies were injected and several models were trained considering different weights and thresholds. In total, each profile had the same six anomalies injected and the models were trained considering them as anomalous. The training parameters are described in Appendix A.4.

At the time this thesis is being written, the weekly detector has yet to have detected a surge in

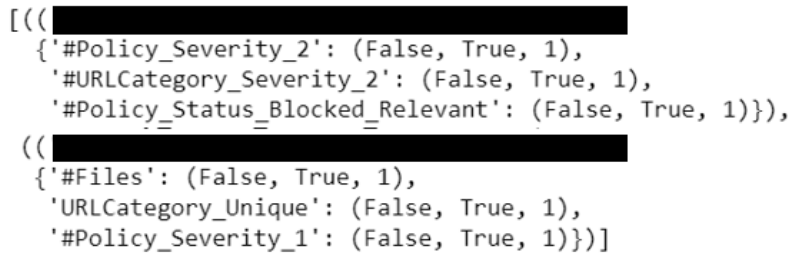


Figure 6.14: Daily profile outlier characterization example.

a user's anomalous behavior. It has detected individual daily anomalies, but the combined score of the week has yet to have reached an anomalous threshold. The maximum weekly score detected for this time window (during the month of may) was of only approximately 0.28. This, though, does not mean that this anomalous surge will not happen in the future.

6.4 Infected Machine Results

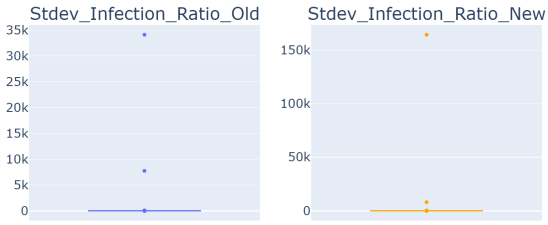
Similarly to the User Detector results, several different detections regarding the Infected Machine Detector will be portrayed in this section. The first step for the detector was to post-process the data to make it appropriate for detection. Table 6.13 shows the evolution of the number of features through the feature processing. Remind that for the anomaly candidate step, only the features regarding the Infection Risk Factor are used. Several combinations of weights were experimented to obtain the best results. After some rounds of result tuning with the aid of SOC analysts, the best combination of weights, along with other important factors, is shown in Appendix A.1.

Table 6.13: Size of feature set in each step of data post-processing for the IMD anomaly detection.

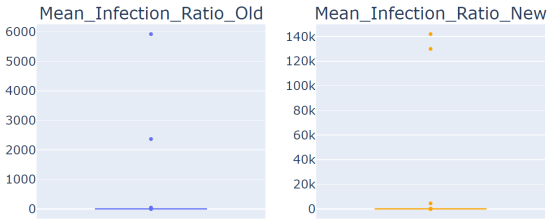
Phase		**#Features**
Machine Dataset		418
Normalization		418
Dataset Separation	Infection Dataset	14
	Critical Dataset	404
Dataset Weighting (Infection)		14

The resulting two datasets allow for both detection and appropriate outlier scoring. In order to get a better understanding of the resulting data, boxplots comparing both the historical time window and recent time window are shown in Figure 6.15 for each of the features of the infected dataset. From these results, the distribution of each field is explained and we see that the new week has much higher values than the history data. This is indicative of the existence of one or more infected machines, which is shown in the outlying points of the boxplots. A process only considering these boxplots would be insufficient, though, since an univariate analysis may not be

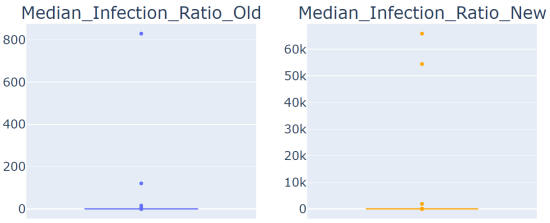
enough evidence of anomalies and can lead to many false positives.



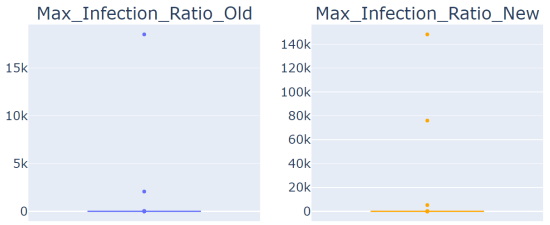
(a) Boxplot for the Standard Deviations of Infection Risk Factor fields.



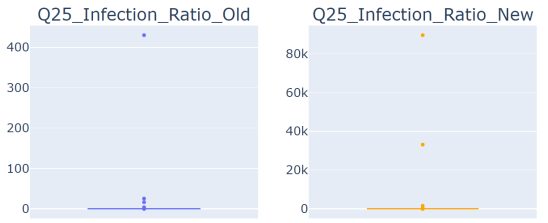
(b) Boxplot for the Means of Infection Risk Factor fields.



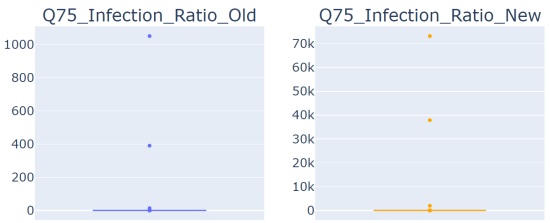
(c) Boxplot for the Medians of Infection Risk Factor fields.



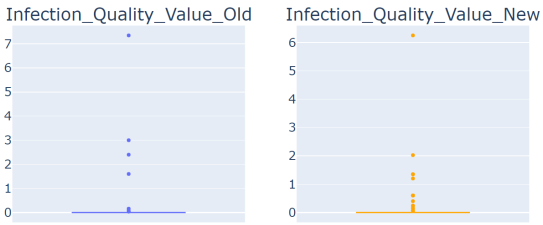
(d) Boxplot for the Maxes of Infection Risk Factor fields.



(e) Boxplot for the Q25s of Infection Risk Factor fields.



(f) Boxplot for the Q75s of Infection Risk Factor fields.



(g) Boxplot for the Infection Quality Values of Infection Risk Factor fields.

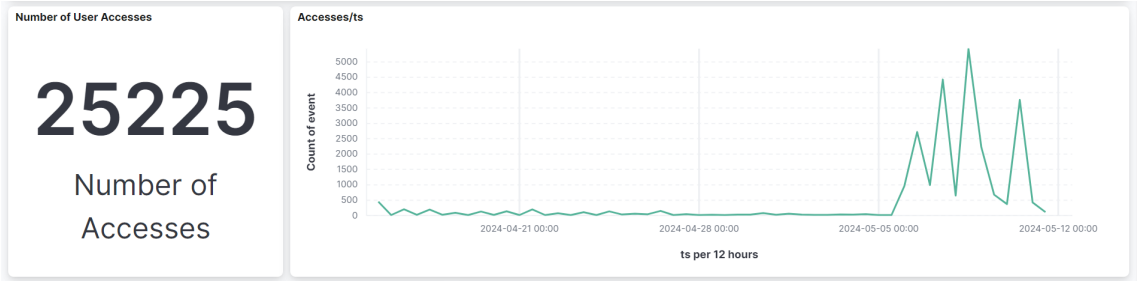
Figure 6.15: Boxplots for all fields related to the infected ratio.

The system was then ran for anomaly detection. The thresholds used to consider a machine an anomaly candidate were tuned the same way the weights were and the results are shown in Appendix A.1. From here on, the results for the time window which was described will be shown. Besides these, several other results from different time windows will be illustrated by displaying sections of the Kibana dashboard for each machine. Several values have been hidden for security reasons. Additionally, for simplicity, only the first case will show the full Excel summary table.

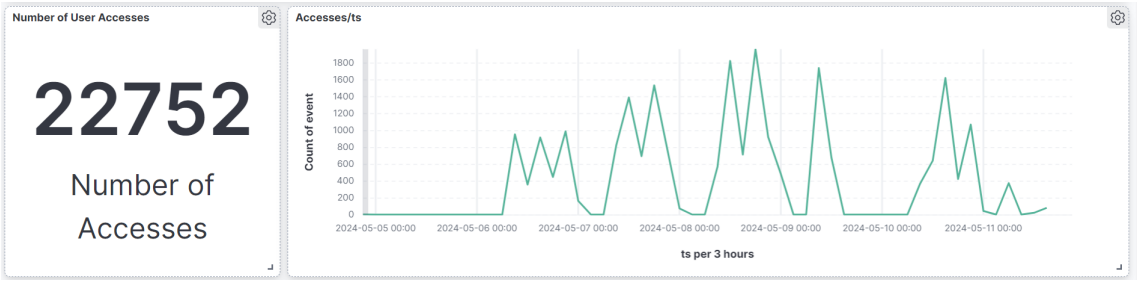
The first two anomalies that were detected are shown in Figures 6.16 and 6.17. Both of them are highly volumetric anomalies. The first anomaly is, however, only an Anomalous Recent Week candidate, since it clearly only shows infected behavior in a recent time frame. As for the second detection, the machine is candidate for both considered cases, displaying an anomalous behavior throughout the whole time frame. Figure 6.16a shows an intermediate value for the Infection Quality Value, while Figure 6.17a shows a high value, although the lower number of accesses.

Machine utilized in analysis	[redacted]
Associated name (MAC Address)	[redacted]
Employees associated to the machine	[redacted]
Total number of filtered accesses	2149
Flagged accesses with relevance	2149
Infection Quality Value	6.75
Detector Score	1
Detection Period	2024-04-15 to 2024-05-12
Report Date	[redacted]

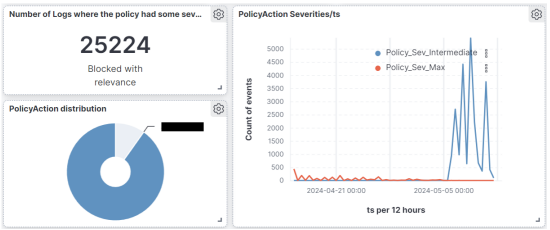
(a) Screenshot of the summary of the Excel report for the detected anomaly.



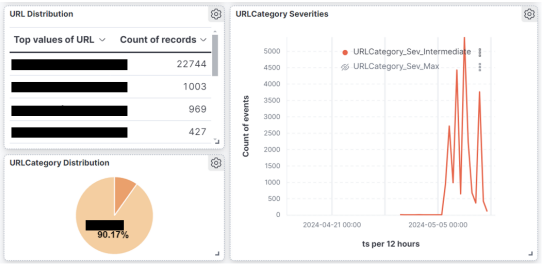
(b) Full time series for the detected anomaly, taken from the Kibana Dashboard.



(c) Most recent week's time series for the infected machine, taken from the Kibana Dashboard.



(d) Policy Action section of the Kibana Dashboard for all of the time frame.

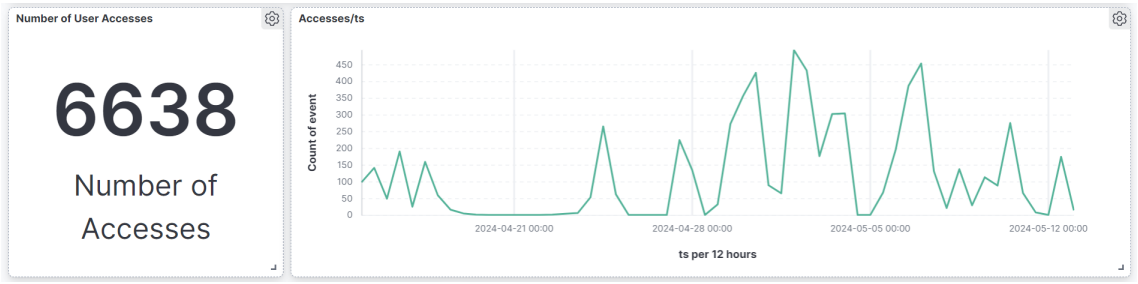


(e) URL section of the Kibana Dashboard for all of the time frame.

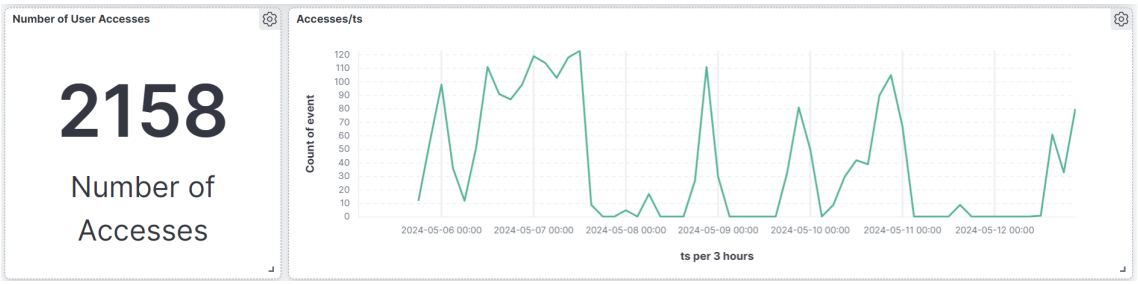
Figure 6.16: Example of a detected infected machine. In this case, the machine was only an anomalous recent week candidate. This is a very clear example of a highly volumetric anomaly.

Total number of filtered accesses	606
Flagged accesses with relevance	606
Infection Quality Value	20.833
Detector Score	1
Detection Period	2024-04-15 to 2024-05-12

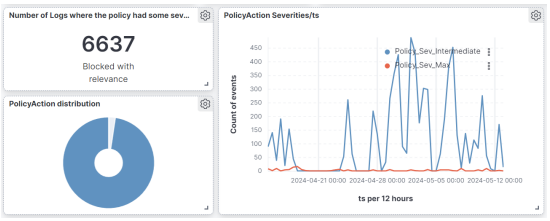
(a) Screenshot of the summary of the Excel report for the detected anomaly.



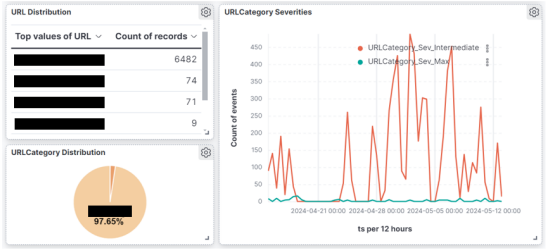
(b) Full time series for the detected anomaly, taken from the Kibana Dashboard.



(c) Most recent week's time series for the infected machine, taken from the Kibana Dashboard.



(d) Policy Action section of the Kibana Dashboard for all of the time frame.



(e) URL section of the Kibana Dashboard for all of the time frame.

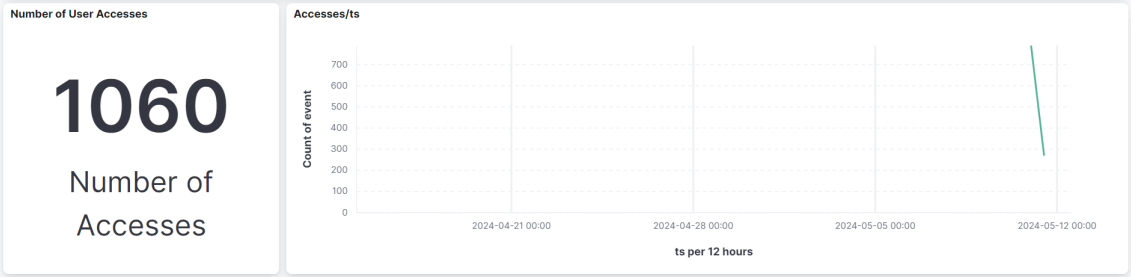
Figure 6.17: Example of a detected infected machine. In this case, the machine had both an anomalous history and recent week. This is another good example of a highly volumetric anomaly.

In the considered interval, one more detection was made. The outlier is displayed in Figure 6.18. For simplicity, only the accesses time series will be shown from now on. This anomaly may fall under one of two categories. It is either a False Positive, being just a data peak of that day (it could have been a website which was left open on the computer throughout the day which was continually flagged) or it is a machine that got infected and was immediately caught by other cybersecurity systems and platforms.

Besides this time window, several other detection periods were taken, leading to further outlier uncovering. Due to space limitations, not all anomalies will be shown, but some relevant cases will be described. In Figure 6.19, the outlier description shows a lower score and an intermediate Q,

Total number of filtered accesses	1060
Flagged accesses with relevance	1060
Infection Quality Value	0
Detector Score	1
Detection Period	2024-04-15 to 2024-05-12

(a) Screenshot of the summary of the Excel report for the detected anomaly.



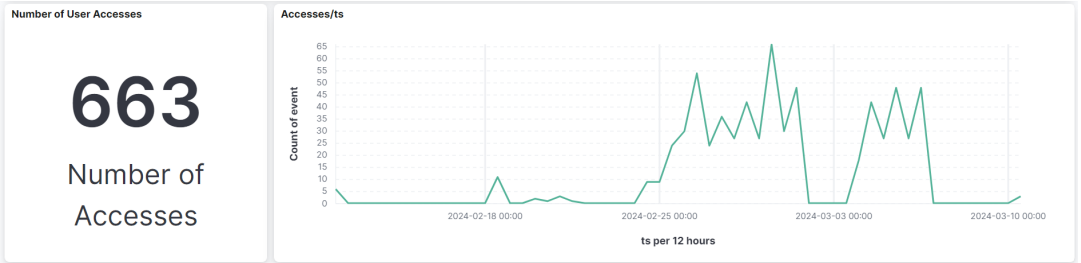
(b) Full time series for the detected anomaly, taken from the Kibana Dashboard.

Figure 6.18: Last example of the detected infected machines in the considered time frame. This detection was either a False Positive or a machine which was immediately detected by different systems.

having a very low number of accesses compared to other a infected machines. However, observing Figure 6.19c, the reason why the detector is acting on this machine is clear. Ignoring logs with the same timestamp (a reason for access correction), the time between logs seems to follow a pattern. Unlike the logs in Figure 6.19d, which refer to the machine described in Figure 6.16, which are very close to each other, these logs are more spaced apart. However, the variance of the time between logs is very small, greatly increasing the Infection Risk Factor.

Total number of filtered accesses	70
Flagged accesses with relevance	70
Infection Quality Value	9
Detector Score	0.86
Detection Period	2024-02-12 to 2024-03-10

(a) Screenshot of the summary of the Excel report for the detected anomaly.



(b) Full time series for the detected anomaly, taken from the Kibana Dashboard.

Figure 6.19: Example of a detected infected machine. Even though the score is lower, an infected behavior was detected. A small variance in Equation 5.1 led to the detection of the machine (Continues in next page).

> 2024-03-06 16:41:53.000
> 2024-03-06 16:01:02.000
> 2024-03-06 16:01:02.000
> 2024-03-06 16:01:02.000
> 2024-03-06 15:20:12.000
> 2024-03-06 15:20:12.000
> 2024-03-06 15:20:12.000
> 2024-03-06 15:00:06.000
> 2024-03-06 15:00:06.000

(c) Timestamps of some logs of the fourth example of detection.

> 2024-05-11 15:42:30.000
> 2024-05-11 15:42:05.000
> 2024-05-11 15:42:05.000
> 2024-05-11 15:42:05.000
> 2024-05-11 15:42:05.000
> 2024-05-11 15:42:05.000
> 2024-05-11 15:42:05.000
> 2024-05-11 15:42:05.000
> 2024-05-11 15:41:47.000

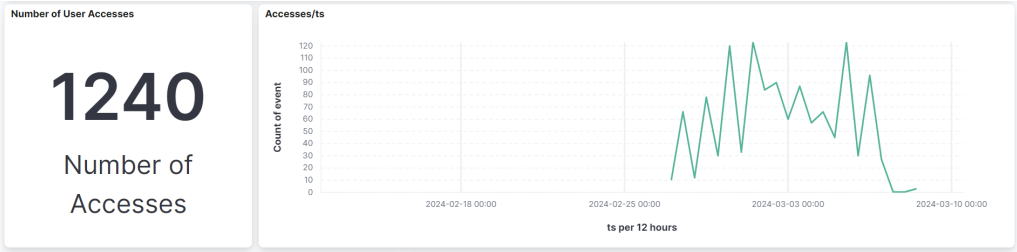
(d) Timestamps of some logs of the infected machine shown in Figure 6.16.

Figure 6.19: (Continuation of previous page) Example of a detected infected machine. Even though the score is lower, an infected behavior was detected. A small variance in Equation 5.1 led to the detection of the machine.

The last 2 cases are similar detections which are one week apart, illustrated in Figures 6.20 and 6.21. Using a sliding window configuration for the detectors built in this project aided on dealing with the dynamic behavior of the entities at hands. What before was not anomalous, quickly can become dangerous for the company. The example presented shows the flexibility the system has in dealing with the aforementioned problem as the detection shown in Figure 6.21 had not been detected in the previous week and, as its behavior worsened, the system managed to quickly capture this change and detected it for further investigation.

Total number of filtered accesses	149
Flagged accesses with relevance	149
Infection Quality Value	12.8
Detector Score	0.99
Detection Period	2024-02-12 to 2024-03-10

(a) Screenshot of the summary of the Excel report for the detected anomaly.

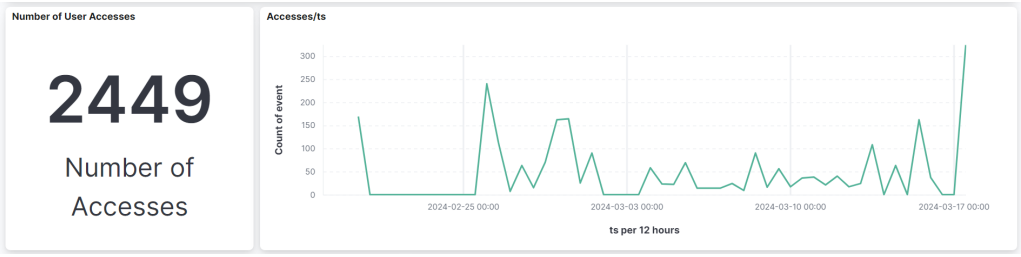


(b) Full time series for the detected anomaly, taken from the Kibana Dashboard.

Figure 6.20: Anomaly taken in the period of 2024-02-12 to 2024-03-10.

Total number of filtered accesses	422
Flagged accesses with relevance	422
Infection Quality Value	36
Detector Score	1
Detection Period	2024-02-19 to 2024-03-17

(a) Screenshot of the summary of the Excel report for the detected anomaly.



(b) Full time series for the detected anomaly, taken from the Kibana Dashboard.

Figure 6.21: Anomaly taken in the period of 2024-02-19 to 2024-03-17. We see that the sliding window configuration led to a new detection, considering the past recent week as historical data.

6.5 Discussion

Insider threat detection is one of the most important systems that enterprises have to employ nowadays. However, building such a system is a big challenge. This project aimed at developing the best system possible for insider threats from flagged web accesses, by using different technologies based on previously done works in the area to overcome several different challenges.

6.5.1 Data Pipeline

In order to detect anomalies, the data was extracted, integrated and processed by the Data Pipeline, which was created taking into account the several limitations and details that had to be considered. These included problems ranging from multi-categorical data to access collection to noisy data. The solution to the different issues had to be both simple and complete so that the equilibrium between quality of information, computational running time and existing RAM could be maintained. Along with the experts of Altice Portugal, the Data Pipeline ended up working globally very well, being fast and maintaining only the most relevant information. This was a product of several result evaluation phases that allowed for further data filtering and access bucketing corrections which permitted the process to take lesser and lesser time and the final product's quality to increase.

Besides filtering and data groupings, the extra steps taken for the IMD also proved to be successful. The IP map quickly and correctly attributed a machine to the respective user by using their IP at the time of the log. The infection features also proved to be a sufficiently good enough approach at defining the infection of a machine, greatly increasing the quality of data and simplifying the modeling pipeline immensely, making it so the process did not have to restrict itself on overly complex models.

6.5.2 User Detector

When it came to the detecting anomalous users, the UD system had to be robust, adaptable in time and in type of anomaly and could not be sensible to data peak surges of accesses. The various techniques used had to effectively filter out users that were not anomalous while trying to give more importance to possible anomalies. The detector that was developed ended up proving to be quite successful. The filters that the data passed through, including the peak smoothing and post recent anomaly filter, vastly helped on tuning the system and on avoiding a big number of False Positives. No detector is perfect; however, leaving the possibility of some users escaping detection open. Although it is better to capture fewer users who are almost certainly anomalous than to capture more users with a higher rate of false positives.

To find anomalous users, an initial statistical analysis was created to find "obvious" outlying users. The approach taken used several concepts, including correlation and weighting. The results proved to work well, revealing users who seemed to be anomalous.

Following this, we had to consider the possibility of there being an anomalous group of users in which they would all present consistently bad behavior. Inside the group, not all users might be of interest, but finding a group of seemingly atypical users would allow further investigation. In principle, the most anomalous users of this group would be the ones who presented the most consistently irregular behavior. Thus, finding an optimal clustering setup would be vital. Different algorithms and clustering evaluation methods were used to determine this. In the end, thanks to the peak smoothing previously done, a two cluster structure was determined to best describe the users, which clearly separated the baseline users from possibly anomalous users. Each of the clusters needed different treatment. While the anomalous group had to proceed into further detection, the baseline group needed a constant detection pipeline to investigate if any user was becoming anomalous or not.

For the anomalous group, basing off different works done in the area and the help of the security team at Altice Portugal, an ensemble of anomaly detection models was created to detect irregular users within the anomalous group. The data, though, still had to be further processed to facilitate the modeling. The high dimensionality of the dataset would make it so the models would not capture the intricacies of the users well. Therefore, the approach taken was to remove correlations, duplicated and no variance columns and to give bigger weights to columns considered more critical to the company. Besides this, optimizing the models was not possible, since we were in an unsupervised learning setup. To surpass this challenge, an artificial ground truth was created by injecting several different anomalies. With this, the models were optimized and evaluated. The final results ended up showing great promise, detecting users who seemed to be real anomalies, demonstrating the capturing power of the model ensemble. Further studies done showed that within the anomalous group there were many users who did not qualify to be anomalous, which additionally proves that the results were good. In addition, the sliding window configuration proved to work, as new anomalies were detected every time, not detecting already found outliers from older time windows. The anomaly characterization also proved to be sufficient, as it was

praised by the security team.

When it came to the baseline users, the biggest challenge was how to create a good enough baseline profile for each user that would allow comparison to new incoming data. To overcome this obstacle, a pure baseline profile was created for each user by finding anomalies within the user profile using an ensemble of simple distance-based models. The pure profiles were then matched against new days in a weekly setup. If, at the end of the week, the seven days gave an anomalous score to the user, then it meant that the user was starting to present dangerous behavior. The pipeline, again, proved to be successful on creating the baseline and comparing it to new days, even though no irregular user has yet been detected.

6.5.3 Infected Machine Detector

Describing the infection of a machine was the biggest task this detector had to tackle. Besides this, the detector had to prioritize faster detections over slower, more spread-out anomalies, like those of the UD. While the infection features effectively addressed the first problem, the issue of faster detections was solved by considering a smaller time window of only one month and splitting it into two: a historical dataset and a recent dataset. Comparing these two and the machine with other machines allowed for optimal infection identification. The results, although still under continual evaluation, just like the UD, show great promise in translating possible infection or on aiding UD detections. Similar to the UD, the sliding window configuration worked well and the anomaly characterization also proved to be a good anomaly description tool for the analysts.

6.5.4 Final remarks

The presented pipelines and results for the two detectors both showed great promise. Future tuning and feedback from the SOC team will surely lead the models to even better results.

One important note, though, is that, while the IMD fully focuses on detecting infected machines, of which only one type of anomaly can be studied, the UD must cover a big range of anomalies. Throughout the year, this detector only managed to really detect one type of anomaly—users with a high volume of consistent critical accesses. Possibly this is due to the fact that the dataset itself shows no other type of possible user. In other words, other types of anomalies have not happened yet. For example, a user that consistently had accesses with an existing ThreatSuperCategory, or that presented dangerous IPs to the company. Therefore, there is not enough information on how well the UD would behave under these conditions. It is expected that the statistical pipeline would be able to catch such users, while the model pipeline fully focus on the described detected anomaly type, thus showing the possible generalization capabilities of the model and the detection of previously unseen cases. But, reiterating, these different cases have not been observed yet.

Additionally, it is expected that most detections by the UD are simply unintentional perpetrators, as described in Section 1.1, where the users just need more training on cyber hygiene and what not to click on the Internet. However, the classification of the user is best left to the analyst team, who possess different tools to determine if the user is a real insider threat or not.

Chapter 7

Conclusion and Future Work

In this thesis, an anomaly detection system on logs regarding the flagged web accesses of users by a web proxy (Zscaler) was suggested and implemented for the goal of both finding possible insiders or infected machines who pose threats to Altice Portugal. The pipeline structure that was developed proved to be efficient and provided good results, being able to deal with a considerably big amount of log-based data in a fast way. The full process of extraction, integration and data processing for each detector takes approximately one hour, which can be considered efficient, given the great amount of data that the detectors handle. They also produced a highly informative and condensed dataset that, when fed to the modeling phase of detection, allowed for an easier process.

When it came to detecting consistently anomalous users (User Detector), dividing the detection phase into statistical, clustering, and ensembling showed great promise. The statistical analysis allowed for evident outliers to easily come to light. Filtering the peaks out of the data and clustering the users further allowed the abnormalities to be found, generating a group fully characterized by users that came out of the norm. Additionally, determining outliers by using an ensemble of three different detection algorithms with different determination techniques enabled for truly anomalous behavior to be detected.

In terms of finding infected machines (Infected Machine Detector), the fast and optimized linking of a user to its machine greatly eased the integration of data. Besides this, the creation of infection related features proved to be enough for a model itself, extremely simplifying the detection process. Furthermore, dividing the dataset into historical and recent data optimized the process greatly, allowing us to consider two different anomaly candidates.

The experimental results that came from both detectors exhibit that the suggested pipeline managed to capture highly confident detections which were confirmed to be cases of interest by the SOC team. The system proved to be capable of adapting itself in time by using a sliding window configuration and is robust. Along with this, the Excel report and the Kibana dashboard for each detected outlier characterization managed to greatly help the analysis of each anomaly thoroughly, easing the analyst's job of finding, explaining and determining if said anomaly poses or not a danger to the company and what the next step should be.

The system, however, has its own limitations. Due to a possible lack of data, many possible

cases have not been found yet, e.g., IP related anomalies. This does not mean that the system would not be able to find them, as it should be able to find previously unseen anomalies. One possible future task would be to create new tests or even completely separate, e.g., the IP features, and create specialized detectors to detect any suspicious IP behavior. Besides this, the two detectors share some found anomalies. Thus combining the information from both detectors to better determine the root of the outlier—infected machine or possible insider—would help on further characterizing each anomaly.

One big limitation the system also has is its lack of model training on real anomalies. Regarding the UD, training the final detection phase models only on synthetic anomalies limits this ending step to focus mostly on behaviors which were considered to be anomalous, possibly leaving out real zero-day anomalies that could be present in the data and not being detected. Also, relying only on outliers generated by hand might not fully represent the reality of an anomalous behavior. This issue could be corrected by having access to a controlled environment in which one could simulate real anomalous behavior and then use it to train the models. In a similar way, having more examples of real infected machines would help fine-tune the pipeline further. In addition, having a real baseline to train the models would allow real precision and recall calculations for each detector, leading to a bigger trust on the results.

Aside from this, there are plans to apply the same model pipeline on a different dataset regarding VPN accesses of users. Some work was started on this topic, as this phase was initially planned to be on this project, but the initial data analysis and exploration proved to be more complex than expected. The data presented a lot of intricacies and details that would take a long time to treat in the data processing phase. Thus, it was put aside for the meantime.

As this dissertation stands, the project is continually being optimized with new feedback from Altice Portugal's DCY team coming in regularly and is being put into production, since the project's results held great interest to the DCY team. The continuous feedback helps in fine-tuning the parameters of the system and diagnosing which parts of the model need improvement, thereby refining the system that was initially set up. This iterative process ensures that the model remains adaptable to the dynamic needs of the enterprise, enhancing its overall performance and reliability.

References

[1] Aggarwal, C. C., Hinneburg, A. & Keim, D. A. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the 8th International Conference on Database Theory (ICDT)*, pages 420 – 434, 2001.

[2] Aldairi, M., Karimi, L., & Joshi, J. A trust aware unsupervised learning approach for insider threat detection. In *Proceedings of the 20th IEEE International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 89–98. IEEE, 2019.

[3] Alomiri, A., Mishra, S. & AlShehri, M. Machine learning-based security mechanism to detect and prevent cyber-attack in IoT networks. *International Journal of Computing and Digital Systems*, 16(1):645–659, 2024.

[4] Alsowail, R. A. & Al-Shehari, T. Techniques and countermeasures for preventing insider threats. *PeerJ Computer Science*, 8:938, 2022.

[5] Ayadi, A., Ghorbel, O., Obeid, A. M., & Abid, M. Outlier detection approaches for wireless sensor networks: A survey. *Computer Networks*, 129:319–333, 2017.

[6] Babu, S. Detecting anomalies in Users-An UEBA approach. In *Proceedings of the International Conference on Industrial Engineering and Operations Management* , pages 863–876, 2020.

[7] V. Bandari. Enterprise data security measures: A comparative review of effectiveness and risks across different industries and organization types. *International Journal of Business Intelligence and Big Data Analytics*, 6(1):1–11, 2023.

[8] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[9] Breunig, M.M., Kriegel, H., Ng, R.T. & Sander, J. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, page 93–104. Association for Computing Machinery, 2000.

[10] Chandola, V., Banerjee, A., & Kumar, V. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009.

[11] Daswani, N. & Elbayadi, M. *The Yahoo Breaches of 2013 and 2014*, pages 155–169. 2021.

[12] Datta, J., Dasgupta, R., Dasgupta, S., & Reddy, K. R. Real-time threat detection in UEBA using unsupervised learning algorithms. In *Proceedings of the 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*, pages 1–6, 2021.

[13] Donoho, D.L. Breakdown properties of multivariate location estimators. Technical report, Harvard University, Boston, 1982.

[14] Erfani, S. M., Rajasegarar, S., Karunasekera, S., & Leckie, C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[15] Fix, E. & Hodges, J.L. Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical report, USAF School of Aviation Medicine, Randolph Field, 1951.

[16] M. D. C. F. R. Guerra. Internal hacking detection using machine learning. Master's thesis, Faculdade de Ciências da Universidade de Lisboa (Portugal), 2020.

[17] Handa, A., Sharma, A. & Shukla, S. K. Machine learning in cybersecurity: A review. *WIREs Data Mining and Knowledge Discovery*, 9(4), 2019.

[18] D. M. Hawkins. *Identification of outliers*, volume 11. Chapman and Hall, London, 1980.

[19] Hubert, M. & Debruyne, M. Minimum covariance determinant. *WIREs Comp Stat*, 2:36 – 43, 2010.

[20] IBM. IBM cost of a data breach report 2023. Technical report, IBM, 2023. `https://www.ibm.com/reports/data-breach`, Accessed November 2023.

[21] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[22] I. Jolliffe. *Principal Component Analysis*. John Wiley & Sons, Ltd, 2005.

[23] Khaliq, S., Tariq, Z. U. A., & Masood, A. Role of user and entity behavior analytics in detecting insider attacks. In *Proceedings of the International Conference on Cyber Warfare and Security (ICCWS)*, pages 1–6, 2020.

[24] Khan, M. Z. A., Khan, M. M., & Arshad, J. Anomaly detection and enterprise security using user and entity behavior analytics (UEBA). In *Proceedings of the 3rd International Conference on Innovations in Computer Science & Software Engineering (ICONICS)*, pages 1–9. IEEE, 2022.

[25] Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., & Srivastava, J. *A comparative study of anomaly detection schemes in network intrusion detection*, pages 25–36. Society for Industrial and Applied Mathematics., 2003.

[26] Le, D. C., & Zincir-Heywood, N. Anomaly detection for insider threats using unsupervised ensembles. *IEEE Transactions on Network and Service Management*, 18(2):1152–1164, 2021.

[27] Legg, P. A., Buckley, O., Goldsmith, M., & Creese, S. Automated insider threat detection system using user and role-based profile assessment. *IEEE Systems Journal*, 11(2):503–512, 2015.

[28] Li, X., Deng, S., Li, L. & Jiang, Y. Outlier detection based on robust Mahalanobis distance and its application. *Open Journal of Statistics*, 9(1):15 – 26, 2019.

[29] Lin, W. C., Ke, S. W., & Tsai, C. F. CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems*, 78:13–21, 2015.

[30] Liu, F.T., Ting, K. & Zhou, Z. Isolation forest. In *Proceedings of the 8th IEEE Internation Conference of Data Minining (ICDM)*, pages 413 – 422, 2009.

[31] Liu, S., Maljovec, D., Wang, B., Bremer, P. T., & Pascucci, V. Visualizing high-dimensional data: Advances in the past decade. *IEEE transactions on visualization and computer graphics*, 23(3):1249–1268, 2016.

[32] Madhulatha, T. S. An overview on clustering methods. *OSR Journal of Engineering*, 2(4):719 – 725, 2012.

[33] J. McQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley symposium on mathematical statistics and probability*, 1(14):281 – 297, 1967.

[34] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[35] Nocera, F., Demilito, S., Ladisa, P., Mongiello, M., Shah, A. A., Ahmad, J., & Di Sciascio, E. A user behavior analytics (uba)- based solution using LSTM neural network to mitigate ddos attack in fog and cloud environment. In *Proceedings of the 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, pages 74–79, 2022.

[36] Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. Deep learning for anomaly detection: A review. *ACM Computing Surveys*, 54(2):1–38, 2021.

[37] Ramaswamy, S., Rastogi, R. & Shim, K. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, page 427–438, 2000.

[38] Raut, M., Dhavale, S., Singh, A., & Mehra, A. Insider threat detection using deep learning: A review. In *Proceedings of the 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 856–863. IEEE, 2020.

[39] P. Rousseeuw and V. Yohai. Robust regression by means of s-estimators. In *Robust and Nonlinear Time Series Analysis*, pages 256–272, 1984.

[40] Rousseeuw, P.J. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871 – 880, 1984.

[41] M. N. Saffaf. Chapter 6 - malware analysis. In *Virtualization for Security*, pages 145–189. Syngress, Boston, 2009.

[42] Salitin, M. A., & Zolait, A. H. The role of user entity behavior analytics to detect network attacks in real time. In *Proceedings of the International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. IEEE, 2018.

[43] Schölkopf, B., Williamson, R.C., Smola, A., Shawe-Taylor, J., & Platt, J. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[44] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P. & de Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[45] Shashanka, M., Shen, M. Y., & Wang, J. User and entity behavior analytics for enterprise security. In *Proceedings of the IEEE International Conference on Big Data (Big Data)*, pages 1867–1874, 2016.

[46] Shinde, P. P. & Shah, S. A review of machine learning and deep learning applications. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–6, 2018.

[47] Shyu, M. L., Chen, S. C., Sarinnapakorn, K., & Chang, L. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE foundations and new directions of data mining workshop*, pages 172–179, 2003.

[48] Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N., & Robinson, S. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. *CoRR*, abs/1710.00811, 2017.

[49] Van der Maaten, L. & Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[50] Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., & Li, K. AI^2: training a big data machine to defend. In *Proceedings of the IEEE 2nd international conference on big data security on cloud (BigDataSecurity), IEEE international conference on high performance and smart computing (HPSC), and IEEE international conference on intelligent data and security (IDS)*, pages 49–54, 2016.

[51] Verizon. 2023 Data Breach Investigations Report. Technical report, Verizon, 2023. https://www.verizon.com/business/resources/reports/2023-data-breach-investigations-report-dbir.pdf, Accessed October 2023.

[52] Xu, W., Grant, G., Nguyen, H., & Dai, X. . Security breach: The case of TJX companies, inc. *Communications of the Association for Information Systems*, 23(1):31, 2008.

[53] Zeadally, S., Yu, B., Jeong, D. H. & Lily Liang. Detecting insider threats: Solutions and trends. *Information Security Journal: A Global Perspective*, 21(4):183–192, 2012.

[54] Zhang, M., Lu, S., & Xu, B. An anomaly detection method based on multi-models to detect web attacks. In *Proceedings of the 10th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 404–409. IEEE, 2017.

[55] Zhao, Y., Nasrullah Z. & Li, Z. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

[56] Zhu, Y., Zimmerman, Z., Shakibay Senobari, N., Yeh, C. C. M., Funning, G., Mueen, A., Brisk, P. & Keogh, E. Exploiting a novel algorithm and GPUs to break the ten quadrillion pairwise comparisons barrier for time series motifs and joins. *Knowledge and Information Systems*, 54:203–236, 2018.

[57] Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

Appendix A

Hyperparameter Information

A.1 Infected Machine Detector

The optimal combination of weights and the thresholds that were determined to be the best for the k-Nearest Neighbors and Euclidean distance models by using previously found infected machines are shown in Tables A.1 and A.2.

Table A.1: Feature weights used in the Infection Dataset.

Feature	**Weight**
Stdev_Infection_Ratio	0.1
Mean_Infection_Ratio	0.2
Median_Infection_Ratio	$\frac{0.2}{3}$
Max_Infection_Ratio	0.2
Q25_Infection_Ratio	$\frac{0.2}{3}$
Q75_Infection_Ratio	$\frac{0.2}{3}$
Total_Infection_Ratio	0.3
Condition $\longrightarrow \sum_f w_f = 1$	

Table A.2: Optimal values for all hyperparameters and weights considered in the Infected Machine Detector

Parameter	**Value**
kNN (Percentile)	0.9
Euclidean Distance	10000

A.2 User Detector - Clustering

The hyperparameter values which were explored during the Clustering Pipeline for the k-Means and Spectral Clustering along with the optimal results that were obtained can be observed in Tables A.3 and A.4.

Table A.3: Parametrization used for the k-Means algorithm and respective optimal values.

Parameter	**Objects**	**Optimal Value**
n_clusters	{2,3,4,...,10,11}	2
init	default 'k-means++'	'k-means++'
n_init	default 'auto'	'auto'
max_iter	default 300	300
tol	default 0.0001	0.0001
random_state	42	42
algorithm	default 'lloyd'	'lloyd'

Table A.4: Parametrization used for the Spectral Clustering algorithm and respective optimal values.

Parameter	**Objects**	**Optimal Value**
n_clusters	{2,3,4,...,10,11}	2
eigen_solver	default 'None'	'None'
n_components	default 'None'	'None'
n_init	default 10	10
gamma	default 1.0	Not used
affinity	{'rbf', 'nearest_neighbors'}	'nearest_neighbors'
random_state	42	42
n_neighbors	default 10	10
eigen_tol	defaut 'auto'	'auto'
n_jobs	-1	-1

A.3 User Detector - Anomalous Group

Table A.5 illustrates the various group weight values that were explored and their conditions for the anomalous group. These weights were then used to make the anomalous group models. The optimal weights for each model are shown in Table A.6. The hyperparameters which were were explored during this phase, along with their optimal values after training, are shown in Tables A.7, A.8 and A.9.

Table A.5: Group weights combinations parameters used to train the models.

Weight Group	**Values**
w_{Threats}	$\{0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$
w_{Policy}	$\{0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$
w_{Accesses}	$\{0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$
$w_{\text{URLCategory}}$	$\{0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$
w_{URLClass}	$\{0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$
$w_{\text{ClientExternalIP}}$	$\{0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$
w_{ClientIP}	$\{0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$
Condition 1 $\longrightarrow \sum_G w_G = 1$	
Condition 2 $\longrightarrow w_{\text{Threats}}, w_{\text{Policy}}, w_{\text{Accesses}} \geq w_{\text{URLCategory}} > w_{\text{URLClass}} > w_{\text{ClientExternalIP}} > w_{\text{ClientIP}}$	

Table A.6: Optimal weight combination for each model in the anomalous group.

Weight Group	**OCSVM**	**iForest**	**LOF**
w_{Threats}	0.3	0.15	0.25
w_{Policy}	0.35	0.2	0.2
w_{Accesses}	0.15	0.3	0.2
$w_{\text{URLCategory}}$	0.08	0.15	0.15
w_{URLClass}	0.06	0.1	0.1
$w_{\text{ClientExternalIP}}$	0.04	0.08	0.06
w_{ClientIP}	0.02	0.02	0.04

Table A.7: Parametrization of the hyperparameters of the OCSVM model along with the optimal value.

Parameter	**Objects**	**Optimal Value**
Kernel	{'rbf', 'sigmoid' }	'rbf'
Nu	{0.1, 0.2, 0.3,...,0.9}	0.3
gamma	{'scale', 'auto', 0.01, 0.25, 0.5, 0.75, 1}	1
coef0	default 0.0	0.0
max_iter	default -1	-1
tol	default 0.001	0.001
cache_size	default 200	200
shrinking	default True	True
contamination	{0.01, 0.03,...,0.09} and {0.10, 0.11, 0.12,...,0.17}	0.17

Table A.8: Parametrization of the hyperparameters of the Isolation Forest model along with the optimal value. The values between square brackets represent number intervals, not sets.

Parameter	**Objects**	**Optimal Value**
n_estimators	{100,101,102,...,1000} = range(100,1001,1)	100
max_samples	default 'auto'	'auto'
contamination	[0.01, 0.17]	0.17
max_features	{5,6,7,...,49,50} = range(5,51,1)	5
bootstrap	{'True', 'False'}	'True'
n_jobs	-1	-1
behaviour	{'old', 'new'}	'new'
random_state	433	433

Table A.9: Parametrization of the hyperparameters of the Local Outlier Factor model along with the optimal value (Continues in next page).

Parameter	**Objects**	**Optimal Value**
n_neighbors	{int($\sqrt{\text{len(dataset)}}$), 10, 15, 20, 30, 50, 75, 100}	30
algorithm	default 'auto'	'auto'
leaf_size	{30, 75, 150, 250, 500}	30

Table A.9: (Continuation of previous page) Parametrization of the hyperparameters of the Local Outlier Factor model along with the optimal value.

metric	default 'minkowski'	'minkowski
p	{1, 2, 3}	3
metric_params	default 'None'	'None'
contamination	{0.01, 0.03,...,0.09} and {0.10, 0.11, 0.12,...,0.17}	0.17
n_jobs	-1	-1
novelty	default 'False'	'False'

A.4 User Detector - Baseline Group

When it came to the baseline group, the weights that were experimented to train a good threshold are also expressed in Table A.5. In parallel, the thresholds for the kNN and Mahalanobis distance model are shown in Table A.10. No optimal values are given since each baseline user has its own optimal value given the injected anomalies.

Table A.10: Different thresholds tried for the kNN and Mahalanobis models for the incoming daily data.

Model	**Objects**
kNN	{0.70, 0.71, 0.72, 0.73,...,0.94, 0.95}
Mahalanobis	{0.70, 0.71, 0.72, 0.73,...,0.94, 0.95}