Artificial Intelligence for SOC
User and Entity Behaviour Analysis
Eurotux Internship Report

Ricardo Alves Oliveira
(ricaoimp@gmail.com)

August 2023

Abstract

This document encapsulates my internship experience at Eurotux, where I undertook a project aimed at enhancing the Security Operations Center (SOC) capabilities through the integration of a User and Entity Behavior Analytics (UEBA) tool. Initially, an exploration of existing UEBA solutions was conducted, followed by a comprehensive presentation of the available options. However, due to specific requirements, a decision was made to design and develop a custom UEBA tool from scratch. This report outlines the research, development process, challenges faced, and the positive outcome achieved. The resulting tool has the potential to significantly contribute to strengthening the SOC's ability to detect and respond to anomalous user and entity behavior, improving further the organization's overall cybersecurity posture.

Contents

List of Figures

Chapter 1

Introduction

During my internship at Eurotux, I had the opportunity to engage in a significant project focused on augmenting the capabilities of the Security Operations Center (SOC) through the integration of a User and Entity Behavior Analytics (UEBA) tool. This endeavor was rooted in a thorough exploration of existing UEBA solutions, which was followed by a comprehensive evaluation and presentation of the available options.

As the project progressed, evolving requirements led to a decision to develop a custom UEBA tool from scratch. This report provides an overview of my journey throughout this internship, detailing the initial research phase, the subsequent development process, the challenges encountered, and the notable outcomes achieved. By delving into the intricacies of creating a tailored UEBA tool, this report highlights the value of adaptability and innovation in addressing unique cybersecurity demands.

Throughout this document, I aim to shed light on the evolution of my understanding, skills, and contributions during this enriching internship opportunity.

Chapter 2

Background and Context

As an intern at Eurotux, it was essential for me to grasp the foundational concepts surrounding User and Entity Behavior Analytics (UEBA) to effectively contribute to the integration project. This chapter provides an in-depth exploration of what UEBA entails and highlights the benefits that such technology brings to the realm of cybersecurity.

2.1 What is an UEBA

In the landscape of modern cybersecurity, UEBA emerges as a crucial tool for identifying anomalous user and entity behavior within an organization's digital environment. UEBA leverages advanced analytics, machine learning, and data correlation techniques to detect patterns that might indicate potential security threats or insider risks. Unlike traditional security solutions that rely on predefined rules, UEBA thrives on adaptive learning and contextual understanding, enabling it to recognize deviations from normal behavior and adapt to evolving threats.

As an intern, I had the task of not only understanding the technical details of UEBA but also grasping its larger role in security. Talking to mentors and researching, I learned that UEBA goes beyond just watching individual events. It looks at how users and things act over time, which helps the SOC team spot small changes that could signal a possible breach or unauthorized actions. This could range from small unusual events to detecting social attacks.

2.2 Benefits of an UEBA

Since UEBA is becoming an indispensable asset for any modern Security Operations Center (SOC), it is important to understand its advantages and uses. One significant advantage is the ability to enhance threat detection and response capabilities. UEBA provides a more proactive approach to cybersecurity by identifying both known and unknown threats that might bypass traditional security measures. This aligns with the commitment to staying ahead of emerging threats and safeguarding the organization's digital assets.

Moreover, UEBA fosters a better understanding of user and entity behavior, enabling the SOC team to distinguish between legitimate and malicious activities. This context-awareness is pivotal, as it minimizes false positives and reduces the noise that can overwhelm incident response. As an intern, I realized that this aspect could significantly help streamline the workload of the SOC team, allowing them to focus on genuinely critical incidents.

Chapter 3

Research Phase

The initial phase of my internship project involved a comprehensive exploration of existing User and Entity Behavior Analytics (UEBA) solutions available in the market, as well as open source projects already being developed. This chapter documents the process of researching, analyzing, and presenting the various UEBA tools.

3.1 Exploration of Existing UEBA Solutions

The journey began with an in-depth survey of the landscape of UEBA solutions. This involved researching established vendors, open-source alternatives, and emerging projects in the field. My goal was to understand the diverse range of tools available, their key features, and their potential fit within Eurotux's security infrastructure.

Taking a first look into the market leaders, a couple options come to the table:

- Splunk - UBA Standalone
- FortiSIEM - SIEM Tool with UEBA Capabilities
- Elastic Stack - SIEM Platform with UEBA Capabilities
- Cynet - XDR Platform with UEBA Capabilities

Furthermore, when starting this project, my atention was brought to an open source alternative called 'OpenUBA'. This is a community project that focus on sharing UBA models for security teams without the price of enterprise solutions.

3.2 Comparative Analysis of UEBA Tools

The main point of this phase of this project is to understand the next step, either it be to implement an existing solution, focus on an OS alternative or to start developing a tailored solution. When taking a further look into the enterprise tools mentioned before, all of them come with similar base features, varying, mostly, when it comes to the systems they are integrated with. Not only that, but as expected, all enterprise solutions come with an associated cost. With this in mind, the main features of enterprise solutions are:

- Possibility to integrate with a SIEM
- Threat detection and prediction
- Continuous monitoring
- Machine learning technology

When referring to the 'OpenUBA', the situation was a little different. Upon a deeper analysis of the project repository a lot of problems came to light. The project was mostly incomplete and abandoned, the source files were messy and mostly hard coded and, more important, no real machine model for behaviour analysis seems to be even available. This presented the first problem as, upon starting this internship, this would be the preferred tool to use and expand to fit the needs of the security team. Upon righting a review about the state of the project and presenting it to my mentor, it became clear that this specific project had to be abandoned.

The final alternative is the development of a custom UEBA tool from scratch. This allows for the development of features tailored towards the security team at the cost of development time. In addition, the models created are fitted to real data from real services that need to be monitored. This could also benefit the company as the development of an advanced internal security tool could, when fully tested and stable, be sold as a service or tool to clients. Even if it's not the main focus, it can be important to consider when trying to understand the cost/benefit of developing a tool from scratch.

3.3 Reflections on the Available Options

The culmination of the research phase involved preparing a comprehensive pitch of the evaluated UEBA tools. The objective was to provide a clear understanding of the pros, cons, and unique features of each UEBA solution, enabling the security team to make an informed decision on how to move forward.

On one side, the enterprise solutions bring a more well rounded software, general purposed and backed by global companies. Its implementation, whilst the process is relatively simple, could imply the migration of existing services and incident responses to new tools. This change can bring the need to retraining of employees, significant increase in security costs and dependence in external services. This can be a major drawback, especially in the beginning phases, where it can be unclear the impact of such tool on the overall performance of a SOC.

In contrast, developing a new UEBA system comes with solutions more specific to the needs of the team, a faster development cycle and a lower cost. Additionally, creating a new product allows taking into account the technologies already used by the company, as well as the feedback from the team, allowing to reduce previously mentioned retraining and migration needs. The downside comes with the increase in the time needed to develop and implement the desired tool, as it implies a deeper understanding of project development, artificial intelligence/machine learning and cyber security.

Finally, when in discussion with my mentor and the head of the security operations team, it was decided to take the latter alternative, as it made more sense for the needs of the company and the team themselves. Thus shifting the focus of my project, from researching and integrating a UEBA system, to developing a complete and versatile UEBA tool.

Chapter 4

System Requisites

The first step of this project was to talk to the security team to understand the needs and expectations for this project. These were, however, expanded during development as to meet new ideas and expectations. A simplified list of the requisites can be found below:

- **Detection and Alert Signaling:** The system has to be able to detect and signal suspicious activity
- **Scalability:** The system has to be capable to handle multiple users/services
- **Model Diversity:** The system has to be able to manage different types of models
- **Parallel Model Prediction:** The system should be capable to handle multiple models simultaneously for predicting user behaviour
- **Alert Classification:** The system should be able to classify alerts according to the danger they represent
- **Resilience and Persistence:** The system should have persistent state recovery
- **Flexible Integration:** The system should support custom integrations
- **User and Model Management:** There should be an interface for the management of users and models
- **Feedback Mechanism:** There should be a feedback system to allow for reviewing predicted risks

Having a solid grasp of what the system required provided me with a better understanding for the subsequent phases. With this in mind I outlined the path ahead, ensuring a structured and purposeful development process. The upcoming chapters will offer an in-depth exploration of how I began assembling the various components that collectively constitute the project. I will provide detailed insights into the practical implementation of each identified requirement, along with the challenges I encountered and the strategies I employed to overcome them.

Chapter 5

System Architecture

This chapter provides an insight into the three main components that constitute the architecture of the system. Each component plays a distinct role in the operation of the system, contributing to its overall functionality.

5.1 Models and Trainers

The "Models and Trainers" serves as the cornerstone of the system's predictive capabilities. Models, in essence, are algorithms designed to make predictions based on input data, in this case, logs or alerts generated by the users/entities. These models are trained using real data from the desired systems, allowing them to learn their patterns and usual behaviours. The trained models are then used to evaluate incoming logs/alerts and predict potential risks and escalate them accordingly.

Training models can be complex, as their performance can't be immediately evaluated. While patterns can be identified by them, it becomes unclear when looking at "normal" data how the system will behave under "attacks". Because of this, testing them involves having them classify alerts, keeping close attention to the results. This can be replaced with custom datasets or systems, designed specifically to test unusual behaviour of a given system and evaluating the outcome.

This component essentially empowers the system with the ability to anticipate and respond to security-related incidents. While only one of the three exisiting components, this models are the main focus of this project.

5.2 UEBA Server

At the heart of the architecture lies the "UEBA Server", responsible for running and managing the entire system, from the users and entities to the models responsible for escalating the alerts. By serving as a centralized hub, the UEBA Server facilitates seamless communication and collaboration among different parts of the system. It ensures that data flows efficiently and decision-making processes are streamlined, enabling effective monitoring and response to security events.

This component also presents a web interface to facilitate the team to manage the system, as well as a route for alerts to get posted to the server and consequently evaluated. All specific features and routes will be further explained on the "Development Phase" chapter.

5.3 Integrations

The ”Integrations” segment forms a vital link between the system and external data sources. These integrations are designed to collect logs and alerts from various sources. By doing so, they provide the system with essential data required for analysis. This approach allows for more varied sources of data, as well as the possibility to integrate with existing systems and/or services.

An important example of this component and its use, is the integration of the project with a SIEM. This can be done in two different moments, when the alerts are generated and when they are escalated. On the first one, the integration should ensure that generated alerts by the SIEM tool get sent to the 'UEBA server' to be analyzed by the machine learning models. Afterwards, if alerts are escalated, an integration can be implemented to send the new data back to the SIEM tool. This allows for a better 'noise filter' amongst the big mass of alerts sent to the SOC, not only bringing light to dangerous alerts, but also to activity that deviates from the norm.

5.4 System Overview

Summing up, the architecture is organized into these three distinct components, each fulfilling a pivotal role in the system's operation. Integrations responsible for the handling of alerts between the system and the external services. The server orchestrating all the components, their interactions and the management of agents and models. Finally the models, learning and classifying alerts based on the user and entity behaviour. Below, there's a visual representation of this desired interactions.

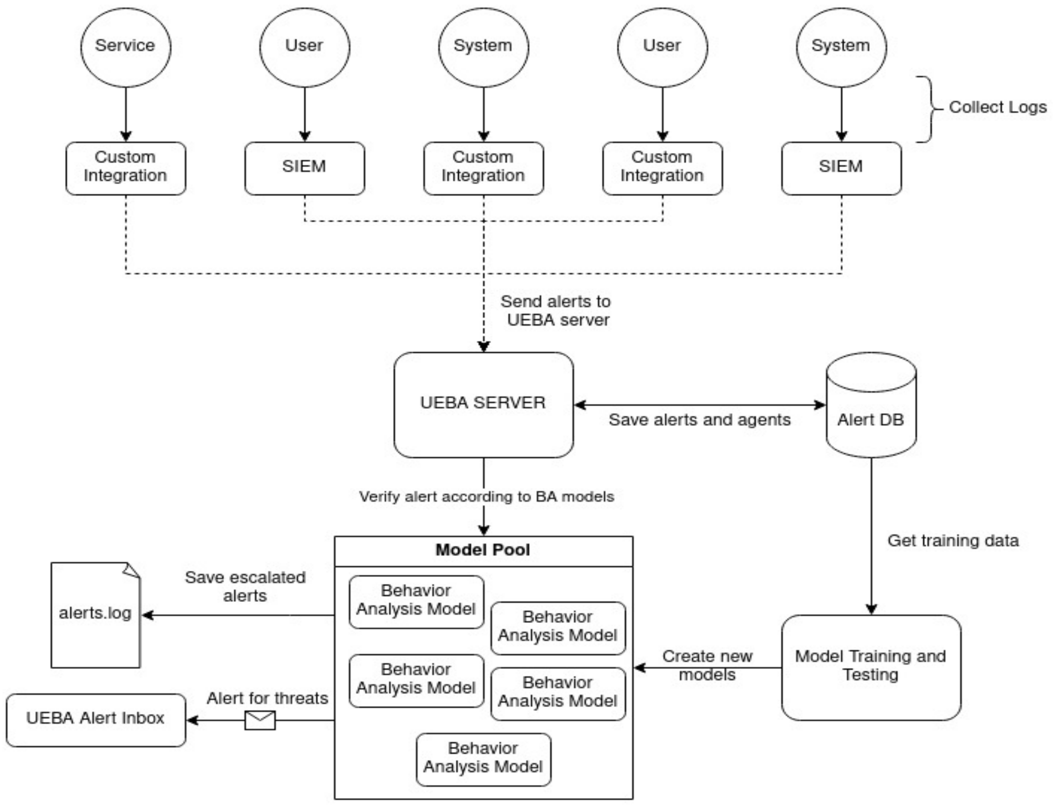


Figure 5.1: General Workflow

Chapter 6

Development Phase

This chapter takes a deeper look into development phase, taking into account steps taken to transition from concepts to a tangible and operational User and Entity Behavior Analytics (UEBA) system, as well as the choices that had to be taken during development. The following sections delve into the specific aspects that marked development, such as data collection, model training, interface development, and system integrations.

6.1 Gathering Data

To start development, a fundamental requisite was gathering data for training. This posed the first major problem for three reasons:

- **Data Security:** The data needed to train models contains, generally, sensitive data, such as IPs, users activity, accessed domains, etc.
- **Data Volume:** To produce accurate behaviour models, large volumes of past data are required so that patterns can be correctly identified
- **Data Sources:** Sources not only need large amounts of data, they need to be used somewhat 'frequently' in order to test the results produced

With this in mind, a couple options were available. The first one, real systems with real users. Whilst it provided the best data, as it had a great history and in depth information, it is also the less secure option. Another option is special machines program to produce large amounts of data. This became a strong contender as it presented the security and the amounts of data needed, but it still had a problem: the data wasn't real. This, while it usually wouldn't be a problem, prevents models from learning real behaviour adequate to real users and thus making the models obsolete in real settings.

In conversation with my mentor, a better solution came to light. Use test servers. While the usage levels are less than ideal, using this service's data grants the best overall option. It originates from real users, there is a decent amount of past logs and, while less secure than fake data, the security trade off was worth it.

6.2 Environment Setup

After the data source was decided, a development server was setup to continue the project development. This consisted of a machine, running 'Rocky Linux' version 8.8, that I'd operate through ssh. While the server configuration needed wasn't much, during this phase, managing it and its resources allowed me to deepen my knowledge in Linux systems.

This included tasks like:

- **Firewall Management:** This included managing ports and allowed IPs;
- **Network Monitoring:** Usage of tools like 'tcpdump' and 'nmap';
- **Resource Allocation:** Responsible for distributing server resources such as CPU, memory, and disk space to optimize performance, specially given the circumstances;
- **Package Management:** Utilizing package managers to install, update, and manage software packages;
- **Log Analysis:** Monitoring and analyzing system logs to identify anomalies and potential issues;
- **Service Monitoring:** Keeping a watchful eye on running services to ensure uninterrupted availability.
- **...**

Additionally, a GitHub repository was created for version control. The project also contains a documentation folder called 'docs', where all manuals and workflows can be found, and a 'Makefile' that allows building and running the project.

When it comes to the technologies used for this project, the source code was written in Python3[6], with a development server hosted using the Flask[2] framework and an interface written in HTML and Bootstrap[1]. As for the machine learning algorithms SKLearn[7] was used during the trainig and evaluting step, being complemented by Numpy[3], Pandas[4] and Plotly[5] for data handling and visualization.

6.3 Model Training

This process started with understanding the different machine learning techniques, both supervise and unsupervised. This proved useful when taking a closer look at the data.

The obvious thing, none of the entries had a big label warning that they were dangerous or unusual, and with the notion that the mass of data needed couldn't be labeled by hand, supervise learning was out of the question. Furthermore, the existing features will have to be processed in order to be used by the models. Strings needed to be turned into new features or parsed to dictionaries to be replaced by numbers, among other things.

6.3.1 Data Processing

With that in mind, the first step to any model or technique would be the preprocessing of data. This involves, depending on what logs/alerts are being use, choosing relevant features, such as origin IP's, creating new columns, like TOD (time of day), and filter out mostly empty columns. This process isn't a 'one fits all' type of solution, in fact, this is the most important step when creating a new model, as feeding it relevant data is what enables it to predict accurately the normal behaviour. Even with this taken into account, training models, involves revisiting this step multiple times to find the best and more relevant features to be used.

Additionally, there are two important steps, that, while not mandatory, showed to be the most appropriate to the project in hands during my study. Data normalization, where, after all the data is converted to numerical values, it is normalized to values between 0 and 1. This allows for a more uniform range across features, granting that all get treated with the same level of "importance" by the model. Applying this step showed to be a great improvement as features like package size, a number in bytes that might range from 0 to the millions, no longer got an advantage on features like success/error codes, for example.

6.3.2 Model Selection

Once the data preprocessing is complete, the next crucial step is selecting an appropriate machine learning model. This decision depends on the nature of the problem at hand. For instance, when dealing with binary classification tasks, logistic regression or decision trees might be suitable choices. Meanwhile, as it been mentioned before, this problem is way more complex. Not only its attributes bring a lot of information, the desired outcome should be a metric that brings more information than a yes/no classification.

The selection process, while it involves evaluating the performance of various models, mostly involved hyperparameter tuning. It was essential to strike a balance between model complexity and generalization to ensure the model can effectively capture patterns in the data without overfitting. This was a real problem as the selected model was Clustering. Way too much clusters can cause to dangerous behaviour being clustered separately and, consequently, not be flagged as 'abnormal'. On the other hand, a reduced number of centroids might cause normal behaviour, that is far enough way from them, to be flagged incorrectly.

6.3.3 Model Training

After choosing the machine learning model, the training process begins. This involves feeding the preprocessed data into the selected model and iteratively adjusting its internal parameters to produce better results. The model learns from the training data and aims to generalize its knowledge to make predictions on unseen data. This can either be past data from datasets or log files, as well as data "in real time", produced by running systems.

During training, various models were trained separatly, different cluster amounts were tested as well as different combinations of features. This brings useful insights on the data being analyzed that can help, not only, understanding the inner workings of the model, but also, understand what features might present dangerous behaviour.

6.3.4 Model Evaluation

Once the model is trained, it was crucial to evaluate its performance. This is typically done using various evaluation metrics like accuracy, precision, recall, etc. But, since the problem at hands brought a more complex evaluation to the table, measuring it success demonstrated to be more difficult than initially suspected.

While the output for every entry is a float (that represents the measured risk), this alone doesn't bring much information about the danger it represents. Only looking at the full alert, and some times more (i.e., like who the IP belongs to), was it possible to identify the "real risk".

Coming up with correct analysis of the labeled data, as it correspond to possible cybersecurity threats, requires a deeper understanding of the subject. At this point, my mentor's help was crucial in order to validate the accuracy of this step. Knowing this, I gathered the results from the trained models, prepared graphs that represented their behaviour and some more data about the clusters.

Together with my mentor, we analyzed the results for the various models. This meant taking into account the different systems where the data was gathered from, the original data for flagged alerts and their predicted risk. This step proved crucial on selecting the best model and, while it multiple times took me back to the drawing board, was the key to moving forward.

6.3.5 Model Deployment

After successfully training and evaluating the model, the final step is deploying it for real usage. This involved integrating the model into a system that I ended up developing, creating a user interface and setting up an API for making predictions in real-time.

Continuous monitoring and maintenance are, as can be guessed, essential to ensure the model's performance remains reliable over time. This is especially true when it comes to online learning models. In this case, models should be monitored to ensure the learning from real data doesn't impact the system in a negative way.

6.4 Interface

To complete the project, and make a more well rounded system, I decided to develop a web interface for the management of models, alerts and agents.

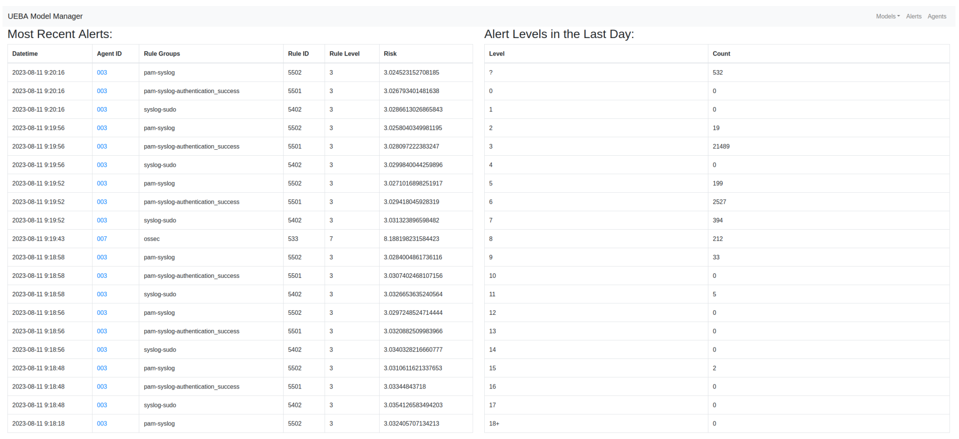
This section of the project involved designing the interface and the respective routes. This was done using base html with bootstrap, as mentioned before. The routes could be accessed by the localhost, if on the server, or, for security reasons, by tunneling the desired port and accessing it in the browser.

The main features of the interface include:

- Adding new models
- Managing existing models
- Check past/current alerts
- The ability to choose different models for different agents
- ...

Additionally, the interface contains a documentation tab, where information about the system can be found (this is also present in the GitHub repository). There, it is possible to found all the available routes, as well as more information on how to integrate new agents.

Bellow there are some example of the final system:



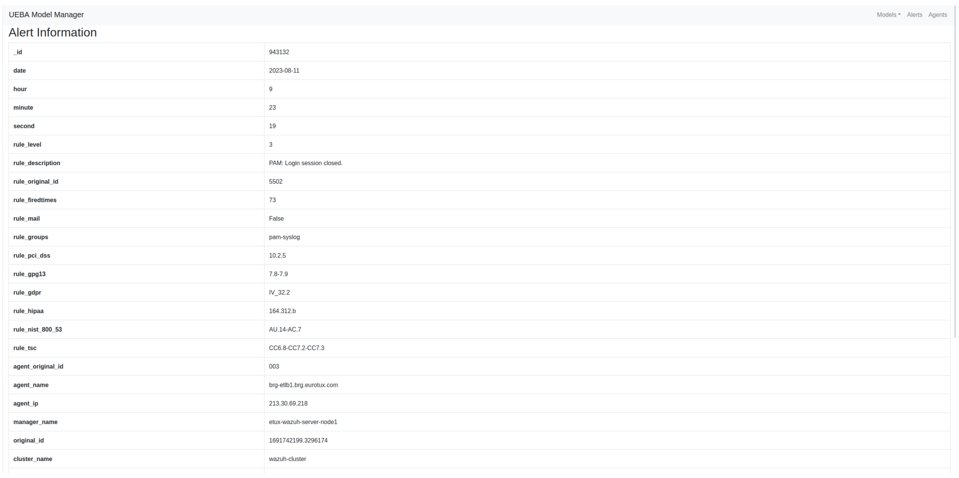
Most Recent Alerts:

Datetime	Agent ID	Rule Groups	Rule ID	Rule Level	Risk
2023-08-11 9:20:16	003	pam-syslog	5502	3	3.024523152708185
2023-08-11 9:20:16	003	pam-syslog-authentication_success	5501	3	3.026793401481638
2023-08-11 9:20:16	003	syslog-sudo	5402	3	3.0286613026865843
2023-08-11 9:19:56	003	pam-syslog	5502	3	3.0258040349981195
2023-08-11 9:19:56	003	pam-syslog-authentication_success	5501	3	3.028097222383247
2023-08-11 9:19:56	003	syslog-sudo	5402	3	3.0299840044259896
2023-08-11 9:19:52	003	pam-syslog	5502	3	3.0271016898251917
2023-08-11 9:19:52	003	pam-syslog-authentication_success	5501	3	3.029418045928319
2023-08-11 9:19:52	003	syslog-sudo	5402	3	3.031323896598482
2023-08-11 9:19:43	007	ossec	533	7	8.188198231584423
2023-08-11 9:18:58	003	pam-syslog	5502	3	3.0284004861736116
2023-08-11 9:18:58	003	pam-syslog-authentication_success	5501	3	3.0307402468107156
2023-08-11 9:18:58	003	syslog-sudo	5402	3	3.0326653635240564
2023-08-11 9:18:56	003	pam-syslog	5502	3	3.0297248524714444
2023-08-11 9:18:56	003	pam-syslog-authentication_success	5501	3	3.0320882509983966
2023-08-11 9:18:56	003	syslog-sudo	5402	3	3.0340328216660777
2023-08-11 9:18:48	003	pam-syslog	5502	3	3.0310611621337653
2023-08-11 9:18:48	003	pam-syslog-authentication_success	5501	3	3.03344843718
2023-08-11 9:18:48	003	syslog-sudo	5402	3	3.0354126583494203
2023-08-11 9:18:18	003	pam-syslog	5502	3	3.032405707134213

Alert Levels in the Last Day:

Level	Count
?	532
0	0
1	0
2	19
3	21489
4	0
5	199
6	2527
7	394
8	212
9	33
10	0
11	5
12	0
13	0
14	0
15	2
16	0
17	0
18+	0

Figure 6.1: First Version of Alerts Dashboard



Alert Information

_id	943132
date	2023-08-11
hour	9
minute	23
second	19
rule_level	3
rule_description	PAM: Login session closed.
rule_original_id	5502
rule_firedtimes	73
rule_mail	False
rule_groups	pam-syslog
rule_pci_dss	10.2.5
rule_gpg13	7.8-7.9
rule_gdpr	IV_32.2
rule_hipaa	164.312.b
rule_nist_800_53	AU.14-AC.7
rule_tsc	CC6.8-CC7.2-CC7.3
agent_original_id	003
agent_name	brg-etlb1.brg.eurotux.com
agent_ip	213.30.69.218
manager_name	etux-wazuh-server-node1
original_id	1691742199.3296174
cluster_name	wazuh-cluster

Figure 6.2: Alert Information Page

Chapter 7

Challenges and Difficulties

Throughout the course of this project, several challenges and difficulties were encountered, highlighting the complexity of developing a machine learning solution for anomaly detection in cybersecurity. These challenges can be categorized into the following areas:

Data Collection and Quality - One of the primary challenges was obtaining and working with the raw data. The project required a significant volume of logs and alerts from various sources, and acquiring, cleaning, and structuring this data proved to be a time-consuming and resource-intensive task.

Feature Engineering - Data preprocessing and feature engineering were particularly challenging due to the nature of the system logs. Converting text-based information into numerical features, handling missing data, and deciding which features to include or exclude required understanding the data and what it represented. This introduced a challenge that led me to a deeper study about the subject.

Resource Constraints - Resource constraints, including computing power, ram and time limitations, presented difficulties during the training as it slowed down development. Training complex models on large datasets required substantial computational resources, and optimizing model performance within these constraints was a constant challenge. Eventually resources were scaled for the needs but, since it was an internship project, they stilled proved to limit the rate of model training.

Deployment and Maintenance - Finally, deploying the trained model into a production environment and ensuring its ongoing maintenance presented challenges. Integrating the model into existing systems, handling real-time data streams, and monitoring the model's performance for drift or degradation required careful planning and periodical monitoring.

Despite all these challenges and difficulties, the project demonstrated the value of machine learning in enhancing cybersecurity efforts.

Chapter 8

Mentor's Feedback

Name: Elvys Marchon de Azevedo

Position: Cyber Security Engineer at Eurotux, SA

Role: Throughout the internship, Marchon acted as my supervisor and knowledge resource. He guided me in troubleshooting complex issues, sharing his expertise in Linux systems and security protocols. His insights were particularly valuable during decision-making processes, where his experience provided direction and clarity, helping me align with the company's goals.

I'd like to share the feedback regarding the performance of the intern, Ricardo Alves Oliveira, during the internship period. It's with satisfaction that I express my appreciation for the work he has accomplished.

Throughout the internship, Ricardo demonstrated excellent performance while working on the development of the UEBA (User Entity Behaviour Analysis) solution and its integration with our SIEM tool in our SOC (Security Operation Center). His ability to quickly grasp the intricate concepts associated with entity behavior analysis and apply them practically was remarkable.

Ricardo not only exhibited skill in acquiring knowledge but also proactively translated it into concrete actions. He successfully achieved the objectives set for the project, showcasing dedication and adeptness in attaining positive outcomes.

His effective collaboration with the team was notable, reflecting his capacity to communicate complex ideas clearly and concisely. His willingness to listen, pose pertinent questions, and contribute insightful perspectives enriched team discussions and contributed to the overall success of the project.

In summary, Ricardo Alves Oliveira was an exemplary intern who exceeded expectations in every aspect. His passion for learning and dedication to applying acquired knowledge resulted in a significant impact on our team and the UEBA project.

I am confident that Ricardo will continue to achieve success in his professional journey, carrying with him the skills and exemplary work ethic he demonstrated during his time with us.

Chapter 9

Conclusion

In conclusion, this project represents a significant step towards leveraging machine learning techniques for anomaly detection in the field of cybersecurity. Throughout the project, I encountered various challenges and difficulties. However, these challenges were met with determination and a commitment to overcoming them either alone or with the help of my mentor and the security team.

The key aspects of this project include:

- Successful preprocessing of cybersecurity (and general systems) logs and alerts, enabling the conversion of textual data into meaningful features.
- The identification and implementation of machine learning models that exhibited promising performance in detecting anomalies within the data.
- Integration of web development components, including the creation of user interfaces and interactive dashboards for visualizing and interacting with the model's outputs.
- Implementation of APIs (Application Programming Interfaces) to facilitate seamless communication between different software components.
- System administration tasks to ensure the reliability and scalability of the project.
- Collaborative efforts between cybersecurity experts and developers to bridge the gap between domain knowledge and machine learning.

While this project was mainly focused on Artificial Intelligence and Cybersecurity, it reflected the multidisciplinary nature of developing a complete system and the valuable skills and knowledge gained during its development.

Bibliography

[1] Bootstrap. *Bootstrap Documentation.* URL: `https://getbootstrap.com/docs/5.1/getting-started/introduction/`.

[2] Flask. *Flask Documentation.* URL: `https://flask.palletsprojects.com/en/2.3.x/`.

[3] NumPy. *The fundamental package for scientific computing with Python.* URL: `https://numpy.org/`.

[4] Pandas. *Pandas Documentation.* URL: `https://pandas.pydata.org/docs/`.

[5] Plotly. *Plotly Open Source Graphing Library for Python.* URL: `https://plotly.com/python/`.

[6] Python. *Python.org.* URL: `https://www.python.org/`.

[7] scikit-learn. *Machine Learning in Python.* URL: `https://scikit-learn.org/stable/index.html`.