TÉCNICO LISBOA

ACADEMIA MILITAR
MILITARY ACADEMY

Visualisation Tool for Automatic Threat Detection in Cyberspace

Pedro Miguel Bonito Marques

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Professor Miguel Nuno Dias Alves Pupo Correia

Major Luís Filipe Xavier Cavaco Mendonça Dias

Examination Committee:

Chairperson: Professor Teresa Maria Sá Ferreira Vazão Vasques

Supervisor: Professor Miguel Nuno Dias Alves Pupo Correia

Members of the Committee: Professor Filipe Manuel Simões Caldeira

Lieutenant Colonel Henrique Martins dos Santos Cunha

November 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgements

All this work would not have been possible without the complete dedication and support of Professor Miguel Correia and Major Tm Luís Dias. I always felt accompanied in the supervision and clarification of all doubts that have arisen during these long months of shared work.

It was also fundamental for the execution of this work, all the moral support, ideas, debates and help in technical details from my great friends: Carlos Valente, Pedro Bilou and Filipe Orvalho.

The support and love of my family and especially the advice and help of my mother Esperança Bonito were essential in this process.

I cannot forget the help and support of my comrades from the Military Academy, who have accompanied me throughout these years.

My humble acknowledgement is to all those mentioned in this simple text and those who may not be explicitly referred.

Abstract

This thesis presents and describes a visualisation tool that provides intrusion detection algorithms with an effective graphical interface.

The goals of the tool are twofold. First, it aims to aid the human analyst on observing ongoing cyber threats, as a step towards responding to these attacks. Second, it aims to help the analyst study the performance of the algorithms with different parameters, in order to decide which combination best suits his goals. In fact, the tool integrates a set of recently-proposed intrusion detection algorithms based on unsupervised machine learning, which it aims to help configuring.

The tool follows a client-server system architecture and implements a REpresentational State Transfer (REST) Application Programming Interface (API) that mediates the communication between the server and the Graphical User Interface (GUI) on the client-side. An important challenge was to define data structures that allow storing and accessing data efficiently. As far as the GUI is concerned, a clear and minimalist visual aspect was chosen, containing functionalities that allow the user to analyze the network data in a simple and clear way.

For the evaluation of the whole system, we used the theoretical requirements for the evaluation of systems and software of ISO/IEC25010:2011. For validation, we compared the results obtained by our tool with the results described in the original papers that present the algorithms.

Key-words: Cybersecurity, Machine Learning, Intrusion Detection System, Security Analytics, Visualisation Tool

Resumo

Esta tese apresenta e descreve uma ferramenta de visualização que fornece algoritmos de detecção de intrusão com um interface gráfico eficaz.

Os objectivos da ferramenta são duplos. Primeiro, visa ajudar o analista humano a observar as ciber ameaças em curso, como um passo para responder a estes ataques. Segundo, visa ajudar o analista a estudar o desempenho dos algoritmos com parâmetros diferentes, a fim de decidir qual a combinação que melhor se adequa aos seus objectivos. Para tal, a ferramenta integra um conjunto de algoritmos de detecção de intrusão recentemente propostos com base na aprendizagem não supervisionada da máquina, que visa ajudar a configurar.

A ferramenta segue uma arquitectura de cliente-servidor e implementa uma REST API que medeia a comunicação entre o servidor e a GUI no lado do cliente. Um importante desafio era definir estruturas de dados que permitissem armazenar e aceder aos dados de forma eficiente. No que diz respeito ao GUI, foi escolhido um aspecto visual claro e minimalista, contendo funcionalidades que permitem ao utilizador analisar os dados da rede de uma forma simples e clara.

Para a avaliação de todo o sistema, utilizámos os requisitos teóricos para a avaliação de sistemas e software da ISO/IEC25010:2011. Para validação, comparámos os resultados obtidos pela nossa ferramenta com os resultados descritos nos artigos originais que apresentam os algoritmos.

Palavras-chave: Cibersegurança, Aprendizagem Automática, Sistema de Deteção de Intrusões, Análise de Segurança, Ferramenta de Visualização

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 3

AM Academia Militar. 1

API Application Programming Interface. v, vii, 2, 10, 12, 13, 19, 21, 23–25, 36, 42, 46, 47, 55

AUROC Area Under the Receiver Operating Characteristics. 11

BLOB Binary Large Object. 37, 39

C2BID Cluster Change-Based Intrusion Detection. 16–18, 23, 55

CAMIAC Cyber Attack Modelling and Impact Assessment Framework. 11, 12

CLINK Complete LINKage. 5

CMIA Cyber Mission Impact Assessment. 10, 12

CSV Comma-separated values. 13, 17, 20, 22, 24, 29, 32, 37, 47

CVSS Common Vulnerability Scoring System. 9

DB Database. 12, 13, 19, 21, 24, 25, 27, 36–41, 46, 47, 55

DB4S DB Browser for SQLite. 24

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 6, 9, 16, 18

DDoS Distributed Denial of Service. 1, 48

DoS Denial of Service. 48

DPI Dots Per Inch. 35

FK Foreign Key. 36, 37

FN False Negative. 11, 20, 30, 48, 50–53

FP False Positive. 7, 11, 20, 30, 48, 50–54

FPR False Positive Rate. 11

GUI Graphical User Interface. v, vii, 2, 11–14, 19–21, 24–26, 32, 34, 36, 44–46, 55

HIDS Host-based Intrusion Detection System. 7

HTTP Hypertext Transfer Protocol. 7, 12, 14, 19, 21, 24, 25, 44, 46, 55

HTTPS Hyper Text Transfer Protocol Secure. 19, 46, 55

ICT Information and Communications Technology. 1

IDS Intrusion Detection System. 1–4, 6–9, 20, 22, 23, 25

IP Internet Protocol. 15, 16, 19, 20, 22, 28, 29, 33, 34, 38–44, 47, 48

IPS Intrusion Prevention System. 6, 7

IRC Internet Relay Chat. 14

IST Instituto Superior Técnico. 1

IT Information Technology. 1, 8, 10

JSON JavaScript Object Notation. I–VIII, 13, 42

MCC Matthews Correlation Coefficient. 11, 49–53

ML Machine Learning. 1–4, 9, 14, 17, 44, 55

MVP Minimum Viable Product. 19

NIDS Network-based Intrusion Detection System. 7

NoSQL Not only SQL. 38, 47

ORM Object-Relational Mapping. 24

OS Operating System. 26, 46

PIDS Protocol Intrusion Detection System. 7

PK Primary Key. 36–39

RELOAD Rapid EvaLuation Of Anomaly Detection. 11–13

REST REpresentational State Transfer. v, vii, 2, 10, 12, 19, 21, 23, 46, 55

RRCF Robust Random Cut Forest. 16–18

SLINK Single LINKage. 5

SMTP Simple Mail Transfer Protocol. 14

SOAP Simple Object Access Protocol. 11, 12

SQL Structured Query Language. 24, 38, 39, 47

SQuaRE Systems and software Quality Requirements and Evaluation. 45, 54

SSH Secure Shell. 14

TN True Negative. 11, 20, 30, 48–50, 52

TP True Positive. 11, 20, 30, 48–50, 52

UPGMA Unweighted Pair-Group Method using arithmetic Averages. 6

URL Uniform Resource Locator. 42

VAS Visual Analytics Systems. 2, 3, 8–10

VS Code Visual Studio Code. 24

WPGMA Weighted Pair-Group Method using arithmetic Averages. 6

XML Extensible Markup Language. 11, 12

1 Introduction

The significant amount of digital information generated in the world continues to grow, even more so at this time when remote working and education has become part of the reality of many people [1]. This growth in the Information Technology (IT) field combined with an environment of constantly changing risks and threats make cyber security essential to us all [2].

With a growing demand for information in the digital world, it is necessary to identify threats to protect ourselves more effectively. According to a study released by Eurostat on Information and Communications Technology (ICT) [3], the most recurrent software or hardware failures problems are related to Distributed Denial of Service (DDoS), which involve overloading a network with many requests to block it [4], and Ransomware attacks. The last one is a type of malware that consists of locking a system or its data, where the attacker demands payment in exchange for the restoration of the system [5].

More recently, with the evolution of the Machine Learning (ML) field, the number of Intrusion Detection System (IDS) using anomaly-based detection methods has grown [6] [7] [8]. Those who use unsupervised ML algorithms do not need to have the signatures of the attacks to detect them, thus bringing advantage in the ability to detect new attacks [9] [8]. IDSs can be enhanced with visualisation tools for faster and clearer verification of the results obtained by the algorithms they use. visualisation tools are a key component of the system, allowing the analyst to visualize the useful information extracted from the huge volume of security data.

1.1 Challenges and Motivation

One of the motivations for the realization of this work is to obtain a graphical tool that facilitates the activity of those responsible for the security of computer networks. Another motivation is to continue the previously carried out research in this area by students of Instituto Superior Técnico (IST) belonging to AM, integrating the IDSs algorithms developed in this area work in a single tool easy to use. It is also a motivation to obtain good results in threat detection (victims and attackers) with these IDSs algorithms that have a current approach by implementing unsupervised ML techniques. The possibility of widening the range of knowledge by contacting with frameworks, libraries, and data structures different from those already used during the course was another reason for challenge and motivation.

The main challenge of this work is to develop a visualisation tool that integrates and facilitates the use of the intrusion detection algorithms already created. Specifically, we wanted the analyst to have a comfortable user experience when accessing and requesting the processing of data in an interactive manner. Another challenge was to define data structures that allow storing and accessing data efficiently.

1.2 Objectives and Key Results

To deal with cyber security problems, it is necessary to have IDSs paired with analytical and visualisation tools that support the response to these challenges [10]. The main objective of visualisation tools is to intuitively and effectively show the data processed by the algorithms used on IDSs. The results obtained are intended to reduce the time to make the right decisions regarding threat handling [11].

The goals of the tool are twofold. First, it aims to aid the human analyst in observing ongoing threats as a step towards responding to these attacks. It aims to be a graphical visualisation tool for the automatic detection of threats in cyberspace that is intuitive and easy to use. Second, it aims to help the analyst study the performance of the detection algorithms with different parameters, in order to decide which combination best suits his goals. To this end, it provides the user with an abstraction of the event analysis processes, allowing the optimization of the utilization of the algorithms applied for intrusion detection. The tool integrates a set of recently-proposed intrusion detection algorithms based on unsupervised ML, which it aims to help configure for achieving high performance [12–14]. The initial version includes 3 algorithms, but it is extensible to as many as needed.

To achieve the main goal is necessary to develop a solid and flexible REpresentational State Transfer (REST) Application Programming Interface (API) that allows easy and fast access to the processed data. The API will be used by our visualisation tool. However, it should also be capable of being accessed by developers/researchers who want to use the data on our server. It is necessary to have well-structured requests to the API and return the data in formats that allow easy use to make it possible.

We consider that the main objective of this work was achieved, although, in this area, it is always possible to improve and optimise the software performance. We created a tool that allows the user to analyse network flows intuitively and abstractly, and the results are presented with appealing and straightforward graphs. All the tool code (GUI and back-end) is available on *a3ceProject*'s public repository on GitHub[1].

1.3 Report Structure

This document is divided into six chapters, including this introduction. Chapter 2 describes the main theoretical concepts that are the basis of the tool, such as ML, clustering algorithms, IDSs, and Visual Analytics Systems (VAS)s. Chapter 3 presents the related work regarding visualisation tools that integrate IDSs and the previously developed intrusion detection algorithms integrated into our tool. Chapter 4 presents the solution that was developed to achieve the objectives, describes the approach used and how the solution was implemented and presents the server-side and the Graphical User Interface (GUI). Chapter 5 describes the evaluation of the whole system according to the requirements and evaluation concepts and compares the results obtained by the algorithms. Chapter 6 concludes and summarizes the work done and presented in this document and suggests future works that could follow from this one.

[1]Repository url: `https://github.com/a3ceProject/CyberVTI`

2 Background

This chapter covers the basic concepts on which this work is based from a theoretical and general point of view. It presents a literature survey on ML methods, describes the main clustering algorithms, IDSs and VASs related to this work.

2.1 Machine Learning

In the last few decades, there has been a rapid growth of data on the internet. Therefore, it is essential to use data analysis systems for processing those data. ML is one of the solutions to these big data and security problems, making it easier to extract information from these extensive data sets [6].

ML is an essential branch of Artificial Intelligence (AI). This field of computer science has been growing over the last years, making its application indispensable nowadays in several areas of society. The main goal of ML is to study, engineer, and improve mathematical models that best adapt to a given set of data to make decisions without complete knowledge of all influencing elements. Based on probability distribution and considering the smallest associated error, the output that best correlates with input data is generated. ML techniques have been growing in many areas due to their ability to handle large volumes of data [6] [15].

There are several learning methods, all of them with the main goal of solving or optimizing problems. The following two groups of ML methods are popular in intrusion detection:

- **Supervised machine learning:** These methods are used to find a function or mathematical model that fits the data. In order to arrive at the function, labelled training datasets are provided, which relate the various pairs of inputs and outputs. Once the function is obtained, the learning model of the already trained machine can classify data points that were not in the training dataset [16].

- **Unsupervised machine learning:** In this method, the difference is that they do not require previously labelled training data, which humans usually label. In this way, the human influence on ML is reduced. The main objective is to find patterns, structures or knowledge from these unlabeled data. The clustering approach is one of the most common ML unsupervised methods [9].

In supervised learning, when an effective model is obtained, it becomes very accurate in classifying the input data. However, for this model to be developed, it is necessary to provide a lot of labelled data, which leads to a high processing and integration cost. This cost can increase further with the complexity of the data involved. On the other hand, unsupervised learning does not require previously known data, so these methods have a great potential to be applied in IDSs. Since cyber threats are constantly evolving and unsupervised learning does not need to know the signatures and behaviour of attacks, this technique becomes very useful to detect anomalous behaviour (possible intrusions) [17].

2.2 Clustering Algorithms

Clustering is one of the techniques used to implement unsupervised ML. The objective of this class of algorithms is to find patterns in multidimensional unlabeled data. These patterns are found after the data are grouped based on a measure of similarity [9].

This subsection presents the main steps and some types of clustering algorithms used in the IDSs algorithms to be implemented in this work.

2.2.1 Overview

The clustering process, which analyses the input data and leads to its classification, can be divided into the following steps [18]:

- **Feature selection and extraction:** Chooses different features from a set of candidates that are important to group. The ideal features should be used to distinguish patterns belonging to different clusters, immune to noise, easy to extract and interpret. A good selection of features can improve the results and decrease the complexity of the algorithm. After selection, the features extraction uses some transformations to generate valuable and new features (also called normalised features) from the original ones.

- **Clustering algorithm selection:** According to the problem, the data and the selected features, one must choose, within a wide range of clustering algorithms, the one that will bring the best results.

  By using a neighbourhood measure and a criterion function, the data points are grouped according to the similarity of their features. Once a neighbourhood measure is defined, the construction of a grouping criteria function makes the clustering a problem of optimisation, which can be defined mathematically. However, there is no criterion function or neighbourhood measure that is generic. Each case must be interpreted according to a division problem.

- **Clusters validation:** At this step, standards and evaluation criteria are applied to provide the users with a degree of confidence for the clustering results derived from the used algorithms because different approaches can generate different clusters.

  Usually, there are categories of testing criteria: external, internal, and relative indices, which can be defined in three types of cluster structures, known as partitions, hierarchical, and individual. The external indices are based on a pre-specified structure, based on prior data information, and used as a standard for validating clustering solutions. Internal testing does not depend on prior knowledge. They examine the aggregation structure directly from the original data. Finally, the relative criteria give weight to the comparison of different clustering structures to decide which can best reveal the characteristics of the objects.

- **Results interpretation:** This step is the last step that generates knowledge and relevant information through the initial data. This step or even the whole process may have to be carried out several times to ensure the knowledge's reliability.

2.2.2 Main Clustering Algorithms

Clustering algorithms can be categorised by the methods they use, which can be: partition, hierarchical, density, and neural. In this section, an algorithm for the first three types of methods is presented.

- **K-Means:** According to Wagstaf et al. [19], K-Means are used to partition a dataset into k groups automatically. It proceeds by selecting k initial cluster centres and then iteratively improving them in the two phases:

  1. Each instance d_i is assigned to its closest cluster center.
  2. Each cluster center C_j is updated to be the mean of its constituent instances.

  Advantages of K-Means clustering algorithm:

  - In the splitting approach, if K is supplied accurately, then splitting the clusters is easy [6];
  - In the case of network anomalies, where there is a large data set, it is possible to group them into a similar number of classes, with a low cost of computational complexity [6];
  - Still regarding intrusion detection, this algorithm can learn from the audit data without requiring the system administrator to provide explicit descriptions of various classes of attacks [9];

  This algorithm only has access to the set of features describing each object. It converges when there is no further change in the assignment of instances to clusters. K-Means has linear complexity, $O(nkdi)$, where: n - number of points to consider; k - number of clusters generated; d - data-point dimension; i - number of iterations until convergence.

- **Hierarchical Agglomerative:** The initial approach starts by considering each object with a single cluster. Then, it merges the clusters after computing the cost function. This type of algorithms can be implemented using various methods [20]:

  - Single LINKage (SLINK) or Nearest Neighbor Method: The Euclidean norm is calculated between the points, thus obtaining coefficients $C_{(i,j)}$. This is a minimised cost function as shown in equation 1 :

    $$C_{SLINK} = Min(C_{(1,1)}, ..., C_{(i,j)}, ..., C_{(m,n)}) \tag{1}$$

  - Complete LINKage (CLINK) or the Furthest Neighbor Method: In this case, the cost function, composed of all distances $C_{(i,j)}$ is maximised.

    $$C_{CLINK} = Max(C_{(1,1)}, ..., C_{(i,j)}, ..., C_{(m,n)}) \tag{2}$$

– Unweighted Pair-Group Method using arithmetic Averages (UPGMA): The similarity measure between two clusters is the arithmetic average of coefficients among all pair entities $C_{(i,j)}$ in the two clusters.

$$C_{UPGMA} = \frac{1}{mn} \sum_{i=1,j=1}^{m,n} C_{(i,j)} \tag{3}$$

– Weighted Pair-Group Method using arithmetic Averages (WPGMA): This is the simple arithmetic average of resemblance coefficients between two clusters without considering the cluster size.

$$C_{WPGMA} = \frac{1}{mn} \sum_{i=1,j=1}^{m,n} W_i C_{(i,j)} \tag{4}$$

This process is done hierarchically until there is a root (complete dataset) or a stop condition. Calculating all pairs of distances makes the complexity quadratic, $O(n^2)$, where n is the number of data points.

- **Density-Based Spatial Clustering of Applications with Noise (DBSCAN):** In this algorithm [21] it is necessary to define a point p, a threshold distance between points (ϵ) and the minimum number of points needed for a group to be considered as a member of a cluster ($MinPts$).

  It is possible to recognise clusters using density because each cluster is a region with a higher density than outside the cluster. Even if points appear outside the cluster due to noise, they are scattered everywhere, which is low density.

  To start, a random point p is chosen, and all density-reachable (depends on ϵ) points from p are determined, taking $MinPts$ into account. If a point is at the border (cluster boundary), it is ignored. On the other hand, if the point is the centre of the cluster region (core), then a cluster based on that point is generated. If a point is too far away from a cluster ($dist_{p,q} > \epsilon$) then it is considered as noise (outlier) and is discarded.

  The average run time complexity of DBSCAN is O($n \times log(n)$), where n is the number of points on the dataset [21].

2.3 Intrusion Detection Systems

An intrusion detection systems can be software or hardware that uses automated monitoring processes for events that occur on a computer system or network. The process of analysing intrusion signals is based on an inspection that tries to identify attempts to compromise the confidentiality, integrity, or availability of the systems [22].

An IDS should not be confused with an Intrusion Prevention System (IPS), although IPSs use IDSs to identify the threat. IPSs analyse the risk and respond to the possible threat in the best possible way, they can stop the threat. Although both systems are very useful for network security, there are differences. IDSs do not interfere with network traffic, they monitor the network and alert the network manager to potential

threats, it is up to him to evaluate the risk and decide how to react to the threat. This way, IDSs are considered passive security measures. Unlike IPSs, which after detecting the threat, have the autonomy to decide how to act, thus they are considered an active security measure [23]. However suppose the IPSs chooses the wrong decision, for example, block network traffic, and it is a False Positive (FP) In that case, this breaks trust in the system and can be very damaging to an organisation (e.g., service downtime). For this reason, IPSs should only block traffic when the used algorithms produce results with full confidence (usually signature-based). Regarding anomaly detection approaches and the use of unsupervised learning, i.e., detection of unknown attacks, FP can occur and are usually only used for detection in their passive way.

2.3.1 IDSs Types

Taking into account the action area of IDSs, where they acquire and monitor data traffic, these can be divided into: 1) Host-based Intrusion Detection System (HIDS) - located in the host, this monitors malicious activities through the analysis of system parameters (e.g., analysis of running applications); 2) Network-based Intrusion Detection System (NIDS) - performs an analysis of the network in which it is inserted, by monitoring the traffic and the behaviour of hosts on the network, it can detect anomalies; 3) Hybrid IDSs - the sources of events of this type of IDS are the hosts and the network, combining NIDS with HIDS [24] [25]. There are, however, within these 3 divisions, IDSs that can be more specific regarding the data source, for example, the Protocol Intrusion Detection System (PIDS) that analyse only the communication protocols of the network (e.g., Hypertext Transfer Protocol (HTTP)) [26].

Other specifications that IDSs can be divided into are about their ability to detect threats in real-time, or delayed if it only detects intrusions later when it is decided to analyse the data [13].

2.3.2 Detection Methods

IDSs can also be divided according to the method they implement to detect threats. There are three main types of detection methods in security analytics: Misuse-based, Anomaly-based, and Hybrid [9], which are detailed below.

- **Misuse-based**

  These methods are designed to detect known attacks by using signatures of those attacks. They effectively detect known types of attacks without generating an overwhelming number of false alarms. However, they can not detect new attacks (zero-day attacks) that do not have any known signatures. Thus, requiring frequent updates with new rules and signatures, usually manually [9].

- **Anomaly-based**

  Anomaly detection tries to find patterns in data, which do not conform to the expected normal behaviour. With these techniques, normal activity profiles are created for each system, application, or network, making it difficult for attackers to know which activities they can perform without being

detected. These are capable of detecting new attacks. Thus, they can be used to define the signatures of the misuse detectors. The main disadvantage of anomaly-based techniques is the potential for high rates of false alarms because previously unseen (but legitimate) system behaviour can be categorised as anomalies [6].

- **Hybrid**

  To improve IDS techniques, researchers have proposed hybrid detection techniques to combine anomaly and misuse detection techniques in IDSs, in order to be able to detect known and unknown attacks [6].

2.4 Visual Analytics Systems

Vision is the main human sense for the acquisition of knowledge, so visualisation tools are essential in several areas [27]. According to Jacob et al. [27] these tools have the following advantages:

- Communicate complex data quickly - through the use of graphs and statistical metrics, it is possible to transmit a large amount of data to the user effectively.
- Enable recognition of patterns - showing the data or the relationships between them, making it possible for the user to identify patterns visually.
- Enable quality control on the data - data errors or results can be quickly identified through visualisation.
- Provide a correct perspective - the visualisation may lead the user to acquire a different perspective of the data, providing a better understanding and approach.

VASs applied in cyber security can have several functionalities. One of them is intrusion detection, which by applying the IDSs mentioned above, obtains useful information that is later presented in VASs. However, these systems can provide other important information to the network analyst by applying other techniques, such as:

- **Vulnerability analysis**

  The main objective is to identify cyber vulnerabilities. Analyses are carried out to obtain knowledge about the Information Technology (IT) devices that should be protected and how to do it [28]. This can be done in the following ways:

  - Manually: Security threats are identified, ordered, mitigated, and validated by the responsible network administrator, using tools such as the Microsoft Threat Modeling Tool [29].
  - Semi-automatically: To automatically identify mission processes and goals from work performed by the IT devices.
  - Automatically: Monitor traffic and perform asset identification using network topology inference, using tools such as NMAP [30].

VASs relating to detecting vulnerabilities can be further divided into two categories: product-based vulnerabilities and organization-based vulnerabilities. The first one tests vulnerabilities related to bad implementation or malfunction of the product. The discovered vulnerabilities are communicated to consumers and placed in public databases. The second aims at identifying security vulnerabilities that arise due to the operational deployment of these products (e.g., bad configuration of products or deployment without providing secure execution).

- **Impact Assessment**

  These methods aim to evaluate and/or estimate the amount of damage caused to the organization's assets and what organization objectives are compromised by a cyberattack [31]. They can show what countermeasures can mitigate these impacts and how these should be applied [32]. According to Kotenko et al. [31], five main groups of security and impact assessment metrics can be present in the VASs, these are:

  1. Metrics related to the topology, criticality and vulnerabilities of the analysed system (hosts).
  2. Metrics that characterize the attack (e.g., probability of attack). These metrics are based on the Common Vulnerability Scoring System (CVSS) and the metrics of the first group, and they allow to calculate the severity and how the system assets will be compromised.
  3. Metrics that characterize the attacker's potential and are destined to define the possibilities of development of the attack. The position of the attackers in the system and their skills are taken into account.
  4. Metrics on the efficiency of the response (ability to reduce the impact of attacks) and the collateral damage that comes from countermeasures.
  5. Characteristics of the system security and its risk level, based on previously obtained metrics.

2.5 Summary

In this chapter, the fundamental theoretical concepts that are the basis of this work were presented. Supervised and unsupervised ML techniques were described, with more emphasis given to unsupervised ML. Some clustering algorithms are presented, such as K-means, Hierarchical Agglomerative, and DBSCAN, as well as how they work. IDSs were presented, their various types regarding the area of action and the detection methods they can use to identify threats. The VASs that are applied in the area of cyber security and their purposes were also described.

3 Related Work

This chapter is divided into two sections. The first presents some existing graphical tools for visualising network intrusions and the main modules of these tools. The second presents the previously developed intrusion detection algorithms that will be embedded in our tool.

3.1 Visual Analytics Systems Frameworks

Visual analysis techniques can be successfully applied to handle large amounts of data, as they can process this data and extract new knowledge better than humans. Using VASs, it is possible to combine the strengths of the human visual system with computational machine power, making it possible for the user to obtain information from a large set of data in an efficient way [33].

3.1.1 Tools

There are several types of tools and models in the cyber security area, which implement, analyse, and show the results in their own way (e.g., icons, graphs, word maps, heatmaps, plots). Some apply specific techniques (of those presented in Section 2.4). Others are more complete and complex, composed of several layers (IT infrastructure, vulnerability, mission, attack detection and impact). Each one is best suited to certain types of data to show.

Below are described the visualisation tools that have relevance and similarities with the work that we developed:

- **Cyber Mission Impact Assessment (CMIA)** [34]: the main objective of this tool is to evaluate the impact on the system/network mission that possible cyber incidents would have. It focuses on the impact that the incident will cause and not on the system's vulnerabilities or the probability of the incident occurring. The effects of cyber incidents on the mission are classified as degradation, disruption, modification, manufacturing, unauthorised use, and interception. Regarding the visualisation part, this tool graphically presents the results of the incident effects on the system mission and a network infrastructure diagram. In sum, it provides data to the network security officer to understand the implications that possible cyber incidents would have on the network mission.

- **CyGraph: Graph-Based Analytics and visualisation for Cybersecurity** [35]: is a system for analyzing and reasoning about network attack relationships. It correlates data from various sources (e.g., topology, vulnerabilities, client/server configurations, firewall rules, events) into a common, normalised model and builds a persistent graphical data store representing network attack relationships and associated network data, providing an interactive visualisation of complex dependency relationships. CyGraph is based on a client-server architecture implemented through a REST API.

- **Cyber Attack Modelling and Impact Assessment Framework (CAMIAC)** [31]: it is a framework for cyber attack modelling and impact assessment. That is based on representing potential malefactors behaviour, generating attack graphs, calculating security metrics, and providing risk analysis procedures. It applies a set of algorithms with different timelines and precision to get near-real-time event analysis and security and impact assessment predictions. The attack graph is modified according to the changes in the analysed network. An application server is responsible for the communication between the various modules of this tool (e.g., GUI network sensors, report generator and attack graphs). This communication is done through Simple Object Access Protocol (SOAP) using messages in Extensible Markup Language (XML) format.

- **VisSecAnalyzer** [36]: is a visual analysis tool for network security assessment that aims to provide visual support to the cybersecurity officer. Its goal is to reveal the most vulnerable points of the information system, forming attack patterns. After assessing the network security and analyzing the severity of the detected vulnerabilities, this tool can suggest possible countermeasures (e.g., software updates). The GUI in VizSecAnalyzer consists of interactive graphs (e.g., treemaps, pie charts) and colourful security reports, allowing to explore a large-scale network in a simple way.

- **Rapid EvaLuation Of Anomaly Detection (RELOAD)** [37]: is a tool that implements network anomaly detection algorithms whose goal is to evaluate the performance of the algorithms in threat/anomaly detection. It allows the user to integrate new algorithms and make their evaluation. However, it provides previously integrated algorithms. The standard RELOAD algorithms are divided into 6 main sets: clustering, statistical, classification, neighbour-based, density-based, and angle-based. This tool takes data from an existing dataset and identifies the most important features automatically. After this stage, anomaly detection algorithms are run, and the classification results are obtained (whether a given data is anomalous or normal). In the final phase, metrics are generated relative to the performance of the algorithms, such as: True Positive (TP), True Negative (TN), False Negative (FN), False Positive (FP), Precision, Recall, False Positive Rate (FPR), F1 Score, Matthews Correlation Coefficient (MCC) and Area Under the Receiver Operating Characteristics (AUROC)[2]. The RELOAD is executed locally and has a simple GUI divided into three main areas: i) Setup - where is choose the configuration parameters for the execution of the algorithm; ii) Paths - where is select the paths of the files to be analysed and the files obtained; iii) Data Analysis: it allows the user to choose the algorithms to be used for the analysis or incorporate a new algorithm. The tool also provides the user with summary graphs of the results obtained.

These existing visualisation tools were the basis of inspiration and knowledge acquisition for the development of our solution. Their goals are different and can be considered as: evaluate the impact of threats

[2]is a single scalar value that measures the overall performance of a binary classifier. Its value is in the range [0.5-1.0], where 0.5 represents the performance of a random classifier and 1 represents a perfect classifier, from: `https://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-9863-7_209`

(CMIA); analyse and relate attacks to the network (CyGraph); analyse risks and provide network security metrics (CAMIAC); present network vulnerabilities and create attack patterns, or evaluate the performance of anomaly detection algorithms. All these tools have a common goal, which is to transmit information in a fast and simple way to the user, although their GUI is different and the results are shown in different ways. As far as the implementation is concerned, these tools also have different architectures. Entirely locally, with the entire process being performed on the user side (e.g., RELOAD) or a client-server architecture based on communication protocols (e.g., CyGraph uses REST API - HTTP).

Table 1 summarises the main aspects of the graphical tools presented in this section.

Table 1: Summary table of the visualisation tool features.

Tool	Objective	Visualized data	Input data type	Real-time	Implementation
CMIA [34]	Assess the impact of intrusion on the system mission	Impact metrics, attack graph, network structure	Automated (from network)	Yes	Local software
CyGraph [35]	Detect, analyse and relate network attacks	Security metrics, attack graph, network structure	Automated (from network)	Yes	Client-Server (REST API - HTTP)
CAMIAC [31]	Analyse the risks and provide system security metrics	Security metrics, attack graph, network structure	Automated (from network)	Yes	Client-Server (SOAP API - XML)
VisSec--Analyzer [36]	Detection of cyber threats, security assessment	Security metrics, attack graph, network structure	Automated (from network)	Yes	Local software
RELOAD [37]	Detection of cyber threats, test detection algorithms	Algorithms score metrics and performance graphs	Manual (from local files or DBs)	No	Local software

3.1.2 Data Storage

The first stage is *Data Onboarding*, the process of obtaining the data that will be further processed in the remaining modules [38]. This can be done in several ways, for example: i) using scripts that extract the data automatically, which is the most effective for large volumes of data; ii) giving data manually by the user, which can be used for example to define the structure of the network or identify hosts. However, this process is inefficient to obtain more extensive and more complex data; iii) data obtained from public datasets for testing purposes.

Before being stored, the data goes through a *Data Validation* process, where it is checked for the requirements for processing and that the data contains no anomalies (e.g., corrupted data). If necessary, the data may have to be treated to be validated [39].

The last phase is *Data Storage*, this is where the data is stored in a previously defined structure. The data has several formats (e.g., JSON) and forms to be stored (e.g., Relational databases), and it is up to the developer to choose which best suits the use of the tool. For example, RELOAD, being a local execution tool makes use of data input from text files or MySQL Database (DB)s and outputs the data in Comma-separated values (CSV)[3] files, in a user specified directory [37].

3.1.3 Processing Module

It is in this module that the data is processed to generate useful information for the user. This module contains the algorithms responsible for intrusion detection, vulnerability analysis, and performance evaluation, among other tools functionalities (e.g., statistical analysis). In turn, these algorithms are made up of functions, services, routines, and data structures that allow for data processing. The processing module can be implemented: i) locally - on the machine where the tool is running, or ii) on a server - the user only receives the results of the processing that occurs on the server, in which case, it is necessary to have an interface responsible for the communication, also called API [35] [37]. Considering an optimized processing module, it should be noted that an implementation based on a client-server architecture will be easier to handle with a large volume of data since the processing capacity of the servers should be larger than that of the user's machine.

3.1.4 Graphical User Interface

The GUI is the part of the software that supports the interaction between the user and the set of services and functionalities provided by the tool. Its main purpose is to make user interaction practical at all stages of the process (configuration, operation, and obtaining results), using graphical visual representations such as text boxes, buttons, menus, and icons [40].

Marcus et. al. [40] refers that the GUIs design can increase the capacity to communicate information, and that the following requirements should be considered:

- A comprehensible mental image.
- Appropriate organization of data, functions, tasks, and roles.
- Efficient navigation schema among these data and functions.
- Quality appearance characteristics and effective interaction sequencing.

The manipulation of visual language is the core of the GUI, so Marcus et. al. [40] present the 3 principles of graphical interface design:

- **Organize**: The user must have a clear and consistent structure. In the screen layout, a grid structure should be used, and the elements should be organized into groups in which the functionalities are related.

[3]It is a delimited text file format that uses a comma to separate values, each line of the file corresponding to a data record.

The main aspects should be distinguished from the secondary ones to focus the user's attention on the most important, thus allowing more effective navigation.

- **Economize**: This principle mentions that the GUI should be simple, should only have the minimum essentials, and should not be overloaded with unnecessary or repetitive aspects. Its components should be clear about their functionality and have a prominent hierarchy according to their relevance and frequency of use.

- **Communicate**: To communicate successfully, the GUI must find a balance between the following factors: i) *legibility* - individual design characters, symbols, and graphic elements must be easily noticeable and distinguishable; ii) *readability* - the display must be easy to interpret, while at the same time appealing and attractive; iii) *typography* - must use tables, lists, graphs and diagrams to present data, use interactive menus and dialogue boxes, all these components must be carefully adjusted in the GUI; iv) *symbolism* - icons, schemes and images can be used to complement the interface; v) *multiple views* - the interface must provide multiple views to show more complex and detailed data, these views can be overlaid or viewed simultaneously; and vi) *colour/texture* - the use of colour and texture should follow the principles of interface design (organize, economize, and communicate), not too much brightness or strong contrast should be used, and the background should have a dark colour.

3.2 Previous Intrusion Detection Algorithms

Since the present work concerns the development of a visualisation tool that integrates intrusion detection algorithms, it is necessary to explain their operation and approach. Below are presented the three main detection algorithms that precede the development of our tool.

3.2.1 FlowHacker

FlowHacker [12] is a network intrusion detection algorithm that uses anomaly-based techniques. It implements unsupervised ML, specifically clustering (K-Means), to group machines with similar network behaviours, so it does not need prior knowledge about the attacks (e.g., signatures) to detect them. Potential threats are grouped in different clusters from the remaining entities, so hosts or users with anomalous behaviour are in isolated clusters and can be identified.

FlowHacker [12] uses twenty-six features, divided into two groups. Half of the features are related to the source computer, and the other half are the same but relative to the destination computer. Eight of the thirteen features consider the bidirectional flow (to and from ports) of four application-layer protocols, which are: HTTP (port 80), Internet Relay Chat (IRC) (port 194 and 6667), Simple Mail Transfer Protocol (SMTP) (port 25), and Secure Shell (SSH) (port 22). The remaining features are: number of connections made, number of ports used by the source, number of ports contacted by the source, the sum of bytes sent by the source, and the sum of packets sent by the source [13].

3.2.2 OutGene

OutGene [13] is a network intrusion detection algorithm that uses anomaly-based techniques. It has several similarities to FlowHacker but is introduces two new concepts:

1. *Genetic zoom*: using a genetic algorithm that identifies the best subset of features leading to the formation of the same clustering output that all features generate. In this way, this algorithm provides a better and easier analysis of the data by humans.

2. *Time stretching*: analyse the flow of events in different time windows at different time scales. This way allows detecting stealth attacks that try to circumvent the detection system, using a low pace of execution. For example, with this notion, a slow network scan-attack can be detected if the traffic is analysed on an hour scale but not on a daily scale. Thus, with several types of time windows, the attack is consistently detected in at least one.

OutGene [13] execution comprises two phases:

1. *Pre-runtime phase* - is executed before the system is deployed and aims to define the extraction and normalization of generic features that will be used in the next phase. It can be divided into three steps: i) Defining normalization - transform and normalise the data to make all values consistent; ii) Feature Selection - choose which features are going to be extracted; and iii) Defining Feature Extraction - selects how the features are extracted.

2. *Runtime phase* - the stream of data that contains the traffic flows that are collected is processed. The processing starts with generic feature extraction and normalization. The extracted features are given as input to the clustering algorithm, which groups entities (typically hosts) with similar behaviour. If clustering produces outliers, there are two options: i) If this kind of outlier/anomaly has already been observed before, it can simply be reported, and ii) manual intervention by a human analyst is required, which is inevitable when the possibility of unknown attacks is considered.

3.2.3 DynIDS

DynIDS [14] is a network intrusion detection algorithm that uses anomaly-based and clustering methods to group hosts, identified by Internet Protocol (IP) address, with similar behaviour and detect threads. This behaviour is characterized by features extracted from network flows.

The main innovation of DynIDS [14] is the selection of part of the features based on traffic flow. This algorithm uses 12 static features, half relative to the source and half relative to the destination, and these static features are: number of different IPs contacted by an entity, number of flows where the entity is the source, number of different source ports used, number of different destination ports contacted, sum of total packets length receiver, and sum of total packets length sent. However, it also uses port-based features obtained dynamically, according to the data analysed in each time window. This uses four features for each

port (number of packets sent and received in a port, source and destination host). The dynamically defined features are identified based on the DYN3_X algorithm that filters at the $x/3$ ratio the ports that appear most in the flows, the ports that appear in fewer flows and the ports used by fewer machines. This approach does not limit the system's ability to detect attacks related to specific ports because the features are correlated with the traffic flow in the network.

As OutGene [13], DynIDS [14] also applies the concept of time stretching, allowing it to analyse data in different time windows. Regarding clustering, it applies three clustering algorithms: K-Means (partition-based), Agglomerative (hierarchical) and DBSCAN (density-based). In K-Means, the elbow method is used, where several k (number of clusters) are tested until the optimal number of clusters (k_{OPT}) for grouping of the data. This way, it is possible to obtain a better performance in identifying outliers, which is done by intercepting the results of the three algorithms.

3.2.4 C2BID

The Cluster Change-Based Intrusion Detection (C2BID) algorithm follows from the two above mentioned. It is able to use the static features of OutGene, however, for better performance, it uses the DynIDS approach for feature extraction, using both static and dynamic features. Using eight fixed features for the destination IPs addresses and the same eight features for the source IP, they are: number of different IPs contacted by the host; number of flows where the host is the source; number of different source ports used by the host; number of different destination ports contacted by the host; sum of total packets length received by the host; sum of total packets length sent by the host; average sent packet size and the ratio of the number of packets sent and their duration. Cluster Change-Based Intrusion Detection (C2BID) uses K-Means algorithm, based on elbow method, to create the clusters [41].

The novelty in the C2BID approach is to detect intrusions by monitoring changes in host behaviour. C2BID analyses the changes in clustering results and the hosts movements between clusters in certain time periods, including the appearance of new hosts or new clusters. This algorithm creates a history of the clusters the machines have belonged to, sequentially grouping the cluster values in various time windows. A history path represents the behaviour of a host seen from the network during a period T_a looking at windows of a defined size. Different attacks are easier to detect at time windows of different durations. A host which has no flows in a period, is assigned to a special cluster that contains inactive hosts. By studying the behaviour of the machines over time, it makes it possible to identify anomalous behaviour and suspicious cluster formation [41].

Outlier detection is done in three steps:

- Calculating the likelihood of each host doing a certain set of changes, taking into account all changes made in the analysed T_a. This is done in parallel for all different time windows;
- Application of the Robust Random Cut Forest (RRCF) algorithm [42], to detect probability values and outliers;

- Filtering of the results produced by RRCF according to previously chosen criteria;

3.2.5 Algorithms Comparison

The intrusion detection algorithms described above have been presented in chronological order of their implementation, being C2BID the most recent. These have been developed in a research work where improvements and different approaches are implemented to make these algorithms the most accurate in identifying anomalies and threats in the network.

The detection method of all these algorithms is anomaly-based and makes use of unsupervised ML, more specifically clustering, to analyse the network data. Another similarity between them is that they load netflow data (i.e., the traffic data from all the machines on the network) through CSV files and are run through the user's command prompt. There are also some characteristics that are found in all three algorithms, such as the use of time windows, the use of K-Means as the clustering algorithm, and some static features.

Table 2 summarizes the main characteristics of the different algorithms. As far as the features are concerned, several differences can be found. OutGene only extracts features for 4 fixed ports (22, 25, 80, and 194), so the type of features of this algorithm is considered static. DynIDS uses both static features and dynamic features, those that are relative to the ports. It is possible to see that the number of ports used in this algorithm is much bigger than in OutGene. C2BID integrates the extraction methods used in OutGene and DynIDS, in the last one, the number of ports to be analysed can be defined by the user, always knowing that they will be filtered by the DYN3_x algorithm. Another feature of C2BID is that it gives the user the option to choose which ports to analyse, generating static features for those ports.

With respect to outlier detection, through the clustering algorithms, we saw that DynIDS implements more robustness compared to OutGene since it is necessary that at least two of the three clustering algorithms identify a machine in an isolated cluster for it to be considered an outlier. In C2BID, although it uses only one clustering algorithm, but due to the fact that it makes use of a historical record of the machines and the clusters they belonged to, it is able to identify the outliers with an analysis of these records through RRCF.

Table 2: Comparison of the intrusion detection algorithms.

Algorithm	No. of ports	No. of Features	Features type	Time window	History	Clustering	Outlier detection
FlowHacker [12]	5	28	Static	No	No	K-Means	Isolated cluster
OutGene [13]	4	26	Static	Yes	No	K-Means	Isolated cluster
DynIDS [14]	100	Over than 400	Static + Dynamic	Yes	No	K-Means Agglomerative DBSCAN	Isolated cluster in 2 clustering algorithms
C2BID [41]	Adaptable	Adaptable	Static + Dynamic (or only static)	Yes	Yes	K-Means	Filtering of the results from RRCF

3.3 Summary

In this chapter, we have presented some tools that allow the visualisation of data related to cyber security, although with different goals. Methods of operation and the main approaches to data storage, processing, and visualisation of the tools were presented. We also described the intrusion detection algorithms (OutGene, DynIDS, C2BID) that are the basis for the operation of our visualisation tool.

4 Design of the Visualisation Tool

This chapter describes the phases that led to the development of the solution in order to meet the objectives described in Section 1.2, starting with the problem approach and going through the implementation of the GUI and server structures.

This visualisation tool aims at analysing datasets using intrusion detection algorithms based on clustering techniques. The traffic data of all the machines in the network (netflows) will have to be previously obtained by the user and loaded in the tool.

4.1 Approach

We have implemented a system based on a client-server architecture using a REST API to develop our visualisation tool. We chose this approach to remove the data storage and complex processing from the user side. The algorithms and the DB are on a server-side with more computing power to do the processing and more storage capacity. We use REST because it is a simple request/response mechanism. Its calls are message-based and follow the HTTP standard. We integrate the network intrusion detection algorithms in the back-end of our system and, through HTTP requests, also supports Hyper Text Transfer Protocol Secure (HTTPS). The results obtained are sent to the client and are displayed in the GUI, which is local (i.e., not a Web-Based GUI - a web application accessed by a browser).

Before developing the tool, we had to understand the requirements and resources to be provided. For this work, this becomes fundamental because it involves the integration of existing intrusion detection algorithms, so factors such as their implementation technologies (e.g., programming language, libraries, frameworks), operation structure (e.g., data flow and processing) were taken into account.

4.1.1 Requirement Analysis

In this work, to achieve the proposed objectives previously presented in Section 1.2, it was necessary to identify the requirements and functionalities that the visualization tool should include. After this survey, it was necessary to proceed to their selection based on their usefulness for the development of the proposed solution. This way, considering the extension and relevance of the requirements initially available, we need to divide the requirements into 2 groups: 1) those that are part of the Minimum Viable Product (MVP), that is, that are essential to achieve the objectives; 2) those that are not part of the MVP, immediately, but that make the tool more useful and intuitive to the user.

Always keeping in mind that the visualization tool should be intuitive and clear, the requirements are:

1. The tool has to be able to show simply and clearly which threats were found in a given period of time and provide information (e.g., IP address) of the potential attackers and victims that have been identified.

2. The analysis of intrusions should be performed using at least the following previously implemented detection algorithms: FlowHacker, OutGene and DynIDS. It should be possible to change the configuration of the data analysis in the various IDSs (e.g., parameters of the algorithms, time window size), the user should provide these values according to the analysis he wants.

3. Provide statistics (e.g., network activity over a period of time) that allow the user to decide how to analyze the data. It should also provide the data that led to a certain result (e.g., how clusters are distributed) to help the analyst decide whether the threat is real or is a false positive. This data will preferably be displayed graphically, for better and faster prediction of what is happening.

The requirements that do not compromise the goals of the visualization tool but add value to the tool are:

1. Considering a development environment and in order to analyze intrusion detection algorithms, with certain chosen parameters, it should provide the metrics of the evaluation (e.g., TP, FP, TN, FN, F1-score, Precision, and Recall) of the algorithms performance. In order to obtain these metrics, it is necessary that the user provides a list where the machines (attackers and victims) are identified, and also the start and end time of the respective attacks must be described.

2. In case the user knows that a certain IP belongs to a machine that is considered safe, but has a suspicious behaviour, e.g., a router that generates a lot of traffic can be considered a suspicious behaviour when compared to hosts, the user should be able to inhibit it (e.g., to create a whitelist) from being shown in the tool as a possible intrusion.

3. The tool should be able to analyze the number of events in the input file and be able to group these values by time intervals. The user must be able to download, in CSV format, the data obtained from the clustering and the recorded activity.

4. The user should have the possibility to graphically customize the view of the tool (e.g., colors, scale of graphs). This aspect has been considered in order to make the user's activity more comfortable and ergonomic.

5. There should be a help button that shows the necessary information in each view of the tool in order to clarify any doubts the user may have while using the tool.

After presenting the requirements of the visualization tool, it is necessary to mention that in this proposal, it is essential to develop a functional back-end[4] that is fast in processing and makes the existing data available in a simple way. Not only because the processed data will be displayed in our GUI, but also to allow easy access to the data by anyone interested in obtaining it.

[4]Refers to any part of a software program that users do not see is the data access layer. Back-end processes include: processing webpage requests, running scripts, handling the storage, updating, and accessing of data in a database.

4.1.2 Architecture

The system developed has a client-server architecture, supported by a REST API as shown in Figure 1. We chose this modular approach considering the amount of data to process and its storage, by implementing the entire back-end on servers with greater processing and storage capacity, the system's performance increases, which would not happen if the tool was exclusively local (implemented only on the client-side). Another criterion for the choice of this structure was to guarantee easy and optimized access to the data already processed by the server.

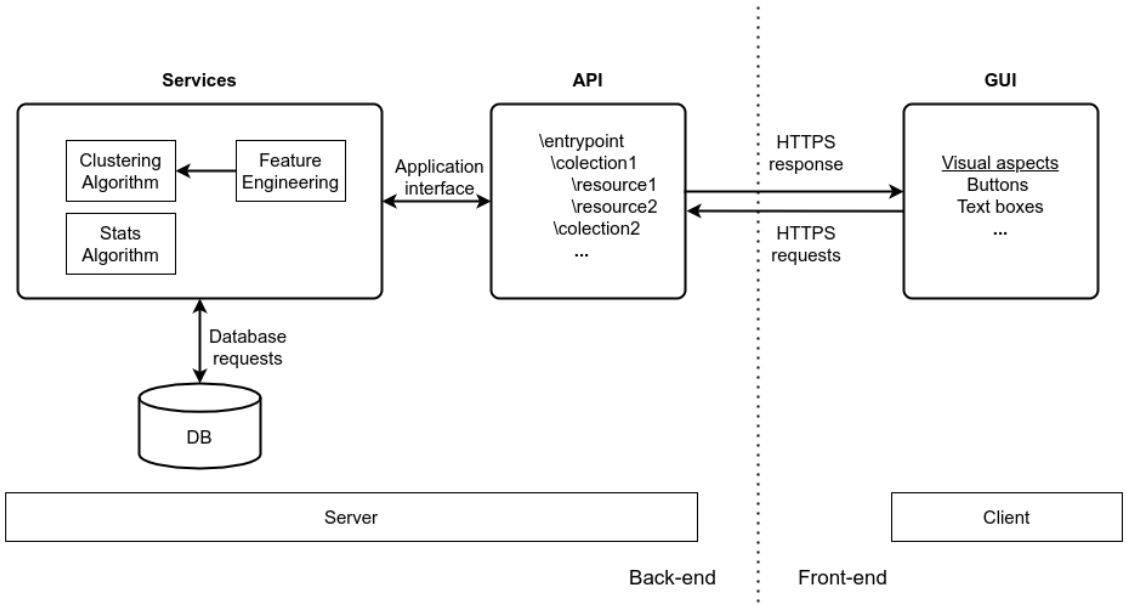

Figure 1: System architecture.

On the server-side is the tool's back-end, which does the processing, storing, and data transmission. This is composed of a DB, a data processing module (represented in Figure 1 by Services), where the algorithms responsible for feature engineering, clustering, and obtaining statistical data are located. The API is also allocated on the server-side, which guarantees communication between the back-end and the GUI (front-end[5]). The communication is done through HTTP requests made by the client and HTTP responses given by the server. This interaction will be presented in more detail in Section 4.4.3.

The data flow of our system is presented in the flow chart in Figure 2, starting with raw data and ending with obtaining processed data (results). These steps of data handling, preparation, processing and storage occur in an automated way, through scripts, functions, and data structures implemented in the back-end of our system. Concerning data storage and processing are explained in detail in Section 4.4.1 and 4.4.2, respectively.

[5]Refers to the part of the application that the user interacts with directly, includes visual aspects such as text colors and styles, images, graphics, tables, buttons, and navigation menus.

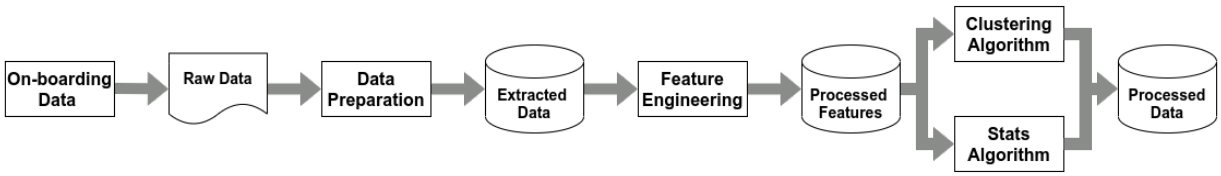

Figure 2: Directional data flowchart of the system.

The first step is to process the netflow data (log files[6]) to use only the necessary parameters and generate an input file. This input file should be provided in CSV format and the columns should meet the requirements shown in Table 3. If the flow is uni-directional, it should be indicated to the system and columns Tot Bwd Pkts and TotLen Bwd Pkts can be omitted.

Table 3: Required columns of the input csv file.

Column name	**Description**	**Format**
Src IP	Source IP adress	Varchar(n)
Src Port	Source port	Integer
Dst IP	Destination IP address	Varchar(n)
Dst IP	Destination port	Integer
Tot Fwd Pkts	Total forward packets	Integer
Tot Bwd Pkts	Total backward packets	Integer
TotLen Fwd Pkts	Total forward packet length	Integer
TotLen Bwd Pkts	Total backward packet length	Integer
Flow Duration	Flow duration	Integer

After the input file is obtained, in Data Preparation module the data are verified and validated to proceed to the next phase with the assurance that the data are correct and in the expected format.

The next step is Feature Engineering where the required features are selected, extracted, and normalized for later use in the algorithms. This step will depend on the extraction method used and how it is configured, i.e., this step depends on the user inputs that define how the features will be extracted. Using OutGene or FlowHacker only fixed features related to the port list used by these IDSs algorithms will be selected. If DynIDS is used, fixed and dynamic features are extracted based on the number of ports used.

In the flowchart in Figure 2, it should be noted that the features already processed will be stored so that they can be used later without having to process them again. This reduces the computation used and increases the execution speed of the program, on the other hand, it will increase the necessary storage space.

Regarding the clustering algorithm module, this is where the clusters will be created based on the processed

[6]They are the main source of data for the network. It contains information about usage patterns, activities and operations within an operating system, application, server or other device.

features. The data obtained will be stored (Processed Data) and can be used for future analysis (e.g., to obtain the history path for C2BID) or to be shown to the user. The same happens with the statistics algorithms that are responsible for the collection and organization of the statistical data (e.g., the number of events that occurred in one day), the processed data will be saved for later display to the user.

4.2 Implementation

In this section are presented which technologies, packages, and libraries were used in the tool. In Section 4.2.2 the steps that led to the creation and development of the tool will also be presented. All code and instructions are available at: `https://github.com/a3ceProject/CyberVTI`, *a3ceProject*'s public repository on Github.

4.2.1 Technologies

The existing intrusion detection algorithms that are integrated into the tool, FlowHacker OutGene and DynIDS, are implemented in Python and use as their main libraries:

- Pandas [43]: It is an open-source, BSD-licensed[7] library that provides high-performance, user-friendly data structures and data analysis tools for Python. Pandas allows data manipulation with integrated indexing in a fast and efficient data structure called DataFrame. It provides tools that make it possible to read and write data in several different formats.

- Scikit-learn [44]: It is an open-source, BSD-licensed Python library that integrates a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This library has easily and efficiently integrated Classification, Regression, Clustering, Dimension Reduction, Model Selection, and Preprocessing tools for data analysis.

Considering the programming language in which the detection algorithms were developed and the aspects mentioned in Sections 4.1.1 and 4.1.2, we chose to use Python (version 3.8) as the base language to effectively integrate the various modules to be used in the tool. Python [45] is an open-source computer programming language optimized for quality, productivity, portability, and integration with other programming languages such as object-oriented programming. Python was chosen because it is a globally applied language for Internet scripting, system programming and graphical interfaces. Another reason for this choice was that the IDSs algorithms we proposed to integrate in the tool were developed in Python, making it easier to integrate them.

For the implementation of the REST API, we used *Flask* [46] [47], which is a web microframework. Unlike frameworks, which are a collection of packages or modules that allow programmers to write applications, microframeworks are not limited to specific libraries. Their main purpose is to provide a simple and extensible application core. This allows the developer to choose which systems to integrate and how (e.g., what database to use).

[7] Type of low-restriction license for open-source software that places no requirements on redistribution.

As far as the database is concerned, this was implemented in *SQLite* [48]. This is an open-source embedded relational database, designed to provide a convenient way for applications to manage data, decreasing the problems sometimes encountered with dedicated relational database management systems. *SQLite* has as its main goal to be highly portable, easy to use, compact, efficient, and reliable. However, there are some input files that are stored on the server side in specified directories, because there is no need to store them in a database.

To manage and interact with our database, we use *SQLAlchemy* as a compatible interface between *SQLite* and Python. *SQLAlchemy* [49] is a Python library that provides a high-level Pythonic[8] interface to relational databases. Additionally, it includes an Object-Relational Mapping (ORM), that lets application developers the flexibility of SQL (e.g., allow to write SQL queries normally or using Python expressions).

To save the results of the feature extraction, *parquet* files were used. *Parquet* [51] [52] is an open-source file format available that offers columnar data storage format different from row-based files such as CSV files. *Parquet* is optimized to work with complex bulk data and has different ways of efficiently encoding data. This approach is best especially for queries that need to read certain columns from a large table. Since Pandas can handle this format and because it is faster to save and access the data and does a better data compression compared to CSV files, it was decided to use *parquet* files.

In the front-end, *Tkinter* was used for all graphic development of the visualization tool. The *Tkinter* [53] module is a standard Python interface based on a thin object-oriented layer on top of Tcl/Tk[9] for building a GUI. However, for management and for making the graphs, we used *matplotlib* [54], a Python package that allows constructing graphics efficiently. It is also allowed to interact and change the visualization aspects of the graphs and insert information that leads to a better perception of the displayed data.

The technologies we used to support the development of our entire system were the following:

- Visual Studio Code (VS Code) - code editing platform that contains a lot of useful resources for development.
- DB Browser for SQLite (DB4S) - open-source interface to view, edit, and interact with DB .sqlite files.
- Postman - application that allows to make HTTP requests from a simple and intuitive interface, making it easy to test and develop the API.
- Pipenv - Automatically creates and manages a virtual environment (virtualenv) for the project, making it easy to set up a working environment. It also allowed the creation of a file with requirements (e.g., requirements.txt) where all libraries (and their versions) that are used will be placed, making it easy to set up the environment in which the project is inserted.
- GitHub - code hosting platform, allowing us to store and share the project's code and its various versions.

[8] Describes a coding style that uses Python's unique features [50].

[9] Open-source widget toolkit that provides a library of basic widgets for GUI. Adapted from: `https://www.tcl.tk/`

4.2.2 Development Phases

When the development of the visualization tool was started, the phases that are described below were followed. However, these phases do not have a strict sequence and have been changed throughout the development, i.e., changing the structure of a phase may imply having to go back to the previous phases to readapt them.

1. Since this is an implementation of previously developed IDSs algorithms, we started by reviewing their code to understand their general operation. In a more particular analysis, special attention was given to the data structures and functions implemented, thus allowing us to identify which parts could be reused and where changes would need to be made.

2. Understand what data, and in what format, would need to be stored in the DB in order to design its architecture. The DB tables and functions for storing and accessing the data were then created, also we develop functions for checking and validating the data to ensure the integrity of the system.

3. Create the functions that generate the processed data (e.g., features, clustering, stats) and integrate them with the DB. Unit tests were implemented to verify that at no point does the data get processed incorrectly.

4. We defined resources, endpoints[10] and methods that should be in the API. Afterwards, these were implemented to allow the data in the DB, already processed, to be accessed by an HTTP request.

5. Structure the visual appearance of GUI and design how to interact with the graphical environment. After the main functions were implemented in the back-end, we could start developing the GUI. For this it is necessary to create the HTTP requests that will get the data to be displayed.

6. Test the behavior of the tool's functionalities and test the results obtained using a previously known dataset.

7. Review the entire implementation and add features relevant to the tool.

It should be noted that in all these phases, good programming practices must be taken into account. The code must be readable, organized, extensible and simple for the whole system to perform well and be efficient.

4.3 Graphical User Interface

This is the component of our system that allows us to achieve the proposed objectives presented in Section 1.2. The GUI offers an abstraction layer that allows any type of user to use the tool. Since it is intuitive and

[10]It is one end of an API's communication channel, from which the API can access the resources it needs. Adapted from: `https://smartbear.com/learn/performance-monitoring/api-endpoints/`

simple, it does not require knowledge of programming or an exhaustive reading of the system documentation to use it. The user can visualise and analyse the data more effectively as it is presented graphically. The functionalities present in the tool are also all presented graphically (e.g., buttons, lists, menus), facilitating the use when comparing to the execution of scripts through the command prompt/terminal. Features were added to the system to obtain relevant data for the user and to provide flexibility in data analysis, for example, the user can choose the time interval that he wants to analyse the file, not being restricted to fixed start and end time values. The main advantage of having a GUI is to optimise the use of algorithms.

4.3.1 Installation

The application's GUI and the necessary files for its execution are supplied in a .zip file depending on the Operating System (OS) where the tool will be used. After unpacking the .zip file, the installation and execution will be done in different ways.

- **Windows** - There is an executable file (CyberVTI_GUI.exe), generated through *PyInstaller* that groups all the components of the graphical application and its libraries in a package. When running this executable the application starts.
- **Linux** - It is necessary to install Python (version 3.8) and the libraries to run the tool. There is a file that makes these installations automatically called installer.sh and should be executed in the terminal with the following code:

```
sudo ./installer.sh
```

To avoid issues with permissions, the following commands should be run:

```
sudo chmod +x installer.sh
```

or

```
sudo chmod 755 installer.sh
```

To start the application, just run the command:

```
sudo ./CyberVTI_GUI.sh
```

The first time the application is executed, the user will be asked to insert the username, after this step the application is ready to be used.

4.3.2 Initial View

The first time the application is opened, the user will not find any files loaded, as shown in Figure 3. It is shown a simple window containing: 1) a navigation menu that will be explained in detail later (Section

4.3.5); 2) two buttons, one to upload a file and the other to refresh the page after uploading, giving the user a view as shown in Figure 4.

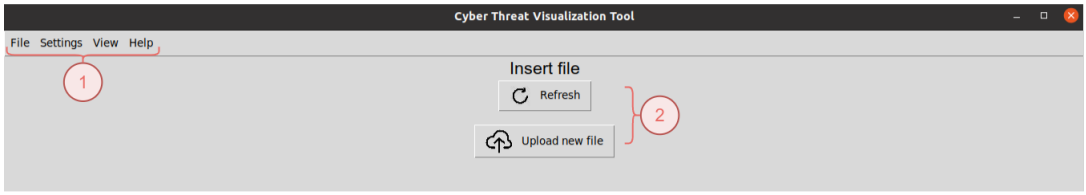

Figure 3: Initial view without files.

On Figure 4 view, it appears: 3) a list of files that the user has access to; 4) a drop-down menu and a button that allows the user to select a file to be deleted. Note that when a file is deleted, all the data extracted from it (e.g., features, clusters) are also deleted from the DB on the server side.

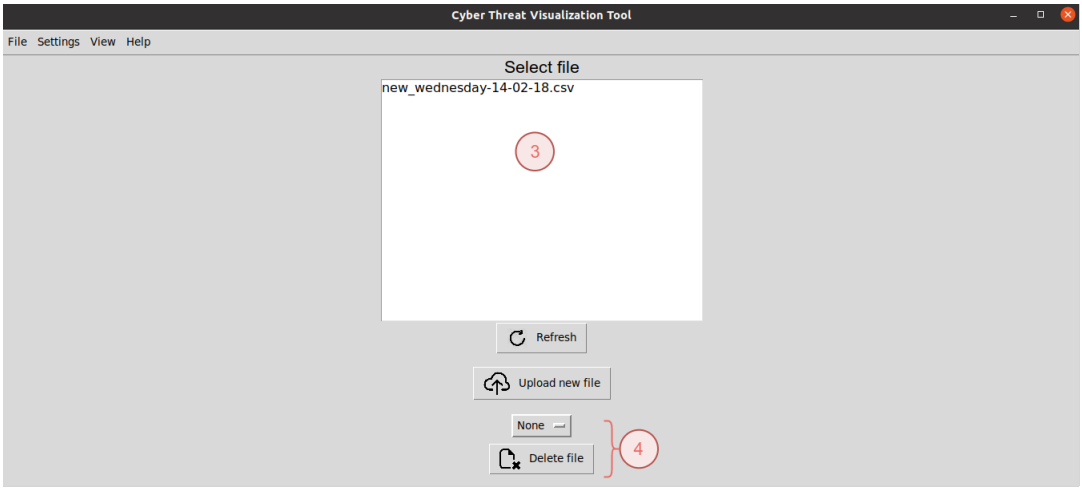

Figure 4: Initial view with files.

This is the view in which the application will open if the user already has files inserted in the server. To advance to the file analysis view, the user must double-click on the file's name (on the file list) that he wants to analyse.

4.3.3 Analysis View

When a file is loaded in the application, several default analyses are performed. However, at this stage, the user can choose the parameters to perform new analyses or proceed to the results. Figure 5 shows the view that the user will have, this can be divided into 5 parts corresponding to its function:

1. It shows the name of the selected file and a graph with the number of events in the file, where the x-axis is the timestamp of the events.

2. To choose the time interval for a new analysis of the file. The user must select the day for the start and end of the analysis from the drop-down menu. The drop-down menu is only filled with values for the days between the oldest and the most recent event in the file. If the user wishes, he can also define the time for the beginning and end of the analysis, with the format HH:MM:SS (hours, minutes and seconds), if uses the defaults times (keep the word: default), then the start time will be 00:00:00 and the end time will be 23:59:59.

3. An input box allows the user to choose the time window (integer values) for the analysis or analyses because the user can put more than one time window (separated by commas) and thus perform several analyses. It is also necessary to choose the method that will be applied to extract the features.

4. Back button, the user goes back to the initial view.

5. This is where the functionality buttons are: start analysis, view the events graph in more detail and view results obtained.

Figure 5: File analysis view.

4.3.4 Results View

Regarding the visualisation of the results, the first view that the user has is the one presented in Figure 6. Identified by point 1 on Figure 6 is the name of the file that is being analysed. In point 2 is where the user will have to choose the maximum number of elements that are in the same cluster. To find outliers, just leave the default value of 1, so only the clusters with one element will be shown. The user can also choose the view of the IPs on a network that are shown in the drop-down menu. This view can be: internal (Int)

- only IPs belonging to the network or external (Ext) - only IPs outside the network. Point 3 of Figure 6 shows a list containing the parameters of the data processing already performed. To move on to the result view, the user must double-click on a line of the list of parameters of the processed data to be displayed.

After choosing the file analysis, the user is presented with a list of results (clusters) obtained, as shown in point 4 of Figure 7. This list comprises the cluster timestamp, the cluster number, the number of machines in that cluster and the IPs of those machines. In case the maximum number of elements per cluster is equal to 1, the IPs shown are those that have been identified as attackers or victims of cyber attacks.

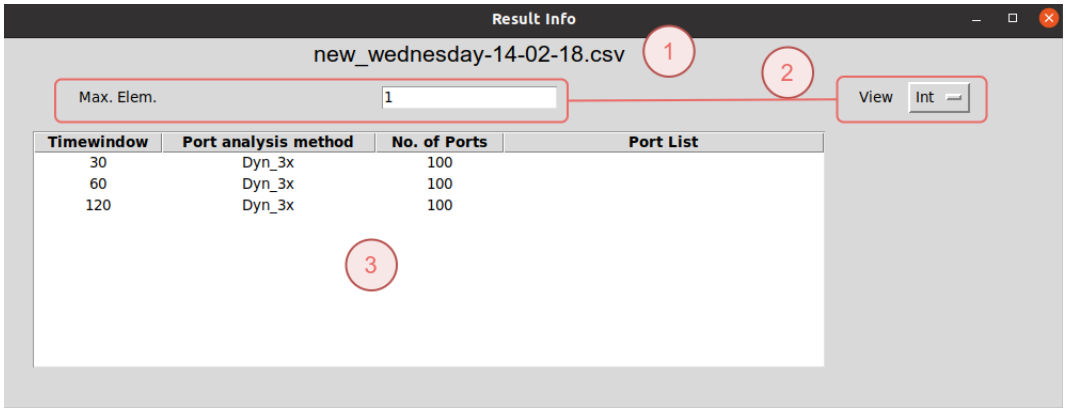

Figure 6: First view of the results.

In the area of Figure 7 marked by point 5, three functionalities are shown:

1. Display the heatmap of the data obtained for the chosen timestamp.
2. Save in CSV format, in a user directory, all the clusters obtained from the file through the selected analysis
3. Present the metrics corresponding to the performance of the analysis performed.

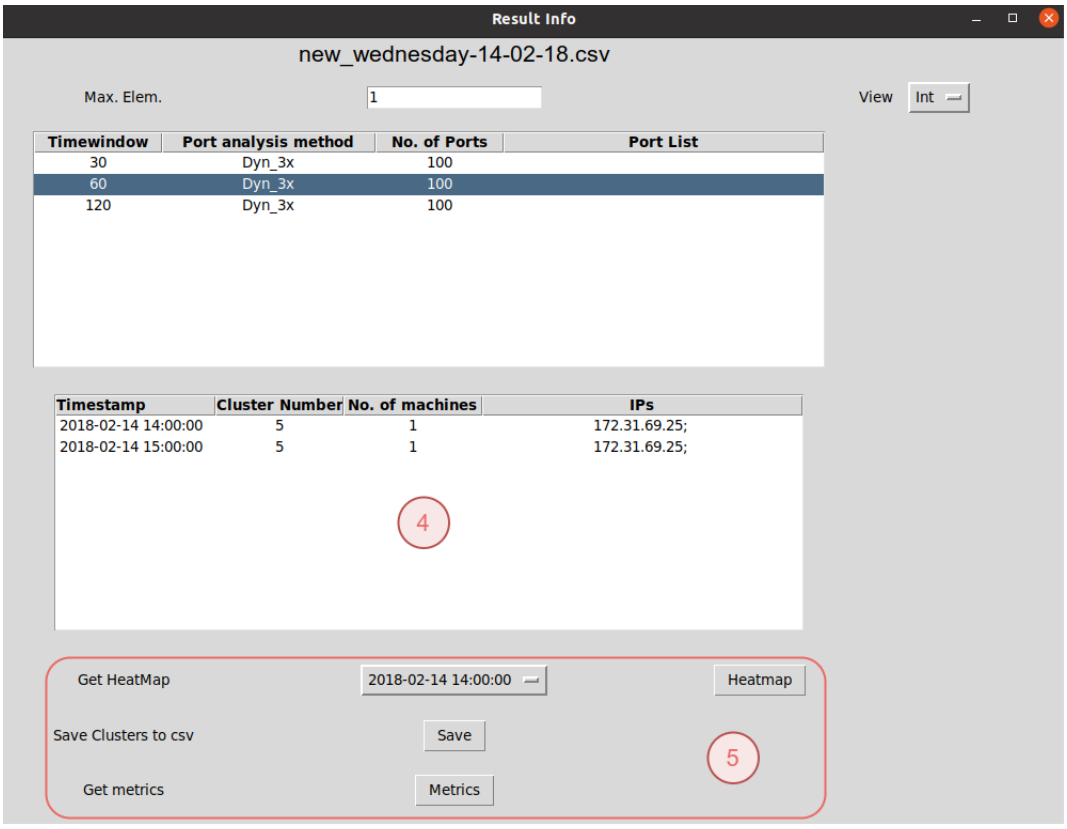

Figure 7: Second view of the results.

The metrics view of the performance analysis is presented in Figure 8, in the zone identified by point 1, we have the metrics that were obtained through the total values of the following classifications: TP, TN, FN and FP, the formulas for these calculations are presented in Section 5.2. In point 2, the metrics are shown but divided by their respective analysis intervals.

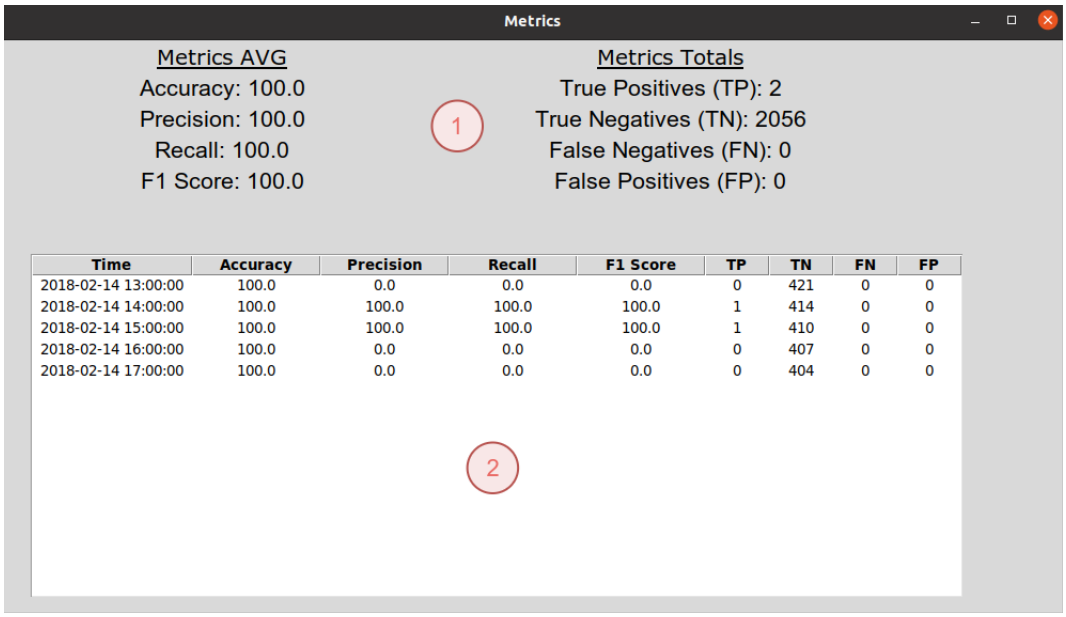

Figure 8: Algorithm evaluation metrics view.

When the user clicks on the button to view the heatmap for a particular timestamp, a new window is opened, as shown by Figure 9. On the x-axis are represented the analyzed features, on the y-axis are the numbers of the various clusters and on the z-axis, represented by the colour scale, are the normalized values (between 0 and 1) of the features. The features that have more influence are the ones with the most different values between clusters, features with the same approximate value in all clusters are irrelevant. In the example shown in Figure 9, it can be seen that in cluster 7 there is an attack on port 22 and it must be bruteforce because it uses source ports that others do not.

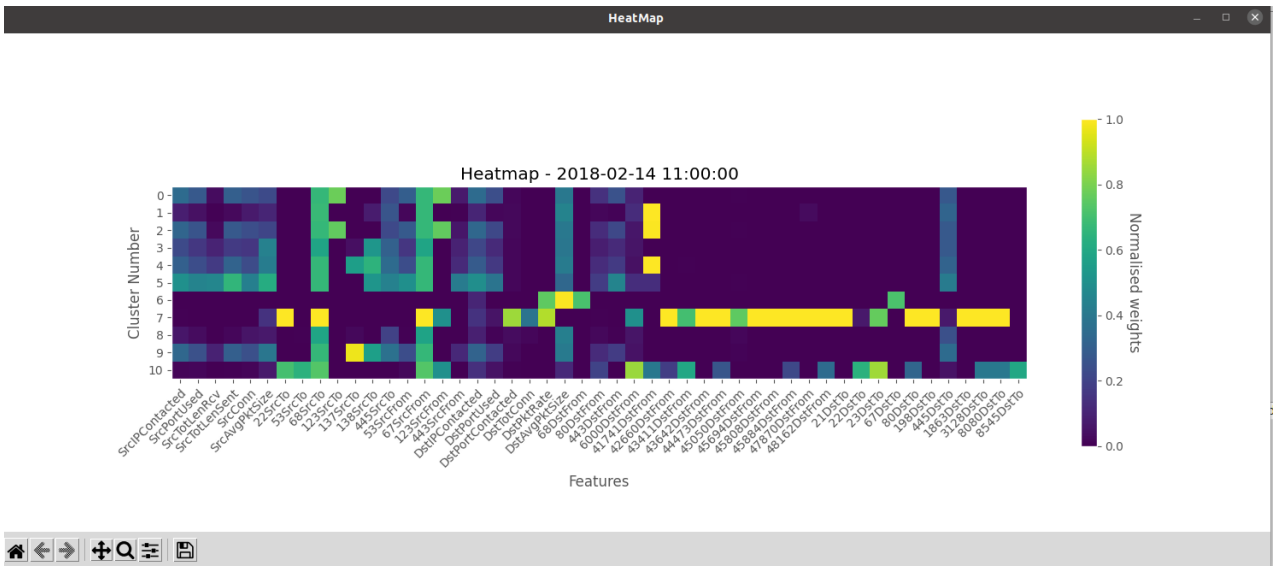

Figure 9: Displayed heatmap.

4.3.5 Navigation Menu

The GUI of the tool has a navigation menu in the upper part of the window, which is divided into 4 submenus: Files, Settings, View and Help, as shown in the diagram in Figure 10.

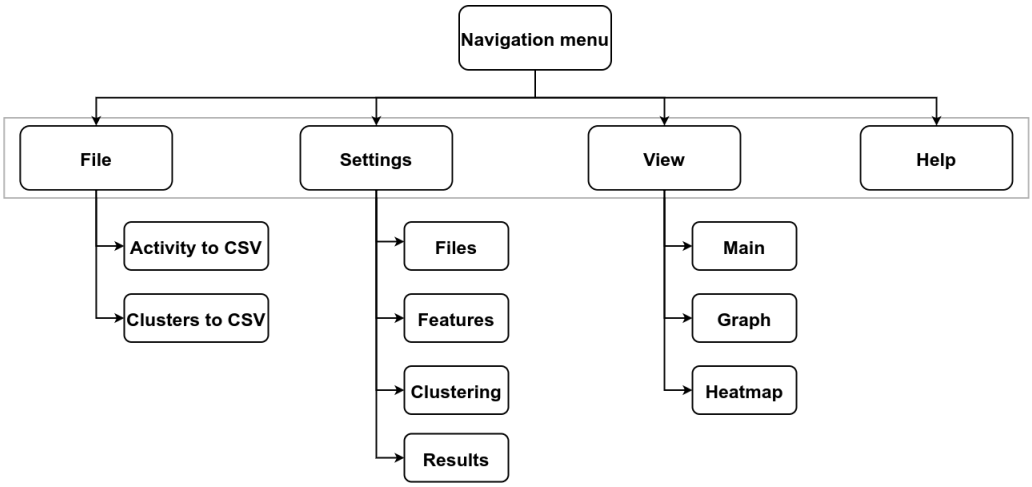

Figure 10: Diagram of the navigation menu.

The functionalities of the sub-menus are described below:

- **File**
  - Activity to csv - saves the total number of events at certain time intervals, in CSV files, in a local directory of the user, whose path is: Saved/<File_name>/Activity.
  - Cluster to csv - saves all clusters of a given analysis, in CSV format, in a local directory of the user, whose path is: Saved/<File_name>/Clusters.
- **Settings**
  - File - In this settings menu, shown in Figure 11, the user has the possibility to change parameters related to the file upload: i) *Frequency* - defines the time intervals in which the number of events in the file are grouped, this parameter is of type string and the intervals must be separated by a comma, as the format: <value><units>(units can be: 'min' - minutes, 'H' - hours or 'D' - days); ii) *Date Format* - timestamp format used in the input file. If the datetime format is integer, the user must put in this field the word 'Integer' or just 'Int', the file will be parsed and the timestamps will be converted to string in the format "YYYY-MM-DD HH:mm:SS"; iii) *Default Time Windows* - list of time windows used for the initial analysis (when the file is loaded); iv) *Default Method* - extraction method used for the initial analysis; and v) *Path* - sets the directory in which the new file upload window will open.

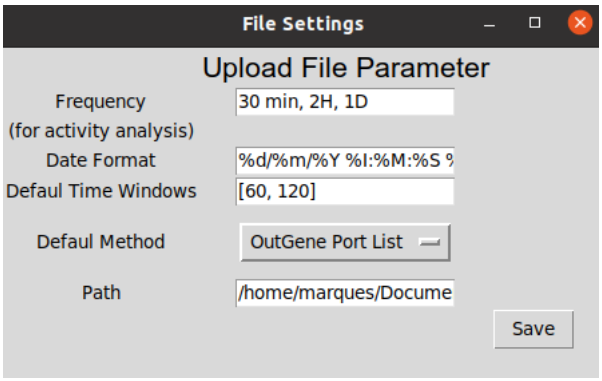

Figure 11: File settings window.

– Features - In the features settings window of Figure 12, the user can configure: the port number used in a dynamic features analysis (Dyn3_x), the IP prefix that will define which external and internal network IPs, and change the port lists used for a static feature extraction.

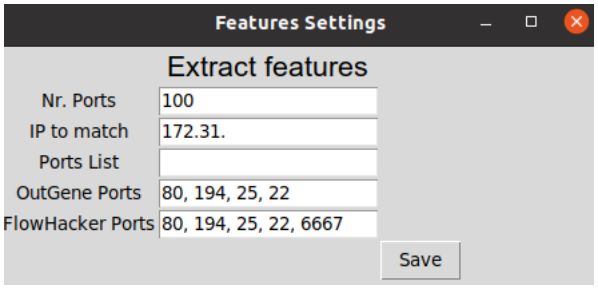

Figure 12: Features settings window.

– Clustering - In the submenu that concerns the cluster settings, in Figure 13, the user can define the number of clusters to apply, if this value is -1, the elbow method will be used to find the best value for k. It is possible to change the minimum and maximum values that will be used in the elbow method.

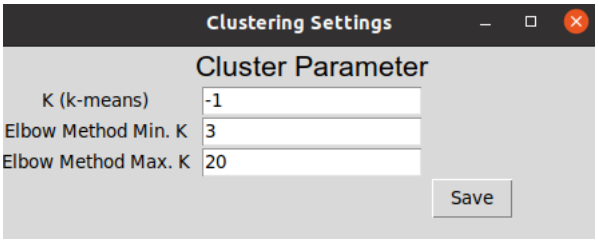

Figure 13: Clusters settings window.

– Results - In the submenu of the results settings in Figure 14, the parameters used in the calculation of the performance metrics can be changed. Two lists are used: 1) white list - where are placed the

IPs addresses of machines that the user is sure are not threats and that the algorithm should not consider, e.g., a router due to its high traffic flow, its behaviour can be interpreted as a threat, but if it is in this list in the metrics calculation will not be considered; 2) red list - the IPs addresses of the machines that the user knows are attackers or victims, and the time intervals in which they are active. Both lists are used to calculate performance metrics and previously labeled datasets.

In Figure 14, in point 1 are the white list and the option to show its IPs if they are considered threats. In point 2, the red list is shown, the id, the IP address, the start and end time of the attack, and the respective view about the network. In point 3, the user can add or remove an element from the red list.

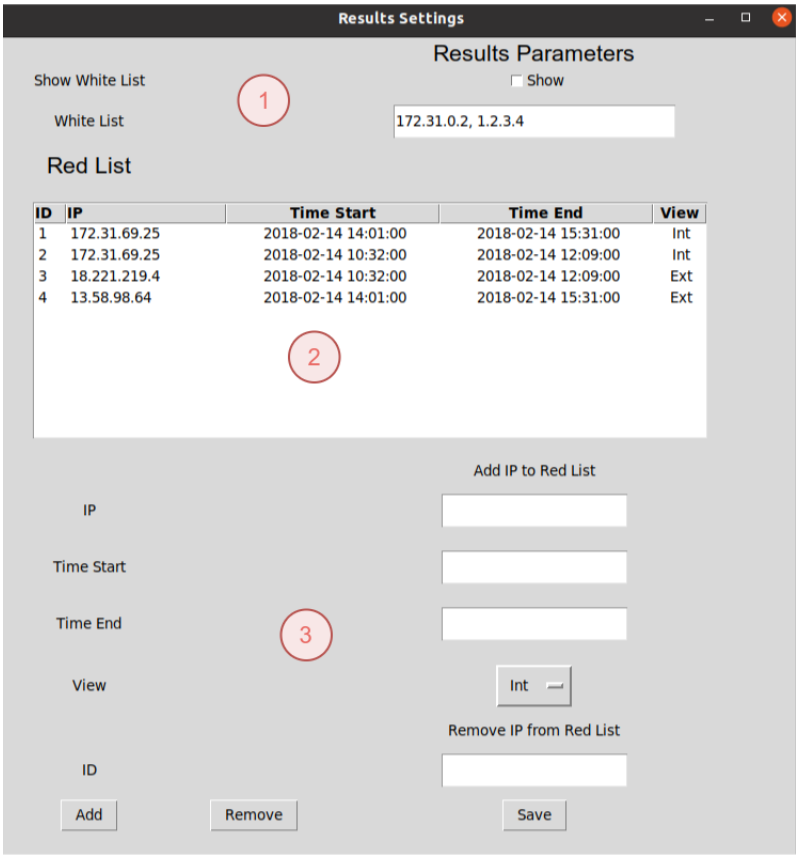

Figure 14: Results and metrics settings window.

- **View**
  - Main - In the main view window, shown in Figure 15, the user can change the parameters corresponding to the GUI view, such as: window length and width, the background colour, and the font type and size.

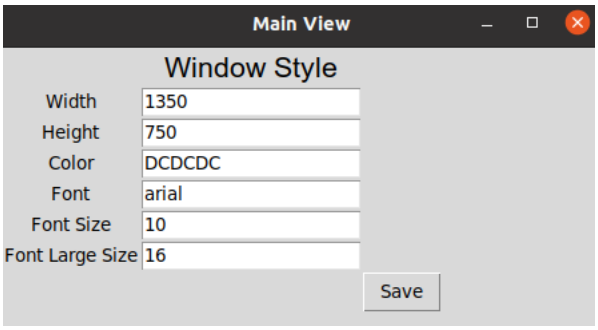

Figure 15: Main display settings window.

– Graph - In the window of the graph view settings, shown in Figure 16, the user can change: the name that is shown in the window, the size of the graph, and its resolution, by changing the figure Dots Per Inch (DPI). Users can also change the time intervals of the results displayed and choose a minimum number of events for which they should be shown on the graph. The scale used in the graph can be logarithmic if this option is enabled.

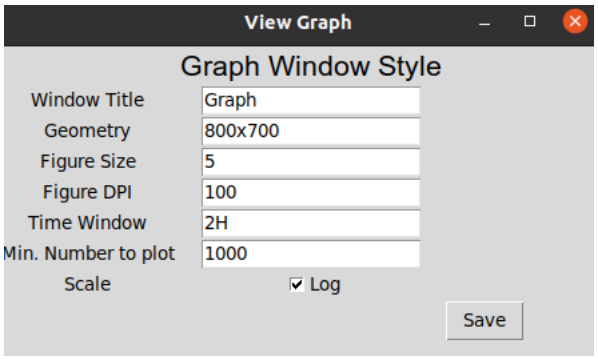

Figure 16: Graph display settings window.

– Heatmap - In the heatmap view settings window, in Figure 17, the user can change the dimensions of the window, the colour theme used in the heatmap. The user also can choose if he wants the colour scale bar to be visible or not, and if he wants the values of each data on the z-axis (colours) to be shown on the heatmap.

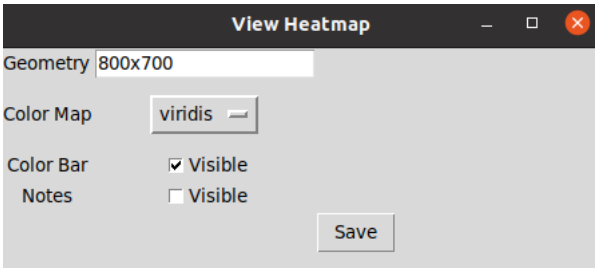

Figure 17: Heatmap display settings window.

- **Help**

  In this submenu, the user will have access to the description of the procedures to help him use the graphical interface. The helps are divided by the different views of the GUI, being them: Initial View (4.3.2), Analysis View (4.3.3), and Results View (4.3.4).

All these settings parameters, shown in the windows above, are saved in a configuration file (.cfg format), which is stored in the path of the visualization tool.

4.4 Back-end

This part of the work, although not visible to the user, is important because it is on the server-side that all the resources, services and functions that allow the tool to be used are implemented. It is fundamental to allow efficient access to data. For that reason, it is necessary to have a solid and stable API as intended in the objectives (Section 1.2). The main server-side modules will be introduced.

4.4.1 Database Structure

For efficient storage management, the first step is to know what data is indispensable to save and how to do it. The DB implemented in *SQLite* is divided into 4 tables: *Users*, *Files*, *Features_Metadata*, *Clusters*, as shown in the diagram in Figure 18. Are also represented the Primary Key (PK) and Foreign Key (FK) of the tables that allow the relationship between them.

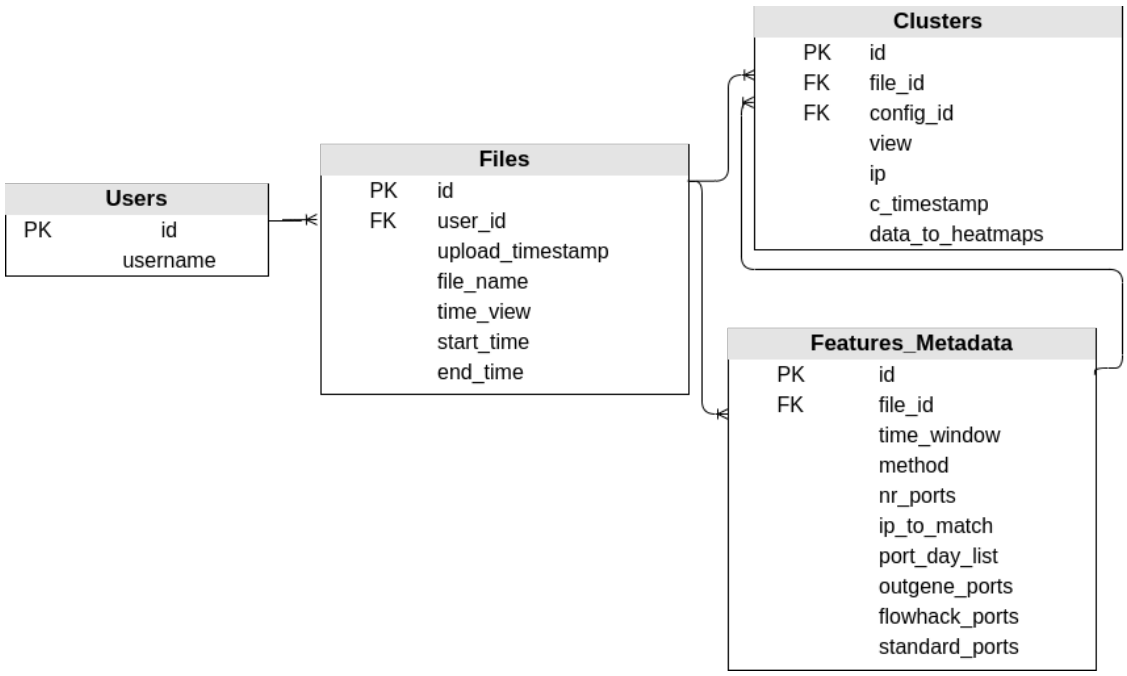

Figure 18: DB architecture diagram.

Once the data for a file is stored on the server, the user who entered it is associated with the file so that we can differentiate between files with the same name placed by different users. Thus, a table for users was created to solve this problem and make the tool extensible in a later production phase. However, no user authentication is performed because this was not the main focus of the development. The user is created and loaded into the DB at the installation phase of the tool. This way it is prevented that the same file (with the same name) is inserted by different users. The Users table (Table 4) is simple and has the user *id* as PK and the *username* that is unique constraint and cannot be null.

Table 4: Users DB table.

Users		
id	Integer	PK
username	Varchar(n)	U

After the user makes a request to upload the input file, in CSV format, it reaches the server where it will be checked against the aspects mentioned in Section 4.1.2. Once everything is correct, the file is validated, analyzed and saved. This storage of the input file occurs in two phases: 1) the file is stored in its entirety in a server directory called *Data_files*, where there is a folder relative to the user; 2) the metadata related to the file is stored in a *SQLite* DB table, as shown in Table 5. This table has as Primary Key (PK) the file *id* and as Foreign Key (FK) the *user_id* and the *file_name*, making it possible to access the file through these two FKs. We also store in this table parameters like: *upload_timestamp* - date the file was uploaded to the DB; *time_view* - this column contains data generated after the file analysis. This data is grouped, according to user defined intervals, of the number of events present in the file. Its format is Binary Large Object (BLOB) for space optimization because the analysis intervals are not restricted, i.e., the user can group the events according to the analysis intervals that he wants; *start_time* and *end_time* - contain the dates corresponding to the oldest and most recent event in the file, respectively.

Table 5: Input file metadata DB table.

Files		
id	Integer	PK
user_id	Integer	FK
upload_timestamp	Varchar(n)	
file_name	Varchar(n)	
time_view	Blob	
start_time	Varchar(n)	
end_time	Varchar(n)	

The features extracted from the input file are saved in a server-side folder in a *parquet* file. The path to these files is shown in Figure 19. The metadata corresponding to the extracted features is also saved, so it is possible to identify the configuration that led to the extraction of the features. This will decrease the amount of processing and storage since no features will be extracted from the same input file whose configuration is the same.

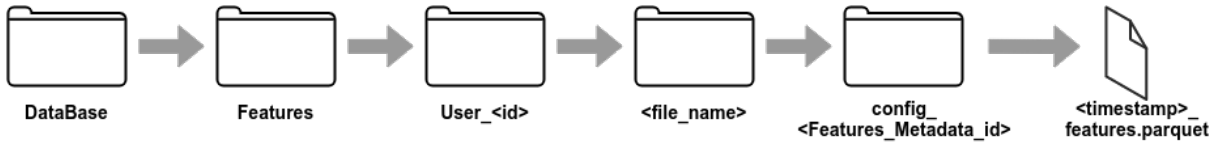

Figure 19: Path of the features files on the server.

This approach is due to the fact that the columns obtained from feature extraction are not fixed. The header and data in these columns will depend on the extraction method, the ports being used, and even the input file. To store this kind of data, it would be better to use Not only SQL (NoSQL)[11] DBs and since we are using an SQL DB, we decided to store this data in *parquet* files. This way we solve the problem of saving the extracted features independently of the columns that were generated.

The *Features_Metadata* Table, shown in Table 6, is identified by its *id* (PK) and has the columns of the parameters used in the features extraction: *file_id*, *time_window*, *method* - the method to extract the features (e.g., OutGene); *ip_to_match* - is the IP address prefix in which the features will be divided, is used to identify which network the IP addresses belong to. There are also parameters related to ports, in these we have port lists like *port_day_list* - it is an independent list that allows the user to run the analysis on the ports that he wants; *outgene_ports*, *flowhacker_ports* - lists the default OutGene and FlowHacker ports respectively; the last parameter stored is *standard_port*s - is a flag[12] that allows to identify if the port lists are the default ones or not, this avoids comparing the various port lists, when we want to know if a certain extraction configuration already exists. None of these parameters can be null.

[11] DB that are not based on tables, and store data differently from relational tables, from: https://www.mongodb.com/nosql-explained

[12] Is a value that acts as a signal for a function or process, obtain from https://techterms.com/definition/flag

Table 6: Input features metadata DB table.

Features_Metadata		
id	Integer	PK
file_id	Integer	FK
time_window	Integer	
method	Integer	
nr_ports	Integer	
ip_to_match	Varchar(n)	
port_day_list	Blob	
outgene_ports	Blob	
flowhacker_ports	Blob	
standard_ports	Binary	

The Clusters table, shown in Table 7, is where the clustering results are stored. It uses id as the PK, but clusters can also be identified by *file_id*, *config_id* (id of the feature extraction configuration) and IP address. In this table, the data are: *view* - type of IP network view, it can be external or internal, it will depend on the selected *ip_to_match* that will identify the prefix of a certain network; *c_timestamp* - is the date that corresponds to the cluster; *cluster_number* - is the number of the cluster to which the IP belongs; *data_to_heatmap* - is all stored data, in BLOB format, that will allow to generate the heatmap for the cluster.

Table 7: Clusters DB table.

Clusters		
id	Integer	PK
file_id	Integer	FK
config_id	Integer	FK
view	Varchar(n)	
ip	Varchar(n)	
c_timestamp	Varchar(n)	
cluster_number	Integer	
data_to_heatmaps	Blob	

All the data that are stored in the presented tables are accessed through functions implemented on the server-side using sqlalchemy, that allows using filters and SQL queries to access the data.

4.4.2 Processing Data

It is in the data processing modules where the data is computed to get information from it. These modules are divided in our system into 3 main modules:

- Activity analysis - In this phase, the already validated input file is received and will be analyzed for all the events it contains. The user must provide a list with the frequencies for the file analysis, i.e., a list with the time interval that he wants to group the events. We use the *Pandas* library to read the file and create a dataframe, where we can apply grouping functions according to the timestamp of the file. After obtaining the data, it is sent to a function that will compress and store it in the DB.

- Features extraction - This module is responsible for extracting features from the input file. It takes as input parameters: the file id, the time window, the feature extraction method, and the list of ports for which it should extract features. The first step is to check if features have already been extracted for these input parameters. If they already exist, this step is skipped, this way, no repeated data is saved, and no unnecessary processing of the same data occurs.

  Feature extraction methods can be divided into two categories, those that extract ports dynamically and those that extract ports statically. In the first case, we have Dyn3_x that uses 1/3 of the most used and least used ports. In this method, it is necessary to provide the number of ports that we want to analyze (by default there are 100 ports). On the other hand, we have feature extraction methods that use predefined port lists, such as the OutGene method (default list of ports: 22, 25, 80, 194) and FlowHacker which uses the same list as OutGene adding port 6667. We also have Dyn3_x + OutGene that uses the ports dynamically obtained by Dyn3_x and statically obtained by OutGene. The last method is based on a list of ports that the user can provide for feature extraction

  The features generated for each IP address can be divided in two: 1) static - will always be present, as source and with destination, presented in Table 8; 2) dynamic - a depend on the analysed traffic, can be for example: Src20To (# of different flows from source port 20 used by an entity).

  After extracting the features, they are saved as shown in Section 4.4.1. To make this processing faster, the pipeline was used, implementing a parallelization function from the *Joblib* library.

Table 8: Description of fixed features.

Column name	Description
IPContacted	# of different IPs contacted by an entity
PortUsed	# of different ports used by an entity
PortContacted	# of different ports contacted by an entity
TotLenRcv	Sum of total packets length received by an entity
TotLenSent	Sum of total packets length sent by an entity
TotConn	# of flows of the entity
PktRate	# of packets per second
AvgPktSize	Average packet size
Pkt	# of packets

- Clustering - This is the main module for the production of results, where the various IPs are grouped using clustering algorithms. This way the outliers can be identified, these can be victims or attackers. The outliers will correspond to the clusters that have only one IP address in them.

  Regarding clustering, we have parameters such as *start_time* and *end_time* - will set the limits for which the clusters should be produced; the *file_id* of the file that is associated and the configuration that was used for extracting the features, this way, it is possible to access the already extracted features that are stored on the server; *time_window* used for clustering; *ip_view* that corresponds to the network prefix where the IP is inserted and finally parameters like k that will set the number of clusters, if k is -1 the elbow method will be used to determine the optimal number of clusters to use. The elbow method will be limited by a value of *k_min* and a value of *k_max* that must also be supplied by the user (by default $k\_min = 3$ and $k\_max = 20$).

  Like feature extraction, clustering will also avoid redundant data processing and uses the *Joblib* library for multithreaded processing.

  After obtaining the features, K-Means, available in the sklearn library, is used for clustering. If the elbow method is used (by default), the value of the number of clusters is iterated between *k_min* and *k_max* and the value of k for the smallest distortion of the data is obtained. The remaining parameters of the K-Means function are fixed and are: max_iter[13] = 1000; random_state[14]=0; tol[15] = 1e-5.

  After this phase, the clustering data is stored in the DB, as shown in Table 7.

[13]Maximum number of iterations of the k-means algorithm in a single run, from [55].

[14]Determines random number generation for centroid initialization, from [55].

[15]Relative tolerance with regards to Euclidean norm of the difference in the cluster centers of two consecutive iterations to declare convergence, from [55].

4.4.3 REST API

We will present and describe all the endpoints that are part of our system, these are divided by the resources that the IP provides. In the development phase, we used localhost[16] on port 5000 to run the API, so we will present the endpoints where the Uniform Resource Locator (URL) is prefixed with {DevEnvironment}. However, it should be noted that these endpoints can be implemented on another type of server (different from localhost) and on another port.

The URL of the request, its description, what data the request should have, and what data comes from response, in JSON format, are presented.

- **Users**
  - **GET** {DevEnvironment}/api/users

    Description: This request returns the user id.

    Body Keys: { *username* }

    Example of Body (JSON):

```
{
 "username": "username"
}
```

    Example of response (JSON):

```
{
 "user_id": 1
}
```

  - **POST** {DevEnvironment}/api/users

    Description: This request inserts a new user into the system.

    Body Keys: { *username* }

    Response: 200 - OK or 400 - User already exists

- **Files**
  - **GET** {DevEnvironment}/api/files?user=x

    Description: This request returns all file names for a given user that are stored on the server.
  - **POST** {DevEnvironment}/api/files?user=x

    Description: This request is to upload a file into the system.

[16] Refers to the local computer on which a program is running, the IP address 127.0.0.1 can also be used to describe localhost, adapted from https://techterms.com/definition/localhost.

- **GET** {DevEnvironment}/api/files/<file_name>

  Description: This request returns the start date and the end date of the chosen file.

- **DELETE** {DevEnvironment}/api/files/<file_name>

  Description: This request has the function of deleting a file and all processed data related to it.

- **GET** {DevEnvironment}/api/files/<file_name>/file_activity

  Description: This request returns the date and the number of events that are in the selected file.

- **GET** {DevEnvironment}/api/files/<file_name>/conf_time_windows

  Description: This request returns a list with the time windows that have been analyzed in the file.

- **Features**

  - **POST** {DevEnvironment}/api/features/extract

    Description: This request is to extract the features from the selected file with the chosen parameters.

  - **GET** {DevEnvironment}/api/features/results

    Description: This request returns the id of the feature extraction configuration for the given parameters that have been passed in.

- **Clusters**

  - **POST** {DevEnvironment}/api/clusters/fast

    Description: This request will process the features to create the clusters. It uses the parameter k to set the number of clusters, if this parameter is -1 it uses the elbow method to find the best number for K between the range [k_min, k_max].

  - **GET** {DevEnvironment}/api/clusters/results?file_name=x&result_id=x

    Description: This request returns information about clusters that exist with fewer than N (n_max) elements, for a file analysis.

  - **GET** {DevEnvironment}/api/clusters/results/all?file_name=x&result_id=x&timestamp=x

    Description: This request returns which clusters all IPs belong to, taking into account the file, the analysis performed, and the timestamp.

  - **GET** {DevEnvironment}/api/clusters/heatmap?file_name=x

    Description: This request returns the values needed to create the heatmap.

- **Metrics**

  - **GET** {DevEnvironment}/api/metrics?file_name=x&result_id

Description: This request returns all data concerning the metrics for evaluating the performance of the ML algorithms. It is necessary to provide a dictionary containing the IPs known as attackers or victims and how long they have been active.

For a more detailed analysis of the HTTP requests described above, examples of the body format, headers and responses of HTTP requests are presented in Appendix A.

4.5 Summary

In this chapter, the implemented tool was presented, the approach and technologies used in the development of the visualisation tool were described. The functionalities of the tool were described and how the user should interact with the GUI was explained. The main modules, functions, and structure that are the basis of the back-end operation were presented.

5 Evaluation of the Tool

This chapter presents the evaluation of the work developed. An assessment is based on the theoretical concepts of systems and software evaluation presented in ISO/IEC 25010:2011. Another evaluation refers to the comparison of the results obtained through the tool and those described in the original articles of the implemented algorithms. This chapter also describes the dataset and the evaluation metrics used to obtain the results.

5.1 Systems and Software Quality Requirements and Evaluation

The evaluation of this tool is based on ISO 25010:2011 - Systems and software Quality Requirements and Evaluation (SQuaRE) [56], following the evaluation process model referred to in this standard.

The first step is to know what characteristics and requirements the application must have (presented in Section 4.1.1) to be evaluated. It is also necessary to verify that all modules in the architecture (Section 4.1.2) meet the objectives for which they were designed.

For the evaluation of our system, we consider the GUI of the visualisation tool and the back-end that was developed. We divided the evaluation parameters into two groups: Key Evaluation Parameters (Section 5.1.1) and Complementary Evaluation Parameters (Section 5.1.2). In these, various evaluation aspects are described and an analysis of our system is made.

5.1.1 Key Evaluation Parameters

These are the parameters that we consider fundamental for our system to achieve its objectives:

- Functional Suitability - This aspect concerns the achievement of the objectives by the functions of the tool. It evaluates if the results presented are in accordance with what was obtained by the various algorithms.

  The tool meets its main objective, which is to detect anomalous behaviour and classify machines as potential threats or victims. Our tool allows the user to choose several different methods of analysis. It also offers the ability to check network activity over several time periods.

- Compatibility - This refers to the ability of the application to function correctly regardless of the hardware or software used. This aspect also considers interoperability and the degree to which two or more systems, products, or components can exchange information and use the information that has been exchanged.

  The GUI of the tool is compatible with both Windows and Linux systems. For Windows, an .exe file is provided that allows the execution of the tool. For the Linux environment, two .sh files are provided, one that installs Python and all necessary libraries, and the second responsible for running the tool.

Regarding the input data format, they are limited to the .csv format and must have the columns presented above in Table 3. However, this format is simple and easy for the user to input data.

In the back-end, a Docker container image is provided that packages the code and all its dependencies so that the application works regardless of the OS used. Using a REST API makes it possible for other platforms and researchers to use our data in a compatible way through HTTP requests.

- Usability - Degree of effectiveness, efficiency, and satisfaction obtained by users when using the application. The difficulty of use, ability to configure and operate, protection against user errors, and whether the outputs are shown are important and properly organized should be considered.

  The GUI of the tool is made in a simple way, containing only the necessary functions for efficient data analysis. The results obtained are organized and presented in tables and graphs. The GUI has a help button that allows the user to clarify doubts about the use of the tool, protecting the system against user errors.

  The user has a navigation menu at the top of the GUI, where it is possible to configure the visual aspect of the results (e.g., the logarithmic scale of the activity graph) and of their presentation (e.g., change the colour theme of the heatmap). Some parameters (e.g., min and max values of the elbow method) for the analysis of the results can also be changed.

- Security - How the data (used and stored) and the whole system are protected and the system's reliability.

  The system is ready to use HTTPS requests in order to obtain more security during communication. However in the development phase, since the server was running on localhost, it was decided to use HTTP requests to optimise and facilitate this phase. In the production phase, user authentication and the use of tookens in requests should also be implemented. The security of stored data is directly related to the server's security where the back-end is running.

  Regarding the reliability of the system, in the graphical interface folder, there is a configuration file that must not be corrupted, otherwise, the GUI may stop working.

5.1.2 Complementary Evaluation Parameters

- Performance Efficiency - This point will consider the efficiency of algorithm execution in terms of processing time and how the memory in the system is used.

  The algorithms integrated into the tool already used multiprocessing, using the parallel computing provided by the *joblibs* library. However, the processing done by the server can still be improved with the use of distributed systems and load balancers.

  In the back-end was implemented a reading of files by parts (chunks) for better use of memory. The choice of sqlite DB did not have the desired performance. When there were many saved data, and using

complex queries to acquire the data took some time. It is expected that the system performance will improve with the use of NoSQL DB or another type of SQL DB. Regarding the storage of features in files on the server, the performance has improved after changing the file format from CSV to *parquet*. The time to get the data decreased, and the space used decreased as well.

- Maintainability - How effectively a system can be modified, the modules should follow the open-close principle, and if system maintenance is done in a simple way.

  The tool is divided into several modules: database, algorithms, and API. Our system requires maintenance to manage the files stored in folders on the server for efficient use of storage space, i.e., files that are not used for a long time should be deleted.

- Portability - Efficiency with which a system and its resources have to be transferred to other hardware (e.g., physical servers), software or used in another environment.

  The Front-end is easily exportable to Windows and Linux environments through a zip file. As already mentioned in the Compatibility point, there is a docker container that allows the back-end to be easily integrated into a physical server.

- Reliability - How the system behaves to execute specified functions under specified conditions in a given period of time. How to react if, for example: a fault occurs, incorrect data is provided, or some function stops or loops.

  The system does not enter infinite loops, however, if there is an error in the processing, the API stops, and it is necessary to restart it. Regarding the incorrect data in the input files, if these do not comply with the formatting (i.e., have the necessary columns), they will not be loaded. If the tool gets incorrect or undefined data (e.g., undefined values), it will be loaded normally, but the results obtained may be incorrect. Since it is a prototype, these reliability aspects are considered acceptable.

5.2 Experimental Evaluation

Regarding the evaluation and validation of intrusion detection algorithms, only the IP addresses belonging to the network under analysis were considered, i.e., only the IPs with the prefix 172.31.0.0/16 were analysed since it is crucial to understand which machines on the network are suffering or carrying out cyber attacks.

It should be noted that DynIDS was not implemented with the 3 clustering algorithms, only K-Means was implemented. Thus, the results obtained based on this detection algorithm will not fully match those described in the DynIDS paper [14].

All development and experiences were done in commodity hardware (AMD Ryzen 7 4800H 2,9 GHz with 16GB RAM).

5.2.1 Dataset Used

We use the CSE-CIC-IDS2018 dataset [57], which was created to test and evaluate network intrusion detection algorithms. This dataset is structured to represent the network of a medium enterprise, with six subnets deployed on the Amazon Web Services (AWS) cloud computing platform. Its authors have created user profiles with abstract representations of the activity seen on the network. There are profiles responsible for generating benign behaviour (B-Profiles) and others for generating malicious behaviour (M-Profiles).

We consider all the attack scenarios provided by the dataset, and these are: brute force attack (to FTP and SSH); DoS and DDoS attacks; web attacks; infiltration attacks, port scan and botnet attack. The start and end date of the attacks, their type and the IP address of the victim are shown in Appendix B

After analysing the dataset with our tool, a few hosts were found that were not considered as victims in the information provided in CSE-CIC-IDS2018, but had suspicious behaviour, these are:

- Day 1 - large amount of traffic from 212.92.116.6 to 172.31.66.82, for a specific port (3389).
- Day 7 - several ports contacted sequentially (port scan) at IP address 172.31.67.62.
- Day 9 - several ports contacted sequentially (port scan) at IP address 172.31.65.77.

Although appearing in some time windows, these hosts were considered FPs since the authors of CSE-CIC-IDS2018 do not confirm them. Examples of netflow data are given in Appendix D and E.

5.2.2 Evaluation Metrics

To analyse the performance of the algorithms, we consider the hosts detected as outliers (i.e., those that are isolated in a cluster), their IP address identifies the hosts. According to the data set used, we considered: TP the hosts that are correctly classified; TN the hosts that are not marked as threats (outliers); FN the hosts that should have been classified as outliers and FP the hosts that were classified as outliers but were not threats.

For a better understanding of how the results were classified, taking into account their duration, an example is shown based on Figure 20. Example: If the attack is not detected in time windows 2, 3, or 4, it is considered an FN for each window. The algorithm must detect the attack in all three windows to consider it a TP for each window. If it only detects it in time window 3, we consider 1 TP and 2 FN.

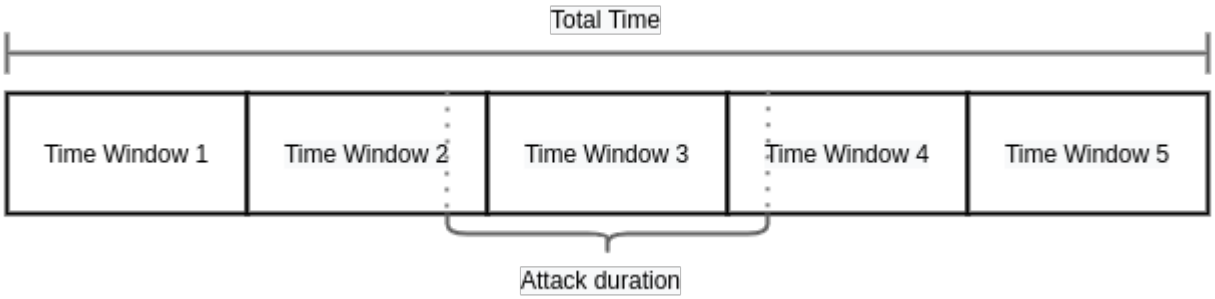

Figure 20: Division of time windows in the analysis of the results.

Below are presented the metrics used and the respective formulas for their calculation. It should be noted that the values of all metrics, given by the equations 5, 6, 7 and 8, are comprised between 0 and 1. However, these values can also be presented between 0 and 100.

- Precision - the fraction of outliers that are real.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

- Recall - the fraction of outliers that are correctly classified by the algorithms.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

- F_1-Score - global evaluation metric of the algorithm performance.

$$F_1 Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

- Matthews Correlation Coefficient (MCC) - used for the quality assessment of binary classifications.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{8}$$

Since the dataset used is not balanced, there are more hosts that are not a threat than those that are, so we chose not to choose the accuracy. Since the number of TNs is quite high compared to TPs, the accuracy values are close to 100%, making this metric of no interest for algorithm classification. The metrics that we privileged for the classification were the F_1-Score and the MCC because they summarise better the performance of the algorithms.

5.2.3 Algorithms Results

The Figures 21, 22, 23 and 24 show the results of the detection algorithms relative to the evaluation metrics described in Section 5.2.1. These were obtained by analysing the CSE-CIC-IDS2018 dataset in several time

windows. The values presented in the graphs were calculated relative to the total values of TPs, TNs, FNs and FPs of the whole dataset (10 days).

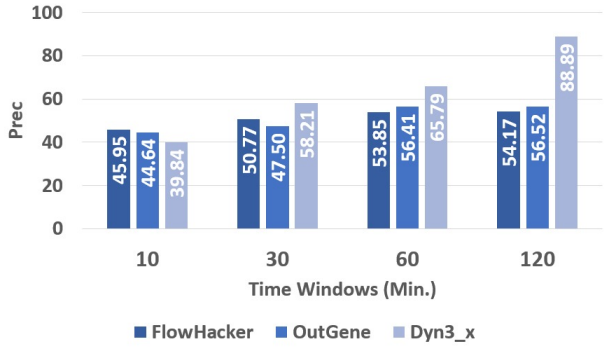

Figure 21: Precision of the 3 algorithms for the entire dataset.

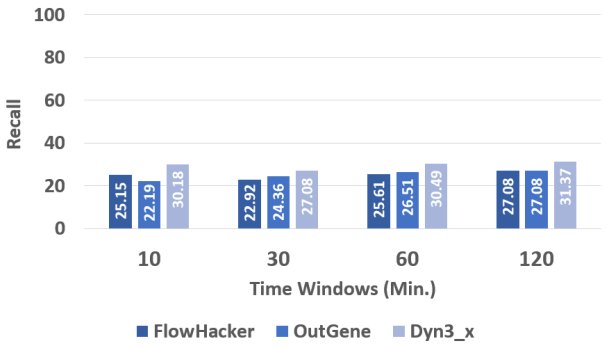

Figure 22: Recall of the 3 algorithms for the entire dataset.

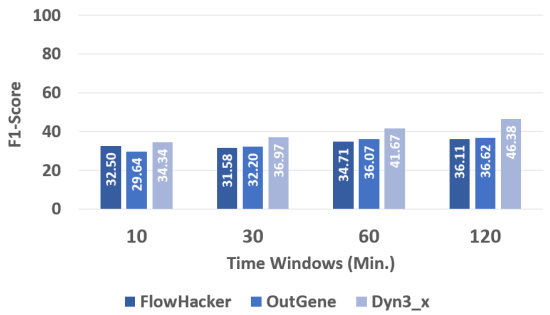

Figure 23: F_1-Score comparison in FlowHacker, OutGene and DynIDS for the entire dataset.

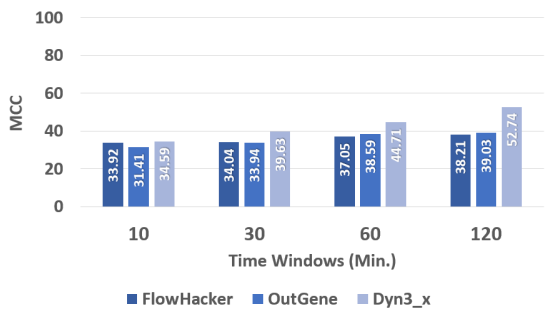

Figure 24: MCC comparison in FlowHacker, OutGene and DynIDS for the entire dataset.

We noticed, after the performed experiments, that the implemented algorithms did not perform well in detecting botnet attacks, failing to detect any victim in the various time windows. In order to demonstrate the satisfactory operation of the algorithms for the detection of the remaining attacks, the same metrics are presented in Figures 25, 26, 27 and 28 but not counting day 10 of the dataset.

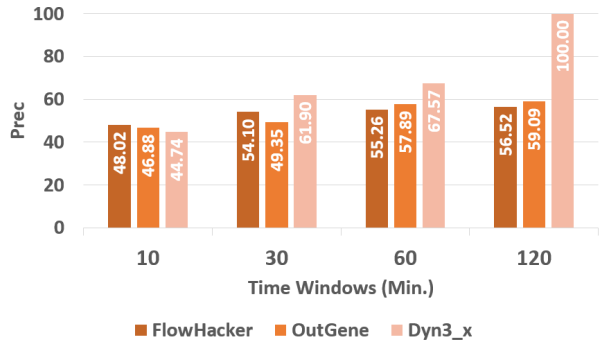

Figure 25: Precision of the 3 algorithms for the dataset without day 10.

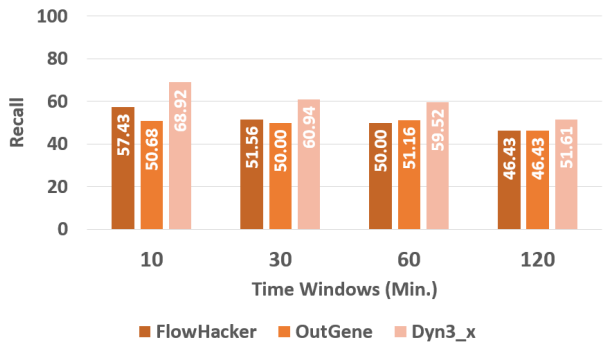

Figure 26: Recall of the 3 algorithms for the dataset without day 10.

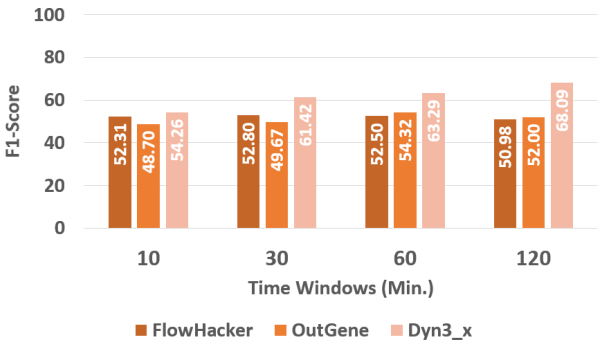

Figure 27: F_1-Score of the 3 algorithms for the dataset without day 10.

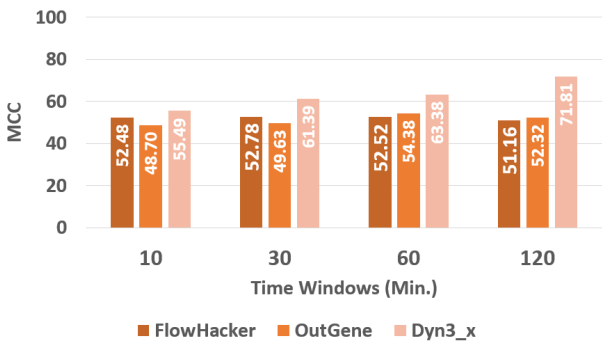

Figure 28: MCC of the 3 algorithms for the dataset without day 10.

Based on the Figures 23, 24, 27 and 28, we conclude that DynIDS (Dyn3_x) is the algorithm that obtains the best results, regardless of the use of day 10 in the analysis. This algorithm has better values of Precision and Recall, which in turn makes the F_1-Score higher than the other algorithms. In the MCC, it is also possible to verify the superiority of DynIDS. It should be noted that for the 10 minutes time windows, DynIDS obtains lower performance values than the others. This fact is due to the way this algorithm obtains the ports and the number of ports used. In small time windows, the DynIDS approach, as it considers more information than the other two algorithms, can sometimes generate FPs. As expected, the results of FlowHacker and OutGene are similar since they use a similar approach, differing in the list of ports analysed.

We saw that the results improved significantly without day 10 as the algorithms could not detect the botnet attacks. These generated many FNs values for the time windows (10 min. $\rightarrow$ 190; 30 min. $\rightarrow$ 80; 60 min. $\rightarrow$ 40 and 120 min. $\rightarrow$ 20). The algorithms, on day 10 also detected other hosts that were not described in the dataset, thus existing some FPs.

It should be noted for the graph in Figure 25, that in the 120 minutes time window, DynIDS is not perfect. The 100 % precision only means that no FP was detected in this time window.

5.2.4 Results Validation

In this subsection, the validation of the results obtained by the tool is presented. It is intended to compare the original metrics values of the implemented detection algorithms with the values obtained by these but integrated into the tool. For that purpose, the results are obtained using a different approach from the one presented in Section 5.2.2. In the approach used to calculate the metrics of the implemented algorithms [12] [13] [14], the victims must be detected in at least one of the time windows to be considered a TP. The start and end time of the attacks in the dataset was taken into account for the computation of the TPs, TNs, FNs and FPs. The metric values presented were calculated without day 10 of the dataset.

Something that comes immediately to mind in the figures is that some results appear to be bad. For instance, precisions of 0.25 to 0.27 are quite bad (Figure 29). However, this is not an issue as the point is that we run the algorithms in several time windows, following OutGene's time stretching approach, in order to detect the malicious traffic in *any* of them, not in *all* of them.

Figures 29, 30 and 31 show the Precision, Recall and F_1-Score of the detection algorithms implemented in our tool. Although the MCC is not presented in the articles of the algorithms, we decided to present it in Figure 32 because it is a reference indicator of the efficiency of the algorithms. In the general case, it can be verified that the values of F_1-Score and MCC increase with the increase of the size of the time window. The algorithms become more accurate with the increase of the time window, according to the prediction presented in the DynIDS paper [14].

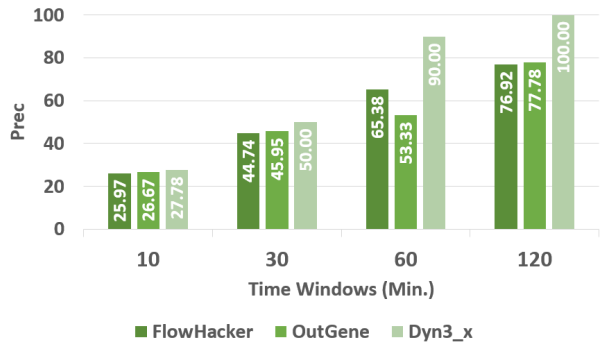

Figure 29: Precision of the 3 algorithms, with the original approach.

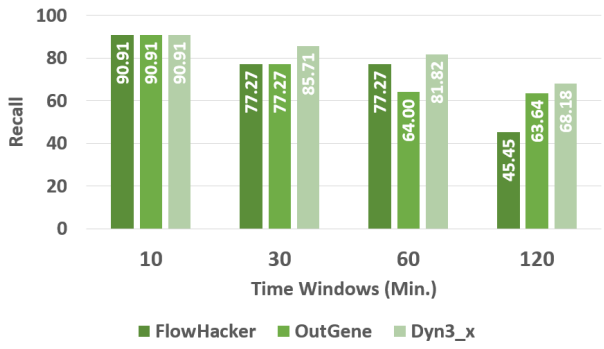

Figure 30: Recall of the 3 algorithms, with the original approach.

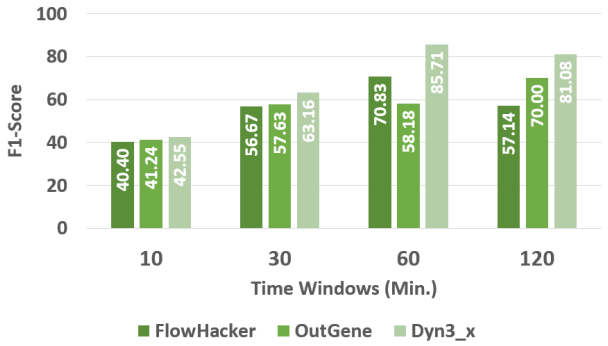

Figure 31: F_1-Score of the 3 algorithms, with the original approach.

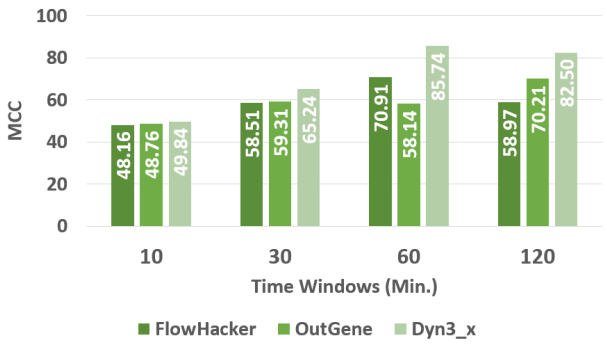

Figure 32: MCC of the 3 algorithms, with the original approach.

Table 9 presents the comparison between the original F_1-Score values of the 3 implemented algorithms and the F_1-Score values obtained through the tool for the 10 and 60 minute time windows. The original F_1-Score values of the 3 algorithms were taken from the DynIDS article [14], where the comparison between them is presented. In the case of Dyn3_100 the original values refer to the use of K-Means and not the values when the correlation between the 3 clustering algorithms is applied.

Table 9: Comparison of F_1-Score results.

	FlowHacker		OutGene		Dyn3_100	
Time window	10 min	60 min	10 min	60 min	10 min	60 min
DynIDS paper	0.38	0.60	0.52	0.61	0.72	0.92
Tool values	0.40	0.71	0.41	0.58	0.43	0.86
Difference	+ 0.02	+ 0.11	- 0.11	- 0.03	- 0.29	- 0.06

Since the K-means clustering algorithm was used in the tool, it is not deterministic, i.e., it does not necessarily always give the same values. This is due to the fact that K-means depends on the initial point, generated randomly, to start the clustering. In this way, the values obtained in the tool and the original values do not need to be strictly equal.

With the approach used for the evaluation, there are 20 victims that must be identified. This means that the appearance of an FP will have a significant weight on the precision value, which in turn affects the F_1-Score.

The positive difference of F_1-Score from FlowHacker is because this algorithm was implemented with the OutGene approach, these differing only in the list of ports used, one more port for FlowHacker.

An immediately visible value is the difference of - 0.29 in the 10 minutes time window in DYN3_100. This value comes from the amount of FPs counted and not from the FNs, since it is the precision (0.28) that is affected and not the Recall (0.91). Since the analysed ports are obtained dynamically and the time

window is relatively small, some machines with traffic peaks in those 10 minutes are considered threats. Even analyzing the logs of the input files, it is not possible to say with certainty that that traffic is malicious, so we prefer to count them as FPs. The same is true for OutGene in 10 minute time window.

5.3 Summary

In this chapter, the theoretical evaluation parameters of our tool based on the SQuaRE ISO norm were presented. The CSE-CIC-IDS2018 dataset and the metrics used to evaluate the intrusion detection algorithms implemented in the tool were described. The results obtained and the validation of the results were also presented in this chapter.

6 Conclusions and Future Work

This thesis presents a visualisation tool for automatic detection of threats in cyberspace, based on a client-server architecture and containing an interactive and simple GUI. In the back-end are the algorithms based on unsupervised ML, responsible for obtaining the data for the GUI, and REST API, that allows the communication data between the back-end and the front-end through HTTP/HTTPS requests. The GUI of the tool has buttons, lists, graphs and menus, allowing its use by any user without previous knowledge. The user can easily change the hyperparameters of the algorithms, obtaining in this way personalised results for different types of analysis.

The evaluation of the developed tool was performed in two parts. The first was based on an analysis of the theoretical parameters of systems and software quality, presented in the standard ISO 25010:2011 - Systems and software Quality Requirements and Evaluation. The second evaluation consisted in testing the tool for a specific dataset (CSE-CIC-IDS2018). Where the results obtained were compared with the results described in the papers of the algorithms that the tool integrates.

The graphical environment provided in the tool makes it simple to use. Following an intuitive sequential order allows the user to know what to do in the various phases. It also provides a greater ability to customise the hyperparameters of the algorithms so that different types of analysis can be performed. We can thus state that the tool is more effective than running the algorithms through the terminal.

In terms of improvements to our tool and optimising the speed of access and data processing, we propose the alteration of the DB, or its tables, and the implementation of the back-end in a distributed system or a physical server with greater processing capacity.

For future work, we suggest implementing more useful features for the use of the graphical environment, which concerns the visual aspect and the aspect of analysis and obtaining data for the user. One of the aspects that can be improved is the inclusion of more clustering algorithms and the correlation of the values obtained by these algorithms, thus improving performance. Another is the implementation of the history of clusters of the network machines, as proposed in C2BID. It could also be interesting, since the API is already developed, to create a Web application based on our REST API.

References

[1] S. Hakak, W. Z. Khan, M. Imran, K.-K. R. Choo, and M. Shoaib, "Have you been a victim of covid-19-related cyber incidents? survey, taxonomy, and mitigation strategies," *IEEE Access*, vol. 8, pp. 124 134–124 144, 2020.

[2] P. Bowen, J. Hash, and M. Wilson, "Information security handbook: a guide for managers," in *NIST special publication 800-100, National Institute of Standards and Technology*, 2007.

[3] Eurostat, "ICT security measures taken by vast majority of enterprises in the EU," 13-Jan-2020, [Accessed: 22-Nov-2020]. [Online]. Available: https://ec.europa.eu/eurostat/documents/2995521/10335060/9-13012020-BP-EN.pdf/f1060f2b-b141-b250-7f51-85c9704a5a5f?t=1578907924000

[4] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[5] A. Gazet, "Comparative analysis of various ransomware virii," *Journal in Computer Virology*, vol. 6, no. 1, pp. 77–90, 2010.

[6] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[7] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM, 2003, pp. 25–36.

[8] E. Min, J. Long, Q. Liu, J. Cui, Z. Cai, and J. Ma, "Su-ids: A semi-supervised and unsupervised framework for network intrusion detection," in *International Conference on Cloud Computing and Security*. Springer, 2018, pp. 322–334.

[9] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[10] D. Staheli, T. Yu, R. J. Crouser, S. Damodaran, K. Nam, D. O'Gwynn, S. McKenna, and L. Harrison, "Visualization evaluation for cyber security: Trends and future directions," in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, 2014, pp. 49–56.

[11] H. de Bruijn and M. Janssen, "Building cybersecurity awareness: The need for evidence-based framing strategies," *Government Information Quarterly*, vol. 34, no. 1, pp. 1–7, 2017.

[12] L. Sacramento, I. Medeiros, J. Bota, and M. Correia, "Flowhacker: Detecting unknown network attacks in big traffic data using network flows," in *17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2018, pp. 567–572.

[13] L. Dias, H. Reia, R. Neves, and M. Correia, "Outgene: Detecting undefined network attacks with time stretching and genetic zooms," in *International Conference on Network and System Security*. Springer, 2019, pp. 199–220.

[14] L. Dias, S. Valente, and M. Correia, "Go With the Flow: Clustering Dynamically-Defined NetFlow Features for Network Intrusion Detection with DynIDS," in *19th IEEE International Symposium on Network Computing and Applications (NCA)*, 2020.

[15] G. Bonaccorso, *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[16] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2018.

[17] J. Lehr, J. Philipps, V. N. Hoang, D. von Wrangel, and J. Krüger, "Supervised learning vs. unsupervised learning: A comparison for optical inspection applications in quality control," in *IOP Conference Series: Materials Science and Engineering*, vol. 1140, no. 1, 2021.

[18] Rui Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[19] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *Proceedings of the 18th International Conference on Machine Learning*, vol. 1, 2001, pp. 577–584.

[20] C.-H. Lung and C. Zhou, "Using hierarchical agglomerative clustering in wireless sensor networks: An energy-efficient and flexible approach," *Ad Hoc Networks*, vol. 8, no. 3, pp. 328–344, 2010.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, vol. 96, no. 34, 1996, pp. 226–231.

[22] R. G. Bace and P. Mell, "Intrusion detection systems," National Institute of Standards and Technology, 2001.

[23] X. Zhang, C. Li, and W. Zheng, "Intrusion prevention system design," in *The Fourth International Conference on Computer and Information Technology*. IEEE, 2004, pp. 386–390.

[24] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: a review," *IEEE Access*, vol. 6, pp. 10 179–10 188, 2018.

[25] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 255–264.

[26] Y.-S. Wu, S. Bagchi, S. Garg, and N. Singh, "Scidive: A stateful and cross protocol intrusion detection architecture for voice-over-ip environments," in *International Conference on Dependable Systems and Networks, 2004.* IEEE, 2004, pp. 433–442.

[27] J. Jacobs and B. Rudis, *Data-driven security: analysis, visualization and dashboards.* John Wiley & Sons, 2014.

[28] I. Williams and X. Yuan, "Evaluating the effectiveness of microsoft threat modeling tool," in *Proceedings of the 2015 information security curriculum development conference*, 2015, pp. 1–6.

[29] Microsoft, "Microsoft Threat Modeling Tool," 16-Feb-2020, [Accessed: 28-Oct-2020]. [Online]. Available: https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

[30] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning.* Insecure, 2009.

[31] I. Kotenko and A. Chechulin, "A cyber attack modeling and impact assessment framework," in *2013 5th International Conference on Cyber Conflict (CYCON 2013)*, 2013, pp. 1–24.

[32] N. Kheir, N. Cuppens-Boulahia, F. Cuppens, and H. Debar, "A service dependency model for cost-sensitive intrusion response," in *European Symposium on Research in Computer Security.* Springer, 2010, pp. 626–642.

[33] Y.-a. Kang, C. Gorg, and J. Stasko, "Evaluating visual analytics systems for investigative analysis: Deriving design principles from a case study," in *2009 IEEE Symposium on Visual Analytics Science and Technology.* IEEE, 2009, pp. 139–146.

[34] S. Musman and A. Temin, "A cyber mission impact assessment tool," in *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2015, pp. 1–7.

[35] S. Noel, E. Harley, K. H. Tam, M. Limiero, and M. Share, "Cygraph: graph-based analytics and visualization for cybersecurity," in *Handbook of Statistics.* Elsevier, 2016, vol. 35, pp. 117–167.

[36] I. Kotenko and E. Novikova, "Vissecanalyzer: A visual analytics tool for network security assessment," in *International Conference on Availability, Reliability, and Security.* Springer, 2013, pp. 345–360.

[37] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "Evaluation of anomaly detection algorithms made easy with reload," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019, pp. 446–455.

[38] G. Vernik, A. Shulman-Peleg, S. Dippl, C. Formisano, M. C. Jaeger, E. K. Kolodner, and M. Villari, "Data on-boarding in federated storage clouds," in *2013 IEEE Sixth International Conference on Cloud Computing*, 2013, pp. 244–251.

[39] F. Zalila, X. Crégut, and M. Pantel, "Formal verification integration approach for dsml," in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2013, pp. 336–351.

[40] A. Marcus, "Principles of effective visual communication for graphical user interface design," in *Readings in human–computer interaction*. Elsevier, 1995, pp. 425–441.

[41] L. Dias, T. Fernandes, and M. Correia, "C2BID: Cluster Change-Based Intrusion Detection," in *Proceedings of Trustcom*, 2020.

[42] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *International conference on machine learning*, 2016, pp. 2712–2721.

[43] W. McKinney and the Pandas Development Team, "pandas: powerful python data analysis toolkit," 25-Jul-2021, [Accessed: 3-Ago-2021]. [Online]. Available: https://pandas.pydata.org/docs/pandas.pdf

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[45] M. Lutz, *Programming python*. " O'Reilly Media, Inc.", 2001.

[46] Matt Makai, "Full Stack Python," Oct-2020, [Accessed: 18-Nov-2020]. [Online]. Available: https://www.fullstackpython.com/flask.html

[47] Python, "WebFrameworks," Nov-2020, [Accessed: 20-Nov-2020]. [Online]. Available: https://wiki.python.org/moin/WebFrameworks

[48] M. Owens and G. Allen, *SQLite*. Springer, 2010.

[49] R. Copeland, *Essential sqlalchemy*. " O'Reilly Media, Inc.", 2008.

[50] Udacity, "What is pythonic style?" [Accessed: 10-Ago-2021]. [Online]. Available: https://www.udacity.com/blog/2020/09/what-is-pythonic-style.html

[51] D. Vohra, "Apache parquet," in *Practical Hadoop Ecosystem*. Springer, 2016, pp. 325–335.

[52] A. S. Foundation, "Apache parquet," [Accessed: 12-Ago-2021]. [Online]. Available: https://parquet.apache.org/documentation/latest/

[53] F. Lundh, "An introduction to tkinter," *URL: www. pythonware. com/library/tkinter/introduction/index. htm*, 1999.

[54] S. Tosi, *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[55] Scikit-learn.org, "sklearn.cluster.kmeans," [Accessed: 12-Ago-2021]. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[56] J. Estdale and E. Georgiadou, "Applying the iso/iec 25010 quality models to software product," in *European Conference on Software Process Improvement*. Springer, 2018, pp. 492–503.

[57] Canadian Institute for Cybersecurity, "Cisco IOS NetFlow," [Accessed: 22-Set-2021]. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[58] R. V. Rao and K. Selvamani, "Data security challenges and its solutions in cloud computing," *Procedia Computer Science*, vol. 48, pp. 204–209, 2015.

A Examples of HTTP requests to the API

- **Users**
  - **GET** {DevEnvironment}/api/users

    Body Keys: { *username* }

    Example of Body (JSON):

```
{
    "username": "username"
}
```

    Example of response (JSON):

```
{
    "user_id": 1
}
```

  - **POST** {DevEnvironment}/api/users

    Body Keys: { *username* }

    Response: 200 - OK or 400 - User already exists

- **Files**
  - **GET** {DevEnvironment}/api/files?user=x

    Query parameters: user = user_id

    Example of response (JSON):

```
{
    "Files": [
        "file_1.csv",
        "file_2.csv",
        "file_3.csv"
    ]
}
```

  - **POST** {DevEnvironment}/api/files?user=x

    Query parameters: user = user_id

    Body Keys: { *freq, date_format, default_method, endpoint, default_tw* }

    File input format:

```
files=[('file',('file_1.csv',open('path/file_1.csv','rb'),'text/csv'))]
```

Example of Body (JSON):

```
{
    "freq": "30 min, 1D",
    "date_format": "%d/%m/%Y %I:%M:%S %p",
    "default_method": 0,
    "default_tw": "30, 60"
}
```

Response: 200 - OK or 400 - Bad Request

– **GET** {DevEnvironment}/api/files/<file_name>

Body Keys: { *user* }

Headers:

```
headers = { 'Content-Type': 'application/json'}
```

Example of response (JSON):

```
{
    "File Info": [
        "2018-02-14 12:28:07",
        "2018-02-15 00:47:34"
    ]
}
```

– **DELETE** {DevEnvironment}/api/files/<file_name>

Body Keys: { *user* }

Example of Body (JSON):

```
{
    "user": 0
}
```

Response: 200 - OK or 400 - Bad Request

– **GET** {DevEnvironment}/api/files/<file_name>/file_activity

Body Keys: { *user, time* }

Example of Body (JSON):

```
{
    "user": 0,
    "time": "30 min"
}
```

Headers:

```
headers = { 'Content-Type': 'application/json'}
```

Example of response (JSON):

```
{
    "File_activity": {
        "times": [
            "2018-02-14 12:00:00",
            "2018-02-14 12:30:00",
            ...
        ],
        "values": [
            19983,
            454313,
            ...
        ]
    }
}
```

– **GET** {DevEnvironment}/api/files/<file_name>/conf_time_windows

Body Keys: { *user* }

Headers:

```
headers = { 'Content-Type': 'application/json'}
```

Example of response (JSON):

```
{
    "TimeWindows": [
        30,
        60,
        120
    ]
```

```
}
```

- **Features**
  - **POST** {DevEnvironment}/api/features/extract

    Body Keys: { *file_name, user, TimeWindows, method, nr_ports, start_date, end_date, DATE_FORMAT, IP_TO_MATCH, PORT_DAY, OUTGENE_PORTS, FLOWHACKER_PORTS* }

    Example of Body (JSON):

```
{
    "file_name":"file_1.csv",
    "user": 0,
    "TimeWindows": [120],
    "method": 0,
    "nr_ports": 100,
    "start_date": "14/02/2018 00:00:00",
    "end_date": "14/02/2018 23:59:59",
    "DATE_FORMAT": "%d/%m/%Y %I:%M:%S %p",
    "IP_TO_MATCH": "172.31.",
    "PORT_DAY":[],
    "OUTGENE_PORTS": [80, 194, 25, 22],
    "FLOWHACKER_PORTS": [80, 194, 25, 22, 6667]
}
```

    Headers:

```
headers = { 'Content-Type': 'application/json'}
```

    Response: 200 - OK or 400 - Bad Request

  - **GET** {DevEnvironment}/api/features/results

    Body Keys: { *file_name, user, tw, method, nr_ports, ip_to_match, ports_lst, std_ports* }

    Example of Body (JSON):

```
{
    "file_name": "file_1.csv",
    "user": 0,
    "tw": 30,
    "method": 3 ,
    "nr_ports": 100,
```

```
    "ip_to_match": "172.31.",
    "ports_lst": [[],[80, 194, 25, 22], [80, 194, 25, 22, 6667]],
    "std_ports": 1
}
```

Headers:

```
headers = { 'Content-Type': 'application/json'}
```

Example of response (JSON):

```
{
    "Config_id": 1
}
```

- **Clusters**

- **POST** {DevEnvironment}/api/clusters/fast

  Body Keys: { *user, file_name, time_start, time_end, time_window, ip_view, k_max, k_min, k, conf_id* }

  Example of Body (JSON):

```
{
    "user": 0,
    "file_name": "file_1.csv",
    "time_start": "2018-02-14 0:00:00",
    "time_end": "2018-02-14 23:00:00",
    "time_window": 60,
    "ip_view": "172.31.",
    "k_max": 20,
    "k_min": 3,
    "k": -1,
    "conf_id": 1
}
```

  Headers:

```
headers = { 'Content-Type': 'application/json'}
```

  Response: 200 - OK or 400 - Bad Request

- **GET** {DevEnvironment}/api/clusters/results?file_name=x&result_id=x

  Query parameters: { *file_name, result_id* }

  Body Keys: { *view, user, n_max* }

  Example of Body (JSON):

```
{
    "view": "Int",
    "user": 0,
    "n_max": 1
}
```

  Headers:

```
headers = { 'Content-Type': 'application/json'}
```

  Example of response (JSON):

```
{
    "Results": [
        {
            "cluster_number": 10,
            "ips": [
                "172.31.69.25"
            ],
            "n_elements": 1,
            "timestamp": "2018-02-14 14:00:00"
        }, ...
    ]
}
```

- **GET** {DevEnvironment}/api/clusters/results/all?file_name=x&result_id=x&timestamp=x

  Query parameters: { *file_name, result_id, timestamp* }

  Body Keys: { *view, user* }

  Example of Body (JSON):

```
{
    "view": "Int",
    "user": 0
}
```

Headers:

```
headers = { 'Content-Type': 'application/json'}
```

Example of response (JSON):

```
{
    "Results": {
        "cluster_number": {
            "172.31.0.2": 1,
            "172.31.64.1": 5,
            "172.31.64.100": 2,
            ...
        }
    }
}
```

- **GET** {DevEnvironment}/api/clusters/heatmap?file_name=x

  Query parameters: { *file_name* }

  Body Keys: { *user, config_id, timestamp, view* }

  Example of Body (JSON):

```
{
    "user": 0,
    "config_id": 1,
    "timestamp": "2018-02-14 14:00:00",
    "view": "Int"
}
```

  Headers:

```
headers = { 'Content-Type': 'application/json'}
```

  Example of response (JSON):

```
{
    "columns":["SrcIPContacted","SrcPortUsed",...],
    "index":[0,1, ...],
    "data":[[data_1.1, ...], [data_2.1, ...], ...]
}
```

- **Metrics**
  - **GET** {DevEnvironment}/api/metrics?file_name=x&result_id

    Query parameters: { *file_name, result_id* }

    Body Keys: { *user, view, white_list, red_list_dict* }

    Example of Body (JSON):

```
{
    "user": 0,
    "view": "Int",
    "white_list":["172.31.0.2"],
    "red_list_dict":  [{"ip": "172.31.69.25", "time_start": "2018-0
       2-14 14:01:00", "time_end": "2018-02-14 15:31:00", "view": "
       Int"}, ... ]
}
```

    Headers:

```
headers = { 'Content-Type': 'application/json'}
```

    Example of response (JSON):

```
{
    "timestamp":{"0":"2018-02-14 12:30:00", ...},
    "accuracy":{"0":100.0, ...},
    "precision":{"0":0.0, ...},
    "recall":{"0":0.0, ...},
    "f1_score":{"0":0.0, ...},
    "TP":{"0":0, ...},
    "TN":{"0":426, ...},
    "FN":{"0":0, ...},
    "FP":{"0":0, ...}
}
```

B Summary table of the CSE-CIC-IDS2018 dataset, from [57]

Date	Attack Name	Victim	Start Time	End Time
Day 1	FTP-BruteForce	172.31.69.25	10:32	12:09
14-02-2018	SSH-BruteForce	172.31.69.25	14:01	15:31
Day 2	DoS-GoldenEye	172.31.69.25	9:26	10:09
15-02-2018	DoS-Slowloris	172.31.69.25	10:59	11:40
Day 3	DoS-SlowHTTPTest	172.31.69.25	10:12	11:08
16-02-2018	DoS-Hulk	172.31.69.25	13:45	14:19
Day 4	DDoS attacks-LOIC-HTTP	172.31.69.25	10:12	11:17
20-02-2018	DDoS-LOIC-UDP	172.31.69.25	13:13	13:32
Day 5	DDOS-LOIC-UDP	172.31.69.28	10:09	10:43
21-02-2018	DDOS-HOIC	172.31.69.28	14:05	15:05
Day 6	Brute Force -Web	172.31.69.28	10:17	11:24
22-02-2018	Brute Force -XSS	172.31.69.28	13:50	14:29
	SQL Injection	172.31.69.28	16:15	16:29
Day 7	Brute Force -Web	172.31.69.28	10:03	11:03
23-02-2018	Brute Force -XSS	172.31.69.28	13:00	14:10
	SQL Injection	172.31.69.28	15:05	15:18
Day 8	Infiltration	172.31.69.24	10:50	12:05
28-02-2018	Infiltration	172.31.69.24	13:42	14:40
Day 9	Infiltration	172.31.69.13	9:57	10:55
01-03-2018	Infiltration	172.31.69.13	14:00	15:37
Day 10	Bot	172.31.69.23; 172.31.69.17; 172.31.69.14; 172.31.69.12; 172.31.69.10; 172.31.69.8; 172.31.69.6; 172.31.69.26; 172.31.69.29; 172.31.69.30	10:11	11:34
02-03-2018	Bot	172.31.69.23; 172.31.69.17; 172.31.69.14; 172.31.69.12 172.31.69.10; 172.31.69.8; 172.31.69.6; 172.31.69.26; 172.31.69.29; 172.31.69.30	14:24	15:55

C Day 1 IP 172.31.66.82 logs

Logs headers: Src IP, Src Port, Dst IP,Dst Port, Timestamp, Tot Fwd Pkts, Tot Bwd Pkts, TotLen Fwd Pkts, TotLen Bwd Pkts, SYN Flag Cnt, Flow Pkts/s, Flow Duration, Protocol

212.92.116.6,58588, 172.31.66.82,3389,14/02/2018 08:31:47 AM,9,12,1396,1699,1,6,3318464,6

212.92.116.6,53103, 172.31.66.82,3389,14/02/2018 08:31:59 AM,9,12,1434,3047,1,7,2969111,6

212.92.116.6,64716, 172.31.66.82,3389,14/02/2018 08:32:33 AM,11,15,1461,3066,1,1,17529903,6

212.92.116.6,53597, 172.31.66.82,3389,14/02/2018 08:33:10 AM,8,10,1128,1618,1,3,5371904,6

212.92.116.6,54395, 172.31.66.82,3389,14/02/2018 08:34:14 AM,8,10,1171,1600,1,7,2534559,6

212.92.116.6,56520, 172.31.66.82,3389,14/02/2018 08:35:03 AM,7,8,1128,1581,1,7,2049612,6

212.92.116.6,65269, 172.31.66.82,3389,14/02/2018 08:35:20 AM,7,8,1128,1581,1,7,2068463,6

212.92.116.6,57187, 172.31.66.82,3389,14/02/2018 08:56:40 AM,9,11,1262,1640,1,3,5481273,6

212.92.116.6,52990, 172.31.66.82,3389,14/02/2018 09:08:56 AM,7,8,1128,1581,1,7,2069119,6

212.92.116.6,55376, 172.31.66.82,3389,14/02/2018 09:09:05 AM,9,12,1347,1933,1,5,3504034,6

⋮

[More than 100 similar logs]

⋮

212.92.116.6,52051, 172.31.66.82,3389,14/02/2018 03:10:55 PM,9,12,1298,2167,1,6,3121095,6

212.92.116.6,54726, 172.31.66.82,3389,14/02/2018 03:10:56 PM,9,11,1171,1600,1,2,8082751,6

212.92.116.6,58386, 172.31.66.82,3389,14/02/2018 03:43:07 PM,2,5,0,0,1,0,21040212,6

212.92.116.6,58443, 172.31.66.82,3389,14/02/2018 03:44:24 PM,10,14,1461,3066,1,5,4265040,6

212.92.116.6,58241, 172.31.66.82,3389,14/02/2018 03:44:31 PM,7,8,1144,1581,1,7,1980099,6

212.92.116.6,61321, 172.31.66.82,3389,14/02/2018 03:44:47 PM,8,9,1821,1581,1,8,2018961,6

212.92.116.6,53283, 172.31.66.82,3389,14/02/2018 03:45:19 PM,9,12,2026,2791,1,7,2951865,6

212.92.116.6,64064, 172.31.66.82,3389,14/02/2018 03:45:44 PM,7,10,1128,1911,1,4,3438536,6

212.92.116.6,61033, 172.31.66.82,3389,14/02/2018 03:46:11 PM,8,11,1262,1699,1,6,3061070,6

212.92.116.6,53241, 172.31.66.82,3389,14/02/2018 03:46:22 PM,7,8,1128,1581,1,7,1992653,6

D Day 7 IP 172.31.67.62 logs

Logs headers: Src IP, Src Port, Dst IP,Dst Port, Timestamp, Tot Fwd Pkts, Tot Bwd Pkts, TotLen Fwd Pkts, TotLen Bwd Pkts, SYN Flag Cnt, Flow Pkts/s, Flow Duration, Protocol

82.94.226.105,80,172.31.67.62,50412,23/02/2018 12:02:34 PM,1,1,0,0,0,74074,27,6

82.94.226.105,80,172.31.67.62,50415,23/02/2018 12:02:39 PM,1,1,0,0,0,19607,102,6

82.94.226.105,80,172.31.67.62,50415,23/02/2018 12:02:39 PM,1,1,0,0,0,19607,102,6

82.94.226.105,80,172.31.67.62,50414,23/02/2018 12:02:39 PM,1,1,0,0,0,74074,27,6

82.94.226.105,80,172.31.67.62,50413,23/02/2018 12:02:39 PM,1,1,0,0,0,19417,103,6

138.108.7.20,80,172.31.67.62,50418,23/02/2018 12:02:59 PM,0,2,0,0,0,181818,11,6

209.85.203.155,80,172.31.67.62,50417,23/02/2018 12:02:59 PM,0,2,0,0,0,64516,31,6

52.202.72.22,80,172.31.67.62,50482,23/02/2018 12:03:59 PM,0,2,0,0,0,250000,8,6

23.46.53.206,80,172.31.67.62,50456,23/02/2018 12:03:59 PM,0,2,0,0,0,333333,6,6

23.15.129.79,80,172.31.67.62,50454,23/02/2018 12:03:59 PM,0,2,0,0,0,153846,13,6

⋮

[More than 2000 similar logs]

⋮

216.58.211.162,80,172.31.67.62,61094,23/02/2018 02:56:48 PM,0,2,0,0,0,60606,33,6

185.114.5.21,80,172.31.67.62,61099,23/02/2018 02:56:53 PM,0,2,0,0,0,142857,14,6

185.114.5.21,80,172.31.67.62,61106,23/02/2018 02:56:58 PM,0,2,0,0,0,58823,34,6

137.74.127.78,80,172.31.67.62,60939,23/02/2018 02:57:06 PM,1,1,128,212,0,0,48127006,6

185.114.5.21,80,172.31.67.62,61113,23/02/2018 02:57:08 PM,0,2,0,0,0,80000,25,6

185.114.5.21,80,172.31.67.62,61111,23/02/2018 02:57:08 PM,0,2,0,0,0,55555,36,6

185.114.5.21,80,172.31.67.62,61115,23/02/2018 02:57:13 PM,0,2,0,0,0,58823,34,6

185.114.5.21,80,172.31.67.62,61122,23/02/2018 02:57:18 PM,0,2,0,0,0,90909,22,6

185.114.5.21,80,172.31.67.62,61121,23/02/2018 02:57:18 PM,0,2,0,0,0,105263,19,6

E Day 9 IP 172.31.65.77 logs

Logs headers: Src IP, Src Port, Dst IP,Dst Port, Timestamp, Tot Fwd Pkts, Tot Bwd Pkts, TotLen Fwd Pkts, TotLen Bwd Pkts, SYN Flag Cnt, Flow Pkts/s, Flow Duration, Protocol

169.254.169.254,80,172.31.65.77, 49178,01/03/2018 08:15:55 AM,0,3,0,0,0,0,94308701,6

68.67.178.246,443,172.31.65.77, 49243,01/03/2018 08:16:04 AM,0,2,0,0,0,222222,9,6

216.58.198.74,443,172.31.65.77, 49211,01/03/2018 08:16:04 AM,0,2,0,0,0,105263,19,6

216.137.41.237,443,172.31.65.77, 49196,01/03/2018 08:16:04 AM,0,2,0,0,0,142857,14,6

68.67.178.246,443,172.31.65.77, 49241,01/03/2018 08:16:04 AM,0,2,0,0,0, 49999,40,6

216.58.198.67,443,172.31.65.77, 49210,01/03/2018 08:16:04 AM,0,2,0,0,0,44444,45,6

68.67.178.246,443,172.31.65.77, 49240,01/03/2018 08:16:04 AM,0,2,0,0,0,55555,36,6

216.137.41.237,443,172.31.65.77, 49194,01/03/2018 08:16:04 AM,0,2,0,0,0,117647,17,6

54.230.19.148,443,172.31.65.77, 49250,01/03/2018 08:16:04 AM,0,2,0,0,0,199999,10,6

216.137.41.237,443,172.31.65.77, 49195,01/03/2018 08:16:04 AM,0,2,0,0,0,142857,14,6

⋮

[More than 25000 similar logs]

⋮

23.15.7.107,443,172.31.65.77, 49938,01/03/2018 05:30:26 PM,1,1,0,31,0,2000000,1,6

34.196.242.1,443,172.31.65.77, 49937,01/03/2018 05:30:26 PM,1,1,0,31,0,2000000,1,6

52.72.253.69,443,172.31.65.77, 49939,01/03/2018 05:30:31 PM,1,1,0,31,0,2000000,1,6

34.196.242.1,443,172.31.65.77, 49941,01/03/2018 05:30:36 PM,1,2,0,31,0,30303,99,6

23.15.7.107,443,172.31.65.77, 49940,01/03/2018 05:30:36 PM,1,1,0,31,0,2000000,1,6

52.219.88.51,443,172.31.65.77, 49942,01/03/2018 05:30:39 PM,0,2,0,0,0,64516,31,6

52.72.253.69,443,172.31.65.77, 49944,01/03/2018 05:30:41 PM,1,1,0,31,0,2000000,1,6

23.15.8.129,443,172.31.65.77, 49943,01/03/2018 05:30:41 PM,1,1,0,31,0,2000000,1,6

23.15.8.129,443,172.31.65.77, 49945,01/03/2018 05:30:46 PM,1,2,0,31,0,32258,93,6

34.196.242.1,443,172.31.65.77, 49946,01/03/2018 05:30:46 PM,1,1,0,31,0,99999999,0,6