Clustering Dynamically-Defined NetFlow and Windows Event Features for Intrusion Detection

João Pedro Esteves Pinheiro

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Professor Doutor Miguel Nuno Dias Alves Pupo Correia

Doutor Luís Filipe Xavier Mendonça Dias

Examination Committee

Chairperson: Professor João Manuel de Freitas Xavier

Supervisor: Professor Miguel Nuno Dias Alves Pupo Correia

Members of the committee: Professor José Silvestre Serra da Silva

Professor João Nuno De Oliveira e Silva

Colonel Henrique Martins dos Santos Cunha

November/2023

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

This master's thesis was developed with essential contributions fundamental to the conclusion and fulfillment of all the objectives.

I would like to thank my supervisors, Professor Miguel Nuno Dias Alves Pupo Correia and Doutor Luís Filipe Xavier Mendonça Dias, specialists in the field with extensive experience, for their support, availability, sharing of knowledge, and important contributions to the completion of this work.

I would also like to thank my family for the continuous support and encouragement provided through the development of this work.

To my fiancée, Daniela, a special thank you for all the love and untiring support throughout this master's thesis. I would also like to thank her for the valuable inputs and revisions of this document.

To my comrades at the Military Academy, I would like to thank you for all your support during these six arduous years in the Military Academy and Instituto Superior Técnico.

To all those who have contributed in one way or another to my academic career, I dedicate my dissertation.

Resumo

Nos últimos anos, o mundo tem assistido a um aumento dramático dos ciberataques. O cenário de ameaças está a evoluir sem precedentes, deixando as organizações vulneráveis a ciberataques. Os sistemas de deteção de intrusões (SDIs) têm como objetivo proteger e monitorizar as redes ameaçadas. Os SDIs devem evoluir para defender os sistemas atuais de forma consistente e robusta.

O principal objetivo deste trabalho é mostrar as vantagens de incluir os eventos do Microsoft (MS) Windows (EMSW) nos SDIs atuais. Os EMSW fornecem uma representação aprofundada do comportamento das máquinas MS Windows. Este sistema implementa a extração dinâmica de características tanto do host como da rede. A única instrução definida pelo utilizador é o número total de características a extrair. Além disso, o sistema seleciona um conjunto fixo de características do host e da rede. O processo de clustering inclui três algoritmos que funcionam em simultâneo. O sistema também explora quatro métodos de deteção de anomalias para ultrapassar a dificuldade de detetar várias vítimas do mesmo ataque.

O sistema proposto é avaliado com um dataset artificial e público que contém ambos os tipos de dados. As métricas de avaliação obtidas são depois comparadas com abordagens semelhantes para demonstrar as vantagens da inclusão de EMSW no sistema. Os resultados mostram um aumento do F1-score de 3%, do recall de 5% e da mesma precisão em comparação com a abordagem com melhor desempenho. Os resultados alcançados pelo sistema proposto sublinham as contribuições oferecidas para a área dos SDIs.

Palavras-chave: Cibersegurança, Aprendizagem Automática, Sistema de Deteção de Intrusões, Análise de Segurança, NetFlow, Logs.

Abstract

In recent years, the world has witnessed a dramatic increase in cyberattacks. The threat landscape is evolving unprecedentedly, leaving organizations vulnerable to cyberattacks. Intrusion Detection Systems (IDS) aim to protect and monitor the threatened networks. IDSs must evolve faster than attackers to defend current systems consistently and robustly.

The main goal of this work is to show the benefits of including Microsoft (MS) Windows Events (MSWEs) in current IDSs, taking advantage of the full range of data types. MSWEs provide an in-depth representation of the behaviour of MS Windows machines. This system implements the dynamic extraction of both host and network features. The only user-defined instruction for feature extraction is the total number of features to extract. In addition, the system selects a fixed set of host and network features. The clustering process comprises three clustering algorithms working simultaneously to detect outliers. The system also explores four methods for outlier detection to overcome the difficulty of detecting multiple victims in the same attack.

The proposed system is then evaluated with a public artificial dataset containing both data types. The evaluation metrics achieved by the different outlier detection metrics are then compared to similar approaches to demonstrate the benefits of including MSWEs in the system. The results show an increase of the F-score by 3%, recall by 5%, and the same precision compared with the best-performing similar approach. The results achieved by the proposed system emphasise the contributions offered by this work to the field of IDSs.

Keywords: Cybersecurity, Machine learning, Intrusion Detection System, Security Analytics, NetFlow, Logs.

Table of Contents

TABLE OF CONTENTS ..... **IX**
TABLE OF FIGURES ..... **XI**
LIST OF TABLES ..... **XV**
ACRONYMS ..... **XVII**
1. INTRODUCTION **1**
1.1 CHALLENGES AND MOTIVATION 1
1.2 OBJECTIVES 2
1.3 REPORT STRUCTURE 2
2. BACKGROUND **3**
2.1 NETWORK FLOWS 3
2.2 MICROSOFT WINDOWS EVENTS 4
2.3 INTRUSION DETECTION SYSTEMS 6
2.4 MACHINE LEARNING 7
2.4.1 Supervised Learning *8*
2.4.2 Unsupervised Learning *9*
3. RELATED WORK **13**
3.1 HOST INTRUSION DETECTION SYSTEMS 13
3.2 NETWORK INTRUSION DETECTION SYSTEMS 15
3.3 CLUSTERING-BASED INTRUSION DETECTION SYSTEMS 16
4. PROPOSED SOLUTION **19**
4.1 OVERVIEW 19
4.2 DETAILED DESCRIPTION 19
4.2.1 Dynamic Feature Selection *19*
4.2.2 Normalisation *21*
4.2.3 Clustering *21*
4.2.4 Outlier Detection *22*
4.3 IMPLEMENTATION 23
5. EXPERIMENTAL EVALUATION **25**
5.1 EVALUATION METRICS 25
5.2 EVALUATION OF THE SYSTEM CONSIDERING HOST DATA 25
5.2.1 BOT (02-03-2018), with host data and x=20 ..... *26*
5.2.2 BOT (02-03-2018), with host data and x=30 ..... *30*
5.2.3 BOT (02-03-2018), with host data and x=50 ..... *32*
5.2.4 BOT (02-03-2018), with host data and x=80 ..... *35*
5.2.5 BOT (02-03-2018), with host data and x=100 ..... *38*

5.2.6 Outlier Detection with host data and x=100 on 02-03-18 (BOT) 43
5.2.7 Infiltration (01-03-18), with host data and x=100 48
5.2.8 Outlier Detection with host data and x=100 on 01-03-18 (Infiltration)..... 50
5.2.9 Summary of results with host data..... 55
5.3 EVALUATION OF THE SYSTEM CONSIDERING HOST AND NETFLOW DATA..... 56
5.3.1 BOT (02-03-18), with host, network data and X=100 56
5.3.2 Outlier detection with host, network data and x=100 on 02-03-18 (BOT)..... 60
5.3.3 Infiltration (01-03-18), with host, network data and x=100 64
5.3.4 Outlier detection with host, network data and x=100 on 01-03-18(Infiltration)..... 67
5.3.5 Summary of results with host and network data..... 71
5.4 COMPARISON WITH DIFFERENT ALGORITHMS 73
6. CONCLUSION AND FUTURE WORK..... 75
REFERENCES..... 77

Table of figures

Figure 1 - Framework general description..... 19

Figure 2 - Implementation of the proposed approach where each rectangle represents a Python script. In red is the transformation of the system logs from XML format to CSV. In blue is the feature extraction of host data and normalisation of the features. In green is the feature extraction of network data and the normalisation of the features. In Orange is the clustering of both types of data, the outlier detection methods, and the attack detection. 24

Figure 3 – Heat map of K-means algorithm on 02-03-18 for host data and X=20. 27

Figure 4 – Machine distribution by cluster with K-means algorithm for x=20 and host data on 02-03-18, with the victims represented in red. 28

Figure 5 - Machine distribution by cluster with DSCAN algorithm for x=20 and host data on 02-03-18, with the victims represented in red. 29

Figure 6 - Machine distribution by cluster with Agglomerative algorithm for x=20 and host data on 02-03-18, with victims represented in red. 29

Figure 7 - Machine distribution by cluster with K-means algorithm for x=30 and host data on 02-03-18, with the victims represented in red. 31

Figure 8 - Machine distribution by cluster with DBSCAN algorithm for x=30 and host data on 02-03-18, with the victims represented in red. 31

Figure 9 - Machine distribution by cluster with Agglomerative algorithm for x=30 and host data on 02-03-18, with the victims represented in red. 32

Figure 10 - Machine distribution by cluster with K-means algorithm for x=50 and host data on 02-03-18, with the victims represented in red. 33

Figure 11 - Machine distribution by cluster with DBSCAN algorithm for x=50 and host data on 02-03-18, with the victims represented in red. 34

Figure 12 - Machine distribution by cluster with Agglomerative algorithm for x=50 and host data on 02-03-18), with the victims represented in red. 34

Figure 13 - Machine distribution by cluster with K-means algorithm for x=80 and host data on 02-03-18, with the victims represented in red. 36

Figure 14 - Machine distribution by cluster with DBSCAN algorithm for x=80 and host data on 02-03-18, with the victims represented in red. 37

Figure 15 - Machine distribution by cluster with Agglomerative algorithm for x=80 and host data on 02-03-18, with the victims represented in red. 37

Figure 16 - Heat Map for the Agglomerative Clustering algorithm for x=100 and host data on 01-03-2018. 39

Figure 17 - Machine distribution by cluster with K-means algorithm for x=100 and host data on 02-03-18), with the victims represented in red. 40

Figure 18 - Machine distribution by cluster with DBSCAN algorithm for x=100 and host data on 02-03-18), with the victims represented in red. 41

Figure 19 - Machine distribution by cluster with Agglomerative algorithm for x=100 and host data on 02-03-18, with the victims represented in red. 42

Figure 20 - Graph bar of the population of cluster where victims were placed for every algorithm and every value of x on 02-03-18. 43

Figure 21 - Mean silhouette coefficient for the clusters in the K-means algorithm for x=100 and host data. The dashed red line represents the average of all the clusters. The value in red represents the average silhouette coefficient for every cluster. 45

Figure 22 - Mean silhouette coefficient for the clusters in the DBSCAN algorithm for x=100 and host data. The dashed red line represents the average of all the clusters. The value in red represents the average silhouette coefficient for every cluster. 46

Figure 23 - Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100 and host data. The dashed red line represents the average of all the clusters. ... 46

Figure 24 - Machine distribution by cluster with K-means algorithm for x=100 and host data on 01-03-18, with the victim represented in red. 49

Figure 25 - Machine distribution by cluster with DBSCAN algorithm for x=100 and host data on 01-03-18, with the victim represented in red. 49

Figure 26 - Machine distribution by cluster with Agglomerative algorithm for x=100 and host data on 01-03-18, with the victim represented in red. 50

Figure 27 - Mean silhouette coefficient for the clusters in the K-means algorithm for x=100 and host data on 01-03-18. The dashed red line represents the average of all the clusters. The cluster where the victim was placed is highlighted in red. 52

Figure 28 - Mean silhouette coefficient for the clusters in the DBSCAN algorithm for x=100 and host data on 01-03-18. The dashed red line represents the average of all the clusters..... 53

Figure 29 - Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100 and host data on 01-03-18. The dashed red line represents the average of all the clusters. The cluster where the victim was placed is highlighted in red. 54

Figure 30 - Evaluation metrics for every method described in Subsection 4.2.4 on 02-03-18 for host data. Method 2 is scoring the machines proportionally to the population of its cluster in all algorithms. Method 3 is analysis of the silhouette coefficients of every cluster. Finally, method 4 is the consideration of any cluster with less than ten elements as a positive occurrence..... 55

Figure 31 - Evaluation metrics for every method described in Subsection 4.2.4 on 01-03-18 with host data. Method 1 is the selection of one-element clusters. Method 2 is scoring the machines proportionally to the population of its cluster in all algorithms. Method 3 is analysis of the silhouette coefficients of every cluster. Finally, method 4 is the consideration of any cluster with less than ten elements as a positive occurrence. 56

Figure 32 - Heat map of K-means algorithm on 02-03-2018, X=100 with host and network data. 58

Figure 33 - Machine distribution by cluster with K-means algorithm for x=100 and host data and network data on 02-03-18, with the victims represented in red. 58

Figure 34 - Machine distribution by cluster with DBSCAN algorithm for x=100, host data and network data, on 02-03-18, with victims represented in red. 59

Figure 35 - Machine distribution by cluster with Agglomerative algorithm for x=100, host data and network data, on 02-03-18, with victims represented in red. 59

Figure 36 - Mean silhouette coefficient for the clusters in the K-means algorithm for x=100, host and network data on 02-03-2018, with host and network data. The dashed red line represents the average of all the clusters. The clusters with victims are highlighted in red. 62

Figure 37 - Mean silhouette coefficient for the clusters in the DBSCAN algorithm for x=100, host and network data on 02-03-2018. The dashed red line represents the average of all the clusters. The clusters with victims are highlighted in red. 62

Figure 38 - Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100, on 02-03-2018, with host and network data. The dashed red line represents the average of all the clusters. The clusters with the victims are highlighted in red. 63

Figure 39 - Machine distribution by cluster with K-means algorithm for x=100, host data and network data, on 01-03-18, with victims represented in red. 65

Figure 40 - Heat map of K-means algorithm on 01-03-2018, X=100 with host and network data. Inside the blue rectangle, MSWE features. Inside the orange rectangle, NetFlow features. 65

Figure 41 - Machine distribution by cluster with DBSCAN algorithm for x=100, host data and network data, on 01-03-18, with victims represented in red. 66

Figure 42 - Machine distribution by cluster with Agglomerative algorithm for x=100, host data and network data, on 01-03-18, with victims represented in red. 66

Figure 43 - Mean silhouette coefficient for the clusters in the K-means algorithm for x=100, on 01-03-2018, with host and network data. The dashed red line represents the average of all the clusters. The victim is in cluster 7. 69

Figure 44 - Mean silhouette coefficient for the clusters in the DBSCAN algorithm for x=100, on 02-03-2018, with host and network data. The dashed red line represents the average of all the clusters. 69

Figure 45 - Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100, on 02-03-2018, with host and network data. The dashed red line represents the average of all the clusters. The victim is in cluster 6. 70

Figure 46 - Evaluation metrics for every method described in Subsection 4.2.4 for host and network data on 02-03-18. Method 1 is the selection of one-element clusters. Method 2 is scoring the machines proportionally to the population of its cluster in all algorithms. Method 3 is analysis of the silhouette coefficients of every cluster. Finally, method 4 is the consideration of any cluster with less than ten elements as a positive occurrence. 72

Figure 47 - Evaluation metrics for every method described in Subsection 4.2.4 for host and network data on 01-03-18. Method 1 is the selection of one-element clusters. Method 2 is scoring the machines proportionally to the population of its cluster in all algorithms. Method 3 is analysis of the silhouette coefficients of every cluster. Finally, method 4 is the consideration of any cluster with less than ten elements as a positive occurrence. 72

Figure 48 - Comparison of the evaluation metrics of the proposed approach with DYNIDS 73

Figure 49 - Comparison of the performance of the proposed approach (host data and network data) versus other approaches, DYNIDS, Outgene and FlowHacker. 74

List of Tables

Table 1 - Information that characterizes a MSWE. 5

Table 2 - MSWEs of interest [11, 12]. 6

Table 3 - Fixed features for the network data, each one is considered from the source and the destination point of view. 20

Table 4 - Fixed features for the host data. 21

Table 5 - IDs of the MSWE selected for x=20 and host data on 02-03-18. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent. 27

Table 6 - Cluster where the victims are placed by every algorithm for x=20 and k=9. 28

Table 7 - IDs of the MSWE selected for x=30 with host data. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent. 30

Table 8 - Cluster where the victims are placed by every algorithm for x=30, with host data on 02-03-18. 30

Table 9 - IDs of the MSWE selected for x=50 and host data. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent. 32

Table 10 - Cluster where the victims are placed by every algorithm for x=50 and host data on 02-03-18. 33

Table 11 - IDs of the MSWE selected for x=80 and host data. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent. . 35

Table 12 - Cluster where the victims are placed by every algorithm for x=80 and host data on 02-03-18. 36

Table 13 - IDs of the MSWE selected for x=100 and host data. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent. . 38

Table 14 - Cluster placement of every victim by the clustering algorithms for x=100 and host data on 02-03-18. 39

Table 15 - Summary of results for the clustering with features extracted from MSWE for the day 02-03-2018. 42

Table 16 - Scores of the victims on 02-03-18 with host data and x=100. 44

Table 17 - Evaluation metrics for the second method on day 02-03-18, with host data and x = 100. 44

Table 18 - Evaluation metric for the third criteria of Subsection 4.2.4, on 02-03-18 with host data. 47

Table 19 - Evaluation metric for the fourth criteria of Subsection 4.2.4, on 02-03-18 with host data. 47

Table 20 - IDs of the MSWE selected for x=100 and host data on 01-03-18. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent. 48

Table 21 - Evaluation metric for the first criteria of Subsection 4.2.4, on 01-03-18 with host data. 51

Table 22 - Scores for the machines on 01-03-18, with host data. The machines with blue background achieved a score below 5% (0.064774) of the average score. 51

Table 23 - Evaluation metric for the second criteria of Subsection 4.2.4, on 01-03-18 with host data. 52

Table 24 - Evaluation metric for the third criteria of Subsection 4.2.4, on 01-03-18 with host data. 54

Table 25 - Evaluation metric for the fourth criteria of Subsection 4.2.4, on 01-03-18 with host data 55

Table 26 - Cluster results for each victim and algorithm for x=100, host and network data on 02-03-18. 57

Table 27 - Evaluation metric for the first criteria of Subsection 4.2.4, on 02-03-18 with host and network data. 60

Table 28 - Scores of the victims on 02-03-18 with host and network data and x=100. 61

Table 29 - Evaluation metric for the second criteria of Subsection 4.2.4, on 02-03-18 with host and network data. 61

Table 30 - Evaluation metric for the third criteria of Subsection 4.2.4, on 02-03-18 with host and network data. 63

Table 31 - Evaluation metric for the fourth criteria of Subsection 4.2.4, on 02-03-18 with host and network data. 64

Table 32 - Evaluation metrics for the first method, considering host and network data, x=100 on 01-03-18 (infiltration attack). 67

Table 33 - Scores of the top 10 machines for 01-03-2018, with network and host data. 68

Table 34 - Evaluation metrics for the second method, considering host and network data, x=100 on 01-03-18 (infiltration attack). 68

Table 35 - Evaluation metrics for the third method, considering host and network data, x=100 on 01-03-18 (infiltration attack). 70

Table 36 - Evaluation metrics for the third method, considering host and network data, x=100 on 01-03-18 (infiltration attack). 71

Acronyms

ADFALD	Australian Defence Force Academy Linux Dataset
BDHDLS	Big Data based Hierarchical Deep Learning System
CIDS	Combined Intrusion Detection System
CNN	Convolutional Neural Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DoS	Denial of Service
EDR	Endpoint Detection and Response
HIDS	Host-based Intrusion Detection System
HTTP	Hypertext Transfer Protocol
ID	Identification
IDS	Intrusion Detection System
IOC	Indicator Of Compromise
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information Export
IPS	Intrusion Prevention System
MIDS	Mixed Intrusion Detection System
ML	Machine Learning
MLP	Multi-Layer Perceptron
MS	Microsoft
MSE	Mean Square Error
MSWE	Microsoft Windows Event
MSWEL	Microsoft Windows Event Log
NBA	Network Behaviour Analysis
NGIDS-DS	Next Generation Intrusion Detection Systems Data Set
NIDS	Network-based Intrusion Detection System
OCSVM	One Class Support Vector Machine
OPTICS	Ordering Points To Identify the Clustering Structure
OS	Operating System
R2L	Root to Local
SCTP	Stream Control Transport Protocol
SSC	Sub-Space Clustering
SSC-OCSVM	Sub-Space Clustering – One Class Support Vector Machine

SSE	Sum of Squared-Error
SVM	Support Vector Machine
TCP	Transmission Control Protocol
U2R	User to Root
UDP	User Datagram Protocol
WIDS	Wireless-based Intrusion Detection System
XML	Extensible Markup Language

1. Introduction

1.1 Challenges and Motivation

With the increasing dependency on technology comes a need for better security systems. The unavoidable digitization of every organization's process exposes them to cyberattacks that can compromise the operation and security of said organization.

According to the Center for Strategic and International Studies, cybercrime monetary cost ascended to 945 billion dollars in the global economy until 2020 [1]. Just in 2020, global spending on cybersecurity mounted to 145 billion dollars globally. Furthermore, in a universe of 1500 companies surveyed, 96% claimed to have experienced cyber incidents; of those, only 951 companies had a response plan. What is even more concerning is that among these prepared companies only 32% characterize the response plan as actually effective.

The Portuguese National Cybersecurity Center states that cyberattacks in Portugal increased by 26% between 2020 and 2021, having registered 1781 incidents just in 2021 [2]. The biggest victims of cyberattacks in Portugal are banking companies, representing 13% of all cyberattacks. Based on these statistics, the current need for better security systems is blatant, given the rise in the number and complexity of recent attacks.

Traditional Intrusion Detection Systems (IDSs) use features extracted from the network or from the host to detect possible attacks on the network. This work aims to develop a system that considers both approaches to deliver more precise results. Furthermore, the most common IDSs need historical data from previous attacks to determine suspicious behaviour, thus limiting the system's capacity to detect new attacks with different characteristics from previous ones. Using machine learning and clustering techniques allows the system to detect abnormal behaviour worthy of being considered an attack without prior knowledge of past attacks. The system also dynamically extracts features from both data types (network and host data), thus enabling the clustering algorithms to consider the most valuable features for each unique scenario. These methods allow the system to detect a large variety of attacks, even if they are entirely new.

Therefore, by combining machine learning techniques, host and network data, this work presents a new and improved intrusion detection approach to detect unseen attacks, providing a reliable, precise, and efficient system. Furthermore, the proposed approach implements four methods for outlier detection. Previous approaches detected difficulties in classifying outliers when the attacks target multiple victims; therefore, a study was conducted to evaluate these methods in order to overcome this obstacle.

The present work aims to fuse network data and host data to apply three different clustering algorithms to detect the existence of victims in an organization's network. The dataset used is CSE-CIC-IDS2018 from the Canadian Institute for Cybersecurity.

1.2 Objectives

In this work is presented an IDS capable of handling real-world data in production conditions. Through machine learning techniques, dynamic selection of features, and a combination of network and host data, the system is able to detect and alert the existence of suspicious activity in the organization's network. The development of this system consisted of the following phases:

1. Acquire data for testing. This data must contain network flows and host logs.
2. Develop an approach to process the data, including dynamic feature extraction and clustering of network and host data.
3. Comparison of the evaluation metrics with similar systems.

1.3 Report Structure

This document is structured in different chapters that provide the reader with a coherent understanding of the proposed system and the required background knowledge. Chapter 1 delves into the challenges and the motivation that accentuate the importance of this document. It explores the reasons that promoted the need for this study, laying the foundation for the subsequent chapters.

Chapter 2 further explores the key concepts, ensuring the reader has the foundational knowledge to understand the proposed system. Chapter 3 explores prior work in different key related areas and gives a contextual background to this work. Chapter 4 provides a general and detailed description of the proposed system. Chapter 5 presents the experimental evaluation of the approach for the dataset CSE-CIC-IDS2018. Finally, Chapter 6 provides the conclusion for this work, summarises the advancements made and future work.

2. Background

This chapter describes basic concepts fundamental to understanding the development and implementation of the proposed approach. The first concept is network flows in Section 2.1. Section 2.2 introduces the Microsoft Windows event logs (MSWELs). The notion of IDS and its various types are presented in Section 2.3. Finally, Section 2.4 describes the fundamental concepts of machine learning and is divided into Subsection 2.4.1 and Subsection 2.4.2, Supervised Learning and Unsupervised Learning, respectively.

2.1 Network Flows

Network Flows are a popular data source for Network Intrusion Detection Systems (NIDS). Flow export is a passive network monitoring approach where packets are aggregated into flows and exported for storage and analysis [3]. Examining network usage is essential for catching denial-of-service (DoS) attacks, data extraction, and other events that may represent a risk to the system. NetFlow and Internet Protocol Flow Information Export (IPFIX) are two examples of flow export protocols.

NetFlow is the most widely used flow measurement solution. Patented in 1996 by Cisco, NetFlow was only widely adopted in 2002 with the launch of NetFlow v5. In 2004, NetFlow v5 was replaced by the more flexible NetFlow v9, which introduced support for adaptable data formats through templates, IPv6, Virtual Local Area Networks and Multi-Protocol Label Switching. Routers running NetFlow maintain a flow cache containing flow records describing the router's traffic. A NetFlow is characterized as a unidirectional sequence of packets. Each flow has a set of seven values that establish unique keys: source Internet Protocol (IP) address, destination IP address, source port, destination port, protocol type, type of service and ingress interface [4]. NetFlow is generated by several algorithms that determine if a packet belongs to an existing flow or a new one should be created. These algorithms also handle the ageing and expiration of flows. The expired flows are transferred to flow collectors through NetFlow Export User Datagram Protocol (UDP) datagrams. NetFlow collectors gather data from different sources and perform data reduction through filtering and aggregation. Then, the flow information is stored in flat files ready for further processing (e.g., analysis).

IPFIX is an alternative to NetFlow established in 2008 by the Internet Engineering Task Force. IPFIX was based on Cisco NetFlow v9 and improved by implementing a flexible definition of network flow and the runtime description of record formats through templates based on a well-defined, extensible information model [5]. IPFIX provides a data model independent of the transport protocol. In an IPFIX flow, a message is the fundamental element and contains a header and one or more sets. A set may be a data set containing records or a template set containing template records. Templates describe the layout of data records, giving IPFIX its flexibility.

The structure of IPFIX's message is well adapted to extensive collections of records with minimal semantic variation. IPFIX protocol specifications define Stream Control Transport Protocol (SCTP) as the primary choice for transport, given that IPFIX meets the identified requirements of reliable, secure, congestion-aware data transfer via transport of SCTP. Transmission Control Protocol (TCP) and UDP

are also supported. Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol can also be used if transport sessions are stored in a file and sent by these protocols.

Although the recommended transport protocol for IPFIX is SCTP, its implementation can be complex because the support for SCTP in central operating systems is limited. Furthermore, transmitting SCTP packets over the open internet can also be challenging due to the need for more support for this technology [3]. Because of that, SCTP should be implemented when possible, and TCP should be implemented otherwise.

2.2 Microsoft Windows Events

Microsoft (MS) Windows is the most popular operating system for desktop and laptop computers worldwide and, as such, is targeted more frequently by malware authors than other operating systems[6]. Therefore, incorporating MSWEL in the proposed intrusion detection system is of most interest. MSWELs make available a large number of events that depict the current and past system's state [7]. MSWELs are a collection of MS Windows Events (MSWEs) organized by the source or nature of the MSWEs.

MSWEs provide a detailed description of a system's activities, representing an essential instrument for detecting unauthorized activities and security threats [8]. The 2012 Verizon Data Breach report states that 84 per cent of cyberattack victims had evidence of the breach in their event logs [9]. Each MSWE is identified with a unique Identification (ID) [10]. MSWEs are stored in Extensible Markup Language (XML) format, a binary language. As such, MSWE can be interpreted by the MS Windows Event Viewer, which translates MSWE from XML format to plain text format. MSWE can be divided into different categories [11]:

- Application log: Events belonging to this category are logged by applications or programs. These logs are related to the software applications running on the system.
- Security log: These events are called audit events and contain security-related events.
- Setup log: Events that are produced when a computer is configured.
- System log: System log events are classified as an error, warning, or information. These events are logged by the Operating System (OS).
- Forward Events log: Events forwarded by other computers.

Furthermore, MSWEs are classified with different priority levels, such as:

- Critical (1): The MSWE classified with a critical priority-level can prevent the system or applications from functioning correctly. These events demand immediate attention.
- Error (2): These events are not as urgent as Critical priority level events but must be addressed to prevent system instability. These events can be linked to severe problems such as data loss.
- Warning (3): MSWEs with a warning priority level provide an alert for possible future problems. It is advisable to resolve warning-level MSWE since they can aggravate to a higher priority level.

- Information (4): These events only provide information about the system's operation, applications, driver, or services.
- Verbose (5): These events are only produced by certain applications. A MSWE with verbose level priority is usually used for debugging and troubleshooting.
- Audit Success (6): Indicate a successfully audited access or action.
- Audit Failure (7): Indicate a failed attempt at an audited access or action.

MSWE can be uniquely identified by a set of information that characterizes the event. This information can be helpful to incorporate MSWE in an IDS since it allows the system to correlate events in a period and to a specific machine. Table 1 summarizes the information portrayed by the MSWE.

Table 1 - Information that characterizes a MSWE.

Information	Description
Date	The date the event occurred.
Time	The time the event occurred.
User	The username of the user who logged on when the event occurred.
Computer	The name of the computer.
Event ID	The identifier of a specific event.
Source	The program or component that caused the event.
Type	The type of the event.

There are around four hundred MSWEs. However, only a percentage of these events are relevant for intrusion detection. Table 2 presents a list of the IDs and a description of important events for an IDS [11, 12].

Table 2 - MSWEs of interest [11, 12].

Event ID	Description
4624	Successful account logon.
4625	Failed account logon.
4634	An account was logged off.
4672	Special privileges assigned to new logon.
4688	A new process has been created.
4720	A user account was created.
4722	A user account was enabled.
4728	A member was added to a security-disabled global group.
4740	User account blocked.
4768	Kerberos authentication ticket requested.
4771	Failed Kerberos pre-authentication.
4778	Reconnected session on a Windows station.
4801	Workstation unlocked.
4905	An attempt was made to unregister a security event source.
4097	Auditing settings on object were changed.

2.3 Intrusion Detection Systems

Soniya defines IDS as “an art of detecting the intruder activities” [13]. An IDS is the process of monitoring events occurred in a network or computer system to detect signs of malicious activities. After detecting an intrusion by a third party, an IDS sends a report to a base station. There are three primary intrusion detection methodologies [14]:

- Signature-based detection, also known as knowledge-based detection, this methodology is based on a pattern or string recognition from previously known attacks. It is a simple and effective method to detect known signatures, however ineffective against unknown attacks, evasion attacks, and variants of known attacks.
- Anomaly-based detection: This method flags attacks by maintaining a typical behaviour profile and detecting deviations from this behaviour. Anomaly-based detection systems are effective in detecting new and unforeseen attacks. However, resorting to this method introduces difficulties in triggering alerts at the right time.

- Specification-based detection: This method compares a system activity to established acceptable protocol behaviours [15]. These acceptable protocol behaviours are based on protocol standards from international standard organizations. Protocol state tracing and examination are resource-intensive processes and might be incompatible with some OSs.

Furthermore, IDS can also be divided into various types depending on the nature of the data used to detect the intrusion. The main types are [16]:

- Host Intrusion Detection Systems (HIDSs) can monitor a computer's internal behaviour [16], detecting internal attacks and other attacks that might be undetectable by traditional NIDSs. The proposed approach is, in part, a HIDS resorting to MSWEs, which are generated in every Windows machine. These events provide helpful information about each system's operations and are an essential resource for an IDS [7]. The two main types of HIDS are the following:
    - Signature-based: This approach searches for known signatures of intrusion events.
    - Anomaly-based: This approach classifies the system activity as normal or abnormal by comparing this activity to a predefined collection of standard patterns.
- NIDS: Unlike the previous IDS, NIDS receives data from the network traffic and monitors the global system or network communications with a low implementation cost. NIDS can inspect traffic and detect attacks at the application level.
- Wireless-based Intrusion Detection System (WIDS): WIDS is similar to NIDS but captures wireless network traffic, sensor networks and mesh networks.
- Network Behaviour Analysis (NBA): The NBA system monitors network traffic to detect attacks with unusual traffic flows.
- Mixed Intrusion Detection System (MIDS): As the name indicates, MIDS aims to combine the benefits of the previous types of IDS.

Nowadays, attackers remain for longer periods in enterprise networks, allowing the increase of the complexity of attacks [17]. This attack strategy type is Advanced Persistent Threats. By avoiding actions that a classic IDS would easily detect, attackers can stay undetected for months or even years. Modern IDSs resort to signature-based and anomaly-based methods and are now known as Endpoint Detection and Response (EDR) systems. EDR combines HIDS and NIDS that can detect and react to a threat by blocking it. The traditional category for this type of system is Intrusion Prevention System (IPS).

2.4 Machine Learning

This section focuses on literature surveys for Machine Learning (ML) algorithms and their main categories: supervised and unsupervised learning. ML algorithms are designed to receive input data and train to predict outcomes. ML algorithms allow the recognition of very complex patterns, otherwise imperceptible to the human observer and the processing of large amounts of data.

ML algorithms receive training data and train a model to generate reasonable predictions for future new data. There are multiple types of machine learning algorithms: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, multi-tasking learning,

ensemble learning and instance-based learning [18]. Given the large number of algorithms and the impossibility of approaching all of them in this work, the focus is only on the two main types: supervised and unsupervised learning. The main difference between these two types is in the input data received. While unsupervised learning algorithms receive unlabelled data, supervised learning algorithms receive labelled data.

2.4.1 Supervised Learning

Supervised learning uses labelled data sets to train an algorithm until it can replicate the patterns and relationships between input and labelled data. This training allows the algorithm to classify or predict new data outcomes using the mapping previously accomplished [18]. The main disadvantage of supervised learning is the need for sufficiently sizeable, labelled data sets. The received data set must be divided into the train and test data sets.

The most popular supervised learning algorithms are the following [19]:

- Linear Classifiers: Linear classifiers group vectors with similar features using linear decision boundaries. This algorithm is beneficial for classification problems where the classification speed is a problem.
- Logistic regression: Logistic regression is a statistical model that estimates the probability of an occurrence based on a data set of independent variables.
- Naive Bayesian networks: This algorithm uses the Bayes rule to compute an outcome based on a data set. Then, the outcome with the highest probability is predicted for the selected data set [20].
- Multi-Layer Perceptron (MLP): MLP is formed by an input, output and one or multiple hidden layers. This algorithm combines the input in a weighted sum according to each weight, and then applies an activation function. The MLP is a feed-forward neural network where the connections are always in the direction of the higher layer [21].
- Support Vector Machine (SVM): This algorithm is defined by Noble as “a mathematical entity, an algorithm (or recipe) for maximizing a particular mathematical function with respect to a given collection of data.”. The following four concepts define SVM classification [22]:
    - The separating hyperplane: This is the plane that defines the border between clusters by establishing a hyperplane that defines which cluster a data point belongs to.
    - The maximum-margin hyperplane: To define the separating plane, it is necessary to define the best hyperplane. In SVM, the chosen separating plane will have the maximum distance to its nearest data point.
    - The soft margin: This allows the SVM to define a hyperplane to deal with errors by pushing some data points to the correct side of the hyperplane. To implement soft margin its necessary to define how many data points are allowed to trespass the separating hyperplane.
    - The kernel function: Adds additional dimension to the data, allowing SVM to perform higher dimensional classification to a data set. This allows the SVM to linearly separate any data set, given that the data set has consistent labels.

- Decision Trees: Decision trees classify the data set by resorting to a series of questions about each feature of the data points. Every question corresponds to a node and according to the answer, the data point is placed in a descending node from the one corresponding to the question asked. The data point is assigned to the corresponding class according to the node it ended on [23].

2.4.2 Unsupervised Learning

Unlike the previous types, unsupervised learning does not require a labelled data set. The system acquires knowledge by learning and familiarizing itself with the vital information in the input data. The training figures and input patterns are offered to the system, which arranges the data into categories or clusters. The main algorithm for unsupervised learning is clustering:

- Clustering: Defines a structure in a collection of unlabelled data [24]. A cluster is a collection of objects which are similar between them and are dissimilar to the objects from other clusters. To group the objects in clusters, there is a need to establish a method to measure similarities between objects [25]. Two main types of measures are used to estimate this relation: distance measures and similarity measures. Distance measures should enforce the minimum distance calculated between objects belonging to the same cluster and the maximum distance between different clusters [26]. Distance metrics are an important component of a clustering algorithm since they are used to determine to which cluster, if any, an object belongs. Therefore, a detailed description of the most popular metrics is presented next:
  - Euclidean distance: Is the distance between two points in a straight line. This distance can be calculated using the Pythagorean theorem as follows: Take x_1, y_1 and x_2, y_2 as the coordinates of P_1 and P_2 respectively. The distance between these points will be equals to:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1.1}$$

    In higher dimensions, the equation is similar:

$$\sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{1.2}$$

    with n being the number of dimensions, p_i and q_i the coordinates of p and q in dimension i.
  - Manhattan Distance: The distance measured along the axes at right angles. In a plane with P_1 at (x_1, y_1) and P_2 at (x_2, y_2) the Manhattan distance is as follows:

$$Distance = |x_1 - x_2| + |y_1 - y_2| \tag{1.3}$$

- Bit-Vector Distance: An N x N matrix Mb is calculated. Each point has d dimensions, and Mb (P_y, P_j) is determined as a d-bit vector. This d-bit vector is then calculated as follows: if the numerical value of the x^{th} dimension of point P_i is greater than the numerical value of the xth dimension of point P_j, then the bit x of Mb(P_y, P_j) is set to 1 and bit x of Mb (P_y, P_j) is set to 0. Afterwards, the d-bit vectors constituting the matrix Mb are converted to integers.
- Hamming Distance: The Hamming distance is the number of positions at which the corresponding symbols differ between two strings of equal length. The Hamming distance represents the number of bits that must be changed to turn a string into another.
- Jaccard Index: The Jaccard index is a well-known measurement of the similarity between two sets, S and T, that is defined by the following formula:

$$J(S,T) = \frac{|S \cap T|}{|S \cup T|} \tag{1.4}$$

where $J(S,T) = 1$ if $|S \cup T| = 0$ [27].

- Cosine Index: Measures the similarity between two vectors of n dimensions. Given two vectors of attributes, A and B, the cosine similarity θ is represented using a dot product and magnitude. The formula is the following:

$$\theta = arcos.\frac{A \,.\, B}{||A||.||B||} \tag{1.5}$$

The value of the angle θ, is comprised between [0, π]. If θ assumes the value π, the vectors A and B are opposite. If θ assumes the value $\frac{\pi}{2}$ means the vectors, A and B, are independent. Finally, if θ assumes the value 0, the vectors, A and B, are the same.

- Dice Index: For sets X and Y of keywords used in information retrieval, dice index is defined by the following formula:

$$S = \frac{2|X \cap Y|}{|X| + |Y|} \tag{1.6}$$

The main methods of data clustering algorithms are the following:

- Hierarchical: Hierarchical algorithms presuppose constructing a hierarchy of clusters [28]. This kind of clustering algorithm is suited for hierarchical data. The number of clusters can be changed by altering the finishing stage of the algorithm. In this algorithm, the similar data points are grouped in the same cluster, and the dissimilar data points are assigned to a cluster further away.

- Partitional: Partitional clustering aims to partition the data points into clusters by minimizing the clustering criterion using iterative processes [30]. During the application of the partitional clustering, algorithm data is classified into k-clusters. To produce homogeneous clusters, criterion functions need to be established. The most popular criterion functions are the following:
  - Sum of squared error (SSE): SSE measures the variation within a cluster. The formula for calculating SSE is the following:

$$SSE = \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{1.7}$$

where x_i is the value of the i^{th} data point, n is the number of data points in the cluster and $\bar{x}$ is the mean of the values that compose the cluster.

  - Mean Square Error (MSE): MSE measures a cluster's error amount. MSE is achieved by using the following formula:

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n} \tag{1.8}$$

where n is the number of the data points composing the cluster, y_i is the data point's value and $\hat{y}_i$ is the predicted value for y_i.

  - Normalised Mean Squared Error: is essentially the same calculation as MSE but with a normalisation that allows comparing results achieved with data points in different scales.
- Density-based: Density-based clustering is an approach where clusters are high-density areas [29]. Unlike other methods, density-based clustering methods do not require the number of clusters as input parameters. Therefore, density-based clusters are data objects spread in a region of a high density of objects-regions of a low density of objects separate clusters. The density level must be established and define a high and low-density region. Some of the most well-known density-based clustering algorithms are the following [30]:
  - Density-Based Spatial Clustering of Applications with Noise (DBSCAN): DBSCAN groups objects in a cluster of a given radius (Eps) with a minimum quantity of objects within that radius (MinPts) [31]. Eps and MinPts are the user-defined input parameters. DBSCAN applies the epsilon-neighbourhood to every object (p) in the data set. Then, if a given epsilon-neighbourhood contains more objects than MinPts, a cluster is created with p as a core object. These core objects are then merged to create density-reachable clusters. This process is until no new object can be added to any cluster. DBSCAN is known to have some problems: (a) it has difficulties in identifying clusters within data of varying density; (b) it is prone to incur in the curse of dimensionality; (c) it is very sensitive to the input parameters, Eps and MinPts.

- Ordering Points To Identify the Clustering Structure (OPTICS): OPTICS overcomes a big problem of DBSCAN, detecting meaningful clusters in data of varying density [32]. Unlike DBSCAN, OPTICS does not explicitly segment the data into clusters. OPTICS produces a measure of reachability distance between points and uses this measure to cluster the data.

3. Related Work

This chapter will provide an overview of recent work in IDSs resorting to host and network data. The current chapter is divided into Section 3.1, Section 3.2, and Section 3.3. Each section will explore several related works to present their contributions to the growth of this expanding scientific field.

3.1 Host Intrusion Detection Systems

As the name indicates, this type of IDS receives data from the host systems to detect suspicious behaviour. A HIDS protects a system by collecting information about events or system logs [33].

Santos [12] proposes an approach to develop a robust and efficient process to manage relevant MSWEs to flag malicious behaviour. Santos's implementation is accomplished in two phases. The first phase takes place in a laboratory environment. The second phase is in a production environment integrated with Altice Portugal. Santos defines three activities of interest and their respective correlation rules:

- Detection of brute force attacks: Multiple MSWEs with ID 4625 in a short time interval indicate a possible brute force attack.
- Detection of Improper Use of Credentials: To detect the improper use of credentials, three MSWEs are used in conjunction. If, in a short time interval, there is a user who performs authentication in multiple workstations, a warning is triggered for improper use of credentials. To detect the possibility of these type of attack, the system makes use of MSWEs with ID 4624 (Successful account login), ID 4778 (Reconnected session on a Windows station), and ID 4801 (Workstation unlocked).
- DoS Attacks on User Accounts: Warning for possible DoS attacks is triggered by detecting multiple MSWEs with ID 4740 in a short time interval.

There are over four hundred MSWEs, and as such, there is a need to reduce the amount of MSWEs to analyse. Santos established the events presented in Table 3 as relevant for intrusion detection. To collect MSWEs, Santos selected Windows Event Collection because it is less intrusive, given that there is no need to install additional agents and no need for licensing. Santos concludes that including MSWEs in IDS increases the capacity of the system to detect malicious activities.

Liu et al. [34] propose CIDS-Net, a Combine Intrusion Detection System (CIDS) that combines host and network data to improve the performance of typical IDSs. CIDS-Net is a transformer-based deep learning model which can outperform typical NIDS by 6.36% (up to 99.89%) in the F1 Score. CIDS-Net consists of four encoders, one for each input type (Network and event features and event messages) and one that aggregates the results and predicts intrusions. The authors chose TabNet as the encoder for the network data. The encoder for event features is a state-of-the-art deep learning model. The same deep learning model is used for the event messages, adding a pre-trained Bidirectional Encoder Representations from Transformers word embedding to transform the plain text messages into vectors. Finally, all three results are aggregated by a Fully Connected Network that

predicts the results. The results are an improvement over standard NIDS, achieving an F1 score of 99.89%.

Tixteco et al. [11] propose a procedure to generate Indicators Of Compromise (IOC) using MSWEL to produce a more efficient diagnostic computer system for incident response. Indicators of compromise are forensic data found in log entries or system files. The authors consider the security events in Table 4 to detect possible signs of computer system intrusion. Tixteco et al. defined some IOC using MSEL that allow early detection of malicious activity. IOC must detect combinations of events to detect a possible attack. The authors defined IOC to be a tree of logical operators ("AND" / "OR") that together form a complex structure. Then, using predefined IOC, the system will automatically search for the sequence of events that form a known IOC tree. In a sampling performed over a thirty-computer system operating for 1 hour, 9996 events were registered. Of these 9996 events, 50% belong to the User Account Management, Logon and Logoff categories. The less-represented events are then paired with other events, which can reduce the rate of false positives by 80%.

Tran et al. [35] propose a HIDS that resorts to a Convolutional Neural Network (CNN) to use system call traces. A chain of system calls represents a series of tasks performed in the attacked system by a malicious attacker. These successions of system calls establish patterns that can be identified. The authors used the following data sets: Next Generation Intrusion Detection Systems Data Set (NGIDS-DS) and Australian Defence Force Academy Linux Dataset (ADFALD). These data sets were generated using modern computing infrastructures and contain up-to-date knowledge. Therefore, these data sets represent realistic representations of recent attacks. Furthermore, NGIDS-DS contains host log and packet capture files, thus being appropriate for developing a HIDS. The extensive amount of data in both NGIDS-DS and ADFALD (hundreds of millions of observations) is enough for the training phase of the CNN, which requires a massive amount of data. Tran et al. established a relatively simple CNN architecture that achieved a false alarm rate of 20.64% and a detection rate of 60.46%. These results are in accordance with other approaches and could be improved by applying a more complex CNN architecture according to the authors.

Berlin et al. [36] propose a system to detect malicious activity in an organization using Windows audit logs and a logistic regression machine learning classifier. The authors have concluded that Windows audit logs provide enough information to implement a robust detection system. The experimental data used for training and testing were collected from two sources: i) logs collected from users of an enterprise network; ii) sandboxed virtual machine-based runs on a set of malicious and benign binaries. Berlin et al. could not detect any malicious activities from the logs that originated in the enterprise network. The data from the sandboxed virtual machine is unlabelled; however, to use supervised learning algorithms, it is necessary to have a labelled data set. Berlin et al. used VirusTotal to label the data set produced by the sandboxed virtual machines. Berlin et al. system achieved an 83% true positive rate and a 0.1% false positive rate.

Park et al. [37] proposed a HIDS using the Siamese Network. The Siamese Network is used to test and train the model with few-shot learning method, which, according to the authors, shows excellent performance by learning a small amount of data. The author's chosen datasets were the LID-DS, a

dataset created by Leipzig University in 2018 for host-based intrusion detection systems, and the NSL-KDD, an improved version of KDD CUP 99. The data is pre-processed with one-hot encoding, which converts the data into binary format and then the values are normalised. The samples are converted to 8-bit vectors, which are grouped to form a grayscale-type image. The Siamese calculates the similarity score of these images with each cyberattack sample to determine the attack types. The performance of the proposed approach was compared with the Vanilla CNN. The results show that the proposed system achieved a precision of 93%, a recall of 92% and an F1-score of 91%.

3.2 Network Intrusion Detection Systems

NIDS monitor internal and external traffic, both incoming and outgoing, to detect unauthorized access to computer networks. Then, applying the methods explored in Chapter 2 flags suspicious behaviour.

Landress [38] produced an unsupervised learning method using the K-means algorithm and J48 Decision Tree algorithm to reduce the high numbers of false positives that current methods of intrusion detection systems produce. Landress chose the k-means algorithm for its simplicity and efficiency in operating in a large data set. The author implements feature selection using the J48 Decision Tree algorithm and self-organizing maps to achieve this purpose. Furthermore, the data set that supports this approach is the KDD CUP 99 data set. Landress uses a traffic generator to produce network traffic that can be separated into five categories: normal, DoS, scanning, Root to Local (R2L), and User to Root (U2R). After the implementation, applying the J48 Decision Tree algorithm in the KDD CUP 99 data set produced 11 features. The detection rate of Landress's approach is 99.96%, with a false positive rate of 0.04%. These results show a decrease in false positive rate over k-means without feature selection, achieving a false positive rate of 1.08%. KDD CUP 99 is an outdated data set created in 1999 with a wide variety of intrusions simulated in a military network environment.

Sahu et al. [39] propose a NIDS resorting to the Decision Tree (J48) algorithm to classify the network packets. The authors used the labelled data set Kyoto 2006+ to train and test the algorithm. Kyoto 2006+ was built on three years of real traffic data obtained from diverse types of honeypots. The authors consider the data set KDD CUP 99 outdated and, as such, not a true reflection of the current network situation and the latest attack trends. The decision tree is a supervised learning technique that classifies objects based on a series of questions about their features. Depending on the answers, the object follows a certain branch until it arrives at the last node representing the classification. This technique requires an extensive set of objects. Sahu et al. used 134665 instances to test the proposed approach. The generated tree achieved a true positive rate of 97.23% and a false positive rate of 2.77%.

Javaid et al. [40] propose an approach for a flexible NIDS capable of processing unprecedented and unpredictable data using Self-Taught Learning based on the sparse autoencoder and soft-max regression and the data set NSL-KDD. The authors aim to use deep learning-based methods to efficiently select non-redundant features. The first step of this process is to produce a good feature representation from unlabelled data. The authors used a sparse autoencoder to perform this task due to its easy implementation and good performance. Then, this feature representation is applied to

labelled data to classify this labelled data set. As described previously, KDD CUP 99 data set compromises network traffic that can be separated into five categories: normal, DoS, scanning, R2L, and U2R. The authors verified that KDD CUP 99 data set has the disadvantage of containing many redundant records, about 78% in the training data set and 75% in the test data set. Therefore, the data set NSL-KDD was proposed as an alternative. NSL-KDD presents an improvement over KDD CUP 99 by eliminating all redundant records. The results presented by Javaid et al. show that the proposed approach achieved an f-measure value of 90.40%.

Guezzaz et al. [41] proposed a reliable network intrusion detection approach using a decision tree qualifier with enhanced data quality. The authors improve the data quality of the datasets NSL-KDD and CICIDS2017 by preprocessing the network data and selecting features with entropy decision. The proposed models achieved an accuracy of 99.42% with the dataset NSL-KDD and 98.80% with the CICIDS2017. According to Guezzaz et al., the preprocessing of data and the feature selection increased the detection rate and the accuracy of the proposed IDS. The author concluded that the proposed IDS is relevant, achieves important performances, and gives relevant training by implementing fast data quality techniques.

Atefinia et al. [42] proposed a model that consists of a feedforward module, a stack of restricted Boltzmann machine modules, and two recurrent modules. The output weights of these modules are the input of an aggregator that detects the attacks. The dataset used by the authors is the CSE-CICIDS2018 dataset. Atefinia et al. resorted to a design called Modular Neural Networks, which allows the separation and scalability of the system. The results show that the proposed approach has an accuracy as high as 100% and as low as 82.83%, depending on the day of the dataset; a precision as high as 1 and as low as 0.012; a recall between 1 and 0.52; and an F1 score between 1 and 0.024; depending on the day of the dataset.

3.3 Clustering-Based Intrusion Detection Systems

Clustering-based intrusion detection systems can simultaneously isolate outliers from the information collected from network flows, MSWEL, or both. The clustering algorithms represent any unusual pattern or abnormal behaviour as an outlier that can be flagged as a possible attack.

Pu et al. [43] propose an approach to detect various attacks using an anomaly detection algorithm called Sub-Space Clustering – One-Class Support Vector Machine (SSC-OCSVM). This algorithm simultaneously implements Sub-Space Clustering (SSC) and One-Class Support Vector Machine (OCSVM). With this algorithm implemented, it will be possible to overcome a limitation found in typical clustering techniques: the result relies excessively upon the algorithm itself and its initial parameters. The authors analyse two popular clustering algorithms: k-means and DBSCAN. Despite being reasonably efficient, K-means is impractical in the real world due to the definition of k. Furthermore, since the starting centroids of the clusters are usually chosen randomly, k-means often get tied to a local optimum. The authors decided to evaluate OCSVM with the KDD-CUP 99 dataset, which was later divided into five different data sets. Four of these data sets are a different type of attack (probe or scan, DoS, U2R, and R2L), and one is a mixed subset. The performance of the proposed algorithm was

compared with the performance obtained by the algorithms K-means, DBSCAN, and SSC-EA. This comparison concluded that SSC-OCSVM could detect a large number of attacks in every subset and produce very few false alarms. Compared with the referred algorithms, SSC-OCSVM was superior, obtaining a better detection rate in three of the five data sets (R2L data set, U2R data set, and mixed data set).

OutGene [44] is an approach that aims to assist human analysts in pinpointing malicious clusters without previous knowledge about attacks. OutGene detects unprecedented attacks by clustering hosts with similar behaviour. The hosts that do not belong to a cluster are considered outliers and flagged as possible attackers. OutGene introduces the idea of using genetic zoom to extract the most relevant features for establishing the clusters. OutGene also introduces the idea of time stretching to analyse data flow at various time windows. Different attacks are executed at different paces. By analysing data flow at various time windows, the system will gain the capacity to detect a wide range of attacks independently of the pace at which they are executed. OutGene's approach is divided into two phases. The first phase is executed in pre runtime and aims to prepare the runtime phase. In the first phase, feature extraction and normalisation are performed to make all data consistent. In the second phase, at runtime, the traffic flows are processed. The K-means algorithm receives the extracted features and groups the entities with similar behaviour. If an outlier is detected, there are two possible responses. If the outlier has been seen before it is reported. Otherwise, there is the need for intervention by a human analyst using genetic zoom. The experimental evaluation was performed using datasets from Los Alamos National Laboratory and a large military infrastructure.

Andresini et al. [45] proposed CLAIRE, a multi-stage intrusion detection system that combines nearest neighbour search, clustering and CNN. The author proposed a novel encoding method that maps the network flow feature vectors into a two-dimensional representation. Then, these representations are mapped to the 2D input of a CNN. This mapping is achieved by combining clustering (with the K-means algorithm) and nearest neighbour search. The neighbourhood relation is evaluated by computing the Euclidean distance. The CNN is trained to learn a classification model that discriminates the attacking flow behaviour from the normal one. The datasets used to evaluate CLAIRE were KDDCUP99, USW-NB15 and CICIDS2017. The results show an accuracy of 98.01 and an F1-score of 95.20 with the CICIDS2017 dataset.

DYNIDS [30] is an approach that flags malicious activity without previous knowledge about attacks or training data. DYNIDS defines features at runtime dynamically. Furthermore, the features are defined according to data analysed in different time windows. Features are defined according to insights of attacker techniques from MITRE's Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) framework. Therefore, the feature selection is based on the top-used and less-used ports. DYNIDS uses three different algorithms: partition-based (K-Means), hierarchical (Agglomerative), and density-based (DBSCAN). Each of these three algorithms will be used according to an outlier score calculated using the results obtained. The experimental evaluation was performed using the NetFlow dataset publicly available (CIC-IDS-2018) and real traffic data from a large military infrastructure. DYNIDS's implementation is divided into feature engineering, dynamic feature definition, normalisation,

parameter inference, and clustering ensemble and outlier scoring. DYNIDS uses a set of 12 fixed features and a set of 4 dynamically defined features for each selected port. The dynamically defined features are selected based on the most and least used ports and the ports used by fewer machines. Then, the extracted features are normalised and given to the clustering algorithm.

Zhong et al. [46] proposed the Big Data based Hierarchical Deep Learning System (BDHDLS). BDHDLS resorts to multiple deep learning models using the hierarchical tree structure to increase the detection rate compared to single learning model approaches. The approach resorts to behavioural and content features to process network traffic characteristics and information stored in the payload. Furthermore, the authors implement big data techniques and parallel strategies to improve the construction time for deep learning algorithms. The dataset is partitioned into multiple clusters on multiple levels using K-means clustering and hierarchical clustering. After the clustering, the deep learning model is built for each cluster produced by the hierarchical tree. The deep learning model learns the distinctive data distribution for each cluster. Finally, the decision values of all the deep learning models are analysed to detect an attack or not. The datasets used by Zhong et al. were DARPA1998, ISCX2012 and CICIDS2017. The authors claim that BDHDLS improved the true positive detection rate, false positive rate, and accuracy over single learning model approaches. Furthermore, the big data techniques and parallel strategies for feature selection implemented reduce the construction times of the deep learning models substantially.

4. Proposed Solution

This chapter describes the proposed solution. It includes a general and a detailed description of the proposed architecture.

4.1 Overview

As stated in Chapter 1, there is a need for new and improved IDSs at a time when cyberattacks increase each year. The vast amount of extracted data requires automated systems which can process the data to detect underlying patterns that a human analyst can overlook. The publications reviewed in Section 3.1 resort to fixed and pre-defined sets of MSWEs to detect an intrusion.

The novelty of this work in relation to the work from the publications reviewed in Chapter 3 is the integration of host and network data in an IDS with dynamically selected network and host features. Figure 1 presents the proposed architecture:

1. First, dynamically extraction of the network and host-based features, which are added to a set of fixed features.
2. Then, clustering is applied to the feature vectors, where each vector represents a machine or user. After that, the three clustering algorithms are implemented.
3. Then, the vectors that fulfil the outlier detection criteria are flagged as possible security threats.

The proposed approach can detect attacks by analysing network and host data in a system capable of detecting unprecedented attacks with a detection rate concordance with industry standards while maintaining a low false positive rate.

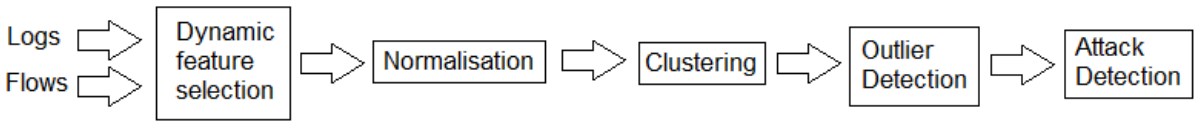

Figure 1 - Framework general description.

4.2 Detailed Description

This section presents a more detailed description of the proposed approach (Figure 1).

4.2.1 Dynamic Feature Selection

Based on the work studied in Chapter 3 and the key concepts explored in Chapter 2, the proposed system will dynamically define the host and network features. Whereas in DYINDS [30] only network features were used, in the proposed system, a procedure is established to integrate the selection of host features. Both sets of features have fixed and dynamically defined features.

For the network features, 12 fixed features and a variable number of variable features are defined (Table 3). Four features are defined dynamically for each selected port at run time proportional to the

number of ports. The ideal number of selected ports (x) is 100, as stated by [30]. Then, DYNIDS is used as the basis for the definition of port-based features. DYNIDS algorithm selects features based on the x/3 ports that appear in more flows, the x/3 ports that appear in fewer flows, and the x/3 ports used by fewer machines.

Table 3 - Fixed features for the network data, each one is considered from the source and the destination point of view.

Feature	Description
IPContacted	Number of different Internet Protocol (IP)s contacted by an entity.
ConnMade	Number of flows where the entity is the source.
PortUsed	Number of different source ports used by an entity.
PortContacted	Number of different destination 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.

The selection of features for the host data follows the same rationale as the feature selection for network data. 7 fixed features are defined for the host data. Six of these features, displayed in Table 4, are related to the priority level of the MSWE described in Section 2.2. The feature Frequency_1 is the frequency of priority level 1 events for a given machine. For example, if feature Frequency_1 has a value of 0.5 for a given machine, 50% of the events for that specific machine have a critical (1) priority level. The features Frequency_2, Frequency_3, Frequency_4 and Frequency_5 are the same as Frequency_1 but for error (2), warning (3), information (4) and verbose (5) priority level events, respectively. Furthermore, Total_Frequency is the total absolute number of events for every given machine. Finally, the feature NEventsPerMachine represents the absolute number of unique events for every machine.

In addition to these fixed features, dynamically defined features are included. The total number of these features (x) is defined in compile time (before run time). Then, x features are dynamically extracted at run time. The proposed approach selects features based on x/3 of the most frequent events, x/3 less frequent events and x/3 of the most frequent events used by less than 10 machines.

Table 4 - Fixed features for the host data.

Feature	Description
NEventsPerMachine	Number of unique MSWEs present for every Windows machine.
Frequency_1	Proportion of MSWEs classified with priority level 1 (critical).
Frequency_2	Proportion of MSWEs classified with priority level 2 (error).
Frequency_3	Proportion of MSWEs classified with priority level 3 (warning).
Frequency_4	Proportion of MSWEs classified with priority level 4 (information).
Frequency_5	Proportion of MSWEs classified with priority level 5 (Verbose).
Total_Frequency	Total number of MSWEs per machine.

4.2.2 Normalisation

After extracting the entire set of features, it is necessary to normalise the vectors of features. Normalisation is required to enforce the same range of values for every feature. In this case, the range is from 0 to 1. Without normalisation, the clustering process would be comparing values at different levels of magnitude. Features that have low absolute values would be dismissed; for example, Frequency_1 that has values lower than one and might have a significant role in detecting the outlier(s), for features with high absolute values, such as Total_frequency that have values in the hundreds of units. Normalisation ensures that the same importance is given to every feature in the clustering process. The technique used to perform the normalisation of the vectors of features is the Min-Max scaling. This technique recurs to the maximum value (X_{max}) of the vector and to the minimum value (X_{min}) of the vector to scale the values of the vector within a range between 0 and 1. This process is represented by the following formula, where X_{scaled} is the normalised value and X is the value to be normalised. The formula is then applied for every value of a given vector of features.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2)$$

4.2.3 Clustering

After the normalisation of features, the conditions to apply the clustering algorithm and identify outliers are met. The objective of clustering is to group machines with similar behaviour. Machines with different behaviour from the majority are assumed to be anomalous. The system uses different algorithms (K-Means, Agglomerative and DBSCAN).

The K-means algorithm is a centroid-based algorithm that partitions the elements of the dataset in k clusters. The algorithm's objective is to minimize the variance within the same cluster and maximize the variance between different clusters. This algorithm is computationally efficient for extensive

datasets. K-means has a time complexity of $O(n * k * i * d)$, where n is the number of data points, k is the number of clusters, i is the number of iterations until convergence, and d is the number of dimensions for each data point. The main disadvantage of the K-means algorithm is the need to specify k before run time; however, in our approach, the k value is defined at run time using the elbow method.

The DBSCAN algorithm is a density-based algorithm that groups the data points based on the density of these data points. The elements that do not belong to any cluster are classified as outliers and placed in cluster -1. This algorithm does not compute well with clusters of varying densities and when there are multiple victims. Since the victims usually have similar behaviour between them, the algorithm will interpret this cluster of victims as normal occurrences and not as outliers. In contrast, this algorithm performs well when only a single victim exists. The DBSCAN algorithm has a time complexity of $O(n^2)$, where n is the number of data points.

The Agglomerative algorithm is a hierarchical clustering algorithm where each data point starts by being its own cluster. Then, every cluster is progressively merged with the nearest cluster until the stopping criteria is achieved. In the proposed implementation, the Agglomerative algorithm stops when k clusters are left. The main advantage of this algorithm is the production of a tree-like diagram that provides insights into the hierarchical structure of data points. The main disadvantages are the computational complexity of big datasets and memory usage. The time complexity of the Agglomerative algorithm is of $O(n^3)$, where n is the number of data points.

The proposed approach integrates these three algorithms in a single system, ensuring the accuracy and comprehensive analysis necessary to process large datasets. Each algorithm provides strengths that mitigate and bridge the weaknesses of the other algorithms. Thus, improving the robustness, resilience, and adaptability of the system to detect malicious activities.

4.2.4 Outlier Detection

To correctly identify the victims as such, the system first needs to establish what criteria are used to determine which clusters are considered positive instances. The dataset CSE-CIC-IDS2018 provides two days of attacks where the victims are Windows machines. The first day is day 01-03-2018, where there is only one victim. When the attacks only target one victim, it is easy to establish a criterion for outlier detection. This criterion will be the existence of one-element clusters in the case of the K-means and Agglomerative algorithms. For the DBSCAN algorithm, the detection will be any machine placed in cluster -1, which is the cluster of machines that do not belong to any cluster.

However, on the second day (02-03-2018), there were ten individual victims. Therefore, the criteria for positive detection can't be the existence of one-element clusters. This work introduces four possible methods to overcome this difficulty of selecting the victims' cluster. The related literature did not provide any insight into methods to make this selection. Furthermore, DYNIDS [30] also faced this problem and could not find a solution. The proposed methods are:

1. One-element cluster method: One-element clusters will be classified as outliers.
2. Machine scores: Measuring machine scores based on the population of the respective cluster in every algorithm. Machines with unusually low scores might be victims despite not being

placed in one-element clusters. Throughout several tests, the optimal threshold achieved for outlier selection with this method is machines with a score lower than 5% of the average score for the entire population.

3. Silhouette coefficient method: Measuring the Silhouette coefficient for every cluster in every algorithm. Silhouette analysis is the method of scoring the clusters based on the average measure of how similar the data points within that cluster are to each other and how dissimilar they are to data points in the nearest cluster they do not belong to. Cluster with less than half of the mean value of all clusters are clusters with unusually low silhouette coefficient.
4. Low population method: Selecting clusters with less than ten elements. This method is not ideal since prior knowledge of the number of victims is used to establish this threshold. However, comparing the results obtained with the other methods might be of interest.

4.3 Implementation

The dataset CSE-CIC-IDS2018 was developed to analyse, test, and evaluate IDSs. This dataset includes seven attack scenarios: Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration from the inside. The authors use CICFlowMeter-V3 to extract network traffic from the 420 machines comprising the organization network. The dataset also provides the system logs for every machine for five of the ten days. The proposed approach requires MSWEs to perform intrusion detection, and as such, only attacks that target machines running the MS Windows OS can be considered. From the five days of system logs provided, only the ones from days 01-03-18 and 02-03-18 are useful. The attack of day 01-03-18 targets one victim running MS Windows OS and the attack of day 02-03-18 targets ten victims with the same OS.

The event log data is stored in EVTX files in XML format, the default file format used by the MS Windows OS. A Python script was developed to parse the records from XML format to a CSV file, which is then used as input for the feature extraction process. DYNIDS [30] provided the CSV files with the network traffic data. The information from the network traffic was extracted with CICFlowMeter (V4.0) from the PCAP files made available by the authors of the dataset. The CSV files for both data types are the input of two Python scripts, one for each CSV file, that perform the feature extraction and the normalisation of the host and network features. The output of these scripts is a CSV file, one with network features and the other with host features, which are then the input of the final script that performs the clustering of both sets of features jointly. The last steps, as mentioned in Section 4.1, are implementing the outlier detection methods that determine the presence or absence of attacks. Figure 2 illustrates the implementation of the system. The figure is divided in different coloured rectangles that allude to the different Python scripts developed.

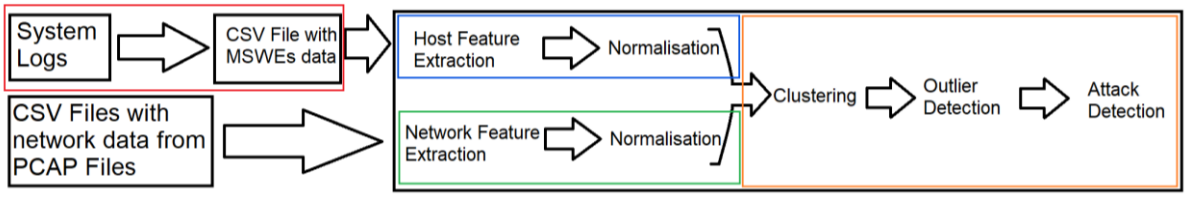

Figure 2 - Implementation of the proposed approach where each rectangle represents a Python script. In red is the transformation of the system logs from XML format to CSV. In blue is the feature extraction of host data and normalisation of the features. In green is the feature extraction of network data and the normalisation of the features. In Orange is the clustering of both types of data, the outlier detection methods, and the attack detection.

Furthermore, the Python (v3) implementation of the proposed approach resorts to several libraries to aid in the normalisation, clustering, and structure of data. The Pandas library is an open-source library that allows for easy manipulation of data frames to read and write data in multiple formats, such as CSV. The Pandas library also allows the filtration and merging of data frames, which is an essential tool when working with large quantities of data.

Another important library used in this implementation is the library Scikit-learn. Scikit-learn is an open-source library for ML and data analysis. This library provides various algorithms for different tasks, such as classification, regression, clustering, dimensionality reduction, and more. In the proposed implementation, the Scikit-learn library is primarily used for applying the clustering algorithms (K-means, DBSCAN, and Agglomerative).

5. Experimental Evaluation

In this chapter, the evaluation metrics described in Section 5.1 are calculated for the results achieved by the proposed approach with host data (Section 5.2) and a combination of host and network data (Section 5.3). The results with solely network data are already performed in DYNIDS [30] and, as such, are not presented in this work. This chapter aims to study the evolution of the evaluation metrics when the system receives host data, network data (described in DYNIDS [30]) and the combination of both. Then, the results achieved by the different data types are compared to analyse the impact of the incorporation of MSWEs information in the system.

5.1 Evaluation metrics

To evaluate the performance of the proposed approach, the evaluation metrics are established. These metrics classify the results obtained and the effectiveness of the approach. The proposed system will be evaluated regarding the following parameters:

- True Positive Rate: The relation between anomalous instances correctly classified and total number of anomalous instances presented in the data set. Anomalous instances are the clusters that fulfil the criteria defined in Subsection 4.2.4.
- False Positive Rate: The relation between anomalous instances incorrectly classified and total number of anomalous instances presented in the data set.
- Precision: This metric evaluates the quality of the positive predictions made by the system and is calculated as $\frac{TP}{TP + FP}$, where TP is anomalous instances classified correctly and FP is normal instances classified as anomalous. The precision metric evaluates how precise the system is when considering that an instance is positive.
- Accuracy: Accuracy is defined by the ratio of correctly predicted instances over the total number of instances.
- Recall: Recall measures how well the system can retrieve the positive instances from the dataset. Recall quantifies the proportion of actual positives instances that the system correctly identified.
- F1 score: F1 score is a metric that considers the precision and recall and provides a balance between both. This metric is important when the class distribution is imbalanced. An imbalanced class distribution is when one class (positive class) has much lower population than the other class (negative class).

5.2 Evaluation of the system considering host data

The following subsections present the results obtained by clustering the host data from the dataset CSE-CIC-IDS2018. This data was extracted from MSWELs. There are seven fixed features: NEventsPerHost, Frequency_1, Frequency_2, Frequency_3, Frequency_4, Frequency_5, and Total_Frequency. The feature NEventsPerHost provides the number of unique MSWE IDs the machine produces. The features Frequency_1, Frequency_2, Frequency_3, Frequency_4, and Frequency_5 are the proportion of MSWEs classified as critical (1), error (2), warning (3), Information (4), and verbose

(5). Finally, the feature Total_Frequency represents the total number of events per machine registered for the day in analysis.

In addition to the fixed features, there are several dynamically defined features. These features are extracted as a proportion of a given x (total number of dynamically defined host features). There are three criteria to extract the features:

- X/3 most frequent MSWE IDs.
- X/3 most frequent and used by less than ten machines MSWE IDs.
- X/3 less frequent MSWE IDs.

The following subsections aim to determine the optimal value of x (total number of dynamically defined host features) that provide the best results for intrusion detection for the dataset in use.

5.2.1 BOT (02-03-2018), with host data and x=20

The attack classified as BOT in the CSE-CIC-IDS2018 dataset refers to a botnet attack executed on 02-03-2018 that compromised ten machines. These machines, known as bots, are under the control of an attacker. The dataset authors use Zeus, a Trojan horse malware package that runs on MS Windows [47]. In addition to Zeus, they use Ares botnet, an open-source botnet with added capabilities. In this attack, the bots were ordered to take screenshots every 400 seconds.

In the following subsections, are performed various tests with an incremental value of the total number of dynamically extracted features (x). These tests aim to determine the optimal number of features that produce the best results. Subsection 5.2.1 concerns the results obtained for the attacks BOT of day 02-03-2018 with 20 dynamically defined features.

For x = 20, there are a total of 18 MSWEs selected; these events are presented in the following Table 5. The number of extracted features is an approximated value of x because the features are extracted according to 3 criteria. Each criterion produces x/3 features, as stated in Subsection 4.2.1. The value considered in this subsection for x is 20, which is not divisible by 3, and as such, the value considered for every criterion is the whole number of the division, which in this case is 6. The extracted features are then added, and the total number is 18 for this case.

Table 5 presents the 18 extracted features represented with three different colours. In blue are represented the event IDs most frequent in the entire population of Windows machines. The event IDs most frequent and used by less than ten machines are in green. Finally, in orange are presented the event IDs least frequent in the entire population of Windows machines.

Table 5 - IDs of the MSWE selected for x=20 and host data on 02-03-18. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent.

MSWE ID					
7036	7031	7024	36888	6	6013
36874	41	6008	7022	265	257
10114	219	1501	36887	1111	1076

After the extraction of these features from the data, the clustering algorithms are implemented. Figure 3 presents a heat map that represents the results obtained by the K-means algorithm. The red arrows indicate the clusters that contain victims in their population. The fixed feature “NEventsPerHost” has a high normalised value in all three clusters. Other than that, there isn’t any feature that stands out from this heat map in the victim clusters.

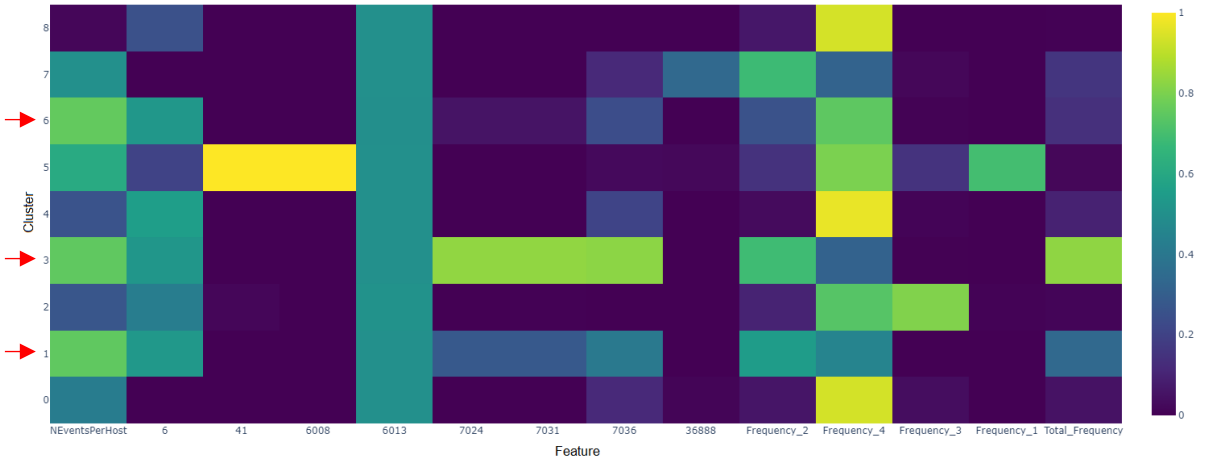

Figure 3 – Heat map of K-means algorithm on 02-03-18 for host data and X=20.

Table 6 presents the results obtained for every victim of the BOT attack. In the column “victim” are the IP addresses of all the ten victims. Then, the columns “K-means”, “DBSCAN”, and “Agglomerative” represent the cluster where the corresponding victim was placed.

Table 6 - Cluster where the victims are placed by every algorithm for x=20 and k=9.

Victim	K-means	DBSCAN	Agglomerative
172.31.69.6	1	7	3
172.31.69.8	3	12	2
172.31.69.10	1	7	3
172.31.69.12	3	12	2
172.31.69.14	6	7	0
172.31.69.17	1	7	3
172.31.69.23	6	7	0
172.31.69.26	6	-1	0
172.31.69.29	1	7	3
172.31.69.30	1	7	3

The results presented in Table 6 lead to the conclusion that the victims are placed in multiple clusters. Figure 4 illustrates the results for the K-means algorithm graphically. The y-axis enumerates the numbers of the clusters, and the x-axis represents the IP addresses of the clustered machines. In red are represented the victims. The results show that 5 of the victims were placed in cluster 1, 2 victims were placed in cluster 3 and 3 victims were placed in cluster 6. Furthermore, cluster 1 has a population of 57 elements, cluster 3 has a population of 26 elements, and cluster 6 has 85 elements.

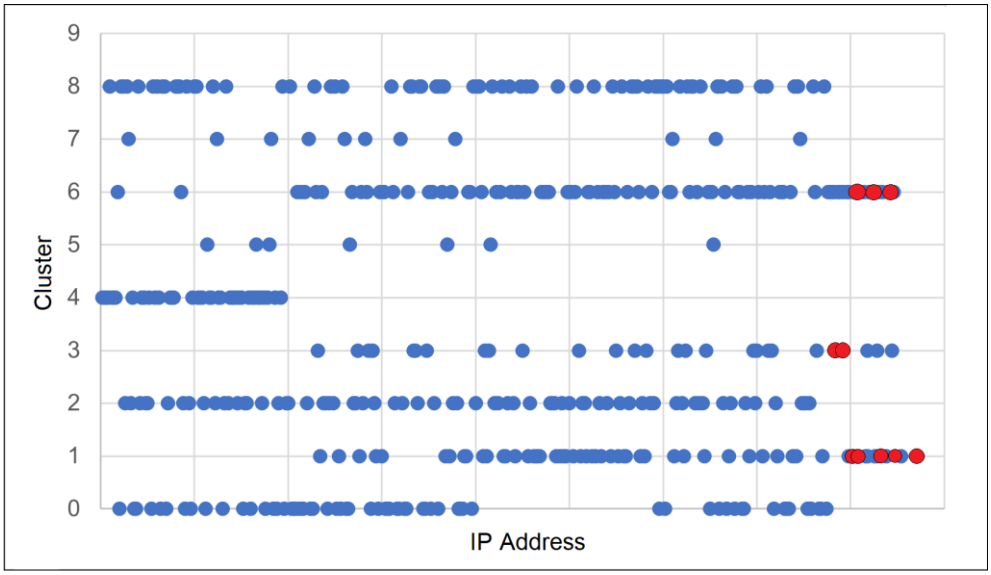

Figure 4 – Machine distribution by cluster with K-means algorithm for x=20 and host data on 02-03-18, with the victims represented in red.

The results obtained by the DBSCAN algorithm are presented in Figure 5. The victims are not concentrated in a single cluster, as is the aim. Only one of the victims is classified as an outlier and consequently placed in cluster -1. Cluster 7 has 7 victims and cluster 12 has 2 victims.

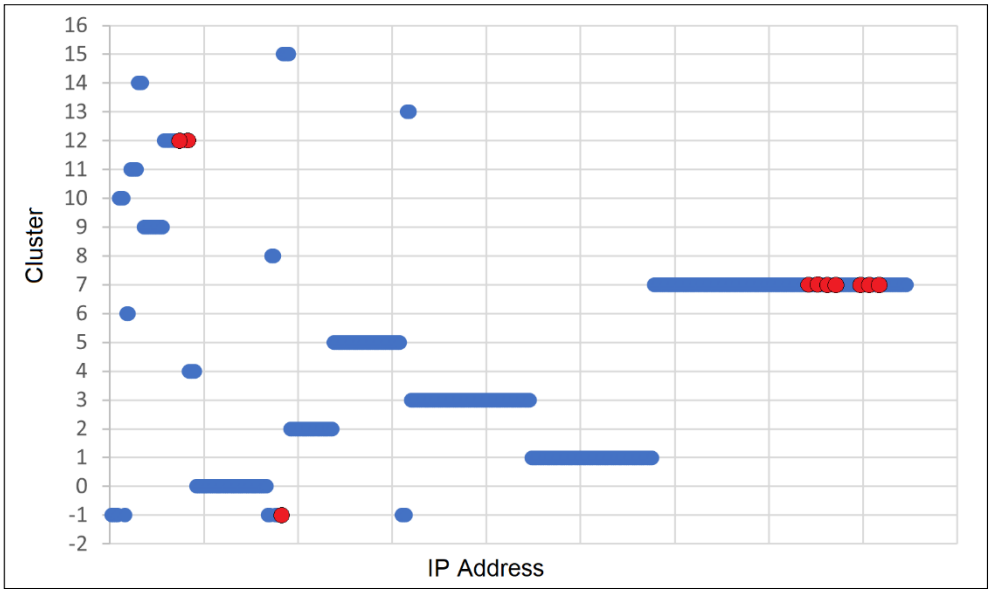

Figure 5 - Machine distribution by cluster with DSCAN algorithm for x=20 and host data on 02-03-18, with the victims represented in red.

The results provided by the Agglomerative algorithm are similar to the other algorithms. The victims are scattered between clusters, as represented in Figure 6.

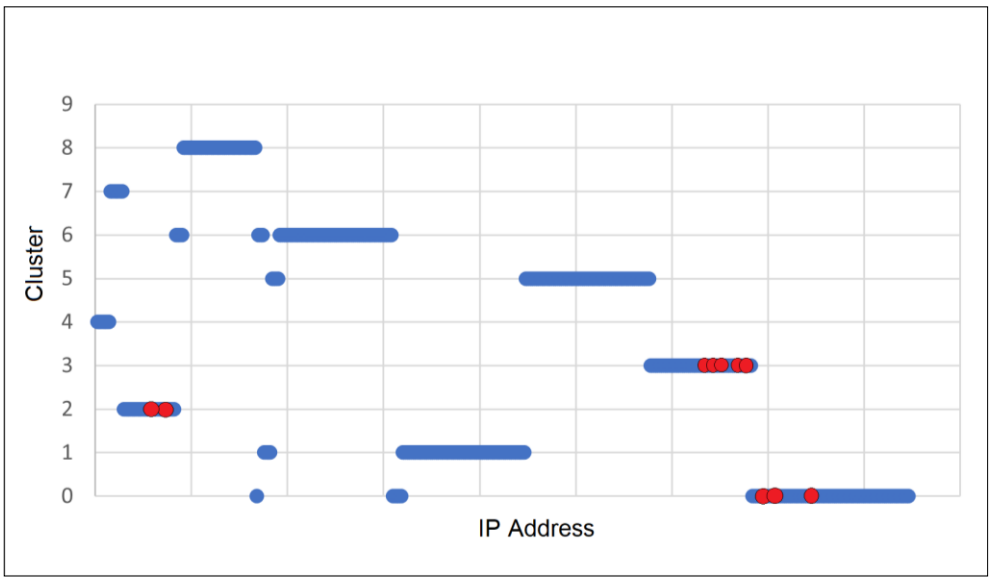

Figure 6 - Machine distribution by cluster with Agglomerative algorithm for x=20 and host data on 02-03-18, with victims represented in red.

The proposed approach aims to identify the victims by placing them as outliers. To achieve this detection, the victims must be differentiated from the other machines. This differentiation is obtained by creating a cluster without non-victim machines and, ideally, the full population of victims. As portrayed in Figure 4, Figure 5, and Figure 6, the victims were dispersed between several clusters. This is because

the number of considered features is insufficient to identify the victims' abnormal behaviour. In the following subsections, the evaluation continues with an increased value of features (x).

5.2.2 BOT (02-03-2018), with host data and x=30

In this subsection, the attack is the same as in the previous subsection. However, the value of x is increased to 30 to improve the results obtained. A value of x=30 provides the extraction of 28 MSWE IDs. These events are listed in Table 7. There are 30 MSWE IDs extracted; however, the event IDs with numbers 4202 and 15 are selected by two criteria ((1) - events most frequently used by less than ten machines and (2) - events less frequent), totalling 28 MSWE IDs.

Table 7 - IDs of the MSWE selected for x=30 with host data. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent.

MSWE ID									
7036	7031	7024	36888	6	6013	7023	98	7045	1
36874	41	6008	7022	265	257	258	1502	15	4202
15	4202	7011	40968	10114	219	1501	36887	1111	1076

Table 8 shows where each victim was placed by the three algorithms in use.

Table 8 - Cluster where the victims are placed by every algorithm for x=30, with host data on 02-03-18.

Victim	K-means	DBSCAN	Agglomerative
172.31.69.6	3	0	2
172.31.69.8	5	0	2
172.31.69.10	3	0	6
172.31.69.12	5	0	2
172.31.69.14	1	0	0
172.31.69.17	3	0	6
172.31.69.23	1	0	0
172.31.69.26	1	0	0
172.31.69.29	3	0	6
172.31.69.30	3	0	6

Compared with the results obtained from the previous subsection, a value of x=30 provides worse results. Whereas in the last subsection, 5 of the victims were placed in the same cluster in the

Agglomerative algorithm, in this subsection, the Agglomerative algorithm could only place 4 victims in the same cluster. The results provided by the K-means algorithm are very similar to those achieved in the previous subsection for x=20, as presented in Figure 7.

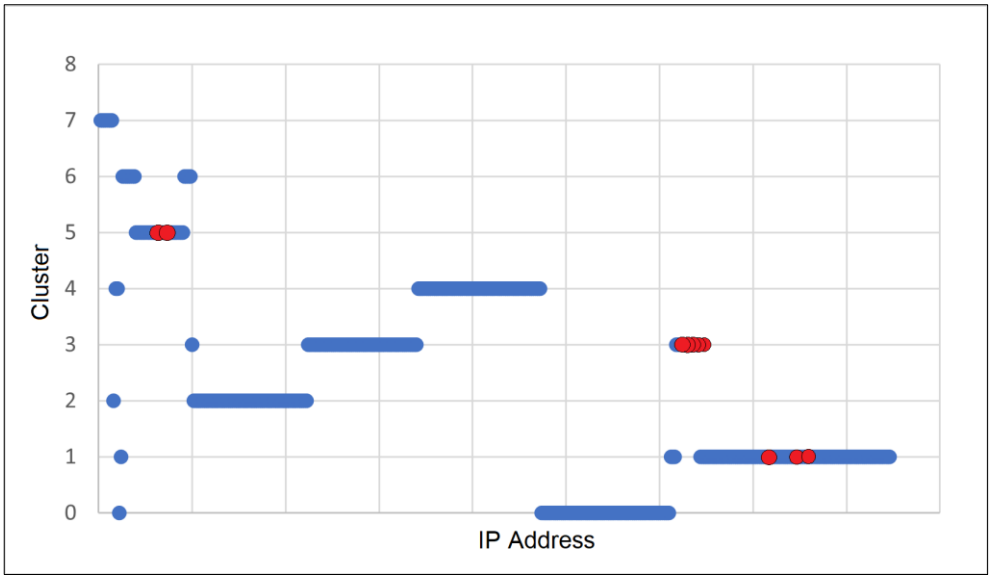

Figure 7 - Machine distribution by cluster with K-means algorithm for x=30 and host data on 02-03-18, with the victims represented in red.

For x = 30, the results of the DBSCAN algorithm are worse than for x = 20. While in the previous case, there were some victims in low populated clusters, for x=30, every victim is placed in cluster 0. As seen in Figure 8, cluster 0 contains most of the machines on this day.

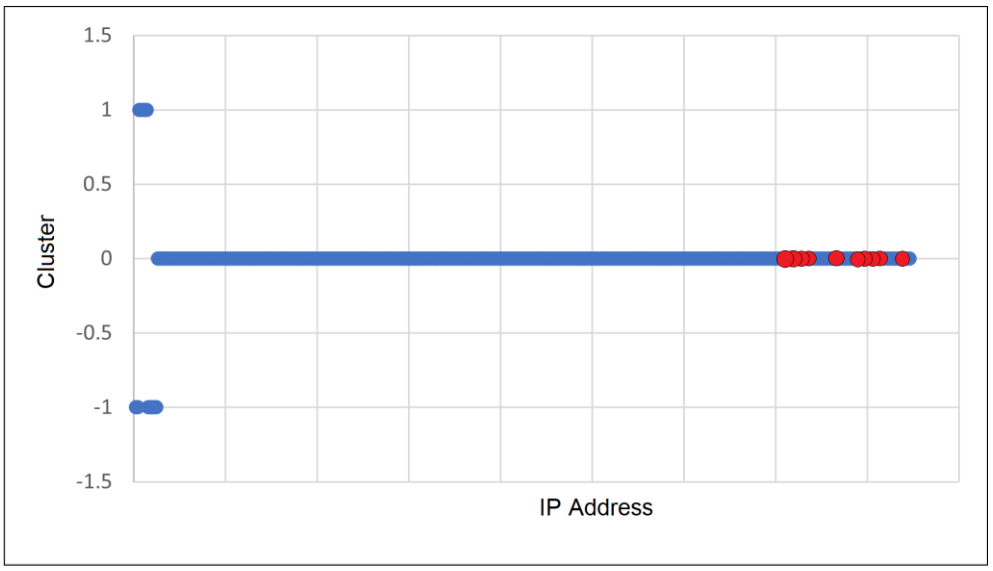

Figure 8 - Machine distribution by cluster with DBSCAN algorithm for x=30 and host data on 02-03-18, with the victims represented in red.

As mentioned, the results obtained by the Agglomerative algorithm for x = 30 decrease in quality compared with the results of the previous subsection. As represented in Figure 9, the victims are distributed between cluster 6, cluster 2 and cluster 0.

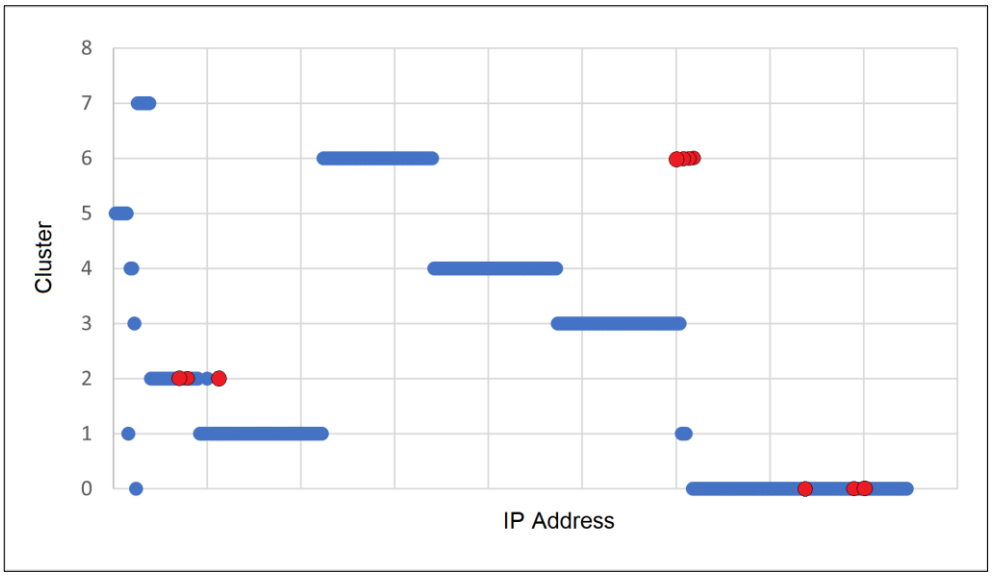

Figure 9 - Machine distribution by cluster with Agglomerative algorithm for x=30 and host data on 02-03-18, with the victims represented in red.

The results presented in this subsection are far from the objective. The expectation was to see an increase in the quality of results; however, they seem to be worse when compared with the previous subsection. Nevertheless, there is a need to increase the value of x and study the impact on the clustering process.

5.2.3 BOT (02-03-2018), with host data and x=50

For a value of x = 50, there is a total of 34 MSWE IDs extracted. In Table 9 are presented the MSWE IDs extracted for a matter of x = 50. There were extracted 16 MSWE IDs for every criterion of extraction. However, there are only 34 unique MSWE IDs when considering the repeated events.

Table 9 - IDs of the MSWE selected for x=50 and host data. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent.

MSWE IDs					
7045	7036	15	36874	7011	6008
1	7031	4202	41	40968	7022
414	7024	7011	6008	10114	265
37	36888	40968	7022	219	257
20	6	10114	265	1501	258
44	6013	219	257	36887	1502
12	7023	1501	258	1111	15
7001	98	36887	1502	1076	4202

Table 10 presents the cluster where every victim was placed in all three algorithms for a value of x = 50 and k = 8.

Table 10 - Cluster where the victims are placed by every algorithm for x=50 and host data on 02-03-18.

Victim	K-means	DBSCAN	Agglomerative
172.31.69.6	0	0	4
172.31.69.8	7	0	5
172.31.69.10	0	0	4
172.31.69.12	7	0	5
172.31.69.14	2	0	3
172.31.69.17	0	0	4
172.31.69.23	2	0	3
172.31.69.26	2	0	3
172.31.69.29	0	0	4
172.31.69.30	0	0	4

As Figure 10 shows, the results obtained by the K-means algorithm for x = 50 continue to fall short from the objective. The victims are in three different clusters. There are 5 victims in cluster 0, 3 victims in cluster 2 and 2 victims in cluster 2.

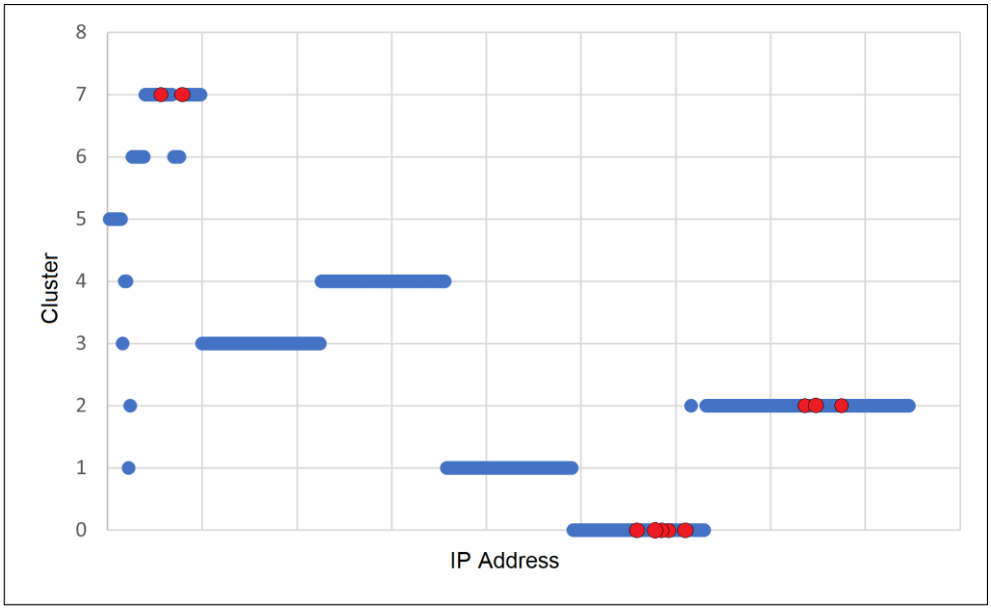

Figure 10 - Machine distribution by cluster with K-means algorithm for x=50 and host data on 02-03-18, with the victims represented in red.

As shown in Figure 11 the DBSCAN algorithm continues to provide poor results for the day in attacks in analysis. All ten victims are placed in cluster 0, as are most machines.

Figure 11 - Machine distribution by cluster with DBSCAN algorithm for x=50 and host data on 02-03-18, with the victims represented in red.

The results presented in Figure 12 are in accordance with the ones provided by the other algorithms. The victims are not placed in the same cluster and continue to be scattered between three clusters with many random machines.

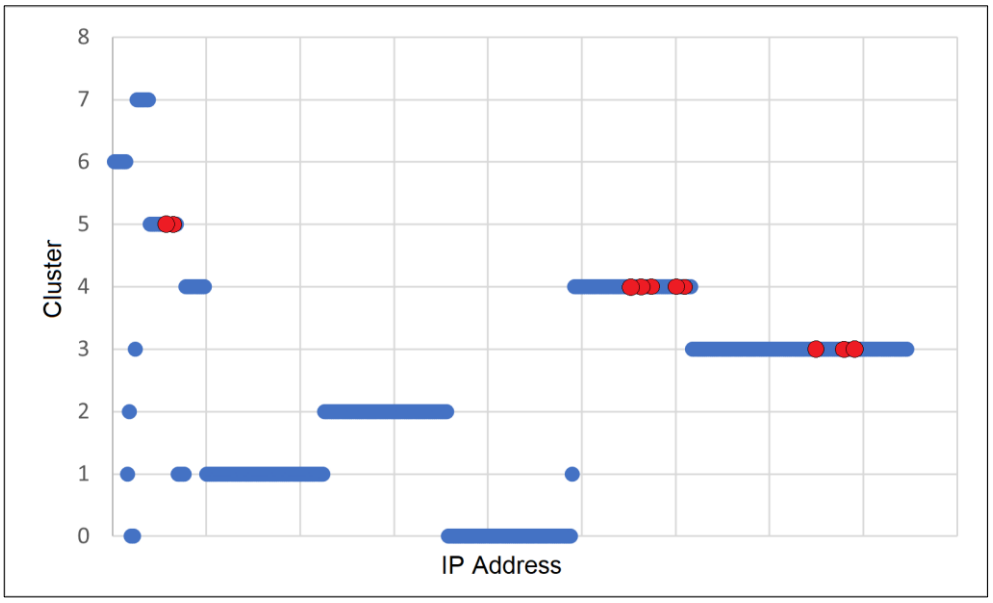

Figure 12 - Machine distribution by cluster with Agglomerative algorithm for x=50 and host data on 02-03-18), with the victims represented in red.

In conclusion, there is a need to increase the number of features to consider in the clustering process. The current number of features is insufficient to isolate the victims in a single cluster.

5.2.4 BOT (02-03-2018), with host data and x=80

For a value of x = 80, there are a total of 52 MSWE IDs extracted. In Table 11, the MSWE IDs extracted are presented. As it is possible to verify in Table 11, the MSWE IDs extracted from the criteria relative to the most frequent IDs used by fewer machines only number 18 instead of 26, equivalent to one-third of x. This is due to the limited number of MSWE IDs used by less than ten machines.

Table 11 - IDs of the MSWE selected for x=80 and host data. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent.

MSWE IDs					
7036	44	36874	219	36874	257
7031	12	41	1501	11	258
7024	7001	6008	36887	26	1502
36888	7002	7022	1111	36	15
6	27	265	1076	33	4202
6013	18	257	-	1014	7011
7023	6009	258	-	134	40968
98	6005	1502	-	2	10114
7045	50036	15	-	10149	219
1	32	4202	-	41	1501
414	7026	7011	-	6008	36887
37	55	40968	-	7022	1111
20	51046	10114	-	265	1076

The results obtained for a value of x = 80 are better than the ones obtained thus far. As shown in Table 12, the Agglomerative algorithm placed every single victim in the same cluster.

Table 12 - Cluster where the victims are placed by every algorithm for x=80 and host data on 02-03-18.

Victim	K-means	DBSCAN	Agglomerative
172.31.69.6	1	0	7
172.31.69.8	6	0	7
172.31.69.10	1	0	7
172.31.69.12	6	0	7
172.31.69.14	5	0	7
172.31.69.17	1	0	7
172.31.69.23	5	0	7
172.31.69.26	5	0	7
172.31.69.29	1	0	7
172.31.69.30	1	0	7

Figure 13 represents the distribution of machines per cluster by the K-means algorithm. The results are similar to the previous results (x=50). The victims are not placed in a single cluster. The cluster with the least elements is cluster 7, with 22 elements. Cluster 6 has 27 elements, 2 of them are victims. Cluster 5 has 3 victims in a total of 106 elements. Cluster 4, 3 and 2 have no victims and 49, 35 and 44 elements, respectively. Cluster 1 has 5 victims in a total of 72 elements. Finally, cluster 0 has 68 elements.

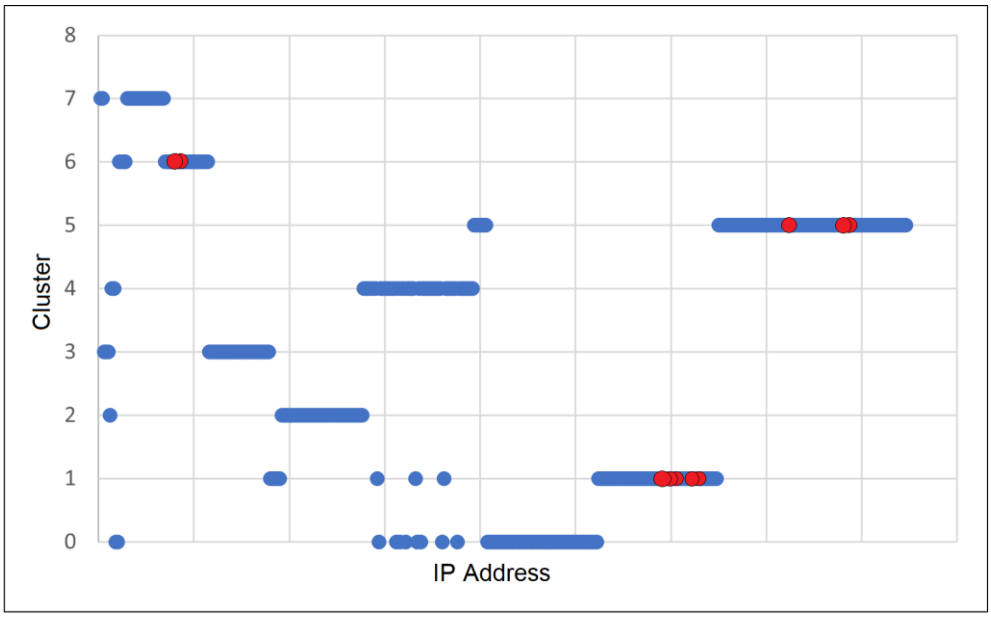

Figure 13 - Machine distribution by cluster with K-means algorithm for x=80 and host data on 02-03-18, with the victims represented in red.

The DBSCAN algorithm does not provide good results as in the previous subsections. All the victims are placed in cluster 0, as represented in Figure 14.

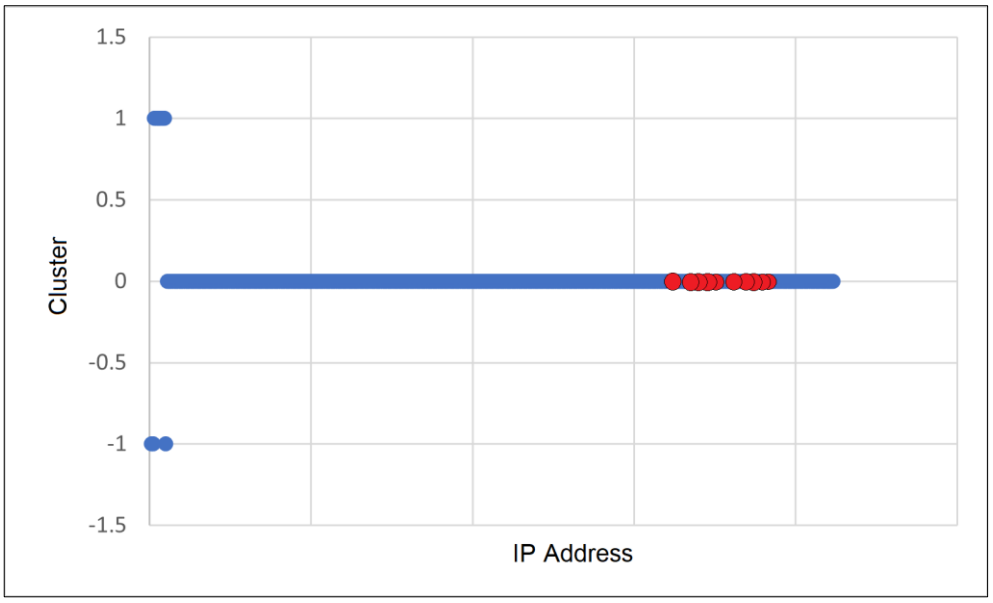

Figure 14 - Machine distribution by cluster with DBSCAN algorithm for x=80 and host data on 02-03-18, with the victims represented in red.

The Agglomerative algorithm performed the best, placing every victim in the same cluster. Figure 15 presents the distribution of machines by cluster. All ten victims were placed in cluster 7 with 7 other non-victim machines. Cluster 7 is the least populated. Cluster 6 has 30 elements, cluster 5 has 32 elements, cluster 4 has 26 elements, cluster 3 has 59 elements, cluster 2 has 48 elements, cluster 1 has 162 elements and cluster 0 has 49 elements.

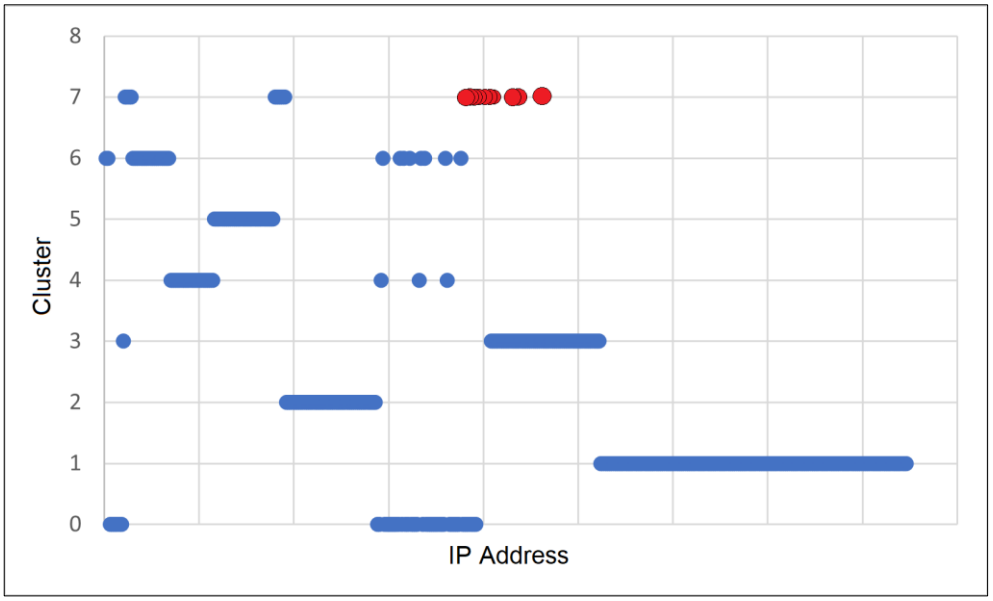

Figure 15 - Machine distribution by cluster with Agglomerative algorithm for x=80 and host data on 02-03-18, with the victims represented in red.

The results have considerably improved. K-means and DBSCAN continue to provide mediocre results. However, the results obtained with the Agglomerative algorithm for a value of x=80 are much

closer to the objective of the experimental process. However, it is necessary to increase the value of x to decrease the number of non-victim machines in the cluster with the victims. Furthermore, the increase in the value of x might improve the results obtained for the other two algorithms.

5.2.5 BOT (02-03-2018), with host data and x=100

For a value of x = 100, there are 66 dynamically defined MSWEs. Table 13 presents the MSWE IDs extracted according to the three criteria of extraction. There are 33 events extracted for the third of most frequent and 33 for the third of less frequent events. The number of MSWE IDs most frequent and used by less than ten machines continues to be the same since the maximum number has already been achieved. There is an overlap in the MSWE IDs selected, and as such, the number of total features extracted is 66 instead of the 84 presented in Table 13.

Table 13 - IDs of the MSWE selected for x=100 and host data. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent.

MSWE ID										
7036	7031	7024	36888	6	6013	7023	98	7045	1	414
37	20	44	12	7001	7002	27	18	6009	6005	50036
32	7026	55	51046	1074	13	109	50037	51047	6006	7040
36874	41	6008	7022	265	257	258	1502	15	4202	7011
40968	10114	219	1501	36887	1111	1076	-	-	-	-
14	172	153	16	19	7009	7000	36874	11	26	36
33	1014	134	2	10149	41	6008	7022	265	257	258
1502	15	4202	7011	40968	10114	219	1501	36887	1111	1076

Table 14 presents the distribution of the victims in the clusters by every algorithm.

Table 14 - Cluster placement of every victim by the clustering algorithms for x=100 and host data on 02-03-18.

VICTIM	K-MEANS	DBSCAN	AGGLOMERATIVE
172.31.69.6	1	0	3
172.31.69.8	1	0	3
172.31.69.10	1	0	3
172.31.69.12	1	0	3
172.31.69.14	7	0	3
172.31.69.17	1	0	3
172.31.69.23	7	0	3
172.31.69.26	7	0	3
172.31.69.29	1	0	3
172.31.69.30	1	0	3

As it is possible to see in Table 14, the Agglomerative algorithm placed the entire collection of victims in the same cluster. Figure 16 is a Heat Map representing every cluster's more prominent features. The features that have the most impact on cluster 3 are depicted in Figure 16 with a red outline.

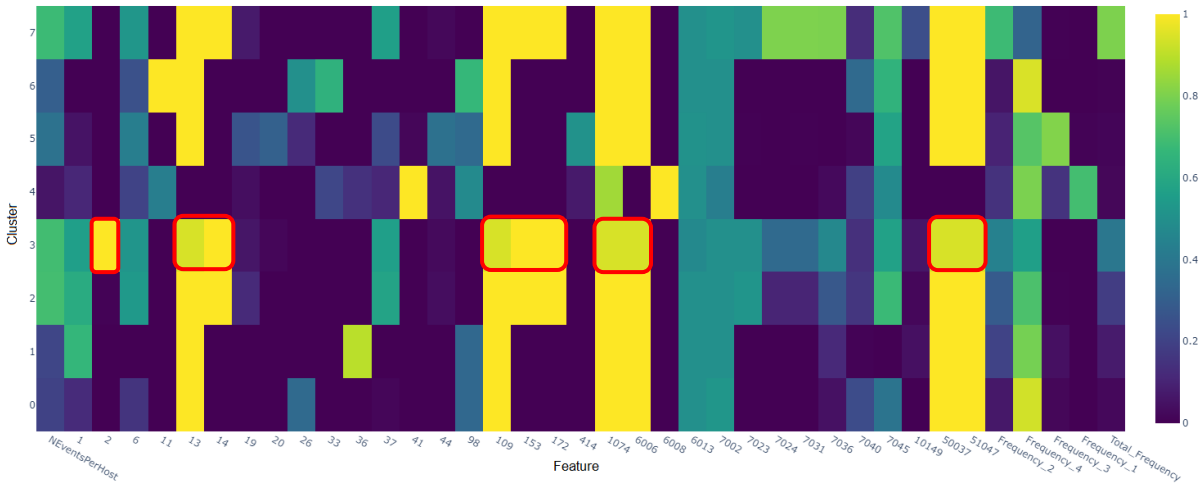

Figure 16 - Heat Map for the Agglomerative Clustering algorithm for x=100 and host data on 01-03-2018.

The frequency of MSWE with ID 2 appears abnormally high in cluster 3. This feature has a high prevalence in the differentiation of cluster 3. After analysing the details of the MSWE with ID 2 in the victims, all the events are from the Location Framework Service (lfsvc), have a priority of level 4 (information) and provide the message: “Geolocation positioning has been disabled by the user”.

The frequency of MSWE with ID 14 is also high; however, in this case, cluster 7 and cluster 2 also have a high frequency in this event. The MSWE with ID 14 originates in Wininit with the message: “Credential Guard configuration: 0x0, 0”. This feature is related to Windows Defender Credential Guard to protect the operating system from unauthorised access. The information portrayed in the event describes that the Credential Guard is disabled. Given that the Credential Guard is a protection layer to prevent attacks from stealing credential information, the MSWE with ID 14 might be relevant to determine if a machine is a victim.

The MSWE with ID 153 is also prevalent in cluster 3. This event has priority level 4 (information), is from Kernel-Boot and has the message: “Virtualization-based security (policies: 0) is disabled.” in every occurrence in the victim machines. This event means that virtualization-based security is disabled, and the assignment of value 0 to the policies field means that features like Credential Guard will also be disabled.

At last, the MSWE with ID 172 is also prevalent in selecting the elements of cluster 3. This event is related to the connectivity of the network interface card in low-power state mode. The information displayed in MS event viewer is that “Connectivity status in standby mode: Disconnected, Reason: NIC compliance”. This means that the network interface card has been disconnected due to compliance with power management policies. All the other features with high frequencies are present in the other clusters and, as such, are not as relevant as the ones mentioned.

The K-means algorithm's distribution of victims in the cluster continues to fall short of the desired results. However, the results improved, as can be seen in Figure 17. Cluster 1 contains 7 of the victims in a population of 45 elements. The other 3 victims are in cluster 7 with other 157 non-victim machines. Furthermore, cluster 0 has 67 elements, cluster 2 has 47 elements, cluster 3 has 32 elements, cluster 4 has 43 elements, cluster 5 has 22 elements, and cluster 6 has 7 elements.

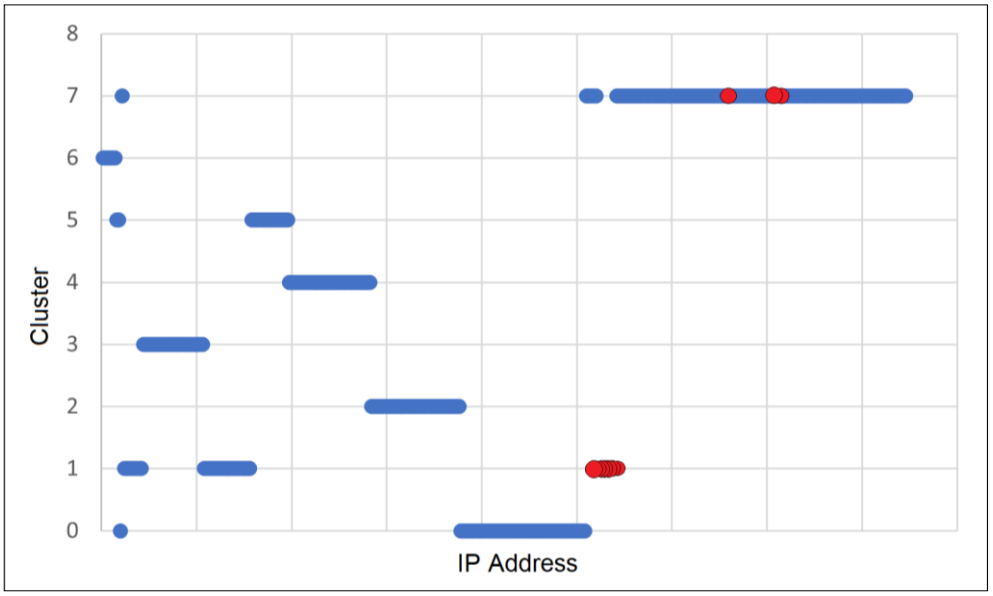

Figure 17 - Machine distribution by cluster with K-means algorithm for x=100 and host data on 02-03-18), with the victims represented in red.

The DBSCAN algorithm does not keep up with the improvements made by the K-means algorithm. As with the previous results, the DBSCAN algorithm placed every single victim in cluster 0 along with 194 other elements, as shown in Figure 18. Furthermore, cluster 1 has 208 elements, cluster 2 has 6 elements, and cluster -1 has 5 elements.

Figure 18 - Machine distribution by cluster with DBSCAN algorithm for x=100 and host data on 02-03-18), with the victims represented in red.

Figure 19 represents the clusters formed by the Agglomerative algorithm for a value of x = 100 and the distribution of the Windows machines for every cluster. This day's high number of victims makes selecting the cluster with the victims challenging. Without previous knowledge of the number of victims, deducing which cluster represents the victims is impossible. Cluster 3 has a population of 17 elements; cluster 4 has a population of 7. The high number of victims on the selected day is an obstacle for this type of approach to intrusion detection. Despite the abnormal behaviour from normal machines, the victims will be represented in a big cluster that the algorithm will interpret as a cluster of normal machines. Next, the proposed methods for outlier detection will be applied to try to overcome this challenge.

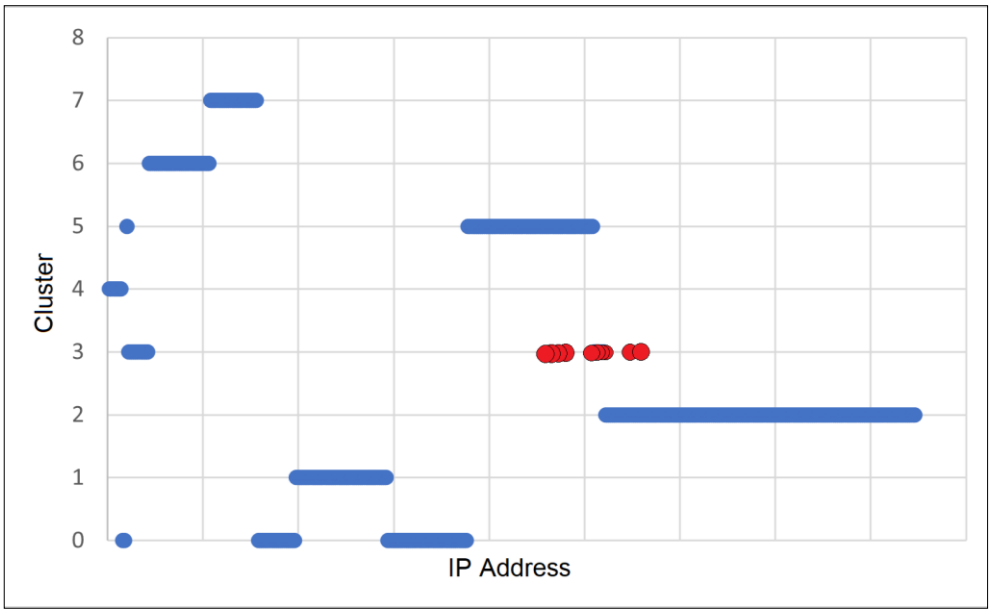

Figure 19 - Machine distribution by cluster with Agglomerative algorithm for x=100 and host data on 02-03-18, with the victims represented in red.

The previous subsections portray an evolution in the precision and accuracy of the results obtained by increasing the value of x. The results obtained for day 02-03-18 are represented in Table 15.

Table 15 - Summary of results for the clustering with features extracted from MSWE for the day 02-03-2018.

	K-means			DBSCAN			Agglomerative		
	Cluster	Total	Victims	Cluster	Total	Victims	Cluster	Total	Victims
x=20	1	57	5	-1	14	1	0	88	3
	3	26	2	7	135	7	2	27	2
	6	85	3	12	13	2	3	53	5
x=30	1	106	3	0	411	10	0	116	3
	3	73	5	-	-	-	2	27	3
	5	26	2	-	-	-	6	62	4
x=50	0	74	5	0	411	10	3	116	3
	2	105	3	-	-	-	4	74	5
	7	26	2	-	-	-	5	15	2
x=80	1	72	5	0	413	10	7	17	10
	5	106	3	-	-	-	-	-	-
	6	27	2	-	-	-	-	-	-
x=100	1	45	7	0	204	10	3	17	10
	7	164	3	-	-	-	-	-	-

Figure 20 is a graphical representation of the results achieved on 02-03-18. With the visual aid of the graph, is possible to verify that as the value of x increases, the number of clusters where the victims are placed decreases, and the percentage of victims in the clusters increases. Therefore, the ideal number of features is achieved at 100. Additional experiments were performed with higher values of x; however, the results did not improve.

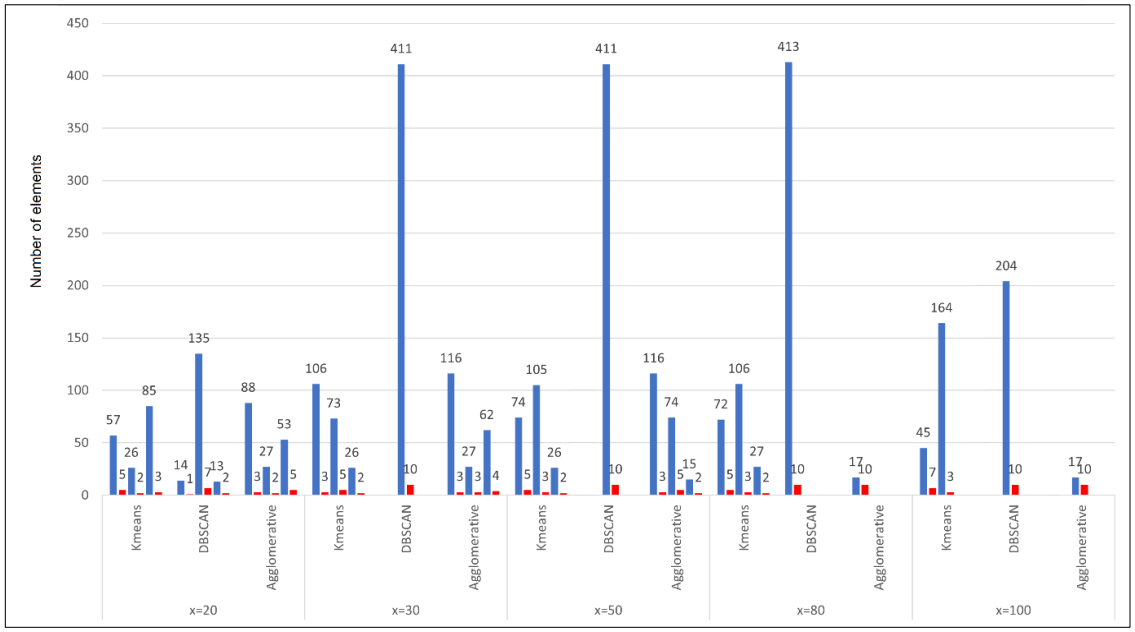

Figure 20 - Graph bar of the population of cluster where victims were placed for every algorithm and every value of x on 02-03-18.

5.2.6 Outlier Detection with host data and x=100 on 02-03-18 (BOT)

By applying the criteria defined in Subsection 4.2.4 for outlier detection, the evaluation metric can be calculated to measure the performance of the proposed system. The best results from the previous subsections were achieved for a value of x=100.

First, the machine scores method is applied, the second method described in Subsection 4.2.4, since there is no one-element cluster to consider necessary for the first method. This method aims to add a score for every machine proportional to the population of the cluster where the machine was placed for every algorithm. Therefore, the machines with unusually low scores would be considered outliers. Table 16 presents the scores and the corresponding rank for every victim on 02-03-18. The average score was 0.903554. Therefore, no victim complies with the requirements described in Subsection 4.2.4. Only one machine scores lower than 5% of the average; however, this machine is a false positive. The scores of the victims are presented in Table 16 and shows that no victim scores less than 5% of the average score (0.045178).

Table 16 - Scores of the victims on 02-03-18 with host data and x=100.

IP Address	Score	Rank
172.31.69.8	0.628842	15
172.31.6929	0.628842	16
172.31.69.10	0.628842	17
172.31.69.30	0.628842	18
172.31.69.6	0.628842	19
172.31.69.12	0.628842	20
172.31.69.17	0.628842	21
172.31.69.26	0.900709	255
172.31.69.14	0.900709	259
172.31.69.23	0.900709	260

With the second method, the machine scores method, the proposed approach achieves 7 false positives, 10 false negatives, 0 true positives and 396 true negatives.

Table 17 - Evaluation metrics for the second method on day 02-03-18, with host data and x = 100.

TP	0
FP	1
TN	396
FN	10
Accuracy	0.96
Precision	0
Recall	0
F1 score	0

The third method for selecting the victim's cluster consists of analysing the Silhouette coefficient of every cluster for every algorithm. Figure 21 presents the mean silhouette values for every cluster of the K-means algorithm.

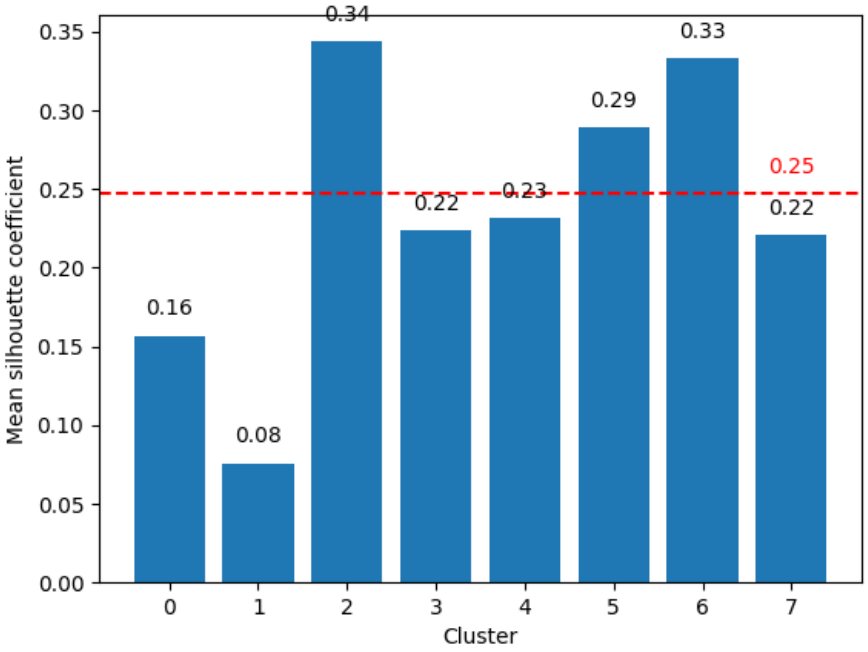

Figure 21 - Mean silhouette coefficient for the clusters in the K-means algorithm for x=100 and host data. The dashed red line represents the average of all the clusters. The value in red represents the average silhouette coefficient for every cluster.

In Figure 21, it is possible to verify that cluster 1 has a mean silhouette coefficient lower than half of the average of all the clusters, therefore complying with the criteria of Subsection 4.2.4. Cluster 1 has 7 victims and a population of 45 machines. Therefore, this method achieves 7 true positives, 38 false positives, 365 true negatives and 3 false negatives.

The DBSCAN algorithm places every victim in cluster 0 as shown in Table 15, with 194 other elements. There are no clusters that comply with the second criteria (mean silhouette coefficient half of the total mean silhouette coefficient). Figure 22 presents the mean silhouette coefficients for cluster 0,1 and 2 of the DBSCAN algorithm. Cluster -1 is not analysed because the elements of this cluster are machines that do not belong to any cluster. Therefore, the analysis of the cohesion of this cluster would not provide results.

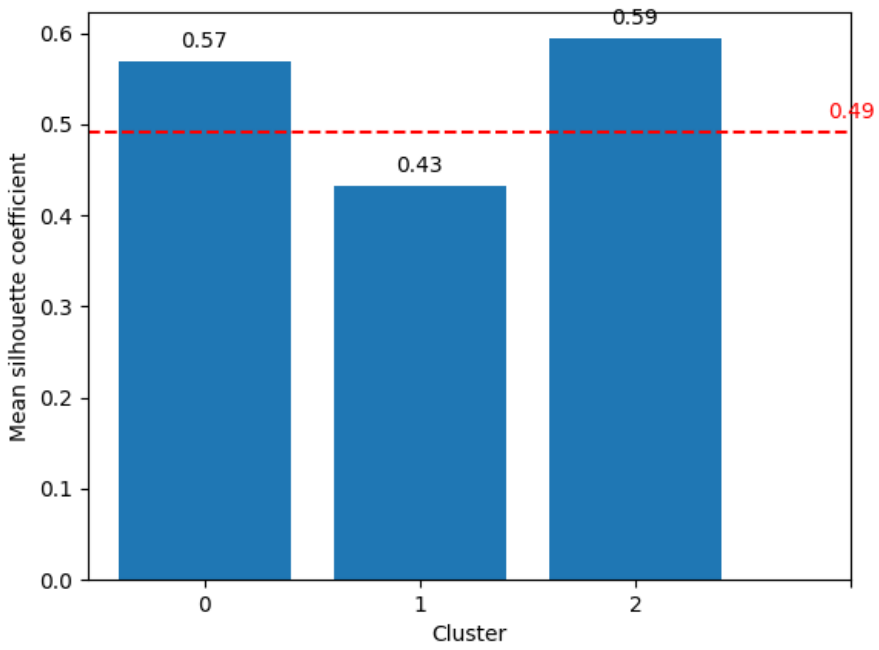

Figure 22 - Mean silhouette coefficient for the clusters in the DBSCAN algorithm for x=100 and host data. The dashed red line represents the average of all the clusters. The value in red represents the average silhouette coefficient for every cluster.

In contrast with the results obtained for the previous algorithms, the Agglomerative algorithm presents good results. As illustrated in Figure 23, Cluster 3 has a mean silhouette coefficient of 0.1 and well below half of the total average silhouette coefficient.

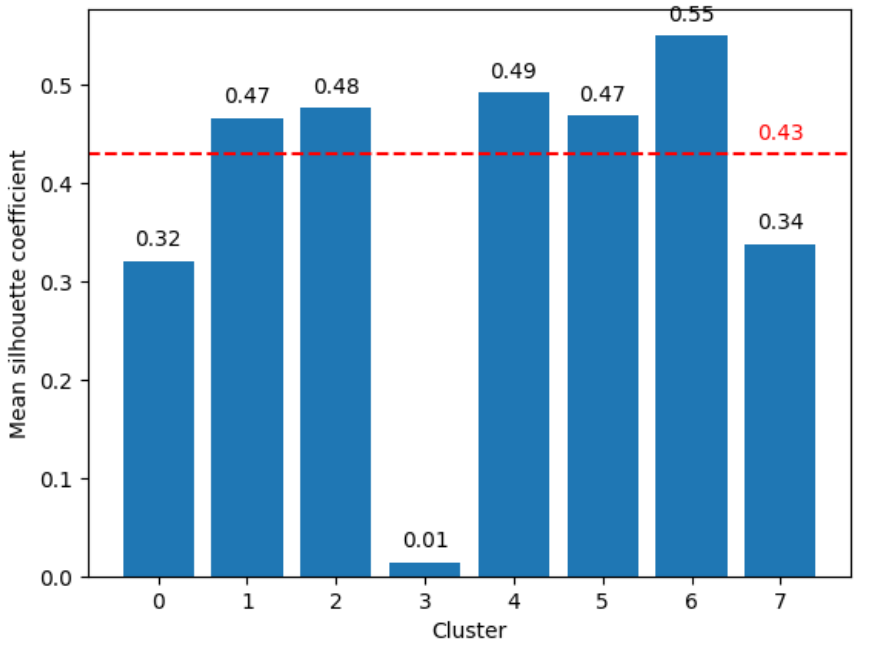

Figure 23 - Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100 and host data. The dashed red line represents the average of all the clusters.

Therefore, with the third criteria of Subsection 4.2.4 and the results in Figure 23 and Table 15, the only positively identified cluster is cluster 3. The results for the evaluation metrics for the second criteria and for day 02-03-18 are presented in Table 18.

Table 18 - Evaluation metric for the third criteria of Subsection 4.2.4, on 02-03-18 with host data.

	K-means	DBSCAN	Agglomerative
TP	7	0	10
FP	38	0	7
TN	365	403	396
FN	3	10	0
Accuracy	0.90	0.98	0.98
Precision	0.16	0	0.59
Recall	0.70	0	1
F1 score	0.26	0	0.74

The fourth and last method described in Subsection 4.2.4 considers every cluster with less than ten elements as a positive cluster. With this criterion, the K-means algorithm classifies cluster 6, which only has a population of 6 elements, as the positive cluster. The DBSCAN algorithm classifies cluster 2, which only has a population of 6 elements, and the elements of cluster -1 as outliers. Finally, the Agglomerative algorithm considers cluster 4, with a population of 7 machines, the positive cluster with this criterion. The evaluation metrics for these results are presented in Table 19.

Table 19 - Evaluation metric for the fourth criteria of Subsection 4.2.4, on 02-03-18 with host data.

	K-means	DBSCAN	Agglomerative
TP	0	0	0
FP	6	11	7
TN	397	392	396
FN	10	10	10
Accuracy	0.96	0.95	0.96
Precision	0	0	0
Recall	0	0	0
F1 score	0	0	0

5.2.7 Infiltration (01-03-18), with host data and x=100

In this subsection, the results achieved for the BOT attack will be applied for the infiltration attack of day 01-03-2018. In an infiltration attack, the attacker gains unauthorised access to a vulnerable application. In the specific case of the attack in focus, the victim received a malicious document through email [47]. The authors used the Metasploit framework to execute a backdoor in the victim's computer, where it is possible to launch a diverse set of attacks.

In contrast with day 02-03-18, the attacks of day 01-03-18 only targeted one victim. Firstly, there is the need to analyse the behaviour of this victim if the conditions that provided the best results in the last subsection are applied. Therefore, the starting conditions will be x =100 and the 5 fixed features (Frequency_1, Frequency_2, Frequency_3, Frequency_4, Frequency_5 and Total_Frequency. Table 20 presents the IDs of dynamically extracted MSWE.

Table 20 - IDs of the MSWE selected for x=100 and host data on 01-03-18. In blue are represented the MSWE IDs most frequent. In green are represented the MSWE IDs most frequent and used by less than 10 machines. In orange are represented the MSWE IDs least frequent.

MSWE ID										
7036	7031	7024	36888	6	7023	6013	98	7040	37	414
16	20	1	44	12	7001	27	7026	51046	50036	55
32	6005	18	6009	7002	1074	109	50037	6006	13	51047
1111	36874	258	257	265	7042	7032	1501	7034	4	10400
104	1001	2012	4202	-	-	-	-	-	-	-
35	10148	25	10016	10010	14	172	153	7009	7000	26
134	2	1111	41	10149	6008	7022	1014	36874	258	257
265	7042	7032	1501	7034	4	10400	104	1001	2012	4202

The victim is placed in cluster 3 by the K-means algorithm. This cluster has 12 total elements, one of whom is the victim. Cluster 7 is a one-element cluster. Cluster 8 has a population of 11 machines. Cluster 6 has a population of 17 machines. Cluster 4 has a population of 41 machines. Cluster 5 has a population of 57 machines. Cluster 2 has a population of 66 machines. Cluster 0 has a population of 75 machines. Finally, cluster 1 has a population of 133 machines. Figure 24 shows the K-means algorithm's distribution of machines in the clusters.

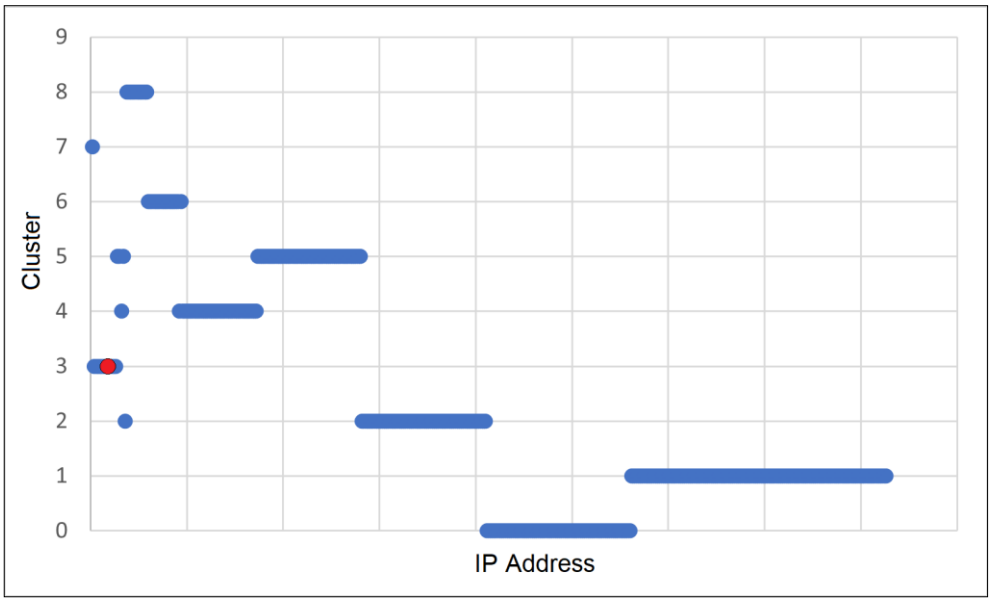

Figure 24 - Machine distribution by cluster with K-means algorithm for x=100 and host data on 01-03-18, with the victim represented in red.

In contrast with the K-means algorithm, the algorithm DBSCAN classified the victim as an outlier. The victim is placed in cluster -1 with six other machines. This cluster agglomerates the outlier machines that did not belong to any specific cluster. Figure 25 presents the distribution of machines in the clusters formed by the DBSCAN.

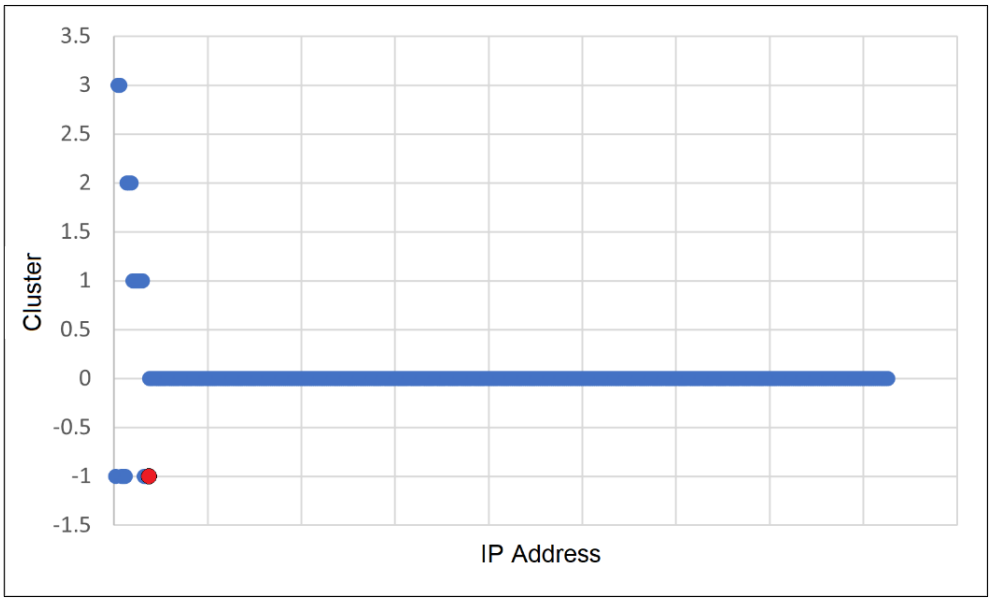

Figure 25 - Machine distribution by cluster with DBSCAN algorithm for x=100 and host data on 01-03-18, with the victim represented in red.

The Agglomerative algorithm produced similar results to the K-means algorithm and placed the victim in cluster 2 with 11 other machines. As with the results from K-means, cluster 2 is not the least populated cluster. Cluster 7 of the Agglomerative algorithm has only 3 elements. Figure 26 presents the machine distribution by the clusters obtained for the Agglomerative algorithm.

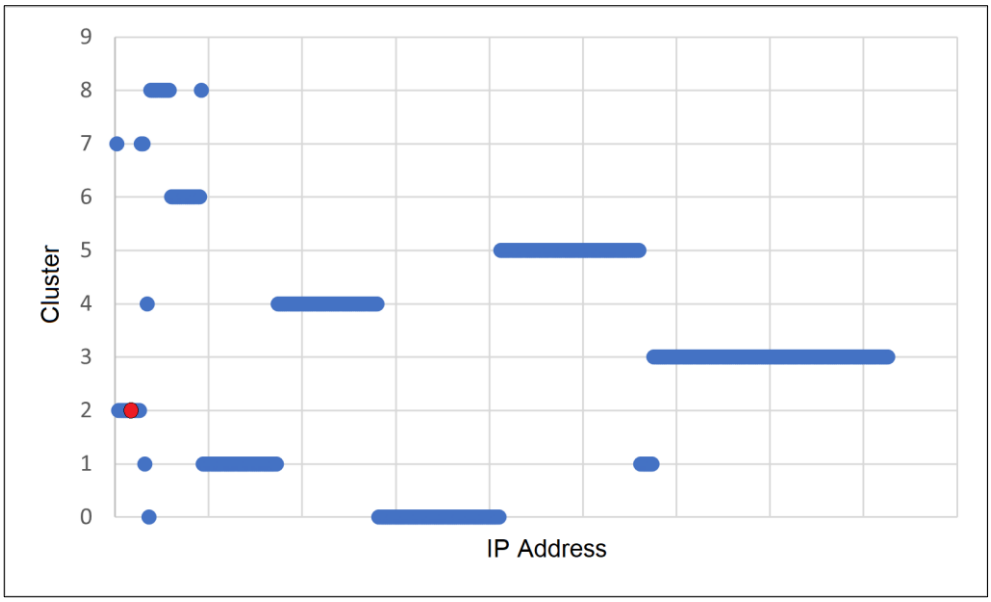

Figure 26 - Machine distribution by cluster with Agglomerative algorithm for x=100 and host data on 01-03-18, with the victim represented in red.

5.2.8 Outlier Detection with host data and x=100 on 01-03-18 (Infiltration)

In this subsection the results obtained for the day 01-03-18 with host data will be evaluated according to the previously defined evaluation metrics and the outlier criteria specified in Subsection 4.2.4. Even though there is only one victim on this day, the same methods established in Subsection 4.2.4 will be applied, given that, in real conditions, the system does not know how many victims are in the network.

The first method is the one-element cluster method. This method is the most straight forward and consists in selecting clusters with a population of one. By analysing Figure 24, Figure 25 and Figure 26 the conclusion can be made that only the K-means and the DBSCAN algorithms produced one-element clusters. K-means in cluster 7 and DBSCAN in each one of the 7 elements that constitute cluster -1. The evaluation metrics for this first method are presented in Table 21.

Table 21 - Evaluation metric for the first criteria of Subsection 4.2.4, on 01-03-18 with host data.

	K-means	DBSCAN	Agglomerative
TP	0	1	0
FP	1	6	0
TN	411	406	412
FN	1	0	1
Accuracy	1	0.99	1
Precision	0	0.14	0
Recall	0	1	0
F1 score	0	0.25	0

The second method described in Subsection 4.2.4 is the machine scores method. This method consists of giving every machine a score for every algorithm directly proportional to the population of the cluster where the machine was placed. The machines which achieved a score below 5% of the average score (the average score is 1.295476 and 5% is 0.064774) are presented in Table 22 in blue shade.

Table 22 - Scores for the machines on 01-03-18, with host data. The machines with blue background achieved a score below 5% (0.064774) of the average score.

IP Address	K-means	DBSCAN	Agglomerative	Score	Rank
172.31.67.116	7	-1	7	0.009685	1
172.31.69.13	6	-1	6	0.058111	2
172.31.69.126	3	-1	2	0.058111	3
172.31.67.77	3	-1	2	0.058111	4
172.31.64.36	3	2	2	0.065375	5
172.31.65.11	3	2	2	0.065375	6
173.31.67.8	3	2	2	0.065375	7
172.31.64.47	3	1	2	0.072639	8
172.31.65.52	3	1	2	0.072639	9
172.31.64.14	3	1	2	0.072639	10

These results allow the calculation of the evaluation metrics, that are presented in Table 23.

Table 23 - Evaluation metric for the second criteria of Subsection 4.2.4, on 01-03-18 with host data.

TP	1
FP	4
TN	408
FN	0
Accuracy	0.99
Precision	0.2
Recall	1
F1	0.33

The Silhouette coefficient method is the third method described in Subsection 4.2.4. Figure 27 presents the mean silhouette coefficient for the K-means algorithm, where the cluster of the victim is cluster 3. The only cluster that meets the criteria for an outlier classification is cluster 7.

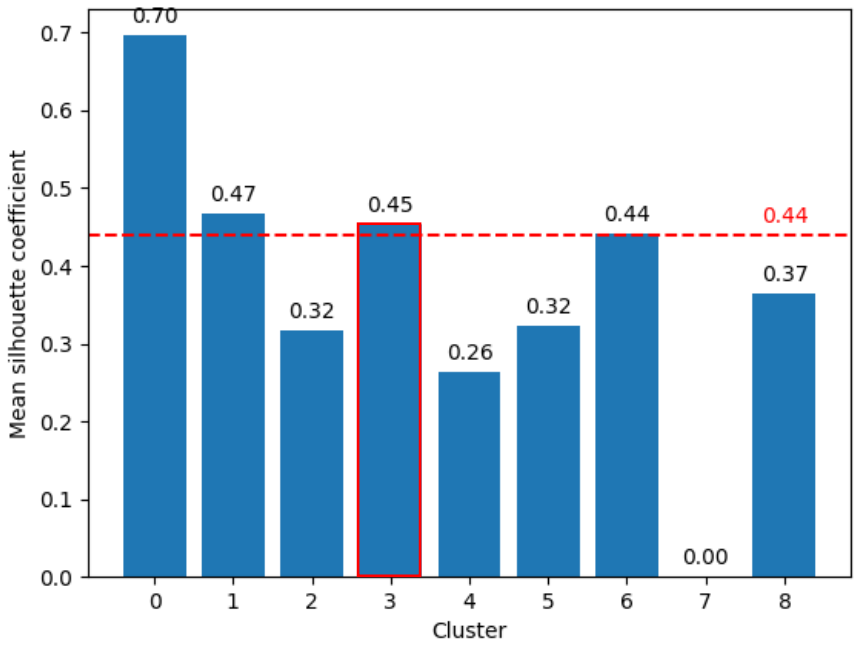

Figure 27 - Mean silhouette coefficient for the clusters in the K-means algorithm for x=100 and host data on 01-03-18. The dashed red line represents the average of all the clusters. The cluster where the victim was placed is highlighted in red.

The DBSCAN algorithm places the victim in cluster -1. Thus, classifying the victim as an outlier. Furthermore, by the silhouette method describe in the Subsection 4.2.4, no cluster is classified as a positive occurrence. Figure 28 illustrates the means silhouette coefficients for every cluster of the

DBSCAN algorithm. Cluster -1 is not represented since measuring the silhouette coefficient of this cluster is not relevant. None of the represented clusters fulfils the criteria of the third method described in Subsection 4.2.4.

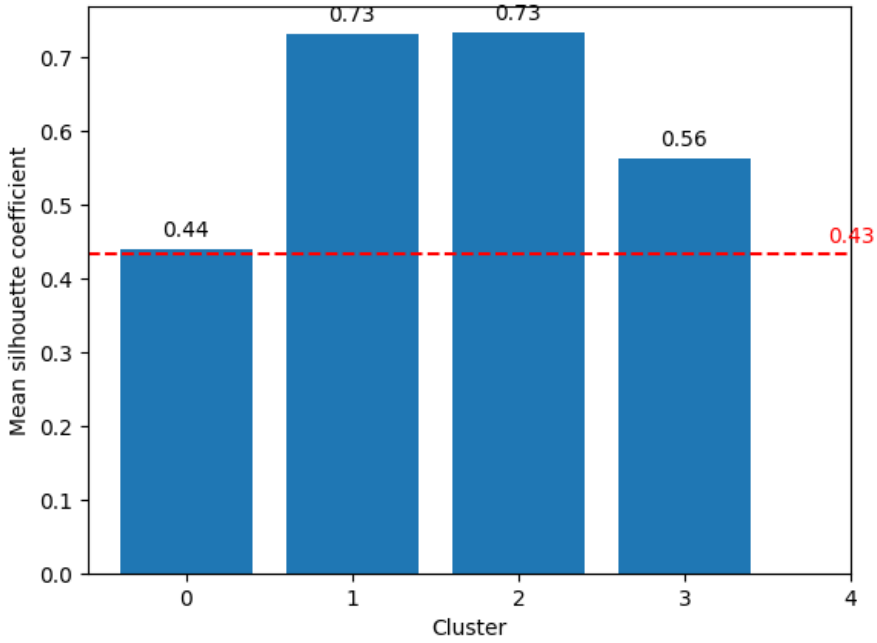

Figure 28 - Mean silhouette coefficient for the clusters in the DBSCAN algorithm for x=100 and host data on 01-03-18. The dashed red line represents the average of all the clusters.

Figure 29 presents the mean silhouette coefficient for the clusters of the Agglomerative algorithm. The only cluster that fulfils the criteria defined in Subsection 4.2.4 is cluster 1. The victim's cluster is highlighted in red, with a mean silhouette coefficient above the overall mean.

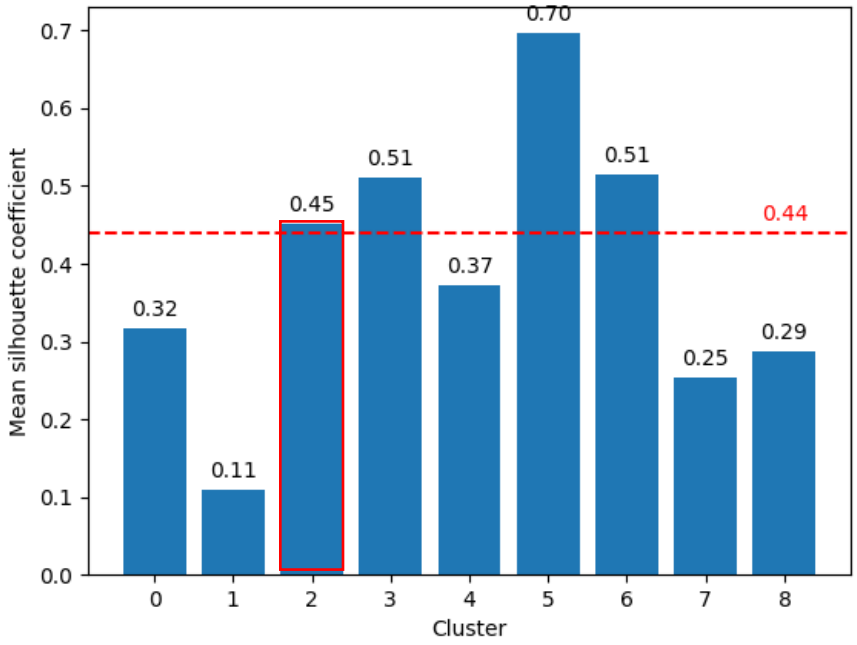

Figure 29 - Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100 and host data on 01-03-18. The dashed red line represents the average of all the clusters. The cluster where the victim was placed is highlighted in red.

Table 24 presents the results achieved by applying the third criteria. None of the three algorithms was able to detect a true positive. Therefore, only the accuracy can be measured for this method.

Table 24 - Evaluation metric for the third criteria of Subsection 4.2.4, on 01-03-18 with host data.

	K-means	DBSCAN	Agglomerative
TP	0	0	0
FP	1	0	48
TN	411	412	364
FN	1	1	1
Accuracy	1	1	0.88
Precision	0	0	0
Recall	0	0	0
F1	0	0	0

The fourth and last method described in Subsection 4.2.4 is the low population method. This method consists of considering every cluster with less than ten elements as positive occurrences. Figure 24, Figure 25 and Figure 26 allow us to determine the number of false positives, false negatives, true negatives and true positives necessary to calculate the evaluation metrics presented in Table 25.

The K-means algorithm has only one cluster with a population lower than ten elements. This cluster is cluster 7 which is a one-element cluster. The DBSCAN algorithm has three clusters plus the

population of cluster -1. These machines add to eighteen. The Agglomerative algorithm only has cluster 7 with a population lower than ten.

Table 25 - Evaluation metric for the fourth criteria of Subsection 4.2.4, on 01-03-18 with host data

	K-means	DBSCAN	Agglomerative
TP	0	1	0
FP	1	17	3
TN	411	395	409
FN	1	0	1
Accuracy	1	0.96	0.99
Precision	0	0.06	0
Recall	0	1	0
F1	0	0.11	0

5.2.9 Summary of results with host data

In summary, the methods described in Subsection 4.2.4 did not achieve the desired performance. On day 02-03-18, the first, second, and fourth methods were unable to produce any true positive. The third method also performed poorly for the DBSCAN algorithm. In contrast, the metrics for the K-means and Agglomerative algorithms are closer to similar approaches. The Agglomerative algorithm, in conjunction with the third method, achieved the best results with an accuracy of 0.98, a precision of 0.59, a recall of 1, and an F1 score of 0.74. However, these results still fall short of expectations. Figure 30 illustrates the results achieved by every method on 02-03-18.

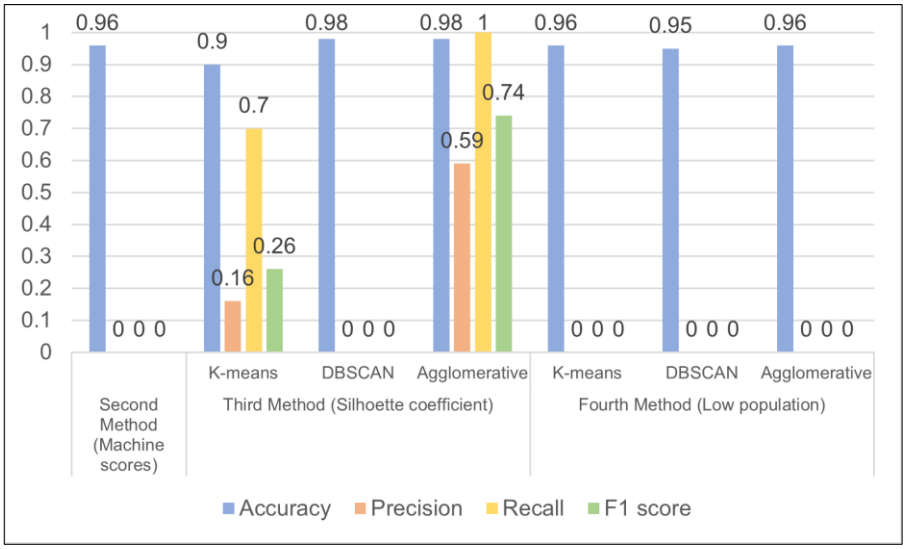

Figure 30 - Evaluation metrics for every method described in Subsection 4.2.4 on 02-03-18 for host data. Method 2 is scoring the machines proportionally to the population of its cluster in all algorithms. Method 3 is analysis of the silhouette coefficients of every cluster. Finally, method 4 is the consideration of any cluster with less than ten elements as a positive occurrence.

Figure 31 compares the evaluation metrics for all the methods enunciated in Subsection 4.2.4 on 01-03-18. DBSCAN was the algorithm that performed the best, while the other algorithms could not detect any true positive. Compared with the results of the attack on 02-03-18, the results on 01-03-18 are considerably worse. The best evaluation metrics were obtained for the second method and the DBSCAN algorithm with an accuracy of 0.99, a precision of 0.2, a recall of 1, and an F1 score of 0.33. These metrics represent a decrease of 66% in precision and 55% in F1 score compared to the best-performing combination on 02-03-18. However, these results are poor when compared with similar approaches. The host features considered are not sufficient to identify the victims of attacks consistently and precisely. Therefore, in the following section, the system will be evaluated by adding network features.

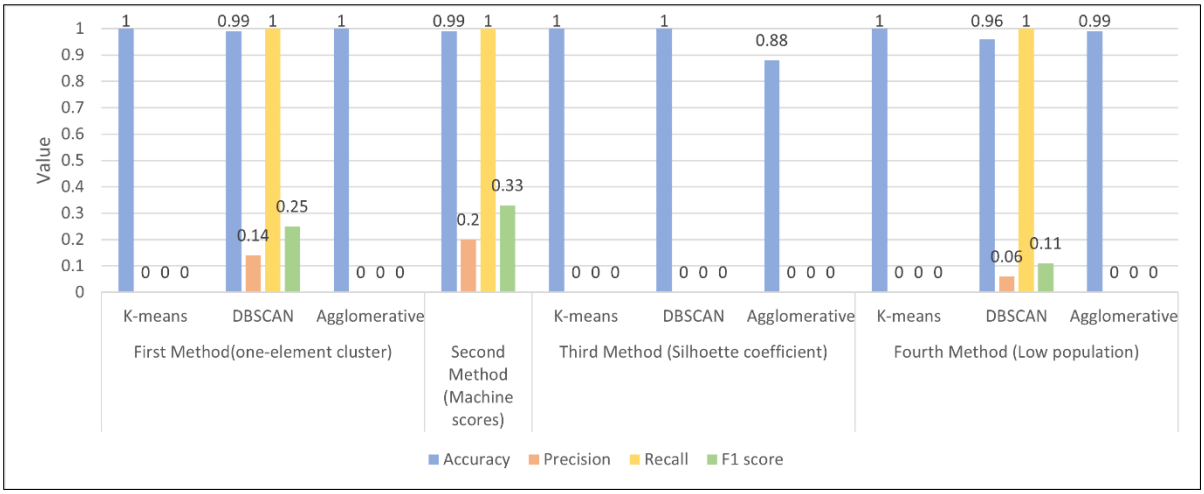

Figure 31 - Evaluation metrics for every method described in Subsection 4.2.4 on 01-03-18 with host data. Method 1 is the selection of one-element clusters. Method 2 is scoring the machines proportionally to the population of its cluster in all algorithms. Method 3 is analysis of the silhouette coefficients of every cluster. Finally, method 4 is the consideration of any cluster with less than ten elements as a positive occurrence.

5.3 Evaluation of the system considering host and NetFlow data

In this section, the study of the impact of the addition of NetFlow data to the MSWELs. The two sets of data were merged based on the IP Address and then the clustering was applied. The objective is the improvement of the results and the isolation of the entire range of victims in a single cluster. This section presents the results achieved by the full implementation of the proposed system.

5.3.1 BOT (02-03-18), with host, network data and X=100

In the previous section, the clustering was executed with only MSWEs information. The features selected can be divided in dynamically defined features, the event IDs, and fixed features, NEventPerHost, Frequency_1, Frequency_2, Frequency_3, Frequency_4, Frequency_5, Total_Frequency. In this subsection, the features will be the same plus the features from the NetFlow extracted data.

The following Table 26 represents the cluster where each victim was placed by each algorithm. The results demonstrate that in the K-means, 9 of the victims were placed in cluster 7 with no other machines. The other victim is placed in cluster 2 with 38 other machines. The DBSCAN algorithm placed

every victim in cluster 0. The Agglomerative algorithm placed 8 of the 10 victims in cluster 7 and the other 2 victims in cluster 2.

Table 26 - Cluster results for each victim and algorithm for x=100, host and network data on 02-03-18.

Victim	K-means	DBSCAN	Agglomerative
172.31.69.6	7	0	7
172.31.69.8	7	0	2
172.31.69.10	7	0	7
172.31.69.12	2	0	2
172.31.69.14	7	0	7
172.31.69.17	7	0	7
172.31.69.23	7	0	7
172.31.69.26	7	0	7
172.31.69.29	7	0	7
172.31.69.30	7	0	7

At first glance, the results seem to be poorer when compared with the results obtained in the previous section since, in the last section, the Agglomerative algorithm placed all the victims in the same cluster. However, with a closer look at the results, it is possible to see that is not the case.

The differentiating features of cluster 7 seem to be mostly from the host data. The features with most weight are the MSWEs with IDs: 2, 14, 153, 172. These events are highlighted in red in Figure 32 and are the same events analysed in Subsection 5.2.5 Furthermore, the features SrcPortUsed, SrcConnMade and 8080SrcFrom also have some prevalence in selecting the victims in cluster 7. These features are highlighted in blue in Figure 32.

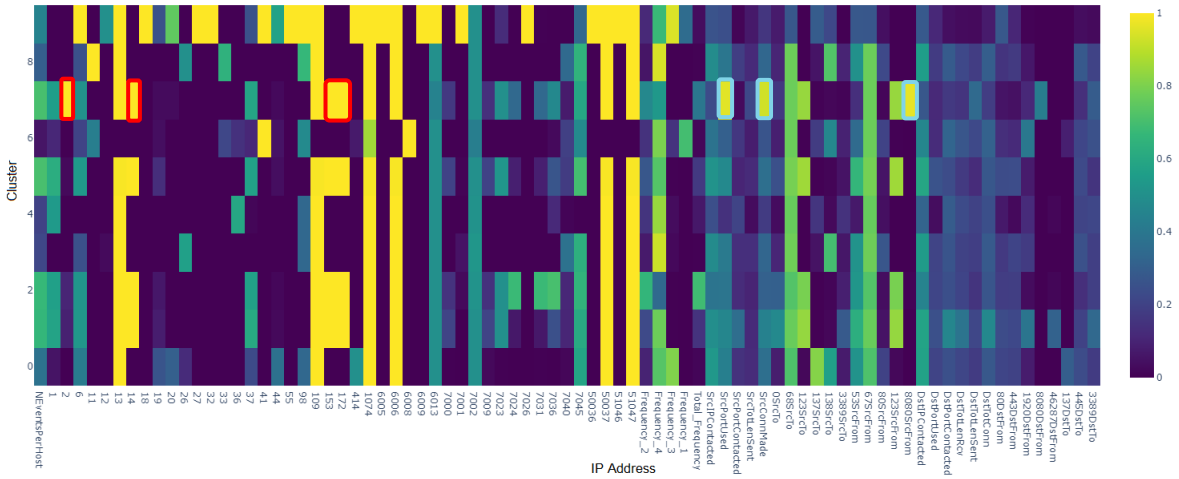

Figure 32 - Heat map of K-means algorithm on 02-03-2018, X=100 with host and network data.

Cluster 9 is a one-element cluster and, as such, the cluster with the least population. Cluster 6 has a population of 7 machines. Cluster 7 has a population of 9 victims. Cluster 1 has a population of 22 machines. Cluster 8 has a population of 32 machines. Cluster 2 has a population of 39 machines, the same as cluster 3. Cluster 0 has 66 elements. Cluster 4 has a population of 73. Finally, Cluster 5 is the most populated cluster with 135 machines. Figure 33 presents the distribution mentioned before by the K-means algorithm.

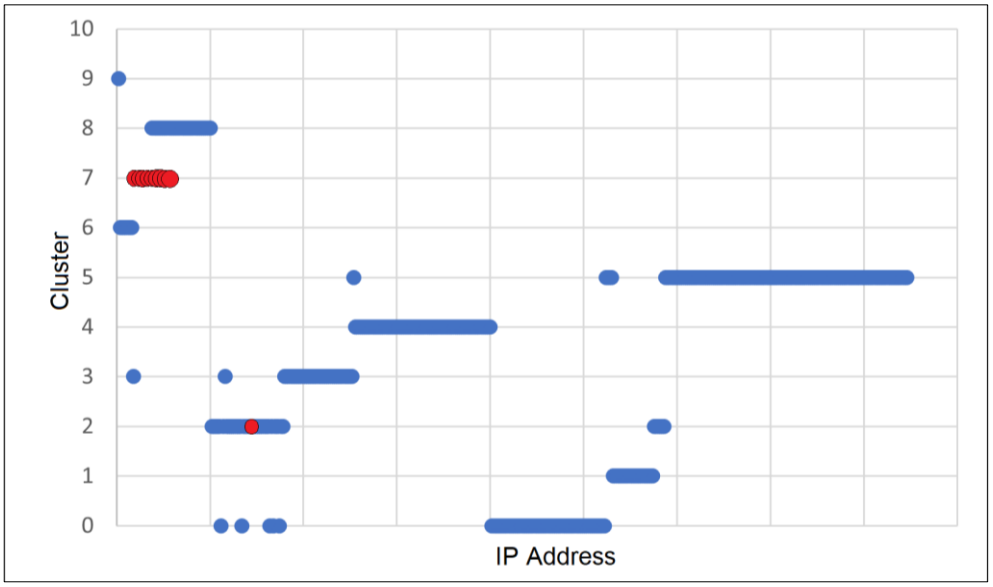

Figure 33 - Machine distribution by cluster with K-means algorithm for x=100 and host data and network data on 02-03-18, with the victims represented in red.

In contrast with the results obtained by the K-means algorithm, DBSCAN did not demonstrate an improvement over the results obtained in the previous section. As presented in Figure 34 all the victims are placed in cluster 0 with 194 other machines.

Figure 34 - Machine distribution by cluster with DBSCAN algorithm for x=100, host data and network data, on 02-03-18, with victims represented in red.

The results of the Agglomerative algorithm are similar to the results of the K-means algorithm, however there are only 8 victims placed in cluster 7, which only has victims in its population, instead of 9 as in the K-means case. One of the victims that is not placed in this cluster is the same that was not present in cluster 7 of the K-means algorithm and is placed in cluster 2 with one other victim and 36 other machines. The following Figure 35 presents the distribution of machines by cluster with the Agglomerative algorithm. Cluster 0 has 159 elements. Cluster 1 has 38 elements. Cluster 2 has 38 elements. Cluster 3 has 32 elements. Cluster 4 has 61 elements. Cluster 5 has 6 elements. Cluster 6 has 23 elements. Cluster 8 has 41 elements. Finally, cluster 9 has 7 elements.

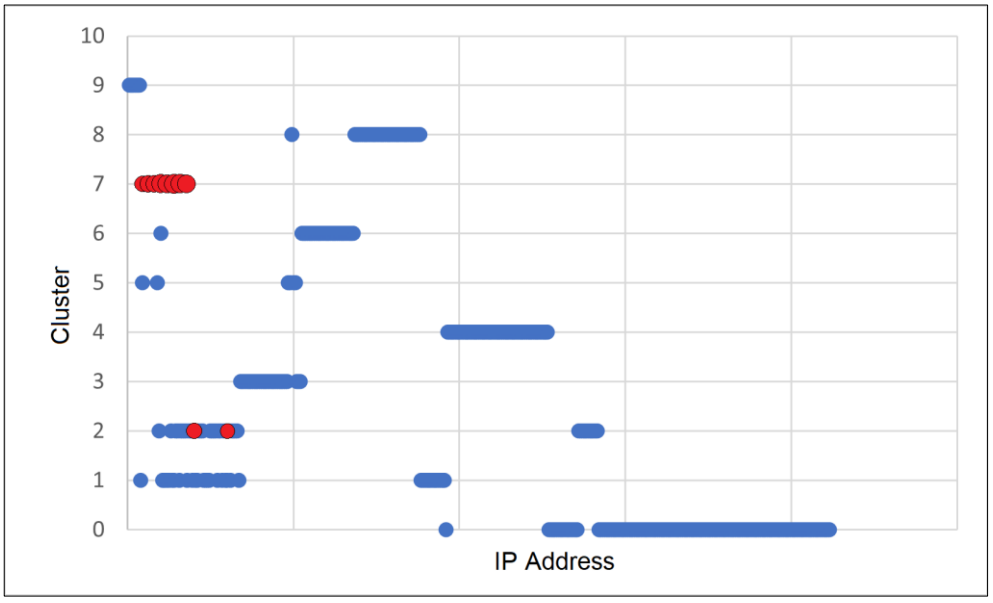

Figure 35 - Machine distribution by cluster with Agglomerative algorithm for x=100, host data and network data, on 02-03-18, with victims represented in red.

5.3.2 Outlier detection with host, network data and x=100 on 02-03-18 (BOT).

In this section the system is evaluated with host and network data for the attacks on 02-03-18. Both K-means and the Agglomerative algorithms performed well. K-means placed 9 of the victims in a single cluster without any other non-victim machine, and the Agglomerative algorithm was able to place 8 of the victims in a single cluster without any other non-victim machine. However, the algorithm still needs to identify the mentioned clusters as victim clusters. To achieve this, the four methods described in Subsection 4.2.4 will be implemented.

The first method consists in considering as a positive occurrence one-element clusters. By analysing Figure 33, Figure 34 and Figure 35, the number of one-element clusters produced by each algorithm can be determined. The results for the evaluation metrics are summarized in Table 27.

Table 27 - Evaluation metric for the first criteria of Subsection 4.2.4, on 02-03-18 with host and network data.

	K-means	DBSCAN	Agglomerative
TP	0	0	0
FP	1	3	0
TN	412	410	413
FN	10	10	10
Accuracy	0.97	0.97	0.98
Precision	0	0	0
Recall	0	0	0
F1	0	0	0

The second method described in Subsection 4.2.4 consists of selecting the clusters with a score lower than 5% of the average score of the entire population. In the case in focus on this subsection, the average score is 0.859391. Therefore, the machines with a score below 0.042970 will be selected. With this criterion, only one non-victim machine is selected with a score of 0.016548. The scores of the victims are summarized in Table 28.

Table 28 - Scores of the victims on 02-03-18 with host and network data and x=100.

IP Address	Score	Rank
172.31.69.8	0.522459	10
172.31.6929	0.522459	11
172.31.69.10	0.522459	12
172.31.69.30	0.522459	13
172.31.69.6	0.522459	14
172.31.69.12	0.522459	15
172.31.69.17	0.522459	16
172.31.69.26	0.522459	17
172.31.69.14	0.593381	18
172.31.69.23	0.664303	72

Therefore, this method only produces one false positive, achieving the evaluation metrics presented in Table 29.

Table 29 - Evaluation metric for the second criteria of Subsection 4.2.4, on 02-03-18 with host and network data.

TP	0
FP	1
TN	396
FN	10
Accuracy	0.96
Precision	0
Recall	0
F1 score	0

The third method described in Subsection 4.2.4 consists of selecting the cluster with a mean Silhouette coefficient lower than half of the total mean value of the silhouette coefficient. Figure 36 illustrates the mean silhouette coefficients for every cluster produced by the K-means algorithm. There are three clusters that meet the requirements. Cluster 1, cluster 2 and cluster 3 all have a mean silhouette coefficient lower than half of the average value.

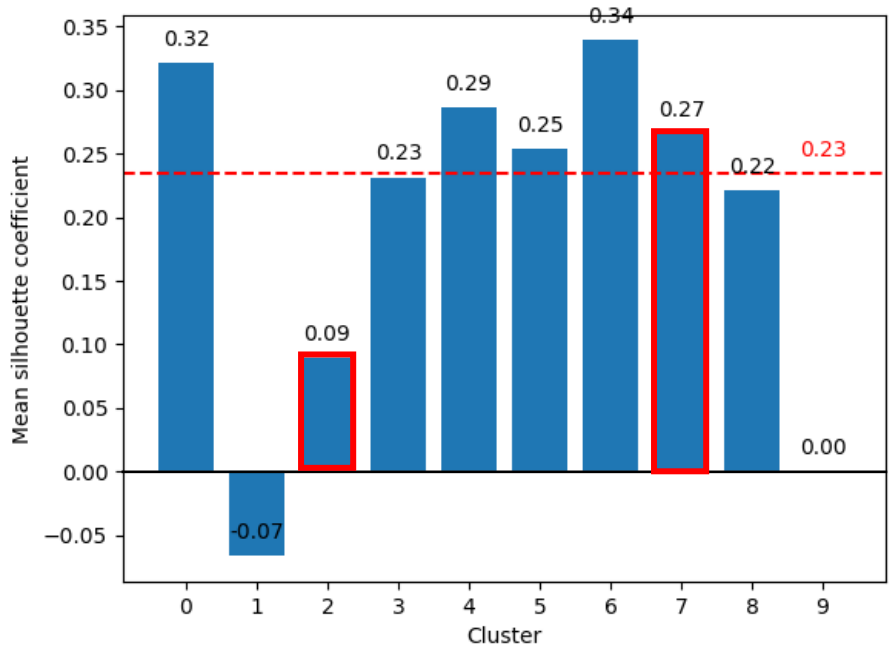

Figure 36 - Mean silhouette coefficient for the clusters in the K-means algorithm for x=100, host and network data on 02-03-2018, with host and network data. The dashed red line represents the average of all the clusters. The clusters with victims are highlighted in red.

Figure 37 presents the mean silhouette coefficients for the three clusters of the DBSCAN. Cluster -1 is a set of elements that do not belong together. Therefore, the analysis of the silhouette coefficient is not useful for this specific cluster. As shown in Figure 37, both cluster 1 and cluster 2 meet the requirements established for the third method.

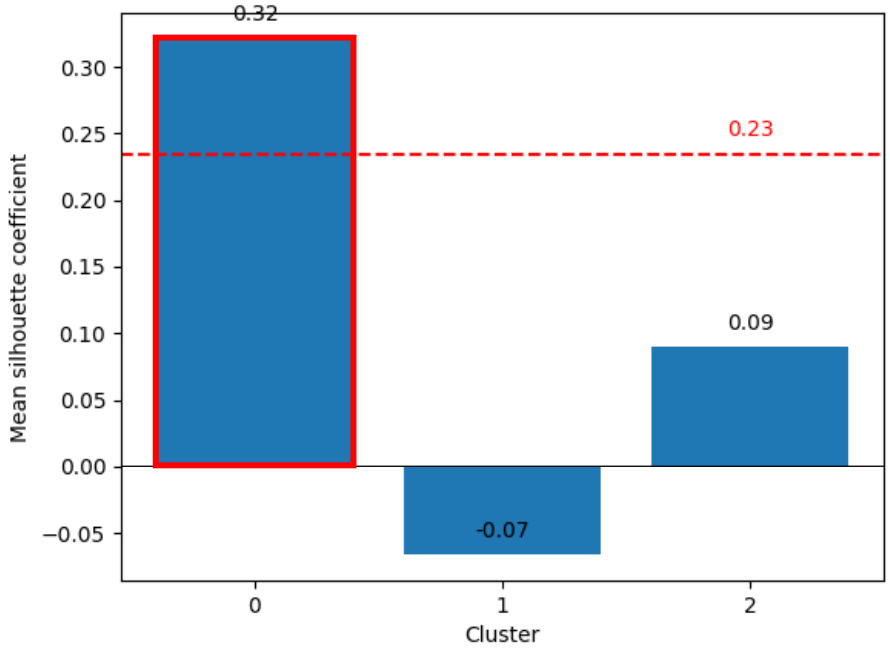

Figure 37 - Mean silhouette coefficient for the clusters in the DBSCAN algorithm for x=100, host and network data on 02-03-2018. The dashed red line represents the average of all the clusters. The clusters with victims are highlighted in red.

Figure 38 presents the mean silhouette coefficients for the clusters produced by the Agglomerative algorithm. The only cluster that fulfils the requisites of the third method described in Subsection 4.2.4 is cluster 2. This cluster has two victims in its constitution and thirty-six non-victim machines.

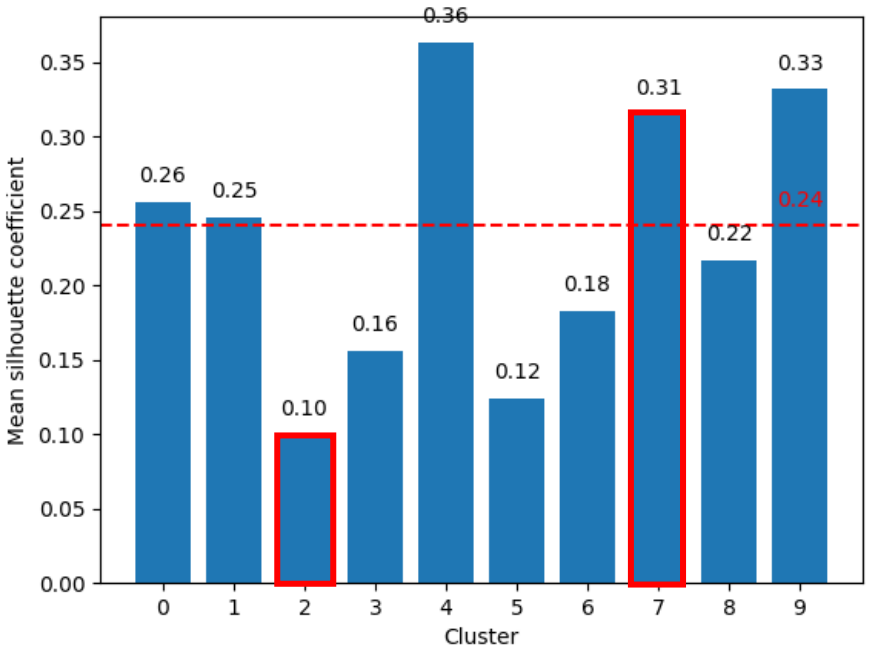

Figure 38 - Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100, on 02-03-2018, with host and network data. The dashed red line represents the average of all the clusters. The clusters with the victims are highlighted in red.

Table 30 presents the results and the evaluation metrics achieved for this method.

Table 30 - Evaluation metric for the third criteria of Subsection 4.2.4, on 02-03-18 with host and network data.

	K-means	DBSCAN	Agglomerative
TP	1	0	2
FP	99	216	36
TN	314	197	377
FN	9	10	8
Accuracy	0.74	0.47	0.90
Precision	0.01	0	0.05
Recall	0.10	0	0.2
F1	0.02	0	0.08

The fourth and final method defined in Subsection 4.2.4 consists of selecting the cluster with less than ten elements as positive occurrences. Figure 33, Figure 34, and Figure 35 allow us to observe these occurrences. The evaluation metrics for this last method are presented in Table 31.

Table 31 - Evaluation metric for the fourth criteria of Subsection 4.2.4, on 02-03-18 with host and network data.

	K-means	DBSCAN	Agglomerative
TP	9	0	8
FP	8	10	13
TN	405	403	400
FN	1	10	2
Accuracy	0.98	0.95	0.96
Precision	0.53	0	0.38
Recall	0.90	0	0.80
F1	0.67	0	0.52

5.3.3 Infiltration (01-03-18), with host, network data and x=100

On this day there is only one victim. Therefore, it is expected that a cluster with only one element created by the algorithms indicates an outlier. The analyses of the results obtained by the clustering of the K-means algorithm show that the victim is placed in a single element cluster. As it is possible to verify in Figure 39 the victim is placed in cluster 7 that contains only one element. Therefore, the victim's cluster is the least populated cluster produced by the K-means algorithm. Cluster 6 has 11 elements. Cluster 5 has 69 elements. Cluster 4 has 29 elements. Cluster 3 has 145 elements. Cluster 2 has 29 elements. Cluster 1 has 75 elements. Finally, Cluster 0 has 54 elements.

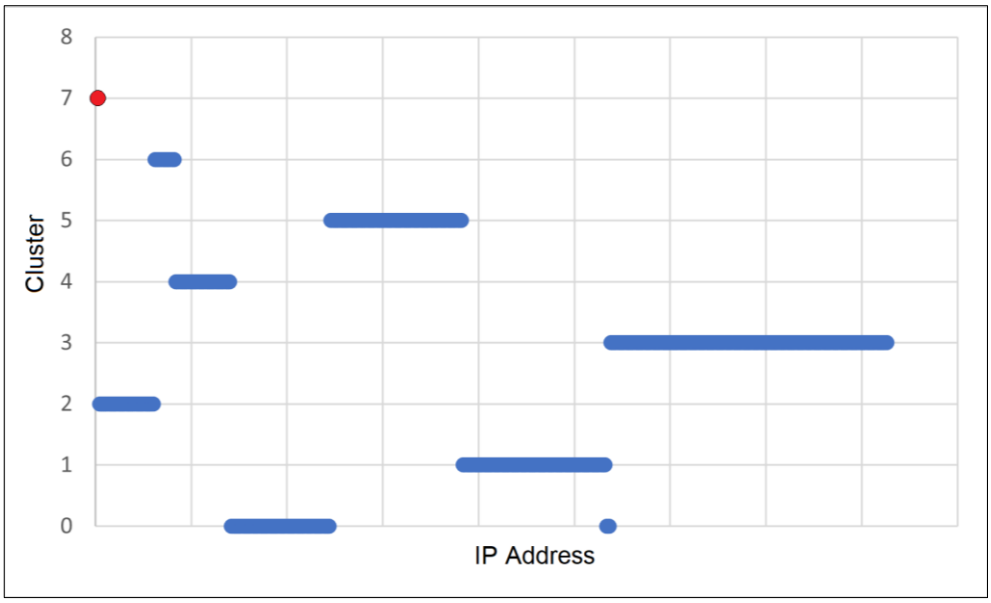

Figure 39 - Machine distribution by cluster with K-means algorithm for x=100, host data and network data, on 01-03-18, with victims represented in red.

These results correspond to the initial objective of this work. The victim was placed by every algorithm in an outlier cluster. From this result it is easy to flag machines that are victims by simply flagging one-element clusters. The features that lead to this result can be observed in the following heat maps. Figure 40 presents the heat map for the K-means algorithm. By a detailed analysis of the heat map one can see that most features that differentiate cluster 7 from the others are from the NetFlow data. However, there are also features from the MSWELs that differentiate cluster 7 from the other clusters. More precisely, the MSWE with ID 104 marked in Figure 40 with a red rectangle, has a high normalised value when compared to every other cluster.

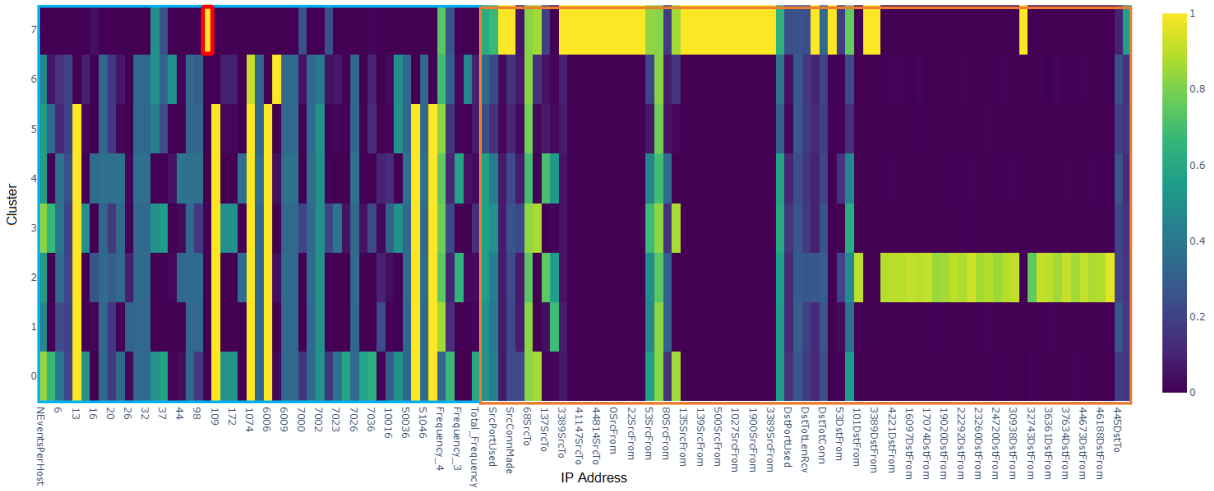

Figure 40 - Heat map of K-means algorithm on 01-03-2018, X=100 with host and network data. Inside the blue rectangle, MSWE features. Inside the orange rectangle, NetFlow features.

Likewise, the results of the DBSCAN algorithm show that the victim was also placed as an outlier in cluster -1 as seen in Figure 41. Cluster -1 has 2 elements. However, as discussed earlier the elements placed in cluster -1 are classified as outliers that don not belong to any cluster.

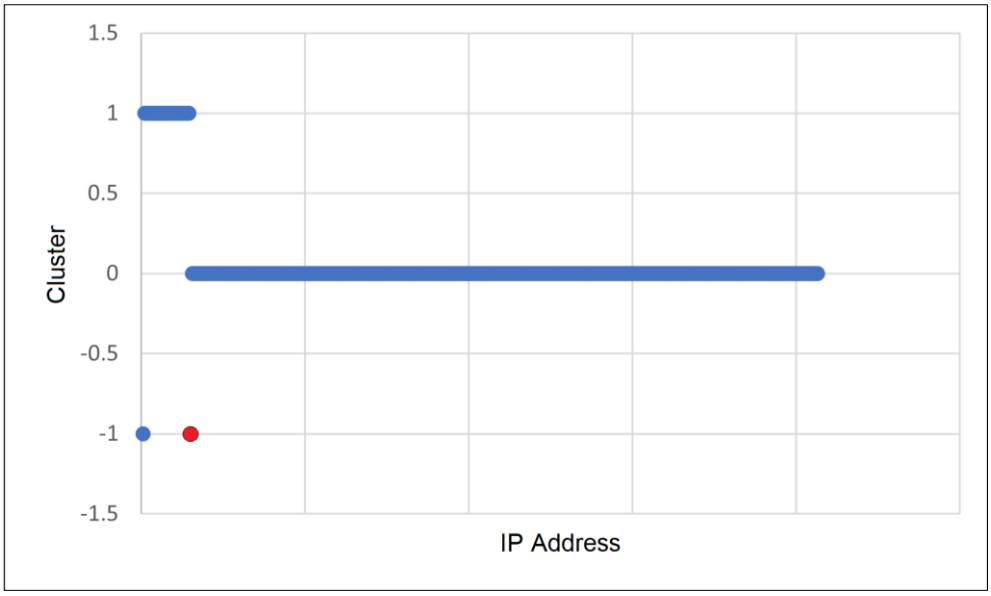

Figure 41 - Machine distribution by cluster with DBSCAN algorithm for x=100, host data and network data, on 01-03-18, with victims represented in red.

The Agglomerative algorithm also produced good results placing the victim in a one-element cluster. The victim is placed in cluster 6 that only has one element as displayed in Figure 42. Cluster 0 has 74 elements. Cluster 1 has 11 elements. Cluster 2 has 61 elements. Cluster 3 has 28 elements. Cluster 4 has 75 elements. Cluster 5 has 138 elements and cluster 7 has 25 elements.

Figure 42 - Machine distribution by cluster with Agglomerative algorithm for x=100, host data and network data, on 01-03-18, with victims represented in red.

5.3.4 Outlier detection with host, network data and x=100 on 01-03-18(Infiltration)

The first method defined in Subsection 4.2.4 is the selection of one-element clusters as positive occurrences. The results demonstrated in the previous subsection show that there is only one cluster that fulfils the criteria both in K-means and Agglomerative algorithms; the victim populates this cluster in both cases. The DBSCAN algorithm classifies two machines as outliers, one of whom is the victim. These results allow the calculation of the evaluation metrics for this method and are presented in Table 32.

Table 32 - Evaluation metrics for the first method, considering host and network data, x=100 on 01-03-18 (infiltration attack).

	K-means	DBSCAN	Agglomerative
TP	1	1	1
FP	0	1	0
TN	412	411	412
FN	0	0	0
Accuracy	1	1	1
Precision	1	0.5	1
Recall	1	1	1
F1	1	0.67	1

The scores obtained for every machine are the sum of the results given by every single algorithm. The victim, with IP Address 172.31.69.13, got a result of 0.004843 and placed first in the overall classification. Table 33 presents the scores of the top 10 machines. As shown, the score of the victim is considerably lower than the score of the second-placed machine, approximately 2.4% of the score of the second machine. The average of the scores was 1.204792, therefore only machines with a score lower than 0,060239 (5% of the average) are selected. Therefore, only the victim is selected.

Table 33 - Scores of the top 10 machines for 01-03-2018, with network and host data.

IP Address	K-means	DBSCAN	Agglomerative	Score
172.31.69.13	**7**	**-1**	**6**	**0.004843**
172.31.66.114	4	1	3	0.205811
172.31.66.112	4	1	3	0.205811
172.31.64.67	4	1	3	0.205811
172.31.64.26	4	1	3	0.205811
172.31.67.50	4	1	3	0.205811
172.31.66.41	4	1	3	0.205811
172.31.64.122	4	1	3	0.205811
172.31.65.56	4	1	3	0.205811
172.31.66.43	4	1	3	0.205811

The evaluation metrics for the second method are displayed in Table 34.

Table 34 - Evaluation metrics for the second method, considering host and network data, x=100 on 01-03-18 (infiltration attack).

TP	1
FP	0
TN	412
FN	0
Accuracy	1
Precision	1
Recall	1
F1 score	1

The third method described in Subsection 4.2.4 consists of selecting clusters with mean silhouette coefficient lower than half of the total mean value of the silhouette coefficient. Figure 43 illustrates the mean silhouette coefficients for every cluster produced by the K-means algorithm. There are two clusters that meet the requirements. Cluster 0 and cluster 7.

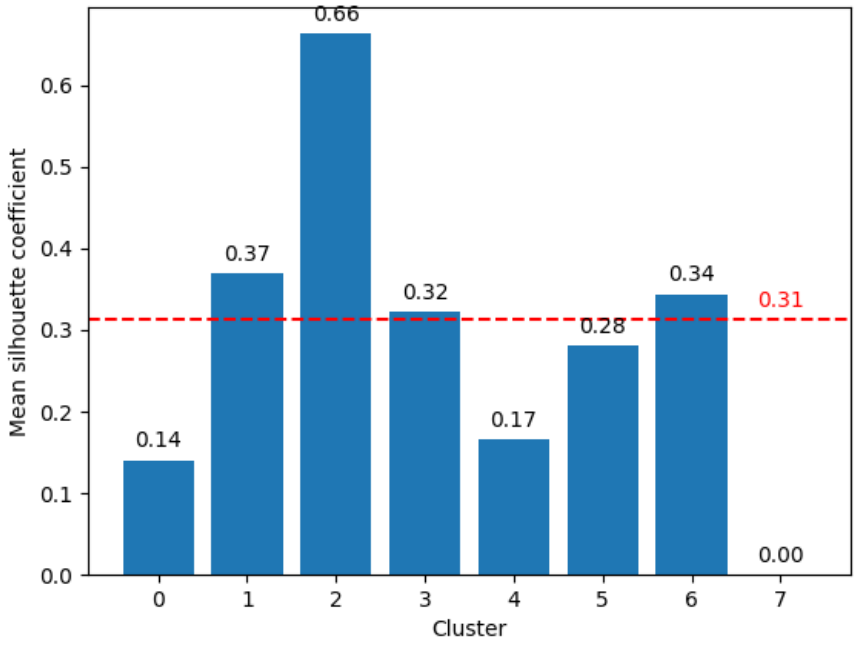

Figure 43 - Mean silhouette coefficient for the clusters in the K-means algorithm for x=100, on 01-03-2018, with host and network data. The dashed red line represents the average of all the clusters. The victim is in cluster 7.

Figure 44 presents the mean silhouette coefficient for cluster 0 and cluster 1. Cluster -1 is not represented because there is no meaning in the value of the mean silhouette coefficient of occurrences that do not belong together. The results show that cluster 0 fulfils the criteria for an outlier according to the third method.

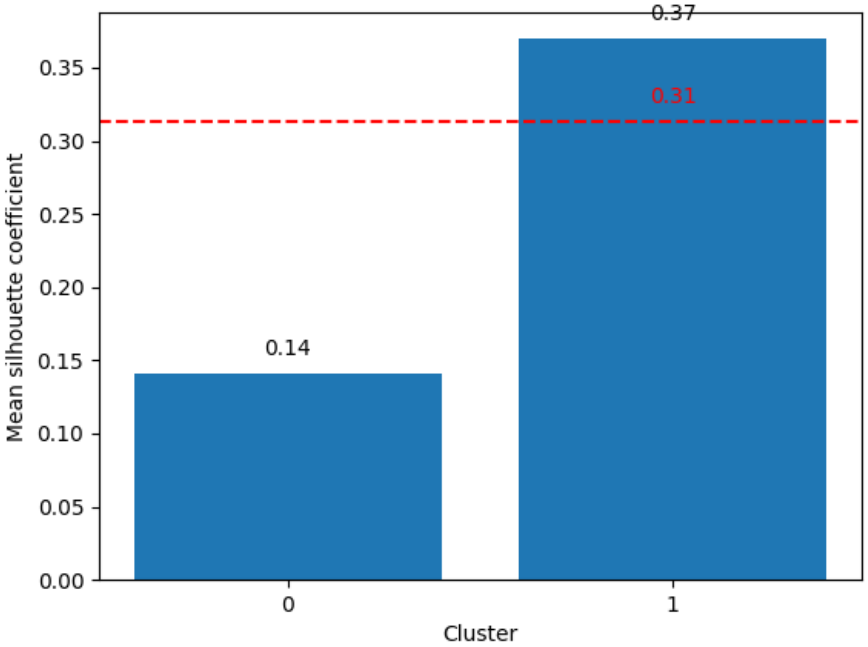

Figure 44 - Mean silhouette coefficient for the clusters in the DBSCAN algorithm for x=100, on 02-03-2018, with host and network data. The dashed red line represents the average of all the clusters.

Figure 45 presents the results for the mean silhouette coefficient of the clusters created by the Agglomerative algorithm. Both cluster 2 and cluster 6 fulfil the criteria for the third method.

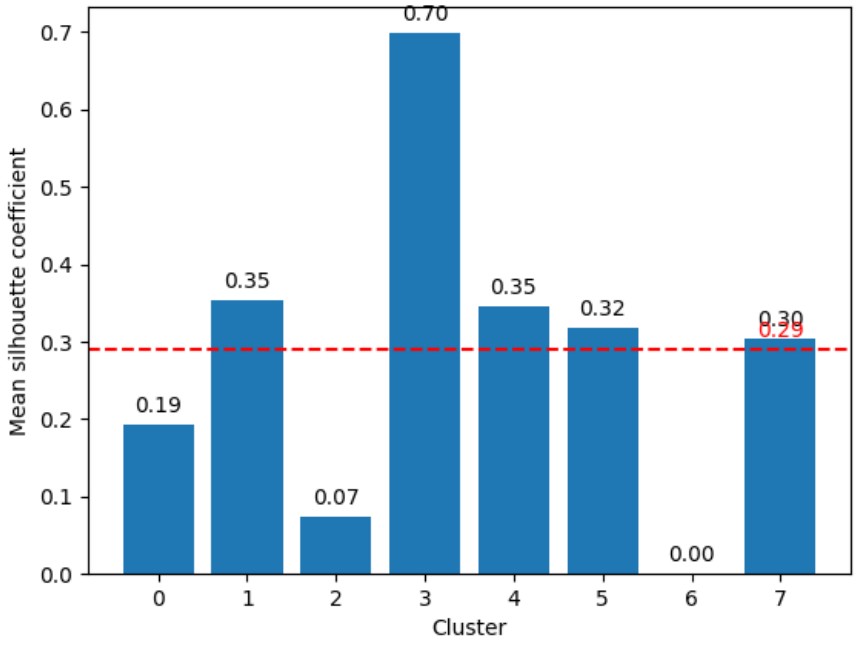

Figure 45 - Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100, on 02-03-2018, with host and network data. The dashed red line represents the average of all the clusters. The victim is in cluster 6.

From the analysis of the mean silhouette coefficients for the three algorithms and the criteria for the third method defined in Subsection 4.2.4, is possible to apply the evaluation metrics. These metrics are presented in Table 35.

Table 35 - Evaluation metrics for the third method, considering host and network data, x=100 on 01-03-18 (infiltration attack).

	K-means	DBSCAN	Agglomerative
TP	1	0	1
FP	55	383	62
TN	357	29	350
FN	0	1	0
Accuracy	0.87	0.07	0.85
Precision	0.02	0	0.02
Recall	1	0	1
F1	0.04	0	0.04

Finally, the fourth and last method consists in the selection of clusters with a population lower than ten. From the figures of the last subsection that illustrate the machine dispersion between the clusters created by each algorithm (Figure 39, Figure 41 and Figure 42) is possible to select the clusters that fulfil the criteria. Both the K-means and Agglomerative algorithms have one cluster each that fulfils the criteria for the fourth method. DBSCAN provides two one-element clusters and as such, two clusters with less than ten elements.

Table 36 - Evaluation metrics for the third method, considering host and network data, x=100 on 01-03-18 (infiltration attack).

	K-means	DBSCAN	Agglomerative
TP	1	1	1
FP	0	1	0
TN	412	411	412
FN	0	0	0
Accuracy	1	1	1
Precision	1	0.5	1
Recall	1	1	1
F1	1	0.67	1

5.3.5 Summary of results with host and network data

Figure 46 illustrates the results obtained for every one of the methods described in Subsection 4.2.4 for host and network data on 02-03-18. The fourth method achieved the best results. All other methods provided very low results, except for accuracy. The DBSCAN continues to provide poor results on the BOT attack. The large number of victims seems to have a more significant impact on the precision of this algorithm since, for every proposed method, the DBSCAN could not select any victim as an outlier.

Furthermore, the best method (fourth method – selection of cluster with a population lower than ten) achieves acceptable results with the K-means and Agglomerative algorithms. However, these results are tailored for an attack involving ten victims. An attack with a higher number of victims would be undetected by this approach. Therefore, there is a need for future work in methods to detect multiple simultaneous outliers. The methods explored in this work do not provide a robust alternative to select varying quantities of victims as outliers consistently.

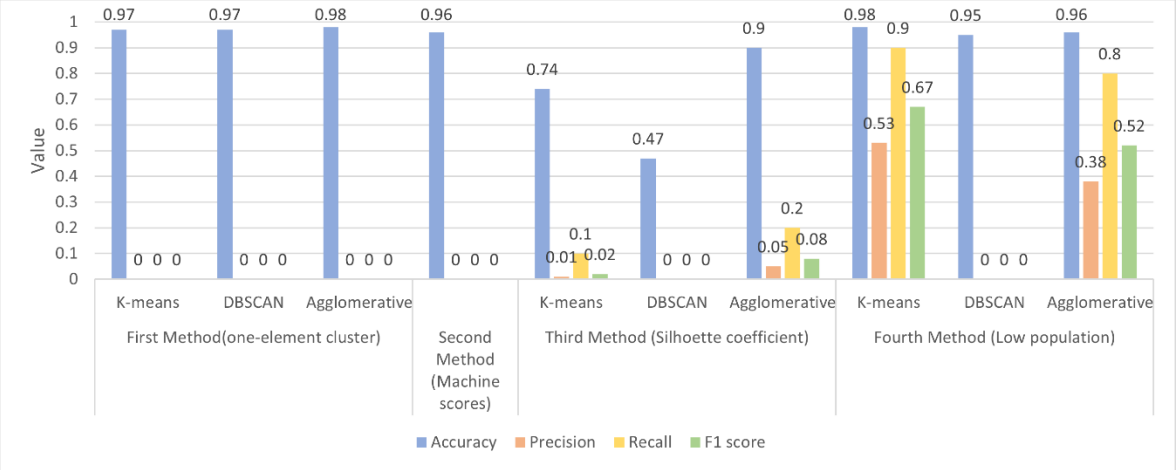

Figure 46 - Evaluation metrics for every method described in Subsection 4.2.4 for host and network data on 02-03-18. Method 1 is the selection of one-element clusters. Method 2 is scoring the machines proportionally to the population of its cluster in all algorithms. Method 3 is analysis of the silhouette coefficients of every cluster. Finally, method 4 is the consideration of any cluster with less than ten elements as a positive occurrence.

The proposed system is much more effective on the attack on 01-03-18 (infiltration). The results, presented in Figure 47, show that the system provides excellent evaluation metrics for all methods except for the third, mean silhouette coefficient analysis. The other methods performed very well and allowed for the unequivocal selection of the victim as an outlier. Furthermore, the K-means and the Agglomerative algorithms did not select any false positive or false negative. The DBSCAN selected a false positive both in the first and the fourth methods. These results are an improvement over similar systems, as is presented in Section 5.4.

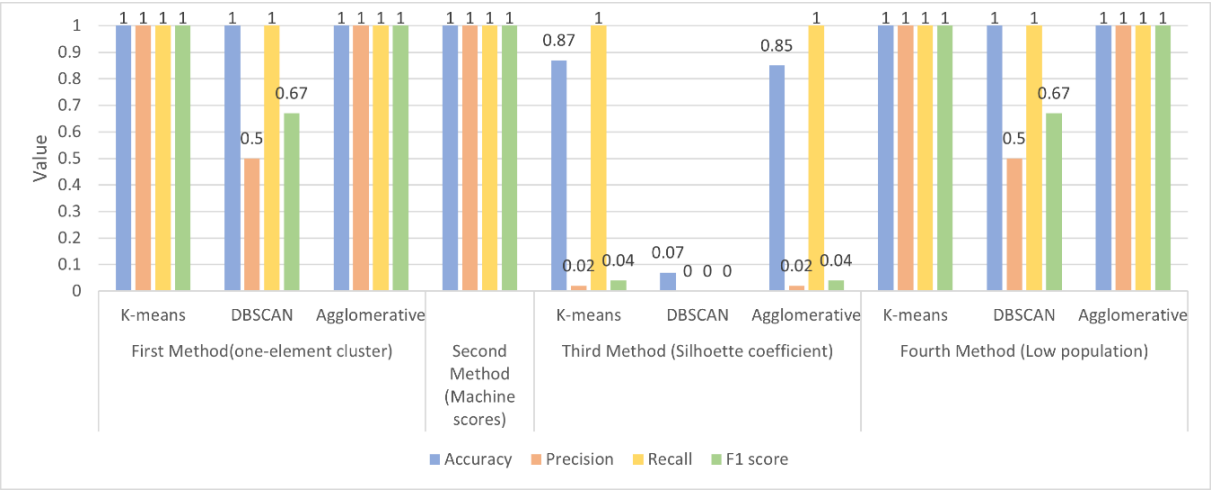

Figure 47 - Evaluation metrics for every method described in Subsection 4.2.4 for host and network data on 01-03-18. Method 1 is the selection of one-element clusters. Method 2 is scoring the machines proportionally to the population of its cluster in all algorithms. Method 3 is analysis of the silhouette coefficients of every cluster. Finally, method 4 is the consideration of any cluster with less than ten elements as a positive occurrence.

5.4 Comparison with different algorithms

In this section, results of similar systems that used the CSE-CIC-IDS2018 dataset are compared with the results achieved by the proposed approach. The proposed system will be evaluated, with host data and with the combination of network and host data separately, for the day 01-03-2018, resorting to the outlier detection method defined in Subsection 4.2.4 (selection of one-element clusters as positive occurrences). The results presented in Figure 48 show that the proposed approach with only host data was unable to identify any true positives with the K-means and Agglomerative algorithms; therefore, the values for the evaluation metrics are zero (except for accuracy). The results for the DBSCAN are marginally better, achieving an F1 Score of 0.25, a precision of 0.14, and a recall of 1.

However, the results with the conjunction of host data and network data are far superior, providing an improvement over the evaluation metrics obtained by DYNIDS both for the 10-minute and 60-minute time windows. The proposed approach achieved a better performance in every evaluation metric with every algorithm, showing the positive impact of including host data in the proposed system.

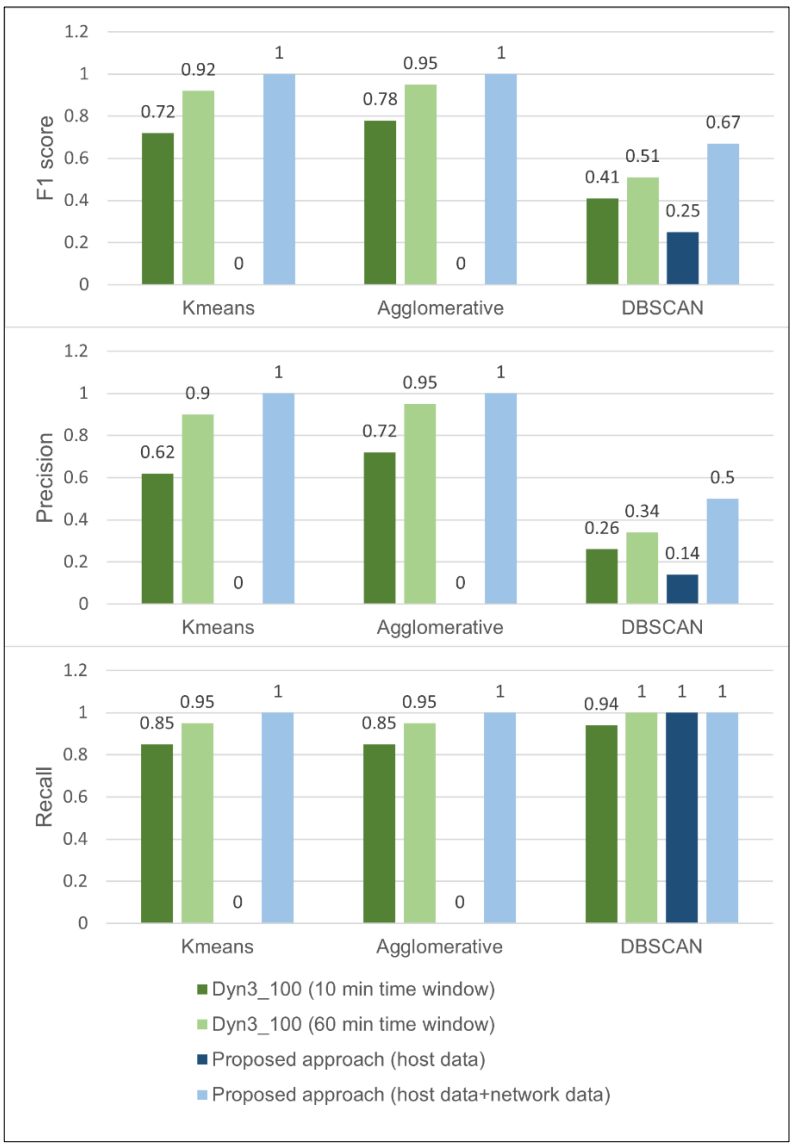

Figure 48 - Comparison of the evaluation metrics of the proposed approach with DYNIDS.

Figure 49 presents the evaluation metrics achieved by several systems for the CSE-CIC-IDS2018. The proposed approach is the system that provides the best results in all metrics. The F1-Score achieved by the proposed approach is 3% higher than the F1-Score obtained by DYNIDS and 5% higher in Recall. The precision of both systems is the same. Furthermore, the proposed system is able to outperform OutGene and Flow Hacker in all metrics.

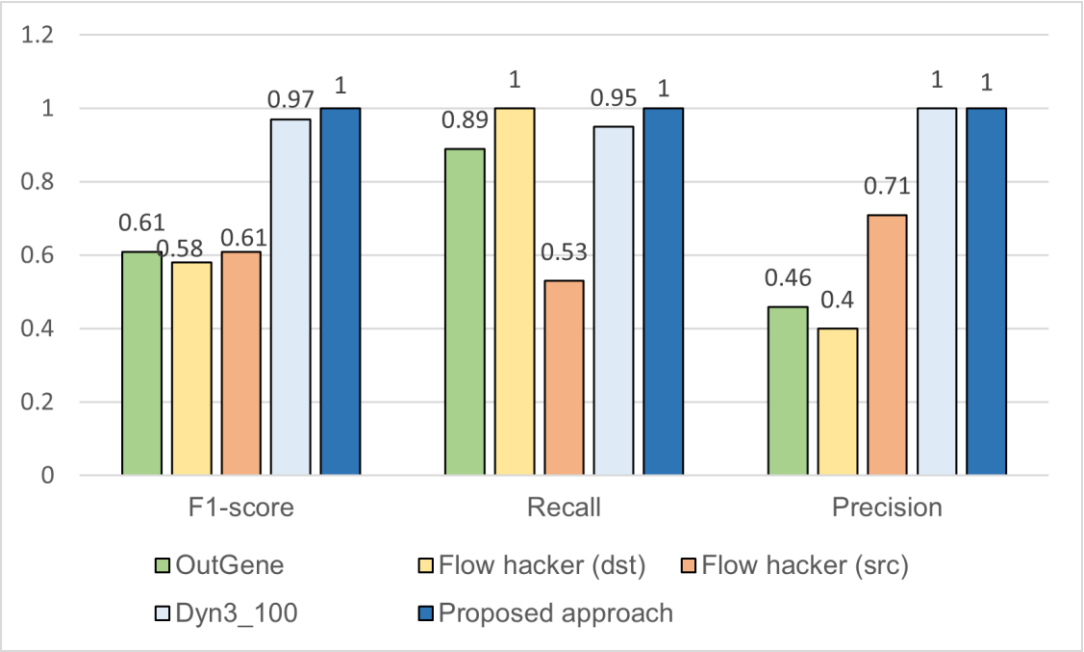

Figure 49 - Comparison of the performance of the proposed approach (host data and network data) versus other approaches, DYNIDS, Outgene and FlowHacker.

The proposed system achieved the desired performance in all the metrics where it was evaluated. The results illustrate the benefits of including host data, more precisely, MSWEs in an IDS. The comparison with DYNIDS and the improvement achieved show that both data types (host and network data) are compatible and can coexist in the same IDS. Furthermore, the valuable insights given by the MSWEs of the network should be utilised to complement the network data. Attacks, such as Botnet where the targeted machines perform automated commands simultaneously can stay undetected under a typical NIDS. However, the abnormal production of rare MSWEs is easily detected by the proposed system.

Moreover, despite the good performance in detecting the multiple victims on 02-03-18, there is still the need to develop methods to effectively select the victim cluster as a positive occurrence. Both K-means and the Agglomerative algorithm were able to consistently place the victims in the same cluster without false positives. The methods used for outlier detection (for Botnet attacks) utilised did not allow the system to achieve acceptable evaluation metrics on this day. In contrast, on 01-03-18 (infiltration attack) the outlier detection is straight forward and did not stand as an obstacle to achieve the best evaluation metrics of the considered systems.

6. Conclusion and Future Work

This work has shown the advantages of including host data, more precisely, MSWEs, in the current IDSs. The MSWEs provide an essential insight into a network activity. These events describe the operation of every machine in detail. By dynamically defining the MSWE IDs, the most relevant features were extracted to detect abnormal behaviour. Adding fixed features that provide an overview of the behaviour of the Windows machines is also an important element. These fixed features allow the system to analyse the proportion of the MSWEs' priority levels, which can indicate abnormal behaviour.

The chosen algorithms achieved the expected results. The main objective was to show that adding host data could improve current IDS systems. The results for day 01-03-18, when there was only one victim, were significantly better than similar approaches. On this day, with the combination of host and network data, every algorithm could consistently detect the victim and place it in a one-element cluster. However, the attacks of day 02-03-18 proved more challenging to the proposed approach. The algorithm DBSCAN performed poorly for this attack, unable to make a single detection. The K-means and Agglomerative algorithms performed better than DBSCAN but achieved worse results than similar approaches. According to the results achieved, an IDS that only considers MSWEs for the detection of attacks is insufficient to guarantee the robust and efficient security of a network. Victim detection is inconsistent and has a high false positive and false negative rate.

The dataset chosen was the dataset CSE-CIC-IDS2018. This dataset allows the comparison with similar approaches that use the same dataset, such as DYNIDS or OutGene. However, during the proposed system's development, failed attempts were performed to use other datasets that provide a combination of host and network data. Datasets such as DARPA1999, UHNDS, NDSEC-1, Unified Host and Network dataset and SSHCure. The extensive number of datasets tested during the development phase attest to the difficulty of finding reliable and good-quality datasets that combine host and network data.

For future work, it would be important to define more robust methods to identify multiple simultaneous victims in the system. The most significant disadvantage of clustering-based IDSs is that with various victims, the system interprets the creation of a multi-element cluster as a regular occurrence. When the attackers only affect one victim, selecting the resultant one-element victim is easy. However, there is a need to establish a robust method to determine the existence of a cluster of multiple victims.

References

[1] Z. Malekos, E. Lostri, and A. Lewis, "The Hidden Costs of Cybercrime," *The Center for Strategic and* International Studies (CSIS) of McAfee. Retrieved from https://www.mcafee.com/enterprise/enus/assets/reports/rp-hidden-costs-of-cybercrime.pdf, 2020.

[2] CNCS, "Relatório Cibersegurança em Portugal," Centro Nacional de Cibersegurança, 2022, vol. 3.

[3] R. Hofstede *et al.*, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," *IEEE Communications Surveys & Tutorials,* vol. 16, no. 4, pp. 2037-2064, 2014.

[4] W. Yurcik and Y. Li, "Internet security visualization case study: Instrumenting a network for NetFlow security visualization tools," in *21st Annual Computer Security Applications Conference (ACSAC)*, 2005.

[5] B. Trammell and E. Boschi, "An introduction to IP flow information export (IPFIX)," *IEEE Communications Magazine,* vol. 49, no. 4, pp. 89-95, 2011.

[6] S. W. A. Hamdani *et al.*, "Cybersecurity standards in the context of operating system: Practical aspects, analysis, and comparisons," *ACM Computing Surveys (CSUR),* vol. 54, no. 3, pp. 1-36, 2021.

[7] P. Sahoo, R. Chottray, and S. Pattnaiak, "Research issues on windows event log," *International Journal of Computer Applications,* vol. 41, no. 19, 2012.

[8] N. M. Ibrahim, A. Al-Nemrat, H. Jahankhani, and R. Bashroush, "Sufficiency of windows event log as evidence in digital forensics," in *International Conference on e-Democracy*, 2011: Springer, pp. 253-262.

[9] V. R. Team, "2012 data breach investigation report," Verizon, New York, 2012, vol. 5.

[10] Q. Do, B. Martini, J. Looi, Y. Wang, and K.-K. Choo, "Windows event forensic process," in *Advances in Digital Forensics X: 10th IFIP WG 11.9 International Conference, Vienna, Austria, January 8-10, 2014, Revised Selected Papers 10*, 2014: Springer, pp. 87-100.

[11] M. d. C. P. Tixteco, L. P. Tixteco, G. S. Pérez, and L. K. Toscano, "Intrusion Detection Using Indicators of Compromise Based on Best Practices and Windows Event Logs," in *Cimp 2016: the 11th international conference on internet monitoring and protection*, 2016.

[12] R. Santos, "Controlos de cibersegurança em ambientes ms windows de grandes empresas:Integração efetiva de eventos relevantes de segurança no SIEM AlienVault USM.," Master Sience, Departamento de informática, Faculdade Ciência Universidade de Lisboa, Lisbon, Portugal, 2018.

[13] S. S. Soniya and S. M. C. Vigila, "Intrusion detection system: Classification and techniques," in *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 2016: IEEE, pp. 1-7.

[14] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications,* vol. 36, no. 1, pp. 16-24, 2013.

[15] Y. Yang, K. McLaughlin, S. Sezer, Y. Yuan, and W. Huang, "Stateful intrusion detection for IEC 60870-5-104 SCADA security," in *2014 IEEE PES General Meeting| Conference & Exposition*, 2014: IEEE, pp. 1-5.

[16] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, "Host-based intrusion detection system with system calls: Review and future trends," *ACM Computing Surveys (CSUR),* vol. 51, no. 5, pp. 1-36, 2018.

[17] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020: IEEE, pp. 1172-1189.

[18] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet],* vol. 9, no. 1, pp. 381-386, 2020.

[19] F. Osisanwo, J. Akinsola, O. Awodele, J. Hinmikaiye, O. Olakanmi, and J. Akinjobi, "Supervised machine learning algorithms: classification and comparison," *International Journal of Computer Trends and Technology (IJCTT),* vol. 48, no. 3, pp. 128-138, 2017.

[20] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning,* vol. 29, pp. 131-163, 1997.

[21] H. Ramchoun, Y. Ghanou, M. Ettaouil, and M. A. Janati Idrissi, "Multilayer perceptron: Architecture optimization and training," 2016.

[22] W. S. Noble, "What is a support vector machine?," *Nature biotechnology,* vol. 24, no. 12, pp. 1565-1567, 2006.

[23] C. Kingsford and S. L. Salzberg, "What are decision trees?," *Nature biotechnology,* vol. 26, no. 9, pp. 1011-1013, 2008.

[24] T. S. Madhulatha, "An overview on clustering methods," *arXiv preprint arXiv:1205.1117,* 2012.

[25] L. Rokach, "A survey of clustering algorithms," *Data mining and knowledge discovery handbook,* pp. 269-298, 2010.

[26] S. Pandit and S. Gupta, "A comparative study on distance measuring approaches for clustering," *International journal of research in computer science,* vol. 2, no. 1, pp. 29-31, 2011.

[27] S. Fletcher and M. Z. Islam, "Comparing sets of patterns with the Jaccard index," *Australasian Journal of Information Systems,* vol. 22, 2018.

[28] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* vol. 2, no. 1, pp. 86-97, 2012.

[29] R. J. Campello, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* vol. 10, no. 2, p. e1343, 2020.

[30] L. Dias, S. Valente, and M. Correia, "Go with the flow: Clustering dynamically-defined netflow features for network intrusion detection with DynIDS," in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, 2020: IEEE, pp. 1-10.

[31] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "DBSCAN: Past, present and future," in *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, 2014: IEEE, pp. 232-238.

[32] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM Sigmod record,* vol. 28, no. 2, pp. 49-60, 1999.

[33] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined CNN/RNN model," in *ECML PKDD 2018 Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings 18*, 2019: Springer, pp. 149-158.

[34] J. Liu, M. Simsek, B. Kantarci, M. Bagheri, and P. Djukic, "Collaborative Feature Maps of Networks and Hosts for AI-driven Intrusion Detection," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, 2022: IEEE.

[35] N. N. Tran, R. Sarker, and J. Hu, "An approach for host-based intrusion detection system design using convolutional neural network," in *Mobile Networks and Management: 9th International Conference, MONAMI 2017, Melbourne, Australia, December 13-15, 2017, Proceedings 9*, 2018: Springer, pp. 116-126.

[36] K. Berlin, D. Slater, and J. Saxe, "Malicious behavior detection using windows audit logs," in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, 2015, pp. 35-44.

[37] D. Park, S. Kim, H. Kwon, D. Shin, and D. Shin, "Host-Based Intrusion Detection Model Using Siamese Network," *IEEE Access,* vol. 9, pp. 76614-76623, 2021.

[38] A. D. Landress, "A hybrid approach to reducing the false positive rate in unsupervised machine learning intrusion detection," in *SoutheastCon 2016*, 2016: IEEE, pp. 1-6.

[39] S. Sahu and B. M. Mehtre, "Network intrusion detection system using J48 Decision Tree," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015: IEEE, pp. 2023-2026.

[40] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, pp. 21-26.

[41] A. Guezzaz, S. Benkirane, M. Azrour, and S. Khurram, "A Reliable Network Intrusion Detection Approach Using Decision Tree with Enhanced Data Quality," *Security and Communication Networks,* vol. 2021, pp. 1-8, 2021.

[42] R. Atefinia and M. Ahmadi, "Network intrusion detection using multi-architectural modular deep neural network," *The Journal of Supercomputing,* vol. 77, no. 4, pp. 3571-3593, 2021/04/01 2021.

[43] G. Pu, L. Wang, J. Shen, and F. Dong, "A hybrid unsupervised clustering-based anomaly detection method," *Tsinghua Science and Technology,* vol. 26, no. 2, pp. 146-153, 2020.

[44] L. Dias, H. Reia, R. Neves, and M. Correia, "OutGene: Detecting Undefined Network Attacks with Time Stretching and Genetic Zooms," in *Network and System Security*, Cham, J. K. Liu and X. Huang, Eds., 2019// 2019: Springer International Publishing, pp. 199-220.

[45] G. Andresini, A. Appice, and D. Malerba, "Nearest cluster-based intrusion detection through convolutional neural networks," *Knowledge-Based Systems,* vol. 216, p. 106798, 2021/03/15 2021.

[46] W. Zhong, N. Yu, and C. Ai, "Applying big data based deep learning system to intrusion detection," *Big Data Mining and Analytics,* vol. 3, no. 3, pp. 181-195, 2020.

[47] "A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)." https://registry.opendata.aws/cse-cic-ids2018. (accessed 11/5/2023, 2023).