TÉCNICO LISBOA

ACADEMIA MILITAR
MILITARY ACADEMY

Security Analytics with Mixed Event Sources and Ensembles

Simão Pedro Avelino Valente

Thesis to obtain the Master of Science Degree in
Electrical and Computer Engineering

Supervisors: Professor Miguel Nuno Dias Alves Pupo Correia
Major TM Luís Filipe Xavier Cavaco Mendonça Dias

Examination Committee:
Chairperson: Professor Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Professor Miguel Nuno Dias Alves Pupo Correia
Members of the Committee: Professor Ibéria Vitória Sousa Medeiros
Lieutenant Colonel José Jaime Soares Pereira

December 2019

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 work was developed with the great help of my supervisors, Professor Miguel Correia and Major Tm Luís Dias. It is important to recognize their crucial role. For this reason, I would like to thank their support and availability to discuss and help in all decisions made. Thanks to the personnel at DCSI which contribute by establishing and providing all the conditions and materials to develop this work. I would like to also thank Lieutenant Fernando Batista for his time and availability. Since our work is the continuation of Lieutenant Hélder Reia's thesis, I also thank him for his work, knowledge, and help provided in an earlier phase. I would also like to thank my brother in law for his help and availability. As a fundamental help for developing this work, I would like to give a special thanks to all my family. They played a crucial role in this work, by keeping me focused and providing great support.

Abstract

The present thesis shows the complexity associated with the creation of an Intrusion Detection System (IDS) considering host and netflow data, as well as the limitations of current detection systems, either host-based or network-based. It presents a new approach based on Windows OS logs and network flows.

The main goal is to provide and test a process capable of detecting intrusions, the use of a high numbers of features and the use of different data sources by themselves and combined together for intrusion detection. The process uses multiple unsupervised machine learning algorithms ensemble to detect outliers.

To achieve the goal, several unsupervised machine learning algorithms were tested for performance either in time or outlier detection capability, based on our classification criteria. The three algorithms that achieved better performance were considered. The process is composed of feature extraction, normalization, clustering and clustering ensemble. After clustering ensemble, the detected outliers are presented to an analyst, in order to mitigate them. Five different approaches were used for feature selection, three of them using only netflow data and one using Windows events. The last approach combined the best approach used on netflow with the Windows events.

To evaluate the developed system, a public artificial dataset was used. This dataset contains logs from machines with Windows OS and netflow information. Additionally, a dataset with real logs and flows from an organization was also obtained.

The present work was developed with the collaboration of the Portuguese Army.

Key-words: Cybersecurity, Machine Learning, Intrusion Detection System, Security Analytics, Logs, Netflow.

Resumo

A presente tese expõe a complexidade associada à criação de um Sistema de Deteção de Intrusões (SDI) baseado em dados de rede e de eventos, assim como as limitações dos atuais sistemas de deteção, quer baseados na máquina quer baseados em rede. É apresentada uma nova abordagem sustentada em logs do sistema operativo Windows e de rede.

O principal objetivo é fornecer e testar: um conjunto de processos capazes de detetar intrusões, a utilização de um elevado número de características e o uso de diferentes fontes de dados, quer isoladas quer combinadas para deteção de intrusões. Este conjunto de processos combina múltiplos algoritmos de aprendizagem automática para a deteção de entidades com comportamentos anormais.

Para atingir o objetivo, testaram-se vários algoritmos de aprendizagem automática não-supervisionada quanto à sua performance quer em tempo quer na sua capacidade de deteção de outliers baseado no critério de classificação criado. Foram escolhidos os três algoritmos que obtiveram melhor performance. O processo é composto por extração de features, seguido de normalização, clustering e clustering ensemble. Depois de realizado o clustering ensemble, os resultados são apresentados a um analista que decidirá a ação a tomar face as entidades consideradas outliers. Foram consideradas cinco abordagens na seleção de features, sendo que três destas aborgadens utilizavam apenas dados de netflow e uma utilizava eventos do sistema operativo Windows. A última abordagem consiste na combinação da abordagem dos eventos Windows com a melhor abordagem dos dados de netflow.

Para avaliar o sistema desenvolvido foi utilizado um dataset artificial disponível publicamente. Este dataset contém logs de máquinas com o sistema operativo Windows e informações de rede. Foi ainda obtido um dataset com dados reais (logs e netflow) de uma organização.

O presente trabalho foi desenvolvido com a colaboração do Exército Português.

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

Contents

List of Figures xi

List of Tables xiii

Acronyms xv

1 Introduction 1
- 1.1 Challenges and Motivation . . . 1
- 1.2 Objectives and Key Results . . . 2
- 1.3 Report Structure . . . 3

2 Background 5
- 2.1 Machine Learning Methods . . . 5
  - 2.1.1 Supervised Machine Learning . . . 5
  - 2.1.2 Unsupervised Machine Learning . . . 7
- 2.2 Libraries . . . 10
  - 2.2.1 Pandas . . . 10
  - 2.2.2 Scikit-learn . . . 10
- 2.3 Summary . . . 11

3 Related Work 13
- 3.1 Anomaly Detection . . . 13
- 3.2 Host Based Intrusion Detection . . . 14
- 3.3 Network Intrusion Detection . . . 16
- 3.4 Hybrid Intrusion Detection Systems . . . 17
- 3.5 Summary . . . 18

4 Datasets Description 21
- 4.1 CSE-CIC-IDS 2018 . . . 21
- 4.2 Portuguese Army Dataset . . . 23
- 4.3 Summary . . . 24

5 Clustering Algorithm Selection 27
- 5.1 Selecting Algorithm Parameters . . . 29
- 5.2 Summary . . . 31

6 Proposed Solution 33
- 6.1 Processing Data . . . 33
  - 6.1.1 Feature Engineering . . . 33
  - 6.1.2 Feature Extraction and Normalization . . . 35
- 6.2 Clustering . . . 35
  - 6.2.1 Applying Clustering Algorithms . . . 35
  - 6.2.2 Defining Classification . . . 36
  - 6.2.3 Ensemble Method . . . 37
- 6.3 Evaluation Metrics . . . 37
- 6.4 Summary . . . 38

7 Experimental Evaluation **41**
7.1 Combining Algorithms in Netflow Sources . . . 41
7.1.1 FTP (14-02-18) . . . 41
7.1.2 SSH (14-02-18) . . . 42
7.1.3 DoS - GoldenEye (15-02-18) . . . 43
7.1.4 DoS - Slowloris (15-02-18) . . . 45
7.1.5 DDoS - LOIC HTTP (20-02-18) . . . 45
7.1.6 DDoS - LOIC UDP (20-02-18) . . . 47
7.1.7 DDoS - LOIC UDP (21-02-18) . . . 47
7.1.8 DDoS - HOIC (21-02-18) . . . 48
7.1.9 Brute Force Web (22-02-18) . . . 49
7.1.10 Brute Force XSS (22-02-18) . . . 50
7.1.11 SQL Injection (22-02-18) . . . 51
7.1.12 Brute Force Web (23-02-18) . . . 52
7.1.13 Brute Force XSS (23-02-18) . . . 53
7.1.14 SQL Injection (23-02-18) . . . 54
7.1.15 Infiltration (28-02-18) . . . 55
7.1.16 Infiltration (01-03-18) . . . 56
7.1.17 Bot (02-03-18) . . . 57
7.1.18 Summary of Results . . . 59
7.2 Combining Algorithms in Windows Event Sources . . . 59
7.2.1 Infiltration (28-02-18) . . . 59
7.2.2 Infiltration (01-03-18) . . . 60
7.2.3 Bot (02-03-18) . . . 61
7.2.4 Summary of Results . . . 62
7.3 Merging Events and Flows . . . 62
7.3.1 Summary of Results . . . 63

8 Conclusions and Future Work **65**

References **67**

A Windows Event Viewer Features **71**

B CIC Features **73**

C Summary Of Results From CSE-CIC-IDS 2018 (Netflow Sources) **76**

List of Figures

1 Example of two-layer ANN adapted from [11]. 6
2 Network topology, adapted from [9]. 21
3 Algorithms performance against large data. 28
5 Processing data to later perform clustering. 33
6 Example of the process of flow features extraction. 34
7 Architecture implemented. 36
8 Ensemble method. 38

List of Tables

1 Essential literature on HIDS, NIDS and Hybrid IDS. 19
2 Summary of attacks, by day (adapted from [9]). 22
3 Summary of attacks, by day. 24
4 Algorithms capacity to detect outliers. 29
5 Fixed features, each one is considered from source and destination point of view. 34
6 Fixed port features. 34
7 Example of the number of entities by cluster for each algorithm in CSE-CIC-IDS dataset for a 10 minutes time-window. 36
8 Cluster considered outliers and their entities. 38
9 Metrics used to analyse results. 39
10 Results of the best time-window for FTP attack from day 14/02/18. 42
11 Relevant victim features extracted from time-window 11:30-12:00 in day 14/02/18. 42
12 Results of the best time-window for FTP attack from day 14/02/18 with ensemble. 42
13 Results of the best time-window for SSH attack from day 14/02/18. 43
14 Victim features extracted from time-window 15:20-15:30 in day 14/02/18. 43
15 Results of the best time-window for SSH attack from day 14/02/18 with clustering ensemble. 43
16 Results of the best time-window for DoS GoldenEye attack from day 15/02/18. 44
17 Victim features extracted from time-windows 09:20-09:30 and 9:50-10:00 in day 15/02/18. 44
18 Results of the best time-window for DoS GoldenEye attack from day 15/02/18 with ensemble. 44
19 Results of the best time-window for DoS Slowloris attack from day 15/02/18. 45
20 Relevant features extracted from time-window 11:00-11:10 and 11:20-11:30 in day 15/02/18. 45
21 Results of the best time-window for DoS Slowloris attack from day 15/02/18 with clustering ensemble. 46
22 Results of the best time-window for DDoS LOIC HTTP attack from day 20/02/18. 46
23 Relevant features extracted from time-window 10:10-10:20 and 10:40-10:50 in day 20/02/18. 46
24 Results of the best time-window for DDoS LOIC HTTP attack from day 20/02/18 with ensemble. 47
25 Results of the best time-window for DDoS LOIC UDP attack from day 21/02/18. 47
26 Relevant features extracted from time-window 10:10-10:20 and 10:20-10:30 in day 21/02/18. 48
27 Results of the best time-window for DDoS LOIC UDP attack from day 21/02/18 with clustering ensemble. 48
28 Results of the best time-window for DDoS HOIC attack from day 21/02/18. 48
29 Relevant features extracted from time-window 14:50-15:00 in day 21/02/18. 49
30 Results of the best time-window for DDoS HOIC attack from day 21/02/18 with clustering ensemble. 49
31 Results of the best time-window for brute force web attack from day 22/02/18. 50
32 Relevant features extracted from time-window 10:20-10:30 in day 22/02/18. 50
33 Results of the best time-window for brute force web attack from day 22/02/18 with ensemble. 50
34 Results of the best time-window for brute force XSS attack from day 22/02/18. 51
35 Relevant features extracted from time-window 14:20-14:30 and 13:50-14:00 in day 22/02/18. 51
36 Results of the best time-window for brute force XSS attack from day 22/02/18 with ensemble. 51
37 Results of the best time-window for SQL injection attack from day 22/02/18. 52
38 Relevant features extracted from time-window 16:20-16:30 in day 22/02/18. 52
39 Results of the best time-window for SQL injection attack from day 22/02/18 with ensemble. 52
40 Results of the best time-window for brute force web attack from day 23/02/18. 52

41 Victim features extracted from time-window 10:00-10:10 in day 23/02/18. 53
42 Results of the best time-window for brute force web attack from day 23/02/18 with ensemble. 53
43 Results of the best time-window for brute force XSS attack from day 23/02/18. 53
44 Victim features extracted from time-window 13:00-13:10 in day 23/02/18. 54
45 Results of the best time-window for brute force XSS attack from day 23/02/18 with clustering ensemble. 54
46 Results of the best time-window for SQL injection attack from day 23/02/18. 54
47 Parcial features from attacker extracted on time-window 15:00-15:10 in day 23/02/18. . . 55
48 Results of the best time-window for SQL injection attack from day 23/02/18 with ensemble. 55
49 Results of the best time-window for infiltration attack from day 28/02/18. 55
50 Relevant features from victim extracted on time-window 11:30-12:00 and 11:50-12:00 in day 28/02/18. 56
51 Results of the best time-window for infiltration attack from day 28/02/18 with clustering ensemble. 56
52 Results of the best time-window for infiltration attack from day 01/03/18. 56
53 Relevant features from victim extracted on time-window 14:30-14:40 and 14:40-14:50 in day 01/03/18. 57
54 Results of the best time-window for infiltration attack from day 01/03/18 with clustering ensemble. 57
55 Results of the best time-window of bot attack from day 02/03/18. 58
56 Relevant features from attacker extracted on time-windows 15:40-15:50 and 11:20-11:30 in day 02/03/18. 58
57 Results of the best time-window of bot attack from day 02/03/18 with ensemble. 58
58 Results of the best time-window from day 28/02/18. 59
59 Features of the considered outliers in DBSCAN (28-02-2018). 60
60 Results of the best time-window from day 28/02/18 with ensemble. 60
61 Results of the best time-window from day 01/03/18. 60
62 Victim features extracted from time-window 10:00-10:10 in day 01/03/18. 61
63 Results of the best time-window from day 01/03/18 with ensemble. 61
64 Results of the best time-window from day 02/03/18. 61
65 Results of the best time-window from day 02/03/18 with ensemble. 62
66 Summary of Windows sources results by attack. 62
67 Summary of features that could be extracted from Windows Events Viewer [27, 43]. . . . 71
68 CIC features (adapted from [9]). 74
69 Summary of Netflow Sources results by attack. 77

Acronyms

AD Active Directory. 34

ANN Artificial Neural Network. xi, 6, 7, 8, 27

APT Advanced Persistent Threads. 18, 19

BIRCH Balanced Iterative Reducing and Clustering using Hierarchies. 17

BN Bayesian network. 6

CIC Canadian Institute for Cybersecurity. ix, x, xiii, 21, 24, 31, 33, 35, 36, 41, 59, 65, 76

CMS Cryptographic Message Syntax. 34

CSE Communications Security Establishment. ix, x, xiii, 21, 24, 31, 33, 35, 36, 41, 59, 65, 76

CSV Comma Separated Values. 10, 23

DBSCAN Density-Based Spatial Clustering of Applications with Noise. xiv, 2, 3, 8, 10, 17, 27, 28, 29, 30, 31, 35, 36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 65

DCE Distributed Computing Environment. 34

DCSI Direção de Comunicações e Sistemas de Informação. iii

DDoS Distributed Denial of Service. x, xiii, 22, 31, 45, 46, 47, 48, 49, 76

DHCP Dynamic Host Configuration Protocol. 34

DM Data Mining. 5, 14, 17, 18, 19

DNS Domain Name System. 34

DoS Denial Of Service. x, xiii, 22, 43, 44, 45, 46, 76

DVWA Damn Vulnerable Web App. 54

ELM Extreme Learning Machine. 16

EM Expectation-Maximization. 7, 9, 16

EVTX Windows XML Event Log. 23, 24

FTP File Transfer Protocol. x, xiii, 3, 22, 33, 34, 41, 42, 76

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise. 28, 29, 35

HIDS Host-based Intrusion Detection System. xiii, 2, 14, 15, 16, 18, 19

HMM Hidden Markov Model. 6, 15

HOIC High Orbit Ion Cannon. x, xiii, 48, 49, 76

HSG High-level Scenario Graph. 18, 19

HTTP HyperText Transfer Protocol. x, xiii, 22, 34, 43, 45, 46, 47, 76

HTTPS HyperText Transfer Protocol Secure. 34

IDS Intrusion Detection System. v, ix, x, xiii, 1, 2, 13, 15, 17, 18, 19, 21, 23, 31, 33, 35, 36, 41, 59, 76

IKE Internet Key Exchange. 34

IoT Internet of Things. 13

IP Internet Protocol. 23, 24, 34, 41, 45, 58

IRC Internet Relay Chat. 34

ISAKMP Internet Security Association and Key Management Protocol. 34

JSON JavaScript Object Notation. 23

KDD Knowledge Discovery in Database. 17

KNN K-Nearest Neighbor. 6, 7, 13

LANL Los Alamos Network Laboratory. 19

LOF Local Outlier Factor. 2, 3, 7, 8, 13, 14, 28, 29, 31, 37, 41, 54, 59, 60, 65, 66

LOIC Low Orbit Ion Cannon. x, xiii, 45, 46, 47, 48, 59, 76

LR Logistic Regression. 16, 17, 19

ML Machine Learning. ix, 2, 5, 6, 7, 10, 11, 13, 14, 17, 18, 19, 23, 41

NetBIOS Network Basic Input/Output System. 34

NIDS Network Intrusion Detection System. xiii, 2, 14, 16, 18, 19

NN Nearest Neighbor. 13

NTP Network Time Protocol. 34

OPTICS Ordering Points To Identify the Clustering Structure. 27, 28, 29, 35

OS Operative System. v, 14, 15, 16, 19, 21, 23, 33, 35, 59, 65

PCA Principal Component Analysis. 9, 10

PCAP Package Capture. 23

RPC Remote Procedure Call. 23, 34

SCM Service Control Manager. 71

SDI Sistema de Deteção de Intrusões. vii

SMTP Simple Mail Transfer Protocol. 34

SOM Self-Organizing Map. 7, 8, 9, 27, 28

SQL Structured Query Language. x, xiii, 22, 51, 52, 54, 76

SSDP Simple Service Discovery Protocol. 34

SSH Secure Shell. x, xiii, 22, 34, 37, 42, 43, 55, 76

SVM Support Vector Machine. 6, 10, 13, 14, 17

TCP Transmission Control Protocol. 45

TLS Transport Layer Security. 21

TTPs Tactics, Techniques and Procedures. 18

UDP User Datagram Protocol. x, xiii, 47, 48, 59, 76

VM Virtual Machine. 19

WCSS within-clusters sum-of-squares. 29

XSS Cross-Site Scripting. x, xiii, 22, 50, 51, 76

1 Introduction

This chapter presents a brief description of the objectives of this work, how it fits in the context of security analytics and the key results obtained.

1.1 Challenges and Motivation

Nowadays we observe an exponential growth in the use of technology and smart devices to connect people among themselves and the world. This makes daily tasks easier but also turns people more vulnerable to computer attacks. The number of attacks is increasing as well as attackers insight, leading to complex attack techniques that are more difficult to detect and, therefore, to mitigate or respond to.

Many modern computer attacks do not use files. These are known as fileless attacks and consist of using tools already installed in the host operating system. Those attacks can also be performed by simply executing scripts and shellcode in memory. A usual location to hide scripts is the Windows Registry. According to [46], in 2017 almost 77% of the attacks experienced by organizations were fileless or used exploits. Depending on file usage, fileless attacks can be classified into three types [21]: type 1 - no file activity performed; type 2 - no files written to disk but files are used indirectly; type 3 - files are required to achieve persistence. It is important to understand which are the entry points and the procedures used to get inside the system. When performing fileless attacks, the attacker can get inside the system by performing injection, an exploit or exploring hardware vulnerabilities. One possible technique to detect such attacks is to analyze logs produced by the computer. Logs contain precious information about applications, services, and processes running on the computer. Their rich content could be used for security analytics. Fileless attacks are becoming more complex and techniques to successfully perform them are increasing. As any attack it needs an entry point, to send a malicious file or gain access to a victim's machine. Normally the attacker exploits vulnerabilities in services and applications on victim's machine to find an entry point. This search for vulnerabilities is performed by analyzing network traffic in order to understand which services and applications are used. Once this information is collected, the attacker begins to exploit their vulnerabilities. This entry point can be used not only to gain access to the victim's machine but also to perform other attacks. This leads us to conclude that besides analyzing logs, it is also important to analyze network traffic to detect anomalies.

Since attacks are rapidly evolving, it is important to detect them and adapt to novel intrusion techniques. In fact, according to [35], in 2018 the mean time to detect an external attack was 184 days and an internal was 50.5 days. These numbers are high because not only attacks are difficult to detect but also many current detection techniques are still based on signatures. Signatures represent well known malicious code used to perform attacks. Basically, most of the IDS analyze code to detect these codes. It is also important to understand how security analytics evolved in terms of logs usage. According to [18], first system administrators had to monitor all user activities in the console to perform security analytics. That time it was enough to detect system anomalies, nevertheless, this solution was not scalable. In the next phase, system administrators analyzed audit logs, to find suspect or malicious activity. This process was time expensive due to a large amount of data produced. With the increasing storage capabilities, audit logs were stored and software was developed to analyze them. Even so it was not enough to detect intrusions in real time. The capability of real-time intrusion detection emerged in early 90s, with the creation of a system capable to review audit data as soon as it was produced. Real-time detection allows response to attacks and their attempts, as they were identified. Currently intrusion detection systems have been largely improved to detect more sophisticated attacks.

Intrusion Detection Systems emerged from the need for reducing the number of attacks and their detection time. These IDSs can be classified in two different ways: according to the information given to the system, and based on input data source. The first classification, according to the information given

to the system can be divided into anomaly and misuse detection. Anomaly detection focus on finding patterns that deviate from the normal behavior of an entity. Misuse detection tries to find a well known malicious behavior previously seen. Depending on the IDS input data source, they can be classified as:

- Host-based Intrusion Detection System (HIDS);
- Network Intrusion Detection System (NIDS).

According to [8, 14], HIDSs cover the events produced in the monitored host, aim for computer protection and prevent malicious code execution in the system. On the other side, NIDSs capture network traffic, covering all the machines/computers within the network. Currently, NIDSs are more used than HIDSs because they can analyze simultaneously multiple users and require less maintenance. Nevertheless, NIDSs have limitations, for example, they cannot analyze encrypted network traffic. Another disadvantage is that sometimes they are not able to analyze all network traffic due to the high cost associated with the deployment of traffic collectors. Using HIDSs presents some advantages such as a lower amount of data to process and the highest quality provided by host logs. Each operation or event performed in a computer may produce a set of log entries that gather precious information for security, performance and debug. The logs main challenge is their content and format, which varies depending on the systems and its components. Nevertheless, security companies have been working to use logs in a way of gathering information about different devices. Their main goal is to detect and prevent intrusions in their systems. A new approach in IDSs is to combine both HIDS and NIDS. It joins precious information from the host with relevant network traffic. It enables the detection of offline and online attacks. Nevertheless, this new approach increases the amount of data provided to systems. This fact leads to the need for selecting the most relevant features for security analytics. Such a high quantity of data is often designated Big Data and the associated research area deals with processing a large amount of data. Big Data can be defined in terms of 3 V's [10]:, Volume, by a large amount of data produced daily and his dimensions; Variety, by the different sources and type of data produced; Velocity, by the ability to process large amounts of data quickly.

Security enterprises have started to use Machine Learning (ML) to improve their IDS. ML came not only to deal with big data but also because it is a powerful tool to analyze and interpret patterns and structures in data. Since ML algorithms can learn from data, it is very useful to security analytics. Nevertheless, the use of ML classifiers implies to know all attacks which represent a misuse approach with a set of well known features. This approach is vulnerable to new attacks. To use ML to detect novelty attacks a system should contain sets of dynamic features. This approach feeds the system with a set of relevant features used in a particular time.

1.2 Objectives and Key Results

To overcome some of these intrusion detection difficulties and deal with big data, a new hybrid system is proposed. It collects information from Windows event sources (application, security and system levels) and the network, so it is a hybrid, or mix, of NIDS and HIDS. This information is processed by ML clustering algorithms to join entities in groups based on their similarities. Clustering was chosen because it allows the detection of outliers without knowing *a priori* how a malicious entity behaves, as in signature-based systems. This process is performed using time-windows, once attacks have different duration and using too much information can attenuate the detectable effects of an attack. Such time-windows represent the duration of a day (24h) divided, for example, in 24 windows each one of 1 hour. In our experiments, we used five time-windows. Several clustering algorithms were tested and the ones with better performance and accuracy were used to perform clustering, namely KMeans, Agglomerative, DBSCAN, and LOF. Several approaches were used based on the number of features given to clustering.

One of those approaches consisted of choosing a high amount of features. Such an approach was considered to avoid missing important behavior and not focus only on well known malicious behaviors. From those features, two-thirds were chosen dynamically based on the most used ports in a processed time-window. Once clustering was performed, the next step was to classify data using a defined classification pattern. This pattern was designed to identify single clusters and apply LOF in the DBSCAN outlier cluster. Our classification pattern was chosen assuming that most of the machines in the network will have 'normal' behavior. Later we used our own method of ensemble clustering in order to improve results.

The process proposed was able to detect several attacks, using either host features or network features. The best results were obtained using network features. From this approach, we were also able to conclude that the use of a high number of features leads to feature weight reduction in clustering. For example, an FTP attack will use port 21 and is expected to find a high number or length of packets sent/received in this port. Considering more 10 features, the values of this port (21) will be considered relevant in determining the clustering. However, using more 100 features port 21 will have less weight in clustering. From host features, our conclusions were limited by the topology of events, which only contained system events. The ensemble method allowed an improvement in results, in some cases, reducing false positives in more than 70% (Table 8). The combination of both, events and network features, led to not detect any victim or attacker. The main contributions of our work to intrusion detection based on clustering are the use of a high number of features, allowing the detection of attacks that might be missed with fewer features (e.g., without considering certain ports) and an ensemble of clustering and outlier schemes to better detect intrusions.

1.3 Report Structure

This document is composed of eight chapters. The second chapter describes the concepts and components of the base framework that will be used. The third chapter introduces work already developed by other authors. Chapter 4 describes the datasets used and created. In the fifth chapter, we analyze algorithms, to choose the better ones. Chapter 6, present proposed solution and decisions made. In Chapter 7 we show results obtained in all approaches made. The last chapter, 8, presents the conclusions drawn from the various approaches taken.

2 Background

This chapter presents important concepts to understand the choices made and the work developed. It is also mentioned some literature in this field, which contributed significantly to develop this work. The present chapter is divided into the following sections: the first defines some ML methods and the second describes the libraries used.

2.1 Machine Learning Methods

The significant increase in data production and complexity made human abilities insufficient to deal with such information. This is the main reason why Machine Learning is used.

In 1959, Arthur Samuel, defined Machine Learning as *"the field of study that gives computers the ability to learn without being explicitly programmed"* [33]. Since then Machine Learning has been used in many applications such as prediction, medical image analysis, human activity recognition, machine translation, self-driving cars, security analytics, and others. Nowadays we can find usage of Machine Learning almost everywhere.

Machine Learning consists is learning from data [16], traditionally an input data is given to the algorithm and output is expected. The outcome measurement can be quantitative (such as fuel prices) or categorical (such as odd/even). It is performed in two phases [6]: training and testing. ML techniques can be applied to solve several problems, for example, it is used for classification, prediction, clustering, and others.

To understand how ML techniques are applied, we present some common steps. The following steps correspond to classification problems:

1. Identify features and classes of training data;
2. Identify a subset of attributes needed for classification;
3. Learn the model using training data;
4. Use trained model to classify the unknown data.

After the training phase, a few models could be chosen. To select the model which leads to better results, it should be provided a third separated data set to perform validation. This operation indicates the presence of a new phase, leaving the ML approach divided into three phases: training, validation, and testing. The prediction model is based on a set of features given in the training data among with the outcome for a specific example. The accuracy of this predictive model will determine the degree of confidence of the outcome achieved. Once the prediction model is built based on the training data, the quality and precision of this data are essential to achieve a good prediction, as well as the choice of the most relevant features.

It is usual to confuse the concept of ML and Data Mining (DM), according to [6] DM focus on discovering unknown properties in data, without a specific goal. It tries to find new and relevant knowledge. ML has two major categories: supervised and unsupervised, which are explained in the next two sections.

2.1.1 Supervised Machine Learning

This category of ML has the objective of the mapping relation between labeled inputs and outputs. In order to achieve their objective, these algorithms must receive a labeled data set for training and as a result, provides a model to map the relation between input and output.

There are several methods to perform this technique, which is commonly used for solving classification and regression problems [16]. The Regression model has their outcome focused on measuring quantities

and can be applied, for example, to detect prostate cancer. For prostate cancer, the process consists of defining features that can indicate the presence of prostate cancer antigen and collecting clinical measurements from patients, in order to find the correlation. With this data, it is expected to get a quantitative outcome measurement indicating the quantity of prostate cancer antigen. Classification problems are normally used to solve binary problems where the outcome is a known class, for example, email spam. For email spam, it is necessary to analyze the number of times in which certain words appear in the email and based on it the outcome should be spam/not spam.

According to [11] the most popular supervised Machine Learning methods are Artificial Neural Network (ANN), Support Vector Machine (SVM), Decision Trees, Bayesian network (BN) and Hidden Markov Model (HMM). There are other methods, nevertheless, the following ones are the most relevant to this work.

Artificial Neural Network [11] is a method that provides a model that transforms inputs into outputs based on nonlinear information. This information is provided from "hidden" layers and those are composed of a connected group of artificial neurons. Figure 1 represents an example of a two "hidden" layers ANN, respectively layer 1 and layer 2. Layers are composed of neurons, in this specific example, layer 1 as three neutrons and layer 2 as only one neutron. Each neuron as an associated weight so the output of a neuron is determined by a transfer function that uses weights and inputs.

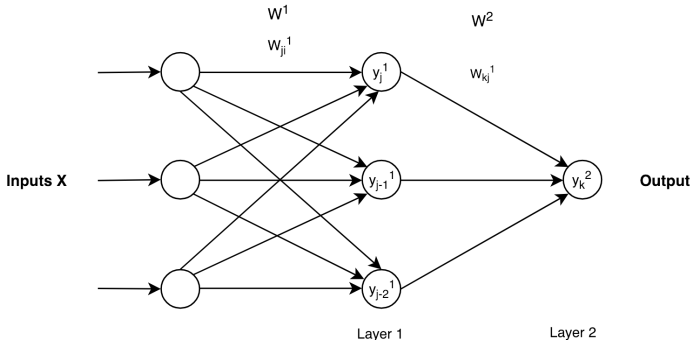


Figure 1: Example of two-layer ANN adapted from [11].

Support Vector Machine [11] consists of separate data points by an n-1 dimensional hyper-plane projected in *n-dimensional* feature space. It finds the separation between two classes, determines which points will be the support vectors[1] and set two margins. Each of these margins will dictate if a certain point belongs to class 1 or class 2. If a point is placed along the boundary, the algorithm will arbitrarily decide which class it belongs to.

Decision Trees [11] is a structural model with tree format. Has almost every tree, decision trees have leaves and branches. In the Machine Learning context, leaves represent classifications or decisions and branches the combinations of the possible features.

Hidden Markov Model [6] is based on the Markov chain. A Markov chain is the interconnection of a set of states through transition probabilities which dictate the model topology. HMM [2] is a statistical model for a system assumed to be a Markov process with unknown parameters. The big challenge of this model is to use observable parameters to find the hidden ones. Different states of HMM represent non observable conditions with different output probability distributions. Those probabilities can change over time, enabling the system to represent non-stationary sequences.

Bayesian network [11] uses a graphical model to represent factored joint probability distribution. This helps decisions about uncertain variables. The base of this classifier is the Bayes rule.

***K*-Nearest Neighbor** [11] consists of assigning data points with the label which provides the high-

[1]Class points closest to the boundary. Each class has at least one support vector.

est confidence among k data points nearest to the query point. To maximize the performance of this algorithm, two important considerations should be taken: the choice of k and the distance measure. A good practice for the determination of the number of neighbors (k) is to define it lower than the square root of the total training patterns. To deal with two classes classification problems, it is helping-full to select an odd number of neighbors to avoid tie votes.

The most used distance measure is Euclidean distance, which consists in receiving two data points in m dimensional feature space, for example, $x_1 = (x_{11}, ..., x_{1m})$ and $x_2 = (x_{21}, ..., x_{2m})$ and perform:

$$dist(x_1, x_2) = \left(\sum_{i=1}^{m} (x_{1i} - x_{2i})^2 \right)^{0.5} \tag{1}$$

The advantages of using this algorithm are easy implementation and interpretation. Nevertheless, it needs much time and storage capacity.

With these brief definitions, it is expected a better understanding of the methods when applied to cybersecurity.

2.1.2 Unsupervised Machine Learning

Contrary to the previous section, this category of Machine Learning receives no labeled data during the training phase of the algorithm. Unsupervised ML goal is to find fundamental features on unlabeled data and form clusters of input patterns based on a specific cost function. This category of ML is very useful to detect unknown attacks because it groups unlabeled data.

Popular methods used are [11] k-Means Clustering, Expectation-Maximization (EM), K-Nearest Neighbor (KNN), SOM ANN, Principal Components Analysis and Subspace Clustering. To understand some definitions presented, it is important to know the clustering concept. It consists of assigning objects with similar features in groups (clusters) and their similarity is based on group proximity from other groups.

***k*-Means Clustering** [11] is the assignment of X data points into k clusters. It consists of finding the most similar centroid for each point. The data point distance to centroid indicates how similar they are. This algorithm can be performed into 4 steps:

1. Selection of initial cluster centroids (k);
2. Assign each data point to the cluster with the nearest centroid;
3. Recompute each cluster centroid based on the elements contained in it;
4. Repeat steps 2 and 3 until achieve convergence.

To provide a successful implementation of K-Means method is important to carefully choose two aspects: the k value, which indicates the cluster number to be partitioned and the distance metric, where the most commonly used is the Euclidean distance. There are no guarantees that selected k is optimal, nevertheless, several approaches to evaluate the clustering performance have been done using stability, accuracy and other metrics.

Local Outlier Factor (LOF) [5] is a density-based method, which uses the nearest neighbor search. This method gives a score to each data point by determining the ratio of the average densities of neighbors points to the density of the point itself. The estimated density of a certain point n is the number of its neighbors divided by the sum of distances of the neighbor's points. For a better understanding consider $M(n)$ as a set of neighbors of point n and k the number of points in this set. The distance of a point y

to point n is represented by $d(n, y)$ and their estimated density is given by:

$$f(p) = \frac{k}{\sum_{y \in N(n)} d(n, y)} \tag{2}$$

Using the previously defined equation, it is possible to define the LOF score as:

$$LOF(n) = \frac{\frac{1}{k} \sum_{y \in N(n)} f(y)}{f(n)} \tag{3}$$

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [12] is an algorithm designed to discover clusters and noise in a dataset. The ideal case is to know the appropriate parameters *Epsilon* and *MinPts* of each cluster. The *Epsilon* parameter defines the density of points needed in a point neighborhood to be considered as a member of a cluster. The *MinPts* represent the minimum number of points needed for a group to be considered a cluster. Without knowing these parameters *a priori*, this method starts with a random point n and determine all points density-reachable from p considering *Epsilon* and *MinPts*. If this point is a border point (point in the ends of a cluster) the algorithm moves to the next point in the dataset. Otherwise, if point n is a core point (point in the center region of a cluster), these procedures yield a cluster. This procedure is performed until all points of a dataset are covered. Once it started with parameter global values, it may merge two clusters into one if their distance to each other is 'close'. If the distance of a point to all clusters is too large, it is considered noise. Those noisy points are merged into a single cluster, normally represented by -1.

Agglomerative [38, 48] consists of recursively merging pairs of clusters that minimally increase a given linkage distance. Similarly to KMeans, Agglomerative can use a parameter k that defines the number of clusters to find. The linkage distance is used to determine the distance between sets of points and this algorithm main goal is to minimize this distance. This algorithm starts by considering all data points as an individual cluster. In each iteration merges similar clusters into each other. The stop criteria are when all data points are in the same cluster or when k clusters are formed. This algorithm can be defined in four steps:

1. Compute the proximity of individual points;
2. Consider each data point as a cluster;
3. Find the two closest clusters and merge them into a single cluster;
4. Recalculate clusters proximity;
5. Repeat steps 3-4, until achieve a single cluster or k clusters.

Self-Organizing Map Artificial Neural Network [11] provide a low-dimensional view of high-dimensional data, by keeping neighborhood characteristics of the input data. In other words, this method forms a semantic map, where it approaches the closest similar samples and moves the different samples. The similarity can be seen as the Euclidean distance between the weight vector of neighbor cells. The SOM algorithm can be performed in six steps:

1. Initialize neutron weights;
2. Randomly select a vector from training data for the lattice;
3. Compares the input vector with the neuron weights and find the most matching;
4. Search for neurons inside the neighborhood of neurons obtained previously;

5. Adjust the weights of each neighbor neuron found in step 4, in order to improve the similarity between input vector and these neurons;

6. Run the steps 1 through 5 until achieve convergence.

SOM ensures that the topological relationship between input vectors is maintained.

Expectation-Maximization [11] methods apply two steps and assume that the distribution of data set points can be described by parametric statistical models. Expectation and Maximization represent the two steps made iteratively. The first, expectation, determine the log-likelihood concerning the actual estimate distributions for the latent variables. The second, maximization, calculates the parameters which maximize the expected log-likelihood found in the previous step. These two steps can be described as follows:

1. (Expectation) Considering a sample data d and undiscovered or missed data z, it is possible to determine the expected log-likelihood function of parameter θ by θ^s:

$$f(\theta|\theta^s) = E[logL(\theta; d, z)] \tag{4}$$

2. (Maximization) With the estimated parameter of log-likelihood function at step s, the maximum is achieved through:

$$\theta^{(s+1)} = \arg\max_{\theta}(f(\theta|\theta^s)) \tag{5}$$

EM algorithm will perform these steps iteratively to achieve convergence, which occurs when the estimated parameters remain unchanged.

This algorithm can provide a high degree of learning accuracy, when is applied to data sets with the same distribution has assumed. Otherwise, algorithm accuracy will be lower, because the model is tendentious.

Principal Component Analysis [11] consists of representing raw data in a lower-dimensional feature space, in order to retrieve maximum useful information. Consider a given multidimensional data set x represented as a matrix X where columns represent a feature and rows a data point. This matrix is represented as the transpose of x. After matrix creation data points are centered by subtracting the mean of all rows in the matrix X. This will leave PCA more tangible facing the differences between features. Then it is necessary to calculate the empirical covariance matrix. Once it is done, compute eigenvectors in multi-dimension and the respective eigenvalue. Then the eigenvectors are ranked by eigenvalue and it is found the significance of variance along with the different directions.

This algorithm can be applied in four steps:

1. Produce a data set of mean zero, by subtracting the mean in each dimension;

2. Determine the covariance matrix;

3. Determine the eigenvectors and eigenvalues of the covariance matrix;

4. Understand the significance of each component ordering the eigenvectors by eigenvalues from highest to lowest.

PCA assumes that the input data is distributed continuously and normally, otherwise, it can fail when data spread is complicated manifold.

Subspace Clustering [11] is the process similar to feature selection. Nevertheless, this technique focuses on searching for a localized subspace and is useful to remove incoherent data from clustered data.

The process to find this data goes through the representation of histogram charts with dimensions as coordinates references.

This algorithm as two categories: top-down and bottom-up. The bottom-up search methods use a priori style approach, where for n units in d dimensions will be projected in d-1 dimensions. This category, bottom-up, as several methods such as CLIQUE, cell-based clustering, and CLTree. CLIQUE forms a cluster, removing the duplicate input data from the cluster and represent it as a disjunctive normal function. Cell-based clustering uses an index structure for recovery data contained in clusters. The CLtree evaluates each cluster separately, unlike previous methods. In top-down search methods, we have many different types of methods such as PROCLUS, ORCLUS, and COSA.

2.2 Libraries

This section describes the libraries and their capabilities used to create our system. Our system was developed using Python (v3) and some of its libraries. Python was chosen because it has several libraries to perform data analysis, as well as ML.

2.2.1 Pandas

This library is an open-source and designed for data analysis [29]. It provides high-performance and easy-to-use data structures. Pandas allow data manipulation with integrated indexing in a fast and efficient object called DataFrame. It provides tools that make it possible to read and write data in several different formats, for example, CSV, text file, Microsoft Excel, SQL databases, Feather and fast HDF5[2] format. It also allows flexible reshaping and pivoting of data sets, as well as, high performance joining and merging data sets. Another powerful tool is the *group by* engine, which allow aggregations or transformations in data. This library provides an easy way to insert and delete columns from data structures. In terms of performance, Pandas has critical code paths written in Cython[3] or C.

2.2.2 Scikit-learn

It provides efficient and simple tools for data analysis. This library [34] is built based on three other libraries: NumPy[4], SciPy[5], and matplotlib[6]. It provides tools for ML in Python. This library has tools for Classification, Regression, Clustering, Dimensionality Reduction, Model Selection, and Preprocessing.

Classification tools are used to identify the category of data. This can be used, for example, to detect spam. These tools implement some algorithms such as SVM and nearest neighbors.

Regression tools help to predict a continuous-valued attribute associated with data. This can be applied, for example, in stock prices. Some algorithms implemented are Ridge regression and Lasso.

Clustering tools are used to group data in similar sets. This is a critical tool for our system. It implements some algorithms, such as KMeans and DBSCAN.

Dimensionality Reduction tools are used to reduce the number of random variables to consider. It is helpful to the visualization of data and can be used to improve efficiency. Such tools implement the following reduction algorithms: PCA, feature selection and non-negative matrix factorization.

Model Selection tools are used to comparison, validation and parameter choice and models. These tools can be used to improve accuracy via parameter tuning. It is composed of three modules: grid search, cross-validation, and metrics.

[2]Hierarchical Data File

[3]An optimising static compiler for both the Python programming language and the extended Cython programming language.

[4]Is the fundamental package for scientific computing with Python.

[5]Is a Python-based ecosystem of open-source software for mathematics, science, and engineering.

[6]Is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Preprocessing tools are used for feature extraction and normalization. This tool is important to find a way to normalize all data into a common scale without losing features critical information. This tool is composed of two modules: preprocessing and feature extraction.

2.3 Summary

The first chapter presented a general overview of ML. The next two sub.chapters described, respectively, supervised and unsupervised ML, as well as some of their algorithms. Since ML algorithms were used, it was necessary to define the most relevant ones. The implementation of this work was developed in Python and using its libraries, namely Pandas and Sckit-Learn.

3 Related Work

The previous chapter gave a brief definition of known Machine Learning methods to understand the work. IDSs have evolved from human analysts to supervised ML algorithms and actually, many of them are based in unsupervised ML. This chapter will complement the previous one by providing the application of some of those methods in the process of detecting intrusions either host or/and network-based.

Some papers presented have been published over 10 years ago, nevertheless, they remain useful and important for security analytics. The papers are presented chronologically to provide an overview of the evolution in intrusion detection systems, either network-based or host-based. Those papers present systems which use different ML algorithms to detect intrusions. The proposed process will neither be anomaly nor misuse, it will receive data without the knowledge if is benign or malicious. Nevertheless, anomaly detection systems are more similar to the approach which will be used. Since the proposed system is a combination of host-based and network-based systems, the present chapter is divided into anomaly detection, host-based intrusion detection, network intrusion detection, and hybrid intrusion detection.

3.1 Anomaly Detection

Anomaly detection consists in processing data and finding patterns which do not correspond to the expected behavior. This process is very useful for several cases; in this work, it will be useful for network monitoring and computer security. Nevertheless, there are other cases where anomaly detection could be useful, such as smart devices and IoT. Anomalies can be categorized in [17]: point anomalies, if an individual instance is considered anomalous through the rest of the data; contextual anomalies, if the anomalous instance emerge in a specific context are classified as contextual or behavioral attribute; collective anomalies, when a set of related data is anomalous through the remaining data. The major anomaly detection algorithms need purely normal data to train the algorithm and consider any data never observed as anomalous. The outliers are defined based on measurements that normally correspond to the distance from data considered normal.

In 2003, [20] made a comparative study to evaluate different anomaly detection methods. They used two data sets, 1998 DARPA Intrusion Detection Evaluation Data [23] and their real network data from the University of Minnesota. They studied four approaches:

- Local Outlier Factor (LOF), which consists in assigning to each data point a degree of being an outlier;
- Nearest Neighbor (NN), which is KNN with k=1;
- Mahalanobis-distance based outlier detection consists in determinate the mean μ of a particular point p and compute the distance as follows:

$$d_M = \sqrt{(p-\mu)^T \times \sum^{-1} \times (p-\mu)} \tag{6}$$

- Unsupervised SVM, is trained with unlabeled data and consist in separate the entire set of training data from the origin. This means smaller regions with many data points will be considered a class and points in another region will be labeled as another class.

The 1998 DARPA data set has some issues associated. It has no presence of background noise (always present in real data) and usage of scripts and programs collected from various sources. They divided features into three categories: Content-based, extracted from *tcpdump* data; Time based and Connection based. The procedures taken to execute the approaches were:

1. Feature extraction using *tcptrace* for content based and then construct time based and connection based;
2. Standard normalization of features;
3. Identification of burst attacks in data;
4. Perform separated tests from data with burst attacks, mixed bursty attacks and non-bursty attacks;
5. Repeat step 4 through the different approaches.

The approaches were evaluated with different data (bursty, mixed bursty and non-bursty). The criteria were the relation between the detection rate and the false alarm rate. The results for bursty attack showed that LOF was the most consistent, despite being worse than NN for false alarm rate lower than 2%. For mixed bursty attacks in data, they had two or more bursty attacks overlapping. LOF, NN and Mahalanobis-based approaches were unable to detect those attacks. Unsupervised SVM was the only approach to detect 2 of 3 attacks. Nevertheless, this algorithm had the highest false alarm rate of 4%. For single connection attacks data, unsupervised SVM achieved the best detection rate, but again with the highest false alarm rate (4%). They exclude the unsupervised approach due to a high false alarm rate when compared with the remaining. LOF approach provided the best results. The authors performed another evaluation with real network data from the University of Minnesota. Once LOF performed better previously, it was the only approach tested with this data and revealed to be a great promise to detect novel intrusions.

In [6] the authors survey several papers about ML and DM methods for cybersecurity. The authors mention the approach, methods, datasets, and results. A method could be evaluated by its accuracy; the time for training a model; time for classifying an unknown instance with a trained model or understandability of the final solution. This paper also defines a hybrid approach which combines, anomaly detection and misuse detection. The authors also present the cyber problem of ML and DM methods as being the regularity in which a model needs to be retrained and the availability of labeled data. They have identified the availability of labeled data as the major gap in the developed models.

A new approach was proposed in [49], combining unsupervised and supervised ML and an analyst, resulting in what they introduce as the analyst-in-loop security system. The system proposed has four components: a big data behavior analytics platform, a group of outlier detection methods, an interface for analyst feedback and a supervised learning component. Their system was tested over 3.6 billion real-world logs. The performance was compared with an unsupervised outlier detection method and found a detection rate improvement of 3.41 times, as well as a reduction of five times at false-positive rate.

There are several anomaly detection systems already developed using both supervised and unsupervised ML, proving that this approach can achieve great results fighting against the attacker's new techniques. NIDSs and HIDSs can both use the anomaly detection approach to find malicious behavior.

3.2 Host Based Intrusion Detection

With the encrypted traffic increasing in networks, NIDS are losing strength against intrusions. Once several attacks are directed to hosts, it is useful to monitor the events produced on the local host (user's computer). This extends the intrusions defense perimeter. Host-based Intrusion Detection System can operate on two levels depending on the information retrieved from the host. These two levels are [50]: operating system-level intrusion detection and application-level intrusion detection. The first one gathers information about low-level system operations, such as system calls, file system modifications and user logons. According to the OS, the auditing system is different and this is where the audit logs are stored. On Windows, this system produces three types [50] of event logs, security log, system log, and

application log. Security logs consist of tracking security events as logon, logoff, changes in access right, and system startup and shutdown. It also tracks resource usage as creating, opening or deleting files. System logs track event logs related to Windows services and drivers. The application log record logs from applications, these records are defined by the developer and could contain information about errors or problems with application configuration. Windows OS also provides tools to filter audit and analyze logs.

The application-level intrusion detection is based on data produced by the application and can be read from log files and other similar sources. New techniques have arisen to approach the IDS and the monitored application. Audit data from applications is very reliable, rich and focused. The application logs can be found in the audit system as an operating log, nevertheless, some detailed information about application errors is found in application-specific log files.

For the operating system and application levels, we could have three approaches to define our IDS [50]:

- Misuse approach is based on attack signatures and consists in analyze the stream audit and compare to look for evidences of an attack;
- Anomaly-based approach which complement the misuse approach, is based on models of normal behavior of users and applications, so if emerges any deviation it will be considered as an attack;
- Specification-based approach which defines modules built from static analysis of application code or written manually.

Yeung and Ding [54] proposed possible intrusions detection based on program and user profiles built from normal data. They focus on Unix system calls and shell commands, and used both dynamic and static approaches. The dynamic approach was based on HMM and *maximum likelihood*. The static approach was based on event occurrence frequency distribution and the principle of *maximum cross entropy*. Based on *novelty detection* they found that dynamic approach was better for detecting malicious activity based on system call data. Nevertheless, the static approach proved to have a higher performance for shell command data. Their static approach also proved to consume fewer resources than similar approaches developed by other authors. They lack on focusing only on system calls and shell commands, leaving other possible intrusions undetected. To reduce this vulnerability our system does not focus on a single source of host data.

In 2002 [51] examined several Host-based Intrusion Detection System and studied their response to mimicry attacks. A mimicry attacks allow an attacker to cloak their intrusion in order to avoid detection systems. They developed a theoretical framework with the main goal to identify whether a malicious system call trace or an equivalent variant passes throughout the IDS without triggering any alarm. This allowed the authors to evaluate the security of an IDS against mimicry attacks. They applied their framework to the IDS proposed by [44] and experimentally confirmed the success of mimicry attacks in that IDS. The authors conclude that is not enough to simply protect against today's attacks, an IDS must be adjustable and detect novel attacks. This system was only tested against mimicry attacks, nevertheless attacks variety is increasing. For this reason, we proposed a system which does not look for a specific attack but focus on host behaviors.

In April 2014 a new approach was proposed by [8], introducing the concept of contiguous and discontiguous system calls patterns in HIDS to reduce false alarms. This method applies a semantic structure to low-level system calls such as the kernel ones, emerging the intrinsic activities hidden behind high-level programming languages. Various decision engines were used to evaluate KDD98[7], UNM[8] and a new

[7]Dataset used for The Second International Knowledge Discovery and Data Mining Tools Competition.
[8]University of New Mexico dataset from 2013.

modern dataset, ADFA-LD[9]. They used ELM methodology and were able to achieve a rapidity decision engine training using the proposed scheme. This new method proved to be resilient to mimicry attacks and to have a high level of portability between different OS versions.

Based on the Windows Audit Log facility, [4] presented a HIDS based on a Logistic Regression (LR) model. They used enterprise audit logs and sandboxed virtual machine based runs on a set of malicious and benign binaries. The authors found that audit logs do not directly record determined malicious behaviors. Nevertheless, their results proved that it was robust enough to detect most malware and was a low-cost signal to detect malicious behavior. Their model achieved a detection rate of 85% at an expected false positive rate of 1% and detect 80% of malware missed by anti-virus systems. Their system leads us to try to catch these unrecorded behaviors by analyzing not only host logs but also network information.

[15] presents a system to detect host misbehavior in a large number of logs. Their approach consists of two phases. In the first phase they define and configure the detection mechanisms by performing 6 steps:

1. Defining how logs data will be normalized;
2. Selecting features and defining the time period for feature extraction;
3. Defining how features are extracted;
4. Extracting features and normalizing logs;
5. Performing clustering in group entities;
6. Defining classification.

The output of this first phase was used in the next for classification. The second phase was the execution of the system at runtime. It was divided into 4 steps: extraction of features, clustering, clusters classification and if classification were successful the system was updated, otherwise it has to be analyzed manually and updated the system. Their system had to deal with Big Data and for that MapReduce was used. To perform clustering they used the EM algorithm because it does not need prior knowledge of the distribution of each feature. The number of clusters is given as input to the EM algorithm. In cluster analysis, they labeled cluster features as primary, secondary and non-relevant. The primary features represent the best characterization of the cluster, based on their deviation from the cluster. The secondary features are similar to primary ones but for a different range. The non-relevant features are the ones that do not characterize the cluster. The author chooses the Naive Bayes algorithm for classification since it converges faster than other models and needs a smaller dataset to obtain accurate results. Their system was not accurate enough to deal with real-time detection but provided analysts with precious security information from logs.

3.3 Network Intrusion Detection

Nowadays the concept of network is used all over the world and the evolution of technology allowed people to use their phones and computers to access sensitive information from anywhere since an internet connection could be found. This gives people great freedom and saves time, but also increases significantly the risk of having intercepted or compromised data. For this propose, the papers presented have contribute to understanding NIDS state of art.

[9]Australian Defense Force Academy Linux Dataset, also available Windows Dataset and Windows Dataset for Stealth Attacks Addendum. Released in 2014.

An approach based on unsupervised ML was presented by [22]. They proposed fpMAFIA, a density-based and grid-based high dimensional clustering algorithm used for big data sets. They made a detailed and complex analysis to find that the system scaled linearly with the number of records in the dataset. The authors used the KDD CUP data set to train and test their detection system. Their approach achieved a reasonable detection rate and provide a detection scheme applied for network intrusion detection systems. They restricted their algorithms to density and grid-based while our system combines density, partition and hierarchical clustering algorithms.

Teles [45] made an extensive study testing DM applied to network anomaly detection using real network logs from the Portuguese Navy. They used five datasets, three from the network firewall in the navy network and the other from logs with classifications from the perimeter firewall. They provide two approaches, one based on supervised learning and another on unsupervised learning. In the first one, supervised learning, they had to collect new data due to an over learn in the classifier, nevertheless, the results were above the expectations. In the unsupervised approach, they found that is possible to cluster connections with a high percentage of accuracy, considering only the distance between clusters. They tested algorithms by themselves not considering a possible improvement of merging clustering results from those algorithms. Our system provides our clustering ensemble method.

[30] presented a system combining supervised and unsupervised machine learning to detect network anomalies automatically. Their system was composed of five phases: feature extraction, clustering to aggregate entities, genetic algorithm to select the most relevant features from clusters, classification and output the misbehaving groups along with the relevant features. This system revealed to be significant in reducing the number of entities to analyze, nevertheless, they had a lack of contextual information about the networks and it could be a great improvement in mitigating the abnormal activity. Their approach was used to compare with our, results are shown in Chapter 7.

Farooq and Otaibi, [13] presented several examples of ML applied to cybersecurity monitoring. First, they present several clustering algorithms for data rate analytics, such as KMeans, DBSCAN, BIRCH. The DBSCAN algorithm creates cluster based on two input parameters, the radius of the neighborhood and the minimum number of points which must exist in the cluster neighborhood. BIRCH is an algorithm that creates a multi-level hierarchical clustering and builds a clustering feature tree, each cluster in this tree is composed of three parameters: number of data points, linear sum, and square sum. They secondly present anomaly detection in process executions using one-class SVM classifier and used Sysmon do extract features as input for the classifier. The authors predict user behavior using LR and finally used message classification using random Forest. They conclude that semi-supervised algorithms will replace supervised learning, due to easier training and effectiveness. They focused on supervised learning, nevertheless unsupervised ML had already proved to be an effective way to detect anomalies. This fact reinforced us to only consider unsupervised ML algorithms.

3.4 Hybrid Intrusion Detection Systems

This chapter presents some literature about IDSs that combine both network and host-based intrusion detection. This combination seems to be recent as few articles have been published.

The most recent hybrid IDS is presented in [26]. Authors proposed a system entitled HOLMES with the main goal of detecting Advanced Persistent Threats. Their system starts by collecting audit data from the host to produce a detection signal. This signal will indicate the stages of the current APT campaign. At high-level, HOLMES techniques provide the correlation between suspicious information flows. Those flows derived from events produced during the attacker campaign. HOLMES provides three main contributions, which are summarized in the following paragraphs.

HOLMES help to map activities derived from host logs. The system also maps alerts found in the

enterprise kill chain. The system design enables the generation of alerts semantically close to the activity steps of APT attackers. The authors collected five days of audit logs which almost 3M of low-level events. Their system only found 86 suspicious activity steps from those 3M events.

The second contribution is the information flow from low-level entities in the system a base for alert correlation. The internal reconnaissance step depends on successful initial compromise and establishing a foothold. Usually, reconnaissance involves running malware downloaded earlier when establishing a foothold. Both lateral movement and exfiltration phases, use data gathered during the reconnaissance phase. Correlating APT steps from low-level events and information flow originates the emerging kill-chain used by a particular APT actor.

The third contribution is the construction of a High-level Scenario Graph (HSG). The graph nodes refer to Tactics, Techniques, and Procedures. The edges of the graph represent the information flow among TTPs and their entities. The authors develop new concepts related to HSG:

- Ancestral cover - help to assess dependencies strengths between nodes, pruning weaker dependencies. This reduces severely false alarms;
- Noise reduction techniques - de-emphasize dependencies known to be associated with benign activities;
- Ranking and prioritization techniques - withdraw nodes and edges unrelated to APT campaign.

The last contribution is a visual summary provided by HSG. This summary can be visualized at any moment of the APT campaign. It is a powerful tool for cyber-analysts since enables faster and better attack understanding.

HOLMES evaluation was performed on 9 real-life APT attack scenarios. It was running as real-time IDS and has a duration of two weeks of the live experiment. The results show high precision and recall discerning attacks from benign scenarios.

3.5 Summary

Related work developed in IDS was presented with several different approaches. It began introducing anomaly detection concepts, then a comparative study followed by ML and DM methods for cybersecurity. Anomaly detection ends presenting a new system that combines supervised and unsupervised ML. The next section approaches HIDS already developed by other authors. This type of intrusion detection was described in detail and then six different works are presented. The approached works show multiple techniques can be applied to achieve good results. NIDS section present systems developed to improve intrusion detection using network logs. Two in four approached systems compare different ML methods to monitor network activities. This last section (Hybrid IDS) comes closer to the present work. It analyzes host events, but also network events occurred in the host. Table 1 summary the important aspects of literature used to understand the state-of-art.

Paper	Publisher	Year	Approach	Dataset	Conclusion
Host Intrusion Detection System					
Host-Based Intrusion	Missing	2005	Explain host logs sources for security analytics, present existing detection systems and tools to help develop security systems based on host behavior	Non applicable	Quality of audit data gathered from application and OS level gave a higher importance to HIDS
A Semantic Approach to Host-Based Intrusion Detection System using Continuous and Discontiguous System Call Patterns	IEEE Transactions on computers	2014	Apply semantic structure to kernel level system calls	KDD98 UNM ADFA-LD	Resilience to mimicry attacks High portability among different OS versions
Malicious Behavior Detection using Windows Audit Logs	ACM	2015	Performed feature engineering to extract features from audit logs Used LR method as classifier	Enterprise logs Sandboxed VM based, running sets of malicious and benign binaries	Audit logs provide effective and low-cost signal for detecting malicious behaviour
Network Intrusion Detection System					
Cibersegurança: Deteção de outliers	Escola Naval	2015	Performed an extensive study on ML and DM methods applied to network anomaly detection. Both unsupervised and supervised methods were used	Naval network logs	Both unsupervised and supervised approaches showed good results for real logs from network
Online security analytics	Portuguese Military Academy Instituto Superior Técnico	2018	Collect logs from network and applied ML method K-Means Performed features selection using genetic algorithm	Portuguese Army network logs LANL	# of entities to analyze were significant reduced
Optimal Machine Learning Algorithms for Cyber Threat Detection	IEEE	2018	Present multiple ML algorithms to cyber security	Not indicated	Semi-supervised algorithms will replace s supervised one due to easier training and effectiveness
Hybrid Intrusion Detection System					
HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows	University of Illinois at Chicago	2019	Use low-level entities info from audit logs to map APT campaign stages HSG as high-level basis to detect APTs	DARPA Trasparent Computing program 9 real-life APT scenarios	Successfully detects APT attacks in real-time with great precision and low false alarm rates

Table 1: Essential literature on HIDS, NIDS and Hybrid IDS.

4 Datasets Description

The present work was developed using two datasets, an artificial dataset from CSE-CIC-IDS 2018 [36] and a real one obtained from this work from the Portuguese Army. The next sections detail both the datasets.

4.1 CSE-CIC-IDS 2018

This dataset was developed with the goal of providing data to analyze, test and evaluate IDS. To generate such a dataset, its authors developed a systematic approach in order to produce a diverse and comprehensive benchmark dataset. In their approach, they created user profiles with abstract representations of activity seen on the network.

The benign behavior of each machine was generated using a B-Profile. This profile[10] was designed to extract abstract behavior from human users. The goal is to apply machine learning and statistical analysis techniques to encapsulate network events produced by users. After generating B-profiles from users a human operator or an agent (CIC-BenignGenerator [37]) can use those profiles to generate realistic benign behavior of a network.

The authors mention malicious behavior as M-Profile, which represents seven attack scenarios. The attacks were performed from one or more machines in a specific network created only to attacker machines. The authors divided machines into five departments and a server room (Figure 2), where four departments had Microsoft Windows 8.1/10 installed. The other department had Ubuntu OS and servers room were running Microsoft Windows servers with versions 2002 and 2016.

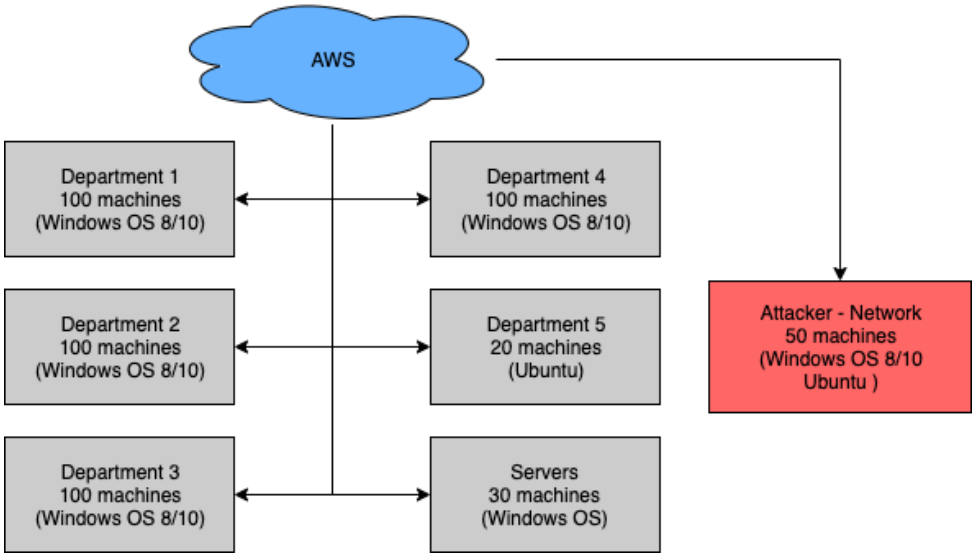


Figure 2: Network topology, adapted from [9].

The attacks performed in this dataset were chosen according to most common attacks families and executed using related tools and techniques. The attacks implemented were:

- Brute Force - is the most popular attack to password cracking, nevertheless, it can also be used to discover hidden pages and content in web application;
- Heartbleed - used to explore a bug in the OpenSSL cryptography library, that is used in Transport Layer Security (TLS) protocol. Basically, an attacker sends a malformed heartbeat request with a small payload and large length field to vulnerable party, waiting for victims response;

[10]Set of behaviors and characteristics which represent someone or something.

- Botnet - number of devices connected via the Internet, owned by a botnet to accomplish attacks (steal data, send spam and access to the device and its connections);
- DoS - consist of flooding a specific machine or resource using requests in an attempt to overload systems. The attacker tries to make a machine or network resource unavailable for some time;
- Distributed Denial of Service (DDoS) - occurs when the DoS attack is performed by multiple systems, it means several machines flooding the bandwidth or resources of a victim;
- Web - they used SQL Injection, by creating strings of SQL commands to steal information from a database. Cross-Site Scripting (XSS) to sites not tested against script injection and Brute Force over HTTP to collect information on lists of passwords to find admin password;
- Infiltration - normally performed by exploiting vulnerable software. Once exploiting is successfully done, the attacker uses a backdoor to perform different attacks on the victim's computer.

This dataset contains information about ten days of normal activity and attacks performed, as shown in Table 2. Due to timezone difference, in our analysis, attacks have occurred in a different time than the ones presented (approximately 4 hours later).

Attacker	**Victim**	**Attack Name**	**Date**	**Attack Start Time**	**Attack Finish Time**
Kali Linux	Ubuntu 16.4 (Web Server)	FTP - Brute Force	14-02-2018	10H32	12H09
		SSH - Brute Force	14-02-2018	14H01	15H31
Kali Linux	Ubuntu 16.4 (Apache)	DoS - GoldenEye	15-02-2018	9H26	10H09
		DoS Slowloris	15-02-2018	10H59	11H40
		DoS - SlowHTTPTest	16-02-2018	10H12	11H08
		DoS Hulk	16-02-2018	13H45	14H19
Kali Linux	Ubuntu 16.4 (Web Server)	Brute Force - Web	22-02-2018	10H17	11H24
			23-02-2018	10H03	11H03
		Brute Force - XSS	22-02-2018	13H50	14H29
			23-02-2018	13H00	14H10
		SQL Injection	22-02-2018	16H15	16H29
			23-02-2018	15H05	15H18
Kali Linux	Windows Vista and Macintosh	Infiltration	28-02-2018	10H50	12H05
				13H42	14H40
			01-03-2018	9H57	10H55
				14H00	15H37
				14H00	15H37
Kali Linux	Windows Vista 7, 8.1, 10 (32-bit) and 10 (64 bit)	Bot	02-03-2018	10H11	11H34
				14H24	15H55
		DDoS + PortScan	20-02-2018	10H12	11H17
				13H13	13H32
			21-02-2018	10H09	10H43
				14H05	15H05

Table 2: Summary of attacks, by day (adapted from [9]).

This dataset contains a victim infrastructure of 420 machines and 30 servers divided into 5 departments and an attacker infrastructure with 50 machines. It includes system logs and network traffic of each

machine. The authors also provide a list of features, based on network traffic, that were used to perform clustering.

Authors used CICFlowMeter to extract the features [7], a network traffic generator which offers flexibility when choosing features parameters. This tool generates bidirectional flows, which means first packet represents the forward (source to destination) and backward (destination to source) directions. It also calculates 83 statistical features (Annex B) used to apply ML algorithms to detect suspicious activity.

This dataset provides logs from machines (Windows XML Event Log (EVTX) files from Windows and text files from Linux). Our work was focused on Windows OS, so we only analyzed those. Authors left default configurations on Windows machines, which means only system logs were saved and extracted, missing information about security and applications running on those machines. As this information comes in EVTX, readable in Windows Event Viewer, we needed to extract the information to a Python readable file such as Comma Separated Values (CSV). The first approach was to use a Python script to read from EVTX[11] to JSON and then convert JSON to CSV, this was a slow process and missed some fields such as the time creation of an event. Once we needed that information from every event we had to change our approach. This second approach used a Powershell script to extract all information contained on EVTX files to a CSV. On a single step, we collect all the information needed, nevertheless, this was also a time-consuming process. Once we had our log information on a Python readable file, we could extract our host-based features. The information from network traffic was provided as Package Capture (PCAP) files and also as CSV (processed with the 83 features proposed by authors). Our approach used IP addresses to identify each entity in a network and this information was not provided in processed files. This leads us to use their PCAP files and CICFlowMeter to extract the information we need. We used the last release of CICFlowMeter (V4.0) which lead to a mistake in the number of packets sent and received in a biflow. This error was detected later with confirmation of our features and corrected by adding one packet to forward ones and subtracting one packet from backward ones.

4.2 Portuguese Army Dataset

This dataset was collected from workstations in an administrative network medium-sized military infrastructure. It contains real logs from each workstation, as well as netflow information of five days. By default, the Windows OS does not enable the most interesting logs for security. In order to collect logs at security and applications level, it is necessary to enable/change (to success and failure state) some configurations, such as: force audit policy subcategory settings, logon, network policy server, other logon/logoff events, special logon, process creation, Remote Procedure Call (RPC) Events, security state change, security system extension, system integrity and file share. The amount of space used to save logs also need to be increased to prevent losing some of them. Each day an attack was simulated in order to produce suspicious data. The attacks simulated were:

- PSEXEC script execution - script used to go through a list of IPs in a subnet, performing ping and if online execute PSEXEC to run AuditPol command and extract logging configurations;
- PingSweep - script in batch to perform a pingsweep through a subnet. This is a technique used to check which host (computer) is live in a range of IPs;
- Port Scan nmap - performed using Zenmap tool, scanned ports in range 1-65535 using T0[12] and T1[13] to avoid IDS detection. This attack was performed against servers;

[11] Windows Event XML Log

[12] Option for -T parameter also known as paranoid, used to avoid IDS evasion. It slows down the scan to use less bandwidth and target machine resources.[24]

[13] Option of -T parameter, also known as sneaky. Similitar to T0, but using a higher period of time.[24]

- Failed Logon - physical access to computer allow several attempts to logon;
- Dictionary attack - remote logon attempt using a list of users and passwords;
- Execution of PowerShell scripts - to enumerate services and their features (IP, serial number, hostname) to further exploit vulnerabilities;
- Logs removal - clean all logs from a machine, this attack simulates an attempt to clean any footprint from an attack;

Attack	Start Time	Duration	# Attackers	# Victims
Execution of PSEXEC script	25/07/2019 11:25	20 min	1	Two subnets
PingSweep 5s delay	25/07/2019 11:50	1h	1	One subnet
Remote Access and log extraction	26/07/2019	10 min	1	-
PingSweep 5s delay	26/07/2019 09H50	2h	1	IPs ended in .1 and some /24 networks
PingSweep 10s delay	26/07/2019 12H07	2h	1	IPs ended in .1 and some /24 networks
PortScan nmap - T1	29/07/2019 10H28	4h	1	1
PortScan nmap - T0	29/07/2019 14H00	19h	1	1
Failed local logons	30/07/2019 09H50	10 min	Non relevant	1
Failed remote logon	30/07/2019 16H10	20 min	1	1
Failed remote logon	30/07/2019 16H30	30 min	1	2
Failed remote logon to internal machines	30/07/2019 17H00	16h	1	All computers connected on network
Failed remote logon to external machines	30/07/2019 10H25	1h	1	Unknown
Powershell scripts execution	31/07/2019 11H45	30 min	1	506
Logs removal	31/07/2019 12H18	Immediate	1	1

Table 3: Summary of attacks, by day.

As performed in the previous dataset, we had to process EVTX files in the same way. Dataset flows were collected from a Netflow tool, which provides less information than CICFlowMeter. Nevertheless, the information provided was enough to apply our system for anomaly detection.

The produced dataset is more complete in terms of Windows logs, once it contains logs from Application, Security and System Levels, while the CSE-CIC 2018 dataset only contains logs from System level.

4.3 Summary

This chapter presented the dataset used to test our system and the produced dataset with real data. Firstly we introduced a synthetical dataset produced by CSE-CIC, which contains logs and flows from 450 machines. This dataset came with 13 different attacks and critical information about them, such as attack start time, attack finish time, IP addresses from victims and attackers. The authors also provide

information about the tools used to produce such attacks. This chapter also details the creation of a dataset from real working machines, collecting more information from host logs than in the previous dataset. This was achieved by changing the logs record policy. The Portuguese Army dataset cames with 14 different periods of attacks. Both datasets present variety in the number of attackers and victims. This unlocks the possibility of testing a system for a 1-to-1 attack, 1-to-many attack, and a many-to-1 attack.

5 Clustering Algorithm Selection

This chapter describes the experimental studies made to choose the most appropriate clustering algorithms for anomaly detection. To evaluate different clustering approaches, we tested the most commonly used algorithms in each category (partition, hierarchical, neural models and density-based). The criteria used to choose the most appropriate clustering algorithms were:

- Performance when applied to data with a big number of features (i.e., high dimensional data);
- Efficiency to identify outliers (from a well-known dataset);
- The time needed to perform clustering must be lower than the smallest window of time (10 minutes), to get results in real-time;

To apply these criteria we used a labeled dataset known to produce good results with KMeans [25] in previous work [30]. The labeled dataset contains network traffic from one day of the Portuguese Army dataset used by [30]. This day contains flows from 4616 entities, one of which is an attacker that performed a PortScan and a Dictionary attack. The day selected allowed us to evaluate each algorithm's performance with a big number of features and their efficiency in identifying attack(s).

According to clustering algorithms metrics and methods used to group data, they can be classified as partition, hierarchical, density or ANN-based. From partition-based algorithms, we selected KMeans, since it is known to produce good results in intrusion detection approaches from previous works [30, 32]. KMeans clusters are represented by a central vector (cluster mean). Consider n the number of d-dimensional vectors (to be clustered), k the number of clusters and i the number of iterations needed until convergence. To find the global optimal of KMeans is considered an NP-Hard problem, although the practical approach is to find a local optimum of KMeans which takes linear time $O(nkdi)$. Hence, in practice KMeans as linear complexity.

From hierarchical based algorithms the choice was Agglomerative Clustering [28]. It starts with each object being considered a cluster. It computes pairwise distances of n data-points and links those together according to a linkage function (e.g., minimum distance). The results are represented by a dendrogram where the root node represents the whole data set (single cluster) and each leaf node is a data object. The complexity of Agglomerative Clustering is $O(n^2)$, mainly due to the cost of computing all pairs of distances.

Regarding density-based clustering, we chose to test Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [12] and Ordering Points To Identify the Clustering Structure (OPTICS) [1]. For each point of the dataset, DBSCAN verifies and groups together points with many nearby neighbors and mark as outliers points that lie alone (i.e., appear in low-density regions). It needs as input epsilon-neighborhood (i.e.,radius) and MinPts (i.e., the minimum quantity of points within radius). In short, DBSCAN generates a new cluster from a data point by absorbing its neighborhood. OPTICS algorithm is heavily inspired in the DBSCAN technique, although this clustering technique is different in the sense that it does not explicitly segment the data into clusters. Instead, it produces a visualization of reachability distances and uses this visualization to cluster the data. OPTICS overcome the problem of DBSCAN's poor performance when clusters are varying density. Both DBSCAN and OPTICS have a time complexity of $O(n \times log(N))$ and $O(n^2)$ respectively. Another chosen algorithm was Self-Organizing Map (SOM) [19] which is a type of ANN trained in an unsupervised manner to produce a discretized representation (typically two dimensions) of the input space. It applies competitive learning as opposed to typical ANN which uses error correction learning. In short, SOM creates a low dimensional view of high dimensional data, preserving topology. SOM algorithm starts with a fully interconnected neural map (e.g., a 2-dimensional grid of neurons). Each time a new input data x is presented, the neuron with the closest

prototype (weight vector of the same size of input vectors) wins and all the weights (winner and neighbors) are updated to become closer to input data. According to [52] the time complexity of SOM is generally high and depends on the layer construction involved in the algorithm. Roussinov et. al. [31] refers that SOM is unscalable for high dimensional or large input data.

Xu et. al. [53, 52] provide a survey of clustering algorithms describing the theoretical time complexity of each clustering algorithm. They refer that KMeans and DBSCAN perform well on large-scale data. Our experimental results shown in Figure 3 are according to the theoretical results.

Most of the algorithms used were provided by scikit-learn. The algorithms used from scikit-learn were: Kmeans, DBSCAN, AgglomerativeClustering, OPTICS and LOF.

To perform tests with SOM algorithm, it was necessary to use other libraries such as SimpSOM [14], a ligthtweight implementation of SOM, and SOMPY[15]. These SOM implementations are similar, we started by using SOMPY. Once it was not very efficient we found SimpSOM, a lighter implementation with hope of increasing efficency. The Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) is based on DBSCAN with the difference of varying epsilon values, this leads to a solution that represent the best stability over epsilon.

The algorithms tested were: SimpleSOM, SOMPY, KMeans, LOF, DBSCAN, OPTICS, AgglomerativeClustering and HDBSCAN.

A benchmark from [3] was adapted to test our algorithms in terms of performance against large datasets. Each algorithm were used with default values, only SOM implementations needed grid size. This size was selected based on [47], which suggested:

$$M \approx 5 \times \sqrt{N} \tag{7}$$

where N represent the number of observations and M the number of neurons. The data for tests was generated locally and data was increased until reach 30000 observations. Figure 3, shows the performance of each algorithm with the increase of data. Observing the results it is clear that SOM does not fullfit performance criteria for large datasets. The remaining algorithms fit the criteria once they could perform cluster to 20000 observation in less than 60 seconds.

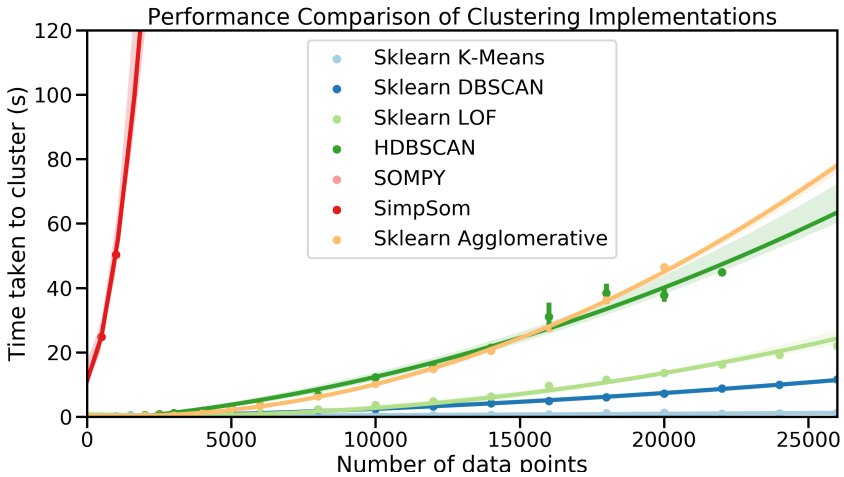


Figure 3: Algorithms performance against large data.

We applied our algorithms using [30] features to understand algorithms ability to detect outliers. Based on experiments we decided to use LOF on clusters with suspicious entities. This entities were

[14]https://github.com/fcomitani/SimpSOM
[15]https://github.com/sevamoo/SOMPY

Algorithm	Attacker Cluster	Cluster Size	Attacker LOF Score (attacker_score/highest_score)
Agglomerative	0	4	Not applicable
OPTICS	-1	2491	4/601
KMeans	13	1	Not applicable
HDBSCAN	-1	2245	4/436
DBSCAN	-1	29	2/2

Table 4: Algorithms capacity to detect outliers.

known *a priori*. Analysing Table 4 it is possible to determine which cluster performed better detecting the attacker. The first column identifies the algorithm and the second one identifies the cluster where the attacker was grouped. The third column specifies attacker cluster size and the last column represents the relation between the attacker's LOF score and the highest score of his cluster. In the cases where attacker cluster size is smaller than the number of neighbors, LOF cannot be applied. From better to worst clustering algorithm we have KMeans, which isolated attacker, Agglomerative, grouped attacker with other 3 machines, DBSCAN, found 29 outliers nevertheless attacker LOF score was the higher in outliers cluster, HDBSCAN, found 2244 outliers and attacker LOF score was far from good, OPTICS detected 2491 outliers and the scores was also far from good.

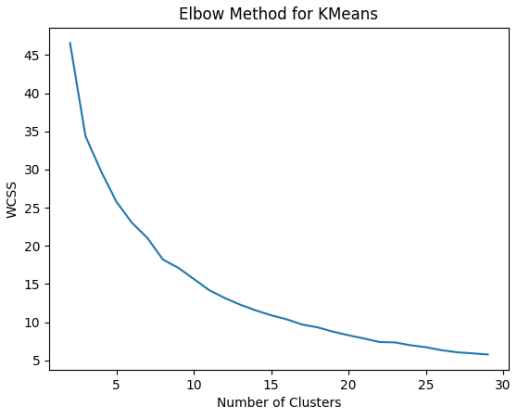
According to results obtained, we decided to use KMeans, Agglomerative, LOF, and DBSCAN. We performed clustering using KMeans, Agglomerative and DBSCAN, applying LOF to outlier cluster from the last one.

5.1 Selecting Algorithm Parameters

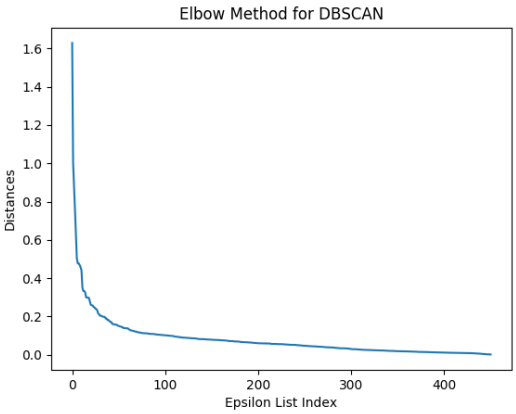
An important decision when using clustering algorithms is to correctly select the parameters. In our case, we had to choose 5 parameters: *n_clusters* for Agglomerative, k for KMeans, *eps* for DBSCAN and *n_neighbors* and *contamination* LOF. To chose parameters for KMeans, Agglomerative and DBSCAN we decided to use *Elbow Method*. The idea of this method is to test various numbers of clusters to achieve the optimal number of clusters. Increasing the number of clusters in KMeans reduce the distance between cluster, nevertheless, the distance between observations intra-clusters will increase. There is a need to find a balance which increases observations similarities inside clusters and increase differences among groups of observations. The big goal is to achieve the minimum distance from observation to the cluster center. We are trying to achieve a minimum of within-clusters sum-of-squares (WCSS). KMeans object from scikit-learn provides an attribute called inertia, which calculates WCSS. This attribute inertia as a negative point: it assumes clusters are convex and isotropic, this means if clusters are long or have an irregular format this metric will not perform so good. We defined inertia to k in the range [2,30] to understand the best k. To find the DBSCAN parameter, *epsilon*, we used the NearestNeighbors [41] module from *sklearn.neighbors* to calculate distances between entities. These module represent an unsupervised learner for implementing neighbor searches, using the following parameters: *n_neighbors*(=2) - number of neighbors to use by default; *radius* (default = 1.0) - range of parameter space to use by default; *algorithm* ('auto', 'ball_tree','kd_tree','brute')

Figure 4a shows how WCSS evolves increasing the number of clusters in a time-window of 10 min. Figure 4b shows the evolution of distances with the increasing of epsilon (x-axis represent index in epsilon list) in a time-window of 10 min. Both of these figures help to understand the best k and *epsilon*. These parameters were determined by selecting the farthest point from the straight line joining first point in the graph with the last one.

Since each time-window of data processed can have different entities and features, our data could vary significantly. This means that fixing k and *epsilon* would not be a good choice, since it could be unfitted



(a) Elbow method to KMeans parameter.



(b) Elbow method to DBSCAN parameters.

to that specific data. To reduce this problem we applied the elbow method to each time-window, for example, to 5 time-windows of 10 min we could have 5 different values of parameters. As in KMeans, to Agglomerative Clustering is also needed do define the number of clusters to find, we decided to use the value of k as *n_clusters* parameter, to compare results to equal parameters. For the remaining parameters of the algorithms, we used default values.

The DBSCAN [39] default values are:

- *leaf_size* (default=30) - represent the leaf size passed to BallTree[16] or KDTree[17]. This parameter can affect the speed of the construction and query, as well as the memory required to store the tree;
- *n_jobs* (default=None) - number of parallel jobs to run;
- *min_samples* (default=5) - The number of samples in a neighborhood for a point to be considered as a core point;
- *metric* (default=euclidean) - to use when calculating distance between entities in a feature array;

The KMeans [40] default values are:

- *init* (default='kmeans') - method for inicialization;
- *n_init* (default=10) - number of time the KMeans algorithm will be run with different centroid seeds, best output is chosen as a final result;
- *max_iter* (default = 300) - maximum number of iterations of the KMeans algorithm for a single run;
- *tol* (default=$1e-4$) - relative tolerance with regards to inertia to declare convergence;
- *verbose* (default=0) - verbosity mode;
- *random_state* (default=None) - determines random number generation for centroid initialization;
- *n_jobs* (default=None) - number of jobs to use for computation;
- *algorithm* (default='auto') - k-means algorithm to use, 'auto' chooses 'elkan' for dense data and 'full' for sparse data.

[16] Is a space partitioning data structure for organizing points in a multi-dimentional space, it partitions data points into a nested set of hypershperes known as *balls*.

[17] Is a space partitioning data structure for organizing points in a k-dimentional space.

The Agglomerative [38] default values are:

- *affinity* (default='euclidean') - metric used to compute linkage;
- *linkage* (default='ward') - determines which distance to use between sets of observation. Ward minimizes the variance of the clusters being merged;

All clustering algorithms used the *euclidean* metric to determine the distances. It determines the distance between two points in a straight line. Each algorithm has other optional parameters that allow the user to personalize it in their own way. Nevertheless, we decided to not use those parameters once we did not consider them useful to our work.

To complete our system and achieve a better understanding of the DBSCAN outlier cluster, we used the Local Outlier Factor. This algorithm calculates an anomaly score for each entity. This is performed by evaluating the local deviation of the density of a given entity with respect to its neighbors. It is considered local since the score represents how isolated an entity is with respect to the surrounding neighborhood. The entities considered outliers are the ones with substantially lower density than their neighbors. Local density is estimated using k-nearest neighbors distance, for this reason, LOF needs *n_neighbors* parameter. This method can receive other parameters, namely:

- *algorithm* ('auto', 'ball_tree', 'kd_tree', 'brute') - algorithm used to calculate nearest neighbors;
- *leaf_size* (default=30) - passed to BallTree and KDTree;
- *metric* (default='minkowski') - metric used to compute distances;
- p (default=2) - used for 'minkowski' metric. If $p = 1$, it corresponds to manhattan distance and when $p = 2$ the euclidean distance is used;
- *metric_params* (default=None) - adicional arguments for metric function;
- *contamination* (default=0.1) - the amount of contamination present in the dataset. This is used to determine threshold on the decision function;
- *novelty* (default=False) - to use LOF for novelty detection;
- *n_jobs* (default=None) - define the number of parallel jobs to run in neighbor search;

From this set of parameters we only defined *contamination* with a value of 0.01, this means we considered 1% of the entities in the dataset as possible outliers. This parameter was above the percentage of outliers in all days of both datasets, except for days of Bot and DDoS attacks from CSE-CIC-IDS dataset. This method provides an array named *negative_outlier_factor_*, which contains the score of each entity. The probability of an entity being an outlier increase with the score. In this case, the most negative score corresponds to the entity with the biggest probability of being an outlier. Entities with lower scores, namely close to -1, are considered as 'normal'. This method also provides, as an attribute, the number of neighbors used for k-nearest neighbors queries.

5.2 Summary

In this chapter, we described our criteria and tests performed to help decide which algorithms we should use and their parameters. Based on results we decided to use KMeans, Agglomerative and DBSCAN to perform clustering. Has an attempt to improve results we used LOF to attribute a score to all entities in the DBSCAN outlier cluster. Considering parameters, the most important ones were *epsilon* and k, which were determined in each time-window of data processed using the elbow method.

6 Proposed Solution

This chapter explains our approach and architecture implemented. Firstly it describes the steps we used to process data, then our method to perform cluster ensemble and improve results.

6.1 Processing Data

The present work combines network traffic and logs from hosts in the network. The use of both implies being able to manage data from at least two different sources (Figure 5). We handled data separately and merged network and host data after feature extraction. In order to perform features extraction and based on results achieved by [30] we decided to use different time-windows. The time-windows used were 10 min, 30 min, 1 hour, 2 hours and 4 hours. With these time-windows, we avoid the attenuation of attack behavior.

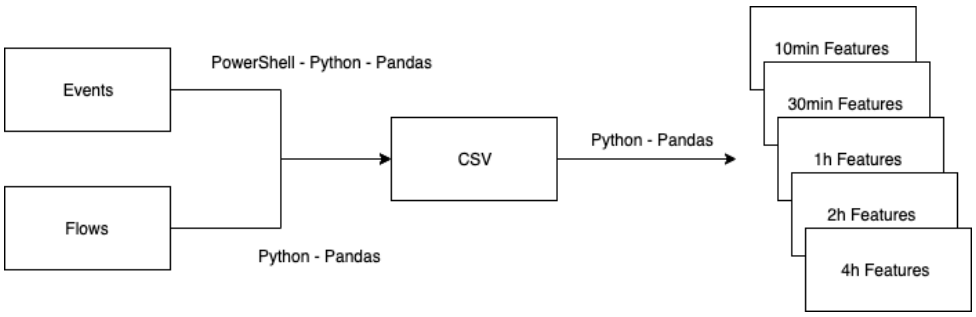


Figure 5: Processing data to later perform clustering.

This process was performed in three phases. The first one consisted of processing only events from Windows OS, the next one only flow data, and finally the combination both (events and flows).

6.1.1 Feature Engineering

To choose features we analyzed each source (host and network) independently based on the information provided by the CSE-CIC-IDS 2018. The approach on host features was to count all Windows OS events individually and their level (Critical, Error, Warning, Informational and Verbose), as well as, the total number of events. For example a machine in a time-window with only 30 events of eventID 7024 (Annex A), of which 10 in Warning level and 20 in Informational level, will have 4 features: eventID 7024 with value 30, Warning with value 10, Informational with value 20 and TotalEvents with value 30.

In relation to network features we started by setting 10 fixed features (Table 5), which were considered in all next approaches. These fixed features describe general network activity of an entity. The first approach considered the 33 ports most used in CIC-CSE-IDS dataset (Table 6) and set dynamically other 77 ports. The 33 ports were selected based on the most used ports in all days of the CIC-CSE-IDS dataset. We selected 33 ports because was the number of ports (mostly well known and registry ports) with a significant use compared to all other ports. The dynamic features emerge as an attempt to detect attacks without using only well known ports, but using the most used ports in a processed time-window. This approach selected the 77 ports most used by all dataset entities in a processed time-window. We choose 77 ports dynamically to make up a total of 100 ports as features. We wanted to test the impact of a huge amount of features in intrusion detection. We started by not defining a limit for those dynamic ports, however, the feature extraction time was high and did not fit our criteria. The second approach was to use only the 33 fixed ports and the final approach was to use [30] port features (ports 80, 194, 25 and 22) plus the ports used by the service attacked (e.g. for FTP attack we added port 21).

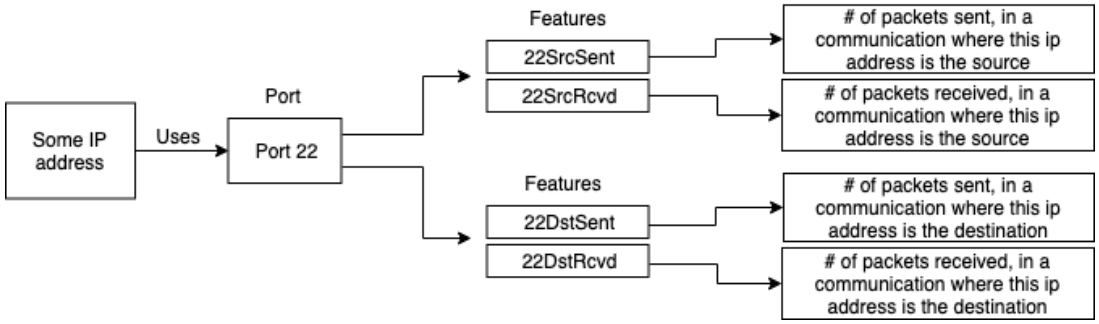


Figure 6: Example of the process of flow features extraction.

Each port taken into consideration produced 4 features. For example, for port 22 we had 22SrcSent, number of packets sent from port 22 in source view, 22SrcRcvd, number of packets received from port 22 in source view, 22DstSent, number of packets sent from port 22 in destination view and 22DstRcvd, number of packets received in port 22 from destination view (Figure 6).

For each time-window, features were extracted individually in order to avoid overlay off fixed features (Table 5). This means if port 80 was contacted once in each of two consecutive time-windows, feature *PortContacted* will count only one appears. If we had used the sum of time-windows to produce next time-window (e.g. 10min+10min+10min=30min) we will have incorrect values on fixed features (Table 5).

Feature	**Description**
IPContacted	# of different IPs contacted by an entity
PortUsed	# of different ports used by an entity
PortContacted	# of different ports contacted by an entity
TotLenRcv	Sum of total packets length received by an entity
TotLenSent	Sum of total packets length sent by an entity

Table 5: Fixed features, each one is considered from source and destination point of view.

Port	**Description**	**Port**	**Description**
Well known ports		Registered Ports	
21	FTP	1900	SSDP
22	SSH	2181	EForward-document transport system
23	Telnet protocol	2816	Unknown
25	SMTP	3128	Squid [18] caching web proxy
53	DNS	3389	Microsoft Terminal Server [19]
67	DHCP	5355	Link-Local Multicast Name Resolution[20]
68	DHCP	6188	Unknown
80	HTTP	6667	IRC
123	NTP	7112	Unknown
135	DCE RPC Locator Service	8080	HTTP
137	NetBIOS Name Service	8443	Unknown
138	NetBIOS Datagram Service	10397	Unknown
139	NetBIOS Session Service	27017	MongoDB[21] (unofficial)
194	IRC	30303	Unknown
443	HTTPS	Dynamic, private or ephemeral ports	
445	Microsoft Directory Services AD	50010	Certificate Management over CMS
500	ISAKMP / IKE		

Table 6: Fixed port features.

[18]Is a catching and forwarding HTTP web proxy.

6.1.2 Feature Extraction and Normalization

After performing the feature engineering we need to understand how each feature will be extracted. In the CIC-CSE-IDS dataset, flow features were extracted using CICFlowMeter and logs were extracted from Windows, Linux and Mac OSs. In the Portuguese Army dataset, we extracted host features using a script to collect the events from all machines in a specific subnetwork. The flow features were extracted from a netflow tool used by the Portuguese Army to control and analyse the network traffic. Normalization is the process of finding a scale to fit all data without loosing critical information. To perform normalization we used a library from *sklearn.preprocessing* [42] named *MinMaxScaler*. This library performs normalization over each feature using the equations presented on 8, 9, where *min* and *max* represent range values. Normalization is performed by the *fit()* method.

$$X\_standard = (X - X.min)/(X.max - X.min) \tag{8}$$

$$X\_normalized = X\_standard \times (max - min) + min \tag{9}$$

MinMaxScaler library also provides a transformation method that is used as an alternative to zero mean. This method transforms data by applying equation 10, where $scale = (max - min)/(X.max - X.min)$.

$$X\_scaled = scale \times X + min - X.min \times scale \tag{10}$$

The *MinMaxScaler* object receives features as a numpy array of shape [n_entities, n_features]. By default returns features values within a range [0,1]. The range can be set by the user, nevertheless, we decided to use the default value. To perform normalization, we used a method from *MinMaxScaler* named *fit_transform()*, which combines the *fit()* and *transform()* methods returning an array with normalized features.

6.2 Clustering

One of the main goals of this work was to combine the results of clustering algorithms to improve accuracy and precision. Figure 7 shows the architecture implemented and identifies all phases of our work.

6.2.1 Applying Clustering Algorithms

As mentioned before, we intended to use four algorithms to perform clustering, KMeans, Agglomerative, DBSCAN, and HDBSCAN. Since in first results OPTICS were unable to identify outliers, we discarded this algorithm. The great advantage of HDBSCAN over DBSCAN is the test of several *epsilon* values. Once we decided to dynamically determine the best *epsilon* for DBSCAN, the HDBSCAN could no longer contribute significantly for this work. We ended using only the remaining three algorithms (KMeans, DBSCAN, and Agglomerative).

To apply these algorithms we divided each dataset into two subsets, external entities, and internal entities. This was performed as an attempt to reduce the differences between entities. Internal entities will have similar behavior with themselves than with external entities. By performing this division we joined internal entities resulting in a more uniform subset. This will reduce the probability of having entities misclassified.

[19] Officially registered as Windows Based Terminal.
[20] Allows hosts to perform name resolution for hosts on the same local link.
[21] Popular database for modern apps.

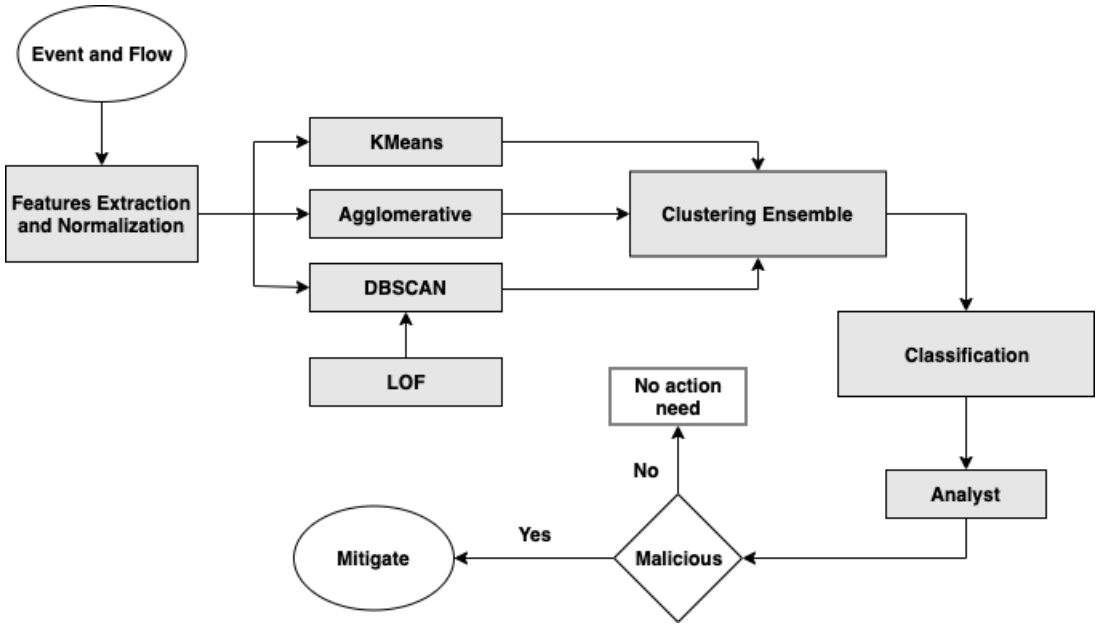


Figure 7: Architecture implemented.

6.2.2 Defining Classification

The entities' classification is the process of labeling an entity as benign or malicious. This is a crucial and maybe the hardest phase of intrusion detection. This phase consist in defining the criteria of classifying an entity as benign or malicious. For example if we define the criteria of being benign as the clusters with more than 5 entities, every cluster which do not fulfill the criteria is considered malicious. Considering this example, if we had only one victim[22] our accuracy might be lower than expected, nevertheless, if we had more than five victims the probability of missing some of them is higher. In the process of classification we defined inliers and outliers. An inlier is an entity considered with 'normal' behaviour and outlier an entity with 'abnormal' behaviour. The behavior of each entity is determined by the values of its features. Our criteria to identify an outlier was to consider all entities isolated in a cluster (cluster with a single entity) by KMeans and Agglomerative and the entities detected as outliers by DBSCAN (assigned to cluster -1).

KMeans		**DBSCAN**		**Agglomerative**	
#cluster	**Cluster size**	**#cluster**	**Cluster size**	**#cluster**	**Cluster size**
0	91	-1	11	0	111
1	155	0	396	1	49
2	16	-	-	2	146
3	1	-	-	3	3
4	42	-	-	4	21
5	97	-	-	5	1
6	1	-	-	6	74
7	3	-	-	7	1
8	1	-	-	8	1

Table 7: Example of the number of entities by cluster for each algorithm in CSE-CIC-IDS dataset for a 10 minutes time-window.

Table 7 present the number of entities in each cluster to the three clustering algorithms. This specific clustering output belongs to an internal subset and this particular day had a single victim. This means the

[22]Represent an entity attacked by other machine(s).

ideal anomaly detection system will only isolate one machine. Analysing Table 7 allows us to understand which clusters are classified as outliers. First, we need to look to cluster size columns and find the clusters with size one, these are considered outliers in KMeans and Agglomerative. Next, we look at the DBSCAN *#cluster* column and in the first row, we found the outlier cluster identified by this method. It classified 11 entities as outliers, nevertheless we knew that this specific day only had one victim. In this outlier cluster, we apply LOF to understand from the 11 entities which ones had the most probability of being an outlier.

6.2.3 Ensemble Method

We could have decided to use only one algorithm, nevertheless, we wanted to test outlier detection with different clustering classes (partition, hierarchical and density). All three algorithms performed good as outlier detectors but left a margin to improvements. As an attempt to improve our system we used a clustering ensemble. We decided to create our own version of clustering ensemble, using clusters intersection. This intersection increases the probability of an entity being an outlier, once it was classified as outlier by all algorithms. Our version consists of intersecting small clusters from algorithms not based in density with entities in outliers clusters from density-based algorithms (Figure 8). The method could be defined in four steps:

- Select all KMeans clusters with only one entity;
- Select all Agglomerative clusters with only one entity;
- Select DBSCAN outlier cluster (-1);
- Intersect all clusters and drop the entities outside the intersection;
- The entities remaining correspond to outliers.

Looking at Table 7, we ensemble the results by intersecting KMeans clusters number 3, 6 and 8 with cluster -1 from DBSCAN and clusters 5, 7 and 8 from Agglomerative. This ensemble method was chosen based on the good results obtained from our classification criteria using Agglomerative and KMeans. Nevertheless, our DBSCAN results were not so good and we wanted to improve them. After analyzing the entities in the DBSCAN outlier cluster and the one entity clusters in Agglomerative and KMeans, we found that all entities in small clusters were considered outliers in DBSCAN. By intersecting those clusters we were able to reduce the number of entities in DBSCAN outlier cluster. This ensemble method is also able to improve the accuracy of Agglomerative or KMeans. Considering an example where we have one single victim and KMeans found two clusters with a single entity. However Agglomerative found one single cluster. Applying this ensemble method will filter KMeans clusters and select the one containing the entity in common.

Observing Table 8 we can see that each algorithm considered had those three entities in common. Performing the intersection of these clusters leads to a reduction in DBSCAN outlier cluster size. Now if we look at LOF score in this cluster, we get a max score of -10.66 which corresponds to the victim (172.31.69.25). With this approach and to this particular attack (SSH-Brute Force) we were able to correctly detect our victim.

6.3 Evaluation Metrics

In both datasets, we knew entities involved (attacker(s) and victim(s)) and based on this information we were able to evaluate our system. To performed this evaluation we defined four basic metrics with the information we had about entities involved. Firstly we decide that clusters with the number of entities

KMeans		DBSCAN		Agglomerative	
Entity	**Cluster**	**Entity**	**Cluster**	**Entity**	**Cluster**
172.31.64.116	8	172.31.64.116	-1	172.31.64.116	5
172.31.0.2	6	172.31.64.126	-1	172.31.0.2	8
172.31.69.25	3	172.31.64.20	-1	172.31.69.25	7
-	-	172.31.64.69	-1	-	-
-	-	172.31.65.42	-1	-	-
-	-	172.31.65.90	-1	-	-
-	-	172.31.66.113	-1	-	-
-	-	172.31.67.72	-1	-	-
-	-	172.31.69.13	-1	-	-
-	-	172.31.69.25	-1	-	-
-	-	172.31.0.2	-1	-	-

Table 8: Cluster considered outliers and their entities.

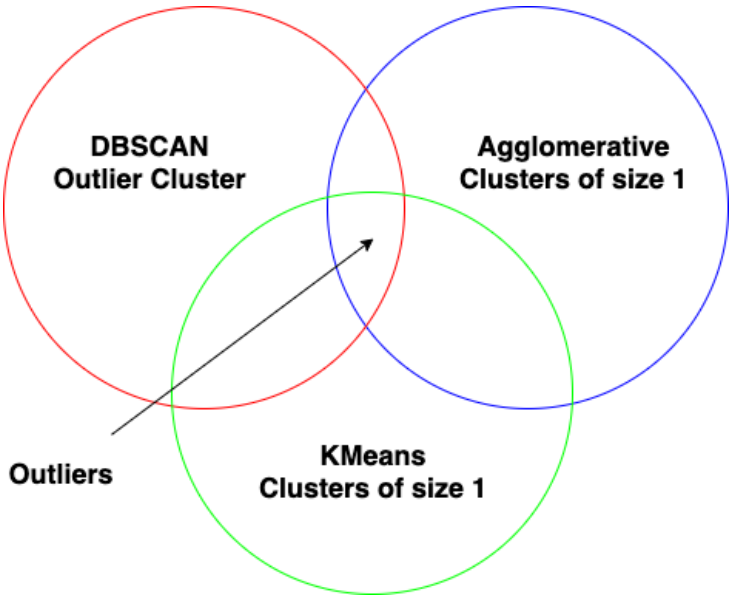


Figure 8: Ensemble method.

equal to 1, were considered possible outliers. To understand Table 9 it is important to define *accuracy* as the degree to which the result of clustering conforms to the correct value of an entity, *precision* the fraction of relevant results, *recall* the fraction of total relevant results correctly classified by the algorithm and *F1* the relation between *precision* and *recall*. Each of these metrics were applied individually to each algorithm in the first phase of our work.

6.4 Summary

This chapter presented in the first section the structure of our system, composed by Feature Engineering, Features Extraction and Normalization, Applying Clustering Algorithms and Defining Classification. Each of these sections describes decisions made during the construction of our system. The section introduces our version of clustering ensemble, which consists of the intersection of specific clusters from the different algorithms.

True Positives (TP)	Entities correctly classified as outliers
False Positives (FP)	Entities wrongly classified as outliers
True Negatives (TN)	Entities correctly classified as inliers
False Negatives (FN)	Entities wrongly classified as inliers
Accuracy (Acc)	$\frac{TP+TN}{TP+TN+FN+FP}$
Precision (Pre)	$\frac{TP}{TP+FP}$
Recall (Rec)	$\frac{TP}{TP+FN}$
F1	$\frac{2\times Precision\times Recall}{Recall+Precision}$

Table 9: Metrics used to analyse results.

7 Experimental Evaluation

This chapter presents the results obtained with the CSE-CIC-IDS 2018 dataset and with the combination of clustering algorithms. It also exhibitions the analysis of network and host features individually, as well as when combined. Our experiments were divided in two phases: the first one analyzed algorithm results individually and the second one used a clustering ensemble method to combine output from the different algorithms.

7.1 Combining Algorithms in Netflow Sources

This section presents the combination of the selected algorithms (KMeans, Agglomerative, DBSCAN, and LOF) applied to network sources. For properly understanding the next sections, it is important to remember our 3 approaches on netflow features:

1. 33 fixed port features plus 77 most used ports determined dynamically for each time-window (total of 100);
2. 33 fixed ports;
3. Ports 80, 194, 25 and 22 as features plus the port from the service attacked (based on [30]).

To all the approaches presented, we add the 10 fixed features (Table 5) which represent the general behavior of an entity based on their network data. The 1^{st} approach was composed of a total of 400 port features plus 10 fixed features, sum 410 features. The second approach was composed of a total of 132 port features plus 10 fixed features, makes a total of 142 features. The 3^{rd} approach was composed of 28 port features plus the 10 fixed features, sum a total of 38 features. These approaches allowed us to understand the effect of the number of features in anomaly detection using clustering algorithms.

The next results were obtained using the two subsets, internal entities, and external entities. The internal entities correspond to entities inside departments 1 to 5 (Figure 2), which belong to the subnetworks of 172.31.x.x. In these internal entities, we tried to identify the known victim(s). The external entities correspond to all other IPs outside the internal entity's range. From external entities, we expect to identify the known attacker(s).

7.1.1 FTP (14-02-18)

This attack was performed using a dictionary of 90 million words and the port attacked was port 21. It was expected that features from port 21 appeared in the relevant ones. Table 10 present the results obtained in this attack. This table contains 13 columns, the first one identify the ML algorithm used. The second column indicates the processed time-window where the respective attack was detected. The third column presents the duration of the processed time-window from the second column. The fourth column corresponds to the number of machines which participated in the attack (attacker(s) or victim(s)). The fifth column presents the LOF score of the entity known as an outlier (victim or attacker) compared with the LOF max score obtained in the DBSCAN outlier cluster. From the sixth column until the thirteenth column are presented the metrics (Table 9) obtained in the processed time-window. These metrics were defined based on our classification criteria. The number of rows depends on which approaches detected the attack. To better understand the number of rows we can divide the table in the three approaches. For each approach we have four rows, the first one indicates the approach and the entity detected. The second, third and fourth correspond to the results obtained respectively in KMeans, DBSCAN, and Agglomerative. Table 10 are presented only the 3^{rd} approach which detected the attack. To simplify the table visualization and since the other approaches did not detect any victim or attacker, we omitted their

results. In order to understand the results from the 3^{rd} approach, we need to analyze the features and their values.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Attacker/Victim not detected)												
3^{rd} Approach (Victim detected)												
KMeans	11:30	30m	1	-	1	1	432	0	0.99	0.5	1	0.67
DBSCAN	11:30	30m	1	-1/-27	1	28	405	0	0.9	0.03	1	0.06
Agglom.	11:30	30m	1	-	1	1	432	0	0.9	0.5	1	0.67

Table 10: Results of the best time-window for FTP attack from day 14/02/18.

Table 11 shows the most relevant features used in the third approach to detect the attack. These features were also used in 1^{st} and 2^{nd} approach. Since those two approaches (1^{st} and 2^{nd}) use more features than the 3^{rd} one, we can assume that (in this attack) increasing the number of features reduced the weight of each feature. This weight reduction increases the difficulty in detecting anomalies because the value of important features would have to be much more higher than the others. Another important fact is the reduced number of relevant features that were considered in the detection of this attack. The relevant features were selected by comparing the victim/attacker features values with the maximum, mean and mode values of each feature. The features with a higher value than mean and mode were considered as relevant.

Entity	Feature	Value	Max	Mean	Mode
3^{rd} Approach (Victim)					
172.31.69.25	21DstSent	59751	59751	140	0
	22SrcSent	15	15	0	0

Table 11: Relevant victim features extracted from time-window 11:30-12:00 in day 14/02/18.

Table 12 present the results obtained in the clustering ensemble. Our ensemble method was able to improve DBSCAN results.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Attacker/Victim not detected)											
3^{rd} Approach (Victim detected)											
KMeans	11:30	30m	1	1	1	432	0	0.99	0.5	1	0.67
DBSCAN	11:30	30m	1	1	1	432	0	0.99	0.5	1	0.67
Agglom.	11:30	30m	1	1	1	432	0	0.9	0.5	1	0.67

Table 12: Results of the best time-window for FTP attack from day 14/02/18 with ensemble.

7.1.2 SSH (14-02-18)

This attack was also executed with a 90 million dictionary in port 22. Based on our features we expect to find port 22 has a relevant feature. One again, the 1^{st} approach did not detect the attack (Table 13). With the 2^{nd} approach, it was possible to isolate both the victim and the attacker. The attacker was isolated in every 30m time-window. In the 3^{rd} approach, our system also detected the victim.

Next Table 14 shows the most relevant features in each approach. As expected, SSH port (22) was the only significant port feature found in both approaches. It is also important to notice that from the destination point of view, features values are higher. This suggests that victims of an SSH attack have

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Victim detected)												
KMeans	15:20	10m	1	-	1	0	408	0	1	1	1	1
DBSCAN	15:20	10m	1	0/0	1	2	406	0	0.99	0.33	1	0.49
Agglom.	15:20	10m	1	-	1	0	408	0	1	1	1	1
3^{rd} Approach (Victim detected)												
KMeans	15:20	10m	1	-	1	1	407	0	0.9	0.5	1	0.67
DBSCAN	15:20	10m	1	-3/-3	1	12	396	0	0.97	0.07	1	0.13
Agglom.	15:20	10m	1	-	1	1	407	0	0.9	0.5	1	0.67

Table 13: Results of the best time-window for SSH attack from day 14/02/18.

higher activity as a destination, which is simple to understand, once the attack consisted of experimenting 90 million words to access SSH.

Entity	Feature	Value	Max	Mean	Mode
$2^{nd}/3^{rd}$ Approach (Victim)					
172.31.69.25	22SrcSent	991	991	2.6	0
	22DstSent	231318	231318	581	0
	SrcTotLenRcvd	80928	51763809	990704	9881
	SrcTotLenSent	111930	672441	63343	3558
	DstPortContacted	10429	10429	71	25
	DstTotLenRcvd	19966233	19966233	92014	19542
	DstTotLenSent	27634357	27634357	117601	20811

Table 14: Victim features extracted from time-window 15:20-15:30 in day 14/02/18.

The clustering ensemble results are shown in Table 15. It reveals an improvement in DBSCAN in both approaches.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Victim detected)											
KMeans	15:20	10m	1	1	0	408	0	1	1	1	1
DBSCAN	15:20	10m	1	1	0	408	0	1	1	1	1
Agglom.	15:20	10m	1	1	0	408	0	1	1	1	1
3^{rd} Approach (Victim detected)											
KMeans	15:20	10m	1	1	1	407	0	0.9	0.5	1	0.67
DBSCAN	15:20	10m	1	1	1	407	0	0.9	0.5	1	0.67
Agglom.	15:20	10m	1	1	1	407	0	0.9	0.5	1	0.67

Table 15: Results of the best time-window for SSH attack from day 14/02/18 with clustering ensemble.

7.1.3 DoS - GoldenEye (15-02-18)

The GoldenEye tool is used to perform an HTTP DoS for security testing purposes. The DoS GoldenEye attack occurred from 9:26 to 10:09 and was executed by one external machine and directed to a single victim. Table 16 shows the best results in the best time-window. The victim was only detected by the 3^{rd} approach, which contains fewer features. Nevertheless, the 2^{nd} approach found the attacker. The most relevant features are presented in Table 17.

The attacker's relevant features correspond to flows started by the attacker and present much higher values than all other entities. This conclusion came from an analysis of mean and mode for these features. Looking at the victim's relevant features, we observe that ones considered represent the destination point

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Attacker detected)												
KMeans	9:50	10m	1	-	1	0	4060	0	1	1	1	1
DBSCAN	9:50	10m	1	-	1	4	4056	0	0.9	0.2	1	0.33
Agglom.	9:50	10m	1	-	1	0	4060	0	1	1	1	1
3^{rd} Approach (Victim detected)												
KMeans	9:20	10m	1	-	1	1	452	0	0.9	0.5	1	0.6
DBSCAN	9:20	10m	1	-44/-44	1	453	0	0	0	0.002	1	0.003
Agglom.	9:20	10m	1	-	1	1	452	0	0.9	0.5	1	0.6

Table 16: Results of the best time-window for DoS GoldenEye attack from day 15/02/18.

of view. It means that victim flows when contacted by other entities registered higher values than flows started by the victim. In other words, it indicates that the victim sent more packets on port 80 in response to a request. It is also important to notice the high number of different ports that contacted the victim. In general, relevant features present much higher values than other machines, according to the mean and mode of each feature. Despite best results in different time-windows, we see some logic on relevant features. Attacker's most relevant port feature corresponds to packets received in port 80 and victims correspond to packet sent from port 80. This indicates that the attack very most likely occurred at port 80.

Entity	Feature	Value	Max	Mean	Mode
2^{nd} Approach (Attacker)					
18.219.211.138	80SrcRcvd	48819	48819	12	0
	SrcPortUsed	13571	13571	9	1
	SrcTotLenRcvd	11701094	11701094	8262	0
	SrcTotLenSent	7294804	7294804	6443	0
3^{rd} Approach (Victim)					
172.31.69.25	80DstSent	16671	16671	38	0
	DstPortContacted	4500	4500	52	29
	DstTotLenRcvd	2560917	2560917	35907	0
	DstTotLenSent	3928593	3928593	47501	0

Table 17: Victim features extracted from time-windows 09:20-09:30 and 9:50-10:00 in day 15/02/18.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Attacker detected)											
KMeans	9:50	10m	1	1	0	4060	0	1	1	1	1
DBSCAN	9:50	10m	1	1	0	4060	0	1	1	1	1
Agglom.	9:50	10m	1	1	0	4060	0	1	1	1	1
3^{rd} Approach (Victim detected)											
KMeans	9:20	10m	1	1	1	452	0	0.9	0.5	1	0.6
DBSCAN	9:20	10m	1	1	1	452	0	0.9	0.5	1	0.6
Agglom.	9:20	10m	1	1	1	452	0	0.9	0.5	1	0.6

Table 18: Results of the best time-window for DoS GoldenEye attack from day 15/02/18 with ensemble.

Table 18 shows the results of the ensemble. It only improved DBSCAN results, nevertheless, in the third approach, the improvement was very significant.

7.1.4 DoS - Slowloris (15-02-18)

A Slowloris attack allows one machine to take down a service in another machine without using minimal bandwidth and causing minimal perturbations on unrelated ports and services. The present attack started at 10:59 and ended at 11:40. During this period, our system was able to detect the victim in two approaches, 2^{nd}, and 3^{rd}. The 2^{nd} approach obtained better results than the 3^{rd} one, as shown in Table 19.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Victim detected)												
KMeans	11:00	10m	1	-	1	1	434	0	0.9	0.5	1	0.6
DBSCAN	11:00	10m	1	-	1	4	431	0	0.9	0.2	1	0.3
Agglom.	11:00	10m	1	-	1	1	434	0	0.9	0.5	1	0.6
3^{rd} Approach (Victim detected)												
KMeans	11:20	10m	1	-	1	1	436	0	0.9	0.5	1	0.6
DBSCAN	11:20	10m	1	-219/-748	1	437	0	0	0.002	0.002	1	0.003
Agglom.	11:20	10m	1	-	1	1	436	0	0.9	0.5	1	0.6

Table 19: Results of the best time-window for DoS Slowloris attack from day 15/02/18.

To understand our results, we should look at Table 20, which contains the most relevant features in each approach. The 2^{nd} approach relevant features show high packets length sent and received considering mean values for all internal entities. Another important feature is the number of packets sent from port 80 when is responding to another machine request, using other words, when a victim appears as destination in the flows. This same feature was also relevant in the 3^{rd} approach, among the number of ports contacted and the total length of packets received from the destination point of view. These features reveal a high number of requests made to the victim's IP address.

Entity	Feature	Value	Max	Mean	Mode
2^{nd} Approach (Victim)					
172.31.69.25	22SrcSent	315	315	1	0
	80DstSent	5244	5244	12	0
	SrcTotLenSent	84932300	84932300	261989	2058
	DstPortContacted	2073	2073	47	29
	DstTotLenRcvd	2823118	2823118	40052	11519
3^{rd} Approach (Victim)					
172.31.69.25	80DstSent	5026	5026	12	0
	DstPortContacted	1981	1981	48	28
	DstTotLenRcvd	2929908	2929908	42493	17778

Table 20: Relevant features extracted from time-window 11:00-11:10 and 11:20-11:30 in day 15/02/18.

The clustering results obtained are presented in Table 21. In the third approach, we were able to improve DBSCAN significantly, by increasing the accuracy and precision from 0.002 to 0.9 and 0.5, respectively. In the second approach, we also improved DBSCAN.

7.1.5 DDoS - LOIC HTTP (20-02-18)

This attack was executed by 10 machines targeting a single victim. It consisted of flooding the server with TCP packets using HTTP requests to disrupt a victim service. The attack occurred from 10:12 to 11:17, and the victim was detected in 2^{nd} and 3^{rd} approaches. The results are shown in Table 22.

To better understanding this attack detection, we must look at their most relevant features. Since this attack used HTTP protocol it was expected to find a high usage of this port. The relevant features

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Victim detected)											
KMeans	11:00	10m	1	1	1	434	0	0.9	0.5	1	0.6
DBSCAN	11:00	10m	1	1	1	434	0	0.9	0.5	1	0.6
Agglom.	11:00	10m	1	1	1	434	0	0.9	0.5	1	0.6
3^{rd} Approach (Victim detected)											
KMeans	11:20	10m	1	1	1	436	0	0.9	0.5	1	0.6
DBSCAN	11:20	10m	1	1	1	436	0	0.9	0.5	1	0.6
Agglom.	11:20	10m	1	1	1	436	0	0.9	0.5	1	0.6

Table 21: Results of the best time-window for DoS Slowloris attack from day 15/02/18 with clustering ensemble.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Victim detected)												
KMeans	10:10	10m	1	-	1	1	453	0	0.99	0.5	1	0.66
DBSCAN	10:10	10m	1	-148/-148	1	8	446	0	0.98	0.11	1	0.2
Agglom.	10:10	10m	1	-	1	1	453	0	0.99	0.5	1	0.66
3^{rd} Approach (Victim detected)												
KMeans	10:40	10m	1	-	1	0	453	0	1	1	1	1
DBSCAN	10:40	10m	1	-75/-75	1	453	0	0	0.002	0.002	1	0.004
Agglom.	10:40	10m	1	-	1	1	452	0	0.99	0.5	1	0.66

Table 22: Results of the best time-window for DDoS LOIC HTTP attack from day 20/02/18.

are shown in Table 23. Analyzing this table, we confirm our expectations on HTTP port (80) which show for both approaches much higher values than other entities. This conclusion is retrieved by comparing the mean and mode values of these features with the victim's value to this feature. It is important to notice that for both approaches, most of the relevant features correspond to data sent from the victim either as the source or as the destination.

Entity	Feature	Value	Max	Mean	Mode
2^{nd} Approach (Victim)					
172.31.69.25	22SrcSent	46	58	0.13	0
	22DstSent	46	46	0.15	0
	80SrcSent	6	6	0.013	0
	80DstSent	110435	110435	245	0
	DstTotLenSent	26622731	26622731	155090	0
3^{rd} Approach (Victim)					
172.31.69.25	22SrcSent	6	6	0.02	0
	80SrcSent	24	24	0.05	0
	80DstSent	184444	184444	410	0
	DstTotLenRcvd	922220	922220	60389	56732
	DstTotLenSent	44451004	44451004	189786	65256

Table 23: Relevant features extracted from time-window 10:10-10:20 and 10:40-10:50 in day 20/02/18.

Table 24 are shown the results obtained from the clustering ensemble. From the second and third approaches, we were able to improve DBSCAN results. In the last approach, the improvement was higher than in the second one.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Victim detected)											
KMeans	10:10	10m	1	1	1	453	0	0.99	0.5	1	0.66
DBSCAN	10:10	10m	1	1	1	453	0	0.99	0.5	1	0.66
Agglom.	10:10	10m	1	1	1	453	0	0.99	0.5	1	0.66
3^{rd} Approach (Victim detected)											
KMeans	10:40	10m	1	1	0	453	0	1	1	1	1
DBSCAN	10:40	10m	1	1	0	453	0	1	1	1	1
Agglom.	10:40	10m	1	1	0	453	0	1	1	1	1

Table 24: Results of the best time-window for DDoS LOIC HTTP attack from day 20/02/18 with ensemble.

7.1.6 DDoS - LOIC UDP (20-02-18)

This attack had the same intention as the previous one, but this time using UDP packets. It occurred from 13:13 until 13:32, i.e. it lasted 19 minutes. During the attack period, our approaches were unable to isolate neither victim nor attackers. Nevertheless, this attack was performed on the following day and was detected by our system.

7.1.7 DDoS - LOIC UDP (21-02-18)

The present attack lasted 34 minutes, started at 10:09 and was finished at 10:43. This attack was undetectable from our system on a previous day. Nevertheless, maybe because it lasts longer, it was detected on this day. The results are presented in Table 25.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Victim detected)												
KMeans	10:10	10m	1	-	1	0	441	0	1	1	1	1
DBSCAN	10:10	10m	1	-	1	3	438	0	0.99	0.25	1	0.4
Agglom.	10:10	10m	1	-	1	0	441	0	1	1	1	1
3^{rd} Approach (Victim detected)												
KMeans	10:20	10m	1	-	1	1	445	0	0.99	0.5	1	0.66
DBSCAN	10:20	10m	1	-4793/-4793	1	446	0	0	0.002	0.002	1	0.004
Agglom.	10:20	10m	1	-	1	1	445	0	0.99	0.5	1	0.66

Table 25: Results of the best time-window for DDoS LOIC UDP attack from day 21/02/18.

Once the best time-window of 2^{nd} and 3^{rd} approaches are different, their values are not equal. From most relevant features we notice five features in 2^{nd} approach with higher values, namely *80SrcSent, 80DstSent, SrcPortContacted, SrcTotLenSent* and *DstPortContacted*. These features prove an exchange of heavy packets using various ports. Once the attack was performed by 10 different machines, this behavior is somehow expected. From the 3^{rd} approach, it is important to notice that at the destination point of view the victim received heavy packets, which excel from the remaining features and other entities.

Table 27 present the results obtained with the clustering ensemble. It was helpful in both approaches, by increasing DBSCAN results. In the third approach, the improvement was very significant, increasing accuracy of 0.002 to 0.99.

Entity	Feature	Value	Max	Mean	Mode
2^{nd} Approach (Victim)					
172.31.69.28	21DstSent	1	1	0.002	0
	22SrcSent	48	48	0.2	0
	80SrcSent	1185022	1185022	2775	0
	80DstSent	988	988	3	0
	8080DstSent	1	1	0.002	0
	50010SrcRcvd	30	30	0.07	0
	50010DstRcvd	30	30	0.07	0
	SrcPortContacted	16389	16389	56	9
	SrcTotLenRcvd	71124667	71124667	1266726	8484
	SrcTotLenSent	221628281	221628281	573621	2719
	DstPortContacted	16396	16396	138	60
3^{rd} Approach (Victim)					
172.31.69.28	22SrcSent	10	10	0.12	0
	22DstSent	69	69	0.33	0
	DstTotLenRcvd	1890139471	1890139471	4397480	0

Table 26: Relevant features extracted from time-window 10:10-10:20 and 10:20-10:30 in day 21/02/18.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Victim detected)											
KMeans	10:10	10m	1	1	0	441	0	1	1	1	1
DBSCAN	10:10	10m	1	1	0	441	0	1	1	1	1
Agglom.	10:10	10m	1	1	0	441	0	1	1	1	1
3^{rd} Approach (Victim detected)											
KMeans	10:20	10m	1	1	1	445	0	0.99	0.5	1	0.66
DBSCAN	10:20	10m	1	1	1	445	0	0.99	0.5	1	0.66
Agglom.	10:20	10m	1	1	1	445	0	0.99	0.5	1	0.66

Table 27: Results of the best time-window for DDoS LOIC UDP attack from day 21/02/18 with clustering ensemble.

7.1.8 DDoS - HOIC (21-02-18)

The HOIC attack occurred at 14:05 and lasted for an hour, ending at 15:05. The results obtained for this attack are presented in Table 28. Once again the 1^{st} approach was unable to find neither victim nor attackers. Comparing to the other approaches, we can assume that the high number of features attenuates relevant features in attack detection. Focusing on results obtained in the 2^{nd} and 3^{rd} approach we found that using more features resulted in higher accuracy, precision, recall, and F1. To understand these results, we must analyze the most relevant features. These features can be found in Table 29.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Victim detected)												
KMeans	14:50	10m	1	-	1	0	434	0	1	1	1	1
DBSCAN	14:50	10m	1	-	1	1	433	0	0.99	0.5	1	0.66
Agglom.	14:50	10m	1	-	1	0	434	0	1	1	1	1
3^{rd} Approach (Victim detected)												
KMeans	14:50	10m	1	-	1	1	433	0	0.99	0.5	1	0.66
DBSCAN	14:50	10m	1	-7/-43	1	434	0	0	0.002	0.002	1	0.004
Agglom.	14:50	10m	1	-	1	1	433	0	0.99	0.5	1	0.66

Table 28: Results of the best time-window for DDoS HOIC attack from day 21/02/18.

Analyzing the most relevant features, we found their values above the mean and mode. It means the victim had less activity than other machines. Comparing features from different approaches we found only two differences, 3389DstSent and 445SrcSent. These differences emerged from two ports that are not considered in the 3^{rd} approach. According to the results shown before we can assume that this difference contributed to improving the results on the 2^{nd} approach.

Entity	Feature	Value	Max	Mean	Mode
2^{nd} Approach (Victim)					
172.31.69.28	80SrcRcvd	0	4139	174	68
	3389DstSent	1	2164	331	285
	445SrcSent	0	100	16	10
	SrcPortUsed	4	950	144	19
	SrcTotLenRcvd	5500	36564012	789440	7698
	DstIPContacted	21	144	45	30
	DstPortContacted	30	355	65	48
	DstTotLenRcvd	33551	391198	56966	48991
3^{rd} Approach (Victim)					
172.31.69.28	80SrcRcvd	0	4139	174	68
	SrcPortUsed	4	950	144	19
	SrcTotLenRcvd	5500	36564012	789440	7698
	DstIPContacted	21	144	45	30
	DstPortContacted	30	355	65	48
	DstTotLenRcvd	33551	391198	56966	48991

Table 29: Relevant features extracted from time-window 14:50-15:00 in day 21/02/18.

Table 30 presents the results after applying the clustering ensemble. It is possible to see that in each approach the only improvement was in DBSCAN. In the third approach, it was a significant improvement.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Victim detected)											
KMeans	14:50	10m	1	1	0	434	0	1	1	1	1
DBSCAN	14:50	10m	1	1	0	434	0	1	1	1	1
Agglom.	14:50	10m	1	1	0	434	0	1	1	1	1
3^{rd} Approach (Victim detected)											
KMeans	14:50	10m	1	1	1	433	0	0.99	0.5	1	0.66
DBSCAN	14:50	10m	1	1	1	433	0	0.99	0.5	1	0.66
Agglom.	14:50	10m	1	1	1	433	0	0.99	0.5	1	0.66

Table 30: Results of the best time-window for DDoS HOIC attack from day 21/02/18 with clustering ensemble.

7.1.9 Brute Force Web (22-02-18)

This attack was performed between 10:17 and 11:24. Just like every brute force it is expected to find a high number of requests. Once it was a brute force web attack the high request volume should be noticed in port 80. This particular attack was not detected by the two first approaches. Nevertheless, the 3^{rd} approach presented good results (Table 31) with KMeans and Agglomerative.

To understand the 3^{rd} approach results it is useful to analyze the most relevant features, as shown in Table 32. In this case, were expected to find several requests on port 80. Nevertheless, we observe that the victim did not receive any packet on port 80, while the mean and mode values of this feature register high values. Despite this fact, the victim port 80 was used in response to requests. It means that from

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Attacker/Victim not detected)												
3^{rd} Approach (Victim detected)												
KMeans	10:20	10m	1	-	1	0	455	0	1	1	1	1
DBSCAN	10:20	10m	1	-6/-29	1	455	0	0	0.002	0.002	1	0.004
Agglom.	10:20	10m	1	-	1	0	455	0	1	1	1	1

Table 31: Results of the best time-window for brute force web attack from day 22/02/18.

destination point of view packets were sent using port 80. Another interesting feature is the high number of packets sent and received from the destination point of view.

Entity	Feature	Value	Max	Mean	Mode
3^{rd} Approach (Victim)					
172.31.69.28	80SrcRcvd	0	26363	675	108
	80SrcSent	5	5	0	0
	80DstSent	1183	1183	3	0
	22SrcSent	45	65	0	0
	DstTotLenRcvd	639551	880115	84999	40599
	DstTotLenSent	806859	9992435	127481	65233

Table 32: Relevant features extracted from time-window 10:20-10:30 in day 22/02/18.

Table 33 shows the results obtained with our method of clustering ensemble. It is possible to observe a siginificant improve in DBSCAN results, by increasing accuracy and precision from 0.002 to 1.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Attacker/Victim not detected)											
3^{rd} Approach (Victim detected)											
KMeans	10:20	10m	1	1	0	455	0	1	1	1	1
DBSCAN	10:20	10m	1	1	0	455	0	1	1	1	1
Agglom.	10:20	10m	1	1	0	455	0	1	1	1	1

Table 33: Results of the best time-window for brute force web attack from day 22/02/18 with ensemble.

7.1.10 Brute Force XSS (22-02-18)

This attack started at 13:50 and ended at 14:29. Our system detected both 2^{nd} and 3^{rd} approaches with a precision range of 0.002 to 0.5. In the second approach, the system detected the attacker with better mean accuracy than in the third approach (Table 34), which detected the victim. The most relevant features of each approach are shown in Table 35.

Analyzing the most relevant features, it is possible to see from the second approach the high values of packets received in port 80 from the source point of view. The attacker also presents relatively high values for the size of packets received and sent in communications where he is the source. On the contrary, looking at victim relevant features from the 3^{rd} approach we found none packets received in port 80 from the source point of view. Nevertheless, from the destination point of view, it used port 80 to sent many more packets than all other machines. In general, the victim as destination length of packets sent and received was relatively higher than mean.

Despite their features belong to different approaches and different time-windows, the attack is the same. From this fact is important to highlight that higher features from attacker and victim complement

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Attacker detected)												
KMeans	14:20	10m	1	-	1	2	4593	0	0.99	0.33	1	0.5
DBSCAN	14:20	10m	1	-	1	4	4591	0	0.99	0.2	1	0.33
Agglom.	14:20	10m	1	-	1	2	4593	0	0.99	0.33	1	0.5
3^{rd} Approach (Victim detected)												
KMeans	13:50	10m	1	-	1	1	445	0	0.99	0.5	1	0.66
DBSCAN	13:50	10m	1	-6/-29	1	446	0	0	0.002	0.002	1	0.004
Agglom.	13:50	10m	1	-	1	1	445	0	0.99	0.5	1	0.66

Table 34: Results of the best time-window for brute force XSS attack from day 22/02/18.

each other. It means that relevant features from the attacker correspond to packets received and victims correspond to packets sent. This indicates requests made from the attacker were answered by the victim.

Entity	Feature	Value	Max	Mean	Mode
2^{nd} Approach (Attacker)					
18.218.115.60	80SrcRcvd	934	934	0.2	0
	80DstRcvd	1	1	0	0
	SrcPortUsed	10	1274	7	1
	SrcTotLenRcvd	1707662	2103697	6240	0
	SrcTotLenSent	505302	1527613	6118	0
3^{rd} Approach (Victim)					
172.31.69.28	80SrcRcvd	0	44160	481	68
	80SrcSent	1	1	0	0
	80DstSent	882	882	2	0
	DstTotLenRcvd	484902	601541	68592	ND
	DstTotLenSent	1582162	8191377	88368	ND

Table 35: Relevant features extracted from time-window 14:20-14:30 and 13:50-14:00 in day 22/02/18.

Table 36 is presented the results after the clustering ensemble. We observe an improvement in DBSCAN by matching KMeans and Agglomerative results.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Attacker detected)											
KMeans	14:20	10m	1	1	2	4593	0	0.99	0.33	1	0.5
DBSCAN	14:20	10m	1	1	2	4593	0	0.99	0.33	1	0.5
Agglom.	14:20	10m	1	1	2	4593	0	0.99	0.33	1	0.5
3^{rd} Approach (Victim detected)											
KMeans	13:50	10m	1	1	1	445	0	0.99	0.5	1	0.66
DBSCAN	13:50	10m	1	1	1	445	0	0.99	0.5	1	0.66
Agglom.	13:50	10m	1	1	1	445	0	0.99	0.5	1	0.66

Table 36: Results of the best time-window for brute force XSS attack from day 22/02/18 with ensemble.

7.1.11 SQL Injection (22-02-18)

The SQL injection began at 16:15 and was performed until 16:29. During this period several queries were performed. The results are shown in Table 37. In this attack only the third approach classified victim as an outlier, the remains approaches were unable to detect him as an outlier. To understand the results, we present the most relevant features in Table 38.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Attacker/Victim not detected)												
3^{rd} Approach (Victim detected)												
KMeans	16:20	10m	1	-	1	1	444	0	0.99	0.5	1	0.66
DBSCAN	16:20	10m	1	-6/-81	1	445	0	0	0.002	0.002	1	0.004
Agglom.	16:20	10m	1	-	1	2	443	0	0.99	0.33	1	0.49

Table 37: Results of the best time-window for SQL injection attack from day 22/02/18.

We only found 3 relevant features from all features in third approach and their values are not so high as expected. Attack duration can be the reason for that. These low values are also the reason why other approaches detected neither victim nor attacker.

Entity	Feature	Value	Max	Mean	Mode
3^{rd} Approach (Victim)					
172.31.69.28	80DstSent	20	20	0	0
	80SrcSent	4	4	0	0
	22SrcSent	22	22	0	0

Table 38: Relevant features extracted from time-window 16:20-16:30 in day 22/02/18.

Table 39 shows the results after applying our clustering ensemble method. In this particular case, we were able to improve both Agglomerative and DBSCAN results by matching KMeans. The DBSCAN results improved in accuracy and precision of 0.002 to 0.99 and 0.5, respectively.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Attacker/Victim not detected)											
3^{rd} Approach (Victim detected)											
KMeans	16:20	10m	1	1	1	444	0	0.99	0.5	1	0.66
DBSCAN	16:20	10m	1	1	1	444	0	0.99	0.5	1	0.66
Agglom.	16:20	10m	1	1	1	444	0	0.99	0.5	1	0.66

Table 39: Results of the best time-window for SQL injection attack from day 22/02/18 with ensemble.

7.1.12 Brute Force Web (23-02-18)

Table 40 shows the results of the brute force web attack. This attack occurred during 1 hour and started at 10:03. The results were not so good, only using the 3^{rd} approach we could detect the victim. This approach also detected the attack in other time-windows.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Attacker/Victim not detected)												
3^{rd} Approach (Victim detected)												
KMeans	10:10	10m	1	-	1	1	442	0	0.9	0.5	1	0.6
DBSCAN	10:10	10m	1	-0/-4	1	27	416	0	0.9	0.04	1	0.08
Agglom.	10:10	10m	1	-	1	1	442	0	0.9	0.5	1	0.6

Table 40: Results of the best time-window for brute force web attack from day 23/02/18.

In this attack, the 3^{rd} approach used 38 features, which stand out the ones in Table 41. It is not surprising to find port 80 as a relevant feature, once this port is used in web communications. The fact

of being such a low number of relevant features might be the reason for not being detected in other approaches.

Entity	Feature	Value	Max	Mean	Mode
3^{rd} Approach (Victim)					
172.31.69.28	80SrcSent	7	7	0	0
	80DstSent	1181	1181	2.7	0

Table 41: Victim features extracted from time-window 10:00-10:10 in day 23/02/18.

Table 42 shows the results obtained from the clustering ensemble. From our ensemble method, we were able to improve DBSCAN results.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Attacker/Victim not detected)											
3^{rd} Approach (Victim detected)											
KMeans	10:10	10m	1	1	1	442	0	0.9	0.5	1	0.6
DBSCAN	10:10	10m	1	1	1	442	0	0.9	0.5	1	0.6
Agglom.	10:10	10m	1	1	1	442	0	0.9	0.5	1	0.6

Table 42: Results of the best time-window for brute force web attack from day 23/02/18 with ensemble.

7.1.13 Brute Force XSS (23-02-18)

The Cross-Site Scripting attack started at 13:00 and ended at 14:10. In this period our system was able to detect the victim using the 2^{nd} and 3^{rd} approaches. As shown in Table 43, the 2^{nd} approach achieved better results with less false positives. Nevertheless, it is important to compare relevant features from both approaches.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Victim)												
KMeans	13:00	10m	1	-	1	0	427	0	1	1	1	1
DBSCAN	13:00	10m	1	-4/-11	1	11	416	0	0.9	0.08	1	1
Agglom.	13:00	10m	1	-	1	1	426	0	0.9	0.5	1	0.6
3^{rd} Approach (Victim)												
KMeans	13:00	10m	1	-	1	2	425	0	0.9	0.3	1	0.5
DBSCAN	13:00	10m	1	-7/-7	1	427	0	0	0.002	0.002	1	0.004
Agglom.	13:00	10m	1	-	1	1	426	0	0.9	0.5	1	0.6

Table 43: Results of the best time-window for brute force XSS attack from day 23/02/18.

Looking at Table 44 we observe that all relevant features count packets from the sent point of view either for the source and destination side. These features are considered relevant because their value represents the maximum. Nevertheless, for a brute force attack, the values are lower than expected.

Table 45 presents the results obtained from our method of clustering ensemble. In this case, it was possible to improve Agglomerative and DBSCAN in the second approach. In the third approach, we were able to improve significantly KMeans and DBSCAN.

Entity	Feature	Value	Max	Mean	Mode
$2^{nd}/3^{rd}$ Approach (Victim)					
172.31.69.28	22SrcSent	4	4	0	0
	22DstSent	13	13	0	0
	80DstSent	1049	1049	3	0
	8080DstSent	1	1	0	0

Table 44: Victim features extracted from time-window 13:00-13:10 in day 23/02/18.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Victim)											
KMeans	13:00	10m	1	1	0	427	0	1	1	1	1
DBSCAN	13:00	10m	1	1	0	427	0	1	1	1	1
Agglom.	13:00	10m	1	1	0	427	0	1	1	1	1
3^{rd} Approach (Victim)											
KMeans	13:00	10m	1	1	1	426	0	0.9	0.5	1	0.6
DBSCAN	13:00	10m	1	1	1	426	0	0.9	0.5	1	0.6
Agglom.	13:00	10m	1	1	1	426	0	0.9	0.5	1	0.6

Table 45: Results of the best time-window for brute force XSS attack from day 23/02/18 with clustering ensemble.

7.1.14 SQL Injection (23-02-18)

The SQL injection was performed against a vulnerable website using the Damn Vulnerable Web App (DVWA)[23]. Table 46 shows the clustering results of each algorithm. The first two approaches were unable to detect this attack, neither victims nor attackers. In the 3^{rd} approach, with fewer features, the attacker was isolated by KMeans and Agglomerative. DBSCAN also detected the attacker, nevertheless their LOF score discourages the classification has an outlier.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Attacker/Victim not detected)												
3^{rd} Approach (Attacker)												
KMeans	15:00	10m	1	-	1	0	4369	0	1	1	1	1
DBSCAN	15:00	10m	1	-3/-121	1	24	4345	0	0.9	0.04	1	0.08
Agglom.	15:00	10m	1	-	1	0	4369	0	1	1	1	1

Table 46: Results of the best time-window for SQL injection attack from day 23/02/18.

To understand the results, we must look at attacker features and their values. Table 47 shows that information. From all features of the 3^{rd} approach, only 8 obtained values, the remaining had zero value. In this case, the attacker was classified has an outlier because of features bellow the mean, never achieving the highest result in anyone. Once it has 4370 external entities, we do not have any information about these machines' origins. They can be servers, bots, PCs from an external network, among others. This elevates the probability of having different behaviors, depending on the machine. Having servers included in this analysis increases the mean of some features, such as *SrcIPContacted, SrcPortUsed, SrcTotLenSent, SrcPortContacted, DstIPContacted, DstPortContacted* and *DstPortUsed*. This fact justifies the results in attacker features.

The clustering ensemble results are shown in Table 48. These results show an improvement in DBSCAN by matching KMeans and Agglomerative.

[23]Application used to aid security professional to test their skills.

Entity	Feature	Value	Max	Mean	Mode
3^{rd} Approach (Attacker)					
18.218.115.60	80SrcRcvd	57	5489	13	0
	SrcIPContacted	1	392	7	1
	SrcPortUsed	13	1491	21	1
	SrcPortContacted	1	497	6	1
	SrcTotLenSent	9521	1440208	10327	0
	DstIPContacted	1	402	7	1
	DstPortUsed	3	916	6	1
	DstPortContacted	1	15517	14	1

Table 47: Parcial features from attacker extracted on time-window 15:00-15:10 in day 23/02/18.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Attacker/Victim not detected)											
3^{rd} Approach (Attacker)											
KMeans	15:00	10m	1	1	0	4369	0	1	1	1	1
DBSCAN	15:00	10m	1	1	0	4369	0	1	1	1	1
Agglom.	15:00	10m	1	1	0	4369	0	1	1	1	1

Table 48: Results of the best time-window for SQL injection attack from day 23/02/18 with ensemble.

7.1.15 Infiltration (28-02-18)

This attack consisted of sending an email with a malicious file to gain access to a victim's machine. Using the 1^{st} approach with 100 ports as features, we were unable to detect neither the victim nor the attacker. Nevertheless using fewer features, 2^{nd} and 3^{rd} approach, we could find the victim with good confidence. The best results were obtained in the 2^{nd} approach (Table 49). To understand the results, we should look at Table 50.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Victim)												
KMeans	11:30	30m	1	-	1	0	279	0	1	1	1	1
DBSCAN	11:30	30m	1	-	1	1	278	0	0.9	0.5	1	0.66
Agglom.	11:30	30m	1	-	1	0	279	0	1	1	1	1
3^{rd} Approach (Victim)												
KMeans	11:50	10m	1	-	1	1	277	0	0.9	0.5	1	0.66
DBSCAN	11:50	10m	1	-1/-14	1	15	263	0	0.9	0.063	1	0.12
Agglom.	11:50	10m	1	-	1	1	277	0	0.9	0.5	1	0.66

Table 49: Results of the best time-window for infiltration attack from day 28/02/18.

Table 50 shows the relevant features with high values. It is important to observe that in both approaches packets received in port 22 and the total length of packets sent from the source point of view was considered as relevant features. This suggests that once the attacker gains access to the victim's computer, it established an SSH connection and sent information to the attacker.

Table 51 shows the results obtained from the clustering ensemble. As a result of the clustering ensemble, it is possible to see an increase in DBSCAN results in both approaches. It results from the intersection of clusters classified as outliers.

Entity	Feature	Value	Max	Mean	Mode
2^{nd} Approach (Victim)					
172.31.69.24	22SrcRcvd	48	48	0	0
	135SrcSent	61	61	0	0
	137SrcSent	2	2	0	0
	139SrcSent	20	20	0	0
	445SrcSent	104	104	0	0
	3389SrcRcvd	74	74	0	0
	3389DstRcvd	10	10	0	0
	SrcTotLenSent	1250144	1250144	241916	ND
3^{rd} Approach (Victim)					
172.31.69.24	22SrcRcvd	48	48	0	0
	SrcTotLenSent	94123	941123	79524	33

Table 50: Relevant features from victim extracted on time-window 11:30-12:00 and 11:50-12:00 in day 28/02/18.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Victim)											
KMeans	11:30	30m	1	1	0	279	0	1	1	1	1
DBSCAN	11:30	30m	1	1	0	279	0	1	1	1	1
Agglom.	11:30	30m	1	1	0	279	0	1	1	1	1
3^{rd} Approach (Victim)											
KMeans	11:50	10m	1	1	1	277	0	0.9	0.5	1	0.66
DBSCAN	11:50	10m	1	1	1	277	0	0.9	0.5	1	0.66
Agglom.	11:50	10m	1	1	1	277	0	0.9	0.5	1	0.66

Table 51: Results of the best time-window for infiltration attack from day 28/02/18 with clustering ensemble.

7.1.16 Infiltration (01-03-18)

This attack was similar to the previous one, and results were also good on the 2^{nd} and 3^{rd} approaches. Once again, the 1^{st} approach detected neither attacker nor victim. Table 52 shows that the 2^{nd} approach has the best approach. Nevertheless, it is crucial to understand the relevant features that lead to these results.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Victim)												
KMeans	14:30	10m	1	-	1	0	418	0	1	1	1	1
DBSCAN	14:30	10m	1	-	1	5	413	0	0.99	0.17	1	0.29
Agglom.	14:30	10m	1	-	1	0	418	0	1	1	1	1
3^{rd} Approach (Victim)												
KMeans	14:40	10m	1	-	1	1	424	0	0.99	0.5	1	0.66
DBSCAN	14:40	10m	1	-0/-13	1	15	410	0	0.96	0.06	1	0.12
Agglom.	14:40	10m	1	-	1	1	424	0	0.99	0.5	1	0.66

Table 52: Results of the best time-window for infiltration attack from day 01/03/18.

Observing Table 53, we conclude that the 3^{rd} approach relevant features were also considered relevant in the 2^{nd} approach. In the 2^{nd} approach, we observe more relevant features and we found port 3389 which is used for Windows remote desktop. This is a serious indicator that the attacker was controlling the victim's machine. This particular attack is also curious to notice that almost all port features correspond to packets received.

Entity	Feature	Value	Max	Mean	Mode
2^{nd} Approach (Victim)					
172.31.69.13	21SrcRcvd	4	4	0	0
	22SrcRcvd	386	386	1	0
	22DstRcvd	16	16	0	0
	23SrcRcvd	4	4	0	0
	25SrcRcvd	4	4	0	0
	80DstRcvd	162	162	10	0
	135SrcRcvd	698	698	2	0
	137SrcRcvd	30	30	0	0
	139SrcRcvd	130	130	0	0
	445SrcRcvd	436	436	1	0
	445DstRcvd	240	240	1	0
	500SrcRcvd	4	4	0	0
	3389SrcSent	4914	4914	54	0
	3389SrcRcvd	315	315	1	0
	3389DstSent	3649	3649	609	0
	3389DstRcvd	150	150	0	0
	SrcPortContacted	1054	1054	36	10
	SrcTotLenSent	2973473	2973473	59215	1292
3^{rd} Approach (Victim)					
172.31.69.13	21SrcRcvd	6	6	0	0
	22DstRcvd	24	24	0	0
	22SrcRcvd	652	652	2	0
	25SrcSent	6	6	0	0
	SrcPortContacted	1035	1035	38	10

Table 53: Relevant features from victim extracted on time-window 14:30-14:40 and 14:40-14:50 in day 01/03/18.

The clustering ensemble results are shown in Table 54. It improved the results of DBSCAN by matching KMeans and Agglomerative.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Victim)											
KMeans	14:30	10m	1	1	0	418	0	1	1	1	1
DBSCAN	14:30	10m	1	1	0	418	0	1	1	1	1
Agglom.	14:30	10m	1	1	0	418	0	1	1	1	1
3^{rd} Approach (Victim)											
KMeans	14:40	10m	1	1	1	424	0	0.99	0.5	1	0.66
DBSCAN	14:40	10m	1	1	1	424	0	0.99	0.5	1	0.66
Agglom.	14:40	10m	1	1	1	424	0	0.99	0.5	1	0.66

Table 54: Results of the best time-window for infiltration attack from day 01/03/18 with clustering ensemble.

7.1.17 Bot (02-03-18)

This day an attacker executed an attack on 10 different machines in the network. According to our outlier classification definitions, an outlier is an entity isolated (KMeans and Agglomerative) and inserted in the outliers cluster from DBSCAN. Given that fact, in this case, we were not expecting to detect any victim. Nevertheless, we had only one attacker, and our system detected it. The attacker results are shown in Table 55.

To understand the results, we should check the relevant features. Table 56 presents the most relevant

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)												
2^{nd} Approach (Attacker)												
KMeans	15:40	10m	1	-	1	1	4297	0	0.99	0.5	1	0.66
DBSCAN	15:40	10m	1	-1/-6	1	6	4292	0	0.9	0.14	1	0.25
Agglom.	15:40	10m	1	-	1	1	4297	0	0.99	0.5	1	0.66
3^{rd} Approach (Attacker)												
KMeans	11:20	10m	1	-	1	0	4931	0	1	1	1	1
DBSCAN	11:20	10m	1	-1/-110	1	14	4917	0	0.9	0.6	1	0.75
Agglom.	11:20	10m	1	-	1	0	4931	0	1	1	1	1

Table 55: Results of the best time-window of bot attack from day 02/03/18.

features. It corresponds to two different periods, 2^{nd} approach best result occurred at 15:40 and 3^{rd} approach at 11:20. It is important to notice that despite different time-windows, both have relevant features in common namely, *DstPortContacted* and *DstPortLenRcvd*. These features represent the flows where the IP address 18.219.211.138 was the destination. It indicates that the number of ports contacted, and the total length of packets received as a destination were crucial to identify this attacker. Another conclusion from Table 56 is the port features considered relevant in the 2^{nd} approach are not contained in 3^{rd} approach features.

Entity	Feature	Value	Max	Mean	Mode
2^{nd} Approach (Attacker)					
18.219.211.138	8080SrcSent	10038	10038	2	0
	8080DstSent	38070	38070	9	0
	50010DstRcvd	15	15	0	0
	SrcPortContacted	857	857	3	0
	DstPortContacted	4292	4292	6	1
	DstPortLenRcvd	2962543	2962543	4900	0
3^{rd} Approach (Attacker)					
18.219.211.138	DstPortContacted	3007	3007	6	1
	DstTotLenRcvd	3976383	3976383	6375	0

Table 56: Relevant features from attacker extracted on time-windows 15:40-15:50 and 11:20-11:30 in day 02/03/18.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
1^{st} Approach (Attacker/Victim not detected)											
2^{nd} Approach (Attacker)											
KMeans	15:40	10m	1	1	1	4297	0	0.99	0.5	1	0.66
DBSCAN	15:40	10m	1	1	1	4297	0	0.99	0.5	1	0.66
Agglom.	15:40	10m	1	1	1	4297	0	0.99	0.5	1	0.66
3^{rd} Approach (Attacker)											
KMeans	11:20	10m	1	1	0	4931	0	1	1	1	1
DBSCAN	11:20	10m	1	1	0	4931	0	1	1	1	1
Agglom.	11:20	10m	1	1	0	4931	0	1	1	1	1

Table 57: Results of the best time-window of bot attack from day 02/03/18 with ensemble.

Table 57 presents the results obtained after the clustering ensemble. It improved DBSCAN results, however, KMeans and Agglomerative do not.

7.1.18 Summary of Results

This section presented the results of the 3 approaches used on netflow data. It corresponds to 17 attacks simulated in an artificial dataset, described in Chapter 4. From those approaches, we conclude that using fewer features improves the results since the third approach was the one that detected more attacks. The 1^{st} approach did not detect any of the attacks and the second approach detected 11 attacks. From all the 17 attacks only the LOIC UDP from day 20/02/18 was not detected. The second approach achieved a mean accuracy of 0.977, a mean precision of 0.60, a mean recall of 1 and a mean F1 of 0.71. The third approach achieved a mean accuracy of 0.77, mean precision of 0.41, mean recall of 1 and mean F1 of 0.503. With our method of clustering ensemble the new metric values were: for second approach we achieved a mean accuracy of 0.996, mean precision of 0.78, mean recall of 1 and mean F1 of 0.848, in the third approach we achieved a mean accuracy of 0.993, mean precision of 0.65, mean recall of 1 and mean F1 if 0.76. We observe an improvement in the metrics, mostly due to DBSCAN results. These metrics proved that the second approach performed better in detecting victims/attackers, however, this approach missed five attacks. The best time-window in the second and third approaches was the 10 minutes window. In Annex C we present a summary of the attackers/victims detected and the respective time-windows, where *N*, *A*, and *V* correspond, respectively to *None*, *Attacker* and *Victim*.

7.2 Combining Algorithms in Windows Event Sources

This section shows the results obtained with our solution using data from Windows Event sources. Our approach was to count events occurrence, their level and the total number of events registered. To CSE-CIC-IDS dataset we could not use features described on Annex A, once this dataset only contains events from the system. Applying the features from Annex A will result in information from only one or two events, which did not perform well in outlier detection. Since event logs were extracted from the hosts, we do not have any information about the attackers. For this reason, all results presented in this chapter correspond to internal entities using Windows OS.

7.2.1 Infiltration (28-02-18)

This attack was performed by a single machine in two periods (Table 2). It was executed by sending an email with a malicious file to gain access to the victim's computer. The next table presents the best time-window results obtained for this attack, which corresponds to victim results. We could only isolate the victim in the attack that occurred from 10:50 to 12:05. In this case, the outlier cluster had less than 6 entities and for this reason, LOF was not applied. This time-window extracted 40 features, where 35 of them came from event IDs.

Algo	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
KMeans	10:30	30m	1	-	1	1	150	0	0.99	0.5	1	0.7
DBSCAN	10:30	30m	1	0/0	1	4	147	0	0.97	0.2	1	0.33
Agglom	10:30	30m	1	-	1	1	150	0	0.99	0.5	1	0.7

Table 58: Results of the best time-window from day 28/02/18.

The most relevant features were: event ID 414, 6, 12, 98, 18, 20, 25, 27, 32, 41, 44, 55, 6005, 6009, 6013, 7001, 7026, 10016, 50036 and 51046. Other important features were: Warning and Critical. DBSCAN considered a total of 5 outliers. Analyzing their relevant features help understand the result.

Table 59 shows the relevant features of each entity in the DBSCAN outlier cluster. It is possible to see that the victim's features are all above the mean and mode metrics, as well as, the remaining entities

Entity	Feature	Value	Max	Mean	Mode
172.31.69.24 (victim)	7024, 7031	14	14	7	0
	7036	44	44	7	0
	37	2	2	0	0
	4200	2	2	0	0
	7045	2	2	0	0
	2, 16, 134	1	1	0	0
	4201, 20001	1	1	0.01	0
	TotEvents	83	83	23	39
	Error	28	28	16	26
	Information	54	54	8	0
172.31.67.82	414	8	8	0	0
	6	7	7	0	0
	12	2	2	0	0
	98	2	2	0	0
	18, 20, 25, 27, 32, 41	1	1	0	0
	55, 6005, 6009, 6013,	1	1	0	0
	7001, 7026, 10016,	1	1	0	0
	50036, 51046	1	1	0	0
	Warning	8	8	0	0
	Critical	1	1	0	0
172.31.67.13 and 172.31.64.67	44	1	1	0	0
	TotEvents	1	83	23	39
	Information	1	54	8	0

Table 59: Features of the considered outliers in DBSCAN (28-02-2018).

considered as outliers. Nevertheless, it is also important to observe that most of their relevant features came from different event IDs.

Algo	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
KMeans	10:30	30m	1	1	1	150	0	0.99	0.5	1	0.7
DBSCAN	10:30	30m	1	1	1	150	0	0.99	0.5	1	0.7
Agglom	10:30	30m	1	1	1	150	0	0.99	0.5	1	0.7

Table 60: Results of the best time-window from day 28/02/18 with ensemble.

In Table 60 it is possible to observe the improvement of the results with the ensemble method. DBSCAN was the only one that improved the results by matching KMeans and Agglomerative.

7.2.2 Infiltration (01-03-18)

The attack performed was an infiltration similar to the previous one. Table 61 shows the results of the attack that occurred between 9H57 and 10H55. Again the best time-window was 10min. It was possible to isolate the victim for this period, nevertheless, the attack from 14:00 to 15:37 was not detected in any time-window. In this time-window, DBSCAN considered 6 entities as outliers, with the victim achieving the highest outlier score from LOF.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
KMeans	10:00	10m	1	-	1	0	91	0	1	1	1	1
DBSCAN	10:00	10m	1	-1/-1	1	16	75	0	0.83	0.06	1	1
Agglom.	10:00	10m	1	-	1	0	91	0	1	1	1	1

Table 61: Results of the best time-window from day 01/03/18.

In this time-window, our system extracted 12 features. This attack was detected based on the following

relevant features: event IDs 16, 104 and 36888. The victim was the only one to record these event IDs in 92 features. On the other way, the remaining victim's features were considered 'normal' since two of them are bellow median and mode value (Table 62).

Entity	Feature	Value	Max	Mean	Mode
172.31.69.13	7024, 7031	1	5	3.8	4
	7036	5	8	3.9	4
	7000, 7009	1	1	0.1	0
	16, 104	1	1	0	0
	7040	0	2	0.04	0
	36888	0	2	0.02	0
	TotEvents	11	18	12	12
	Error	4	10	7.8	8
	Information	7	8	4.0	4

Table 62: Victim features extracted from time-window 10:00-10:10 in day 01/03/18.

Analyzing Table 62 and based on the metrics presented it is possible to justify the not so good results obtained. Once victim features do not stand out significantly from other entities, grouping this entity with other ones was expectable.

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
KMeans	10:00	10m	1	1	0	91	0	1	1	1	1
DBSCAN	10:00	10m	1	1	0	91	0	1	1	1	1
Agglom.	10:00	10m	1	1	0	91	0	1	1	1	1

Table 63: Results of the best time-window from day 01/03/18 with ensemble.

Table 63 shows the results after applying our ensemble method. It is possible to see a significant increase in DBSCAN metrics. Once again, our ensemble method was able to improve the results.

7.2.3 Bot (02-03-18)

The Bot attack was performed against 10 victims and using our criteria of outlier classification we were not expecting to achieve good results. Nevertheless, our system detected an anomaly minutes later from the last attack period (ended at 15H55). Those results are presented in Table 64.

Alg	Start	Dur	#Mac	LOF	TP	FP	TN	FN	Acc	Pre	Rec	F1
KMeans	16:10	10m	1	-	1	2	84	0	0.98	0.3	1	0.5
DBSCAN	16:10	10m	1	-1/-1	1	6	80	0	0.93	0.14	1	0.25
Agglom.	16:10	10m	1	-	1	2	84	0	0.98	0.3	1	0.5

Table 64: Results of the best time-window from day 02/03/18.

For this day our system detected 11 features which were: event IDs *7024, 7031, 7036, 36888, 7000, 7009, 35* and *37*, and *TotEvents, Error,* and *Information*.

Analyzing clusters and their entities regardless of their size, we could identify clusters that contained more than 50% of victims. For example, at 10:20, considering a time-window of 10 minutes, we identified a cluster of size 8 which contained 7 of the 10 victims. In this case, we could have adapted our system to consider clusters with less than 11 entities as outliers. Nevertheless, we decided to keep systems classification with a single entity. In a real case where we know neither the number nor the machines attacked, it is difficult to establish a threshold from which a cluster can be considered anomalous.

In Table 65 we observe the results after ensemble. Intersecting the clusters with a single entity (KMeans and Agglomerative) with the outliers cluster (DBSCAN) removed four entities from the DB-

Alg	Start	Dur	#Mac	TP	FP	TN	FN	Acc	Pre	Rec	F1
KMeans	16:10	10m	1	1	2	84	0	0.98	0.3	1	0.5
DBSCAN	16:10	10m	1	1	2	84	0	0.98	0.3	1	0.5
Agglom.	16:10	10m	1	1	2	84	0	0.98	0.3	1	0.5

Table 65: Results of the best time-window from day 02/03/18 with ensemble.

SCAN outlier cluster, which equals the results obtained with KMeans and Agglomerative. The ensemble increased results in DBSCAN, nevertheless for KMeans and Agglomerative the results did not improve.

7.2.4 Summary of Results

From the three days of attacks to Windows machines, we were able to detect the attacks performed in two of them. On the last day, the Bot attack was executed to more than one machine. It means with our classification criteria the probability of detecting all the victims was low, either because the victims have the same behavior and were grouped in a single cluster or they all had different behaviors and were isolated in 10 different clusters. The results in this source of data achieved a mean accuracy of 0.96, a mean precision of 0.44, a mean recall of 1 and a mean F1 of 0.66. With our method of clustering ensemble, the results achieved a mean accuracy of 0.99, mean precision of 0.6, mean recall of 1 and mean F1 of 0.733. A summary of the victims and the time-windows which were detected are presented in Table 66.

Windows Event Sources										
Attack	**Infiltration (28-02-18)**					**Infiltration (01-03-18)**				
Window	10m	30m	1H	2H	4H	10m	30m	1H	2H	4H
Detected	N	V	V	V	N	V	V	N	N	N
Attack	**Bot (02-03-18)**									
Window	10m		30m		1H		2H		4H	
Detected	V		N		N		N		N	

Table 66: Summary of Windows sources results by attack.

7.3 Merging Events and Flows

This chapter presents the results obtained from the combination of event and flow sources. We merged each flow approach with event features and applied our clustering algorithms and metric of evaluation. It resulted in not a single detection, from all approaches combination.

To understand these results, we should look at their results individually. Beginning with flow approaches and looking at Tables 49, 52 and 55, we can observe that in both Infiltration attacks the victims were detected and in Bot attack the attacker was found. It is important to notice that relevant features of Infiltration attack from day 28/02/18 (Table 50) were not high. This might indicate that victim behavior does not override normal behavior when merged with event features. Nevertheless, second Infiltration attack relevant features from day 01/03/18 show high values comparing with mean and mode. Considering relevant features from the event approach in this attack (Table 62) we can observe that most of the victim's values were above mean and mode. In this particular case, we found the victim based on high values from flow features and low values in event features, which proved to not be good to combine them. At last, looking at Bot attack we cannot compare the results since flow approaches detected the attacker and event approach detected one victim in 10.

7.3.1 Summary of Results

In the days which was possible to combine both events and flows, we were unable to detect a single anomaly. These results show that event logs from the system and flow approaches are incompatible, either because the big amount of features or attack footprints in event logs are unrelated with flows. Nevertheless, with a restricted choice of features, it should be possible to combine both.

8 Conclusions and Future Work

This work has shown the impact of the number of features in anomaly detection using an ensemble of clustering algorithms. It also contributed to testing how combined logs from windows and netflow perform on detecting anomalies in networks. Once data vary in each time-window, it was important to adapt the parameters of our algorithms to data in order to achieve algorithms parameters that fit better. The use of the *elbow method* to determine those parameters proved to be an extremely good improvement. Clustering algorithms chosen proved to be able to detect anomalies, with good precision and accuracy. Despite KMeans and Agglomerative belong to different classes (partition and hierarchical), their behavior for anomaly detection was similar in almost every trial. The use of LOF in the outlier cluster from DBSCAN was helpful sometimes, nevertheless, it should have performed better in detecting victim/attacker. The clustering ensemble method proposed proved to be helpful to improve DBSCAN results. In some cases, it also helped KMeans and Agglomerative, when only one of them wrongly considered more than one single cluster. For example, if KMeans classified one entity as outlier and Agglomerative consider two entities, the ensemble method improved the Agglomerative results.

Separating external entities from internal entities helped to join similar entities in the same subset. It means that internal entities should behave completely different from external entities, thus, making no sense to compare (i.e., cluster) all in the same set. The netflow approaches helped to conclude that the higher the number of features the lower the anomaly detection rate. From these approaches, we also conclude the importance of considering entities behavior in both points of view (source and destination). Depending on the attack the most relevant features can be from one or another point of view, as seen in relevant features tables.

From the host information (event logs) we realized that using the counter of events occurrence and their respective levels can produce results with an accuracy range of 91% to 99%. We believe that using application, security and system logs of Windows will lead to more accurate results. Once these logs contain precious data about not only the system but also application and security. For this reason, the dataset produced contains these logs information.

We had some expectations in the combination of netflow and event features because it merges two great sources of information about host and network. Nevertheless, the results did not correspond to our expectations. Those results can also have been achieved by increasing the number of features, which was proved to attenuate features importance.

As part of this work, a new dataset was obtained containing real data from an organization. It contains logs from application, security and system levels, as well as network traffic. This dataset is only available to students from the Military Academy due to sensible information contained.

For future work, one improvement could be to understand entities cluster changes. It means if one entity changes from one cluster to another in two followed time-windows it should be analyzed why it changed. This improvement is supposed to upgrade our classification definition allowing us to detect attacks performed at the same time to multiple machines.

Another improvement would be to analyze Windows OS logs from system, application, and security. This should provide richer data to the system, improving classification metrics. It should also be set a maximum limit of features in order to avoid the attenuation of relevant ones.

A possible improvement would be to consider a list of most important events for security from Windows OS. Restricting those events to the more frequent ones in each time-window and considering them as features.

Another improvement would be to consider CIC-CSE 2018 features and perform an extensive analysis of them. Defining the most relevant features and creating a system capable to collect such information in a real network.

An improvement in classification could be made by analyzing the distance between each cluster center and apply LOF to the farthest.

Another improvement would be to consider all ports and events and provide each one a degree of importance or weight. It will provide the system with increased relevance of known ports and events. This will help to reduce the attenuation when considering a big number of features.

References

[1] Mihael Ankerst et al. "OPTICS: ordering points to indentify the clustering structure". In: *ACM Sigmod record*. Vol. 28. 2. ACM. 1999, pp. 49–60.

[2] Baum, Leonard E., and J. A. Eagon. "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology". In: *Bull. Amer. Math. Soc.* 73.3 (May 1967), pp. 360–363.

[3] *Benchmarking Performance and Scaling of Python Clustering Algorithms*. URL: `https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html`. accessed: 05.06.2019.

[4] Konstantin Berlin, David Slater, and Joshua Saxe. "Malicious behavior detection using windows audit logs". In: *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*. ACM. 2015, pp. 35–44.

[5] Markus M Breunig et al. "LOF: identifying density-based local outliers". In: *ACM sigmod record*. Vol. 29. 2. ACM. 2000, pp. 93–104.

[6] Anna L. Buczak and E. Guven. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection". In: *IEEE Communications Suveys & Tutorials* 18.2 (2016).

[7] *CICFlowMeter (formerly ISCXFlowMeter)*. URL: `https://www.unb.ca/cic/research/applications.html#CICFlowMeter`. (accessed: 07.08.2019).

[8] G. Creech and J. Hu. "A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguous and Discontiguous System Call Patterns". In: *IEEE Transactions on Computers* 63.4 (Apr. 2014), p. 808.

[9] *CSE-CIC-IDS2018 on AWS*. URL: `https://www.unb.ca/cic/datasets/ids-2018.html`. (accessed: 24.07.2019).

[10] D.Laney. "3D Data Management: Controlling Data Volume, Velocity and Variety". In: *Application Delivery Strategies*. Technical Report 949 (Feb. 2001).

[11] S. Dua and X. Du. *Data Mining and Machine Learning in Cybersecurity. Data Mining, Inference and Prediction*. Auerbach Publications, 2011.

[12] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *KDD*. Vol. 96. 34. 1996, pp. 226–231.

[13] H. Farooq and N. Otaibi. "Optimal Machine Learning Algorithms for Cyber Threat Detection". In: (2018).

[14] G.Vigna and C.Kruegel. "Host-Based Intrusion Detection". In: (June 2005), p. 1.

[15] D. Gonçalves, J. Bota, and M. Correia. "Big Data Analytics for Detecting Host Misbehavior in Large Logs". In: *Proceedings of IEEE Trustcom*. 2015.

[16] H.Trevor, R.Tibshirani, and J.Friedman. *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. 2nd ed. Springer-Verlag New York, Feb. 2009.

[17] R. A. Ariyaluran Habeeb. "Real-time big data processing for anomaly detection: A Survey". In: *International Journal of Information Management* (Sept. 2018).

[18] R. A. Kemmerer and G. Vigna. "Intrusion detection: a brief history and overview". In: *Computer* 35.4 (Apr. 2002).

[19] Teuvo Kohonen. "The self-organizing map". In: *Proceedings of th IEEE* 78.9 (1990), pp. 1464–1480.

[20] Aleksandar Lazarevic et al. “A comparative study of anomaly detection schemes in network intrusion detection”. In: *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM. 2003, pp. 25–36.

[21] A. Lelli. *Out of sight but not invisible: Defeating fileless malware with behavior monitoring, AMSI, and next-gen AV*. URL: https://cloudblogs.microsoft.com/microsoftsecure/2018/09/27/out-of-sight-but-not-invisible-defeating-fileless-malware-with-behavior-monitoring-amsi-and-next-gen-av/. (accessed: 09.10.2018).

[22] Kingsly Leung and Christopher Leckie. *Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters*. ACSC 05. 2005, pp. 333–342.

[23] R. P. Lippmann et al. “Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation”. In: *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX’00*. Vol. 2. Jan. 2000, 12–26 vol.2.

[24] Gordon Lyon. *NMAP*. accessed: 13.10.2019.

[25] James MacQueen and others. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[26] S. Momeni Milajerdi et al. “HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows”. In: *2019 IEEE Symposium on Security and Privacy (SP)*. Vol. 00, pp. 430–445.

[27] D. Murcdoch. *Blue Team Handbook: SOC, SIEM, and Threat Hunting Use Cases*. Vol. 2. 2018.

[28] Fionn Murtagh and Pedro Contreras. “Algorithms for hierarchical clustering: an overview”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.1 (2012), pp. 86–97.

[29] *pandas*. URL: https://pandas.pydata.org/. accessed: 02.09.2019.

[30] H. Reia. “Online Security Analytics”. MA thesis. Instituto Superior Técnico, Oct. 2018.

[31] Dmitri G Roussinov and Hsinchun Chen. “A scalable self-organizing map algorithm for textual classification: A neural network approach to thesaurus generation”. In: (1998).

[32] Luis Sacramento et al. “FlowHacker: Detecting Unknown Network Attacks in Big Traffic Data using Network Flows”. In: *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE. 2018, pp. 567–572.

[33] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3.3 (June 1959), pp. 210–229.

[34] *scikit-learn*. URL: https://scikit-learn.org/stable/index.html. accessed: 03.09.2019.

[35] Fireeye Mandiant Services. *M-TRENDS 2019*. 2019.

[36] Iman Sharafaldin, Arash Habibi Lashkari, and Ali Ghorbani. “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”. In: (Jan. 2018), pp. 108–116.

[37] Iman Sharafaldin et al. “Towards a Reliable Intrusion Detection Benchmark Dataset”. In: *Software Networking* 2017 (Jan. 2017), pp. 177–200.

[38] *sklearn.cluster.AgglomerativeClustering*. URL: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html. accessed: 10.09.2019.

[39] *sklearn.cluster.DBSCAN*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN. accessed: 12.09.2019.

[40] *sklearn.cluster.KMeans*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans. accessed: 01.09.2019.

[41] *sklearn.neighbors.NearestNeighbors*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html. accessed: 02.09.2019.

[42] *sklearn.preprocessing.MinMaxScaler*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler. accessed: 05.09.2019.

[43] R. Franklin Smith. *Windows Security Log Events*. URL: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx. (accessed: 08.10.2018).

[44] A. Somayaji and S. Forrest. "Automated Response Using System-Call Delays". In: *9th Usenix Security Symposium* (Aug. 2000), pp. 185–197.

[45] T. Teles. "Cibersegurança: Deteção de outliers". MA thesis. Escola Naval, 2015.

[46] "The 2017 State of Endpoint Security Risk". In: *Ponemon Institute LLC* (Nov. 2007), p. 3.

[47] Jing Tian, Michael H Azarian, and Michael Pecht. "Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm". In: *Proceedings of the European Conference of the Prognostics and Health Management Society*. 2014.

[48] *Understanding the concept of Hierarchical clustering Technique*. URL: https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec. accessed: 10.09.2019.

[49] K. Veeramachaneni et al. *AI2: Training a Big Data Machine to Defend*. Apr. 2016, pp. 49–54.

[50] G. Vigna and C. Kruegel. "Host-Based Intrusion Detection". In: (June 2015).

[51] David Wagner and Paolo Soto. "Mimicry Attacks on Host-based Intrusion Detection Systems". In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*. 2002, pp. 255–264. ISBN: 1-58113-612-9.

[52] Dongkuan Xu and Yingjie Tian. "A comprehensive survey of clustering algorithms". In: *Annals of Data Science* 2.2 (2015), pp. 165–193.

[53] Rui Xu and Donald Wunsch. "Survey of Clustering Algorithms". In: *Neural Networks, IEEE Transactions on* 16 (June 2005), pp. 645–678.

[54] D. Yeung and Y. Ding. "Host-based intrusion detection using dynamic and static behavioral models". In: *Pattern Recognition* 36 (Dec. 2001), pp. 229–243.

A Windows Event Viewer Features

Windows Event Viewer			
Event ID	**Name**	**Features name**	**Type**
4723	Attempt to change account's password	psswdChange	Counter
4724	Attempt to reset account password	psswdReset	Counter
4725	User account disabled	disAccount	Counter
4726	User account deleted	delAccount	Counter
4738	User account changed	changeAccount	Counter
4624	Account successfully	logon	Categorical
4625	Account failed to	accountFail	Categorical
4648	Logon attempted	logAttempt	Counter
4740	A user account was lockedout	userLockedOut	Binary
1000	Application error	appError	Counter
41	System has rebooted without cleanly shutting down first	rebootSystem	Binary
104	Event Log was cleared	eventCleared	Counter
1069	Cluster resource in clustered service or application failed.	clusterFailed	Counter
7013	A service failed to start	srvcFaliedStart	Counter
7024	SCM cannot properly	srvcFailedTerminate	Counter
7032	The SCM tried to take a corrective action after the unexpected termination of service, but this action failed with error	correctionFailed	Categorical
7034	Service terminated unexpectedly	srvcUnexEnd	Categorical

Table 67: Summary of features that could be extracted from Windows Events Viewer [27, 43].

B CIC Features

Feature Name	Brief Description
FL_dur	Flow duration
TOT_fw_pk	Total packets in the forward direction
TOT_bw_pk	Total packets in the backward direction
TOT_l_fw_pkt	Total size of packet in forward direction
FW_pkt_l_min	Minimum size of packet in forward direction
FW_pkt_l_avg	Average size of packet in forward direction
FW_pkt_l_std	Standard deviation size of packet in forward direction
BW_pkt_l_max	Maximum size of packet in backward direction
BW_pkt_l_min	Minimum size of packet in backward direction
BW_pkt_l_avg	Mean size of packet in backward direction
BW_pkt_l_std	Standard deviation size of packet in backward direction
FL_byt_s	flow byte rate that is number of packets transferred per second
FL_pkt_s	flow packets rate that is number of packets transferred per second
FL_iat_avg	Average time between two flows
FL_iat_std	Standard deviation time two flows
FL_iat_max	Maximum time between two flows
FL_iat_min	Minimum time between two flows
FW_iat_tot	Total time between two packets sent in the forward direction
FW_iat_avg	Mean time between two packets sent in the forward direction
FW_iat_std	Standard deviation time between two packets sent in the forward direction
FW_iat_max	Maximum time between two packets sent in the forward direction
FW_iat_min	Minimum time between two packets sent in the forward direction
BW_iat_tot	Total time between two packets sent in the backward direction
BW_iat_avg	Mean time between two packets sent in the backward direction
BW_iat_std	Standard deviation time between two packets sent in the backward direction
BW_iat_max	Maximum time between two packets sent in the backward direction
BW_iat_min	Minimum time between two packets sent in the backward direction
FW_psh_flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
BW_psh_flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
FW_urg_flag	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
BW_urg_flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
FW_hdr_len	Total bytes used for headers in the forward direction
BW_hdr_len	Total bytes used for headers in the forward direction
FW_pkt_s	Number of forward packets per second
BW_pkt_s	Number of backward packets per second
PKT_len_min	Minimum length of a flow
PKT_len_max	Maximum length of a flow

Table 68: Continued on next page

Table 68: Continued from previous page	
Feature Name	**Brief Description**
PKT_len_avg	Mean length of a flow
PKT_len_std	Standard deviation length of a flow
PKT_len_va	Minimum inter-arrival time of packet
FIN_cnt	Number of packets with FIN
SYN_cnt	Number of packets with SYN
RST_cnt	Number of packets with RST
PST_cnt	Number of packets with PUSH
ACK_cnt	Number of packets with ACK
URG_cnt	Number of packets with URG
CWE_cnt	Number of packets with CWE
ECE_cnt	Number of packets with ECE
DOWN_up_ratio	Download and upload ratio
PKT_size_avg	Average size of packet
FW_seg_avg	Average size observed in the forward direction
BW_seg_avg	Average size observed in the backward direction
FW_byt_blk_avg	Average number of bytes bulk rate in the forward direction
FW_pkt_blk_avg	Average number of packets bulk rate in the forward direction
FW_blk_rate_avg	Average number of bulk rate in the forward direction
BW_byt_blk_avg	Average number of bytes bulk rate in the backward direction
BW_pkt_blk_avg	Average number of packets bulk rate in the backward direction
BW_blk_rate_avg	Average number of bulk rate in the backward direction
SUBFL_fw_pk	The average number of packets in a sub flow in the forward direction
SUBFL_fw_byt	The average number of bytes in a sub flow in the forward direction
SUBFL_bw_pkt	The average number of packets in a sub flow in the backward direction
SUBFL_bw_byt	The average number of bytes in a sub flow in the backward direction
FW_win_byt	Number of bytes sent in initial window in the forward direction
BW_win_byt	# of bytes sent in initial window in the backward direction
FW_act_pkt	# of packets with at least 1 byte of TCP data payload in the forward direction
FW_seg_min	Minimum segment size observed in the forward direction
ATV_avg	Mean time a flow was active before becoming idle
ATV_std	Standard deviation time a flow was active before becoming idle
ATV_max	Maximum time a flow was active before becoming idle
ATV_min	Minimum time a flow was active before becoming idle
IDL_avg	Mean time a flow was idle before becoming active
IDL_std	Standard deviation time a flow was idle before becoming active
IDL_max	Maximum time a flow was idle before becoming active
IDL_min	Minimum time a flow was idle before becoming active

Table 68: CIC features (adapted from [9]).

C Summary Of Results From CSE-CIC-IDS 2018 (Netflow Sources)

Netflow Sources										
Attack	**FTP (14-02-18)**					**SSH (14-02-18)**				
Window	10m	30m	1H	2H	4H	10m	30m	1H	2H	4H
1^{st} Approach	N	N	N	N	N	N	N	N	N	N
2^{nd} Approach	N	N	N	N	A	V/A	V/A	A	V/A	N
3^{rd} Approach	V/A	V/A	V/A	V	V	V/A	V/A	V/A	V/A	N
Attack	**DoS-GoldenEye (15-02-18)**					**DoS-Slowloris (15-02-18)**				
Window	10m	30m	1H	2H	4H	10m	30m	1H	2H	4H
1^{st} Approach	N	N	N	N	-	N	N	N	N	-
2^{nd} Approach	A	N	N	N	-	V	A	N	N	-
3^{rd} Approach	V/A	V/A	N	V/A	-	V/A	V/A	V/A	N	-
Attack	**DDoS-LOIC HTTP (20-02-18)**					**DDoS-LOIC UDP (20-02-18)**				
Window	10m	30m	1H	2H	4H	10m	30m	1H	2H	4H
1^{st} Approach	N	N	N	N	N	N	N	N	N	N
2^{nd} Approach	V	N	N	N	N	N	N	N	N	N
3^{rd} Approach	V	V	V	V	V	N	N	N	N	N
Attack	**DDoS-LOIC UDP (21-02-18)**					**DDoS-HOIC (21-02-18)**				
Window	10m	30m	1H	2H	4H	10m	30m	1H	2H	4H
1^{st} Approach	N	N	N	N	N	N	N	N	N	N
2^{nd} Approach	V	N	N	N	N	V	V	V	V	N
3^{rd} Approach	V	V	V	V	V	V	V	V	V	V
Attack	**Brute Force Web (22-02-18)**					**Brute Force XSS (22-02-18)**				
Window	10m	30m	1H	2H	4H	10m	30m	1H	2H	4H
1^{st} Approach	N	N	N	N	N	N	N	N	N	N
2^{nd} Approach	N	N	N	N	N	A	N	N	N	N
3^{rd} Approach	V/A	V	V	N	N	V/A	V/A	V	V	N
Attack	**SQL Injection (22-02-18)**					**Brute Force Web (23-02-18)**				
Window	10m	30m	1H	2H	4H	10m	30m	1H	2H	4H
1^{st} Approach	N	N	N	N	N	N	N	N	N	N
2^{nd} Approach	N	N	N	N	N	N	N	N	N	N
3^{rd} Approach	V/A	V/A	N	N	N	V/A	V/A	V	V	N
Attack	**Brute Force XSS (23-02-18)**					**SQL Injection (23-02-18)**				
Window	10m	30m	1H	2H	4H	10m	30m	1H	2H	4H
1^{st} Approach	N	N	N	N	N	N	N	N	N	N
2^{nd} Approach	V	N	N	N	N	N	N	N	N	N
3^{rd} Approach	V/A	V	V	V	V	A	N	N	N	N
Attack	**Infiltration (28-02-18)**					**Infiltration (01-03-18)**				
Window	10m	30m	1H	2H	4H	10m	30m	1H	2H	4H
1^{st} Approach	N	N	N	N	N	N	N	N	N	N
2^{nd} Approach	V	V	V	V	V	V	V	V	V	V
3^{rd} Approach	V	V	V	V	V	V	V	V	V	V
Attack	**Bot (02-03-18)**									

Table 69: Continued on next page

Table 69: Continued from previous page					
Netflow Sources					
Window	10m	30m	1H	2H	4H
1^{st} Approach	N	N	N	N	N
2^{nd} Approach	A	A	N	N	N
3^{rd} Approach	A	A	N	N	N

Table 69: Summary of Netflow Sources results by attack.