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

João Pedro Esteves Pinheiro
Instituto Superior Técnico
Academia Militar
joaopepinheiro@gmail.com

***Abstract*—In recent years, the world has witnessed a dramatic increase in cyberattacks. The threat landscape is evolving unprecedentedly, leaving organizations vulnerable to cyberattacks. Intrusion Detection Systems (IDS) aim to protect and monitor the threatened networks. IDSs must evolve faster than attackers to defend current systems consistently and robustly. The main goal of this work is to show the benefits of including Microsoft Windows Events (MSWEs) in current IDSs, taking advantage of the full range of data types. MSWEs provide an in-depth representation of the behaviour of Microsoft (MS) Windows machines. This system implements the dynamic extraction of both host and network features. The only user-defined instruction for feature extraction is the total number of features to extract. In addition, the system selects a fixed set of host and network features. The clustering process comprises three clustering algorithms working simultaneously to detect outliers. The system also explores four methods for outlier detection to overcome the difficulty of detecting multiple victims in the same attack. The proposed system is then evaluated with a public artificial dataset containing both data types. The evaluation metrics achieved by the different outlier detection metrics are then compared to similar approaches to demonstrate the benefits of including MSWEs in the system. The results show an increase of the F1-score by 3%, recall by 5%, and the same precision compared with the best-performing similar approach. The results achieved by the proposed system emphasise the contributions offered by this work to the field of IDSs.**

Keywords— Cybersecurity, Machine learning, Intrusion Detection System, Security Analytics, Netfow, Logs.

I. Introduction

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

Traditional Intrusion Detection Systems (IDS) use either features from network data or features collected from the hosts to detect possible attacks on the network. This work aims to develop a system that considers both approaches to deliver more precise results. Furthermore, the most common IDSs need information from previous attacks to determine suspicious behaviour, thus limiting the system's capacity to detect new attacks. Using machine learning and clustering techniques allows the system to detect abnormal behaviour worthy of being considered an attack.

The present work aims to fuse network data and host data to apply machine learning techniques to produce a new and improved intrusion detection approach to detect unseen attacks, providing a reliable, precise, and efficient system. The dataset used is the CSE-CIC-IDS2018 from the Canadian Institute for Cybersecurity.

II. Background

This section describes basic concepts fundamental to understanding the development and implementation of the proposed approach.

A. Network Flows

Network Flows are a popular data source for Network Intrusion Detection Systems (NIDS). Flow export is a passive network monitoring approach where packets are aggregated into flows and exported for storage and analysis [1].

NetFlow is the most widely used flow measurement solution. Patented in 1996 by Cisco, NetFlow was only widely adopted in 2002 with the launch of NetFlow v5. In 2004, NetFlow v5 was replaced by the more flexible NetFlow v9, which introduced support for adaptable data formats through templates, IPv6, Virtual Local Area Networks and Multi-Protocol Label Switching. Routers running NetFlow maintain a flow cache containing flow records describing the router's traffic.

A NetFlow flow is characterized as a unidirectional sequence of packets. Each flow has a set of seven values that establish unique keys: source Internet Protocol (IP) address, destination IP address, source port, destination port, protocol type, type of service and ingress interface [2]. NetFlow is generated by several algorithms that determine if a packet belongs to an existing flow or if a new one should be created. NetFlow collectors gather data from different sources and perform data reduction through filtering and aggregation. Then, the flow information is stored in flat files ready for further processing (e.g., analysis).

B. Microsoft Windows Events

Microsoft (MS) Windows is the most popular operating system for desktop and laptop computers worldwide and, as such, is targeted more frequently by malware authors than other operating systems [3]. Therefore, incorporating MSWEL in the proposed IDS is of most interest. MSWELs make available a large number of events that depict the current and past system's state [4]. MSWELs are a collection of MS Windows Events (MSWEs) organized by the source or nature of the MSWEs.

MSWEs provide a detailed description of a system's activities, representing an essential instrument for detecting unauthorized activities and security threats [5]. The 2012 Verizon Data Breach report states that 84 per cent of cyberattack victims had evidence of the breach in their event logs [6]. Each MSWE is identified with a unique Identification (ID) [7]. MSWEs are stored in Extensible Markup Language (XML) format, a binary language. As such, MSWE can be interpreted by the MS Windows Event Viewer, which translates MSWE from XML format to plain text format.

C. Intrusion Detection Systems

Soniya defines IDS as "an art of detecting the intruder activities" [8]. An IDS is the process of monitoring events occurred in a network or computer system to detect signs of malicious activities. After detecting an intrusion by a third party, an IDS sends a report to a base station. There are three primary IDSs [9]:

- NIDS: Unlike the previous IDS, NIDS receives data from the network traffic and monitors the global system or network communications with a low implementation cost. NIDS can inspect traffic and detect attacks at the application level.
- Host Intrusion Detection Systems (HIDSs) can monitor a computer's internal behaviour, detecting internal attacks and other attacks that might be undetectable by traditional NIDSs. The proposed approach is, in part, a HIDS resorting to MSWEs, which are generated in every Windows machine. These events provide helpful information about each system's operations and are an essential resource for an IDS.
- Clustering-based Intrusion Detection Systems are systems that apply clustering algorithms on the features extracted from the population of a network to detect and flag outliers.

D. Machine learning

Machine learning techniques are mainly used in the proposed approach to cluster the machines that belong to the network. Clustering is a process where objects with similar behaviour are grouped based on pre-selected features. After clustering the occurrences that do not belong to any cluster are classified as outliers. These outliers are then flagged as possible security risks. There are several algorithms to perform clustering. In this work are implemented three algorithms that have complementing strengths and weaknesses. Together these algorithms constitute a robust system. The chosen algorithms are K-means, DBSCAN and Agglomerative. K-means is a partitional algorithm that is sensible to initial conditions and assumes spherical clusters. In contrast, the Agglomerative is a hierarchical algorithm that recognises clusters of any shape. DBSCAN is a density-based algorithm that is robust to noise and that can handle variable cluster density. K-means is efficient with large datasets; however, the Agglomerative and DBSCAN algorithms struggle with these datasets.

III. Related Work

There are several papers that study the importance of MSWEs for intrusion detection [10, 11]. However, these papers only implement a search for known sequences of events in determined intervals that can be linked to known attacks. The proposed approach diverges from these related works by considering a dynamic selection of MSWEs.

Furthermore, the majority of related work in the area of Network Intrusion Detection Systems (NIDS) only takes into consideration fixed sets of features [12, 13]. The proposed approach implements dynamic feature selection to improve the adaptability of the system. The dynamic selection of network features is not a new approach, since it was already explored by DYNIDS [14] and by OutGene [15]. However, this work complements the advances achieved by DYNIDS with the inclusion of host data.

IV. Proposed Solution

This section describes the proposed solution. It includes a general and a detailed description of the proposed architecture and an overview of the implementation of the proposed approach.

A. General Description

As stated before, there is a need for new and improved IDSs at a time when cyberattacks increase each year. The vast amount of extracted data requires automated systems which can process the data to detect underlying patterns that a human analyst can overlook.

The proposed approach differs from the publications reviewed by integrating host and network data in a unique system with dynamically selected network and host features. Fig. 1 presents the proposed architecture:

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

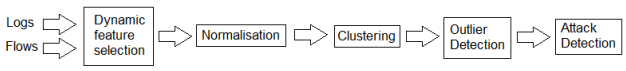

Fig. 1. Framework general description.

B. Detailed Description

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

1) Dynamic Feature Selection

The proposed system dynamically defines host and network features. Whereas in DYNIDS [14], only network features were used, in the proposed system, a procedure is established to integrate the selection of host features. Both sets of features have fixed and dynamically defined features.

For the network features, are extracted 12 fixed features and a variable number of dynamic features. Four features are defined dynamically for each selected port at run time proportional to the number of ports. The ideal number of selected ports (x) is 100, as stated by DYNIDS [14]. Then, we use DYNIDS as the basis for the definition of port-based features. The DYNIDS algorithm selects features based on the x/3 ports that appear in more flows, the x/3 ports that appear in fewer flows, and the x/3 ports used by fewer machines.

The selection of features for the host data follows the same rationale as the feature selection for network data. Seven fixed features are defined for the host data. Six of these features are related to the priority level of the MSWE. The feature Frequency_1 is the frequency of priority level 1 events for a given machine. For example, if feature Frequency_1 has a value of 0.5 for a given machine, 50% of the events for that specific machine have a critical (1) priority level. The features Frequency_2, Frequency_3, Frequency_4 and Frequency_5 are the same as Frequency_1 but for error (2), warning (3),

information (4) and verbose (5) priority level events, respectively. Furthermore, Total_Frequency is the total absolute number of events for every given machine. Finally, the feature NEventsPerMachine represents the absolute number of unique events for every machine.

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

2) Normalisation

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

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

3) Clustering

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

The K-means algorithm is a centroid-based algorithm that partitions the elements of the dataset in k clusters. The algorithm's objective is to minimize the variance within the same cluster and maximize the variance between different clusters. This algorithm is computationally efficient for extensive datasets. K-means has a time complexity of $O(n * k * i * d)$, where n is the number of data points, k is the number of clusters, i is the number of iterations until convergence, and d is the number of dimensions for each data point. The main disadvantage of the K-means algorithm is the need to specify k before run time; however, in our approach, the k value is defined at run time using the elbow method.

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

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

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

4) Oultlier Detection

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

However, on the second day (02-03-2018), there were ten individual victims. Therefore, the criteria for positive detection cannot be the existence of one-element clusters. This work introduces four possible methods to overcome the difficulty of selecting the victims' cluster. The related literature did not provide any insight into methods to make this selection. Furthermore, DYNIDS [14] also faced this problem and could not find a solution. The proposed methods are:

1. One-element cluster method: One-element clusters will be classified as outliers.
2. Machine scores: Measuring machine scores based on the population of the respective cluster in every algorithm. Machines with unusually low scores might be victims despite not being placed in one-element clusters. Machines with a score lower than 5% of the average score for the entire population will be classified as outliers.
3. Silhouette coefficient method: Measuring the mean Silhouette coefficient for every cluster in every algorithm. Silhouette analysis is the method of scoring the clusters based on the average measure of how similar the data points within that cluster are to each other and how dissimilar they are to data points in the nearest cluster they do not belong to. Clusters with less than half of the mean value of all clusters represent occurances with unusually low silhouette coefficient.

4. Low population method: Selecting clusters with less than ten elements. This method is not ideal since prior knowledge of the number of victims is used to establish this threshold. However, comparing the results obtained with the other methods might be of interest.

C. Implementation

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

The event log data is stored in EVTX files in XML format, the default file format used by the MS Windows OS. A Python script was developed to parse the records from XML format to a CSV file, which is then used as input for the feature extraction process. DYNIDS [30] provided the CSV files with the network traffic data. The information from the network traffic was extracted with CICFlowMeter (V4.0) from the PCAP files made available by the authors of the dataset. The CSV files for both data types are the input of two Python scripts, one for each CSV file, that perform the feature extraction and the normalisation of the host and network features. The output of these scripts is a CSV file, one with network features and the other with host features, which are then the input of the final script that performs the clustering of both sets of features jointly.

The last step, as mentioned before, is implementing the outlier detection methods that determine the presence or absence of attacks. Fig. 2 illustrates the implementation of the system. The figure is divided in different coloured rectangles that allude to the different Python scripts developed.

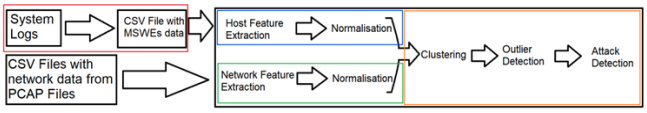

Fig. 2. Implementation of the proposed approach where each rectangle represents a Python script.

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

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

V. Experimental evaluation

In this section, the evaluation metrics are calculated for the results achieved by the proposed approach with host data and with the combination of host and network data.

A. Evaluation metrics

This section presents the evaluation metrics used to measure the performance of the system. These metrics classify the results obtained and the effectiveness of the approach. The proposed system will be evaluated regarding the following parameters:

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

B. Evaluation of the system considering host data

This subsection presents the results obtained by clustering the host data from the dataset CSE-CIC-IDS2018. This data was extracted from MSWELs. There are seven fixed features: NEventsPerHost, Frequency_1, Frequency_2, Frequency_3, Frequency_4, Frequency_5, and Total_Frequency. The feature NEventsPerHost provides the number of unique MSWE IDs the machine produces. The features Frequency_1, Frequency_2, Frequency_3, Frequency_4, and Frequency_5 are the proportion of MSWEs classified as critical (1), error (2), warning (3), Information (4), and verbose (5). Finally, the feature Total_Frequency represents the total number of events per machine registered for the day in analysis.

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

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

- X/3 less frequent MSWE IDs.

1) BOT attack (02-03-18), host data

There are two types of attacks on the days selected from the dataset. The first in analysis is the BOT attack. The attack classified as BOT by the CSE-CIC-IDS2018 dataset refers to a botnet attack executed on 02-03-2018 that compromised ten machines. These machines, known as bots, are under the control of an attacker. The dataset authors use Zeus, a Trojan horse malware package that runs on MS Windows [16]. In addition to Zeus, they use Ares botnet, an open-source botnet with added capabilities. In this attack, the bots were ordered to take screenshots every 400 seconds.

Table I presents the results obtained by the proposed approach for host data on 02-03-18 with different values of x. The results illustrate the convergence of the dispersion of the victims and demonstrates that a value of x=100 is ideal. Test with higher number of x were performed but the performance did not increase.

TABLE I. RESULTS OBTAINED WITH HOST DATA FOR DIFFERENT VALUES OF X FOR EACH ALGORITHM ON 02-03-18.

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

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

After clustering, outlier detection methods were applied. There are four methods to make this selection, however the first method is not applied since none of this clusters had a population of one.

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

TABLE II. SCORES OF THE VICTIMS ON DAY 02-03-18 WITH X=100 AND HOST DATA.

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

The third method for selecting the victim's cluster consists of analysing the mean Silhouette coefficient of every cluster for each algorithm. Fig. 3 presents the mean silhouette coefficients for every cluster of the K-means algorithm.

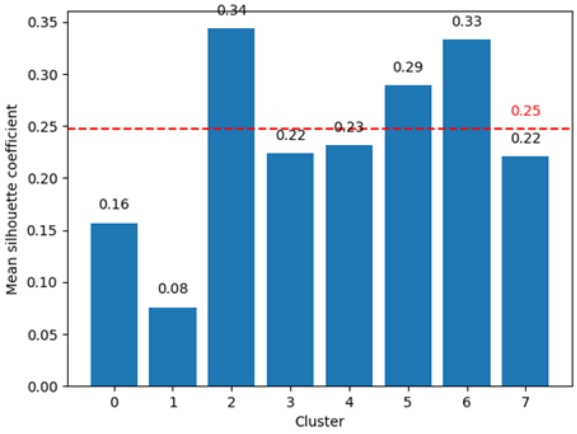

Fig. 3. Mean silhouette coefficient for the clusters in the K-means algorithm for x=100 on 02-03-18. The dashed red line represents the average of all the clusters. The value in red represents the average silhouette coefficient for every cluster.

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

For the DBSCAN algorithm this method did not provide results since every cluster has a means silhouette coefficient near the average value.

In contrast with the results obtained for the previous algorithms, the Agglomerative algorithm presents good results. As illustrated in Fig. 4, Cluster 3 has a mean silhouette coefficient of 0.1 and well below half of the total average silhouette coefficient.

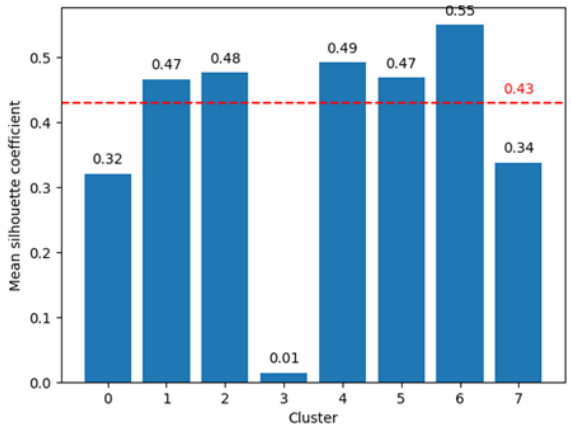

Fig. 4. Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100 on 02-03-18 with host data.

The fourth and last method for outlier detection considers every cluster with less than ten elements as a positive occurrence. With this criterion, the K-means algorithm classifies cluster 6, which only has a population of 6 elements, as the positive cluster. The DBSCAN algorithm classifies cluster 2, which only has a population of 6 elements, and the elements of cluster -1 as outliers. Finally, the Agglomerative algorithm considers cluster 4, with a population of 7 machines, the positive occurrence with this criterion.

The evaluation metrics for every method and for each algorithm are presented in Fig. 5. The best performing outlier detection method is the third method since the other methods were unable to select true positive occurrences.

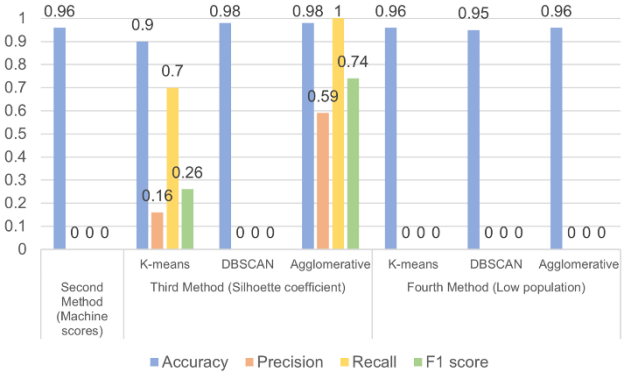

Fig. 5. Evaluation metrics for every outlier detection method on day 02-03-18 for host data.

3) *Infiltration attack (01-03-18), host data*

The second type of attack is the infiltration attack (01-03-18). In an infiltration attack, the attacker gains unauthorised access to a vulnerable application. In the specific case of the attack in focus, the victim received a malicious document through email [16]. The authors used the Metasploit framework to execute a backdoor in the victim's computer, where it is possible to launch a diverse set of attacks.

In contrast with day 02-03-18, the attacks of day 01-03-18 only targeted one victim. Firstly, there is the need to analyse the behaviour of this victim with the conditions that provided the best results in the last section. Therefore, the starting conditions will be x =100 and the seven fixed features. Table III presets the results achieved.

TABLE III. CLUSTERING RESULTS FOR THE INFILTRATION ATTACKS (01-03-18) ON HOST DATA AND X=100.

K-means			**DBSCAN**			**Agglomerative**		
Cluster	***Population***	***Victim***	***Cluster***	***Population***	***Victim***	***Cluster***	***Population***	***Victim***
0	75	-	**-1**	**7**	**1**	0	66	-
1	133	-	0	395	-	1	48	-
2	66	-	1	6	-	**2**	**12**	**1**
3	**12**	**1**	2	3	-	3	126	-
4	41	-	3	2	-	4	55	-
5	57	-	-	-	-	5	75	-
6	17	-	-	-	-	6	16	-
7	1	-	-	-	-	7	3	-
8	11	-	-	-	-	8	12	-

4) *Oulier detection with host data and x=100 on 02-03-18 (infiltration)*

Even though there is only one victim on this day, the same methods established for outlier detection will be applied, given that, in real conditions, the system does not know how many victims are in the network.

The first method is the most straight forward and consists in selecting clusters with a population of one. By analysing Table III, the conclusion can be made that only the K-means and the DBSCAN produced one-element clusters. K-means in cluster 7 and DBSCAN in each one of the 7 elements that constitute cluster -1.

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

TABLE IV. SCORE RESULTS FOR THE SECOND METHOD ON 02-03-18 WITH HOST DATA. THE VICTIM IS HIGHLIGHTED IN RED.

IP Address	**K-means**	**DBSCAN**	**Agglomerative**	**Score**	**Rank**
172.31.67.116	7	-1	7	0.009685	1
172.31.69.13	**6**	**-1**	**6**	**0.058111**	**2**
172.31.69.126	3	-1	2	0.058111	3
172.31.67.77	3	-1	2	0.058111	4

The Silhouette coefficient method is the third method for outlier detection. Fig. 6 presents the mean silhouette coefficient for the K-means algorithm, where the cluster of the victim is cluster 3. The only cluster that meets the criteria for an outlier classification is cluster 7.

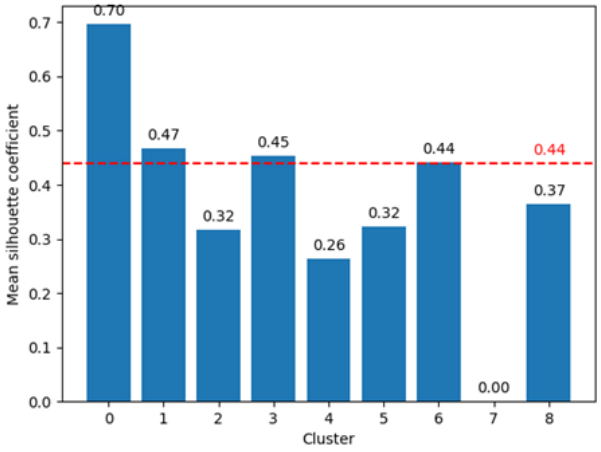

Fig. 6. Mean silhouette coefficient for the clusters in the K-means algorithm for x=100 on 01-03-18 with host data.

Furthermore, the DBSCAN does not provide any cluster that meets the criteria for positive occurrence according to the third method. Fig. 7 presents the mean silhouette coefficient for the clusters of the Agglomerative algorithm. The only cluster that fulfils the criteria for positive occurrence is cluster 1.

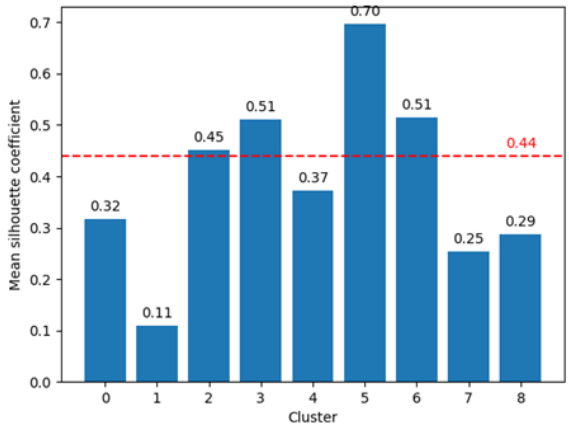

Fig. 7. Mean silhouette coefficient for the clusters in the Agglomerative algorithm for x=100 on 01-03-18 with host data. The dashed red line represents the average of all the clusters. The value in red represents the average silhouette coefficient for every cluster.

The fourth and last method consists of considering every cluster with less than ten elements as positive occurrences. The results of Table III allow the determination of the number of false positives, false negatives, true negatives, and true positives necessary to calculate the evaluation metrics.

The K-means algorithm has only one cluster with a population lower than ten elements. This cluster is cluster 7 which is a one-element cluster. The DBSCAN algorithm has three clusters plus the population of cluster -1. These machines add to eighteen. The Agglomerative algorithm only has cluster 7 with a population lower than ten.

5) Summary of results with host data

In summary, the methods for outlier detection did not achieve the desired performance. On 02-03-18, the first, second, and fourth methods were unable to produce any true positive occurrences. The third method also performed poorly for the DBSCAN algorithm. In contrast, the metrics for the K-means and Agglomerative algorithms are closer to similar approaches. The Agglomerative algorithm, with the third method, achieved the best results with an accuracy of 0.98, a precision of 0.59, a recall of 1, and an F1 score of 0.74. However, these results still fall short of expectations. Fig. 8 illustrates the results achieved by every method on 02-03-18.

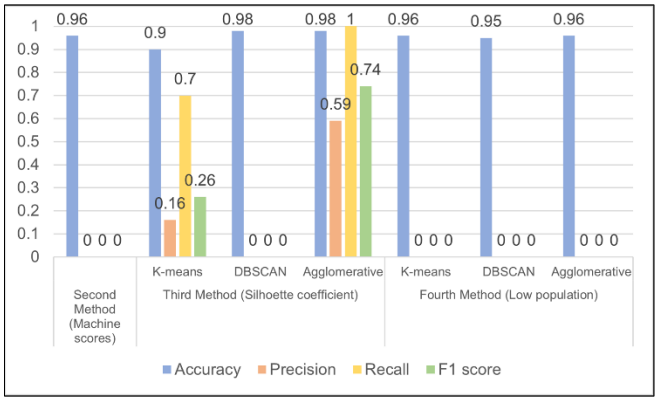

Fig. 8. Evaluation metrics for every outlier detection method on 02-03-18 for host data.

Fig. 9 compares the evaluation metrics for all the outlier detection methods on 01-03-18. DBSCAN was the algorithm that performed the best, while the other algorithms could not detect any true positive occurrences. Compared with the results of the attack on 02-03-18, the results on 01-03-18 are considerably worse. The best evaluation metrics were obtained for the second method and the DBSCAN algorithm with an accuracy of 0.99, a precision of 0.2, a recall of 1, and an F1 score of 0.33. These metrics represent a decrease of 66% in precision and 55% in F1 score compared to the best-performing combination on 02-03-18. However, these results are poor when compared with similar approaches. The host features considered are not sufficient to identify the victims of attacks consistently and precisely.

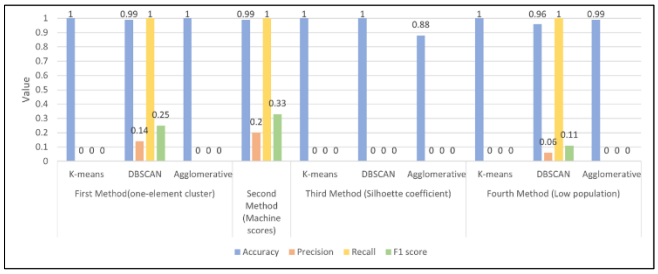

Fig. 9. Evaluation metrics for every outlier detection method on 01-03-18 for host data.

C. *Evaluation of the system considering host and network data*

This section presents the results achieved by the proposed approach with host and network data.

1) BOT attack (02-03-18), network and host data

Table V illustrates the results achieved by the clustering of network and host features for each considered algorithm. The results show that the majority of the victims are concentrated in the same cluster both for the K-means and the Agglomerative algorithms. In contrast, the results for the DBSCAN are very poor as it is not able to detect a single true positive occurrence.

TABLE V. CLUSTERING RESULTS FOR THE BOT ATTACK (02-03-18) WITH NETWORK AND HOST DATA AND X=100.

K-means			DBSCAN			Agglomerative		
Cluster	***Population***	***Victim***	***Cluster***	***Population***	***Victim***	***Cluster***	***Population***	***Victim***
0	66	-	-1	3	-	0	159	-
1	22	-	**0**	**204**	**10**	1	38	-
2	**39**	**1**	1	203	-	**2**	**38**	**2**
3	39	-	2	4	-	3	32	-
4	73	-	-	-	-	4	61	-
5	135	-	-	-	-	5	6	-
6	7	-	-	-	-	6	23	-
7	**9**	**9**	-	-	-	**7**	**8**	**8**
8	32	-	-	-	-	8	41	-
9	1	-	-	-	-	9	7	-

2) Oulier detection with host, network data and x=100 on 02-03-18 (BOT)

The first method consists in considering as a positive occurrence one-element clusters. The K-means algorithm produces a one-element cluster which is not a victim. The DBSCAN algorithm classified three machines in cluster -1 but all three are not victims. Finally, the Agglomerative algorithm produced no one-element cluster. Therefore, no algorithm was able to identify a true positive instance with this method.

The second method consists of selecting the clusters that have a score lower than 5% of the average score of the entire population. The average score is 0.859391. Therefore, the machines with a score below 0.042970 will be selected. With this criterion only one non-victim machine is selected with a score of 0.016548. The scores of the victims are summarized in Table VI.

TABLE VI. SCORES OF THE VICTIMS ON 02-03-18 WITH HOST AND NETWORK DATA AND X=100.

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

The third method for outlier detection consists of selecting the cluster with a mean Silhouette coefficient lower than half of the total mean silhouette coefficient. This method achieved the results illustrated in Table VII.

TABLE VII. RESULTS ACHIEVED BY THE THIRD METHOD (SILHOUETTE COEFFICIENT) FOR HOST AND NETWORK DATA AND X=100.

	K-means	DBSCAN	Agglomerative
TP	1	0	2
FP	99	216	36
TN	314	197	377
FN	9	10	8

The fourth and final method consists in selecting the cluster with less than ten elements as positive occurrences. The results achieved by this method are presented in Table VIII.

TABLE VIII. RESULTS ACHIEVED BY THE FOURTH METHOD (CLUSTERS WITH LESS THAN TEN ELEMENTS) FOR HOST AND NETWORK DATA AND X=100.

	K-means	DBSCAN	Agglomerative
TP	9	0	8
FP	8	10	13
TN	405	403	400
FN	1	10	2

3) Infiltration attack (01-03-18), network and host data

On this day there is only one victim. Therefore, it is expected that a cluster with only one element created by the algorithms indicates an outlier. The analyses of the results obtained by the clustering of the K-means algorithm show that the victim is placed in a single element cluster.

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

The Agglomerative algorithm also produced good results placing the victim in a one-element cluster. The victim is placed in cluster 6 that only has one element, as illustrated by Table IX.

These results correspond to the initial purpose of this work. The victim was placed by every algorithm in an outlier cluster. From this result it is easy to flag machines that are victims by simply flagging one-element clusters.

TABLE IX. CLUSTERING RESULTS FOR THE INFILTRATION ATTACKS (01-03-18) ON HOST DATA AND X=100.

K-means			DBSCAN			Agglomerative		
Cluster	***Population***	***Victim***	***Cluster***	***Population***	***Victim***	***Cluster***	***Population***	***Victim***
0	55	-	**-1**	**2**	**1**	0	74	-
1	75	-	0	366	-	1	11	-
2	146	-	1	45	-	2	61	-
3	11	-	-	-	-	3	28	-
4	29	-	-	-	-	4	75	-
5	67	-	-	-	-	5	138	-
6	**1**	**1**	-	-	-	**6**	**1**	**1**
7	29	-	-	-	-	7	25	-

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

Table X presents the scores of the top 5 machines. The score of the victim is considerably lower than the score of the second-placed machine. The average of the scores was 1.204792, therefore only machines with a score lower than 0,060239 (5% of the average) are selected. Consequently, only the victim is selected by the first method.

TABLE X. SCORES OF THE TOP 5 MACHINES ON DAY 01-03-18 WITH NETWORK AND NETWORK DATA AND X=100.

IP Address	Score	Rank
172.31.69.13	**0.004843**	**1**
172.31.66.114	0.205811	2
172.31.66.112	0.205811	3
172.31.64.67	0.205811	4
172.31.64.26	0.205811	5

5) Summary of results with host and network data

Fig. 10 illustrates the results obtained for each outlier detection method for host and network data on 02-03-18. The fourth method achieved the best results. The DBSCAN continues to provide poor results on the BOT attack. The large number of victims seems to have a more significant impact on this algorithm.

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

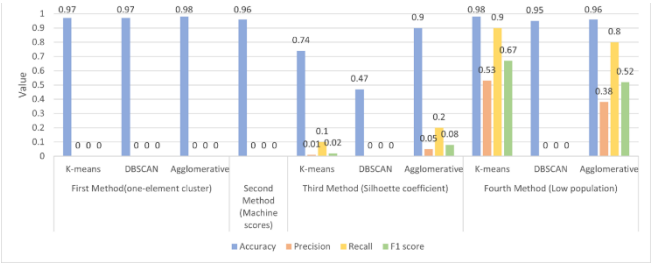

Fig. 10. Evaluation metrics for every method described in Subsection 4.2.4 for host and network data on 02-03-18.

The proposed system is much more effective on 01-03-18 (infiltration). The results, presented in Fig. 11, show that the system provides excellent evaluation metrics for all methods except for the third. These results are an improvement over similar approaches.

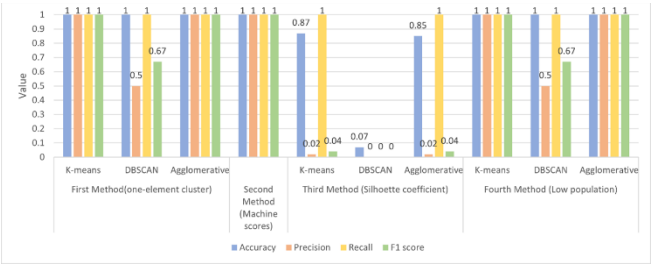

Fig. 11. Evaluation metrics for every outlier detection method for host and network data on 01-03-18.

D. *Comparison with different algorithms*

In this section, the results achieved are compared with similar systems that used the CSE-CIC-IDS2018 dataset, for the day 01-03-2018, resorting to the first outlier detection method.

Fig. 12 presents the evaluation metrics achieved by several systems for the CSE-CIC-IDS2018. The F1-Score achieved by the proposed approach is 3% higher than the F1-Score obtained by DYNIDS and 5% higher in Recall. The precision of both systems is the same. Furthermore, the proposed system is able to outperform OutGene and Flow Hacker in all metrics.

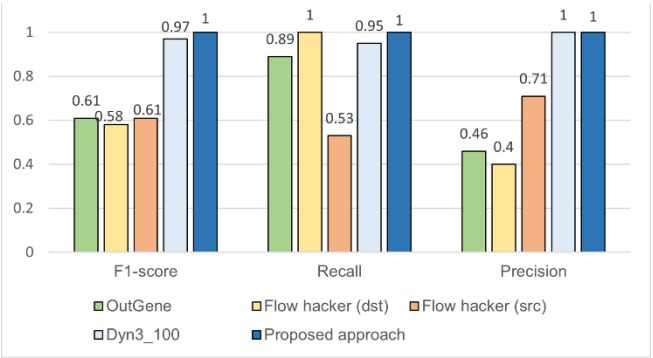

Fig. 12. Comparison of the performance of the proposed approach (host data and network data) versus other approaches.

The proposed system achieved the desired performance in all the metrics where it was evaluated. The results illustrate the benefits of including host data, more precisely, MSWEs in an IDS. The comparison with DYNIDS and the improvement achieved show that both data types (host and network data) are compatible and can coexist in the same IDS. Furthermore, the valuable insights given by the MSWEs of the network should be utilised to complement the network data. Attacks, such as

Botnet where the targeted machines perform automated commands simultaneously can stay undetected under a typical NIDS. However, the abnormal production of rare MSWEs is easily detected by the proposed system.

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

VI. Concluison and future work

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

The chosen clustering algorithms proved to be sufficient to achieve the expected results. The main objective was to demonstrate that adding host data could improve current IDS systems. The results for day 01-03-18, when there was only one victim, were significantly superior to similar approaches. On this day, with the combination of host and network data, every algorithm could consistently detect the victim and place it in a one-element cluster. However, the attacks of day 02-03-18 proved to be more challenging to the proposed approach. The algorithm DBSCAN performed poorly for this attack, unable to make a single detection. The K-means and Agglomerative algorithms performed better than DBSCAN but achieved poorer results than similar approaches.

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

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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