Detecting Collaborative Scanners Using Clustering Methods

Andrei Ionescu[1]

Supervisor(s): Georgios Smaragdakis[1]**, Harm Griffioen**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Andrei Ionescu
Final project course: CSE3000 Research Project
Thesis committee: Georgios Smaragdakis, Harm Griffioen, Kubilay Atasu

Abstract

This paper investigates the effectiveness of various clustering algorithms in detecting collaborative Internet scanning groups. The packet dataset used is collected from TU Delft's network telescope, and is aggregated into scanning sessions and analyzed using K-Means, Hierarchical Clustering, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Clustering Using Representatives (CURE), and Bradley-Fayyad-Reina (BFR). This paper also introduces an evaluation framework based on five degrees of certainty to assess the likelihood that a cluster is collaboratively scanning. The findings indicate that DBSCAN consistently outperforms other methods in identifying collaborative scanning groups, while CURE shows superior performance to BFR, K-Means, and Hierarchical Clustering. It is hoped that these insights help provide a strong foundation for enhancing network security through improved detection of collaborative scanning behaviors.

1 Introduction

1.1 Background

Cybersecurity has always been and always will be a game of cat and mouse. The first step in this game involves the attackers finding and abusing a vulnerability, and the second step consists of the cybersecurity specialists finding a way to resolve the vulnerability and impede the attackers, after which it is the attackers' turn to find or introduce a new vulnerability. This back-and-forth also applies to malicious Internet scanning.

Internet scanning refers to the systematic process of scanning various networks and systems across the Internet to gather information about them. It involves sending data packets to target Internet Protocol (IP) addresses and analyzing the responses to identify open ports, services, or other characteristics of the target systems. Internet scanning can be conducted for various purposes, including security assessments, network mapping, reconnaissance, or research, in which cases it is called *benign*. However, Internet scanning can also be leveraged by hackers and other types of adversaries to identify vulnerable systems, in which case it is called *malicious*.

Initially, attackers massively scanned the Internet to find which hosts were active and what services they ran, as this is the first step to compromising them [1]. Later on, research conducted by cybersecurity specialists revealed a way to detect and block these scanners based on the large number of packets they would send using intrusion detection systems (IDS) and firewall thresholds. In response to this, attackers started distributing their scans over multiple hosts to stay below the thresholds and, as such, they remain undetected. That is the state in which we find ourselves today, and it is now the turn of cybersecurity specialists to find a way to detect these collaborative scanning groups.

1.2 Motivation

Detecting collaborative scanners is crucial because these coordinated efforts by attackers can evade traditional detection methods, leading to significant security risks. Traditional systems often flag high activity from a single IP address as suspicious, but when scanning activity is divided among many sources, it appears as low-intensity traffic from each source, which is less likely to trigger alarms [1]. Due to the nature of the Internet, every single vulnerability can be exploited at scale, resulting in widespread sensitive data leaks, disrupting services, and causing financial damage.

Finding these collaborative scanners requires new methods that can detect subtle patterns in large amounts of data. Cutting-edge methods, such as the one presented by Griffioen et al. [1], fall short when the packets' fields are all randomized and no fingerprints can be distinguished. One promising possible approach to solving this shortcoming is clustering, which could be used to uncover hidden temporal structures in data and help identify behavior linked to collaborative scanning.

Clustering is a promising approach for several reasons. Firstly, clustering algorithms are designed to identify groups within data that share similar characteristics, making them well-suited for detecting coordinated behavior among seemingly random network activities. By grouping data points that exhibit similar temporal patterns, clustering can reveal the presence of collaborative scanners even when individual packets do not exhibit identifiable fingerprints. Secondly, clustering can handle large volumes of data effectively, which is essential given the scale of Internet traffic. Thirdly, clustering algorithms can adapt to evolving scan patterns. As scanners modify their strategies to evade detection, clustering may still be able to identify new patterns of behavior that emerge as a result of these changes.

This paper explores how clustering methods can be used to detect collaborative scanning. By looking closely at different clustering algorithms and how they might work for this task, the aim of this research is to contribute to creating new ways to defend against malicious parties on the Internet.

1.3 Research Questions

The main question that this paper aims to answer is:
"Is it possible to detect collaborative scanners using clustering methods?"

In order to guide the research process, the following subquestions have been derived:

- **RQ1**: *"How do collaborative scans work and what assumptions does the proposed methodology make?"*
- **RQ2**: *"What data attributes should be considered for clustering?"*
- **RQ3**: *"Once a cluster has been identified, how can we check that it is indeed a collaborative scanning group?"*
- **RQ4**: *"What clustering methods should be used?"*
- **RQ5**: *"What values should the hyperparameters have?"*

- **RQ6**: *"If multiple clustering approaches can detect collaborative scanning groups, how do their performances compare?"*

1.4 Contributions

This paper makes the following contributions:

- It proposes a new methodology that uses temporal correlations in network traffic to detect collaborative scanners, moving beyond traditional packet header analysis.
- It applies and compares various clustering algorithms to effectively identify collaborative scanners, even when individual packet fields are fully randomized.
- It proposes a new evaluation mechanism which utilizes five degrees of certainty to assess the likelihood that a cluster is a collaboratively scanning group.

1.5 Structure

The remainder of this paper is structured as follows: Section 2 presents the previous work conducted towards the detection of collaborative scanners, Section 3 details the methodology used for the research procedure, spanning the processes of data collection (Section 3.1), aggregating data and selecting attributes for training (Section 3.2), clustering (Section 3.3), and hyperparameter optimization (Section 3.4). Section 4 presents and analyses the experimental setup and the ensuing results, explaining the concept of degrees of certainty (Section 4.1), presenting method validation (Section 4.2), followed by a comparison of the clustering methods considered for this research paper (Section 4.3), an analysis of the clusters which are not described by any of the proposed degrees of certainty (Section 4.4), and lastly a presentation of the answers formulated to the research sub-questions (Section 4.5). The research at hand will be framed in the context of responsible research in Section 5, discussing ethical implications and privacy concerns. In Section 6, the produced results and the method's limitations are discussed. Lastly, Section 7 concludes the research findings and proposes potential future research points.

2 Related Work

This section will explore previous research which has attempted to solve the problem of detecting collaborative scanners, and present the shortcomings of the methods proposed by these papers.

There have been multiple studies which have corroborated the existence of advanced adversaries using distributed scanners to stay below firewall and IDS packet thresholds [2] [3], and a number of solutions have been proposed to identify these collaborative groups. Gates [4] employs the set cover technique to identify connections between IP addresses, but the set cover problem is NP-complete [5], and therefore applying this algorithm is impractical for handling large volumes of data. Robertson et al. [6] identify distributed scans provided that the IP addresses involved in these scans are situated within the same subnet, but this is a strong assumption to make seeing as adversaries can avoid detection by this method by simply renting or using hosts on differing subnets. In fact, there already exists proof of the existence of large collaborative scanning groups utilizing hosts belonging to different subnets [2], [3]. Yegneswaran et al. [7] detect coordinated behavior by examining destination ports and IP addresses to identify slow-scanning malware, discovering that many scans exhibit coordinated characteristics, with IP addresses displaying the same behavior. Their evaluation focuses on the destination ports and IP addresses in the packet headers, ignoring other fields.

Griffioen et al. [1] demonstrate that the above methods are inadequate for reliably detecting collaborative port scans in two main ways: firstly, at the scale of the Internet or within a large organization, the volume of incoming data is too vast to analyze for complementarity, and the prefiltering required for Gates' set cover solution [4] is usually unavailable. Secondly, as highlighted by Blenn et al. [2], adversaries often disregard subnet boundaries, with their activities spanning different network ranges, Internet Service Providers (ISPs), and country borders. The method proposed by Griffioen et al. [1] instead involves detecting distributed and stealthy network scanners by clustering source IP addresses based on commonalities in their scan probes and header values. The method shows high precision and recall in detecting scanner groups that use common scanning tools or have detectable patterns in their scan behavior. However, it has significant shortcomings when the scanners exhibit almost no commonality within their packets' header values, making it difficult to identify them. Specifically, the method struggles with groups that randomize almost all fields, do not use XOR relations, and are distributed over many hosts sending low amounts of packets, leading to a weak combined signal that the method cannot effectively detect.

3 Methodology

This section will first explore why detecting hidden temporal structures in the data is a promising approach to addressing the challenge of fully randomized packet fields. The first subsection will explain how the data used for clustering was collected. The second subsection will detail how the data was aggregated and which attributes were selected for training. The third subsection will present the clustering methods considered and their respective advantages in the context of this research. The fourth subsection will discuss the optimization of the hyperparameters for the methods described in the previous section.

High-speed port scanners operate by sending TCP SYN packets to numerous target IP addresses and ports, recording responses to identify active hosts and open ports. Unlike traditional scanners, which maintain detailed records of each probe to avoid mistaking backscatter from Distributed Denial-of-Service (DDoS) attacks for legitimate responses, modern high-speed scanners like Masscan and ZMap streamline this process. They achieve this by embedding a unique identifier, such as a 32-bit initial sequence number (ISN), into each outgoing packet. This method allows them to differentiate genuine responses from backscatter based on the ISN in the return packets, minimizing the need for extensive tracking and enabling the scanning of the entire IPv4 address space in

less than an hour [8] [9].

However, by doing this, some of these scanners create fingerprints within their packets which can be used to identify the tools used to generate them and, as a result, they can be grouped into scanning groups [1]. Notably, ZMap does not exhibit this behavior, making it an important exception. Advanced adversaries that have access to large amounts of computational resources could forfeit this strategy in favour of being harder to detect by randomizing all packet fields. ZMap achieves this by using Advanced Encryption Standard (AES), effectively thwarting fingerprint-based detection methods such as the one presented in Griffioen et al. [1].

As such, instead of utilizing the data in the packets themselves to detect collaborative scanners, this paper proposes using the temporal correlations of the packets, under the assumption that distributed instances running the same scanning tool would do so at similar times and with similar frequencies. It must be mentioned that, just like with the packet fields, timings and frequencies can also be randomized in the interest of stealthiness, but comes at the cost of efficiency, because it would lead to the under-utilization of the scanning capacity of the infrastructure, as the source hosts would have to stay inactive for intermittent periods of time [1].

3.1 Data Collection

The data used in this research originates from the TU Delft network telescope. A network telescope refers to a designated block of IP address space where legitimate traffic is minimal or nonexistent. By observing the unexpected traffic that appears in this space, it is possible to monitor and analyze remote network security events, including denial-of-service attacks, Internet worm propagation, and network scanning activities [10].

The telescope used contains three partially populated /16 networks which collect incoming packets for approximately 65,000 unused IP addresses [2], though this amount varies based on the university's network strain at different moments of the week.

The dataset utilized for clustering is made up of TCP packets which were collected throughout February 2024. UDP packets are excluded from the analysis because most port scanning traffic targets TCP [1]. Packets which have the IPID equal to 54321 are also excluded from the analysis because these are ZMap packets (mainly originating from benign sources) which can be simply dropped using a firewall rule. The presence of these packets is well-documented, and their exclusion helps ensure that the analysis focuses on potentially more relevant and less predictable network traffic.

3.2 Aggregating Data and Selecting Attributes for Training

In order to train clustering models, this paper proposes the aggregation of packet data into scanning sessions. A scanning session is defined as an accumulation of multiple packets received from the same IP address, with no more than 3 hours between consecutive packets. This timeframe is inspired by Aniket et al. [11] and has been extended to better suit the smaller scale of the TU Delft network telescope. For each scanning session, the following attributes are created:

- The starting time of the session, obtained from the first packet encountered within each session
- The ending time of the session, obtained from the last packet encountered within each session, meaning that there is a break of more than 3 hours between this packet and the next one received from the same IP address
- The total time between the starting time and the ending time of the session
- The mean amount of time between the packets making up the session
- The number of packets received within the time frame of the scanning session

Therefore, the packet dataset collected by the network telescope during February 2024 is aggregated into a scanning session dataset as presented above, with resulting dataset containing almost 4.4 million entries. The newly created attributes can be used to cluster these sessions into groups, and these groups can then be evaluated to understand how likely they are to be collaboratively scanning.

3.3 Considered Clustering Methods

The clustering methods considered in this research are: K-Means Clustering [12], Hierarchical Clustering [13], DBSCAN [14], CURE [15] and BFR [16].

Unlike the other three methods, which typically assume that the entire dataset can be loaded into memory, CURE and BFR are specifically tailored to work efficiently with massive datasets by using disk-based storage and scalable processing techniques. CURE achieves this by representing clusters with multiple representative points, reducing the effect of outliers, and allowing the algorithm to manage data in chunks that can be processed independently [15]. BFR, on the other hand, is an adaptation of the K-Means algorithm designed to work in a high-dimensional space and incorporates a framework to handle data that is too large to fit into memory by partitioning the dataset and progressively processing each partition. [16]

In contrast to CURE and BFR, DBSCAN approaches the challenge of large datasets from a different angle, as its ability to handle large datasets stems from its reliance on spatial indexing structures such as k-d trees or R-trees, which significantly enhance the efficiency of neighborhood queries, a critical feature of the algorithm [14].

Additionally, CURE, BFR, and DBSCAN determine the number of clusters based on the data itself, making them particularly useful for this research since the actual number of collaborative scanners is unknown. These design considerations make CURE, BFR, and DBSCAN stand out as top candidates for the clustering approach, which is why it is expected that they will perform better in comparison to K-Means and Hierarchical Clustering.

3.4 Hyperparameter Optimization

The first step in the process of optimizing the hyperparameters of the aforementioned clustering methods is splitting the data into a training set, a test set and a validation set. From the dataset spanning February 2024, the first 50% of the data will be used to train the models with various hyperparameters

picked through an iterative process, with the remaining data being split into a 25% test set and a 25% validation set. The hyperparameters that provide the best cluster quality based on the degrees of certainty presented in Section 4.1 will subsequently be used to evaluate the quality of the models when clustering the previously unseen test set scanning sessions.

4 Experimental Setup and Results

This section will discuss the way in which the evaluation of the detected clusters will take place and the final results obtained. The first subsection will define the degrees of certainty with which we can claim that a cluster is collaboratively scanning. The second subsection will explain the evaluation of the clustering methods based on the degrees of certainty. The third subsection will compare the results of the evaluations conducted on the previously mentioned clustering methods using their optimized hyperparameters. The fourth subsection will present an in-depth analysis of the clusters to which no degree of certainty can be assigned, in an effort to explain how they appear, and identify whether they are indeed just residual noise. The fifth subsection will present the answers formulated to the research sub-questions.

4.1 Degrees of Certainty

In order to assess how likely it is for a cluster to be a collaboratively scanning group, we define the following five degrees of certainty:

- Degree 1: All IP addresses within the cluster are part of the same subnet
- Degree 2: All IP addresses within the cluster correspond to the same ISP
- Degree 3: All IP addresses within the cluster correspond to the same country
- Degree 4: The IP addresses within the cluster correspond to at most three different subnets in total
- Degree 5: The IP addresses within the cluster correspond to at most three different ISPs in total

These degrees describe in descending order how certainly a cluster can be classified as a collaboratively scanning group (i.e. 1 is the highest degree of certainty, 5 is the lowest). If all IP addresses within a cluster are part of the same subnet, then it can be claimed with high confidence that they are collaborating, especially since the IP addresses themselves are not used during the training of the clustering models. However, if the IP addresses within the cluster correspond to at most three different ISPs in total, it may still be claimed that they are collaborating (in this case, the potential adversary would have made an additional effort to remain hidden by using hosts belonging to different ISPs), but with considerably less certainty.

4.2 Method Validation

The results of training the previously mentioned clustering methods on the training set using various hyperparameters can be seen below. These hyperparameters were tested in an iterative manner, where each iteration involved adjusting one or more parameters to observe their effects on clustering performance. Unfavourable parameter changes, characterized by a decline in clustering performance, indicated that such directions should not be pursued further.

If an extended amount of computational time did not yield considerably better results, this was interpreted as an indication that the hyperparameters were converging to their optimal values. Specifically, when further adjustments to hyperparameters resulted in negligible or no significant improvement in clustering quality, it suggested that the current set of hyperparameters was close to optimal. Thus, through this systematic approach, the optimal hyperparameters for each clustering method were determined, ensuring robust and reliable clustering performance.

DBSCAN

In the case of DBSCAN, the hyperparameters to be fitted are `epsilon`, which is the maximum distance between two samples for one to be considered as being in the neighborhood of the other, and `min_pts`, which is the minimum number of samples in a neighborhood for a point to be considered as a core point. Seeing as a collaboratively scanning group can be made up of as little as 2 IP addresses, `min_pts` has been set to 2, leaving only `epsilon` to be optimized. The values tested and the results they yield can be seen in Table 1, in the Appendix. To determine the optimal value for `epsilon`, the number of clusters corresponding to Degree 1 (the highest degree of certainty for a cluster being a collaboratively scanning group) was calculated for each option. The results indicate that the maximum number of Degree 1 clusters is achieved when `epsilon` is set to 0.001. Applying DBSCAN with these hyperparameters to the test dataset produces the results seen in Figure 1.

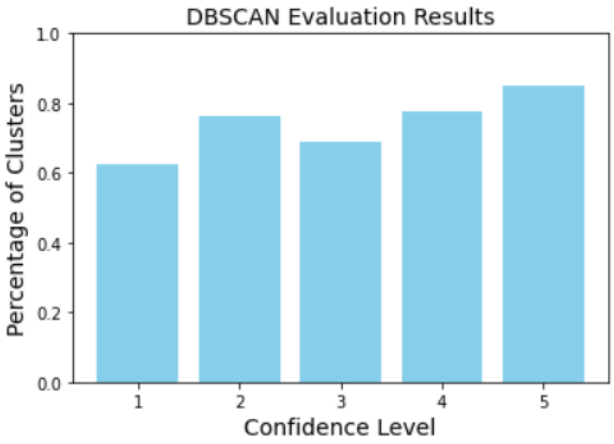

Figure 1: Results of running DBSCAN on the test dataset using `min_pts` = 2 and `epsilon` = 0.0001

K-Means Clustering

In the case of Mini-Batch K-Means, the hyperparameters to be fitted are the number of clusters `no_of_clusters`, which determines how many centroids will be initialized, the batch size `batch_size`, which specifies the number of samples to be used in each iteration to update the centroids, and the maximum number of iterations `max_iterations`.

The values tested and the results they yield can be seen in Table 2 in the Appendix. The results indicate that the maximum number of Degree 1 clusters is achieved when `no_of_clusters` is set to 700000, `batch_size` is set to 10000 and `max_iterations` is set to 100. Applying Mini-Batch K-Means using these hyperparameters to the test dataset produces the results seen in Figure 2.

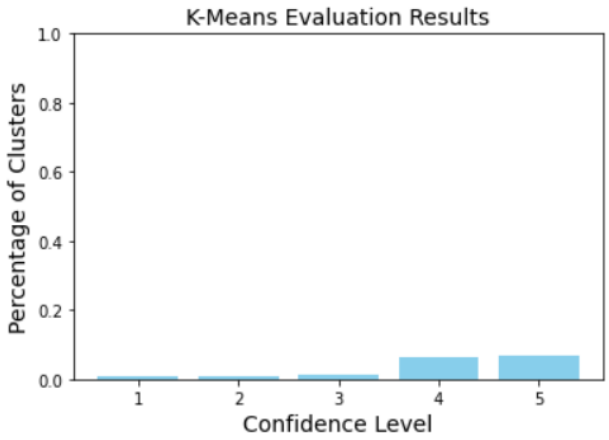

Figure 2: Results of running Mini-Batch K-Means Clustering on the test dataset using `no_of_clusters` = 70000, `batch_size` = 10000 and `max_iterations` = 100

CURE

In the case of CURE, the hyperparameters that must be tuned are the number of clusters `no_of_clusters`, the compression factor `compression`, and the number of representative points `no_repr_points`. Since no pre-existing implementation of CURE was found that doesn't rely on computing a distance matrix (which becomes too large to store when processing large amounts of data), sampling was used to approximate the results of applying the algorithm on the aforementioned datasets. `no_of_clusters` represents the desired number of clusters in the dataset (the values tested for this hyperparameter were inspired from the number of clusters obtained using DBSCAN, adjusted to the sizes of the samples used), while `compression` controls how much the representative points are shrunk towards the centroid of the cluster during the merging process. `no_repr_points` is the number of points used to represent each cluster. The values tested and the results they yield can be seen in Table 3, in the Appendix. The results indicate that the maximum number of Degree 1 clusters is achieved when `no_of_clusters` is set to 200, `no_repr_points` is set to 5 and `compression` is set to 0.01. Applying CURE using these hyperparameters to the test dataset produces the results seen in Figure 3.

Hierarchical Clustering

In the case of Hierarchical Clustering, the hyperparameter to be fitted is the number of clusters `no_of_clusters`. Ward linkage was chosen as the optimal linkage criterion for this situation because it minimizes the variance within each cluster, leading to more compact and well-separated clusters [13]. This is particularly advantageous when identifying collaborative scanning groups, as it ensures that clusters are as distinct as possible, reducing the likelihood of overlapping clusters that could misrepresent the data. Similarly to the case of CURE, sampling was used due to the fact that Hierarchical Clustering implementations also rely on computing a distance matrix. The values tested and the results they yield can be seen in Table 5 in the Appendix. The results indicate that the maximum number of Degree 1 clusters is achieved when `no_of_clusters` is set to 800. Applying Hierarchical Clustering using this hyperparameter to the test dataset produces the results seen in Figure 4.

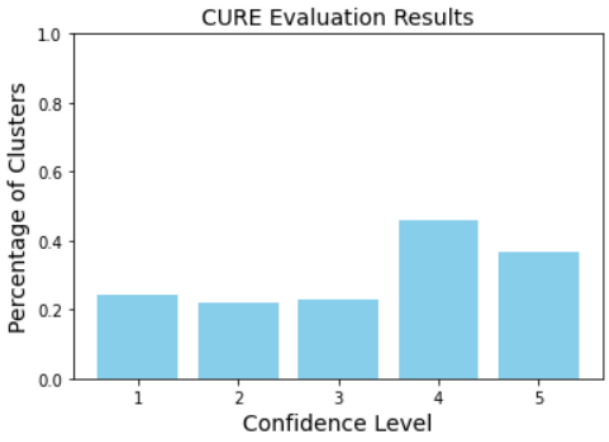

Figure 3: Results of running CURE on the test dataset using `no_of_clusters` = 200, `no_repr_points` = 5 and `compression` = 0.01

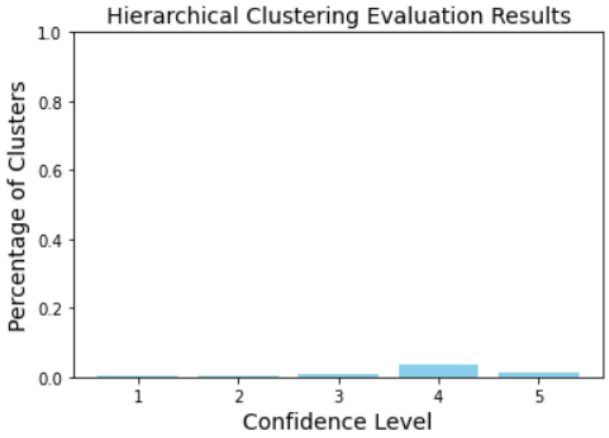

Figure 4: Results of running Hierarchical Clustering on the test dataset using `no_of_clusters` = 800

BFR

In the case of BFR, the hyperparameter to be fitted is the number of clusters `no_of_clusters`, which determines how many centroids will be initialized. Similarly to CURE, no pre-existing implementation of BFR was found that doesn't rely on computing a distance matrix, and as such sampling was used to approximate the results of applying BFR on the train and test datasets. The values tested and the results they yield can be seen in Table 4 in the Appendix. The results indicate that the maximum number of Degree 1 clusters is

achieved when no_of_clusters is set to 900. Applying BFR clustering using this hyperparameter to the test dataset produces the results seen in Figure 5.

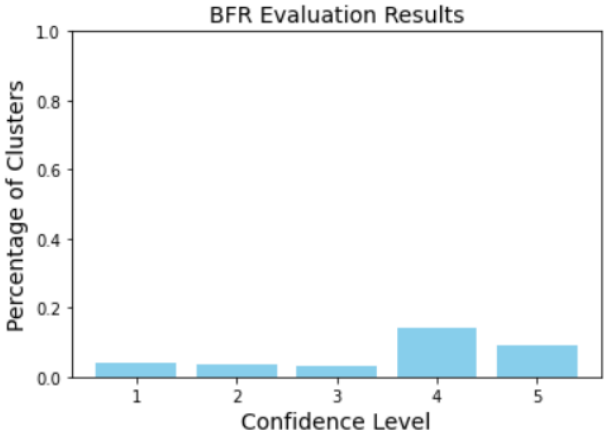

Figure 5: Results of running BFR on the test dataset using no_of_clusters = 900

4.3 Comparison of Clustering Methods

In order to compare the clustering methods, a score is computed for each one by taking the mean of the percentages achieved for every degree of certainty.

DBSCAN significantly outperforms the other four clustering methods in terms of the given results, as it achieves a score of approximately 74%. In comparison, the next best method, CURE, has an approximate score of 30%, meaning DBSCAN performs about 247% better than CURE. When compared to BFR, which has an approximate score of 7%, DBSCAN is about 1057% better. Similarly, DBSCAN outperforms K-Means, with its average score of 3%, by around 2467%. Lastly, compared to Hierarchical Clustering, which has an approximate score of 1%, DBSCAN is roughly 7400% better.

CURE outclasses BFR, K-Means, and Hierarchical Clustering, at it performs significantly better. CURE's score of 30% is about 429% higher than BFR's score of 7%. Compared to K-Means' score of 3%, CURE is roughly 1000% better. Lastly, CURE outperforms Hierarchical Clustering's score of 1% by about 3000%. These comparisons highlight the significant performance gaps between the methods, with DBSCAN and CURE being notably ahead of the rest.

The comparison of the evaluations above highlights the advantages and limitations of each clustering method in the context of collaborative scanner detection. DBSCAN's density-based approach allows it to naturally adapt to the data structure, identifying clusters regardless of their shape or size, and effectively filtering out noise. This is particularly advantageous in network traffic analysis where collaborative scans may not conform to fixed patterns.

CURE's ability to handle outliers and represent clusters with multiple points makes it a strong candidate. BFR's chunk-based processing is useful for large datasets but does not seem compensate for its centroid-based limitations.

K-Means and Hierarchical Clustering, while useful for certain applications, do not perform well in the noisy environment of network traffic. Their limitations in handling noise and large-scale data make them less suitable for detecting collaborative scanners.

4.4 Analysis of Clusters Not Covered Under Any Degree of Certainty

This subsection will more closely analyze the clusters detected by DBSCAN when ran on the test set using the optimal hyperparameters obtained in Section 4.2. These clusters can be split into 3 categories based on the number of source IP addresses they contain:

- Clusters containing less than 100 source IP addresses: 63 clusters in the test set
- Clusters containing more than 100 and less than 1000 source IP addresses: 4 clusters in the test set
- Clusters containing more than 1000 source IP addresses: 5 clusters in the test set

The GreyNoise[1] analysis tool was used to attempt to further investigate these clusters and understand why they do not conform to any degree of certainty. GreyNoise is a cybersecurity company that focuses on analyzing and categorizing internet-wide scan and attack traffic to help organizations differentiate between benign, background noise and targeted malicious activity. The results of this analysis on the larger clusters which were encountered can be seen in Table 6 in the Appendix.

The analysis reveals that the majority of IP addresses in most of these clusters are located in China. This is especially true for the largest clusters, which contain more than 20000 IP addresses.

The existence of some large clusters can be attributed to Internet worms, which are malicious programs that replicate themselves across networks to infect systems without requiring user interaction. This phenomenon is exemplified by cluster 7480. It is made up of 198 hosts, out of which 74% were detected by the GreyNoise analysis tool to be infected with Mirai or a Mirai-like variant of the worm. This would provide a reasonable explanation as to why this cluster is not covered by any degree of certainty: by infecting multiple hosts across various subnets, ISPs, and countries, botnets can avoid being detected when using such an evaluation scheme. The same reasoning can be applied to the other clusters, where large percentages of hosts were identified as being either Secure Shell (SSH) Bruteforcers, Telnet Bruteforcers or Mirai infected. As a result, hosts that are infected by an Internet worm can avoid detection in the above manner.

The conclusion of this analysis is that additional consideration needs to be attributed to the evaluation of the formed clusters, so that clusters made up of hosts infected by Internet worms won't be discarded as simply being badly formed.

[1]The GreyNoise analysis tool can be found here: https://www.greynoise.io/

4.5 Results

RQ1: *"How do collaborative scans work and what assumptions does the proposed methodology make?"*
Collaborative scans work by sending packets from a multitude of IP addresses to remain under IDS and firewall thresholds. The assumption this paper makes about collaborative scans is that they commence and conclude at similar times and that the addresses within a group send messages with approximately the same frequency. Therefore, no assumptions are made about the contents of the packets themselves, enabling the proposed method of detection to work regardless of the tool used to create these packets, as long as the above-mentioned temporal correlations hold.
RQ2: *"What data attributes should be considered for clustering?"*
This paper proposes aggregating the received packets into scanning sessions, as detailed in Section 3.2. After this aggregation, the following attributes should be used for clustering:

1. The start time of the sessions
2. The end time of the sessions
3. The total duration of the sessions
4. The average time between packets within each session
5. The total number of packets received during each session

These attributes facilitate the effective clustering of the sessions.
RQ3: *"Once a cluster has been identified, how can we check that it is indeed a collaborative scanning group?"*
There is no method by which to say with full confidence that an identified group is collaboratively scanning. This paper proposes approximating the likelihood that a cluster is collaboratively scanning based on the best degree of certainty that it achieves. If, for example, it achieves Degree 1, then there is a high chance that the cluster is indeed a collaborative scanning group, but if it can only achieve Degree 5, then this probability is considerably lower.
RQ4: *"What clustering methods should be used?"*
Based on the evaluations above, DBSCAN stands out as the optimal clustering method by a considerable margin, as it performs about 247% better than CURE, 1057% better than BFR, 2467% better than K-Means and 7400% better than Hierarchical Clustering.
RQ5: *"What values should the hyperparameters have?"*
Based on the evaluations above, the hyperparameters for DBSCAN should have the following values: `min_pts` = 2 and `epsilon` = 0.0001. In order to further validate this outcome, the results of running DBSCAN using these hyperparameters on an unseen validation dataset can be observed in Figure 6.
RQ6: *"If multiple clustering approaches can detect collaborative scanning groups, how do their performances compare?"*
Both DBSCAN and CURE perform considerably better than the other three clustering methods based on the evaluations undertaken using the five degrees of certainty, but it is clear from the results presented above and in the Appendix that

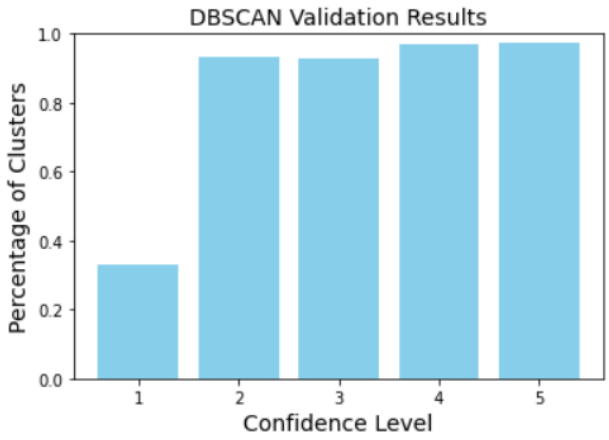

Figure 6: Results of running DBSCAN on the validation dataset using `min_pts` = 2 and `epsilon` = 0.0001

DBSCAN considerably outperforms CURE while also detecting a larger number of meaningful clusters (meaning a cluster that scans more than 10% of the network telescope).

5 Responsible Research

This section will discuss the ethical implications of the research, the privacy concerns regarding the data collection method, and the reproducibility of the research methods presented previously, followed by some additional concerns about the environmental impact of using machine leaning models for collaborative scanner detection.

This research can help enhance cybersecurity by identifying and mitigating coordinated malicious activities, thus protecting sensitive information, and maintaining system integrity. There exists the question of the risk of false positives, where legitimate users may be incorrectly flagged as malicious, leading to unwarranted scrutiny or actions against them. Since the method outlined above only considers packets that have the SYN flag set and the ACK flag unset, internet backscatter isn't included in the sanitized dataset used for training. Users could still have their IP addresses spoofed and be mistakenly flagged, although from the perspective of Internet scanning it makes no sense for malicious parties to engage in this behaviour, since receiving the responses from the scanned hosts is a vital part of Internet scanning and these responses would be sent to the legitimate users instead of the scanning parties if this method is employed. Another situation in which incorrect flagging could happen is in the case of benign scanning. However, scientific institutions or groups that undertake scanning for research-related purposes usually announce their presence, and as such they can be excluded from any subsequent actions following their initial flagging by the clustering methods.

In terms of the privacy concerns that could be raised regarding the data collection procedure, data is collected by the network telescope and filtered such that the considered packets have the SYN flag set and the ACK flag unset. This is done to remove backscatter containing information about hosts being under DDoS attack. The resulting data is inher-

ently anonymous because it is generated by the passive observation of an unused IP address space where no legitimate traffic exists [10]. Therefore, any data gathered in this way consists of only unsolicited traffic.

The research methods explained previously have been meticulously detailed and presented with full transparency in order to ensure the reproducibility of the work. All of the results presented have been verified multiple times to guarantee that they are accurate and reliable.

Another important aspect to consider is the sustainability of implementing machine learning models for detecting collaborative scanners. These models require significant computational power, which may lead to increased energy consumption and, as a result, negative environmental impact. The use of these models has the potential to improve cybersecurity by identifying and addressing threats. However, it is essential to evaluate the long-term sustainability of these methods, especially as data volumes and model complexities grow. Efficient and scalable algorithms are necessary to ensure high performance without an excessive environmental footprint.

6 Discussion

The results of this study highlight the efficacy of the DBSCAN clustering algorithm in detecting collaborative scanning activities. By leveraging temporal correlations between packet transmissions, DBSCAN demonstrates superior performance in identifying meaningful clusters compared to other methods such as CURE, BFR, K-Means, and Hierarchical Clustering.

One significant implication of this research is its potential to enhance network security by providing a robust method for detecting distributed scanning activities. Collaborative scanning poses a substantial threat as it allows adversaries to map network vulnerabilities without triggering conventional security alerts. By accurately clustering these scanning activities, network administrators can proactively identify and mitigate potential threats.

However, the study also emphasizes the importance of accounting for Internet worms when assessing the produced clusters. Many clusters may not fit within any of the five degrees of certainty, but they should not be dismissed outright.

7 Conclusion

In conclusion, this study successfully demonstrates the effectiveness of DBSCAN in identifying collaborative scanning activities by focusing on temporal patterns in packet transmissions. The research answers the primary question of how collaborative scans can be detected using clustering methods, with DBSCAN emerging as the most effective technique. This method does not rely on the content of the packets, allowing it to adapt across various tools and scenarios, provided that the temporal correlations are maintained.

The findings recommend DBSCAN for its superior performance in clustering compared to other methods, with specific hyperparameters (`min_pts` = 2 and `epsilon` = 0.0001) showing optimal results.

Future research could further test the performance of DBSCAN by injecting artificial collaborative scanner packets into the network telescope dataset and computing how many of these scanners are detected using this clustering method. Creating these packets should be done using a wide array of scanning tools in order to obtain the most realistic evaluation possible.

Additionally, future research should focus on developing energy-efficient algorithms that offer high performance with sustainability principles in mind.

References

[1] H. Griffioen and C. Doerr, "Discovering collaboration: Unveiling slow, distributed scanners based on common header field patterns," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–9. DOI: 10 . 1109 / NOMS47738.2020.9110444.

[2] N. Blenn, V. Ghiëtte, and C. Doerr, "Quantifying the spectrum of denial-of-service attacks through internet backscatter," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ser. ARES '17, Reggio Calabria, Italy: Association for Computing Machinery, 2017, ISBN: 9781450352574. DOI: 10.1145/3098954.3098985. [Online]. Available: https://doi.org/10.1145/3098954.3098985.

[3] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescapé, "Analysis of a "/0" stealth scan from a botnet," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 341–354, Apr. 2015, ISSN: 1063-6692. DOI: 10 . 1109 / TNET.2013.2297678. [Online]. Available: https://doi.org/10.1109/TNET.2013.2297678.

[4] C. Gates, "Co-ordinated port scans: A model, a detector and an evaluation methodology," AAINR16682, Ph.D. dissertation, CAN, 2006, ISBN: 9780494166826.

[5] M. Garey, D. Johnson, and L. Stockmeyer, "Some simplified np-complete graph problems," *Theoretical Computer Science*, vol. 1, no. 3, pp. 237–267, 1976, ISSN: 0304-3975. DOI: https : / / doi . org / 10 . 1016 / 0304 - 3975(76) 90059 - 1. [Online]. Available: https : / / www . sciencedirect . com / science / article / pii / 0304397576900591.

[6] S. Robertson, E. V. Siegel, M. Miller, and S. Stolfo, "Surveillance detection in high bandwidth environments," *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 1, 130–138 vol.1, 2003. [Online]. Available: https://api.semanticscholar.org/CorpusID:213945.

[7] V. Yegneswaran, P. Barford, and J. Ullrich, "Internet intrusions: Global characteristics and prevalence," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 138–147, Jun. 2003, ISSN: 0163-5999. DOI: 10 . 1145/885651.781045. [Online]. Available: https://doi.org/10.1145/885651.781045.

[8] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman, "Zippier zmap: Internet-wide scanning at 10 gbps," in *Proceedings of the 8th USENIX Conference on Offensive Technologies*, ser. WOOT'14, San Diego, CA: USENIX Association, 2014, p. 8.

[9] R. D. Graham, "Masscan: Mass ip port scanner," 2014. [Online]. Available: https : / / github . com / robertdavidgraham/masscan.

[10] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, "Network telescopes: Technical report," 2004. [Online]. Available: https : / / api . semanticscholar . org / CorpusID:18094494.

[11] A. Anand, M. Kallitsis, J. Sippe, and A. Dainotti, "Aggressive internet-wide scanners: Network impact and longitudinal characterization," in *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT 2023, New York, NY, USA: Association for Computing Machinery, 2023, pp. 1–8, ISBN: 9798400704079. DOI: 10.1145/3624354.3630583. [Online]. Available: https://doi.org/10.1145/3624354.3630583.

[12] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10, Raleigh, North Carolina, USA: Association for Computing Machinery, 2010, pp. 1177–1178, ISBN: 9781605587998. DOI: 10.1145 / 1772690 . 1772862. [Online]. Available: https : //doi.org/10.1145/1772690.1772862.

[13] B. P. C. "Unveiling hidden patterns: An introduction to hierarchical clustering." (2023), [Online]. Available: https : / / www . kdnuggets . com / unveiling - hidden - patterns - an - introduction - to - hierarchical - clustering.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Knowledge Discovery and Data Mining*, 1996. [Online]. Available: https://api.semanticscholar.org/CorpusID:355163.

[15] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," *SIGMOD Rec.*, vol. 27, no. 2, pp. 73–84, Jun. 1998, ISSN: 0163-5808. DOI: 10.1145/276305.276312. [Online]. Available: https://doi.org/10.1145/276305.276312.

[16] P. S. Bradley, U. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, ser. KDD'98, New York, NY: AAAI Press, 1998, pp. 9–15.

Appendix

epsilon	10^{-6}	5^{-6}	10^{-5}	5^{-5}	10^{-4}	5^{-4}	10^{-3}
Noise Points	1924832	1523003	1395843	913590	759666	412713	240682
Meaningful Clusters	54	160	333	562	720	1030	1784
Degree 1	37.04%	59.38%	59.46%	57.17%	55.14%	18.25%	9.30%
Degree 2	90.74%	83.75%	71.72%	64.41%	61.67%	25.40%	12.78%
Degree 3	27.78%	72.50%	69.37%	66.73%	66.25%	27.18%	14.35%
Degree 4	98.15%	96.88%	92.19%	88.08%	85.28%	72.23%	62.95%
Degree 5	98.15%	97.50%	92.49%	89.50%	86.67%	75.08%	66.03%

Table 1: DBSCAN results generated from clustering on the training set using varying values for `epsilon`.

no_of_clusters	100000	100000	200000	300000	300000	300000	400000	450000	500000	700000
batch_size	10000	20000	10000	10000	20000	20000	10000	10000	10000	10000
max_iterations	100	100	100	100	100	150	100	100	100	100
Meaningful Clusters	8152	8077	8862	8793	8714	8691	8321	7910	7596	6075
Degree 1	2.07%	2.20%	3.74%	5.45%	6.01%	5.79%	8.28%	9.47%	10.90%	17.19%
Degree 2	2.26%	2.30%	4.27%	5.98%	6.86%	6.36%	9.28%	10.87%	12.32%	19.54%
Degree 3	3.40%	3.52%	5.98%	8.45%	9.10%	8.56%	11.92%	13.68%	15.19%	23.37%
Degree 4	7.23%	6.43%	22.78%	32.79%	34.52%	32.02%	45.57%	48.94%	53.78%	72.72%
Degree 5	7.97%	7.21%	24.80%	35.06%	36.98%	34.78%	47.55%	51.08%	55.40%	74.77%

Table 2: Mini-Batching K-Means results generated from clustering on the training set using varying values for the `no_of_clusters`, `batch_size` and `max_ierations`.

no_of_clusters	100	200	300	200	300	200	200	200	200	200
no_repr_points	2	2	2	2	5	5	5	5	5	5
compression	0.5	0.5	0.5	0.2	0.5	0.5	0.1	0.01	0.001	0.0001
Meaningful Clusters	53.6	51.6	128.8	97.2	122.8	91.6	96	86.4	79.6	84.8
Degree 1	3.72%	7.16%	7.02%	12.87%	8.33%	9.45%	9.01%	14.85%	10.65%	14.10%
Degree 2	3.36%	4.83%	6.39%	12.67%	7.34%	8.99%	8.05%	14.39%	9.87%	13.86%
Degree 3	3.36%	5.88%	7.95%	13.58%	8.83%	9.45%	9.63%	14.88%	11.12%	14.35%
Degree 4	12.31%	19.88%	23.29%	26.76%	25.80%	23.40%	22.79%	24.83%	23.11%	23.06%
Degree 5	6.30%	12.57%	14.29%	20.89%	16.80%	16.88%	16.05%	20.20%	17.13%	19.06%

Table 3: CURE results generated from clustering on samples of the training set using varying values for `no_of_clusters`, `no_repr_points` and `compression`.

no_of_clusters	100	50	200	300	400	500	700	800	900	1000	1100
Meaningful Clusters	66.6	40.8	109.4	144.4	174	196.8	241	260.6	275.6	292.2	304.8
Degree 1	0.63%	0%	1.68%	0.86%	1.74%	2.21%	2.50%	2.53%	2.76%	2.13%	2.31%
Degree 2	0.63%	0%	1.52%	0.57%	1.10%	1.79%	1.74%	1.94%	2.13%	1.94%	1.70%
Degree 3	0.63%	0%	1.35%	0.98%	1.44%	2.30%	1.98%	2.25%	2.70%	2.42%	2.09%
Degree 4	40.63%	0.47%	3.72%	4.11%	6.19%	6.79%	7.79%	8.78%	10.04%	10.32%	12.36%
Degree 5	0.94%	0%	2.83%	2.56%	4.25%	4.34%	4.79%	5.06%	5.65%	5.57%	6.15%

Table 4: BFR results generated from clustering on samples of the training set using varying values for `no_of_clusters`.

no_of_clusters	100	200	300	400	500	600	700	800	900
Meaningful Clusters	64.2	104.6	142.6	176.8	206	232.2	258.6	280.6	303.2
Degree 1	0%	0%	0%	0.21%	0.18%	0.33%	0.44%	0.47%	0.31%
Degree 2	0%	0%	0%	0.21%	0.18%	0.33%	0.30%	0.34%	0.31%
Degree 3	0%	0%	0%	0.21%	0.18%	0.33%	0.44%	0.47%	0.31%
Degree 4	0%	0.18%	0.54%	0.97%	1.77%	2.11%	3.32%	3.89%	3.80%
Degree 5	0%	0.18%	0.13%	0.32%	0.37%	0.41%	0.59%	0.61%	0.70%

Table 5: Hierarchical Clustering results generated from clustering on samples of the training set using varying values for `no_of_clusters`.

Cluster ID	IPs	Malicious	Benign	Unknown	Notes
2	>20000	27%	0%	73%	78% CN, 15% SSH, 6% Telnet, 4% Mirai
1	>20000	29%	2%	69%	77% CN, 18% SSH, 8% Telnet, 4% Mirai
5	>20000	26%	5%	69%	77% CN, 14% SSH, 5% Telnet, 4% Mirai
103	19906	32%	1%	67%	72% Web Crawler, 20% SSH, TLS/SSL Crawler
7504	68	99%	0%	1%	94% SSH, 76% HK
7480	198	86%	0%	14%	74% Mirai
7673	39	100%	0%	0%	95% Unidentified
7667	20	0%	0%	0%	100% Unidentified
13764	4	0%	0%	0%	100% Unidentified
9277	25	74%	0%	26%	84% Web Crawler, 74% CN
78	98	51%	3%	46%	71% Web Crawler
28309	12	0%	0%	0%	100% Unidentified
9277	25	74%	0%	26%	84% Web Crawler
78	98	51%	3%	46%	71% Web Crawler, 26% SSH, 10% Mirai, 56% CN
28309	12	0%	0%	0%	100% Unidentified
15214	24	59%	0%	41%	88% CN, 12% Mirai
609	53	35%	4%	61%	26% SSH
13754	5	75%	0%	25%	75% Mirai
1509	43	45%	0%	55%	55% CN, 27% Telnet
8595	19	88%	0%	13%	87.5% UCLOUD HK
7857	51	47%	0%	53%	47% CN, 38% Telnet
2043	41	67%	0%	33%	76% Web Crawler, 45% SSH, 42% CN
8113	22	43%	0%	57%	71% TLS/SSL Crawler, 29% Mirai
5221	23	100%	0%	0%	88% Web Crawler, 69% SSH
8123	5	100%	0%	0%	33% Mirai
458	54	53%	0%	47%	47% CN, 43% SSH
160	371	45%	4%	51%	69% CN, 22% SSH, 14% Mirai, 13% Telnet
19099	16	40%	0%	60%	40% CN, 20% SSH, 20% Telnet, 20% Mirai
708	495	54%	1%	46%	68% CN, 27% SSH, 19% Mirai, 16% Telnet
3233	19	93%	0%	7%	86% Web Crawler
6809	23	25%	0%	75%	58% Web Crawler
28	1249	59%	0%	41%	97% CN, 47% SSH, 13% Telnet

Table 6: Results of the analysis undertaken using the GreyNoise tool on the larger clusters which aren't covered under any degree of certainty. SSH refers to SSH Bruteforcers, Telnet refers Telnet Bruteforcers and Mirai refer to Mirai Botnet infected hosts. CN refers to China and HK refers to Hong Kong.