Received: 12 November 2018 | Revised: 2 January 2020 | Accepted: 3 January 2020
DOI: 10.1002/cpe.5679

WILEY

RESEARCH ARTICLE

Mitigating DDoS using weight-based geographical clustering

Madeleine Kongshavn[1] | **Hårek Haugerud**[2] | **Anis Yazidi**[2] | **Torleiv Maseng**[3] | **Hugo Hammer**[2]

[1]Department of Information and Communication Technology, University of Agder, Grimstad, Norway
[2]Department of Computer Science, OsloMet—Oslo Metropolitan University, Oslo, Norway
[3]Department of Technology Systems, University of Oslo, Oslo, Norway

Correspondence
Anis Yazidi, Oslo Metropolitan University, PO Box 4, St. Olavs plass, N-0130 Oslo, Norway.
Email: anis.yazidi@oslomet.no

Summary
Distributed denial of service (DDoS) attacks have for the last two decades been among the greatest threats facing the internet infrastructure. Mitigating DDoS attacks is a particularly challenging task as an attacker tries to conceal a huge amount of traffic inside a legitimate traffic flow. This article proposes to use data mining approaches to find unique hidden data structures which are able to characterize the normal traffic flow. This will serve as a mean for filtering illegitimate traffic under DDoS attacks. In this endeavor, we devise three algorithms built on previously uncharted areas within mitigation techniques where clustering techniques are used to create geographical clusters in regions which are likely to contain legitimate traffic. We argue through extensive experimental results that establishing clusters around this narrative is a superior solution to clustering algorithms which rely on bitwise distances between IP addresses. In addition, the DDoS filtering algorithm is deployed in a virtual Linux environment using Nfqueue and tested in a simulated real-life DDoS attack.

KEYWORDS
Address Clustering, Anomaly Intrusion Detection, Clustering Techniques, Geographical IP, Mitigation Techniques, Mitigating DDoS Attacks

1 | INTRODUCTION

A denial of service (DoS)[1,2] attack can be described as an explicit attempt to render a server or network incapable of providing normal service to its users. DoS attacks often rely on continuously and excessively consuming limited resources such as bandwidth, memory, storage, and CPU, which are vital to the well-functioning of the targeted service.[3] We can differentiate between DoS attacks and distributed DoS (DDoS) attacks; DoS is where one attacker, with a single network connection, executes an attack to exhaust the resources of a server. On the other hand, a DDoS attack adds a many-to-one dimension to the DoS problem. Instead of using one connection, a DDoS attack often uses thousands of compromised hosts to execute an attack, amplifying both the available attacker resources and the complexity of the attack.[4]

DDoS attacks have for the last two decades been among the greatest threats facing the Internet infrastructure. Attacks have become commonplace with a wide range of global victims, from commercial websites and educational institutions to public chat servers and government organizations.[5] Attacks have grown in complexity over the last couple of years as criminals constantly launch more complicated attack patterns, containing a multitude of vectors, tailored to specific victims. In October 2016, one of the largest DDoS attacks was witnessed and had as a target the major domain name system (DNS) provider, DYN. The attack was accomplished by sending large numbers of seemingly legitimate DNS requests. At its peak, the DDoS attack accomplished an estimated traffic rate of 1.2 Tbps.

In the endeavor to propose efficient solutions to mitigate DDoS attacks, there has been a variety of approaches.[6] More recently attention has been brought to solutions specific to cloud[7-9] and software-defined networks.[10,11] An important part of the research has been focused on anomaly

Correction added on 22 February 2020, after first online publication: Summary and keywords have been changed.
A very preliminary version of this article was presented in Reference 1.

intrusion detection techniques.[12-14] Anomaly-based detection techniques build on the principle that the traffic distribution of a service will deviate from its normal pattern under an attack. These techniques analyze traffic based on a model the system defines as normal before proceeding to the detection phase. In the detection phase, the system makes decisions on incoming traffic by comparing it against the normal perceived behaviour. Peng et al[15] proposed a mechanism termed history-based IP filtering (HIF) for the edge router to admit the incoming packets according to a prebuilt IP address database. This was the first approach in this direction and since then there have been several attempts to improve on this method.[16,17]

From a machine learning perspective, anomaly-based detection techniques can be categorized into the broad categories of classification-based and clustering-based techniques. Classification-based techniques fall under the class of supervised machine learning techniques where the system learns the normal pattern distribution by looking at labeled data and using predefined classes. The system should then, based on knowledge of previous known data, be able to classify new and unknown data.

Clustering-based techniques do not require labeled data to learn the pattern distribution. The goal of clustering is to identify some unknown data patterns based on the hypothesis that some features exhibit similarity. Clustering can be seen as the task of transforming a finite set of unlabeled data into a finite set of naturally hidden data structures.[18] Given a dataset D, the goal of clustering is then formally to divide the elements into clusters $C_1, C_2, \dots, C_N$. Elements assigned to a cluster are similar among themselves but dissimilar among elements in different clusters.[19,20] To group together similar objects, a distance measurement between the considered attributes is required. The most used distance measurement in this context is the Euclidean distance which in our case simply reduces to the distance between two IP addresses, that is, the absolute value of the arithmetic difference of the two numbers. Given two 32-bit IP addresses IP_i and IP_j, the Euclidean distance between them is formally defined as:

$$\triangle\,\mathrm{eucl}(\mathrm{IP}_i, \mathrm{IP}_j) = |\mathrm{IP}_i - \mathrm{IP}_j|.$$

A different measurement used by Goldstein et al[16] is to employ a matching prefix between addresses, specifically 24 bit-prefixes. According to this measure, the distance between two IP addresses IP_i and IP_j given a prefix p is defined as 1 if the IP addresses share the same prefix and infinite otherwise.

Pack et al proposed Xor to measure the distance between addresses.[17,21] Given two IP addresses IP_i and IP_j, the distance is given by applying the Xor operator on the addresses and keeping the highest order bit while setting all the lower bits to 0. The distance ranges from 2^0 to 2^{31}. Formally this distance measurement can be defined as:

$$\triangle\,\mathrm{Xor}(\mathrm{IP}_i, \mathrm{IP}_j) = 2^{\lfloor \log_2(\mathrm{IP}_i \oplus \mathrm{IP}_j) \rfloor}.$$

Clustering approaches within anomaly intrusion detection rely on IP and TCP header fields to model the normal traffic behaviour and create meaningful clusters.[22] Advancements within this field have been accomplished by using different clustering mechanisms or by trying to select relevant attributes to base the clustering on. Clustering approaches which employ IP addresses take advantage of the IP neighborhood relationship between networks to cluster addresses together. If an IP address appears in address space A, the assumption is that this address is close to an IP address which lies in the address space A' which shares the same prefix as A and thus their bitwise distance is short. In our opinion the use of distance measurements based on bitwise similarity is not enough to represent the structure of the traffic pattern as addresses which lay close in the address space does not necessarily lay close in the geographical space.

In this article, a new clustering method which uses clustering techniques based on geographical regions to cluster data is proposed. These clustering methods build on the following hypothesis: *if network x at location y reaches the server, there is a higher chance for network z from location y' in vicinity of y to reach the server, than network q that does not belong to a location close to network x*. We accomplished this by using databases available online which consist of a set of IP address blocks with known geographical position.[23,24] By establishing a set of clusters $C_1, \dots, C_N$ where each cluster C_i contains a set of geographical points $p(x,y)_1, \dots, p(x,y)_n$ where x represents an x-coordinate and y represents a y-coordinate, we are able to represent the normal traffic pattern of the system. We argue that this clustering approach creates more meaningful patterns that are better equipped for the task of differentiating between legitimate and illegitimate data patterns than approaches which only consider bitwise distance.

The rest of this article is organized as follows: Section 2 describes relevant related work. Three clustering approaches are proposed and discussed in Section 3. Section 4 explains the implementation of these algorithms. Section 5 discusses the results found through data mining and simulation. Finally, we draw our conclusion and delineate future work in Section 6.

2 | RELATED WORK

Clustering is a well-known mitigation technique within anomaly intrusion detection and countless clustering methods have been suggested to differentiate between abnormal and normal traffic flow. Clustering approaches are divided into two phases: first, the normal traffic pattern is modeled by creating clusters, second, this model is used to drop or accept new data. In the following, some relevant works within mitigation approaches regarding clustering techniques based on traffic flow are presented.

K-means clustering approaches aim to divide N observations into K clusters, where each observation belongs to the cluster with the nearest *mean*.[25] One of the more popular k-means algorithms is to use an iterative solution to find the local minimal solution. This algorithm is often called the k-means algorithm or the Lloyd's algorithm.[26] There are several variants of this algorithm. However, Lloyd's algorithm is based on the observation that the optimal position of a *mean* is at the centroid of the associated cluster.[26] The approach begins with choosing different cluster centroids or *means*. When cluster centers have been chosen, the algorithm follows two steps. The first step determines which data belong to which cluster via nearest distance calculation from the different points to the different *means*. In the second step, the position of the cluster centers is then recomputed and moved based on finding the nearest center for all points in a cluster. The k-means algorithm will follow these two steps until convergence is reached.[27]

Even though this technique is able to find the local minimal solution, it is not necessary the global minimal solutions, as the Lloyd's algorithm does not specify the initial starting position of the clustering centers.[26] This is a serious weakness as the iterative technique is sensitive to the initial starting positions of the cluster centers. In other terms, how well the clustering is, heavily depends on where the initial cluster centers are set.[25,28] There is currently no known efficient and widely accepted solutions to this problem. However, in order to obtain optimal clusters or solutions using the k-means algorithm, the algorithm is often ran several times with different starting positions for the cluster centers.[25]

K-means clustering is one of the most used applied techniques within anomaly intrusion detection. Qin et al used an entropy-based approach to identify values which then are used to model normal traffic flow through a k-means clustering technique.[22] Euclidean distance is used to identify the similarity between two entropy vectors and the weighted average radius of clusters is used to identify the number of required clusters. After k-means clustering computation is finished, clusters that contain less than 5% of points are removed. Each cluster identifies the cluster center *c* based on the weighed average of all points in the cluster and the radius *r* which is the maximum distance from *c* to all points in this cluster. New data flow is considered normal if the distance to its closest aligned cluster center *c* is less than the measured *r* for this cluster. The technique is further evaluated on the DARPA dataset and results are obtained in form of a DF rate. DF rate is defined as the detection rate divided by the false positive rate. This technique managed at best to achieve a DF rate of 7.[22]

Goldstein et al proposed to take advantage of the neighbourhood relationship between IP addresses to estimate the likelihood of an address to occur in the traffic flow. Euclidean distance and Xor were used to estimate the distances between IP addresses. A modified k-means algorithm was used to compute cluster centers before an area around the centers was defined as where normal traffic flow is expected to be seen. An algorithm was used to grow this area based on the weighted average of all points seen in a cluster. This weighting gave larger defined areas for clusters with high density and smaller defined areas for smaller and less dense clusters.

K-means clustering has been used widely within mitigation approaches. However, the method is dependent on the placement of initial centroids and on the number of clusters and is susceptible to outliers in the dataset. In essence, singular points which lay far away from the remainder of the dataset points can affect the centroid placement in a negative way. Ranjan and Sahoo proposed a modified k-medoids algorithm to decrease some of the disadvantages accompanied by k-means.[29] The algorithm used a squared Euclidean distance to estimate similarity between objects. The technique was further evaluated using the KDD cup99 dataset and managed to outperform k-means with a higher detection rate and a smaller false negative rate. The research does not focus on handling outliers in the datasets.

Over the last few years, intrusion detection and DDoS mitigation techniques have experienced a shift in focus towards density-based clustering approaches to mitigate attacks. Density-based clustering overcomes some of the shortcomings possessed by earlier clustering techniques, as the clustering methods are not equally susceptible to anomalies in the dataset. The main focus of density-based clustering approaches is to evaluate an object relevance with regard to its local neighbors. Based on this assumption, irrelevant objects which belong to sparse clusters or lie in the outliers of clusters can be removed.

Density-based spatial clustering of applications with noise (DBSCAN) is an algorithm which builds clusters based on the assumption that objects or points are closely linked together. DBSCAN is a popular density-based clustering algorithm which allows clusters to expand in any shape or form. Unlike k-means, which assumes all points in a dataset are legitimate, DBSCAN has a higher resistance against noise as points can be considered outliers in the dataset. The algorithm takes two parameters; an integer minpts or minimum points and distance *d*. If a point Y can reach *minpts* in a radius, based on distance *d*, point Y is considered a *core-point*. If point Y then has a path $p_1, p_2, \ldots, p_{n-1}, p_n$ to a point W, where each point between point Y and point W is a *core-point*, all points p_m on the path needs to be density-reachable to p_{m+1}. Density-reachable means p_m need to have p_{m+1} within distance *d*. These *core-points* on the path $p_1, \ldots, p_n$, including the points that is density-reachable to a *core-point* in this path, is considered as a cluster. Points that are density-reachable to a *core-point*, but is not considered as a *core-point* because it does not contain *minpts* within radius *d* are considered outliers of a cluster. Since DBSCAN builds clusters based on points that are linked together, the algorithm does not need to define the amount of start-clusters. However, since the clustering algorithm needs to define *minpts* and distance *d* this can cause different weaknesses. For example, the problem of the system to cluster the data pattern meaningfully with high differences in densities or problems with the dataset not being well enough understood to choose meaningful parameters.[30,31]

To position our work in the right perspective, we note that there are two main families of DDoS defense techniques depending on where the detection of mitigation of the DDoS attack is performed: source-end DDoS methods and victim-end DDoS methods.[32] Our approach is a victim-end DDoS defense. Simply put, in source-end DDoS, the defense is deployed close to the source of the attacks, in the ISP or the upstream routers. In victim-end DDoS defense, the defense takes place at the victim network. For some of the source-end DDoS approaches the filtering of attack traffic is performed through the help of upstream ISPs. The ISP of the victim requests the upstream ISP of the source to deploy filtering rules along the forwarding paths from the source of the attack to the victim. Those approaches include for instance Pushback,[33] D-WARD,[34] AITF,[35] StopIt,[36] and

Senss.[37] In one of the earliest DDoS mitigation mechanisms, Pushback, collaboration between routers is required. Whenever a congested link is detected, the downstream routers send a message to the upstream routers to throttle traffic. However, these approaches require standardisation as they depend on communication and cooperation between ISPs.[38]

According to a recent report,[39] DDoS mitigation approaches are divided into three main categories:

- packet marking where routers along the path add their identity to the packet;[40,41]
- hop counting filtering:[42] IP addresses along with hop count are stored. Deviation from expected hop counts serves to detect spoofed IP addresses;
- history-based approaches which include our work and works by for example.[15,16,43]

Recently, researchers from Colorado State University[43] have addressed DDoS mitigation using an extended of the history-based approach originally proposed by Peng et al.[15] The authors use a Bloom filter where they store three octets of the IP addresses in order to save space. The usage of three octets corresponds to our notion of subnets. The authors circumvent the disadvantages of history-based approaches which are unable to distinguish between spoofed and legitimate IP addresses. In order to do this, the authors use the frequency of source IP, port number, and packet size as features for detecting spoofed IPs. They use addresses from Auckland University, CAIDA, and Colorado State University. They extend the approach in Reference 39 to distribute the Bloom filters close to the routers. For a recent survey on the usage of of Bloom Filter for DDoS mitigation we refer the reader to Reference 44. To put it in a nutshell, although there is a stream of history-based approaches in the literature such as References 15, 16, 39, and 43, those approaches rely merely on distance between IP addresses ignoring actual geographical proximity. In our work, we account for the fact that *two IP addresses that are far apart in the address space, using for example Xor, Xor+, or hamming distance, might be close to each other in terms of geographical distance*.

Many studies identify DDoS as unusual volume of traffic. However, it can be just a false alert as flash crowds might take place. During a DDoS attack, most IP addresses are new to the victim, while in the case of flash crowds, this is not the case. Jung et al[45] mention that 82.9% of IP addresses seen in a flash crowd is returning visitors.[46]

Jian et al proposed a method to model traffic flow by using the traditional and popular DBSCAN.[47] The method builds on the principle that clusters are created in higher density areas. The approach is to use rational methods to calculate the distance and design an efficient method to choose correct parameters. A pure DBSCAN algorithm generates a large amount of small clusters. Therefore, small clusters are considered noise and are either rearranged into larger clusters or removed. As a result the number of clusters decreases and the overall efficiency and false alarm rate is improved. The decision whether to merge clusters or not is based on the boundary distance between clusters. Xue-yong and Guo[48] extend the research of Jian. A decision tree algorithms is used to select important features and a heterogeneous distance function HVDM proposed in Reference 49 is used to measure the distance between the desired objects. Xue-yong and Guo used a more efficient way of deciding which cluster to merge together by taking into account whether the distribution is uniform or not. The KDD Cup 1999 dataset is used to evaluate the proposed solution and it is shown to perform better than DBSCAN and the method proposed by Yang Jian.

Liue et al proposed to use density-based fuzzy clustering to distinguish between normal and abnormal traffic flow. The mechanism estimates the relevance of a cluster based on three factors; distance, density, and the trend of density change of data instances in the memberships degree calculation.[50]

3 | PROPOSED METHODS

Previously proposed clustering algorithms within anomaly intrusion detection employ attributes like time to live (TTL) and IP addresses to establish clusters. Well-known publicly available datasets used in anomaly intrusion detection anonymize IP addresses by completely removing any neighbourhood relationship.[51] Proposed clustering algorithms within anomaly intrusion detection which rely on these available datasets to differentiate between normal and abnormal traffic flow are not able to calculate the real relationship between IP addresses. Other algorithms applied to real web-server logs use distance measurements between IP addresses under the assumption that addresses which lay close in the address space also lay close geographically. We argue that clustering or classification techniques which establish patterns based on this assumption, do not necessarily manage to capture the structural properties.

We propose to use weight-based geographical (WBGEO) clustering to overcome this problem and establish a more meaningful clustering. WBGEO builds on DBSCAN which is based on the idea of expanding clusters around points that are considered frequent. A point is considered frequent if there exist at least minpts other points within a radius r. These points then form clusters by expanding to nearby points which are also considered frequent. In weight-based spatial clustering, we modify DBSCANs definition of frequency and introduce a threshold function α to define areas where clusters can form. Weight-based spatial clustering defines a weight function for all existing points. The weight is defined by how much data a point has received. Given an IP address IP_j, the weight ω_{IP_j} associated with this address, is the frequency of requests for this address. Given a set of IP addresses $\mathrm{IP}_1, \ldots, \mathrm{IP}_n$ which share a common geographical position point $p(x, y)_i$, where x is an x-coordinate and y is a y-coordinate, the weight $\omega_{p(x,y)_i}$ associated with this geographical position is given by the sum of weights associated with the IP addresses mapped to the point $p(x, y)_i$ itself, in essence:

$$\omega_{p(x,y)_i} = \sum_{j=1}^{n} \omega_{\mathrm{IP}_j}.$$

Every geographical location point $p(x, y)_i$ can contain several IP addresses $\mathrm{IP}_1, \dots, \mathrm{IP}_n$ which are not necessarily close to each other in the address space. By clustering IP addresses into geographically points, the distance measurement used between the geographically points is able to maintain the inherent pattern. We further define the notion of *core-points*; core-points are geographical location points $p(x, y)_i$ which satisfy the constraint of acquiring traffic over a certain threshold α. Therefore if $\omega_{\mathrm{IP}_j} \le \alpha \& \omega_{\mathrm{IP}_k} \le \alpha$ and if $\omega_{\mathrm{IP}_j} + \omega_{\mathrm{IP}_k} \ge \alpha$ and these addresses share a common location point $p(x, y)_i$, then the point $p(x, y)_i$ is considered a core-point.

In this section, we show that a simple implementation of weight-based clustering can be made by clustering IP addresses into frequent networks. We further expand this notion by collecting IP addresses into their given geographical location point $p(x, y)_i$ and propose two clustering algorithms which establish clusters in geographical regions based on geographical points $p(x, y)_i$ which are considered frequent. We define that a set of acquired geographical clusters $C_1, \dots, C_N$ each consists of a set of geographical location points $p(x, y)_1, \dots, p(x, y)_n$. The two clustering variants use weight-based clustering and builds on the previous stated hypothesis: *If network x at location y reaches the server, there is a higher chance for network z from location y' in vicinity of y to reach the server, than network q that does not belong to a location close to network x.*

3.1 | Apriori-based frequent networks

The first clustering algorithm, Apriori-based frequent networks (AFNs) builds on the principle that common prefixes which occur in the training phase also occur later, when new and unknown data traffic is received. AFN is based on the well-known Apriori[52] algorithm designed for finding frequent itemsets. The Apriori algorithm focuses on finding frequent itemsets by first identifying individual frequent items before extending these items to larger items as long as the itemsets are considered frequent.

AFN assumes it is probable that networks which contain a high weight will reoccur under new and unknown traffic flow. Given a known set of weights associated with a set of 32-bit IP addresses $\omega_{\mathrm{IP}_1}, \dots, \omega_{\mathrm{IP}_n}$ and a threshold α, AFN discovers frequent networks by iterative exploring individual frequent bit-patterns before extending frequently found bit-patterns to larger bit-patterns. Every bit-pattern which is considered frequent is extended into two new bit-patterns consisting of the old bit-pattern including either a *0* or a *1*. *AFN* first checks if bit *0* and *1* contain more than α listings. If both of them do, the bit patterns *00*, *01*, *10*, and *11* are checked. AFN continues extending every frequent bit-pattern until only frequent bit-patterns which contain 24 bits are left. Pseudo code for *AFN* can be seen in Algorithm 1.

Algorithm 1. Pseudo code for *AFN* where T is the transaction database consisting of all IP addresses, α is the frequency level that each itemset or bit format must reach to be considered frequent, and L_k is the current set of items that is checked against frequency for level k

```
function AFN(T, $\alpha$)
    $L_1 \leftarrow [0, 1]$                              ▷ On the first iteration an item is either 0 or 1
    $k \leftarrow 2$
    while $L_{k-1} \neq 0$ and $k \le 24$ do
        $L_k \leftarrow L_{k-1} + 0 \wedge L_{k-1} + 1$    ▷ Increase every bit pattern by 1 and 0.
        for transaction in T do
            $bits \leftarrow transaction[0 : k]$         ▷ Get data from 0th to kth bits in transaction entry
            if $bits \in L_k$ then                       ▷ If bit pattern exist in $L_k$
                $L_k \leftarrow L_k(bits) + 1$           ▷ Increase frequency by 1
            end if
        end for
        $L_k \leftarrow L_k(bits) \ge \alpha$            ▷ Keep all bit patterns with higher frequency than $\alpha$
        $k \leftarrow k + 1$                             ▷ Increase k by 1
    end while
    if $L_k \neq 0$ thenreturn $L_k$
    end if
    return $L_k - 1$
end function
```

When using prefix as a distance measurement, cluster establishment gets constrained into a hierarchy of logical components where these components contain the inability to pass between these logical structures. Consider the two IP addresses 158.121.10.254 and 158.121.11.4. These addresses are close in the address space, however any clustering based on a 24-bit prefix distance will not be able to cluster these elements together. It is possible to lower the prefix matching to 23-bit to account for 24-bit addresses laying close. However, a 23-bit network is still constraint to its own logical component and can lay close to other networks again.

3.2 | Weight-based spatial geographical clustering

A simple distance metric based on prefix matching or bitwise distance in the address space is not enough to represent the geographical differences between these addresses. Our second proposed clustering algorithm is the WBGEO clustering algorithm which extends the notion of finding frequent networks to discovering core-points and expanding clusters around these core-points. In essence, if there exists a location point $p(x, y)_j$ which has acquired a weight $\omega_{p(x,y)_j}$ above a threshold α, this point is defined to be a core-point $cp(x, y)_j$. If the core-point $cp(x, y)_j$ has a path to a neighbouring point $p(x, y)_i$ of a distance d or less, then these two points are said to be *direct-density connected* to each other. If there now exists a path from core-point $cp(x, y)_j$ to point $p(x, y)_n$:

$$cp(x, y)_j, \dots, p(x, y)_{n-1}, p(x, y)_n,$$

where every $p(x, y)_i$ is *direct-density connected* to $p(x, y)_{i+1}$, then the points $cp(x, y)_j$ and $p(x, y)_n$ are defined to be *indirect-density connected*. The points $cp(x, y)_j$ and $p(x, y)_n$ are then considered to be in the same cluster and new traffic from this area is accepted as authentic traffic. If there exists a *indirect-density-connected* path from a core-point $cp(x, y)_j$ to a different core-point $cp(x, y)_i$, the *indirect-density connected* points are merged to form a larger cluster. The pseudo code for this density-based clustering algorithm is described in Algorithm 2:

Algorithm 2. The WBGEO algorithm takes three arguments; the dataset containing a list of latitude/longitude locations with the frequency of packets from those locations. d, which is the maximum distance from a point to a cluster for this point to still be considered a part of the cluster, and α which is the frequency threshold for a point to be considered a core-point

```
1:  function DENSITY_CLUSTERING(dataset, d, $\alpha$)
2:      all_clusters ← (dataset.points ≥ $\alpha$ )
3:      for cluster in all_clusters do
4:          cluster ← (dataset.points ≤ cluster.d)
5:          while cluster ≠ converged do
6:              points ← (dataset.points ≤ cluster.d)
7:              if points in all_cluster.points then
8:                  cluster.merge (points in all_cluster.points)
9:              end if
10:         end while
11:     end for
12:     return all_clusters
13: end function
```

This clustering technique is inspired by DBSCAN. However, instead of expanding clusters as long as points have a certain set of *direct-density connected* points in the neighbourhood, the algorithm assumes that a core-point is surrounded by an area of points which by association should be considered legitimate. Clusters created with WBGEO continues to expand a cluster as long as there exists an indirect-density connected path from a core-point $cp(x, y)_j$ to a point $p(x, y)_n$

3.3 | Reduced WBGEO clustering

The main issue with having a relaxed mechanism for establishing clusters and not restricting cluster growth, is that points which are *indirect-density connected* to at least one core-point is considered equally relevant as any other *indirect-density connected* point regardless of density and cluster association. This is a problem for approaches which choose to discover new blocks of addresses based on a set of known clusters $C_1, \dots, C_N$. We would prefer to only establish clusters in areas which have a high likelihood of reoccurring. For this reason we propose the reduced WBGEO (RBGEO) clustering algorithm to limit cluster growth. First, we make the assumption that a request is more likely to occur at a point $p(x, y)_i$, if the point is

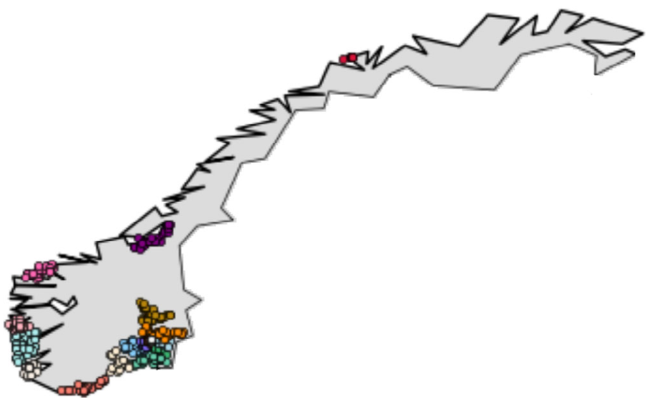


FIGURE 1 Shows a set of clusters computed in Norway. The cluster is created using reduced weight-based geographical with α of 1000, d of 20, minpts of 4, and minlen of zero

direct-density connected to at least $p(x, y)_1, \dots, p(x, y)_{\text{minpts}}$ points. That is, a cluster exists with core-point $cp(x, y)_j$ if there exists an *indirect-density connected* path from $cp(x, y)_j$ to $p(x, y)_n$

$$cp(x, y)_j, \dots, p(x, y)_n,$$

where every point $p(x, y)_i$ on the path is *direct-density connected* or have a path based on distance d to at least minpts other points including $p(x, y)_{i-1}$ and $p(x, y)_{i+1}$.

Clusters are not merged to prevent pattern composition in clusters from deteriorating. Clusters are forced to expand and converge separately. This means a point $p(x, y)_j$ might have an *indirect-density connected* path to several core-points $cp(x, y)_1, \dots, cp(x, y)_n$ where every *direct-density connected point* on the path satisfies a *direct-density connected* path to at least minpts. The point $p(x, y)_j$ is set to be grouped with the closest core-point. In essence, if there exist two core-points $cp(x, y)_i$ and $cp(x, y)_v$ with two *indirect-density connected* paths to $p(x, y)_j$, then the point $p(x, y)_j$ is grouped with the closest core-point of $cp(x, y)_i$ and $cp(x, y)_v$. Finally, clusters can potentially consist of only one core-point. We expect any geographical area which contain legitimate traffic to cover a larger area and consist of a set of points $p(x, y)_1, .., p(x, y)_n$. We therefore remove every cluster which does not consist of minlen points or more. The points, including the core-point, should then be repositioned to other clusters nearby. The repositioning of dismantled points is done with regards to the point's closest core-point, excluding the core-point for the dismantled cluster. Geographical clusters resulting from this algorithm can be seen in Figure 1.

4 | IMPLEMENTATION

4.1 | Datasets

Our three proposed algorithms are tested using two datasets from Norwegian web-servers, Dataset 1 (D1) and Dataset 2 (D2). D1 is obtained from the institute website *www.cs.hioa.no* and consists of 1.5 million requests. D2 is obtained from the campus website *www.hioa.no* and consists of 1.6 million requests. The datasets contain what is perceived as the normal traffic flow of the system and is used to model the perceived behaviour of a system under normal circumstances. The raw database of D1 and D2 originally consisted of 32-bit IP addresses, but for data privacy reasons the last eight bits were removed before handed to us, thus identifying only the source subnets and not individual source IPs. Each unique subnet address can repeat itself in the database corresponding to the amount of times this subnet address sent requests to the server. We divide the database into two sets, a training set and a testing set. The training set consists of 2/3 of the database and is used to model the normal perceived behaviour, while the testing set consists of the remaining 1/3 and is used to test how well the achieved model is able to recognize new traffic as legitimate.

4.2 | Normalization

Under the training phase, we normalize the database to consist of unique 24-bit subnet addresses with an extra field corresponding to the frequency of the encountered address. For geographical clustering, we normalize the database further and locate the geographical location $p(x, y)_i$ for each unique 24-bit address. This is accomplished by using online databases which consist of a set of address blocks with known geographical position.[23,24] We change our database fields to now consist of the geographical location and the amount of times requests from this location were sent to the server. The AFN algorithm generates, from the training set, a set of 24-bit IP addresses which is considered frequent if above the threshold α. On the other hand, the geographical clustering algorithms generate a set of clusters $C_1, \dots, C_N$ where every C_i consists of a set of geographical location points $p(x, y)_1, \dots, p(x, y)_n$ and at least one core-point $cp(x, y)_i$ which has a source frequency larger than or equal to α. In geographical clustering,

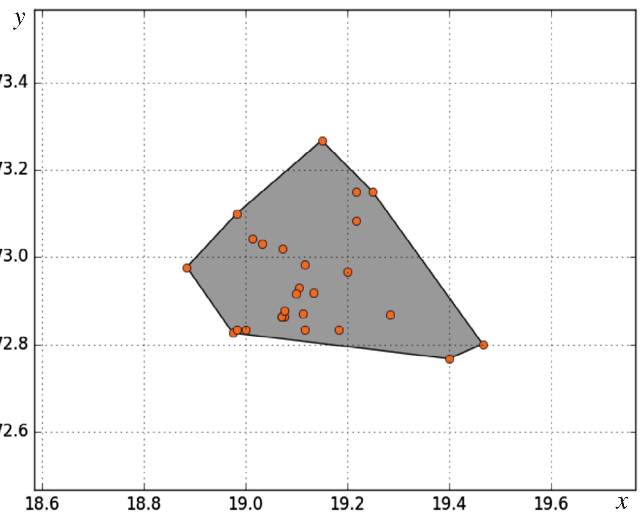


FIGURE 2 A small cluster created from weight-based geographical with an initial cluster threshold $\alpha = 1000$ and a maximum distance of $d = 25$ km. All subnets within the grey zone are accepted as a part of this cluster. Address amplification consists of adding all the subnets that the Geo-IP database lists within this area.

there are two main options which will be investigated in the next section. We can traverse our original list consisting of a set of IP,$p(x, y)_i$ entries to achieve the original IP addresses obtained from the calculated clusters $C_1, \dots, C_N$ during the training phase. And we can then choose to only accept these addresses as legitimate traffic. The second option is to populate the database with more addresses which exist in the geographical regions the different clusters encompass. Hereafter we will refer to this operation as *address amplification*. The addition of new IP addresses, *address amplification*, is achieved by creating a *polygon* for each cluster C_i. This idea can be seen from a small cluster in Figure 2. Every IP address which originates from the area within the *polygon* will be accepted as legitimate traffic. We resort to a Geo-IP database consisting of two fields; address blocks and geographical location points $p(x, y)_i$ associated with each block. We then check which location points fit within each polygon. From this we obtain a set of address blocks $\text{ab}_1, \dots, \text{ab}_n$ which can be added to the address database.

4.3 | Structures

To quickly determine whether an IP address is present in the discovered 24-bit addresses in the training set, we test both a Bloom filter structure and a tree structure. A tree structure, is a hierarchical data structure and can consist of several nodes or data items which have direct or indirect relations to new nodes. We represent a 24-bit network which should be accepted as a series of 1s and 0s. From the root node, left indicates a 0, while a right indicates a 1. Any IP address which can traverse 24 levels is indicated as accepted. A packet which is accepted has a time complexity of $\mathcal{O}(n)$. For packets that are dropped, the best scenario for traversing is $\mathcal{O}(1)$, while the worst scenario is $\mathcal{O}(n-1)$. A tree structure can be seen as an efficient structure but is less space efficient than Bloom filters.

Bloom filters have previously been used within anomaly detection[53] and DDoS mitigation[54] and is a space efficient probabilistic structure which uses hashing to determine whether elements are in a list S. If an element x is determined to be in the set S, x is in set S with a known probability. If x is determined to not be in the set, x is with a 100% certainty not in the set. This means that Bloom filters do not allow false negatives but allow false positives. In our context a false positive is when the filter falsely accepts an address which should have been rejected. A higher false positive rate of the filter generally leads to less storage usage. The drawbacks of false positives are often out-weighted by the space saved. Hash functions are demanding elements to calculate and the time complexity of a Bloom filter is not based on the amount of elements in the set, but the number of hash functions. The time complexity of a Bloom filter is $\mathcal{O}(k)$, where k is the number of hash functions.[55,56] We use *PyBloom* which is a python implementation of the Bloom filter probabilistic data structure. The Bloom filter takes as input an error rate and the amount of elements which will be inserted. In our scenario, we insert 24-bit IP addresses obtained from the algorithms before checking whether new elements exist in the filter.

4.4 | Botnets

To which extent the geographical clustering is able to differentiate between normal traffic flow and attack traffic from a botnet, depends on the geographical positions of the generated attacker traffic in comparison with the perceived geographical positions of the normal traffic flow. Dagon et al conclude that bots often are geographically spread apart while also being concentrated in particular regions.[57] It is reasonable to assume that different spreading-mechanisms might help the botnet to have a higher concentration of bots in some geographical regions. Rajab et al was able to capture these structural features of a botnet as seen in Figure 3.[58]

We test three generated botnets which contain different geographical patterns. The first simulated botnet, denoted as B1, contains 100 000 unique IP addresses which are randomized over the complete IP address space. The second simulated botnet is B2 and contains 118 000 IP

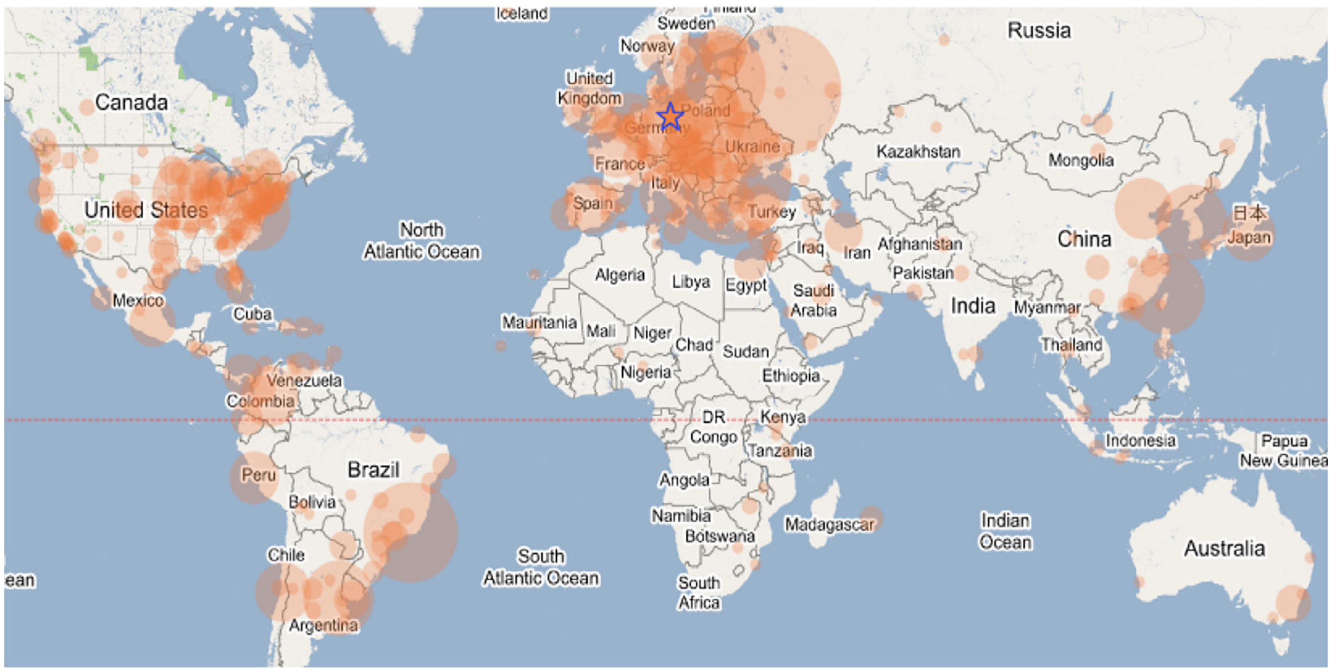


FIGURE 3 The domain name system cache hits for one of the tracked botnets as reported by Rajab et al.[58] The star indicates the location of the IRC server of the botnet

addresses. The IP addresses of B2 are situated in Europe and stretches mainly over central Europe. The bots of B2 are for the most part situated in Great Britain, France, Spain, Italy, Germany, and Poland. Unlike B1 which contains a randomized botnet, B2 and B3 have a unique geographical patterns and are more similar to real botnets than B1. Botnet B3 has addresses from all of Norway and consists of 38 000 unique IP addresses.

4.5 | The testbed

A real DDoS attack is performed using BoNeSi,[59] an open source botnet simulator for generating TCP based HTTP-GET flood attacks. The IP packets are generated using a set of botnet addresses on one server and sent to a standard Linux Web server running on another node in the same physical subnet. The Linux Web server is responsible for answering legitimate HTTP requests and mitigating the attack. To the Linux server, the incoming HTTP-packets are completely similar to what it would receive during a real DDoS attack and the experiment is perceived as authentic from the web server's point of view. The reply packets addressed to servers all over the world are dropped by the gateway of the lab network. Our mitigation approach is coded in Python. The mitigation technique requires access to network packets in kernel space. We use Nfqueue which is a C extension module to delegate the decisions on packets to our user space software. At the same time as the botnet attack takes place, normal connections to the Web server is generated from another server using Apache Traffic Replay Generator (Repache). This is a tool that can replay Apache log-files to generate HTTP requests based on recorded IP addresses.[60]

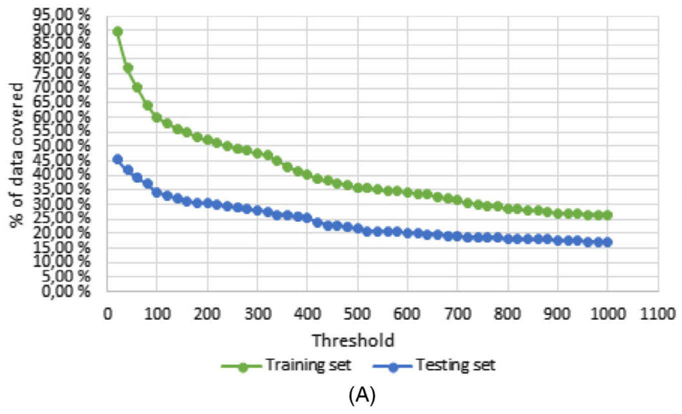
5 | RESULTS

The result section is divided into two parts, data mining and simulation. In the data mining part we investigate how well the model obtained from the training phase is able to fit the traffic flow of the testing phase. We consider benefits and drawbacks from supplementing geographical clusters with new IP addresses. In the simulation phase, we test our approach using different botnets and perform an actual DDoS attack.

5.1 | Data mining

5.1.1 | Apriori-based frequent network

The results of the AFN algorithm applied to the datasets D1 and D2 using different frequency thresholds α can be seen in Figure 4. First all the subnets of a frequency larger or equal to α in the training set is collected as a database. Then the percentage of connections which are matched



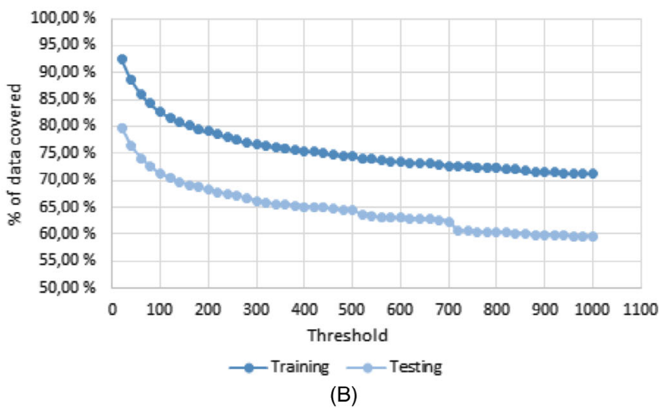
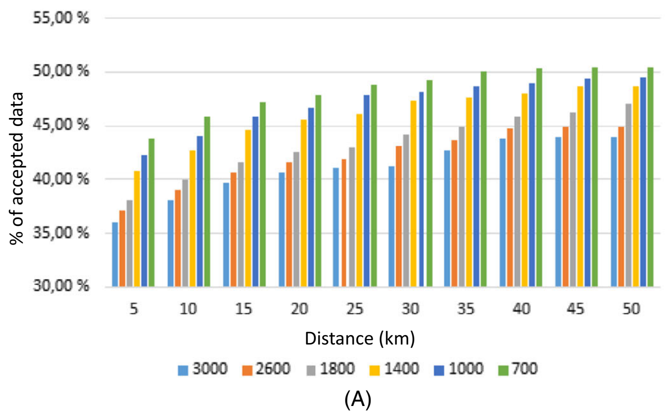


FIGURE 4 Percentage of connections covered by the database generated by the Apriori-based frequent network algorithm with αs in the range from 20 to 1000 when matching the training and the testing set. Graph (A) shows results for the dataset D1 and graph (B) for the dataset D2



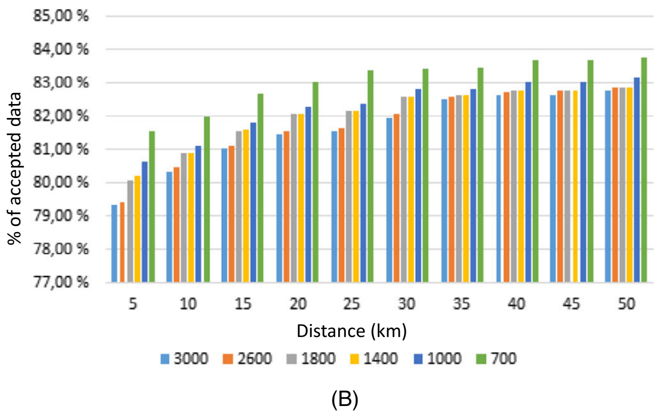


FIGURE 5 The amount of accepted testing data for D1 (A) and D2 (B) when clusters are computed using weight-based geographical. The clustering is done with distances from 5 to 50 km and α of 3000, 2600, 1800, 1400, 1000, and 700

by the database for the incoming subnets of the training and the testing sets respectively is calculated and this is shown in the figure. For $\alpha = 1$, a 100% of the connections of the training set would be covered, but not all of the testing set. 54.4% of the testing set is covered with $\alpha = 1$ for D1 while 86.3% is covered for D2. This means that for the web server from which the D2 data are obtained, it is in principle possible to accept 86% of the normal benign the traffic during a DDoS attack, if making a firewall filter which accepts only subnets from this database. Our hypothesis is that requests from the same subnets repeat over time and this hypothesis is more accurate for D2 than for D1.

However, the complete database covering 86% of the testing connections is very large and might lead to a too large firewall filter. The purpose of introducing the frequency threshold α for including subnets, was in order to reduce the database size. When increasing the frequency threshold α in the database, the size of the database is strongly reduced. However, the reduction of the number of connections which are accepted by the database is not reduced as much. As seen from Figure 4B for dataset D2 the number of connections of the testing set covered by the database is still 60% for $\alpha = 1000$.

The corresponding percentage for the training set of connections covered by the database is 70% for $\alpha = 1000$. This means that 70% of the connections of the training set comes from subnets which have been seen 1000 times or more. The difference of 10 percentage points between the training set and testing set coverage is remarkably small. Most of the frequent subnets of the training set also occur in the testing set. The same can be seen from Figure 4A, although the total coverage is less for the D1 dataset.

5.1.2 | Geographical clustering

Figure 5 shows the acceptance rate for clusters computed applying WBGEO to the datasets D1 and D2 with different distances and core-point threshold α. The acceptance rate is calculated without address amplification.

TABLE 1 The number of subnets found for D2 clustering when address amplification is used to populate the clusters with more networks

Distance (km)	WBGEO	RBGEO
5	608117	107788
10	683489	279804
15	927265	360763
20	941675	376277

Note: Clusters are created with a α of 3000 and d between 5 and 20. The RBGEO clusters have an added constraint of a minsize of 2 and a minpts of 3. Abbreviations: RBGEO, reduced weight-based geographical; WBGEO, weight-based geographical.

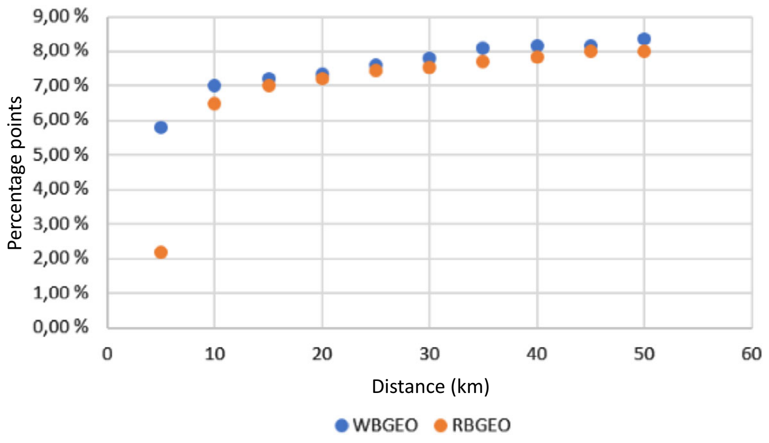


FIGURE 6 The percentage point increase in accepted connections during the testing phase when adding subnet addresses using address amplification compared to when only using addresses recorded during the training phase. WBGEO is run with $\alpha = 3000$ while RBGEO has added constrains of minlen = 2 and minpts = 3. RBGEO, reduced weight-based geographical; WBGEO, weight-based geographical

As seen for AFN in Figure 4, the D2 dataset fits our model better than D1 as roughly twice as many connections are accepted in the D2 case. Reducing the size of the frequency threshold α, which sets a lower limit for the frequency of a cluster core-point, improves the percentage of accepted connections substantially. However, it should be noted that this also leads to larger databases. Indirectly increasing the geographical size of the clusters by increasing the distance d also leads to a higher connection acceptance, but not as profoundly as when decreasing α. Increasing the distance d to more than 20-25 km will, as also can be seen from Figure 4, lead to a somewhat larger acceptance, but the corresponding increase in database size is drastic.

Without using address amplification, the upper limits for the resulting subnet databases when using geographical clustering are the corresponding complete AFN databases for $\alpha = 1$, containing all subnets seen. Unless any of the geographical core-points are listed with other subnets in the Geo-IP database, then these subnets will be included in the geographical subnet database.

RBGEO clustering contains some of the same interesting patterns as WBGEO. The amount of accepted data varies depending on additional parameters like minlen and minpts. The intention of RBGEO is to by stricter constraints reduce the size of the generated clusters when compared to WBGEO. Interestingly this also decreases the difference between the accepted amount of connections in the training and the testing set. The main drawback of applying address amplification to a set of geographical clusters $C_1, \dots, C_N$ computed by these two algorithms, is that the amount of discovered address blocks increases rapidly with sets of wider and larger clusters. Table 1 shows the resulting amount of accepted 24-bit networks when address amplification through reverse lookup is used to populate D2 clusters for WBGEO and RBGEO with more networks.

RBGEO was designed in order to reduce the size of the wider and larger clusters of WBGEO. It is interesting to compare the difference between these two clustering approaches and whether the benefits of the larger number of networks found using WBGEO leads to a higher increase in accepted connections during the testing phase than RBGEO. In Figure 6 the effect on the two algorithms when doing subnet amplification can be seen. This is the process of looking up addresses of all subnets inside the geographical area of the clusters; addresses which have not been seen during the testing phase. The graph shows that the effect of address amplification is almost just as profound for the RBGEO algorithm and this is despite that the size of the resulting subnet databases is much smaller as seen in Table 1.

RBGEO is to some degree able to include areas in a cluster from which future connections have a higher likelihood of originating. As seen from Table 1, a maximum distance of 10 km gives about 400.000 less subnets than WBGEO. When comparing this to Figure 6, the percentage point increase in this case is practically the same.

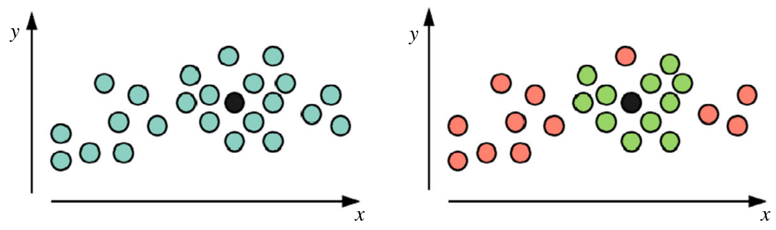


FIGURE 7 A geographical cluster consisting of blue points. An improved method could be to apply address amplification only to high density areas in the cluster, represented by the green points. The black point is the core-point

TABLE 2 Percentage of IP addresses from the botnets B1, B2, and B3 which are accepted for the datasets D1 and D2 by the AFN database

α	B1		B2		B3	
	D1	D2	D1	D2	D1	D2
1	0.206	0.123	0.355	0.32	41.5	45.7
5	0.089	0.055	0.142	0.097	19.2	23.4
10	0.065	0.034	0.126	0.032	13.9	14.9
20	0.050	0.026	0.060	0.020	8.7	9.7
50	0.022	0.011	0.025	0.003	3.4	4.1
100	0.008	0.006	0.013	0	1.8	1.6
500	0.001	0	0.001	0	0.12	0.16
1000	0	0	0	0	0.02	0.09

Abbreviation: AFN, Apriori-based frequent network.

It could be possible to combine the two geographical clustering algorithms to maximize the amount of accepted data while maintaining a low rate of accepted networks. This could be done by applying address amplification in high density areas of a cluster while only accepting seen addresses in low density areas. This could maximize the connection acceptance rate while minimizing the number of networks in the database. This proposed method is illustrated in Figure 7.

5.2 | Simulation

In this section botnet attacks are simulated using the fabricated IP addresses of the botnets B1, B2, and B3. The amount of accepted botnet-addresses given the different algorithms is investigated. For geographical clustering we use a Bloom filter to store the networks extracted during the training phase, while for AFN we use a tree structure to store the learned networks before checking how much botnet traffic is accepted. Finally, as a proof of concept, we perform an actual DDoS attack using IP addresses from B2 and a subnet database generated from dataset D2. We show that geographical clustering is able to differentiate between normal and abnormal traffic flow as long as the attack traffic does not follow a normal geographical pattern.

5.2.1 | Botnet traffic

Any attack which manages to adapt itself to the normal incoming traffic pattern will be able to counter any mitigation designed to differentiate between normal and abnormal traffic flow. How successfully geographical clustering is able to differentiate between DDoS traffic and normal traffic flow depends on the attackers ability to resemble the normal traffic pattern. Services which contain a diverse geographical pattern often face a more difficult task in mitigating a DDoS attack.

Table 2 shows the percentage of subnet addresses which are accepted when our systems are attacked using addresses from the botnets B1, B2, and B3 when using the AFN algorithm trained on datasets D1 and D2 for various thresholds α.

The addresses are stored using a tree structure so only addresses contained in the filter will be accepted. For the botnets B1 and B2 with random addresses and addresses from Europe, the accepted percentage is very low for both datasets, even for $\alpha = 1$ when all addresses seen before are included in the filter. This means that a DDoS filter will be quite effective and as long as the server is able to cope with less than a percent of the incoming requests, a DDoS attack will not take the server down. In case of the B3 botnet, the table shows that more than 40% of the addresses of the botnet coincide with the addresses seen before. This makes it difficult to make an efficient filter since close to half of the traffic would be accepted.

TABLE 3 Percentage of IP addresses from the botnets B1 and B2 which are accepted for the datasets D1 and D2 when WBGEO is used to generate the geographical clusters, using $\alpha = 3000$

d	B1		B2		B3	
	D1	D2	D1	D2	WBGEO	RBGEO
5	0.049	0.040	0.006	0.003	22.4	21.3
10	0.062	0.043	0.007	0.007	26.9	24.6
15	0.075	0.055	0.013	0.008	31.1	30.6
20	0.081	0.055	0.007	0.001	32.6	32.6
25	0.077	0.056	0.008	0.009	33.0	32.7
30	0.086	0.066	0.10	0.021	36.7	36.3

Note: In the two rightmost columns, results for botnet B3 and dataset D2 are shown. A Bloom filter with an error rate of *f* 0.0001 is used. Address amplification is not applied. Abbreviations: RBGEO, reduced weight-based geographical; WBGEO, weight-based geographical.

However, it is possible to make a good filter using a large α, as the accepted botnet traffic will be less than 1%. And at the same time, if for instance an α of 500 is used, the amount of accepted benign traffic in this case was shown to be as high as 65% in the testing phase as seen in Figure 4B.

It is interesting to notice the difference between a simple algorithm like AFN and a more complex algorithm which tries to define a clear geographical structure. The results using WBGEO can be seen in Table 3 for the botnet B1, where a Bloom filter is used to accept new traffic. Since WBGEO has a clearly defined structure, even large *d*s will not cause a lot of accepted botnet traffic, but it will lead to an increase in the amount of accepted networks as was seen in Figure 5.

This effect is even more profound when looking at a botnet with a more unique geographical pattern. The addresses of the B2 botnet are from all of Europe, but most of them are from central Europe. Both the datasets D1 and D2 contain traffic from Norwegian web-servers and there should be little to none accepted traffic from the B2 botnets.

AFN selects an address based only on the value of α and does not coincide the local neighbourhood. Hence it is not able to detect IP addresses that are geographically close to each other while not sharing a common prefix. Geographical clustering, which looks at geographical patterns with regards to a point $\omega_{p(x,y)}$, is by design better equipped for removing abnormal subnets in the set. This can be seen by comparing AFN in Table 2 with geographical clustering using WBGEO in Table 3 for B1 and B2; the latter in general accepts less of the European addresses.

It is important to reiterate that geographical clustering in general contains a larger amount of discovered networks from the training set compared to AFN. However, since the botnet contains a pattern which is quite different from the normal geographical pattern, it is easier to differentiate between normal and abnormal traffic flow.

However, as seen from the two rightmost columns of Table 3, for the botnet B3 which contains addresses in the same country as the datasets D1 and D2, the geographical algorithms WBGEO and RBGEO are accepting too large fractions of the botnet addresses. RBGEO is designed to reduce the size of the clusters and accepts slightly less addresses, but still too many for an efficient DDoS filter.

In the results above, address amplification has not been applied. If the attack pattern is not geographical near acquired clusters, supplementing with new addresses will have little to no effect on the acceptance rate of illegitimate traffic. Widely distributed clusters have a higher risk of allowing more botnet traffic. Stricter limitations on cluster growth can help mitigate this effect to some degree but cannot completely prevent it.

5.2.2 | A real life DDoS attack

As a proof of concept we performed a botnet attack using the B2 botnet addresses and the recorded D2 dataset of actual connections to our local web server. A subset of D2 was used to replay the real network traffic consisting of 50 000 requests over 24 hours. WBGEO applied to the training set was used to generate a set of clusters with a distance of $d = 10$ km and $\alpha = 3000$. Based on the generated database, 28 800 of the 50 000 requests of the testing set should be accepted. This amounts to 57.6% of the addresses and is lower than the acceptance rate which would be achieved when using the whole testing set of D2, a rate of 80.3%. A Bloom filter with a *0.0001* error rate was used to store the 24-bit subnet addresses generated by WBGEO that should be accepted. The attack was performed by sending 120 thousand requests every second using the B2 source addresses together with the normal Web traffic. The acceptance rate can be seen in Table 4 while the statistics for a legitimate request to be processed and accepted, can be seen in Table 5.

The average number of NFQUEUE queue elements waiting to be processed during the attack was 211 elements. The kernel reported that 105.170 packets were dropped because netlink was not able to send packets to user space. This might imply that either the netlink buffer size was too small or that NFQUEUE was not able to handle this amount of traffic over a larger period of time. However, the rate of packet loss corresponds

Traffic	Accepted (%)	Dropped (%)
Legitimate	57.59	42.41
Illegitimate	0.007	99.993

Abbreviation: DDoS, distributed denial of service.

TABLE 4 The percentage of accepted and dropped requests of the DDoS attack

Minimum	Maximum	Average	Median
4 ms	7.02 s	15.64 ms	11 ms

Abbreviation: DDoS, distributed denial of service.

TABLE 5 Statistics for the request time that was recorded for legitimate requests during the DDoS attack

to just slightly more than a single packet dropped every second and the amount of accepted legitimate packets was 57.6% as expected. And the amount of accept botnet packets was 0.007 % as expected from Table 3.

6 | DISCUSSION

Clustering is an unsupervised learning approach which tries to model the normal traffic pattern of the network. This approach assumes that the abnormal traffic flow deviates from the acquired model. Seeing the differences between the normal and abnormal traffic flow allows us to mitigate DDoS attacks. It is preferable to create a model which represents the normal traffic flow as accurately as possible.

In the following we discuss how to determine the various parameters of the models presented in this article. A simple model would be to accept all previously seen traffic. The AFN algorithm defines a request threshold α. If the number of requests from a subnet is below this threshold, it will not be included in the DDoS filter. For $\alpha = 1$, the AFN algorithm includes all subnets that have sent requests and the model is equivalent to history-based filtering.[15] It is obvious that accepting all previous traffic leads to the highest rate of accepted legitimate traffic. However, such a model will statistically also accept the highest amount of botnet traffic. We can decrease the number of subnets in the filter by increasing α, accepting less traffic and avoiding a total DoS, but at the cost of denying a slightly larger part of the normal traffic.

In general, there is no exact α which is optimal for a system. The choice of the optimal value of α will depend on the given dataset and determining α for a particular network must be done through a specific analysis for this network. Some servers receive requests from all of the world while others receive requests from certain parts of the world only and hence from a much smaller number of subnets.

The choice of optimal model parameters will also depend on the actual subnet addresses taking part in the attack. If there is a substantial overlap with the subnet addresses of the filter, the mitigation will fail as too much of the attack traffic will be accepted. Moreover, the determination of α needs to be seen in relation to the capabilities of the system. A system should accept as much traffic as possible as long as the server can handle the incoming requests.

A possible solution to the problem of determining an optimal α suitable for any attack is a layered approach with several filters. In each filter, a different set of parameters is used to determine a traffic model of the system. At the beginning of an attack we use the largest filter, accepting as much of the benign traffic as possible. If the traffic accepted by this filter overloads the server, a smaller filter is selected. The various filters can be computed in advance and exchanged as needed, either because of increasing attack intensity or changes of the attack pattern. A next step in such a dynamic approach could be to adapt the filters to the incoming traffic seen during the attack.

The geographical clustering algorithms introduce a few additional parameters. The α parameter for these algorithms is not a threshold for the number of requests from a network, but a threshold for the number of requests from a geographical location. The geographical clustering algorithms try to improve the traffic model by generating clusters of subnets located in geographical regions which are not assumed to host botnet IPs. By increasing the distance *d*, larger geographical regions are covered with the additional risk of covering an area where botnet IPs reside. WBGEO generates clusters based only on the parameters α and *d*. By populating the database with additional addresses from areas not considered hostile, clustering has the potential to produce a model which accepts more data than what is possible with AFN and history-based filtering.[15] However, geographical clustering may lead to clusters of subnets which are too large and filters which accepts too many addresses. Just as for the AFN algorithm, the parameters must be tuned so that the resulting filter drops just enough of the incoming traffic to avoid a total DoS. Increasing the threshold α and decreasing the distance parameter *d* lead to less accepted traffic.

The RBGEO algorithm was introduced in order to minimize the disadvantages of adding too many new addresses to the database by reducing the cluster areas which can be populated. This reduction is obtained by adding two parameters; minsize and minpts. A cluster is required to contain a at least minsize points at the end of the clustering process and each point in a cluster is required to have a minimum of minpts neighbours within a radius *d*. Increasing these parameters means that fewer clusters are found and this decreases the corresponding filter size while keeping the ability to accept benign traffic from subnets which historically have not been seen before. In one of our cases, the WBGEO algorithm finds and adds 680 000 new addresses to the database. In the same case, the RBGEO algorithm finds and adds 279 000 new addresses, but still, both algorithms accept approximately the same amount of traffic from subnets never seen before.

Filtering traffic based on IP addresses has promising results. Addresses which appear in the normal traffic flow have a high likelihood of appearing again later. Peng et al[15] showed that 80-90% of the traffic in packet traces from the University of Auckland which appeared in one day, reappeared during the next day. Roughly the same was the case for a small ISP Trace from a C class network in Australia. These results correspond to our model in the case of choosing a threshold $\alpha = 1$ and when using 1/3 of the datasets as testing set, 54% of addresses from subnets seen before reappeared for one of the sets and 86% for the other. Peng et al used individual IP addresses while we used subnets and the large deviation in the two cases in terms of the number of IP addresses that reappear shows that it is in general difficult to directly compare the results of various DDoS history-based mitigation methods.

7 | CONCLUSION

Three different algorithms have been proposed in this article to mitigate DDoS attacks. The algorithms are mainly applied to mitigate HTTP flood attacks, but may also be used to mitigate other attacks at the application layer. The first algorithm, AFNs, is based on finding networks frequently seen in historical data. It is quite similar to the HIF algorithm proposed by Peng et al.[15] The efficiency of such a method depends on to which extent connections from the same subnets reappear. We apply it successfully to collected datasets of two different web servers. A novelty in this context is that AFN uses a three-structure which makes both generating and using the resulting subnet filter more efficient.

There have been several attempts to improve on the history-based method using clustering techniques.[16,17] One of the most important aspects of a clustering process is to select a distance measurement between objects. Until now clustering has been based on the distances between binary addresses, but the geographical similarity between IP addresses cannot simply be unveiled based on the addresses themselves.

The other two algorithms we propose, WBGEO clustering and RBGEO clustering, are based on creating clusters in geographical areas utilizing Geo-IP databases. Anomaly-based mitigation techniques often resort solely on the IP addresses in the clustering process.[21,61] However, these clustering methods takes advantage of the mathematical IP neighborliness between networks which are close in the address space to cluster them together. This is not necessarily ideal, as networks which are close in the address space, might not be close geographically. Moreover, this assumption might lead to networks which are far away in the address space being assumed to be anomalous, even though they are close together geographically.

To the best of our knowledge, clustering techniques within anomaly intrusion detection have never explored the possibility of defining clusters based on geographical regions. Geographical density-based clustering gives several advantages that cannot be achieved with normal IP address clustering. In this perspective, anomalies can be more correctly determined, new networks and IP addresses can be found based on defined clusters, and clusters can correctly identify data points which have a higher likelihood of occurring in the testing phase. Moreover, based on geographical clusters, different attributes such as TTL can be related to each independent cluster and can give a more correct view of hidden data structures. This will make it harder for an attacker to conceal within the normal traffic flow.

The ultimate goal of the data-mining part of this work was to make a DDoS filter based on the generated subnet databases. We were able to scale down the size of the databases by making the inclusion of an address depending on how frequent the subnet had been seen for AFN and on geographical radius for the other two algorithms. This makes it possible to use a small filter when the DDoS attack is excessive and a larger filter when the system is able to cope with a larger fraction of the attack traffic. Using address amplification, we were able to increase the amount of addresses accepted by the filter with up to 8% points as was seen in Figure 6. These extra addresses were not seen in the historic data, but were among the candidates predicted by the geographical algorithms.

Finally, as a proof of concept, we simulated a close to real life DDoS attack using European botnet subnet addresses. As 99.99 % of the illegitimate traffic was dropped, we showed that geographical clustering is able to differentiate between normal and abnormal traffic flow as long as the attack traffic is sufficiently different from the normal geographical pattern learned from the training data. It would be very difficult to make an efficient filter using addresses from a botnet in the same geographical area as most of the normal traffic came from, but such a scenario can generally be considered as quite unlikely.

ORCID

Anis Yazidi https://orcid.org/0000-0001-7591-1659
Hugo Hammer https://orcid.org/0000-0001-9429-7148

REFERENCES

1. Kongshavn MV, Yazidi A, Haugerud H, Hammer H. Data mining approaches for IP address clustering. Paper presented at: Proceedings of the International Conference on Industrial Networks and Intelligent Systems; 2017:266-275; Springer.

2. Thangavel S, Kannan S. Detection and trace back of low and high volume of distributed denial-of-service attack based on statistical measures. *Concurr Comp: Pract Exp*. 2019;e5428.
3. Fenil E, Mohan Kumar P. Survey on DDoS defense mechanisms. *Concurr Comp: Pract Exp*. 2019;e5114.
4. Mahjabin T, Xiao Y, Sun G, Jiang W. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *Int J Distribut Sens Netw*. 2017;13(12):1-33.
5. Moore D, Shannon C, Brown DJ, Voelker GM, Savage S. Inferring internet denial-of-service activity. *ACM Trans Comput Syst (TOCS)*. 2006;24(2):115-139.
6. Kalkan K, Gür G, Alagöz F. Filtering-based defense mechanisms against DDoS attacks: a survey. *IEEE Syst J*. 2017;11(4):2761-2773.
7. Vetha S, Vimala Devi K. A trust-based hypervisor framework for preventing DDoS attacks in cloud. *Concurr Comp: Pract Exp*. 2019;e5279.
8. Osanaiye O, Choo KR, Dlodlo M. Distributed denial of service (DDoS) resilience in cloud: review and conceptual cloud DDoS mitigation framework. *J Netw Comput Appl*. 2016;67:147-165.
9. Somani G, Gaur MS, Sanghi D, Conti M, Buyya R. DDoS attacks in cloud computing: issues, taxonomy, and future directions. *Comput Commun*. 2017;107:30-48.
10. Dong S, Abbas K, Jain R. A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments. *IEEE Access*. 2019;7:80813-80828.
11. Wang B, Zheng Y, Lou W, Hou YT. DDoS attack protection in the era of cloud computing and software-defined networking. *Comput Netw*. 2015;81:308-319.
12. Behal S, Kumar K, Sachdeva M. D-FACE: an anomaly based distributed approach for early detection of DDoS attacks and flash events. *J Netw Comput Appl*. 2018;111:49-63.
13. Cheng J, Xu R, Tang X, Sheng VS, Cai C. An abnormal network flow feature sequence prediction approach for DDoS attacks detection in big data environment. *Comput Mater Con*. 2018;55(1):95-119.
14. Najafabadi MM, Khoshgoftaar TM, Calvert C, Kemp C. User behavior anomaly detection for application layer DDoS attacks. Paper presented at: Proceedingsof the 2017 IEEE International Conference on Information Reuse and Integration, IRI 2017; 2017-January; 2017:154-161.
15. Peng T, Leckie C, Ramamohanarao K. Protection from distributed denial of service attacks using history-based IP filtering. Paper presented at: Proceedings of the Communications, 2003. ICC'03. IEEE International Conference on. Vol. 1. IEEE; 2003:482-486.
16. Goldstein M, Lampert C, Reif M, Stahl A, Breuel T. Bayes optimal ddos mitigation by adaptive history-based ip filtering. Paper presented at: Proceedings of the Networking, 2008. ICN 2008. 7th International Conference on IEEE; 2008:174-179.
17. Pack G, Yoon J, Collins E, Estan C. On filtering of DDoS attacks based on source address prefixes. Paper presented at: Proceedings of the Securecomm and Workshops, 2006 IEEE; 2006:1-12.
18. Xu R, Wunsch D. Survey of clustering algorithms. *IEEE Trans Neural Netw*. 2005;16(3):645-678.
19. Schaeffer SE et al. *Algorithms for Nonuniform Networks*. Helsinki, Finland: Helsinki University of Technology; 2006.
20. Agrawal S, Agrawal J. Survey on anomaly detection using data mining techniques. *Proc Comput Sci*. 2015;60:708-713.
21. Goldstein M, Reif M, Stahl A, Breuel T. Server-side prediction of source IP addresses using density estimation. Paper presented at: Availability, Reliability and Security, 2009. ARES'09. International Conference on IEEE; 2009:82-89.
22. Qin X, Xu T, Wang C. DDoS attack detection using flow entropy and clustering technique. Paper presented at: Proceedings of the Computational Intelligence and Security (CIS), 2015 11th International Conference on IEEE; 2015:412-415.
23. MaxMind. GeoLite legacy databases. 2017. https://dev.maxmind.com/geoip/legacy/geolite/.
24. MaxMind. MaxMind's GeoIP2 databases. 2017. https://www.maxmind.com/en/geoip2-databases.
25. Likas A, Vlassis N, Verbeek JJ. The global k-means clustering algorithm. *Pattern Recogn*. 2003;36(2):451-461.
26. Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Trans Pattern Anal Mach Intell*. 2002;24(7):881-892.
27. Hartigan JA, Wong MA. Algorithm AS 136: a k-means clustering algorithm. *J Royal Stat Soc Ser C (Appl Stat)*. 1979;28(1):100-108.
28. Bradley PS, Fayyad UM. Refining initial points for K-means clustering. Paper Presented at: Proceedings of the ICML.'98; 1998:91-99; Citeseer.
29. Ranjan R, Sahoo G. A new clustering approach for anomaly intrusion detection. 2014. arXiv preprint arXiv:1404.2772.
30. Birant D, Kut A. ST-DBSCAN: an algorithm for clustering spatial–temporal data. *Data Knowl Eng*. 2007;60(1):208-221.
31. Ester M, Kriegel HP, Sander J, Xu X, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. Paper presented at: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (Kdd.'96); 1996:226-231; Menlo Park, CA.
32. Bhatia S, Behal S, Ahmed I. Distributed denial of service attacks and defense mechanisms: current landscape and future directions. *Versatile Cybersecurity*. Cham: Springer; 2018:55-97.
33. Mahajan R, Bellovin SM, Floyd S, Ioannidis J, Paxson V, Shenker S. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM Comput Commun Rev*. 2002;32(3):62-73.
34. Mirković J, Prier G, Reiher P. Attacking DDoS at the source. Paper presented at: Proceedings of the Network Protocols, 2002. 10th IEEE International Conference on IEEE; 2002:312-321.
35. Argyraki K, Cheriton DR. active internet traffic filtering: real-time response to denial-of-service attacks. Paper presented at: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC '05. USENIX Association; 2005:10Berkeley, CA.
36. Liu X, Yang X, Lu Y. To filter or to authorize: network-layer DoS defense against multimillion-node botnets. Paper presented at: Proceedings of the ACM SIGCOMM Computer Communication Review; 38. 2008:195-206; ACM.
37. Ramanathan S, Mirkovic J, Yu M, Zhang Y. SENSS against volumetric DDoS attacks. Paper presented at: Proceedings of the 34th Annual Computer Security Applications Conference; 2018:266-277; ACM.
38. Mortensen A, Reddy T. R. Moskowitz, DDoS Open Threat Signaling (DOTS) Requirements. tech. rep., RFC 8612, May 2019. doi https://doi.org/10.17487/RFC8612.
39. Mosharraf N, Jayasumana AP, Ray I, Bezawada B. History-based throttling of distributed denial-of-service attacks. In: Obaidat MS, Cabello E, eds. *E-Business and Telecommunications*. Cham: Springer International Publishing; 2019:199-223.
40. Yaar A, Perrig A, Song D. StackPi: new packet marking and filtering mechanisms for DDoS and IP spoofing defense. *IEEE J Select Areas Commun*. 2006;24(10):1853-1863.
41. Lee YJ, Baik NK, Kim C, Yang CN. Study of detection method for spoofed IP against DDoS attacks. *Pers Ubiquit Comput*. 2018;22(1):35-44.

42. Jin C, Wang H, Shin KG. Hop-count filtering: an effective defense against spoofed DDoS traffic. Paper presented at: Proceedings of the 10th ACM Conference on Computer and Communications Security; 2003:30-41; ACM.
43. Mosharraf N, Jayasumana AP, Ray I. Using a history-based profile to detect and respond to DDoS attacks. Paper presented at: Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Vol 4: SECRYPT; July 24-26, 2017:175-186; Madrid, Spain.
44. Patgiri R, Nayak S, Borgohain SK. Preventing DDoS using bloom filter: a survey. *EAI Endors Trans Scal Inform Syst*. 2018;5(19):1-9. https://doi.org/10.4108/eai.19-6-2018.155865.
45. Jung J, Krishnamurthy B, Rabinovich M. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. Paper presented at: Proceedings of the 11th International Conference on World Wide Web; 2002:293-304; ACM.
46. Bhandari A, Sangal AL, Kumar K. Characterizing flash events and distributed denial-of-service attacks: an empirical investigation. *Secur Commun Netw*. 2016;9(13):2222-2239.
47. Jian Y. An improved intrusion detection algorithm based on DBSCAN. *Microcomput Inform*. 2009;25(13):58-60.
48. Xue-Yong L, Guo-hong G, Jia-xia S. A new intrusion detection method based on improved DBSCAN. Paper presented at: Proceedings of the Information Engineering (ICIE), 2010 WASE International Conference on 2. IEEE; 2010:117-120.
49. Wilson DR, Martinez TR. Improved heterogeneous distance functions. *J Artif Intell Res*. 1997;6:1-34.
50. Liu D, Lung CH, Seddigh N, Nandy B. Network traffic anomaly detection using adaptive density-based fuzzy clustering. Paper presented at: Proceedings of the Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on IEEE; 2014:823-830.
51. Coull SE, Wright CV, Monrose F, Collins MP, Reiter MK. Playing Devil's advocate: inferring sensitive information from anonymized network traces. *NDSS*. Vol 7; San Diego, CA: Internet Society; 2007:35-47.
52. Shmueli G, Bruce PC, Patel NR, Yahav I, Lichtendahl KC Jr. *Data Mining for Business Analytics: Concepts, Techniques, and Applications in R*. New York, NY: John Wiley & Sons; 2017.
53. Verma K, Hasbullah H, Kumar A. An efficient defense method against UDP spoofed flooding traffic of denial of service (DoS) attacks in VANET. Paper presented at: Advance Computing Conference (IACC), 2013 IEEE 3rd International, IEEE; 2013:550-555.
54. Rashidi B, Fung C, Rahman M. A scalable and flexible DDoS mitigation system using network function virtualization. Paper presented at: Proceedings of the NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium; 2018:1-6; IEEE.
55. Torleiv M. *Communication and Information theory*; 2017:97-106.
56. Broder A, Mitzenmacher M. Network applications of bloom filters: a survey. *Internet Math*. 2004;1(4):485-509.
57. Dagon D, Zou CC, Lee W. Modeling botnet propagation using time zones. *NDSS*. Vol 6; San Diego, CA: The Internet Society; 2006:2-13.
58. Abu Rajab M, Zarfoss J, Monrose F, Terzis A. A multifaceted approach to understanding the botnet phenomenon. Paper presented at: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement; 2006:41-52; ACM.
59. Goldstein M. BoNeSi. 2016. https://github.com/Markus-Go/bonesi.
60. Goldstein M. Apache traffic replay generator. 2016. https://github.com/Markus-Go/bonesi.
61. Münz G, Li S, Carle G. Traffic anomaly detection using k-means clustering. In: *GI/ITG Workshop MMBnet*; Bamberg, Germany: University of Bamberg Press; 2007.

How to cite this article: Kongshavn M, Haugerud H, Yazidi A, Maseng T, Hammer H. Mitigating DDoS using weight-based geographical clustering. *Concurrency Computat: Pract Exper*. 2020;32:e5679. https://doi.org/10.1002/cpe.5679