Data Mining Approaches for IP Address Clustering

Madeleine Victoria Kongshavn, Anis Yazidi(✉), Hårek Haugerud, and Hugo Hammer

Department of Computer Science,
Oslo and Akershus University College of Applied Sciences, Oslo, Norway
Anis.Yazidi@hioa.no

Abstract. 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 masks the attack traffic among legitimate users. Mitigation approaches within DDoS has therefore often been investigated within the field of anomaly intrusion detection. This means that even a successful mitigation approach will risk a high disregard of legitimate users. This article proposes to use data mining and machine learning approaches to find unique hidden data structures which keep a high degree of accepted legitimate traffic, while still being able to remove illegitimate and irrelevant data traffic which don't follow the hidden structure. In this perspective, we devise and evaluate two novel IP Address clustering algorithms for DDoS mitigation, namely, Geographical Clustering (GC) and Reduced Geographical Clustering (RGC).

1 Introduction

A Denial of Service (DoS) attack is an explicit attempt to render a server incapable of providing normal service to its users. This can be accomplished through a variety of methods, however, the common pattern is that a DoS attack will try to exploit some limited system or network resources to accomplish this goal. We can differentiate between DoS attacks and DDoS attacks; A DoS is where one attacker, with one network connection, executes an attack, while a DDoS attack adds a many-to-one dimension problem to the equation. Instead of using one connection, a DDoS attack often uses thousands of compromised hosts to execute an attack. This amplifies both the available resources of an attacker and the complexity of the DoS problem. DDoS attacks have over the last decade become an immense threat to the Internet infrastructure where attacks have become commonplace with a wide range of global victims in everything from commercial websites and educational institutions to public chat servers and government organizations [1]. Akamai, which is a leading network services provider for media and software delivery as well as for delivering cloud security solutions, reported a 125% increase in total DDoS attacks from 2015 to 2016 [2].

Y. Chen and T. Q. Duong (Eds.): INISCOM 2017, LNICST 221, pp. 266–275, 2018.
https://doi.org/10.1007/978-3-319-74176-5_23

In a general sense, it is preferable to mitigate DDoS attacks as close to the source as possible, in order to reduce as much of the collateral damage as possible. However, this is generally difficult for a number of reasons; it's generally hard to identify attack traffic close to the source, as the attack traffic is highly distributed and comes from multiple sources. Moreover, the Internet is built around an authorization-free nature which makes it difficult to implement any scheme to a high degree. It is therefore a strong need to mitigate DDoS attacks near the target victim, as this seems to be the only solution for the current Internet infrastructure [3]. This article proposes two clustering algorithms to mitigate DDoS attack. The algorithms are tested on an anonymous dataset gathered from a web-server in Oslo and Akershus University College of Applied sciences. The dataset is henceforth denoted by D10C and consists of a training phase and a testing phase. The training phase consists of 1 million unique requests and the testing set consists of 500 thousands unique requests. The proposed algorithms build on the following research question that we shall explore in this article:

How can we use data mining to find pattern correlations on data history to build efficient filtering rules that are able to mitigate DDoS attacks near the targeted server?

2 Related Research

In this section, some prominent closely related research is reviewed which deals directly or indirectly with filtering DDoS attacks based on elements from data mining.

Peng et al. suggested to use a database of previously seen legitimate IP addresses to counter DDoS attacks. This method bases itself on the common assumption that normal network traffic differentiates itself from network traffic under an attack [4]. This mechanism, known as History-based IP filtering(HIF), uses an IP address database (IAD) containing previously seen IP addresses over a certain time period. Under an attack, only IP addresses from the IAD are allowed to access the network or service.

Goldstein et al. [5] suggested taking advantage of IP neighborhood relations, by using density estimation, to counter DDoS attacks. The idea is that IP addresses that are close or similar to each other, share similar characteristics. To evaluate the distances between the different objects, $Xor+$ and Euclidean distance is used on the IP address. To avoid that distances within the given network mask are always constant, regardless of variation within the subnet, Euclidean distances are added to the distance calculation. The approach used a modified K-means clustering algorithm to compute the clustering centers. Once the clustering centers were computed, an area was defined around the clusters, which would be the IP addresses the model expected to see in the future [5].

3 Proposed IP Address Clustering Algorithms

Previous clustering techniques, as seen in the last section, focus on clustering data based on existing data features in the traffic pattern. These features include packet attributes as source addresses, time to live (TTL), flags and protocols [5–12]. Clustering based on IP address has not been well investigated in the literature and is shown to be a very promising step towards mitigating DDoS. At this juncture, we shall present two novel IP address clustering algorithms; Geographical Clustering (GC) and Reduced Geographical Clustering (RGC). Instead of focusing the initial clustering approach on the distance between existing features, the clustering approaches try to expand clusters based on the distance between geographical points where traffic are located. GC tries to expand a cluster based on frequent geographical points. Since GC relays on a packets location repeating, this can amplify the amount of data that can be covered as GC can identify new networks which has not necessarily been seen under the training phase. RGC further expands on this idea, by limiting the amount of accepted location points. This limitation is done by employing different constraints on the cluster expansion by only accepting the most legitimate points in RGC. The amount of accepted networks can be limited without affecting the amount of accepted data from the testing phase.

3.1 GC

The first proposed density-based clustering algorithm, *GC*, tries to estimate a data pattern based on geographical location and builds on the hypothesis: *If network x from location y reaches the server under the training phase, there is a higher chance for network z from location y + 1 to reach the server, than network q that doesn't belong to a location close to network x.*

The clustering algorithm, *GC*, which is based on *DBSCAN* [13], tries to expand a cluster as long as there are points close to the cluster. *GC* begins by defining core points from where a cluster can continue to expand. Defining start points is a computationally hard task, as the definition of the start points will essentially define how and where the cluster will expand and how well the clustering algorithm will be able to represent new and unknown data. We can make the assumption that frequent networks often in some form reoccur in the data pattern. Since frequent networks represent a huge portion of the incoming traffic, it is reasonable to assume that every frequent network, or in this case location, is automatically assumed as a start point or core point. The core points will then look in their close proximity based on distance x to see if there are more points that can belong to this cluster. If a core point finds a point y that is within distance x, it will include the point to its own cluster. Point y will then look in its close proximity to find any new points that can be included in the cluster. If any points that are part of a cluster are within the given distance to other points that are part of a different cluster, the two clusters will merge and become one cluster. The pseudo code for density-based clustering can be seen in Algorithm 1 below. The algorithm takes three arguments: the dataset containing a list of latitude/longitude locations with the frequency of packets from that locations, D which is the maximum distance from a point to a cluster for this point to still be considered a part of that cluster and T which is the frequency threshold for a point to be considered a core point. According to this algorithm,

Algorithm 1. Pseudo code for the GC.

```
1: function DENSITY_CLUSTERING(dataset, D, T)
2:     all_clusters ← (dataset.points ≥ T)
3:     for cluster in all_cluster 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_cluster
13: end function
```

two essential parameters, core-point threshold and maximum distance, need to be pre-defined. The definition of these attributes varies greatly between different datasets and consequently the data pattern that these clusters cover, will have different strengths and weaknesses based on the chosen attributes.

Deciding Core-Point Threshold

As aforementioned, the definition of frequent locations, or core points, is just the definition of an initial cluster, which is just a single start point for a cluster to continue to expand. This threshold would vary greatly between different datasets and with a new and unknown dataset, it could be beneficial to decide these location points based on an upper percentile of all points which have the highest frequency of seen traffic. The definition of the threshold for a core-point is beyond the scope of this paper. However, as seen in Fig. 1 more initial clusters will exist with a low threshold and this is not necessarily preferable as this will lead to clusters where there are seen considerable little data traffic. Moreover, as it is the location points we are talking about, considering if we are able to identify new locations correctly, we might risk to cover a substantial location pattern, and therefore start accepting traffic which is not necessarily representative for the known data pattern. Therefore, when choosing the initial cluster points, the amount of covered training data should be seen in correspondence with the amount of initial clusters. Since we want clusters to expand based on the initial clusters, we don't need to cover most of the training pattern from the beginning, as we assume, most of the relevant data pattern, will still be covered, when the cluster expand to points nearby. Therefore, we should preferably choose a high core-point threshold which will cover the most relevant locations. These location points will have a higher possibility of repeating later, and we should be able to make the assumption that areas near these points will also have a high chance of repeating, although little traffic might be seen in the area during the training phase.

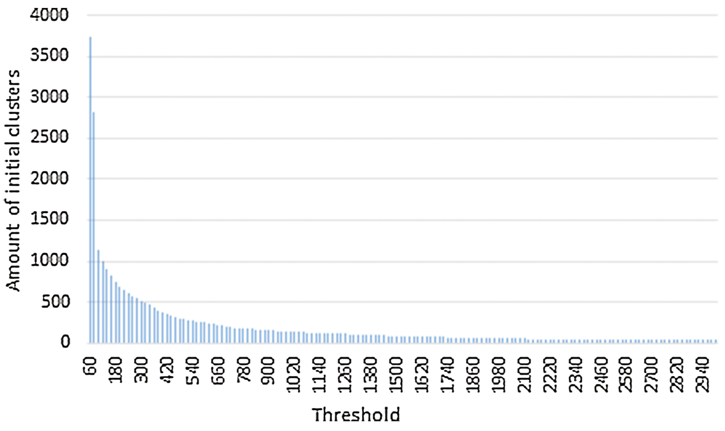

Fig. 1. The amount of initial clusters for D10C with different thresholds.

Deciding Maximum Distance

As we are assuming, from the hypothesis, a lot of lower networks or locations with less frequent packets will still be covered as they exist in the surrounding area of the defined core points. This leads to the question; what is the maximum distance from point x to point y, for point x to still be a part of point *y's* cluster? We have previously stated that every core point is automatically assumed to be cluster, the question now is; What is the maximum distance for one or more cluster points to any new point, for the point to be a part of the given cluster? The answer to this question would vary widely based on the training data. However, we should try to minimize the maximum distances, so we don't end up with clusters where data have little to none similarities with other data in the same cluster. Based on our hypothesis, we can assume that data points exist close to our cluster in all directions of the initial core points. Therefore, we can based on this, get a certain view of the necessary distance, by measuring how much training pattern is covered with different maximum distances.

Figure 2 shows the amount of data for *D10C*, which is covered with different thresholds and maximum distances. A lower threshold will cover more of the training data, regardless of the maximum distance and at most, with a threshold of 700, *81.56%* of D10C training data is covered with a maximum allowed distances of *50* km. It would be preferable to cover most of the training as this will result in a higher coverage for the test data. However, the coverage of new points should, to a large degree, be proportional to the higher maximum distance. In other words, we should be able to assume that a cluster does not expand forever.

Figure 3 gives a clearer view of the difference in increased percentage points for each threshold. Most points have a good increase in the amount of covered data until around 20–25 km. At this point, the different calculated clusters have managed to increase around 10% points. At the remaining calculated distances of 30–50 km, clusters have a high variance, with no particular pattern for the

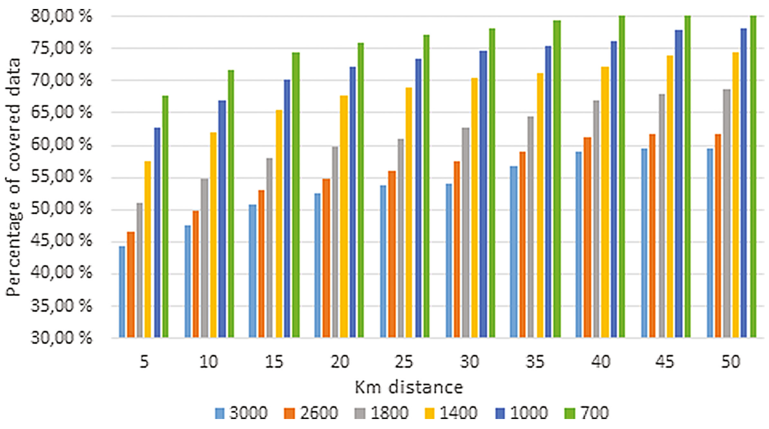

Fig. 2. The amount of data, for D10C, which is covered with various initial cluster sizes and with different maximum distances.

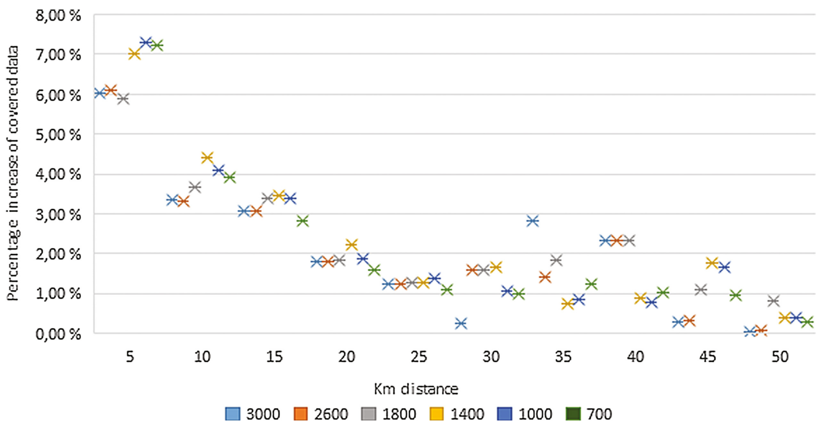

Fig. 3. The increase in % points that is covered with different km threshold and core points threshold, for D10C.

remaining increased percentage points. It would therefore, for most clusters, be beneficial to have a maximum km distance between 20–25 km.

GC Result

To find the optimal result for geographical clusters, data or source IP addresses, which has not necessarily been seen in the training phase, but is part of a cluster based on the geographical locations should be counted as well. We should therefore, not only count the direct points that have appeared in the training phase, but also count points that fit inside a cluster, but which has not necessarily been seen in the training phase. *GC* manages with an optimal result, as seen in Fig. 4 where a packet location can be identified, to achieve from 40% to 77%.

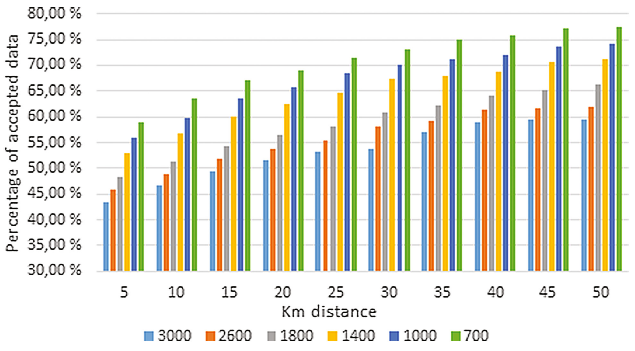

Fig. 4. The optimal result of accepted packets for the D10C testing set.

For D10C, when only 24-bit networks which have previously been seen in the training phase and are a part of a cluster are accepted, the amount of accepted data result diminishes radically, compared to an optimal result where a packets location can be identified. At the minimum, when new data can not be identified, 7.50% points less are accepted in the testing phase for D10C. At the highest point, with a cluster computation of 50 km and a threshold of 700, 27.02% points less are accepted in the testing phase.

3.2 RGC

Although, *GC* has a potential of acquiring an unique pattern and performing well under unknown traffic, the algorithm does not define any consideration when defining a pattern. The previous algorithm, which states that if a point x is near one or more points that are a part of cluster y, point x should also be stated as a part of y*'s* cluster, is not necessarily ideal, as multiple issues are raised in regards to the algorithm's ability to sustain a pattern without accepting too much illegitimate traffic.

To create a stronger algorithm, several steps can be taken. First, to prevent the pattern composition in clusters from deteriorating when clusters are merged, merging of clusters can be overlooked. Instead, clusters can be forced to expand and converge separately. Assume that $x_1, x_2, ..., x_{n-1}, x_n$ is a set of data points in a two dimensional space and $c_1, c_2, ..., c_{n-1}, c_n$ is a set of core-points which contains at least the amount y of seen requests. Let $d(x_m, c_z)$ be the geographical distance between the point x_m and the core point c_z. Potentially x_m could be part of any of the clusters with core point c_n. The algorithm decides that x_m will be a part of cluster c_z if $d(x_m, c_z) < d(x_m, c_1), ..., d(x_m, c_{n-1}), d(x_m, c_n)$ for all core points. Moreover, clusters should only expand to points that are more likely to occur later. We can build on the hypothesis and make the assumption that a request is more likely to occur in a point x if it has at least y points in its near area. Only points that satisfy this criterion, are allowed to be a part of a cluster. Finally, to prevent big cities, which have a natural ability to easier create core points, from being able to create a cluster if there are not other points nearby, clusters that are not able to expand over a certain amount of points should be dismantled. The dismantled points, including the core point, should then be re-positioned to clusters nearby. As seen in Fig. 5, different minimum applied lengths, in a D10C cluster, radically and rapidly diminishes the amount of accepted data. Both the amount of covered data in the training and testing phase decrease fast with higher minimum cluster length. However, the amount of covered data in the training phase decreases faster than the amount of covered data in the testing phase. At a minimum length of 10, the amount of covered data in D10C training set decreases below the amount of covered data in the testing set.

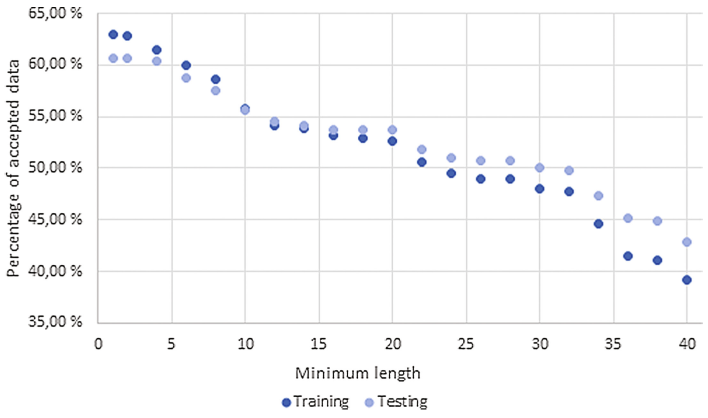

Fig. 5. The amount of covered data in the training and testing phase with different minimum lengths constraints set for a cluster.

RGC Result

When clusters are computed with a minimum length of 10, minimum points of 3, different km distances and different core-point thresholds, the clusters will (for D10C) in most instances cover 40% to 50%. This calculation can be seen in Fig. 6. These calculations give a lower acceptance rate than for GC. However, while GC manages to have a zero percentage differences between the accepted data in the training and testing phase, RGC manages to accept upwards of 30% points more in the testing phase than in the training phase. This implies that RGC is to a higher degree able to identify the most relevant points that have a high likelihood of occurring later.

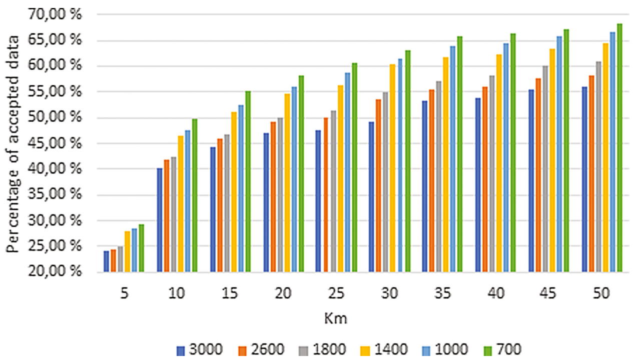

Fig. 6. The optimal amount of accepted D10C testing data that was seen in a cluster.

4 Conclusion

Previous clustering techniques focused on clustering data based on existing data features in the traffic pattern. These features included packet attributes as source addresses, TTL, flags and protocols. Clustering on IP address has not been well investigated in the literature and is in this paper shown to be a very promising step towards mitigating DDoS. Clustering IP addresses based on distances between bits gives a limited clustering ability when addresses are close geographically, but not necessarily close in the address range. Two novel IP address clustering algorithms were devised and evaluated. GC expanded a cluster based on frequent geographical points. This gave GC clustering an advantage over previous clustering approaches. As *GC* relayed on a packets locations repeating, the amount of accepted legitimate traffic was amplified. GC was shown to be able to identify new networks which had not necessarily been seen under the training phase. RGC further expanded on this hypothesis, by limiting the amount of accepted location points.

References

1. Moore, D., Shannon, C., Brown, D.J., Voelker, G.M., Savage, S.: Inferring internet denial-of-service activity. ACM Trans. Comput. Syst. (TOCS) **24**(2), 115–139 (2006)
2. Akamai: Akamai's state of the internet report q1 2016 (2016)
3. Goldstein, M., Lampert, C., Reif, M., Stahl, A., Breuel, T.: Bayes optimal ddos mitigation by adaptive history-based IP filtering. In: Seventh International Conference on Networking (ICN 2008), pp. 174–179. IEEE (2008)
4. Peng, T., Leckie, C., Ramamohanarao, K.: Protection from distributed denial of service attacks using history-based IP filtering. In: IEEE International Conference on Communications (ICC 2003), vol. 1, pp. 482–486. IEEE (2003)
5. Goldstein, M., Reif, M., Stahl, A., Breuel, T.: Server-side prediction of source IP addresses using density estimation. In: International Conference on Availability, Reliability and Security, 2009 (ARES 2009), pp. 82–89. IEEE (2009)

6. Qin, X., Xu, T., Wang, C.: DDoS attack detection using flow entropy and clustering technique. In: 2015 11th International Conference on Computational Intelligence and Security (CIS), pp. 412–415. IEEE (2015)
7. Yu, J., Li, Z., Chen, H., Chen, X.: A detection and offense mechanism to defend against application layer DDoS attacks. In: Third International Conference on Networking and Services (ICNS), p. 54. IEEE (2007)
8. Li, Z., Li, Y., Xu, L.: Anomaly intrusion detection method based on k-means clustering algorithm with particle swarm optimization. In: 2011 International Conference on Information Technology, Computer Engineering and Management Sciences (ICM), vol. 2, pp. 157–161. IEEE (2011)
9. Mingqiang, Z., Hui, H., Qian, W.: A graph-based clustering algorithm for anomaly intrusion detection. In: 2012 7th International Conference on Computer Science & Education (ICCSE), pp. 1311–1314. IEEE (2012)
10. Ranjan, R., Sahoo, G.: A new clustering approach for anomaly intrusion detection, arXiv preprint arXiv:1404.2772 (2014)
11. Guan, Y., Ghorbani, A.A., Belacel, N.: Y-means: a clustering method for intrusion detection. In: Canadian Conference on Electrical and Computer Engineering (CCECE), vol. 2, pp. 1083–1086. IEEE (2003)
12. Xue-Yong, L., Guo-hong, G., Jia-xia, S.: A new intrusion detection method based on improved DBSCAN. In: 2010 WASE International Conference on Information Engineering (ICIE), vol. 2, pp. 117–120. IEEE (2010)
13. Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Kdd*, vol. 96, no. 34, pp. 226–231 (1996)