Fireeye

# DGA detection

Vikramaditya Singh

10-07-2020

# Contents

# 1. Summary

[*The goal of the project is to detect the use of Domain Generating Algorithm in the network*]

# 2. Introduction

Adversaries have stopped using hard-coded domain lists and IP addresses, which are useless once blocked DGAs by comparison is easy to implement, difficult to block, and maybe impossible to predict in advance and can be quickly modified if the previously used algorithm becomes known.

A Domain Generating Algorithm (DGA) is a program or subroutine that provides malware with new domains on-demand or on the fly. Each bot of a given botnet is shipped with the same DGA.

Advantages of DGA botnet is that if the domain of C&C server is detected and blocked all connections to these addresses, the botnet has not been eliminated completely. Botmaster just registers a new address in domain dataset and the bot will still operate as normal. This strategy improves the robustness of botnet even though one or more CC servers are located and taken down, the bots will finally get the relocated CC server via DNS queries to the next set of automatically generated domains. There are some well-known DGA botnets such as Conficker, Kraken, etc, each of them has its own domain generation algorithm for a list of candidate CC domain.

## 2.1 Communication with C&C

To contact the botmaster, each bot periodically executes a domain generation algorithm (DGA) that, given a random seed (e.g., the current date), produces a list of candidate CC domains. The bot then attempts to resolve these domain names by sending DNS queries until one of the domains resolves to the IP address of a C&C server.

The key concept is that, at any moment, only a small number of domains is active and resolves to the true IPs of the C&C server. This characteristic makes dealing with DGAs very expensive for the security defenders because they would need to track—and register Only the DGA authors (or the botmasters) know exactly when the upcoming rendezvous domain has to be registered and activated. Only the DGA authors (or the botmasters) know exactly when the upcoming rendezvous domain has to be registered and activated.

## 2.2 Other Detection Approaches

1. DGAs can be determined from captured malware code, we can generate the domains ahead of time and still identify and block botnet C&C traffic. The existing solutions are largely based on reverse engineering of the bot malware executable, which is not always feasible.
2. A blacklist of known command and control (C&C) domains to detect bots and block their traffic, such a botnet detection approach is static because the blacklist is updated only after running an external (and often manual) process of domain discovery.

3. Traffic analysis, DNS traffic logs analysis, statistical analysis, and so on

4. Look for clusters of similar domains being queried by multiple clients. This only catches infections in their later stages, when they have spread to several clients.

5. Some deep learning approaches have been identified using LSTM and adversarial neural net for detection of the malicious domains.

## 2.3 DGA & its types

Even when a certain DGA is known (for example, by reverse engineering a malware sample), it's still difficult—or even impossible—to effectively block it.

(a) First, there is the sheer number of possible domains that can be generated. Gameover Zeus, for example, generates 1,000 domains every day. This amounts to 365,000 domains that need to be generated in advance and blocked, which would strain on firewalls and other network-filtering solutions.

A DGA typically has three components:

(a) A time-sensitive "seed"
(b) A domain "body" generator that uses this seed
(c) A set of top-level domains (TLDs)

The DGA can be basically categorised in two broad types.

(a) Word-based : The difficulty of detecting this simple algorithm is that the domains do not seem to be randomly generated and the commonly used words may appear in many legitimate domain names.
(b) Randomly generated : Our approach targets these type of AGD.

## 2.4 Our Approach

We tested our classifying models with the dataset containing more than 30,000 domain, including domain generated by the DGA botnets like Conficker, Tinba, Bebloh, Tovar, Goz, Kraken and 100.000 domains with a high ranking from Alexa. The proposed method can detect up to 92-98 per cent of Algorithmically generated domain and distinguish 80 per cent of DGA correctly.

# 3. Considerations for detection of AGD

## 3.1 Requirements

1. DGA domains and Benign domain
2. Proper filtering out of illegal data
3. Pre-processing of the data
4. Feature set to represent a domain
5. Distance Measure and other parameter to be used in the models.
6. Clustering algorithm
7. Classifier algorithm

## 3.2 Open Question

1. How to deal with dictionary based DGA.
2. Overlapping features for different DGA.
3. Feature to distinguish among different AGD.
4. Use of host addresses while clustering.
5. Subset of feature's to be used in classifiers.
6. Testing in live environment
7. Misspelling checker needs to be tested.
8. Use of small cluster to form a final cluster.

# 4. Architecture

The machine learning framework consists of the following four main components:

1. The pattern filter is used to filter the NxDomains queries in order to obtain the domains from them and check if they were due to a human typo.
2. A feature extractor extracts feature from the incoming domains that are not in the blacklist. Those domains will be processed in the next component.
3. A two-level machine learning model, the first-level clustering and the second-level classification. We apply the clustering method to group domains sequenced by the DGA and then try classifying these groups to known DGA those that do not fit any known DGA model are (automatically) assigned a New-DGA- vX label, where X is a unique identifier.
4. A prediction model to identify DGA domains, we first use various classification models to classify DGA domains and normal domains.

## 4.1 Passive Module

The DNS is a crucial component as whenever a host tries to communicate with C&C server by sending queries, it is responsible for the translation of domain name to its corresponding IP address.

The architecture for detection of DGA based botnet consists of two modules, the goals of our method are to analyse NXDomain which is generated by DGA botnet, so all benign domains have to be removed from the collection.

1. Filtering & Feature Extraction module
2. Bot Detection module

The brief description of these modules, their components and usage for C&C bot detection is discussed.

1. Each of the compromised assets will generate several DNS queries resulting in NXDomains, and a subset of these NXDomains will likely be queried by more than one compromised machine.
2. We first automatically identify and filter out "accidental", user-generated NXDomains due to typos or misconfigurations.
3. Then we find cluster's of NXDomains, each cluster corresponds to a type of DGA botnet.
4. The clustering algorithm clusters domains based on the similarity in the make-ups of domain names and can be extended as well to use the groups of machines that queried these domains.

5. Followed by using statistical learning techniques to build a model of the DGA, these clusters are used to detect future compromised machines running the same DGA.
6. **Instances of AGD names drawn from the same DGA can be generalized into "fingerprint" of the generation algorithm itself**, without reversing its implementation.
7. If a cluster cannot be assigned to a known model, then a new model is produced, indicating a new DGA variant or family.

Botnet access behaviour of the domain name has a strong regularity, which does not follow the change of DGA algorithm. By using a DGA algorithm, malware can use different domains to connect back to the C&C server. In this way, a large number of failure domain names will be generated, whose active time is more concentrated. They can be collected in the access behaviour of the client and DNS traffic flow..

## 4.2 Active Module

1. On the first phase, the NXDomains will be filtered using blacklist & white list.
2. To identify compromised hosts, we collect the domains generated by a host, hi, and we ask the AGD Classifier whether it likely "belongs" to a DGA or not.
3. If the answer is yes, we ask the DGA Classifier whether it likely "belongs" to a previously seen DGA or not.
4. If there is a high probable match and this repeats certain number of times for a host then the host is considered to be compromised and will be labeled with the name of the (suspected) DGA bot that it is running.
5. Otherwise, the domain resolution is allowed based on the assumption the passive mode will detect them again if it is from DGA due to high number of NXDomain generated by a DGA which will form a cluster due to similar semantic features.

# 5. Feature Template

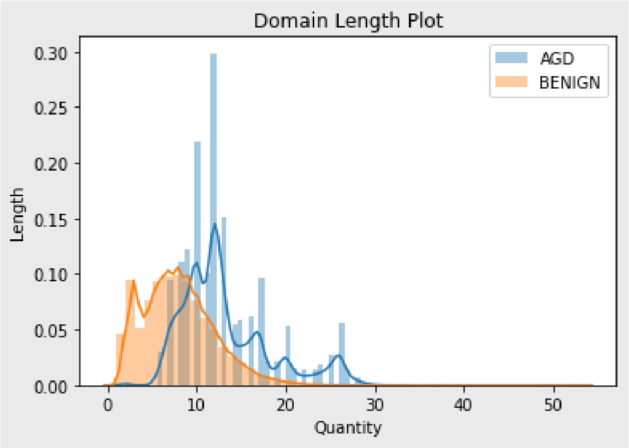
Differences between the white list and the blacklist were analyzed to build a feature template from the angle of lexicology and linguistics.

DGA domain name is commonly composed of letter, number and symbol by random combinations, which is different from the character probability distribution of legal domain names
We designed the following feature template for RF:

1. Length of domain
2. Vowel Count to Domain Length Ratio
3. Digit Count to Domain Length Ratio
4. Repeat Character Count To Domain Len Ratio
5. Entropy of domain
6. N-gram score in white list
7. N-gram score in dictionary
8. Meaningful Characters To Domain Len Ratio
9. Length Of Longest Meaningful Substring To Domain Length
10. Consecutive Characters To Domain Length Ratio

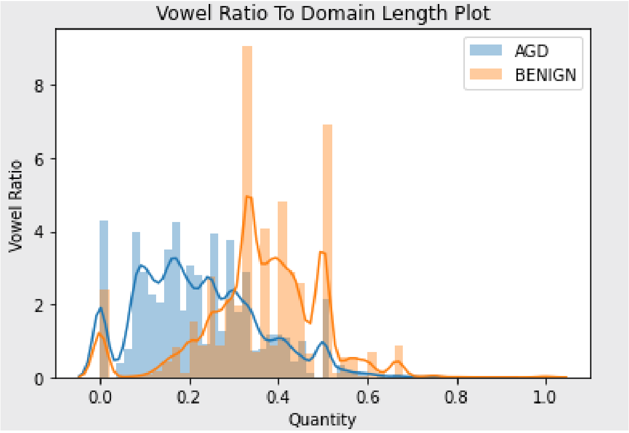
## 5.1 Length

We calculated the length of each second-level domain. ( eg: google.com - 6 ). The AGD domains are longer in length than that of benign domains because the DGA algorithms generate the malicious domains randomly in abundance so the average length of the domain name is large as compared to benign domains.



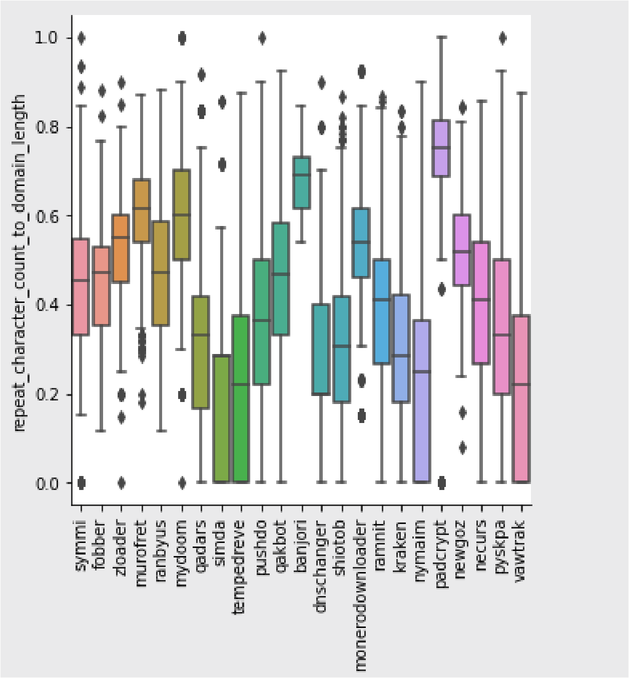
## 5.2 Vowel Count to Domain Len Ratio

Vowel ratio is obtained by frequency calculation of vowel letter "a", "e", "i", "o", "u".



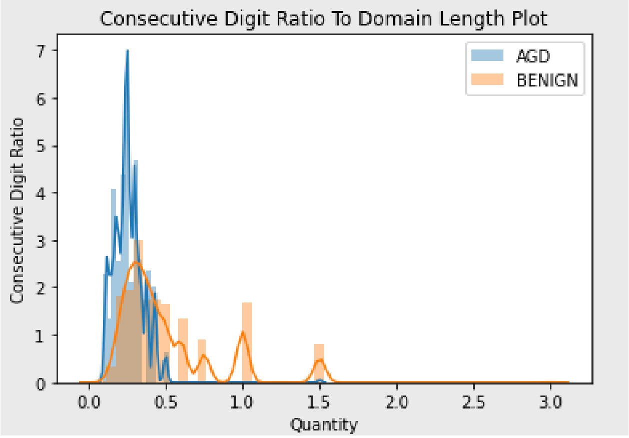
## 5.3 Repeat Character Count To Domain Length Ratio

The repetition of more than three consecutive letters is not observed in English vocabulary as well as legitimate domain names. However, the DGA generated domains tend to contain alphabetic sequences with more repeated characters.



## 5.4 Consecutive Characters To Domain Len Ratio

The value of two consecutive characters being same.

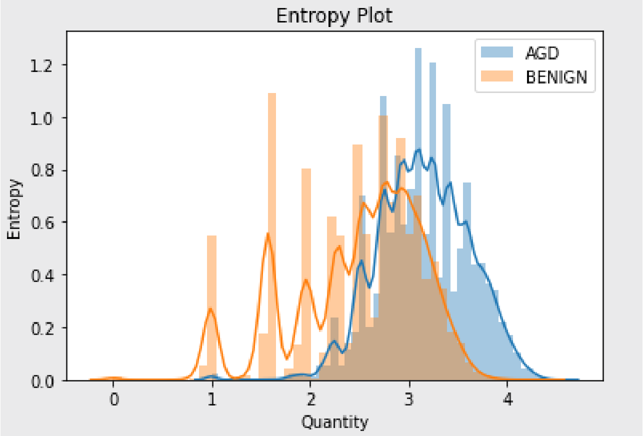


## 5.5 Entropy

We use entropy to calculate the chaotic stage of the distribution in AGD domain names in order to distinguish them from the normal ones.

### 5.5.1 Shanon Entropy

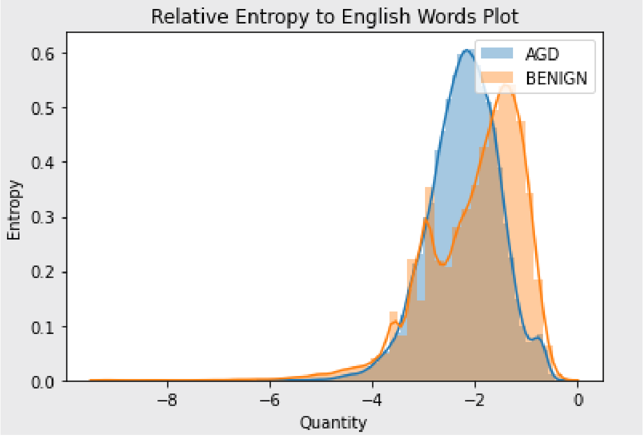
In information theory, systems consist of a transmitter, channel, and receiver. The transmitter produces messages that are sent through the channel. The channel modifies the message in some way. The receiver attempts to infer which message was sent. In this context, entropy (more specifically, Shannon entropy) is the expected value (average) of the information contained in each message. This feature computes the entropy of character distribution and measures the randomness of each domain names.



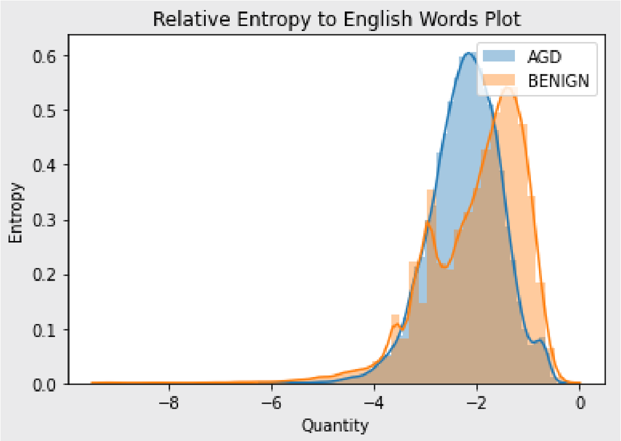
### 5.5.2 Relative Entropy

Relative entropy compares two distributions, so to perform the calculations some baseline data needs to be collected this requires additional work, it improves the fidelity of the randomness detection. We do two comparisons:

1. With Benign domain distribution



2. With English word distribution

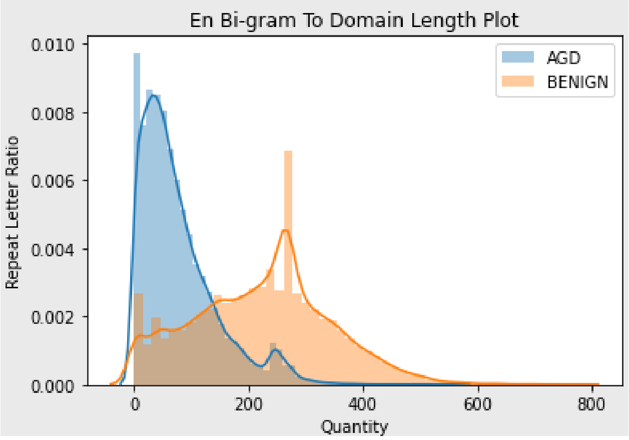


## 5.6 N-Gram Features

We noticed that DGA domain either uses some random characters as text string or uses a dictionary to make up a new text string. Therefore, we build up our own corpus for these features.

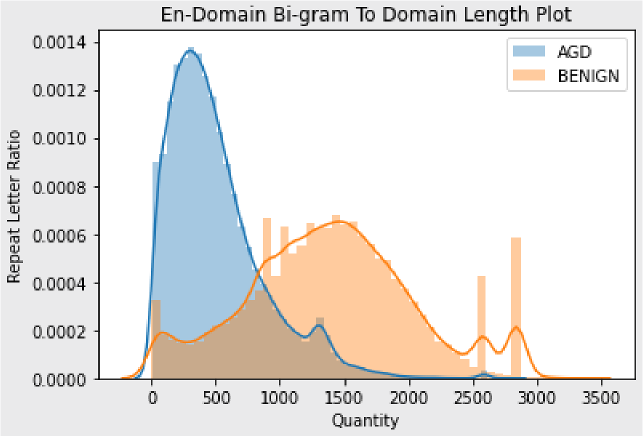
### 5.6.1 Benign grams

If a domain is a legit domain, it more likely exists in the Alexa ranking list. Thus, it is necessary to find the similarity of legit domains. We could use some text analysis techniques. The first step is to build up a legit text corpus. Given a subsequence of domains, we summarize the frequency distribution of N-gram among the Alexa domain name string with n = [2,3,4]. We called it benign grams.



### 5.6.2 Dictionary grams

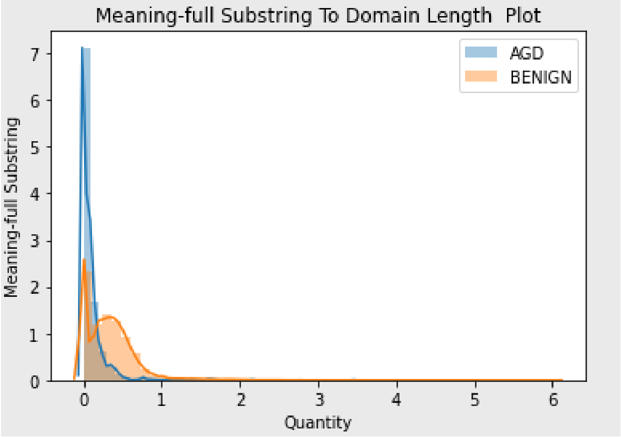
We used a dictionary that contains 7,000 common used word terms. The terms are a combination of English vocabulary and commonly used words with a mix of number and alphabet. We will use a words dictionary. After basic cleaning up work, we calculated the dictionary gram using N-gram, n = [3,2,4].



We use these N-grams to create a score for the domain

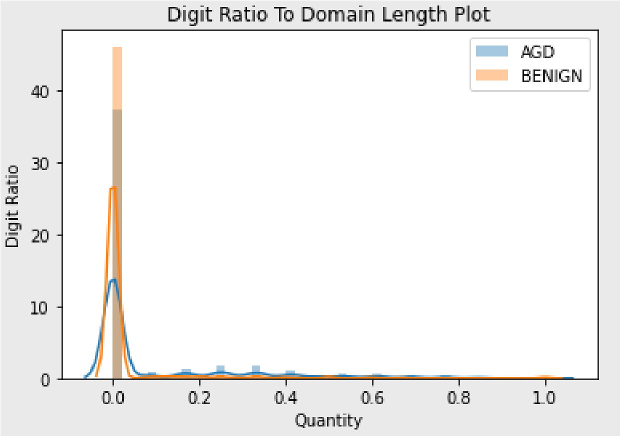
## 5.7 Meaningful Word Ratio

This feature measures the ratio of meaningful words in a string (domain name). The ratio is calculated as follows: where wi is the i-th meaningful substring of this string, |wi| is the length of i-th meaningful substring. Since DGA domain names usually contain meaningless words; therefore, a small value of a ratio usually means that the domain could be a DGA domain name and a higher ratio implies a safer domain name.



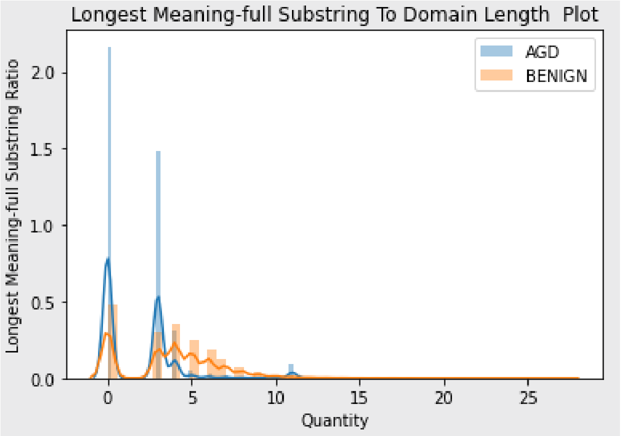
## 5.8 Ratio of numerical character to domain length

This feature measures the percentage of numerical characters in a string.



## 5.9 Longest Meaningful string

This feature is to measure the length of the longest meaningful string in a domain name.



# 6. Components

## 6.1 Classifier

Classification in machine learning would help in DGA domains detection. The purpose of building a DGA classifier is not to take down botnets, but to discover and detect the use on our network or services.

1. When we have a list of domains resolved and accessed at one's organization, it is possible to see which of those are potentially generated and used by malware. (Classifier I)
2. Every time a new DGA is discovered, we use a supervised learning approach to build models of what the domains generated by this new DGA "look like". (Classifier II )

It is common to use several supervised approaches to identify domains here we will use Random Forest classifier. Thus, the first step in any classifier is getting enough labelled training data. We need two types of data for our two classifiers.

1. A list of legitimate domains and a list of domains generated by an algorithm.
2. A list of labelled algorithmically generated domain labelled to which DGA they belong.

### 6.1.1 Random Forest Algorithm

Random Forest Algorithm is one of the most popular methods or frameworks used by data scientists. It is an ensemble learning method for classification and regression that construct a number of decision trees at training time and outputting the class that is the mode of the classes output by an individual tree. The specific process is as follows:

1. In each round, RF extracts samples with replacement from the original training. The training set's size is M. In this proposed process, the probability that each sample will be selected is the same and independent. After Q round, the RF algorithm gets a set of samples: S1, S2.. SQ.
2. We assume that feature space size is E.In each round, RF selects a size of e(e<E) feature randomly for building a decision tree with the corresponding sample S. Each decision tree is independent as well.
3. Because of the independence between Q decision trees, RF will not take tree weight into account when votes for final prediction.

In RF, the features are randomly selected in each decision split. The correlation between trees is reduced by randomly selecting the features which improves the prediction power and results in higher efficiency. As such the advantages of RF are shown as follows:

1. In training data, it is less sensitive to outlier data.

2. It can balance error in class population unbalanced data sets.
3. In training data, it is less sensitive to outlier data.
4. It can balance error in class population unbalanced data sets.
5. It estimates the importance of features in determining classification. RF provides an experimental way to detect feature interaction.
6. It handles a very large number of input features, without the problem of overfitting.

In more detail, RF is appropriate for high dimensional data modelling, because it can handle missing values and can handle continuous, categorical and binary data.

The bootstrapping and ensemble scheme makes RF strong enough to overcome the problems of overfitting and hence there is no need to prune the trees. Besides high prediction accuracy, RF is efficient, interpretable and non-parametric for various types of datasets.

The model interpretability and prediction accuracy provided by RF is very unique among popular machine learning methods. Accurate predictions and better generalizations are achieved due to the utilization of ensemble strategies and random sampling. Besides, open-source implementations for RF are sufficient in the different programming language.

## 6.2 Clustering

We tried the clustering on the basis of semantic features using two algorithms

### 6.2.1 DBSCAN

DBSCAN algorithm is based on the density of elements and has several advantages over the K-means clustering. This algorithm does not need to indicate a number of clusters, and it has the ability to find any geometric clusters. But, in the course of running, this algorithm requires high computational resource when running online in comparison with K-means.

In our DBSCAN algorithm, we use the features described above to calculate the domain distance and to group the domains that are generated by the same DGA together according to their domain feature difference. Let 'di' and 'dj' be two different domain names, where i = j. We first set i = 0 representing the first domain and then calculate the overall distance between di and all other domains.

### 6.2.2 K-means

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into K-predefined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intracluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The results from clustering only based on linguistic semantic feature gave 30 per cent accuracy. It was observed that a huge amount of AGD domains from different DGA had overlapping features, but on clustering using K-means and specifying the amount of cluster as (Total AGD / 10 ) gave clusters with maximum domains belonging to one cluster thus solidifying the belief that proper cluster can be formed if we merge the cluster based on the IP addresses of the underlying domains.

## 6.3 Data Sets

### 6.3.1 Alexa Domains

For legitimate domains, an obvious choice is the Alexa list of top web sites. The Alexa Top Sites web service provides access to lists of web sites ordered by Alexa Traffic Rank.

We only concentrated on the second level domain without top level domain. It is important to shuffle the data randomly for training/testing purpose and sample only 80 per-cent of total data, for training purpose rest should be used for testing the classifier.

### 6.3.2 DGA Domains

The AGD's are created by running the DGA algorithms available on [16].

## 6.4 Typo Filter

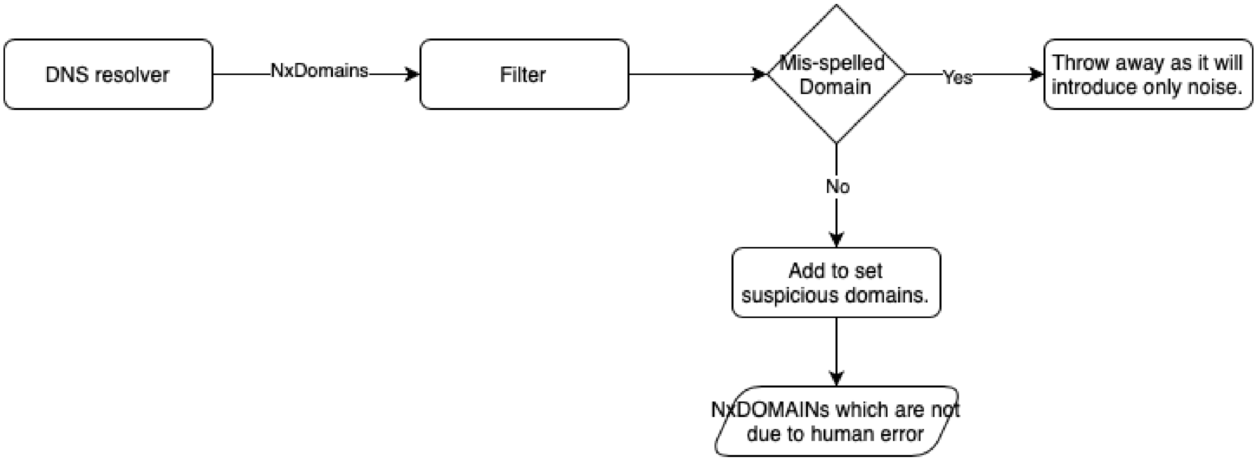
To automatically identify and filter out "accidental", user-generated NXDomains due to typos or mis-configurations. When we receive a domain we match it along the trie as long as it is possible then from the last matched node do a depth-first search to get all the domains that have the matched part as the sub-string. From all the domains we get we calculate the Levenshtein distance from the queried string if the results are within a certain threshold we discard the domain by labelling it as typo.

### 6.4.1 Levenshtein Edit Distance

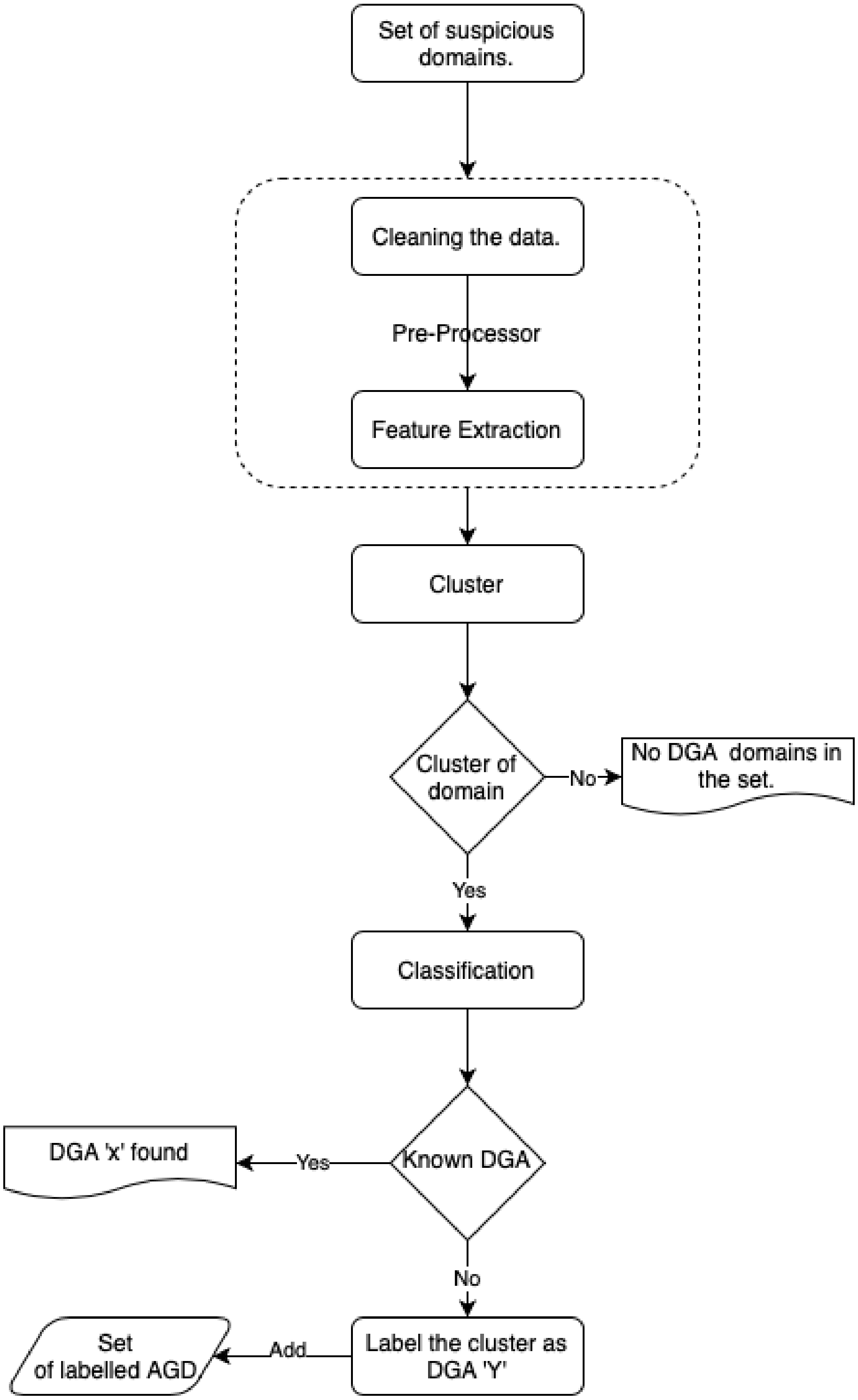
It measures the minimum number of single-character edits between a current domain and its previous domain in a stream of DNS queries received by the server. The Levenshtein distance is calculated based on a domain and its predecessor.
For example, given two strings "test" and "task," the Levenshtein Edit Distance between them is 2 because the characters that need to be edited are 'e' to 'a' and 't' to 'k'. Another example is that for "word" and "world", the Levenshtein Edit Distance is 1 because we just need to add a 'l' after the 'r'.

# 7. Flow Diagram

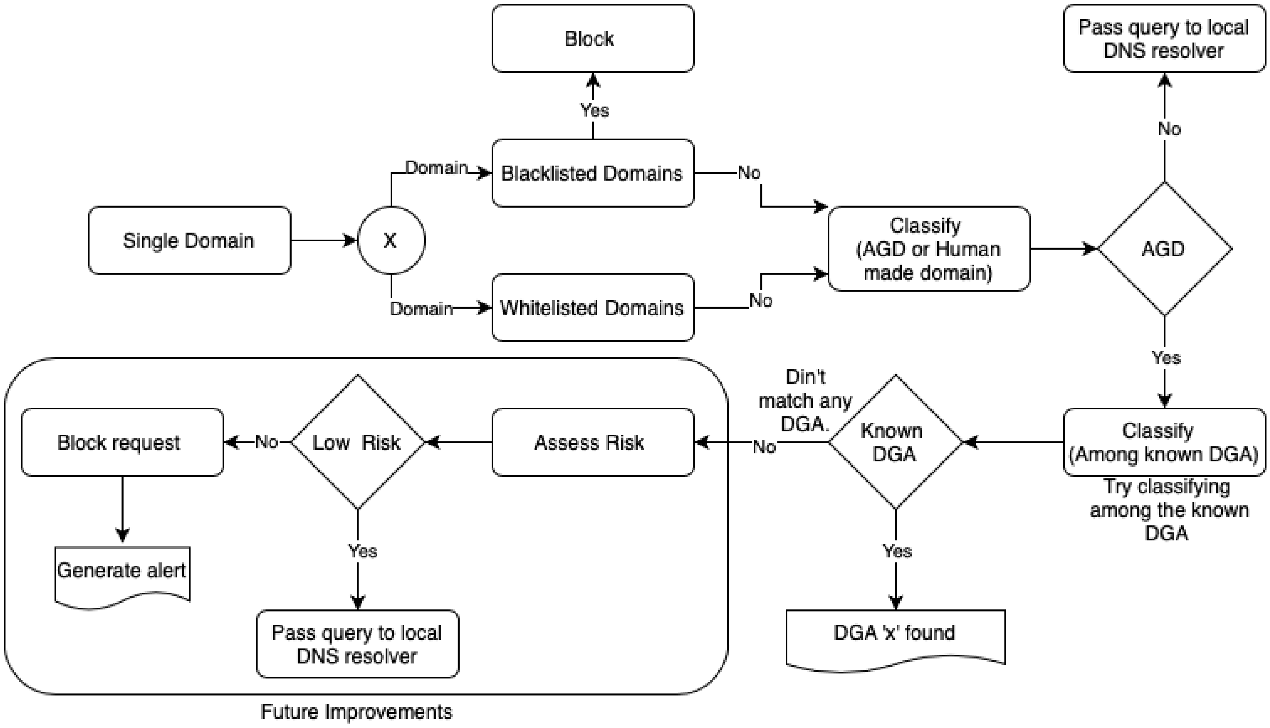
## 7.1 Set Generation


DNS resolver
NxDomains
Filter
Mis-spelled Domain
Yes
Throw away as it will introduce only noise.
No
Add to set suspicious domains.
NxDOMAINs which are not due to human error

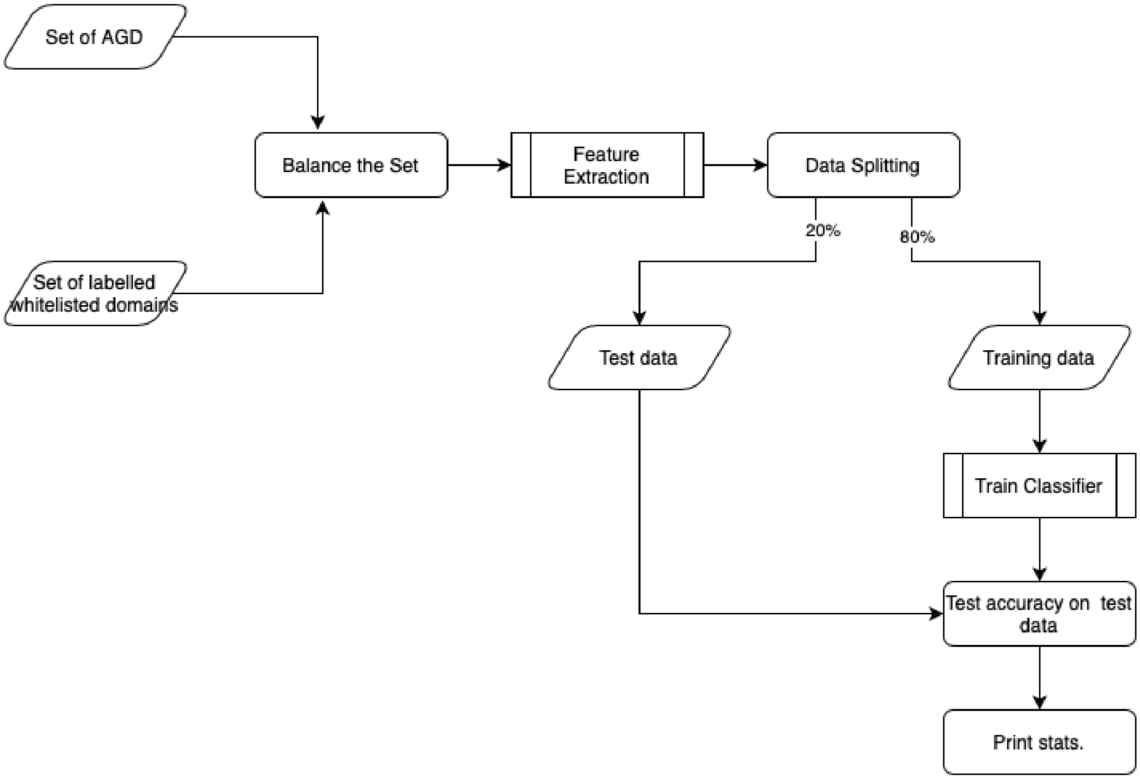
## 7.2 Passive Discovery



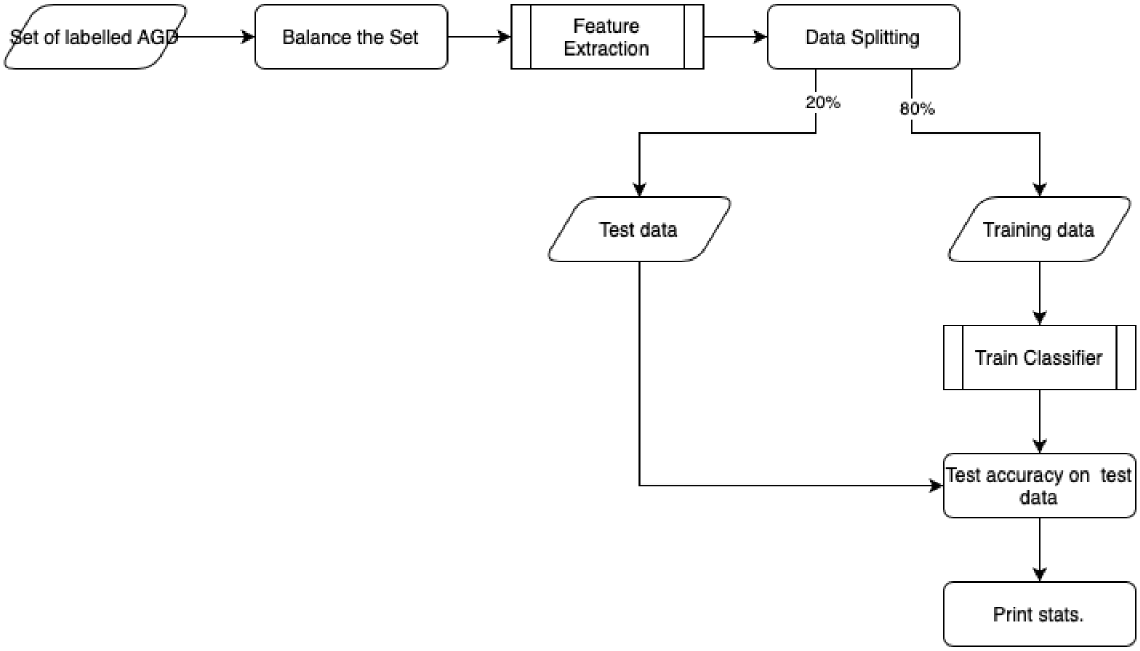
## 7.3 Active Discovery



## 7.4 Training Classifier ( AGD or Human generated domain )



## 7.5 Training Classifier (For different DGA)



# 8. Suggestions

The suggestions include

1. Word graph-based approach to deal with dictionary DGA.
2. Use more additional semantic features.
3. Use modified Mahalanobis distance to achieve lower computational complexity for distance calculation that provides better system performance in real-time.
4. Using K-means in place of DBSCAN algorithm with modifications to improve performance.
5. We can analyze the access behaviour to guarantee the completeness of the detection system.

# 9. Reference

1. A Machine Learning Framework for Domain Generation Algorithm-Based Malware Detection
2. A Method for Detecting DGA Botnet Based on Semantic and Cluster Analysis
3. A Method to Detect Machine Generated Domain Names Based on Random Forest Algorithm
4. A Taxonomy of Domain-Generation Algorithms
5. BotScoop: Scalable detection of DGA based botnets using DNS traffic
6. CLEAN : an Approach for Detecting Benign Domain Names based on Passive DNS Traffic
7. Dissecting Domain Generation Algorithms Eight Real World DGA Variants
8. Detecting Algorithmically Generated Domains Using Data Visualization and N-Grams Methods
9. Detecting the DGA-Based Malicious Domain Names
10. Finding Domain-Generation Algorithms by Looking at Length Distributions
11. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware
12. Dictionary Extraction and Detection of Algorithmically Generated Domain Names in Passive DNS Traffic
13. Tracking and Characterizing Botnets Using Automatically Generated Domains
14. Kindred Domains: Detecting and Clustering Botnet Domains Using DNS Traffic
15. Malicious Domain Names Detection Algorithm Based on N-Gram
16. https://github.com/baderj/domain$_{g}eneration_{a}lgorithms$