AutoEncoded Domains with Mean Activation for DGA Botnet Detection

Binay Dahal
Department of Computer Science
University of Nevada, Las Vegas
Las Vegas, Nevada 89119
Email:binay.dahal@unlv.edu

***Abstract*—Botnets are the powerful and effective way of performing malicious activities over the internet. Over the years, it has evolved into many forms. Earlier bots used static IP to communicate with their command and control server. This method stopped working as soon as that specific IP was identified and blocked. These days, domain fluxing botnets are mostly in practice. The idea is, using Dynamically Generation Algorithm (DGA) to generate domains and use it to connect with C&C server. Numerous researches have been done to detect DGA botnets. These includes deriving features based on alphanumeric distribution of DGA domains and performing classification on it. Other studies include network logs analysis, time series analysis etc. Most of these domain classification works rely upon the features developed and may not work well if the botmaster decides to generate domain with completely new features. We are concerned with developing algorithm that is resilient to feature change that also work well for domain generated by completely new algorithm that was not seen before. We generated 16 bit representation of domains using autoencoder and classified it as benign or DGA generated using supervised learning(with neural net and SVM). To make it work with previously unseen algorithm, we tweaked our method with mean activation of 16-bit domain representation. This helped improve classification accuracy for completely new set of domain generation algorithm by up to 16%.**

Keywords—DGA Botnets, AutoEncoder, Malicious domain Detection.

I. Introduction

The most prevalent malwares these days are the botnets. Botnets are the network of infected computers called Bot which communicate with a single Command & Control Servers (C&C Servers) to perform malicious task such as DDoS attacks, email spamming, click fraud etc. For the botnet to be effective, it must connect with a single C&C server which provide the instruction to perform specific nefarious activity. Using the static IP for the C&C server might not be a good idea, as that specific ip can be discovered and blacklisted. Hence, these days botnets have a new way of communicating with the C&C server. This new method employs domain fluxing which is changing the domain names of the server to avoid getting blacklisted.

The bots use some sort of Domain Generating Algorithm (DGA) to generate large number of domains and tries to connect with each of them. As Botmaster already know the set of domains that are generated by the bots, they can register some of those domains and point them to the C&C server. Once, the request sent from bot to any domain is resolved, it then connects to the server pointed by that domain name and start getting instructions. The domain name to the server is periodically changed to avoid detection. This DGA based botnets are resilient to ip blacklisting or sinkholing and hence pose a serious challenge to network administrators. But as mentioned above, each of these botnets send hundreds to thousands of request to the domains they have generated. Only few (one or two) domains are actually registered, so there are lots of unresolved DNS response with NXDOMAINS. This trend encouraged network security researchers to look into network data and find if the host is infected with botnets.

Various attempts have been made to detect a DGA based botnets based on the network activities. This involves finding some sort of anomaly in a network behavior to conclude it is infected. For example, DNS records of infected host contains a lot of unresolved queries. This can be a symptom that it is a part of botnet. Research have been done to analyze the time sequence of a host activity. If there exists a specific pattern like sending DNS queries on some fixed time intervals, this can also signal that it is a part of botnet. Although, these methods of identifying a botnet have yielded pretty good result, it involves analyzing the complete DNS records which may not be always available or it involves manually designing features. If the botmaster decide to alter the botnet in some form, previously formed method wont work well. Hence, we are using new way of devising feature which is resilient to change in the form of botnets. We propose to use deep learning to detect if an individual domain is DGA generated or not. Deep Neural Networks are renowned for automatically engineering features based on the large number of training examples. This developed deep network will be able to detect if a domain is malignant for the new class of DGA algorithms as well.

II. Related Works

Numerous researches have been done in identifying DGA botnets. One of such works is done by Gu et al. [1]. It analyzes network traffic and aggregate it in some format. It look for the suspicious activities performed on a network. Such activities can include port scanning, spam or malicious binary downloading by hosts etc. The assumption is bots in the same botnet perform exhibit similar malicious activities. So they cluster the hosts based on such similar patterns. One disadvantage of this approach is they cant identify single infected host since it is based on correlation of network events.

Similarly, BotDigger[2] is proposed by Zhang et al. It monitors the DNS traffic of a single network with the aim of detecting DGA botnets. To do this, it uses the quantity, temporal and linguistic evidence. Another method is proposed by Antonakakis et al. [3] which processes the domain names of the DNS queries resulting in a NXDOMAIN response. Yadav et al [4] have proposed a method to tell if a domain is generated by DGA or not by looking into the distribution of alphanumeric characters, as well as bigrams in all domains that are mapped to the same set of IP-addresses. They present three metrics :- KL-distance, Edit distance and Jaccard measure for classify the given domain. Their methodology is primarily based on the observation that most of the current botnets do not use well formed and pronounceable language works since the likelihood that such a word is already registered at a domain registrar is very high. They have obtained pretty good result with this method. The detection accuracy is up to 100% with 2.56% of false positive rate.

Instead of looking into the formation or distribution of domain names, some works have been done by analyzing the temporal pattern of a network activity of a host. One of such work is done by Bonneton et al [5]. Clustering the time series of a network activities, they demonstrate that the DNS traffic generated by hosts belonging to a DGA botnet exhibits discriminative temporal patterns. Then they build decision tree classifiers to recognize these patterns. Detecting network anomaly to identify the DGA botnets has been researched extensively as well. Gavrilut et al [6] have used the complete network traffic to develop various algorithms for DGA detection. Clustering the dns queries, checking for random domain generation, analyzing the distribution of the DNS queries are performed. In case of unavailability of complete network traffic due to privacy issues, they identify botnets based on netflow logs.

Most of the methods described above involves utilizing a static feature of a domain like bigram or distribution of characters etc. Small change in DGA by botmaster can be hard to detect using such methods. However, to the best of our knowledge, there is one work which employs deep learning to predict DGAs. Predicting DGAs using LSTMs[7] uses Long Short Term Memory Network to classify the given domain as DGA or not. This model is resilient to feature change however, it applies no measure to make sure it gives equally good response to the DGAs that were not used to train the model. Hence, we are proposing a model which detects the feature automatically and identifies the botnets based on those features. It will be resistant to change in DGA as it learns the new features based on the changed domains.

III. Method

The main concern while developing our model is feature change resilience and Domain Generating Algorithm change resilience. That is, our model should work even if the features representing the domains get changed and it should yield appropriate result to the domains belonging to new set of DGA that the model has not been trained with before. The later part of previous concern expects our model to correctly identify a new class of DGA. For feature change resilience, we borrow the idea from Deep learning which primarily excels on learning high level features from input data. The model parameters gets updated according to the data so the features learned will be dynamic.

A. AutoEncoder Model

AutoEncoders are the class of algorithm in deep learning which tries to encode the input data in lower dimension with minimum loss. That means input data with higher dimensions can be represented in lower dimensions with the aim of completely decoding the original input if required. It is one of the unsupervised algorithm, meaning it doesn't need labeled data to train the model. The autoencoder model consists of encoder layer and a decoder layer. It is shown as in figure below:

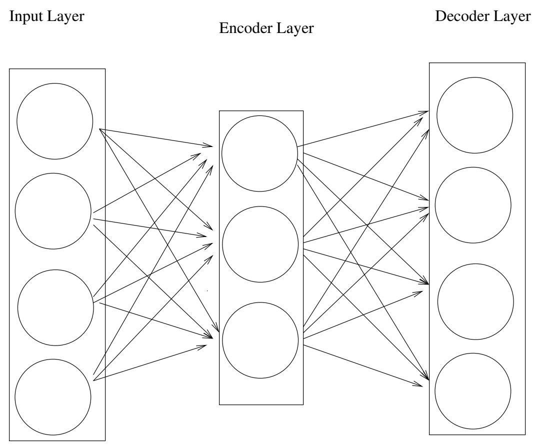

Fig. 1. An autoencoder model

Let us consider the input to the model is $x = x_1, x_2, ..., x_n$, α is the activation function from input to encoder layer and β is the activation function for decoder layer from encoder. As the output of from decoder is just the reconstruction of input x, output y should be equal to x. Hence, the model is trained with the objective of finding optimal parameters θ by maximizing the following posterior probability:

$$\hat{\theta} = \arg\max_{\theta} p(\theta|x, y)$$

Given a domain $d = d_1, d_2, d_3, ..., d_n$ where n is the size of input domain, we get the encoded representation of d as $d_e = d_{e_1}, d_{e_2}, d_{e_3}, ..., d_{e_m}$ where m is the size of encoded domains.

B. Supervised training model

Our supervised model comprises of a 1 layer deep neural network with its hidden layer equal to the trained encoder layer from autoencoder model. This is the fine-tuning of the model[8] pre-trained with autoencoder. It is shown in figure below:

This model is concerned with reducing classification loss J given input x and label y. Mathematically, the objective of this model is:

$$\min_{J(x,y)}$$

, where J can be any loss function like cross-entropy error or squared cost error.

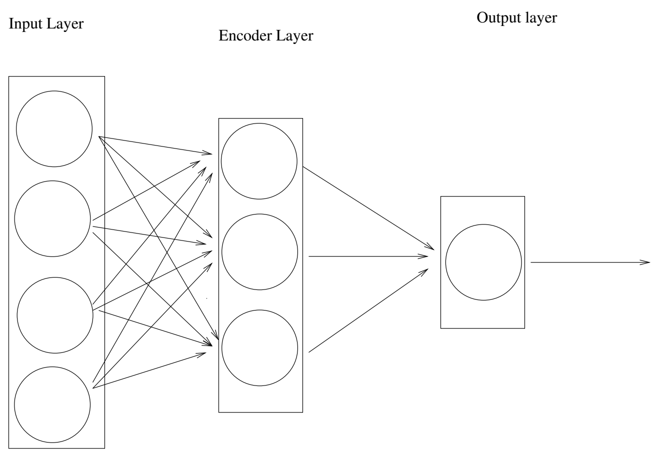

Fig. 2. Supervised learning model

C. Mean Activation Encoder Units

The two models described above appropriately deal with the feature resilience of input data. That means, supervised learning model with encoder layer can tune its model parameters according to the data so any change in input characteristics can be accommodated my simply training the model with new data. But, it may not be good if the model performs poorly to the domains of new domain generating algorithm that it was not trained with before. Our Model may optimize its parameters after we get the data from this new algorithm and train with it. However, for a model to be more robust, it should perform reasonably well for a new DGA without even training with it.

The concept of Mean Activation Encoder Units is finding the mean number of Encoder layer units that is firing when the network predicts the specific input domain is "malicious". To put it more clearly, let us say, the neurons in our model activates to 1 if the input domain is "malicious" or 0 when it is "benign". If N is the total size of dataset, m is the number of encoder layer dimension, e_0 is the encoder layer neuron firing 0 and e_1 is the neuron firing 1 for each input,

$$C_{e_1} = \sum_{i=1}^{m} 1 \quad if \quad e = e_1$$

,

C_{e_1} is the number of encoder layer neurons firing 1 and e is each encoder layer neuron. Now we can define Mean Activation Encoder Units (MAEU) as,

$$Mean\,Activation\,Encoder\,Units \;=\; \frac{\sum_{n=1}^{N} C_{e_1}}{N}$$

It is the mean encoder units that is firing 1 across the whole data set. After we get the value for MAEU after training with our available dataset, we add one more filter to the model. For the domains that are not considered "malicious" by the model, we calculate the number of encoder units firing 1 i.e. malicious, if that number equals or exceeds the MAEU of the network it is categorized as malicious otherwise it is benign.

IV. Experiments and Results

A. Dataset

For benign domains, we used the Alexa-1M[9] data which is the list of 1 million urls deemed benign. This list was compiled by Amazon. For malicious domains, we execut the python script for various DGA classes provided by Endgame on their github site. The class of DGA are Banjori, Corebot, Cryptolocker, Dircrypt, kraken, luckyv2, pykspa, quackbot, ramdo,ramnit and simda. For each of these classes, 10,000 malicious domains are generated. Out of these, ramdo, ramnit and simda are not used for training the model. These are simulated as the new class of DGA. Each data contains top level domain and "benign" or name of dga class as label. Later, we represent benign data by 0 and malicious data by 1.

B. Experimental Setup

For each of the domain, we generate its vector representation by encoding each character to integer. For that purpose, we create a vocabulary of all the characters present in dataset. The size of vocabulary is 37. Feature vector for each domain is created by counting the number of characters i in vocabulary present in input domain. Hence, input dimension is 37.

We create an autoencoder model with input layer size 37 and encoder layer size 16 and train it with input data.This is unsupervised phase. First it is trained with 186,712 data with 121,534 training data, 59,102 test data, and 6,075 validation data. The model is trained for 50 epochs. Each units in the encoder layer has ReLU as activation function while each units in decoder layer has sigmoid activation.

After 50 epochs, our model is tuned with some appropriate weights. Now, we create a supervised learning model with an input layer, a hidden layer and an output layer. The hidden layer is the trained encoder layer from our previous layer. We train this model with our labeled data for another 100 epochs. The output layer activation in this case is sigmoid as well. This is fine tuning phase which is supervised. We use validation data to check if the model is overfitting. After 100 epochs of finetuning, validation data accuracy starts to saturate, so we stop our training. We use binary crossentropy as loss function in each of our training phase.

After unsupervised encoding the input and finetuning the model, it is ready to test whether the domain is benign or malicious for new data. But we haven't applied the Mean Activation Encoder Units filter to our model yet. So, while training the model we keep track of the number of encoder layers units firing when the data is malicious. Using this data, we calculate the MAEU to be 6.98. This means out of the 16 units in encoder layer, about 6.98 units fires to 1 when the data is malicious. Now, we use this value as a filter to generate new result. The results of each of these experiments are discussed below.

C. Results

After 50 epochs of autoencoder training and 100 epochs of finetuning , the model settles to the loss of 0.28. The accuracy of the model at this point is 0.88. We also trained SVM with the encoded representation of domains as input in hope of getting some gain in accuracy. SVM yielded accuracy of 0.89 which is almost same as our supervised model. The result of [7] using LSTM network yielded AUC of 0.99 which is considerably higher than ours however it performs no evaluation on new class of DGAs. Now, to see the impact of applying MAEU on our model, first we test the model with 29,552 malicious data of categories ramdo, ramnit and simda. Domains generated by these algorithms are not present in training. The accuracy of the model for these new class of domains reduces to 0.78. This means, our model has some problem detecting newly inventing DGAs. So, we apply MAEU filter to our model and again test it. This time the classification accuracy for new DGAs rises 0.875. That means, now our model is almost resilient to change in DGAs.

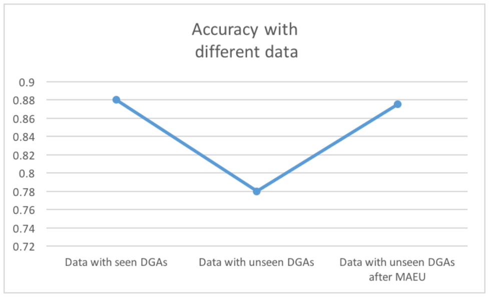

Fig. 3. Accuracy comparison

We test our model for 3 unseen DGAs separately to see the impact of our MAEU filter in each of the classes. We can observe from the figure 4 that there is an improvement of about 16% for simda class of DGA. Other improvements are 2% and 4% for ramnit and ramdo DGA respectively.

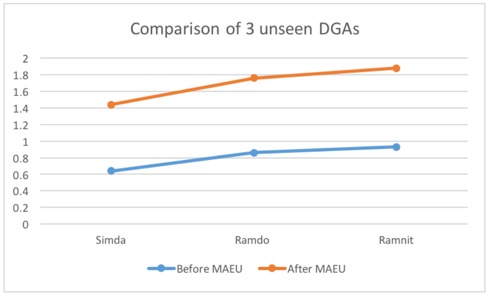

Fig. 4. Improvement in accuracy for 3 unseen DGAs after MAEU

V. Conclusion

In this paper, we develop a model for detecting domains as DGAs using the concept from deep learning. We use autoencoder to generate 16 bit encoding of the input data and use it to train supervised netural network. The accuracy obtained from this model is 0.88. While, this is not the state of the art for DGA detection using deep learning. Models using LSTMs have found accuracy of over 95%. But we invent the concept of Mean Activation Encoder Units to make the model resilient to new class of DGAs without the need to train it with those data. LSTMs are more appropriate for text based analysis, however, using autoencoder to generate intermediate representation of input data enabled us to calculate MAEU.

References

[1] Guofei Gu et al. "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection." In: *USENIX security symposium*. Vol. 5. 2. 2008, pp. 139–154.

[2] Han Zhang et al. "Botdigger: Detecting dga bots in a single network". In: *Proceedings of the IEEE International Workshop on Traffic Monitoring and Analaysis*. 2016.

[3] Manos Antonakakis et al. "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware." In: *USENIX security symposium*. Vol. 12. 2012.

[4] Sandeep Yadav et al. "Detecting algorithmically generated domain-flux attacks with DNS traffic analysis". In: *IEEE/Acm Transactions on Networking* 20.5 (2012), pp. 1663–1677.

[5] Anaël Bonneton et al. "DGA Bot Detection with Time Series Decision Trees". In: *Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2015 4th International Workshop on*. IEEE. 2015, pp. 42–53.

[6] Dragos Teodor Gavrilut, George Popoiu, and Razvan Benchea. "Identifying DGA-Based Botnets Using Network Anomaly Detection". In: *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2016 18th International Symposium on*. IEEE. 2016, pp. 292–299.

[7] Jonathan Woodbridge et al. "Predicting Domain Generation Algorithms with Long Short-Term Memory Networks". In: *CoRR* abs/1611.00791 (2016). arXiv: 1611.00791. URL: http://arxiv.org/abs/1611.00791.

[8] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[9] Amazon. "http://s3.amazonaws.com/alexa-static/top-1m.csv.zip". In: ().