Middle East Technical University, Computer Engineering Department
19 Jun 2017, Ankara, Turkey

DGA Domain Detection using Deep Learning

Halil Burak NOYAN[#], Can CARLAK[#]

[#]*Computer Engineering Department, Middle East Technical University*

***Abstract*— Today, the most powerful attacks such as large scale DDOS attacks, spam messages or other kinds of crimes and fraud are coming from botnets. Attackers use ingenious techniques to hide themselves and control the zombie computers without getting attention. In this article we propose an approach to detect a malicious domains by using its lexical features. Our proposal include a deep learning approach to detect DGA based malware communication.**

I. Introduction

Today, the most powerful attacks such as large scale DDOS attacks, spam messages or other kinds of crimes and fraud are coming from hijacked computers as known as zombies. Malwares with this kind of effect usually requires to be managed from a central facility which is called Command and Control Center (CnC). Malwares usually make use of DNS system to communicate with the CnC. Because of that, security professionals have taken countermeasures such as reverse engineering and sinkholing the domain name the malware communicates with. But attackers did not give up, on contrary they came up with more intelligent systems which makes it infeasible to passivate the botnet only by blocking related domain names or ip addresses. Next chapter explains dns abusement tactics in detail. After that, third chapter includes our data gathering mechanism. In fourth chapter, we propose our solution. Fifth chapter is evaluation and it is followed by the conclusion.

II. Abusing Dns

When a malware is first installed, it registers itself to the botmaster, waiting for the commands and act according to them. The botmaster can be identified using various techniques such as IP, IRC or DNS protocols. Within scope of this article we are only interested in DNS since it is the most commonly used system because it provides versatility and flexibility.

The most common way to prevent the communication between a bot utilizing DNS and its master is to block the domain name it uses. When we know which domain name is associated to the Cnc Server, we can block all the traffic that is directed to there. But this is a static measure and can be bypassed using various tactics.

One well known tactic is using Domain Generation Algorithm (DGA). A DGA algortihm randomly generates vaious domain names. The bot polls on them to connect to the CnC. All botmaster needs to do is to execute the same algorithm with same settings and register any one of the generated domain names. Usually, the registered domain names are short-lived and only needs to be active for a small amount of time. This makes blocking the CnC domains infeasible to fight with the botnet.

Other well known tactics are Fast Flux and Double Fast Flux. They are used to evade detection by changing resolved ip addresses frequently. They are not taken into consideration in this project.

III. Data Acqusition

PassiveDNS tools are great candidates to collect DNS data and the features. Creators of *EXPOSURE* [1], a much broader system than ours, collected two and a half months of passive dns data consisting of 100 billion dns requests. They have used public blacklists to identify malicious domains and acquired their DNS query characteristics. For our proposal, we originally aimed for a similar approach.

Since we were not able to acquire such data, we went for the readily available data sets. We have applied to GT Malware Passive DNS Data Daily Feed [2]. They have agreed to provide us some historical part of the data set. Unfortunately, this dataset does not contain adequete information to extract any of the features in EXPOSURE.

Because of these limitations, we have decided to use publicly available data sets which include DGA and legitimate domains. The best source available is in github repo baderj/domain_generation_algorithms [3].

Even though we are unable to acquire a data set with desired features, we are able to download malwares and execute them in a sandbox to obtain data. This data set is not sufficient for collecting various large scale patterns for DNS requests, so we did not used this set in training phase but for real world evaluation. For that sake, using Cuckoo Sandbox [4], we have executed 53 different malware which include suspicious dns traffic and recorded their DNS queries. As a result, we obtained 2323 DGA Domains and identified 20 CnC servers, which were dissolved into thin air at short notice. We tested final version of our algorithm with them and discussed the results in chapter IV. Evaluation. We also obtained Alexa 1M Domains [5] and Cisco Umbrella Top 1 Million Domains [6] as benign domains and executed the algorithm with them.

We also propose a delicate solution to obtain a complete dataset with desired attributes in Figure 1.

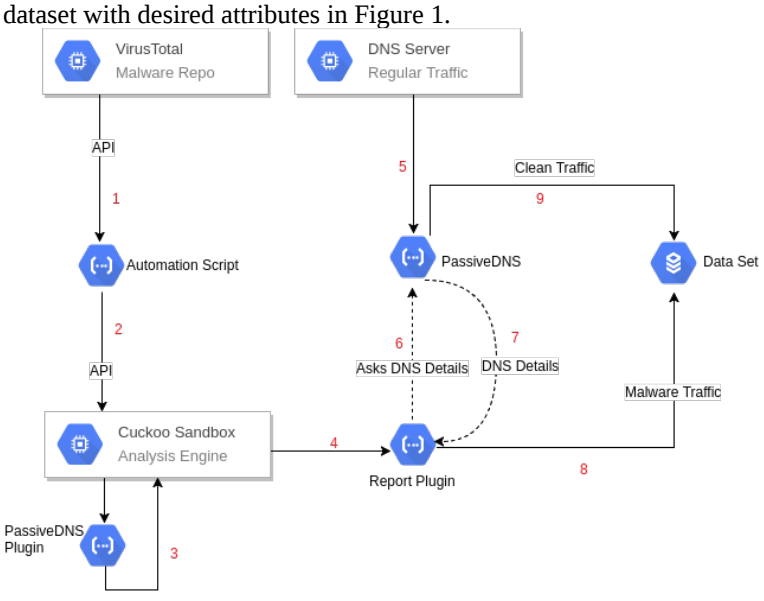

Figure 1

1. An automation script is implemented to fetch malware with specific tags from VirusTotal API.
2. Same automation script sends those malware to Cuckoo Sandbox using its API.
3. Cuckoo Sandbox can save DNS data but needs to be enhanced to perform real time feature enrichment.
4. Cuckoo creates report from the execution of malware.
5. DNS queries can be captured using PassiveDNS.
6. Enrichment similar to item 4 can be done tweaking report plugins.
7. Correlated DNS data is returned from PassiveDNS with some additional fields (for example, total resolution count)
8. Enriched malware traffic can be added to dataset as malicious.
9. Regular traffic can be added to dataset as clean.

IV. Algorithms

Our primitive algorithm can be found at github funcure/dga_predict [7]. Outline of the algorithm is presented in Figure 2 below:

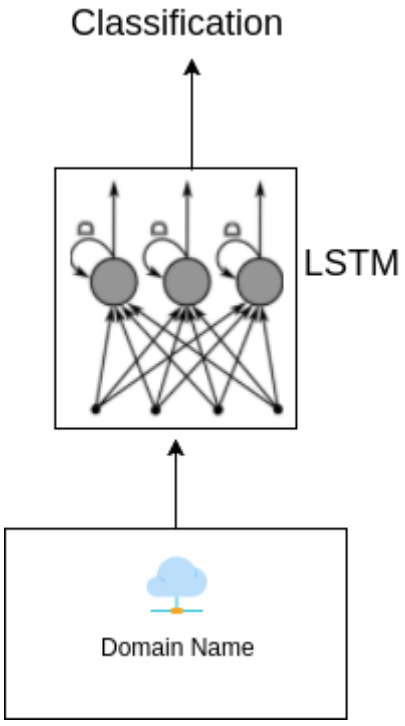

Figure 2

This algorithm is trained and tested with Alexa 1M Domains which are labeled benign and malicious domains generated via baderj/domain_generation_algorithms.

Our first model utilizes LSTM. It have a word embedding layer, an LSTM layer and a fully connected layer with sigmoid activation function. We have used dropout to avoid over-fitting. We have used binary cross-entropy loss with stochastic gradient descent.

We also created a model utilizing CNN for DGA domain detection and trained it with same data set. It consist of convolution-pooling*2 layers, then 2 fully-connected layers. Activation functions are RELU. We have used binary cross-entropy loss with stochastic gradient descent.

Even though we were unable to enrich our data set with additional features, actually, DNS promises much more chance to match different patterns. TTL values, response ip addresses, website popularity graphics can be used as features for future design.

V. Evaluation

Our training data set consist of 104438 malicious and 104438 benign domain names. Real world test data set includes 1 million legitimate domain names and 2323 DGA domains freshly obtained from sandbox as it is explained in section II. Data Acqusition.

LSTM Accuracy for Test Set (0-1 Loss): 94.4%
CNN Accuracy for Test Set (0-1 Loss): 99.3%

CNN yields better results itself compared to our LSTM architecture. At first, we had no better results than 60% with CNN architecture. After various hyper-parameter changes which took hours, we found out that too much iteration was resulting in severe over-fitting. At the beginning our iteration count was 100000 and it we finally decreased it down to 5000 iterations.

We were unable to conduct real world experiments with CNN architecture. We think something is programatically wrong but did not have time to address what it is.

Real World experiments:
LSTM Accuracy for DGA Domains:
True Positive: 76.2%
False Negative 23.8%
LSTM Accuracy for Legitimate Domains:
True Negative: 72.6%
False Positive: 27.4%

After our experiments, at it's best, LSTM yields a terrible false positive rate.

V. Conclusion

Within scope of this article, two types of deep learning algorithms trained and tested using various test sets. As a self assesment, we can say that evaluation part needs a little more content and we need to experiment more using different activation and loss functions. Also, methods to avoid over-fitting such as dropout or regularization should have been discussed in detail and results for both cases should have presented.

Although test sets resulted in good accuracies, our proof of concept research reveals that, without other parameters, it is infeasible to detect DGA based malware just by looking at lexical features of domain names.

References

[1] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding malicious domains using passive dns analysis. In Proceedings of NDSS, 2011.
[2] GT Malware Passive DNS Data Daily Feed, Georgia Tech University, https://www.impactcybertrust.org/dataset_view?idDataset=520
[3] https://github.com/baderj/domain_generation_algorithms
[4] https://www.cuckoosandbox.org/
[5] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
[6] https://umbrella.cisco.com/blog/blog/2016/12/14/cisco-umbrella-1-million/
[7] https://github.com/funcure/dga_predict