A Tutorial for the DAISEC Project

Identify Machine Generated Domain Names

Li-Chiou Chen and Tianyu Wang

Seidenberg School of Computer Science and Information Systems
Pace University

October 2019

CAE Knowledge Unit Covered: Network Defense, Intrusion Detection, Advanced Network Technology and Protocols, Basic Data Analysis

Copyright for Material Reuse

Acknowledgment

The author(s) would like to acknowledge the support from the Department of Defense under CNAP Grant No. H98230-17-1-0335. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. government.

Table of Contents

I. What is a Domain Name System?

The Domain Name System (DNS) is a distributed system maintained by administrative authorities across the Internet. An Internet connected computing device is typically assigned with a domain name so that users can connect to it using the domain name. DNS servers maintain databases mapping from the domain name that is easier for human to understand and interpret to an IP address that is easier for machines to process. The process that maps from a domain name to an IP address is called “name resolution”.

The naming system used by DNS is a hierarchical and logical tree structure called the domain name space. Using their own domain namespaces, organizations can also create private networks that are not visible from the Internet. As in Figure 1, the root of the domain name space is the “.” node. Figure 1 shows a subtree of the domain name space and the path to the root. Every node is called a level domain. The nodes at the base of the tree is called the first level domains or Top-Level Domains (TLDs), such as “edu”. Under this hierarchy, nodes below the TLDs are called the second level domains (2LDs), such as “pace” and the third level domains (3LD), etc.

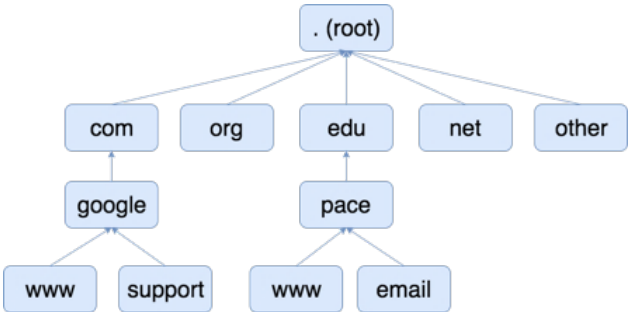

Figure 1. Domain Name Hierarchy.

DNS requests are sent between DNS servers and clients. The DNS protocol is based on TCP at the transport layer. Therefore, these requests are typically neither encrypted nor authenticated. DNS-based misuse and malicious activities are common and are often utilized by botnets to avoid detection.

II. What are Domain Generation Algorithms?

A botnet is a number of Internet-connected devices used by an owner, called botmasters, to perform various tasks. These botnets are groups of malware machines or bots that could be remotely controlled by botmasters. Botnets can be used to launch Distributed Denial of Service (DDoS) attacks, steal data, send spams, or allow the attacker to get access to the target. The software used by a botmaster to control a botnet is called Command and Control (C&C) software.

Most of botnets rely on a centralized server to host C&C software. A bot could query a predefined domain name of the C&C program that resolves IP address of a server that the malware

commands will be received. However, botmasters can lose control of the entire botnet because a C&C server is taken down. Many botnet detection systems use a blacklist of Command and Control (C&C) domains to detect bots and block their traffic. As a response, bot-masters have begun employing domain generation algorithms (DGA) to dynamically produce a large number of random domain names and select a small subset for actual use so that static domain blacklisting becomes ineffective.

Domain Generation Algorithms (DGA) automatically generate domain names for candidate C&C servers based on a given random seed. The bots attempt to resolve these domains by sending DNS queries until one of the domains resolves to the IP address of a C&C server. This method introduces a convenient way to keep attacks resilience.

III. DGA Domain Name Detection

Typically, legitimate domain names preserve a certain meaning so it is easier for human to relate to or remember. A malicious site is more likely to use a machine generated domain name that is changing frequently to elude detection. Based on this property, can we automatically identify machine generated domain names given the knowledge learned from existing legitimate domain names?

The purpose of detecting DGA domain names is to discover whether the domain names are potentially generated by malware or not. Domain names are series of text strings consisted of alphabets, numbers and dash signs. Thus, classification in machine learning would help in DGA domains detection. In this tutorial, we will provide sample data sets and programs for analyzing the domain name data. The tutorial aims to be an introduction to manipulate and analyze domain name data and then classify domain names into legitimate domains and DGA domains using supervised machine learning classifiers such as Support Vector Machines (SVM) and Random Forest.

IV. Domain Name Data Sets

The first step in supervised machine learning is getting labeled training and test data. In this case, we need a list of legitimate domains and a list of domains generated by Domain Generation Algorithms (DGAs).

Legitimate Domains: For legitimate domains, we will use data from the Alexa list of top web sites [1]. Figure 2 shows the first ten entries of this data set. The Alexa Top Sites web service provides access to lists of web sites sorted by Alexa Traffic Rank. Using the web service, developers can understand traffic rankings from the largest to the smallest sites. The entire data

1 Amazon ALEXA project, Available at https://aws.amazon.com/alexa-top-sites/ Last accessed at April, 2017.

set has more than one million domain names. In this tutorial, we will use the top 1000 domains for the analysis.

	A	B	C
1	1	google.com	
2	2	youtube.com	
3	3	facebook.com	
4	4	baidu.com	
5	5	wikipedia.org	
6	6	yahoo.com	
7	7	google.co.in	
8	8	reddit.com	
9	9	qq.com	
10	10	taobao.com	

Figure 2: Top ten domain names from Alexa data set.

DGA Domains: The DGA domain data is collected from the DataDrivenSecurity website[2]. These DGA domains are from recent botnets including "Cryptolocker", two separate "Game-Over Zeus" algorithms, and an anonymous collection of algorithmically generated domains. In total, there are 52,665 entries DGA domains in this data set. In this tutorial, we will use 1000 DGA domains randomly selected from this data set for the analysis.

All of the analyses in this tutorial are done in Python 3. The examples are shown in Jupytre Notebook[3]. You can access the Python developer environment either locally or remotely using a development platform such as Anaconda[4] or Kaggle[5].

V. Process Legitimate Domain Names

The data set, "domains.csv", contains 1000 legitimate domains. In order to support our analysis later, we will first import "domains.csv" into Python, remove the Top Level Domain Names such as ".com",".org" and only keep the first part of the domain names such as "google", "youtube", or "facebook".

As in Figure 3, the program reads "alexa.csv" into Jupytre Notebook as a Pandas[6] data frame, *df*. Pandas is an open-source data structure and data analytics tool for Python. A simple code "*df.shape[0]*" shows the size of the data frame. The result shows that there are 1000 rows of data in the data frame.

2 Data Driven Security Dataset Collection. Available at http://datadrivensecurity.info/blog/pages/dds-dataset-collection.html Last accessed at April 2017.

3 https://jupyter.org/

4 https://www.anaconda.com/

5 https://www.kaggle.com/

6 https://pandas.pydata.org/

```
import pandas as pd
```

```
df = pd.read_csv('alexa.csv', header=None,encoding='utf-8' )
```

```
df.shape[0]
```

```
1000
```

Figure 3: Import “alexa.csv” file.

To see the contents loaded, you can use the code in Figure 4 to see the first ten entries.

```
df.head(n=10)
```

	domain
0	google.com
1	youtube.com
2	facebook.com
3	baidu.com
4	wikipedia.org
5	yahoo.com
6	google.co.in
7	reddit.com
8	qq.com
9	taobao.com

Figure 4: Show first ten entries of the data frame.

To further process the data, we will need to assign the column name. Figure 5 shows that the data frame does not have column names initially and then we assign “domain” as the column name.

```
df.columns.tolist()
```

```
[0]
```

```
df.columns=['domain']
```

Figure 5: Assign column name.

We then split the column “domain” using “.” as the delimiter to separate the different parts of a domain name. After splitting, the data frame contains three column names, including “domain_0”,”domain_1” and “domain_2”, as in Figure 6. Each column contains one part of the original domain name.

```
df1=pd.DataFrame(df.domain.str.split('.').tolist()).add_prefix('domain_')
```

```
df1.columns.tolist()
```

```
['domain_0', 'domain_1', 'domain_2']
```

Figure 6: Split a column using a delimiter.

As in Figure 7, some domain names only contain two parts and the third column contains no data after column splitting. For example, “google.com” is split to “google” and “com”. Some domain names contain three parts and each column contains one of the parts after column spitting. For example, “google.co.in” is split into “google”, “co”, and “in”.

```
df1.head(n=10)
```

	domain_0	domain_1	domain_2
0	google	com	None
1	youtube	com	None
2	facebook	com	None
3	baidu	com	None
4	wikipedia	org	None
5	yahoo	com	None
6	google	co	in
7	reddit	com	None
8	qq	com	None
9	taobao	com	None

Figure 7: The domain names after column splitting.

Next, we assign a label “0” to the domain names in this data set to indicate that they are legitimate domain names. We then keep only the first part of the domain names and the labels and output the data frame to a file called “legit.csv” for further analysis later (Figure 8).

```
df1['label']='0'
```

```
df2=df1[['domain_0', 'label']].copy()
```

```
df2.columns=['domain','label']
```

```
df2.head(n=5)
```

	domain	label
0	google	0
1	youtube	0
2	facebook	0
3	baidu	0
4	wikipedia	0

```
df2.to_csv('legit.csv', encoding='utf-8', index=False)
```

Figure 8: Label the domain names and then output them to a file.

VI. Create a Combined Data Set

The list of DGA domain names, “dga.csv”, contains 1000 domain names. As in Figure 9, we read the file, examine the number of data and take a look at the first ten entries. The data set includes the DGA domain names and the labels associated with it. We labeled the legitimate domain names as “0” and the DGA domain names as “1”.

```
import pandas as pd
```

```
dga = pd.read_csv('dga.csv', header=0,encoding='utf-8' )
```

```
dga.shape[0]
```

```
1000
```

```
dga.head(n=10)
```

	domain	label
0	okbopqhyiiwnmw	1
1	vsiabumtvbjpctm	1
2	pojeubilmngepzmfpheojzdozs	1
3	iteowjkecxfdl	1
4	vhuredvalwdtwcp	1
5	qdeqghbxdwtfpi	1
6	sujknqicwreri	1
7	gutpwbekjejhm	1
8	1jrvm4it2ool9ijipisc5b6c0	1
9	lswxddwowdcray	1

Figure 9: Read “dga.csv” and show its contents.

In order to analyze all of our data together , we need to create a data set combining the DGA domain data set and the legitimate domain data set that we created previously. We import “legit.csv” first (Figure 10) and then combine the data set with the DGA domain data set (Figure 11). There are 2000 records in this combined data set, which is output to a file “domains.csv” for later analysis. We will refer this combined data set to the domain data set.

```
legit = pd.read_csv('legit.csv', header=0,encoding='utf-8' )
```

```
legit.shape[0]
```

```
1000
```

```
legit.head(n=10)
```

	domain	label
0	google	0
1	youtube	0
2	facebook	0
3	baidu	0
4	wikipedia	0
5	yahoo	0
6	google	0
7	reddit	0
8	qq	0
9	taobao	0

Figure 10: Read “legit.csv” and show its contents.

```
data_combined = pd.concat([dga,legit])
```

```
data_combined.shape[0]
```

```
2000
```

```
data_combined.to_csv('domains.csv', encoding='utf-8', index=False)
```

Figure 11: Combine the DGA and legitimate domains.

VII. Calculate features of the domain data set

In this section, we will calculate different features of the domains in the domain data set (“domains.csv”) to prepare for classification. We will calculate two features: the length of a domain and Shannon Entropy of a domain.

As in Figure 12, We first read the data from “domains.csv”. The data set contains 2000 records in which half of the records are DGA domains and the other half are legitimate domains. We then calculate the length of each domain by applying the calculation on each cell in the column

"domain". The results are stored in a new column "length". The lengths are also converted to float numbers for later analyses.

```
import pandas as pd
```

```
df = pd.read_csv('domains.csv', header=0,encoding='utf-8' )
```

```
df.shape[0]
```

```
2000
```

```
df['length'] = df['domain'].map(str).apply(len).astype(float)
```

Figure 12: Calculate the length of a domain.

As in Figure 13, a sub-function *entropy* is used to calculate the Shannon Entropy[7] which measures the amount of information in a given variable, in our case, a domain name. We then calculate the entropy of each cell in the "domain" column and then store the results in a new "entropy" column.

```
import numpy as np
from math import log, e
from scipy.stats import entropy


# Shannon Entropy
def entropy(infor):
    prob_arr = {x:infor.count(x)/len(infor) for x in infor}
    probs = np.array(list(prob_arr.values()))

    return -probs.dot(np.log2(probs))
```

```
size=df.shape[0]
k = np.zeros(size)
for x in range (0,size):
    data1=df['domain'].loc[x]
    k[x]=entropy(data1)
```

```
df['entropy']=k
```

Figure 13: Calculate the Shannon Entropy of a domain.

Figure 14 shows the length and entropy of the first five entries. We then stored the results in an output file "features.csv". We exclude the "domain" column using *df.drop* since this column is not needed for classification (Figure 15).

7 "A Mathematical Theory of Communication", Claude E. Shannon, Bell Telephone System Technical Publications, 1948.http://cm.bell-labs.com/cm/ms/what/shannday/paper.html

```
df.head()
```

	domain	label	length	entropy
0	okbopqhyiiwnmw	1	14.0	3.378783
1	vsiabumtvbjpctm	1	15.0	3.373557
2	pojeubilmngepzmfpheojzdozs	1	26.0	3.815072
3	iteowjkecxfdl	1	13.0	3.546594
4	vhuredvalwdtwcp	1	15.0	3.506891

Figure 14: The first five entries of the analysis results.

```
df1=df.drop('domain',1)
```

```
df1.to_csv('features.csv', encoding='utf-8', index=False)
```

Figure 15: Output the results to “features.csv”.

VIII. Classifying DGA and Legitimate Domains

In this section, we will use the feature file generated from the previous section to classify the domain names into two classes: the DGA domains and the legitimate domains. As in Figure 16, we read the domain data from “features.csv” and then examine the size and contents of the data set.

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv('features.csv', header=0,encoding='utf-8' )
```

```
df.shape[0]
```

```
2000
```

```
df.head()
```

	label	length	entropy
0	1	14.0	3.378783
1	1	15.0	3.373557
2	1	26.0	3.815072
3	1	13.0	3.546594
4	1	15.0	3.506891

Figure 16: Import the domain data set and examine it.

As in Figure 17, we assign all of the columns, except for "label", to the independent variable x, and assign the column "label" to the dependent variable y. Variable x contains column "length" and "entropy" that we calculated earlier. Variable y contains two classes: either "0" indicating a legitimate domain or "1" indicating a DGA domain.

```
x = df.drop('label', axis=1)
y = df['label']
```

Figure 17: Assign independent and dependent variables.

For classification, we will use classifiers from Scikit Learn[8], which provides a set of open-source Python tools in machine learning. First, we will need to divide the data set into the training data and the test data. Figure 18 uses the *train_test_split* module from Scikit Learn to split both x and y into training and test data sets. In this analysis, 80% of the data is used for training and 20% is for testing across both classes. The samples are randomly drawn within each class.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=1, stratify=y)
```

Figure 18: Split training and testing data sets.

We also need to scale both the training and the test data based on the training data. Scaling[9] is an important step to pre-process the data before using machine learning algorithms. This additional step scales the data to fit in the observed scope based on the training data. As in Figure 19, a standard scaler is used to scale the data against the mean value of each column normalized by the standard deviation.

```
from sklearn.preprocessing import StandardScaler
s = StandardScaler()
x_train = s.fit_transform(x_train)
x_test = s.transform(x_test)
```

Figure 19: Data scaling.

In Figure 20, we train the SVM classifier and predict the classes of the test data using the trained SVM classifier.

```
from sklearn.svm import SVC
svcc = SVC(kernel='linear')
svcc.fit(x_train, y_train)
y_pred = svcc.predict(x_test)
```

Figure 20: Classifying the domains using SVM.

8 https://scikit-learn.org/stable/

9 Kaggle is a good source to learn more about data scaling and normalization. https://www.kaggle.com/jfeng1023/data-cleaning-challenge-scale-and-normalize-data

As in Figure 21, we then generate three outcomes: a confusion matrix, a classification report and the overall accuracy score. Confusion matrix (Figure 22) shows the outcomes as true positive (TP), false positive (FP), true negative (TN), and false negative (FN). Both TP and TN are the desirable outcomes when the classifier makes correct predictions. FP refers to a negative outcome but predicted as a positive one. FN refers to a positive outcome but predicted as a negative one. Accuracy score refers to how accurate overall the classification is, calculated as (TP+TN)/total number of test data. Precision refers to how accurate the positive data is classified, calculated as TP/(TP+FP). Recall is calculated as TP/(TP+FN). f1-score is the weighted average of the precision and recall.

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print("Confusion Matrix\n",confusion_matrix(y_test, y_pred),"\n")
print("Classification Report\n",classification_report(y_test,y_pred),"\n")
print("Accuracy: ",accuracy_score(y_test, y_pred))
```

```
Confusion Matrix
 [[177  23]
 [  0 200]]

Classification Report
               precision    recall  f1-score   support

           0       1.00      0.89      0.94       200
           1       0.90      1.00      0.95       200

   micro avg       0.94      0.94      0.94       400
   macro avg       0.95      0.94      0.94       400
weighted avg       0.95      0.94      0.94       400


Accuracy:  0.9425
```

Figure 21: Results from domain classification in SVM.

	Positive (Actual)	Negative (Actual)
Positive (Predicted)	True Positive	False Positive
Negative (Predicted)	False Negative	True Negative

Figure 22: Confusion Matrix.

As in Figure 22, we use another classifier, Random Forest (RF) to classify the domain data. Figure 23 shows the results of this test. Compared to the results using SVM, RF classifier performs better overall, a higher accuracy score. However, although RF performs better than SVM for the DGA class (labled as “1”), 23 FP in SVM and 16 FP in RF, SVM slightly outperforms RF for the legitimate class (labeled as “0”), no FN in SVM and 1 FN in RF.

For leaning purpose, this analysis uses a very small data set in classifying domain names. To investigate the problem of identifying DGA domain names, a larger data set and a more thorough analysis using various features and different types of classifiers will be needed.

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 100, random_state=0)
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)
```

Figure 22: Classifying the domains using Random Forest.

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print("Confusion Matrix\n",confusion_matrix(y_test, y_pred),"\n")
print("Classification Report\n",classification_report(y_test,y_pred),"\n")
print("Accuracy: ",accuracy_score(y_test, y_pred))
```

```
Confusion Matrix
 [[184  16]
 [  1 199]]

Classification Report
               precision    recall  f1-score   support

           0       0.99      0.92      0.96       200
           1       0.93      0.99      0.96       200

   micro avg       0.96      0.96      0.96       400
   macro avg       0.96      0.96      0.96       400
weighted avg       0.96      0.96      0.96       400


Accuracy:  0.9575
```

Figure 23: Results from domain classification in Random Forest.

IX. Appendix

Data sets and programs used in this tutorial can be downloaded from https://drive.google.com/drive/folders/1AegP717YDynwYTaNeLjTaA553Ea116_0?usp=sharing .

The data sets and programs are described below.

alexa.csv: This file contains top 1000 domain names from Alexa Top Sites list.

dga.csv: This file contains 1000 domain names generated by domain generation algorithms. They are randomly selected from DGA list from DataDrivenSecurity. The domain names are labeled as DGA domains.

DgaProg1.ipynb: This program, used in Section V, processes legitimate domain names from the Alexa List. It reads input file “alex.csv” and generates output file “legit.csv”.

DgaProg2.ipynb: This program used in Section VI, combines legitimate domain names with DGA domain names as one file for later analysis. It reads two input files: “legit.csv” and “dga.csv”, and generates a output file “domains.csv”.

DgaProg3.ipynb: This program used in Section VII, calculates features of the domain data set to prepare for classification analysis. It reads one input file “domains.csv” and generates one output file “features.csv”.

DgaProg4.ipynb: This program used in Section VII, classifies domain names into either DGA domains or legitimate domains. It reads one input file “features.csv”.

X. Bibliography

Usman Malik. “Implementing SVM and Kernel SVM with Python's Scikit-Learn,” Available at Stack Abuse (https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/).

Sebastian Raschka and Vahid Mirjalili. “Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow,” 2nd Edition, Packt Publishing, 2017.

Tianyu Wang, Li-Chiou Chen, Yegin Genc. “Detecting Algorithmically Generated Domains Using N-grams Methods,” Working Manuscript, 2019.