National College of Ireland

Using Domain-Based on Machine Learning for Malware Detection

MSc Internship
Cyber Security

Dai Hoang Vu
Student ID: x17165423

School of Computing
National College of Ireland

Supervisor: Ben Fletcher

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Dai Hoang Vu
Student ID:	X17165423
Programme:	Cyber Security **Year:** 2019
Module:	Internship
Supervisor:	Ben Fletcher
Submission Due Date:	12/12/2019
Project Title:	Using Domain-Based on Machine Learning for Malware Detection
Word Count:	4774 **Page Count:** 18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.
I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature: Dai Hoang Vu

Date: 12/12/2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	x
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	x
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	x

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Using Domain-Based on Machine Learning for Malware Detection

Dai Hoang Vu – X17165423

National College of Ireland

Keywords: Malicious Domain Detection, botnet detection, machine learning-based botnet detection, Domain Generated Algorithm (DGA).

Abstract

Cybersecurity attacks are constantly occurring and tend to increase every year. Defensive and preventive measures taken by security experts to protect users are constantly being updated, but attackers are always using more sophisticated techniques. In recent times, many malware has used domain generation algorithms (DGA) to create malicious domains to maintain the C&C infrastructure network (Command and Control). With the use of algorithms from machine learning, we have used approaches from the Logistic Regression, Random Forest and Naïve Bayes algorithms to sort out legitimate domain names and malicious domain names. The result data at the end of the article shows the positive of these methods.

1. Introduction

Strong technology development carries certain risks for both local and global-local area networks. Potential risks from cybersecurity have caused serious consequences in many social fields such as finance, education, health ... According to statistics from reputable rating agencies such as CIS (Center for Internet Security)[1] or statistical reports on the impact of threats from the world's leading security corporation Symantec, in 2019 leading malware like Emotet, Kovter, Dridex SpeakUp or Agent Smith's .. making a big impact, they are infecting millions of electronic devices, stealing information, impersonating and losing a lot of users' assets.

Narrowing the scope of this article, we want to mention the impact of malicious domain names on network security. For example, malicious domain names are used to run C&C botnet servers, when users accidentally download malicious websites and steal sensitive data or blackmail. By using DNS, the

attackers have flexibly altered their IP addresses to avoid detection by security experts. Currently, there are 2 active and passive analysis methods to analyze poisoned domain names. While active analysis is often more expensive than the remaining method, passive analysis is often used. As mentioned in the title, in this article I want to get more from using machine learning techniques to classify domain names to determine which domain is malicious.

In the process of learning about this field, I discovered many similar topics, which proves that the study of machine learning techniques to search for malware is the current research trend. Different from other studies, in this article, we want to focus on researching and developing machine learning techniques to detect botnets based on domain name analysis to detect cybersecurity attacks early. , this is not a completely new method, it incorporates many DGA detection tools from before but was added to improve.

Regarding the layout of the article, after the introduction, the next section, we will go into understanding knowledge of machine learning techniques, approaches and machine learning classification, DGA, compare related articles about the previously reported Fast-flux, Domain-flux, and malicious domain detection systems. The focus of this lesson is methodology, application, and development. Next are the results, summarizing along with future development trends.

2. Literature Review and Background Knowledge

2.1. Machine Learning

Machine learning (ML) is an artificial intelligence software. ML algorithms are computer programs that can learn how to perform tasks and work overtime. In the interpretation of baseline data and the choice of suitable methods for data analysis, ML requires human judgment. The data must at the same time be dry, unbroken, and without false information before use.

A large amount of data is required for ML models to "practice" and evaluate the model. In the past, the large amount of data needed in modeling relationships between data has been lacking from ML algorithms. Big data growth provided sufficient data to improve the accuracy and prediction of the model for ML algorithms.

Commonly used methods of Machine Learning are supervised and unattended learning, in addition to semi-supervised or intensive study. In the next section, we will learn more about these methods.

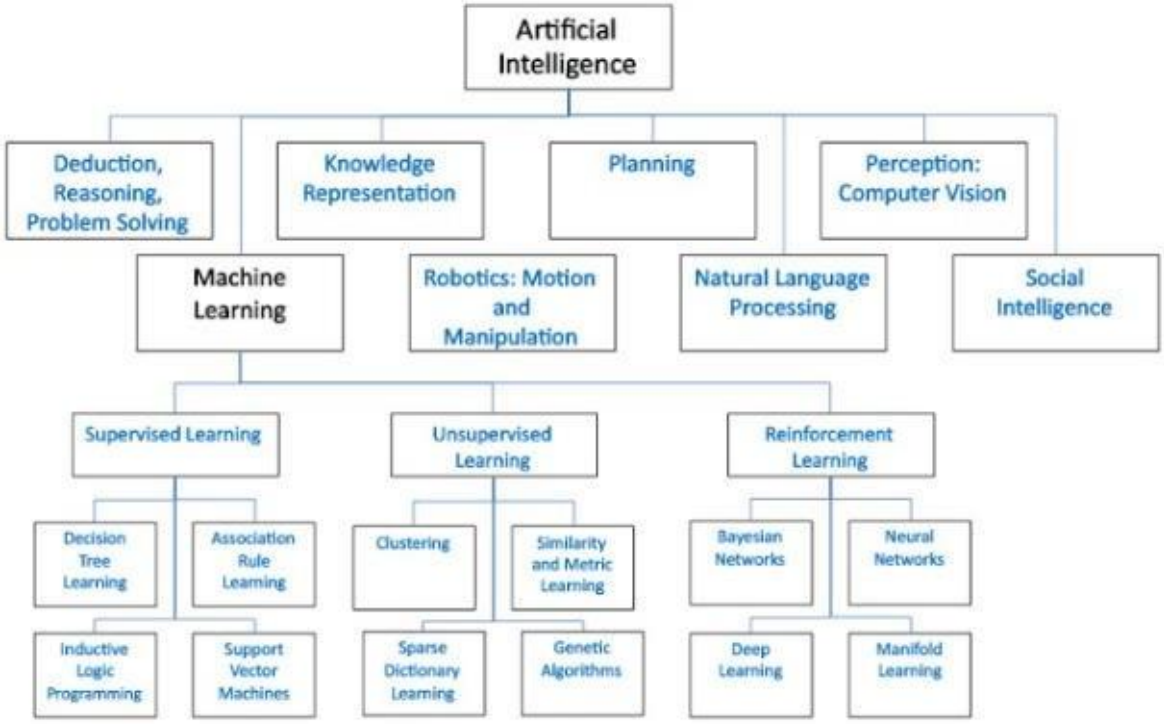

Fig 1: An overview of AI and Machine Learning.

2.1.1 Supervised Learning

Supervised Learning is a machine learning technique to learn from a given set of data sets[2]. The given data set will contain multiple data sets. Each data set structured in pairs {x, y} where x is considered raw data and y is the label of that data. The task of Supervised Learning is to predict the desired output based on the input value. It is easy to recognize, supervised learning means that machine learning relies on human help, in other words, people teach machine learning and the desired output value is predetermined by humans. Training data sets are completely labeled based on people. The smaller the file is, the less computer learning.

Supervised Learning is also applied to two main groups of problems: regression problem and classification problem.

2.1.2 Unsupervised Learning

Unsupervised learning is a machine learning technique that uncovers a model or structure hidden from a set of unlabeled data sets. Unsupervised learning is different from Supervised learning because it is not possible to determine the output from the training data set in advance. Depending on the training, the results will vary. In contrast to Supervised learning, Unsupervised learning training data sets are not labeled by humans, computers will have to learn by themselves. It can be said that learning without supervision, the output value will depend on the Unsupervised learning algorithm.

The most common application of unsupervised learning is clustering. According to this author [3], The most recognizable apps are Google and Facebook. Google can group articles with content close to each other, or Facebook can suggest making friends with many friends in common for you. Articles with the same content will be grouped into a cluster (cluster) distinguished from other groups.

This is an appropriate application to use to classify malicious code based on grouping them based on common factors.

2.2. Cyber Security and related issues

2.2.1. Domain Name System DNS

Yi-da Yan et al [4] presented the concept of DNS as a domain name resolution service. This system guarantees us the search for information resources on the vast internet using the easily remembered URL domain name.

DNS queries are divided into recursive and iterative queries. The recursive query is the query from the client to the DNS server while the recursive query is the query between DNS servers.

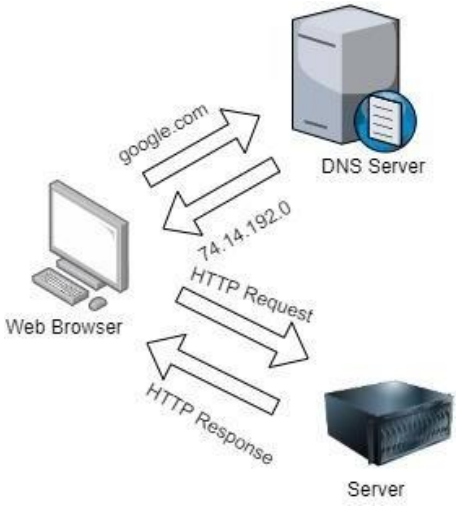

Fig 2: DNS Queries Example: google.com

Hackers attack servers based on using DNS. There are famous attacks like DDoS Attack, Domain hijacking or DNS Spoofing ... It can be said that through DNS Spoofing or DNS

Hijacking is a Man in The Middle Attack attack [5], changing DNS records redirect users' access to another server, this server contains malicious code where the user's data may be stolen or eavesdropped. Or DDoS attack, this is a type of DNS server attack, this server collapses or denies service, another way is to use the DNS server to attack other servers on the network, causing the server to be The attacker must deny the service[4].

2.2.2. Malicious Command and Control (C&C)

Command and Control are servers that control electronic systems compromised by malicious code[6]. By using this server, hackers can attack large numbers of computers at the same time without fear of being detected. The use of malicious code to gain control of C&C will facilitate hackers to penetrate and infringe with malicious intent.

2.2.3. DNS Issues: Botnet

According to J.Jiang et al [7], Botnets are computer networks made up of computers that hackers can remotely control. The computers in the botnet are infected with malware and controlled by hackers. A botnet can have hundreds of thousands or even millions of computers.

If your computer is part of a botnet, it is already infected with one of the malware (viruses, worms, etc.). Hackers creating this network will use and control hundreds of thousands of victims' computers to serve their purposes.

Botnets can be used for a variety of purposes. Because a botnet is a network of many computers, hackers can use a botnet to launch denial of service (DDoS) attacks on a particular web server. Accordingly, hundreds of thousands of computers will "bombard", access to a target website at the same time, causing traffic to that site is overloaded. As a result, many users accessing the website become congested, resulting in inaccessibility.

In recent studies on how to prevent botnets, researchers found that to avoid detection from static mechanisms, attackers took advantage of the creation of Domain Generation Algorithms (DGAs). The creation of these domain names makes the number of domain names turn into random numbers, most of them do not exist or contain malicious codes that users cannot expect.

2.2.4. DNS Issues: Domain Generation Algorithms (DGA)

Domain Generation Algorithms (DGA) is an algorithm that creates a large number of automated domains to be the key between command servers and their modifications[8]. This method allows you to retain attacking re-silence because the Bot can eventually get the correct IP address and use DNS queries for the next DGA domains if a domain name has been found and downloaded.

```python
from datetime import date
from hashlib import sha256

def dyre_dga(num, date_str=None):
    if None == date_str:
        date_str = '{0.year}-{0.month}-{0.day}'.format(date.today())

    tlds = ['.cc', '.ws', '.to', '.in', '.hk', '.cn', '.tk', '.so']
    hash = sha256('{0}{1}'.format(date_str, num)).hexdigest()[3:36]
    replace_char = chr(0xFF & ((num % 26) + 97))

    return '{0}{1}{2}:443'.format(replace_char, hash, tlds[num % len(tlds)])

todays_domains = [dyre_dga(i) for i in xrange(333)]
```

Fig 3. Example of DGA

Figure 4 shows a botnet mechanism that uses DGA to automatically generate and register domain names for its C&C server. Accordingly, C&C servers and bots use the same DGA algorithm with the same seed, so they can generate the same set of domain names. DGA botnets often use date and time as the kernel to initiate the domain generation algorithm, and as such, the DGA botnet creates a set of domain names every day it operates. To initiate a connection to the C&C server, a bot first needs to implement the DGA algorithm to generate a domain name and this domain name can also be automatically generated by the C&C server used by both the C&C server and the bot. a DGA algorithm and a kernel. After creating the domain name, the bot uses the DNS system to resolve the domain name into the IP address of a C&C server. If the domain resolution process fails, the bot uses the DGA to generate a new domain name and repeat the request for address resolution. If the domain resolution process is successful, the bot uses the IP address to connect to the C&C server to receive commands and controls from the botmaster.

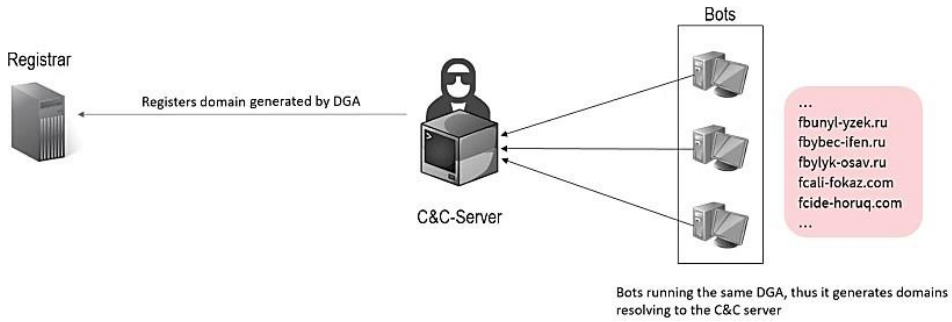

Figure 4. Botnet mechanism uses DGA to automatically generate and register domain names for C&C server [9]

2.3. Related Work: Several techniques apply Machine Learning to detect malicious domain names.

During my research and selection of methods in machine learning engineering, I researched and read many articles that were researched by other authors about how they deal with problems and optimal solutions. most to them. Each of these methods belongs to a diverse subset of

Machine Learning, but most give the same results. In my research, I will list the methods they use along with their pros and cons.

- Wenjie et al [10] introduced the Random Forest algorithm to detect DGA. By collecting domain data and dividing it into two types, the Black and White domain. Black domains are malicious domains created from Conficker, Aqlm, KSWebShield and DGA domain names in which legit domain names from Alexa are classified as a White list. The results from using this algorithm are very effective with accuracy ranging from 0.894 to 0.935. Most malicious domains are found with very high accuracy. It can be said that this algorithm is a new development trend of Machine Learning which is widely used by scientists. The team also proposed plans to develop the algorithm to achieve higher results.

- Xuan Dau et al [11] presented how their team discovered Botnet based and used DNS Query Data. After discussing the effects of botnets on the current cybersecurity situation, the team came up with this method based on several monitoring algorithms of machine learning techniques including kNN, decision trees, Naive Bayer and random forest. The domain data after being collected has been divided based on the n-gram formula. With 18 features extracted from each domain name form, they split datasets into 3 parts T1, T2, T3, and training test, with such a division and flexible use of 4 algorithms they have made conclusions about degrees. accuracy of each of these algorithms. Naive Bayer has the lowest accuracy while Random Forest gives unexpected high results. However, it is not possible to say that Random Forest is the best because the test time is longer than the other ways. In summary, the team concluded that they would choose the random forest algorithm to guide their future research at the highest accuracy of this algorithm.

- Similar to the method mentioned above, Xuan Hanh et al [12] added training data for one more episode. The Training Data T4 is supplemented with a combination of data from T2 and T3. Domain names will be processed through algorithms to remove top-domain. The results obtained at this test after adding new features, the classification measurements are significantly improved compared to the original model. Explaining, for this reason, the author said that the new features introduced by the author contributed to the ability to distinguish between good and bad domain names. However, the disadvantage of adding these new features is the increased training time due to the increased size of the domain representation vector. This is a good development step to reduce errors and increase accuracy when classifying domain names.

- Chhaya C et al [13] presented a method for detecting DGA using a combination of the Random Forest algorithm for classification along with the Deep Neural Network algorithm to see the performance of more than 2 million domain names used by the group. This is a good combination of Machine Learning and Deep Learning with 28 features extracted from domain names. Both algorithms have an extraordinary approach and play an important role in the prevention of malicious domains

- In the process of researching to complete my essay, I discovered an integrated essay that enabled me to develop my essay. The group of students from China Yi-da Yan et al [14] gave me a comprehensive view of malicious domain detection methods based on machine learning techniques. After presenting their views on DNS, DGA, and related issues, they linked to the methods of detection using Fast-flux and Domain Flux. According to their presentation, these are quick and effective methods and can detect a large number of botnets. To summarize their research, this group of students from China has come up with a combination of solutions such as using network traffic, passive DNS information and Machine Learning Algorithms to bring very practical results. This is one of the prerequisite studies for other in-depth studies.

- Together with a diverse combination of surveillance machine learning methods for detecting malicious domain names, the team of researchers from Vietnam, Hieu Mac et al [15] presented their views on how to view receive a selection of appropriate methods to detect a large number of the malicious domain using Machine Learning. After making hypotheses and practices of several experimental programs including the Hidden Markov Model, LSTM, Recurrent SVM, Decision Tree, Support Vector Machine and Bidirectional LSTM, the team evaluated that Bidirectional LSTM and Recurrent SVM has achieved the highest detection rate on both binary and multilayer classifications. However, the team also assessed that depending on the cost and time conditions, there must be a specific method of choosing.

- In contrast to the above authors, Tianyu et al [8] suggested that the use of Data Visualization and the N-gram Method in Machine Learning brought positive results. By providing a panoramic view of the application of computational formulas for n-grams along with testing three common machine learning methods that the authors still use such as SVM, Random Forest or Naive Bayer to divide This kind of malicious domain name, this author group gave the N-gram results giving the highest data in the classification. During the analysis process, they had many problems setting up the DNS server, but they eventually obtained positive results and created conditions to expand the analysis in the future.

- To conclude the comparative analysis of related articles, I will mention another group of authors who have used common methods but combined with a new algorithm, Hieu Ho Duc [16] and his colleagues. I presented my method of detecting DGA by applying the machine learning method. By collecting data from many sources including good and bad data, they used the Viterbi algorithm to swap domain names into vectors. From this point, they used Logistic Regression and Convolutional Neural Networks algorithms to produce the final results. After 80000 training steps on training set and test set, the team has achieved an almost 98% absolute result after applying the above algorithms. The team also concluded that using the combination of Viterbi algorithm and convolutional neural network results in higher results than the other method. These are good preconditions for botnet detection along with early detection of DDoS attacks.

3. Methodology

3.1. Data Gathering

- Legit Domain File:

  Good domain data is gathered from many sources. Based on several reports already if above I have collected the top stable source names from Alexa. Source names from this domain are often verified, so it's absolutely a clean source of data to download.

  For example: facebook.com, google.com, dantri.com.vn..

- Malicious Domain File:

  Large numbers of malicious domain names were collected from [17], 360 Lab DGA and Bambenek Consulting. These are the domain names created from DGAs: Zeus, chinad, corebo, gozy, pizd, Cryptolocker, Goz, ... Taking malicious code from third-party parties needs to be very careful and should be tested carefully first. when put to use.

  For example: avhdmamfeaw.eu, sjaiciewic.com, awqdwq.us, ...

3.2. Applied Algorithms

In my thesis, I will apply 3 common algorithms to classify and detect malicious domain names including Logistic Regression, Random Forest and Naive Bayer. First of all, we will go into how these three types of algorithms work:

3.2.1. Logistic Regression

Logistic regression is another algorithm that machine learning borrows from statistics. This is the best method for binary classification problems (problems with 2-layer values)[18].This is a model that can learn quickly and effectively with binary classification problems.

3.2.2. Random Forest

The Random Forest is a collection of hundreds of Decision Trees, in which each Decision Tree is randomly generated from reselection (select a random part of the data to build) and random variables from all variables in the data[19]. With such a mechanism, the Random Forest gives us a very accurate result but trade-off by not being able to understand the mechanism of this algorithm's operation due to the complicated structure of this model - thus the algorithm This is one of the Black Box methods - that is, we will put our hands inside and draw the results but cannot explain the mechanism of action of the model. That is the trade off between explanatory ability and predictive ability as I stated in the first article.
Random Forest is a supervised learning method so it can handle problems of classification and classification (prediction of values).

3.2.3. Naive Bayer

A classification model is a Machine Learning model used to classify samples based on defined characteristics[20]

Naive Bayes is a classification algorithm modeled based on Bayes's theorem in statistical probability: where:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

P (y | X) is called posterior probability: the probability of the goal y with the condition having the characteristic X
P (X | y) is called likelihood: the probability of the X characteristic when the target y is known
P (y) is called the prior probability of goal y
P (X) is called the prior probability of characteristic X
Here, X is the feature vector, which can be written as:

$$X = (x_1, x_2, x_3, ..., x_n)$$

Then, the Bayes equation becomes:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

In Naive Bayes model, two assumptions were made:

The features included in the model are independent of each other. That is, a change in the value of one characteristic does not affect the other.
Features included in the model have equal effects on the target output.
Then, the objective result y so that P (y | X) reaches its maximum becomes:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Because these two assumptions are almost non-existent in the above reality, this model is called naive. However, its simplicity with the very fast prediction of the output makes it very much used in practice on large data sets, resulting in positive results. Some applications of Naive Bayes include spam filtering, text classification, text nuance prediction, ...

Three different Naïve Method used for this research :

- Multinomial Naive Bayes

This model is mainly used in text classification. The input characteristic here is the frequency with which words appear in the text.

- Bernoulli Naive Bayes

This model is used when the input characteristics only receive binary values 0 or 1 (Bernoulli distribution).

- Gaussian Naive Bayes

When the characteristics receive continuous values, we assume those features have a Gaussian distribution. The likelihood will look like this:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

3.3. Constructing Data

The data is collected from Alexa and the 3rd party sources will be dissected using algorithms from the skrpt source in the Jupyter Lab library. The attributes will be extracted from the use of such algorithms. domain length, number of subdomains or subdomain length,...
We can example the following 2 domains after using the algorithm we have the following results:

Attributes	dantri.com	sdjwfndjciekd.ie
Domain name Length DNL	10	16
Number of Subdomains NoS	1	1
Subdomain Length SL	6	13
Does it have WWW prefix(HwP)	0	0
Does it have a valid top-level domain	1	1
Contain Single-Character Subdomain	0	0
Does it contain the IP address	0	0
Does it contain the digit	0	0
Ratio of consonant	4/6	11/13
Ratio of vowel	2/6	2/13

Table 1: Analyse Domain Name

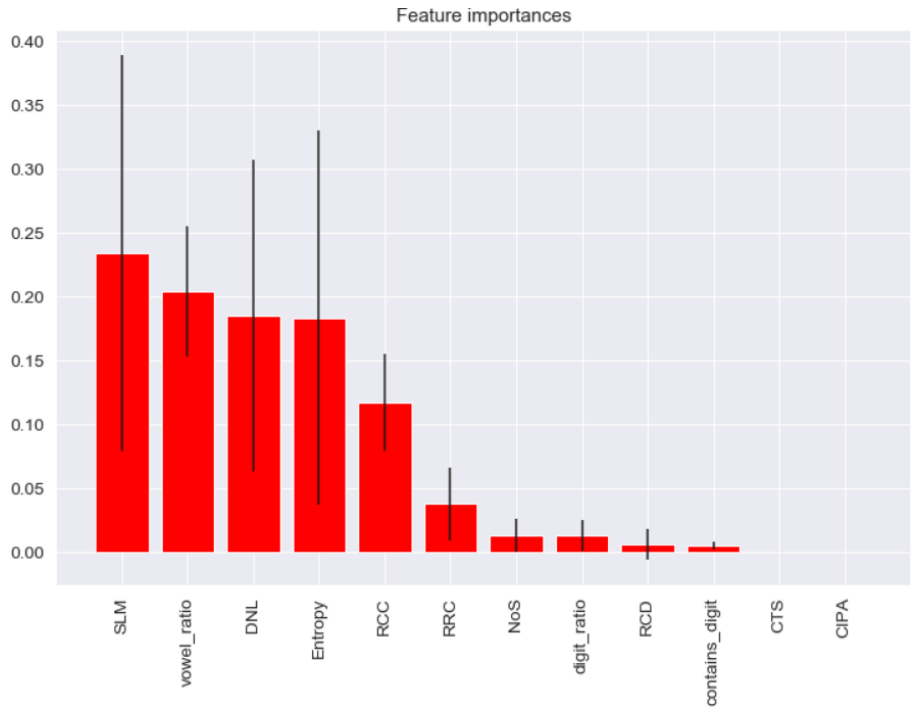

Figure 5: The importance of attributes in order.

As can be seen in the chart above, Subdomain Name Length and vowel ratio, Domain Name Length are the most important most influential attributes leading to assessing whether the source name is a bad or good domain name.

3.4. Building Machine Learning algorithm model

In this essay to apply for the domain exploitation by Logistic regression, Random Forest and Naive Bayer algorithms, I have built the following model to make the essay easier to understand.

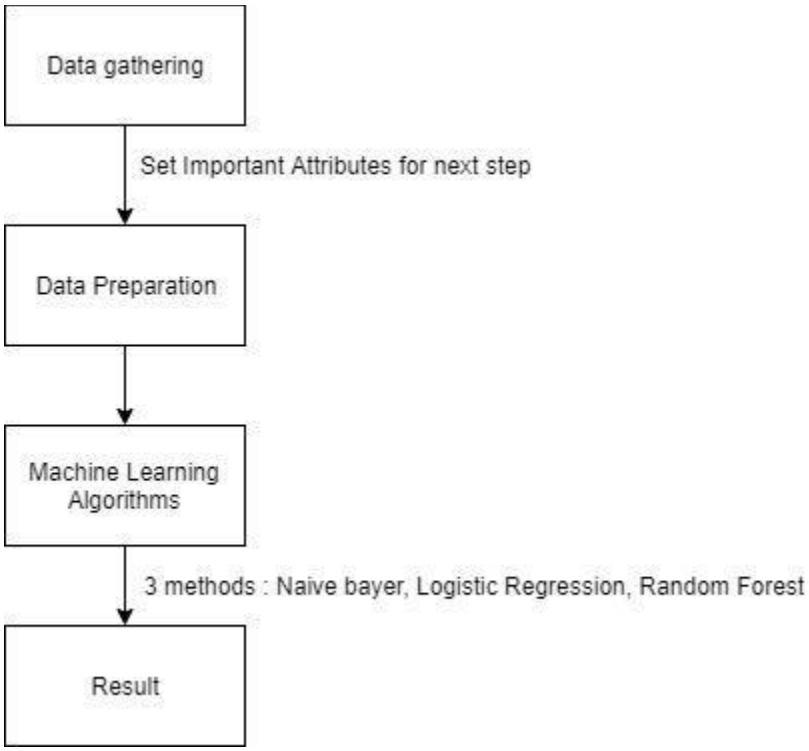

Figure 6: Machine Learning Model

Briefly explain the model on the stored data and be prepared by analyzing the domain to remove non-important information, then use the three algorithms as shown in the figure to exploit important attributes to give the rate of detecting malicious domain names.

3.5. System Model

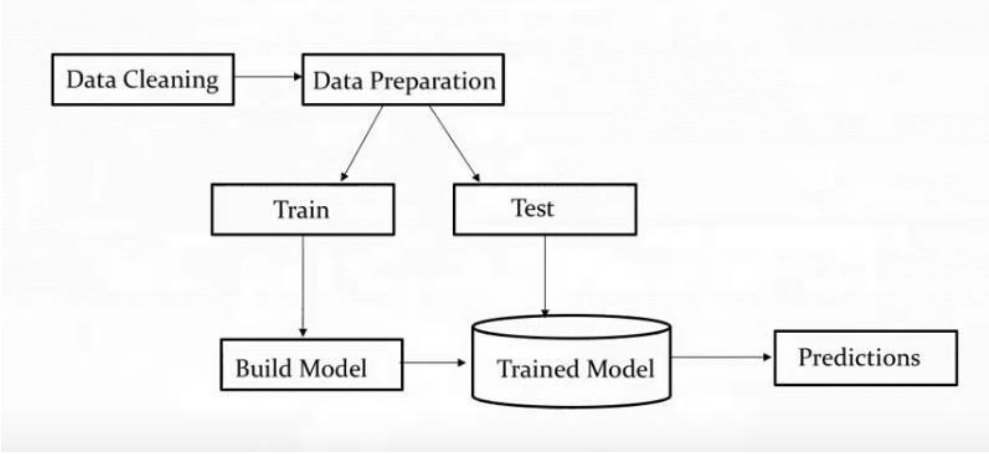

Figure 7: System Model

The data, including bad and good malware, was broken down for testing by two groups of Training Dataset and Testing Dataset at the rate of 80% and 20%.

The division of data for experimentation aims to be more satisfactory and tends to be more extensive. After completing this step, we will apply the algorithms to the support framework to get results.

3.6. Processing:

The algorithm implementation is built through Jupyter Lab. Jupyter is a free and open-source tool aimed at data science and education, making it easier for everyone to learn Python. Jupyter is interactive so it can be used as a testing and teaching environment. Jupyter is a tool that allows you to put both Python code and complex text elements such as images, formulas, videos, expressions ... into one file to make the presentation easier to understand, just like a presentation file but can run interactive code on it. The implementation of ML algorithms on Jupyter will make it easier to track the process and results while also fixing errors but also simpler.

3.7. Results :

- Implementing the first Logistic Regression algorithm obtained Training Dataset result of 84.92% of which Testing Dataset is 84.91%. This algorithm takes 30 seconds to produce a result. With a result above 80%, this algorithm is relatively safe and takes less time to use.

```
# Calculate the accuracy
score_lg_train = round(accuracy_score(train_y, train_lg_pred) * 100, 2)
score_lg_test = round(accuracy_score(test_y, test_lg_pred) * 100, 2)
print("Accuracy of Logistic Regression on training dataset: ", score_lg_train)
print("Accuracy of Logistic Regression on test dataset: ", score_lg_test)
```

```
Accuracy of Logistic Regression on training dataset:  84.92
Accuracy of Logistic Regression on test dataset:  84.91
```

Figure 8: Result Logistic Regression Algorithm

- The results obtained after applying the Random Forest algorithm are impressive with 93.81% for the Training dataset and 93.44 for Testing Dataset. However, to produce such results, the time it takes for the process to be executed usually takes very long. The algorithm is very time-consuming with 5 minutes and 15 seconds of processing. The result is completely worth the processing time.

```
# Calculate the accuracy
score_rf_train = round(accuracy_score(train_y, train_rf_pred) * 100, 2)
score_rf_test = round(accuracy_score(test_y, test_rf_pred) * 100, 2)
print("Accuracy of Random Forest Model on training dataset: ", score_rf_train)
print("Accuracy of Random Forest Model on test dataset: ", score_rf_test)
```

```
Accuracy of Random Forest Model on training dataset:  93.81
Accuracy of Random Forest Model on test dataset:  93.44
```

Figure 9: Result Random Forest

- The Naive Bayes algorithm is implemented in 3 directions and each direction has different results.
    - Gaussian Naive Bayes gave the training dataset result of 66.52 and testing dataset of 66.6
    - More impressive is the Multinomial Naive Bayes with similar results for both training dataset and testing dataset: 79.94
    - Bernoulli Naive Bayes gave a similar result to Gaussian, which is 62.73 for training dataset and 62.71 for testing dataset.
    - All three algorithms of Naive Bayes process in a very fast time of less than 1s. This is a fast algorithm but not highly effective.

```
# Caluate the accuracy
score_gnb_train = round(accuracy_score(train_y, train_gnb_pred) * 100, 2)
score_gnb_test = round(accuracy_score(test_y, test_gnb_pred) * 100, 2)
print("Accuracy of Gaussian Naive Bayes on training dataset: ", score_gnb_train)
print("Accuracy of Gaussian Naive Bayes on test dataset: ", score_gnb_test)
```

```
Accuracy of Gaussian Naive Bayes on training dataset:  66.52
Accuracy of Gaussian Naive Bayes on test dataset:  66.6
```

```
# Calculate the accuracy
score_mnb_train = round(accuracy_score(train_y, train_mnb_pred) * 100, 2)
score_mnb_test = round(accuracy_score(test_y, test_mnb_pred) * 100, 2)
print("Accuracy of Multinomial Naive Bayes on training dataset: ", score_mnb_train)
print("Accuracy of Multinomial Naive Bayes on test dataset: ", score_mnb_test)
```

```
Accuracy of Multinomial Naive Bayes on training dataset:  79.94
Accuracy of Multinomial Naive Bayes on test dataset:  79.94
```

```
# Calculate the accuracy
score_bnb_train = round(accuracy_score(train_y, train_bnb_pred) * 100, 2)
score_bnb_test = round(accuracy_score(test_y, test_bnb_pred) * 100, 2)
print("Accuracy of Bernoulli Naive Bayes on training dataset: ", score_bnb_train)
print("Accuracy of Bernoulli Naive Bayes on test dataset: ", score_bnb_test)
```

```
Accuracy of Bernoulli Naive Bayes on training dataset:  62.73
Accuracy of Bernoulli Naive Bayes on test dataset:  62.71
```

Figure 10: Result from Naïve Bayes

The chart below illustrates the precision that each method has shown. Looking at the chart, the Random Forest brings the most accurate results and close to this algorithm is Logistic Regression. Bayes Naive has not yet produced reliable results for classifying malicious domains.

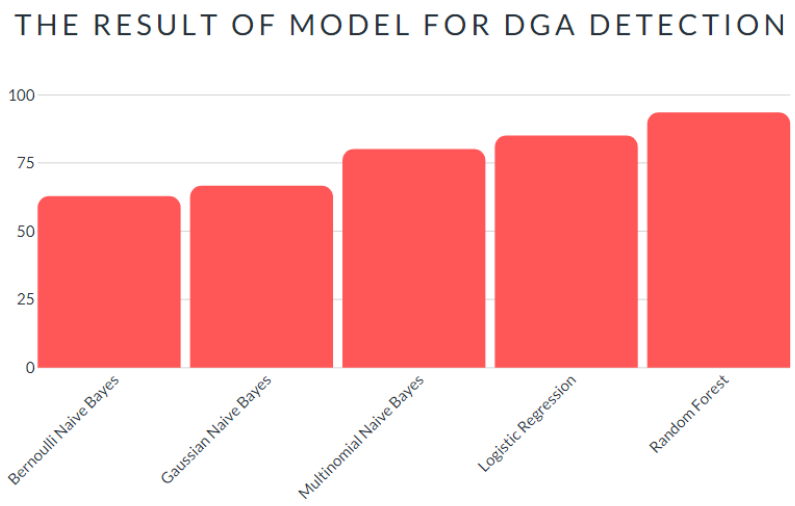

Figure 11: Ranking of accuracy results when detecting malicious domain names of machine learning methods.

4. Conclusion, Discussion and Future Work

The growing situation of malware infiltration systems through domain names is increasing, so we need to be careful and evaluate solutions to protect network information. In this essay, I have analyzed machine learning techniques to provide relevant evaluation analysis along with the application of basic machine learning techniques to detect malicious domain names. Through researching with JupyterLab on the Python framework, I discovered malicious domain names at a high rate thanks to the Random Forest method. This is an attractive method at present, the disadvantage of this method is only more time consuming but brings a very good effect. By detecting malicious code soon, we will prevent network security concerns.

In the future, with updated learning techniques and prior knowledge, I will delve deeper into Deep Learning, one of the studies that can help detect malicious domain names. with greater accuracy and enhanced performance as well as more cost-effective research.

5. Reference

[1] C. Crane, "New Malware: The Landscape of New & Evolving Cyber Threats in 2019 - Hashed Out by The SSL Store™", *Hashed Out by The SSL Store™*, 2019. [Online]. Available: https://www.thesslstore.com/blog/new-malware-the-landscape-of-new-evolving-cyber-threats-in-2019/. [Accessed: 07- Dec- 2019].

[2] Jordan, M. I., & Mitchell, T. M. (2015). *Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255–260.* doi:10.1126/science.aaa8415

[3] J. van Dam and M. van de Velden, "Online profiling and clustering of Facebook users", *Decision Support Systems*, vol. 70, pp. 60-72, 2015. Available: 10.1016/j.dss.2014.12.001.

[4]Y. YAN, Z. LIU, J. ZHONG, D. CHENG, J. XUE, and Y. WANG, "Malicious Domain Detection Based on Machine Learning", *DEStech Transactions on Computer Science and Engineering*, no., 2018. Available: 10.12783/dtcse/iceit2017/19866.

[5]O. Dubey, "Man In The Middle: DNS Spoofing", *Medium*, 2019. [Online]. Available: https://blog.usejournal.com/man-in-the-middle-dns-spoofing-df77ab2cae35. [Accessed: 08- Dec- 2019].

[6]S. Chowdhury, "Malware Hunter - Detects Command and Control (C&C) Server! - hackersterminal.com", *hackersterminal.com*, 2019. [Online]. Available: https://hackersterminal.com/malware-hunter-powered-by-shodan/. [Accessed: 08- Dec- 2019].

[7]J. JIANG, J. ZHUGE, H. DUAN and J. WU, "Research on Botnet Mechanisms and Defenses", *Journal of Software*, vol. 23, no. 1, pp. 82-96, 2012. Available: 10.3724/sp.j.1001.2012.04101.

[8] Tianyu Wang, Li-Chiou Ch, "Detecting Algorithmically Generated Domains Using Data Visualization and N-Grams Methods", Proceedings of Student-Faculty Research Day, 2017.

[9] E. Durmaz, DGA classification and detection for automated malware analysis, [Online] https://cyber.wtf/ 2017/08/30/dga-classification-and-detection-for-automated-malware-analysis/,.

[10]]W. SONG and B. LI, "A Method to Detect Machine Generated Domain Names Based on Random Forest Algorithm", 2016. Available: https://ieeexplore.ieee.org/document/7816767. [Accessed 16 January 2017].

[11]X. Hoang and Q. Nguyen, "Botnet Detection Based On Machine Learning Techniques Using DNS Query Data", *Future Internet*, vol. 10, no. 5, p. 43, 2018. Available: 10.3390/fi10050043.

[12]X. Vũ and X. Hoàng, "PHÁT HIỆN DGA BOTNET SỬ DỤNG KẾT HỢP NHIỀU NHÓM ĐẶC TRƯNG PHÂN LOẠI TÊN MIỀN", *FAIR 2019*, 2019. Available: https://www.researchgate.net/publication/333132204_PHAT_HIEN_DGA_BOTNET_SU_DUNG_KET_HOP_NHIEU_NHOM_DAC_TRUNG_PHAN_LOAI_TEN_MIEN. [Accessed 6 July 2019].

[13]C. Choudhary, R. Sivaguru, M. Pereira, B. Yu, A. C. Nascimento and M. De Coc, "Algorithmically Generated Domain Detection and Malware Family Classification", 2019. Available: https://www.researchgate.net/publication/330588285_Algorithmically_Generated_Domain_Detection_and_Malware_Family_Classification_6th_International_Symposium_SSCC_2018_Bangalore_India_September_19-22_2018_Revised_Selected_Papers. [Accessed 19 September 2018].

[14]Y. YAN, Z. LIU, J. ZHONG, D. CHENG, J. XUE, and Y. WANG, "Malicious Domain Detection Based on Machine Learning", *DEStech Transactions on Computer Science and Engineering*, no., 2018. Available: 10.12783/dtcse/iceit2017/19866.

[15]H. Mac, D. Quang Tran, V. Tong, G. Nguyen and H. Tran, "DGA Botnet Detection Using Supervised Learning Methods", 2017. Available: https://www.researchgate.net/publication/321741275_DGA_Botnet_Detection_Using_Supervised_Learning_Methods. [Accessed 11 December 2017].

[16]H. Ho Duc and H. Ho Van, "Technical research of detection algorithmically generated malicious domain names using machine learning methods", 2019. Available: http://antoanthongtin.vn/Detail.aspx?CatID=afad3c1b-8ab0-41b3-9364-fe76366f1531&NewsID=58e8d3ad-6001-498f-bae9-fbac4a851b9b. [Accessed 22 April 2019].

[17]"baderj/domain_generation_algorithms", *GitHub*, 2019. [Online]. Available: https://github.com/baderj/domain_generation_algorithms. [Accessed: 05- Nov- 2019].

[18]A. Pant, "Introduction to Logistic Regression", *Medium*, 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148. [Accessed: 29- Sep- 2019].

[19]T. Yiu, "Understanding Random Forest", *Medium*, 2019. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. [Accessed: 06- Sep- 2019].

[20]R. Gandhi, "Naive Bayes Classifier", *Medium*, 2019. [Online]. Available: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c. [Accessed: 03- Dec- 2019].