Anomaly detection based on machine learning techniques

By

Ramesh Babu Dilli

B.Tech., RVR&JC College of Engineering, India, 2017

A REPORT

submitted in partial fulfilment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2019

Approved by:

Major Professor

Dr. William H. Hsu

ABSTRACT

This report presents an experimental exploration of supervised inductive learning methods for the task of Domain Name Service (DNS) query filtering for anomaly detection. The anomaly types for which I implement a learning monitor represent specific attack vectors, such as distributed denial-of-service (DDOS), remote-to-user (R2U), and probing, that have been increasing in size and sophistication in recent years. A number of anomaly detection measures, such as honeynet-based and Intrusion Detection System (IDS)-based, have been proposed. However, IDS-based solutions that use signatures seem to be ineffective, because attackers associated with recent anomalies are equipped with sophisticated code update and evasion techniques. By contrast, anomaly detection methods do not require pre-built signatures and thus have the capability to detect new or unknown anomalies. Towards this end, this project implements and applies an anomaly detection model learned from DNS query data and evaluates the effectiveness of an implementation of this model using popular machine learning techniques. Experimental results show how this machine learning approach uses existing inductive learning algorithms such as k-NN (k-nearest neighbour), Decision trees and Naive Bayes can be used effectively in anomaly detection.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my committee members Dr.William H. Hsu, Dr. Torben Amtoft, and Dr. Lior Shamir for taking time to serve on my committee and their support in the process of this project.

I am especially indebted to my major advisor Dr. William H. Hsu for believing in my abilities and for his constant support from my very first semester here at Kansas State University.

I am also grateful to Dr.Jorge Valenzuela, Dr. Eugene Vasserman, and Ms. Julie Thornton for being amazing mentors during my semesters as a teaching assistant.

I would like to extend a huge thanks to all my family members without whom this journey would not have been possible. I am extremely grateful to my father, Venkateswara Rao Dilli, for his encouragement and support in my life. I thank my brother, Ugandhar Delli who is also graduated from this university for his valuable suggestions and support.

And last, but not least, I would like to thank my friends Venkat Surya, Dishan Nahitya, Nitesh Verma, Rakshandha Reddy for their support and friendship. The past two years would not have been memorable and fun without all of you

CHAPTER 1-Introduction

This project addresses a broad of supervised machine learning tasks for classification of network anomalies, an approach that has been studied in security even as the volume and variety of cybersecurity threats and internet-based attack vectors proliferates. The mass usage of computerized systems has given rise to critical threats such as zero-day vulnerabilities, mobile threats, etc. Despite research in the security domain having increased significantly, are yet to be mitigated. The evolution of computer networks has greatly exacerbated computer security concerns, particularly internet security in today's networking environment and advanced computing facilities. Although Internet Protocols (IPs) were not designed to place a high priority on security issues, network administrators now have to handle a large variety of intrusion attempts by both individuals with malicious intent and large botnets. According to Symantec's Internet Security Threat Report, there were more than three billion malware attacks reported in 2010 and the number of denial of service attacks increased dramatically by 2013 (Symantec internet security threat report, 2014). As stated in Verizon's Data Breach Investigation Report 2014, 63,437 security breaches carried out by hackers (Verizon's data breach investigation report, 2014). The Global State of Information Security Survey 2015 (The Global State of Information Security Survey, 2015) found an increase in great rise of incidents. Figure 1 shows the security incidents growth from 2009 to 2018. Therefore, the detection of network attacks has become the highest priority today. In addition, the expertise required to commit cyber-crimes has decreased due to easily available tools (Hacking and cracking tools, 2014).

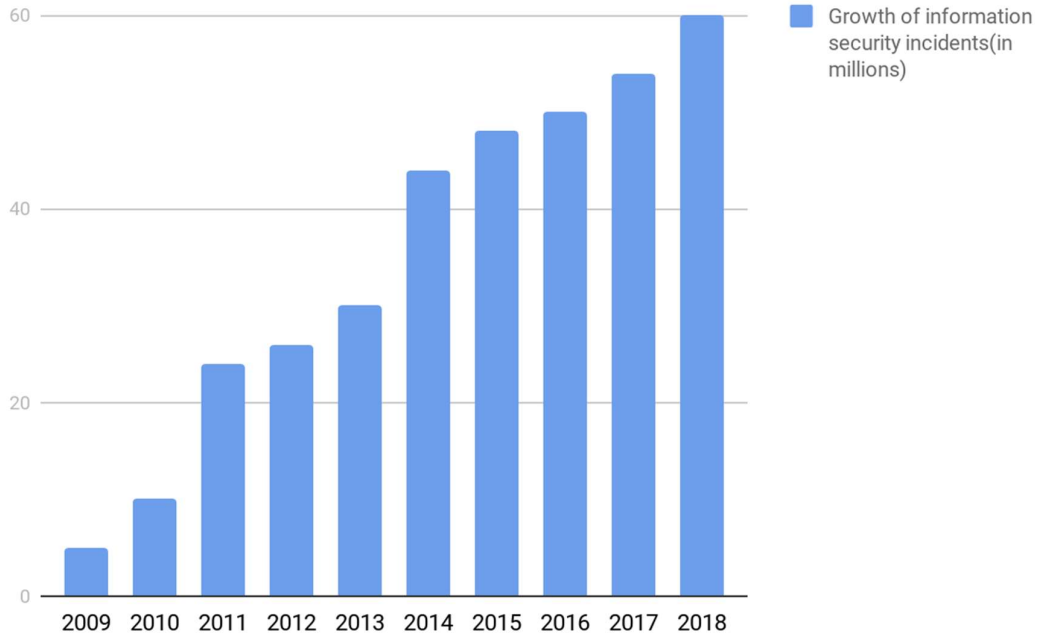

Figure 1.1: Information security incidents per year, 2009 – 2018 (count in millions).

1.1 Types of anomalies

Anomalies are defined as patterns in data that do not conform to a well-defined characteristic of normal patterns. They are generated by a variety of abnormal activities, e.g., credit card fraud, mobile phone fraud, cyberattacks, etc., which are significant to data analysts. An important aspect of anomaly detection is the nature of the anomaly. An anomaly can be categorized in the following ways.

1.1.1 Point anomaly. (Introduction to Anomaly Detection: Concepts and Techniques 2015)
When a particular data instance deviates from the normal pattern of the data set, it can be considered a point anomaly. For a realistic example, if a persons' normal car fuel usage is five litres per day but if it becomes fifty litres in any random day, then it is a point anomaly.

1.1.2 Contextual anomaly. (Introduction to Anomaly Detection: Concepts and Techniques 2015)
When a data instance behaves anomalously in a particular context, it is termed a contextual or conditional anomaly; for example, expenditure on a credit card during a festive period, e.g., Christmas or New Year, is usually higher than during the rest of the year. Although it can be high,

it may not be anomalous as high expenses are contextually normal in nature. On the other hand, an equally high expenditure during a non-festive month could be deemed a contextual anomaly.

1.1.3 Collective anomaly. (Introduction to Anomaly Detection: Concepts and Techniques 2015) When a collection of similar data instances behaves anomalously with respect to the entire data set, the group of data instances is termed a collective anomaly. For example, in a humans Electrocardiogram (ECG) output, the existence of low values for a long period of time indicates an outlying phenomenon corresponding to an abnormal premature contraction (Lin et al., 2005) whereas one low value by itself is not considered anomalous.

1.2 Types of network attacks

The task of network security is to protect digital information by maintaining data confidentiality and integrity, and ensuring the availability of resources. In simple terms, a threat/attack refers to anything which has detrimental characteristics aimed at compromising a network. The poor design of a network, carelessness of its users and/or mis-configuration of its software or hardware can be vulnerable to attacks.

1.2.1 Denial of service. (A survey on network attacks and Intrusion detection systems 2017) (DoS) is a type of misuse of the rights to the resources of a network or host which is targeted at disrupting the normal computing environment and rendering the service unavailable. A simple example of a DoS attack is denying legitimate users access to a web service when the server is flooded with numerous connection requests. As performing a DoS attack requires no prior access to the target, it is considered a dreaded attack.

1.2.2 Probe. (A survey on network attacks and Intrusion detection systems 2017) A probe is a software instrument used to gather information about a targeted network or host and, more formally, for reconnaissance purposes. Reconnaissance attacks are quite common ways of gathering information about the types and numbers of machines connected to a network, and a

host can be attacked to determine the types of software installed and/or applications used. A Probe attack is considered the first step in an actual attack to compromise a host or network. Although no specific damage is caused by these attacks, they are considered serious threats to corporations because they might obtain useful information for launching another dreadful attack.

1.2.3 User to Root (U2R). (A survey on network attacks and Intrusion detection systems 2017) This is an attack launched, when an attacker aims to gain illegal access to an administrative account to manipulate or abuse important resources. Using a social engineering approach or sniffing the password, the attacker can access a normal user account and then exploits a or some vulnerability to gain the privilege of a super user.

1.2.4 Remote to User (R2U). (A survey on network attacks and Intrusion detection systems 2017) This is launched when an attacker wants to gain local access as a user of a targeted machine to have the privilege of sending packets over its network (also known as R2L). Most commonly, the attacker uses a trial and error method to guess the password by automated scripts, a brute force method, etc. There are also some sophisticated attacks whereby an attacker installs a sniffing tool to capture the password before penetrating the system.

1.3 Domain Name Service (DNS)

A DNS is an essential service on the Internet that allows the resolution of hostnames (or domain names) to Internet Protocol (IP) addresses, and vice versa. For example, every time a web client browser accesses a web page, it first sends a request to the DNS system to find the IP address of the web server, and then it uses the IP address to access the web server and load the requested web page. As such, most legitimate applications use DNS services when making requests for accessing network services. However, DNS services are also used by bots of botnets as legitimate applications. Bots send DNS queries to find the IP address of the Command and Control (C & C) server and when they have an IP address, they access the C & C server to receive commands, as

well as to download the updated bot code. To evade the scanning and detecting of C & C servers, the botmaster is constantly changing the names and IP addresses of C & C servers using predefined techniques, such as Domain generation algorithm (DGA), or Fast flux (FF). The names and IP addresses of C & C servers are constantly being pushed to the DNS system. Bots are also equipped with the ability to automatically generate C & C server names in accordance with these techniques. As a result, bots can still find IP addresses of C & C servers by generating their hostnames automatically and using these hostnames to query the DNS service. Therefore, monitoring and analysing DNS query data can reveal the existence of malicious activities in the monitored network, since some of the DNS query data may be generated by botnets. An in-depth analysis of suspicious DNS queries may reveal valuable information regarding the presence of C & C servers and botnets.

This project examines and evaluates the effectiveness of the bonnet detection method using DNS query data based on a number of commonly used machine learning techniques, including k-NN, Decision trees, Random forest, Naïve Bayes and Logistic Regression. Based on the evaluation results, we propose a bonnet detection model based on machine learning using DNS query data. The rest of the paper is structured as follows: Chapter 2 presents background and related works and Chapter 3 first introduces some common machine learning techniques and then describes the machine learning based botnet detection model. Chapter 4 describes the experiments and evaluation and Chapter 5 describes experimental outcomes, results, and their interpretation and Chapter 6 presents conclusions and summarizes future work and Chapter 7 lists references.

CHAPTER 2- Background and Related Work

Due to the robust development of botnets and the information security risks that they may cause, the detection of botnets in the networks is a hot research topic. Currently, there are two main directions in the botnet detection research, including honeynet-based techniques and IDS-based techniques. The techniques in direction usually build honeynets to collect botnet information, and then analyse the characteristics and behaviours of botnets. In general, honeynet-based botnet detection systems have the advantages of being easy to build and requiring less computing resources. However, these systems are often limited in their ability to scale and interact with malicious behaviours. In addition, hackers can use honey-nets to develop new evasive techniques. The direction used in the botnet detection is based on the IDS. An IDS is a software application, or a hardware device that monitors networks or systems to detect malicious behaviours, or the violation of security policies and to notify the administrators. In particular, IDSs are usually installed to monitor data packets transmitted over a network gateway, or events occurring on a system, and then to conduct the analysis to look for the signs of a botnet. IDS-based botnet detection techniques are divided into two groups: signature-based detection and anomaly-based detection. In the anomaly-based IDS botnet detection group, the botnet detection research based on the analysis of DNS queries is one of the most promising approaches. The following part of this section describes some typical solutions.

The proposed solutions for detecting botnets based on the collection and analysis of DNS queries have been published quite abundantly. In particular, Ramachandran et al. [5] proposed counter-intelligence techniques for detecting botnet members by monitoring DNS blacklist (DNSBL) database queries from botnet owners to determine if bot members are on the sending spam blacklist. Research results show that the proposed techniques are capable of accurately detecting botnets' spam bot members.

An interesting approach to discovering botnets is to detect botnets by monitoring queries from bots sending to DNS systems to find IP addresses of C & C servers to download the commands and bot updated code. By monitoring DNS requests, one can detect the existence of bots and botnets. In this direction, Villamari-Salomo et al. [6] proposed the botnet identification techniques based on the anomaly detection by monitoring DNS queries. The proposed approach attempts to detect domain names with very high query rates and non-existence domain names. This is feasible because there are a large number of bots that query the DNS service for IP addresses of C & C servers at the same period of time, and many of them have not yet been updated so they still query the terminated domains More broadly, Perdisci et al. [7] proposed a method for monitoring recursive DNS queries to detect malicious services related to botnet-controlled computers, such as junk blogging, junk messaging and spamming. The test results on two actual Internet Service Provider (ISP) networks in 45 days suggest that the proposed method accurately detects the malicious services.

Regarding the monitoring of DNS queries for botnet detection, there have been some proposed solutions for detecting domain names that are automatically generated using algorithms, or that are maliciously used. Accordingly, Yadav et al. [8] proposed a method for distinguishing algorithm generated domain names commonly used in botnets and legitimate domain names using the character distribution in the domain names. Similarly, Stalmans et al. [9] used the C5.0 decision tree method and Bayesian statistics to classify automatically generated domain names with legitimate domain names. Also for the purpose of detecting domain names associated with botnets, Antonakakis et al. [10] proposed a novel detection system, called Kopis. Kopis monitors high-level DNS queries in the DNS hierarchy and analyses the patterns of global DNS query responses for detecting malware-related domain names. The test results for Kopis show that the system achieves more than 98% accuracy and under 0.5% false detection rates. Kopis also has the ability

to detect malware-related domain names many days before they are included in the public blacklists.

Using a different approach, Bilge et al. [11] introduced the EXPOSURE system that allows for the extensive monitoring of DNS traffic to detect domain names that are associated with malicious behaviours. EXPOSURE uses 15 features to distinguish suspicious domain names from legitimate domain names. The test results show that the system is scalable and can recognize new domain names related to malicious behaviours, as they are used for C & C servers, for spamming and phishing websites. For the purpose of detecting suspicious and malicious behaviours originating from botnets, Jiang et al. [12] proposed a method for identifying suspicious behaviours based on the analysis of failed DNS queries using a tool called DNS Failure Graphs.

DNS failure queries are queries with non-existent domain names, expired domain names, or errors in the DNS system. This suggestion comes from the fact that suspicious behaviours triggered by spamming, trojans' activities, especially bot activities include DNS lookup queries for IP addresses of mail servers, or C & C servers. A significant number of these queries are DNS lookup queries that fail because of the non-existence domain names or expired domain names. Test results on a three-month DNS query data set of a large network indicate that the proposed method can identify new behavioural abnormalities with a high probability of originating from unknown bots.

In the opposite direction, Kheir et al. [13] proposed a system called Mentor that allows the removal of legitimate domain names from the blacklist of domain names of C & C servers using a positive reputation-based DNS system. Mentor uses a crawler and statistical characteristics of site content and domain names to categorize legitimate domain names and suspicious domain names based on supervised learning. Experimental results on some botnet domain blacklists show that the system is capable of accurately identifying legitimate domain names with low error rates.

In this project, I am examining and evaluating the performance of the bonnet detection method using DNS query data based on several supervised machine learning techniques, including

k-NN, decision trees, random forest, Naive Bayes and Logistic Regression. Based on the results, I propose the botnet detection model based on machine learning using DNS query data.

CHAPTER 3-Machine Learning techniques and Detection Model

Machine learning is a field of computer science, involving the study and construction of techniques that enable computers to self-study based on the input data to solve specific problems. Based on the learning methods, machine learning techniques are usually divided into three main categories: supervised learning, unsupervised learning and semi-supervised learning. Supervised learning is a form of learning in which training data is labelled. The machine will “learn” from the labelled patterns to build the classifier and use it to predict labels for new data. In contrast, unsupervised learning is a form of learning in which training data has not been labelled. In this form, the machine will “learn” by analysing the data characteristics to construct the classifier. Semi-supervised learning is the hybrid approach between supervised and unsupervised learning. In semi-supervised learning, the input training data set will contain both labelled and unlabelled data. Each method has its own advantages and disadvantages and has its own application domain. In this paper, we only examine the effectiveness of supervised learning techniques in botnet detection and the next subsection briefly describes some of the common supervised machine learning algorithms, including k-nearest neighbour (k-NN), decision tree, random forest, and Naive Bayes.

3.1 k-NN

k-NN (k-Nearest Neighbor) is one of the simplest supervised machine learning algorithms. The idea of k-NN is that it will classify the new object based on its k nearest neighbours, where k is a predefined positive integer. The k-NN algorithm is comprehensively described in the following steps:

Step 1: Determine the parameter value k.

Step 2: Calculate the distance between the new object that needs to be classified with all objects in the training data set.

Step 3: Arrange computed distances in the ascending order and identify k nearest neighbours with the new object.

Step 4: Take all the labels of the k neighbours selected above.

Step 5: Based on the k labels that have been taken, the label that holds the majority will be assigned to the new object.

3.2 Decision tree

Decision tree is a prediction model that is a mapping from observations of a thing, or a phenomenon to the conclusions about the objective value of things, or phenomena. Decision tree creates models that allow the classification of an object by creating a set of decision rules. These rules are extracted based on the set of characteristics of the training data. In a decision tree, leaves represent classes and each child node in the tree and its branches represent a combination of features that lead to classification. Thus, classifying an object will begin with checking the value of the root node, and then continuing downward under the tree branches corresponding to those values. This process is performed repeatedly for each node until it cannot go any further and touch the leaf node. For the best model, the decision to select the root node and sub node while building the decision tree is based on the information gain (IG) and Gini impurity measurements. The decision tree algorithm used for experiments in this project is C4.5.

3.3 Random forest

Random forest (RF) is a member of the chain of decision tree algorithms. The idea of this algorithm is to create some decision trees. These decision trees will run and give independent results. The answer predicted by the largest number of decisive trees will be chosen by the random forest. To ensure that the decision trees are not the same, random forest randomly selects a subset of the characteristics of each node. The remaining parameters are used in the random forest as those in the decision trees.

3.4 Naïve Bayes

Naive Bayes is a conditional probability model based on the Bayes theorem. In the classification problem, the data includes: D is the set of training data that has been vectorized as → x = (x1, x2, ..., xn), Ci is subclass i, i = {1, 2, ..., m}. Assume that all properties are conditionally independent of each pair together. According to Bayes theorem, we have:

$$P(C_i \mid X) = P(X \mid C_i) * P(C_i)/P(X)$$

Equation 3.1 Equation of probability of class i

Based on the conditionally independent characteristic, we have:

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

Equation 3.2 Equation of Naive Bayes

where, $P(C_i \mid X)$ is the probability of class i given sample X,$P(C_i)$ is the probability of class i and $P(x_k|C_i)$ is the probability of the kth property that has the value of $x_k$given X in class i.

3.5 Logistic Regression

The logistic regression is a regression analysis that uses the logistic function to map the one or more nominal, ordinal or interval independent variable to a dependent categorical variable. The dependent output variable has categories like “yes” or “No”, “1” or “0”, “drop out” or “not dropped out”. The input variables can be independent and the values for the input variables can be of any type.

In machine learning, a sigmoid function is used to map the predictions to the probabilities. This function helps in mapping any real values to real values in the range 0 to 1.

$$\boldsymbol{S(z) = \frac{1}{1+e-z}}$$

Equation 3.3 Sigmoid Function

S(z) is the output variable that needs to be predicted using the algorithm. The z is the input variables for the function. E is the base of the natural log.

3.6 Lasso Regression

Lasso regression is explained as Least absolute shrinkage and Selection operator (Saptashwa, n.d.). Lasso regression is a method of regression analysis that helps in Feature selection and a regularization technique. This model is used to improve the prediction accuracy using the L1 regularization technique. Regularization techniques like Lasso regression are used to solve the problem of overfitting. Overfitting happens when the model learns from the noise and signal in the training data and becomes a bad predicting model for the testing data i.e. The data on which it is not trained. Regularization is used when there is complexity in the model and the parameters should be penalized. Regularization penalizes the parameters so that they won't overfit. It adds the "absolute values of magnitude" of coefficient as penalty term to the loss function.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=0}^{p}|\beta_j|$$

Equation 3.4 Equation of Lasso Regression

3.7 Ridge Regression

Ridge Regression is also another regularization technique that uses the L2 regularization technique. Ridge regression adds the "squared magnitude of the coefficient" as a penalty to the loss function. If the lambda value is too large it adds importance to the feature resulting in underfitting solves the problem of overfitting. The features importance can be reduced by removing the features completely by forwarding or backward selection or setting their value to 0. By this way, we can never predict the effect of the removed variable on the dependent variable. So, Ridge regression helps to set the value of the coefficient of these features if they are far from the common values. Since it shrinks the parameters it is used to prevent the multicollinearity.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=0}^{p}{\beta_j}^2$$

Equation 3.5 Equation of Ridge Regression

3.7 Proposed Botnet Detection Model Based on Machine Learning

Figure 3.1 describes the proposed botnet detection model based on machine learning using DNS query data. The model is built on the analysis in Section 1 that bots of botnets routinely send lookup queries to the DNS system to find IP addresses of C & C servers using automatically generated domain names. The detection model is implemented in two phases: (a) the training phase and (b) the detection phase. During the training phase, the DNS query data is collected, and then domain names in DNS queries are extracted. Next, the set of domain names is pre-processed to extract the features for the training. In the training phase, machine learning algorithms are used to learn the classifiers. Through the evaluation process, the machine learning algorithm that gives the highest overall classification accuracy will be selected for use in the proposed detection model. During the detection phase of the model, the DNS queries are monitored and passed through the process of extracting the domain names, pre-processing, and classifying using the classifier produced from the training phase to determine if a domain name is legitimate, or a botnet domain name. The pre-processing step for each domain name in the training and detection phase is the same. However, this step is done in the offline mode for all the domain names of the training data set in the training phase while it is done for each domain name extracted from the DNS query on the fly in the detection phase.

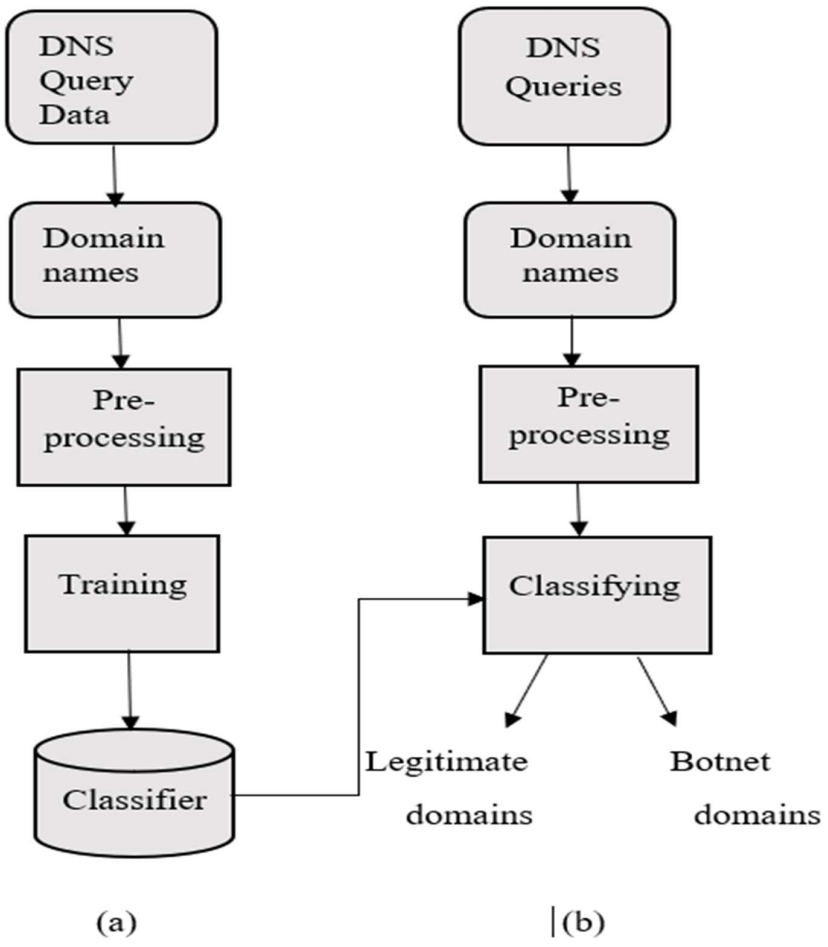

Figure 3.1: Proposed botnet detection model based on machine learning using Domain Name Service (DNS) query data: (a) training phase and (b) detection phase.

CHAPTER 4-Experiments and Evaluation

4.1 Experimental Data set

To evaluate the classification performance of botnet domain names using machine learning algorithms, we use the extracted and labelled domain name data sets, which include a set of benign domain names and a set of malicious domain names used by botnets. The set of benign domains includes 30,000 top domain names ranked by Alexa. Benign domain names are checked at the virustotal.com website to make sure they are really benign. The set of malicious domain names collected at [17,18] includes 30,000 domain names created by the Conficker botnet and the DGA botnet. From the original data sets, the domain names are processed to remove the top-level domain. For example, with the domain name “example.com”, after processing, the resulting data is “example”.

4.2 Data Pre-Processing

According to [8,9,14], automatically generated botnet domain names often have DNS characteristics, network characteristics and lexical features that are different from those of legitimate domain names. Yadav et al. [8] proposed to use the distribution analysis of vowels, numbers, and other characters in domain names to distinguish legitimate domain names and domain names generated by botnet algorithms. More broadly, Stalmans et al. [9] proposed to use two groups of domain names’ characteristics, including DNS characteristics (IP address, network address, etc.) and lexical features (the character distribution of the domain names). Meanwhile, da Luz [14] proposed to use 36 characteristics in two groups, including 18 vocabulary characteristics (mean, variance and standard deviation of 1-g, 2-g, 3-g and 4-g; entropy; characteristics of characters, numbers, vowels and consonants) and 18 network characteristics (TTL, number of network addresses, etc.). In this project, the focus is on exploiting the statistical vocabulary characteristics of 2-g and 3-g clusters, and the vowel distribution characteristics of domain names. Specifically, we use 16 statistical vocabulary features of 2-g (bi-gram) and 3-g (tri-gram) clusters,

and 2 vowel distribution characteristics to extract 18 features of each domain name for the training phase and also for the detection phase.

4.3 Features of Bi-gram and Tri-gram Clusters

Bi-gram is a cluster of two adjacent characters extracted from a string. For example, the domain name "example" (top level domain ".com" was removed) includes the following bi-grams: ex, xa, am, mp, pl and le. A domain name can contain characters in 26 alphabetical characters (a–z), numeric characters (0–9), "." and "-" characters. Therefore, the total number of possible bi-grams, TS(bi-gram) is 38 × 38 = 1444. From the set of benign domain names, we extract a list of N most frequently encountered bi-grams, denoted as DS(bi-gram). DS(bi-gram) is used for calculating 8 bi-gram related features of each domain name.

Tri-gram is a cluster of three adjacent characters extracted from a string. For example, the domain name "example" consists of the following tri-grams: exa, xam, amp, mpl and ple. Similar to the calculation of the total number of bi-grams, the total number of possible tri-grams, TS(tri-gram) is 38 × 38 × 38 = 54,872. From the set of benign domain names, we extract the list of M most frequently occurring tri-grams, denoted as DS(tri-gram). DS(tri-gram) is used for calculating 8 tri-gram related features of each domain name.

The features of bi-grams and tri-grams (we call both of them as n-gram) for each domain name d include:

- count(d) is the number of n-grams of the domain name d that are also found in DS(n-gram).
- m(d) is the general frequency distribution of the n-grams of the domain name d, which is calculated by the following formula:

$$m(d) = \sum_{i=0}^{count(d)} f(i) * index(i)$$

Equation 4.1: Equation of general frequency distribution

where f(i) is the total number of occurrences of n-gram i in DS(n-gram) and index(i) is the rank of n-gram i in TS(n-gram).

- s(d) is the weight of n-grams of the domain name d, calculated by the following formula:

$$s(d) = \sum_{i=0}^{count(d)} f(i) * \frac{vt(i)}{count(d)}$$

Equation 4.2: Equation of weight of n-grams

where vt(i) is the rank of n-gram i in DS(n-gram).

- ma(d) is the average of the general frequency distribution of the n-grams of the domain name d, which is calculated by the following formula:

$$ma(d) = \frac{m(d)}{len(d)}$$

Equation 4.3: Equation of average of the general frequency

where len(d) is total number of n-grams in the domain name d.

- sa(d) is the average of the weight of n-grams of the domain name d, calculated by the following formula:

$$sa(d) = \frac{s(d)}{len(d)}$$

Equation 4.4: Equation of average of the weight of n-grams

- tan(d) is the average number of popular n-grams of the domain name d, calculated by the following formula:

$$\tan(d) = \frac{count(d)}{len(d)}$$

Equation 4.5: Equation of average number of popular n-grams

- taf(d) is the average frequency of popular n-grams of the domain name d, calculated by the following formula:

$$taf(d) = \sum_{i=1}^{count(d)} \frac{f(i)}{len(d)}$$

Equation 4.6: Equation of average frequency of popular n-grams

- entropy(d) is the entropy of the domain name d, calculated by the following formula:

$$entrophy(d) = -\frac{vt(i)}{L} * \log\left(\frac{vt(i)}{L}\right)$$

Equation 4.7: Equation of entropy

where L is the number of popular n-grams in the set of benign domain names, L = N for bi-grams and L = M for tri-grams.

4.4 Vowel Distribution Features

According to [8,9], benign domain names often have higher vowel numbers than that of automatically generated domain names by botnets, because benign domain names are often built on grammatical features, while botnet automatically generated domain names usually consist of random characters and do not have meanings. Based on this idea, 2 vowel features of each domain name are built, including:

- countnv(d) is the number of vowels of the domain name d.
- tanv(d) is the average number of vowels of the domain name d, calculated by the following formula:

$$tanv(d) = \frac{countnv(d)}{len(d)}$$

Equation 4.8: Equation of average number of vowels

4.5 Feature Analysis

The feature analysis is an important task. The features which do not play any role in the prediction task or which does not help the model to learn or whose presence hinders the performance of the algorithm needs to be eliminated to improve the performance of the algorithm. Feature engineering is one of the important techniques that play a vital role in predicting the output. Various techniques are used to analyse the feature importance. Recursive feature elimination technique is used to estimate the importance of each feature. Depending on how feature importance is carried out the accuracy and precision of the model may increase or decrease. Regularization techniques like Lasso regression are also used to eliminate the features which play a negligible role in predicting the dependent variable.

Most of the DS(tri-gram) features and vowel distribution features are having higher values when compared DS(bi-gram) features and some of the features like ma(d), sa(d) are not playing any role in the prediction task. Therefore, the prediction of botnet domains would be more driven by DS(tri-gram) features and vowel distribution features.

CHAPTER 5-Experimental Scenarios and Results

5.1 Experimental Scenarios

After the pre-processing process, 18 features extracted from each domain name form a record in the experimental data set. We selected three training data sets, namely T1, T2 and T3, and a testing data set, called TEST from the experimental data set. The T1 set contains 10,000 records from benign domain names and 10,000 records from malicious domain names. The T2 set contains 5000 records from benign domain names and 15,000 records from malicious domain names and the T3 set contains 15,000 records from benign domain names and 5000 records from malicious domain names. The selection of 3 training data sets with different numbers of benign and malicious domain names to assess their effect on detection performance. The TEST data set includes 10,000 records from benign domain names and 10,000 records from malicious domain names. Records from benign domain names are labelled “good” and records from malicious domain names are labelled “bad”.

Data sets for Training and Testing	Data set Labelled “Good”	Data set Labelled “Bad”
T1 training data set	10,000	10,000
T2 training data set	5000	15,000
T3 training data set	15,000	5000
TEST testing data set	10,000	10,000

Table 5.1 The training data sets and the testing data set

Data for testing is not part of the training sets. Table 1 describes the dimensions of the components of the training data sets and the testing data set.

5.2 Classification Measures

The classification measures used in our experiments include Precision, Recall, ACC (Accuracy) and F1 (F1 measure). These measures are computed using the following formulas:

$$Precision = \frac{TP}{(TP+\ )}$$

Equation 5.1: Definition of Positive Predictive Value

$$Recall = \frac{TP}{(TP+FN)}$$

Equation 5.2: Definition of True Positive Rate

$$ACC = \frac{(TP+TN)}{(TP+FP+TN+F\ )}$$

Equation 5.3: Definition of Accuracy

$$F1 = \frac{2TP}{(2TP+FP+FN)}$$

Equation 5.4: Definition of F1 measure

where, TP (True Positives) is the number of records labelled "bad" that are classified correctly, TN (True Negatives) is the number of records labelled "good" that are classified correctly, FP (False Positives) is the number of records labelled "good" that are misclassified to "bad" and FN

	Predicted BAD	Predicted GOOD
Actual BAD	TP (True Positives)	FP (False Positives)
Actual GOOD	FN (False Negatives)	TN (True Negatives)

Table 5.2 Confusion Matrix

(False Negatives) is the number of records labelled "bad" that are wrongly classified as "good". Table 2 provides the confusion matrix of TP, TN, FP, and FN.

For the k-NN machine learning algorithm, the model is tested with k = {from 1 to 20} with two cases of non-weighted and weighted neighbours. In particular, the weight is assigned based

on the principle: the smaller the distance the greater the weight. Based on the test results, we determine the coefficient k that gives the highest classification accuracy.

For the random forest algorithm, the general prediction result is based on the predictions of the decision trees. Therefore, the number of decision trees that may affect the accuracy of the algorithm. The model was tested with the number of decision trees of {from 5 to 50} to determine the number of trees that produce the highest classification accuracy.

5.3 Cross-validation

K-fold cross validation is one of the important and most used types of cross-validation. In this technique, the data set in randomly divided into k subsets. The model is trained with k-1 subsets and tested on the remaining 1 subset. This process is repeated k number of times each time considering a different subset for testing. 5-fold cross validation is preferred and is used in this project.

5.4 Experimental Results

The experimental results for my model using 5 machine learning algorithms on the T1, T2, T3 training data sets and the TEST testing data set described in Section 5.4 show that the k-NN algorithm's overall classification accuracy is better with weighted neighbours than with non-weighted neighbors. The k value that gives the best classification performance is not stable and it depends on the training sets. The best classification performance using the k-NN algorithm with k values of 13, 9 and 15 on the T1, T2 and T3 training sets, respectively.

Similar to k-NN, the number of decision trees of the random forest algorithm that produces the best classification results is also unstable and it depends on the training sets. The random forest algorithm gives the best classification performance with the number of decision trees of 30, 38 and 27 on the T1, T2 and T3 training data sets, respectively. Tables 5.3-5.5 provide the model's testing results using 6 machine learning algorithms on the T1, T2 and T3 training sets, respectively

Machine Learning Algorithms	Precision	Recall	Accuracy	F1
k-NN(k=13)	0.895	0.910	0.902	0.903
C4.5	0.891	0.912	0.901	0.901
RF	0.907	0.910	0.908	0.908
Naive Bayes	0.831	0.902	0.859	0.865
Logistic Regression- L1 Regularization	0.885	0.904	0.898	0.894
Logistic Regression- L2 Regularization	0.882	0.903	0.894	0.892

Table 5.3 Classification performance of the detection model using T1 training set

Machine Learning Algorithms	Precision	Recall	Accuracy	F1
k-NN(k=9)	0.827	0.964	0.881	0.89
C4.5	0.815	0.973	0.876	0.887
RF	0.842	0.966	0.892	0.8996
Naive Bayes	0.828	0.905	0.858	0.865
Logistic Regression- L1 Regularization	0.82	0.958	0.872	0.883
Logistic Regression- L2 Regularization	0.817	0.956	0.869	0.881

Table 5.4 Classification performance of the detection model using T2 training set

Machine Learning Algorithms	Precision	Recall	Accuracy	F1
k-NN(k=15)	0.941	0.804	0.877	0.867
C4.5	0.934	0.809	0.876	0.867
RF (27 trees)	0.944	0.809	0.881	0.872
Naive Bayes	0.839	0.801	0.86	0.864
Logistic Regression- L1 Regularization	0.932	0.803	0.872	0.867
Logistic Regression- L2 Regularization	0.925	0.803	0.865	0.859

Table 5.5 Classification performance of the detection model using T3 training set

From the experimental results given in Tables 5.3-5.5, we can see that the Naive Bayes algorithm produces the lowest overall classification accuracy (ACC) and the random forest algorithm produces the highest classification accuracy among 6 machine learning algorithms. However, the differences in the classification accuracy among algorithms are not so great. Two machine learning algorithms, including k-NN with weighted neighbours and C4.5 decision trees have roughly the same overall classification accuracy. For the k-NN algorithm, the choice of parameter k plays an important role in the classification performance and is usually determined empirically. The Naive Bayes algorithm provides the lowest overall classification accuracy, but it has the advantages of being simple and low time and computation requirements for training and testing.

It can also be seen that the random forest algorithm yields significantly better results than the C4.5 decision tree algorithm. However, the training time of the random forest is longer due to the number of trees requiring training is more. However, the training of the random forest classifier can be done offline, so it does not affect the classification speed in the testing period. Thus, the

random forest machine learning algorithm has the highest overall classification accuracy and is selected for implementation in the proposed botnet detection model.

In addition, the ratio between the number of benign domain names and the number of malicious domain names has a significant impact on classification measures. Accordingly, the T1 training set gives the highest ACC and F1 measures. However, its FPR is relatively high. The ACC and F1 measures produced by the T3 training set are not high, but the FPR is relatively low. At the same time, the T2 set gives the worst results of low ACC and F1 measures and very high FPR measure. As such, the ratio of the number of benign domain names and the number of malicious domain names to be chosen as 1:1 (the T1 data set) will produce the best classification results.

5.5 Receiver Operating Characteristic Curve (ROC)

The ROC curve is one of the important metrics in analysing the performance of the model. It is plotting of the true positive rate against false positive rates at various threshold settings. The true positive rate is defined as precision and the false positive rate is defined as 1-specificity.

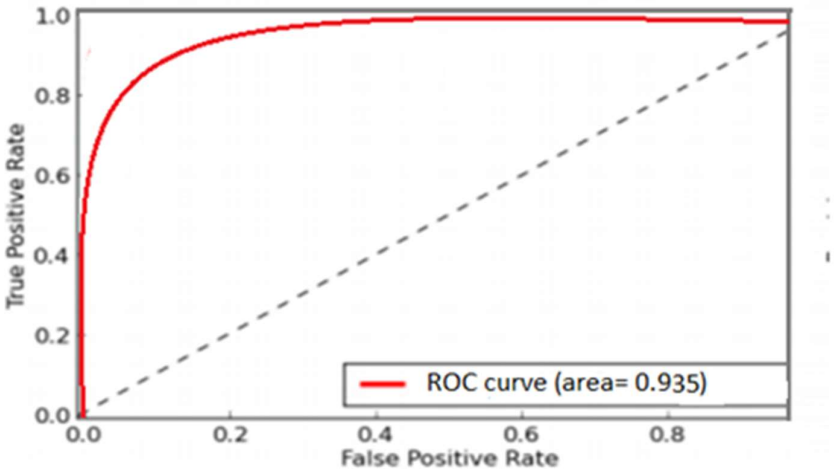

Figure 5.1: ROC Curve - Random Forest

The area under the curve (AUC) is the probability that the model will rank the randomly chosen positive value higher than the randomly chosen negative instance. The maximum value for the area under the curve is 1 and the minimum of 0.5 is considered as a good model. The higher the AUC value the model performs better. The Random Forest model which showed high values of a precision of 0.907, recall value of 0.901 and accuracy value of 0.908 has high AUC value of 0.935.

CHAPTER 6-Conclusions and Future work

In this project, I investigated the effectiveness of the botnet detection model based on machine learning techniques using DNS query data. The experimental results on DGA botnet and FF botnet data sets show that most of the machine learning techniques used in the model achieved the overall classification accuracy over 85%, among which the random forest algorithm gives the best results with the overall classification accuracy of 90.80%.

Based on this result, it is best to select the random forest algorithm for the proposed botnet detection model using DNS query data. In the future, we can continue to test the proposed model with larger data sets and analyse the effects of the domain name features on the detection accuracy, as well as research and propose new features to improve the detection accuracy of the proposed model.

CHAPTER 7-References

1. Authority of Information Security. The 2016 Vietnam Information Security Report; Authority of Information Security, MIC: New York, NY, USA, 2016.

2. Ferguson, R. The History of the Botnet.

3. Alieyan, K.; Almomani, A.; Manasrah, A.; Kadhum, M.M. A survey of botnet detection based on DNS. Nat. Comput. Appl. Forum 2017, 28, 1541–1558.

4. Li, X.; Wang, J.; Zhang, X. Botnet Detection Technology Based on DNS. J. Future Internet 2017, 9, 55.

5. Ramachandran, A.; Feamster, N.; Dagon, D. Revealing botnet membership using DNSBL counter-intelligence. In Proceedings of the 2nd USENIX: Steps to Reducing Unwanted Traffic on the Internet, San Jose, CA, USA, 7 July 2006; pp. 49–54.

6. Villamari-Salomo, R.; Brustoloni, J.C. Identifying botnets using anomaly detection techniques applied to DNS traffic. In Proceedings of the 5th IEEE consumer communications and networking conference (CCNC 2008), Las Vegas, NV, USA, 10–12 January 2008; pp. 476–481. 7. Perdisci, R.; Corona, I.; Dagon, D.; Lee, W. Detecting malicious flux service networks through passive analysis of recursive DNS traces. In Proceedings of the Annual Computer Security Applications Conference, Honolulu, HI, USA, 7–11 December 2009; pp. 311–320.

8. Yadav, S.; Reddy, A.K.K.; Reddy, A.; Ranjan, S. Detecting algorithmically generated malicious domain names. In Proceedings of the 10th ACM sigcomm Conference on Internet Measurement, Melbourne, Australia, 1–30 November 2010; pp. 48–61.

9. Stalmans, E.; Irwin, B. A framework for DNS based detection and mitigation of malware infections on a network. In Proceedings of the 2011 Information Security for South Africa (ISSA), Johannesburg, South Africa, 15–17 August 2011; pp. 1–8.

10. Antonakakis, M.; Perdisci, R.; Lee, W.; Vasiloglou, N., II; Dagon, D. Detecting malware domains at the upper DNS hierarchy. In Proceedings of the USENIX security symposium, San

Francisco, CA, USA, 8 August 2011; p. 16. 11. Bilge, L.; Kirda, E.; Kruegel, C.; Balduzzi, M. Exposure: Finding Malicious Domains Using Passive DNS Analysis; NDSS: New York, NY, USA, 2011.

12. Jiang, N.; Cao, J.; Jin, Y.; Li, L.; Zhang, Z.L. Identifying suspicious activities through DNS failure graph analysis. In Proceedings of the18th IEEE International Conference on Network Protocols (ICNP), Kyoto, Japan, 5–8 October 2010; pp. 144–153.

13. Kheir, N.; Tran, F.; Caron, P.; Deschamps, N. Mentor: positive DNS reputation to skim-off benign domains in botnet C&C blacklists. In ICT Systems Security and Privacy Protection; Springer: Berlin/Heidelberg, Germany, 2014; pp. 1–14.

14. Da Luz, P.M. Botnet Detection Using Passive DNS; Radboud University: Nijmegen, The Netherlands, 2014.

15. Sangani, N.K.; Zarger, H. Machine Learning in Application Security. Advances in Security in Computing and Communications; IntechOpen: Karnataka, India, 2017.

16. Smola, A.; Vishwanathan, S.V.N. Introduction to Machine Learning; Cambridge University Press: Cambridge, UK, 2008.

17. Conficker. Available online: http://www.cert.at/static/conficker/all_domains.txt (accessed on 10 November 2017).

18. DGA Data set Available: https://github.com/nickwallen/botnet-dga-classifier/tree/master/data (accessed on 10 November 2017).

19. (Introduction to Anomaly Detection: Concepts and Techniques 2015) United Nations, accessed 17 November 2015, (https://iwringer.wordpress.com/2015/11/17/anomaly-detection-concepts-and-techniques/).

20. (A survey on network attacks and Intrusion detection systems 2017) United Nations, accessed 24 August 2017, (https://ieeexplore.ieee.org/document/8014614).