Project Report On

A Machine Learning Framework for Domain Generation Algorithm (DGA)-Based Malware Detection

Submitted in partial fulfillment of the requirements of the degree of

Fourth Year of Engineering in Information Technology

Submitted by

Harsh Dobariya	714
Akshay Kalapgar	731
Mohit Kamble	732
Siddhesh Parab	750

Guided by

Prof. Abhay E. Patil

MCT
MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI
(Permanently Affiliated to University of Mumbai)
Juhu Versova Link Road, Andheri (West), Mumbai-53

DEPARTMENT OF INFORMATION TECHNOLOGY

UNIVERSITY OF MUMBAI

2020 - 2021

MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI
(Permanently Affiliated to University of Mumbai)
Juhu Versova Link Road, Andheri (West), Mumbai-53

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

Date: __________

This is to certify that, the project work embodied in this report entitled, "***A Machine Learning Framework for Domain Generation Algorithm (DGA)-Based Malware Detection***" submitted by "***Harsh Dobariya*** bearing Roll No. 714", "***Akshay Kalapgar*** bearing Roll No. 731", "***Mohit Kamble*** bearing Roll No. 732", "***Siddhesh Parab*** bearing Roll No. 750" for the award of ***Fourth Year Of Engineering (B.E.)*** degree, is a work carried out by them under my guidance and supervision within the institute. The work described in this project report is carried out by the concerned students and has not been submitted for the award of any other degree of the University of Mumbai.

Further, it is to certify that the students were regular during the academic year 2020-21 and have worked under the guidance of concerned faculty until the submission of this project work at ***MCT's Rajiv Gandhi Institute of Technology, Mumbai***.

[illegible]

Prof. Abhay E. Patil

Project Guide

Dr. Sunil B. Wankhade	Dr. Sanjay U. Bokade
Head of Department	**Principal**

CERTIFICATE OF APPROVAL

This project report is entitled

A Machine Learning Framework for Domain Generation Algorithm (DGA)-Based Malware Detection

Submitted by:

Harsh Dobariya	**714**
Akshay Kalapgar	**731**
Mohit Kamble	**732**
Siddhesh Parab	**750**

In partial fulfillment of the requirements of the degree of **Fourth Year** in **Bachelor of Engineering** in **Information Technology** is approved.

Internal Examiner

SEAL OF

INSTITUTE

____

External Examiner

____

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

ROLL NO.	NAME	SIGNATURE
714	HARSH DOBARIYA	[illegible]
731	AKSHAY KALAPGAR	Akshay
732	MOHIT KAMBLE	Mohit
750	SIDDHESH PARAB	[illegible]

Date:

Place:

Acknowledgement

With all reverence, we take the opportunity to express our deep sense of gratitude and wholehearted indebtedness to our respected guide, **Prof. A.E. Patil**, Department of Information Technology, Rajiv Gandhi Institute of Technology, Mumbai. From the day of conception of this project his active involvement and motivating guidance on day-to- day basis has made it possible for us to complete this challenging work in time.

We would like to express a deep sense of gratitude to our respected **Head of the Department**, **Dr. Sunil B. Wankhade** who went all the way out to help us in all genuine cases during the course of doing this project. We wish to express our sincere thanks to **Dr. Sanjay Bokade**, **Principal**, Rajiv Gandhi Institute of Technology, Mumbai and would to like to acknowledge specifically for giving guidance, encouragement and inspiration throughout the academics.

We would like to thank all the staff of Information Technology Department who continuously supported and motivated during our work. Also, we would like to thank our colleagues for their continuous support and motivation during the project work. Finally, we would like to express our gratitude to our family for their eternal belief in us. We would not be where we are today without their support and encouragement.

HARSH DOBARIYA

AKSHAY KALAPGAR

MOHIT KAMBLE

SIDDHESH PARAB

Date:

Place:

Abstract

Attackers usually use a Command and Control (C2) server to manipulate the communication. In order to perform an attack, threat actors often employ a Domain Generation Algorithm (DGA), which can allow malware to communicate with C2 by generating a variety of network locations. Traditional malware control methods, such as blacklisting, are insufficient to handle DGA threats. In this paper, we propose a machine learning framework for identifying and detecting DGA domains to alleviate the threat. We collect real-time threat data from the real-life traffic over a one-year period. We also propose a deep learning model to classify a large number of DGA domains. The proposed machine learning framework consists of a two level model and a prediction model. In the two-level model, we first classify the DGA domains apart from normal domains and then use the clustering method to identify the algorithms that generate those DGA domains. In the prediction model, a time-series model is constructed to predict incoming domain features based on the Hidden Markov Model (HMM). Furthermore, we build a Deep Neural Network (DNN) model to enhance the proposed machine learning framework by handling the huge dataset we gradually collected. Our extensive experimental results demonstrate the accuracy of the proposed framework and the DNN model. To be precise, we achieve an accuracy of 95.89% for the classification in the framework and 97.79% in the DNN model, 92.45% for the second-level clustering, and 95.21% for the HMM prediction in the framework.

Keywords: Antigen, Blood Samples, GPU, Histogram, LBP (local binary pattern), Nearest Neighbour Classifier, Image Processing, Pattern Matching.

Table of Contents

Table of Figures

Chapter 1.
Introduction

Malware attackers attempt to infiltrate layers of protection and defensive solutions, resulting in threats on a computer network and its assets. Anti-malware software have been widely used in enterprises for a long time since they can provide some level of security on computer networks and systems to detect and mitigate malware attacks. However, many anti-malware solutions typically utilize static string matching approaches, hashing schemes, or network communication white listing. These solutions are too simple to resolve sophisticate malware attacks, which can hide communication channels to bypass most detection schemes by purposely integrating evasive techniques. The issue has posed a serious threat to the security of an enterprise and it is also a grand challenge that needs to be addressed.

In this paper, we first propose a machine learning framework to classify and detect DGA malware and develop a DNN model to classify the large datasets of DGA domains that we gradually collected. We then experimentally evaluate the proposed framework through a comparison of various machine learning approaches and a deep learning model. Specifically, our machine learning framework consists of the following four main components: A dynamic blacklist consists of a pattern filter. A two level machine learning model: the first-level classification and the second-level clustering. To identify DGA domains, we first use various classification models to classify DGA domains and normal domains. Then, we apply the clustering method to group domains sequenced by the DGA. A time series prediction model: we propose a Hidden Markov Model (HMM) to predict incoming DGA domain features in order to better identify the DGA domains. The general goal of our machine learning framework is to determine which algorithm is employed so that our proposed framework can prevent future communications from the C2.

Furthermore, we have gradually collected the data for over one year and have obtained a large amount of datasets from real traffic. To analyze these data, we also propose a deep learning approach for large dataset classification. We first build a DNN model and then compare it with our machine learning models. The comparison results provide us a useful guideline for our future study in DGA detection and prediction. In our future research, we will also apply deep learning in clustering and prediction that are out of the scope of this paper.

Chapter 2.

Aim & Objectives

Aim:-

To solve the problem of detecting DGA sequences using machine learning techniques derived from observations in a network.

Objectives:-

The objectives of the systems development and event management are:

1. In DBSCAN algorithm, we use the features described above to calculate the domain distance and to group the domains that are generated by the same DGA together according to their domain feature difference.
2. Distinguish the model from training and prediction stages.
3. The nodes in each layer are fully connected to the nodes in the next will not miss any local minima, but it will take a long time to converge.

Chapter 3.
Literature Survey

❖ Literature Survey in a tabular format for better understanding.

Title	**Authors**	**Advantages**	**Disadvantages**	**Result**
A Machine Learning Framework for Domain Generation Algorithm-Based Malware Detection	Yi Li, Kaiqi Xiong, Tommy Chin, Chengbin Hu. Institute of Electrical and Electronics Engineers, 2019.	In the second-level clustering we apply the DBSCAN algorithm. Only the DGA domains obtained from the first-level classification will be used for clustering.	Research problem is to accurately identify and cluster domains that originate from known DGA-based techniques where we target to develop a security approach that autonomously mitigates network communications to unknown threats in a sequence.	Domain Generation Algorithm (DGA) is used.
Learning and Classification of Malware Behavior	Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, Pavel Laskov. Kluwer Academic Publishers, Dordrecht (2002).	Results show that 70% of malware instances not identified by an anti-virus software can be correctly classified by our approach.	Proposed machine learning framework aims to solve the problem of detecting DGA sequences using machine learning techniques derived from observations in a network.	• Normalize Compression Distance. • Benign Executable
An SDN based framework for guaranteeing security and performance in information-centric cloud networks	U. Ghosh, P. Chatterjee, D. Tosh, S. Shetty, K. Xiong, and C. Kamhou. 11th IEEE International Conference on Cloud Computing (IEEE Cloud), 2017.	The framework has the ability to dynamically compute the routing path to guarantee security and performance of the network.	Queries not matching the knowledge are stored in a backlog of the software.	SDN-based framework and Information centric services
A two-hashing table multiple string pattern matching algorithm	C. Khancome, V. Boonjing, and P. Chanvarasuth. Tenth International Conference on Information Technology: New Generations (ITNG). IEEE, 2013.	The attempting times were less than of the traditional algorithms especially in the case of a very long minimum pattern length.	The lengthy processing time when directly extended to the multiple string patterns matching.	Executes multiple string pattern matching algorithm.

Chapter 4.

Existing System

Threat models: Multiple conditions for a DGA to function in a network environment where filtering results in a firewall that protects the communication and an empty cell in an Internet domain those results in NXDOMAIN error.

Each HMM date record represents a series of domain observations. First sequences of domain name are processed by a feature extractor and each of these feature vectors is used as a training record.

Then, similar sequences are clustered as a group of DGA domain names with certain outcomes. After the training process, if a sequence does not have an HMM sequence representation (or it is not presented in the training data but the test data), the HMM model then generates the future predicted results. Otherwise, we will use an existing HMM sequence presentation.

Disadvantages of Existing System:

1. Firewall protects the communication and an empty cell in an internet domain that results in no domain error.
2. Queries not matching the knowledge are stored in a backlog of the software.

Chapter 5.
Problem Statement

The malware that communicates with an appropriate domain correctly, a threat actor must register each respective domain name in the sequence to maintain the C2 or risk the loss of a node in the distribution.

Our research problem is to accurately identify and cluster domains that originate from known DGA-based techniques where we target to develop a security approach that autonomously mitigates network communications to unknown threats in a sequence.

Chapter 6.
Proposed System

In our proposed system, Domains extracted from DGAs. Machine learning framework that encompasses multiple feature extraction techniques and the models to classify the DGA domains from normal domains, cluster the DGA domains, and predict a DGA domain.

A deep learning model to handle large datasets multiple on- line sources from simple Google searching provide example codes for a DGA construction.

Online threat intelligence feeds give an approach to examining current and live threats in real-world environment.

Using real-time active malicious domains derived from DGAs on the public Internet measures the accuracy of the proposed approach.

The structure of the data is presented in a CSV format of domain names, originating malware, and DGA membership with the daily file size of approximate 110MB.

We propose a machine learning framework that consists of three important steps, as shown in Figure below.

We first have the DNS queries with the payload as the input.

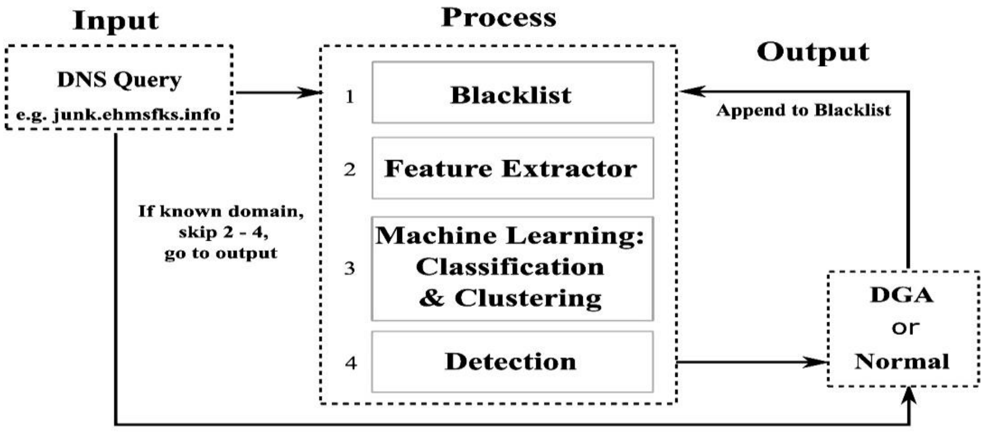


Fig. 6.1: Diagrammatical Representation of working of the system

Advantages of Proposed System:

1. Domain Generation Algorithm (DGA), which allows malware to generate numerous domain names until it finds its corresponding C&C server.
2. It is highly resilient to detection systems and reverse engineering, while allowing the C&C server to have several redundant domain names.

Chapter 7.

Methodology

This project we will develop using python and web technology.

1. Filtering packet data:-

To filter packet data we are using pyshark which captures network packets.

We will store this packet information in pcap format

By reading packet we will filter the data and obtain domain name.

Packet flow also obtained from this.

If domain name extracted in this found in blacklist we will stop further steps.

2. Feature extraction work:-

With the python coding we will calculate the following feature

- Length- length of domain name.
- Meaningful Word Ratio,:- dictionary will be maintained of meaningful word and output will be taken by dividing with length of domain name
- Percentage of Numerical Characters,:- numeric character involved in domain name system.
- Pronounce ability Score—frequency of text in domain calculated.
- Percentage of the Length of the Longest Meaningful String (LMS):- dividing the meaningful word with the length of domain.
- Levenshtein Edit Distance:- It measures the minimum number of single-character edits between a current domain and its previous domain in a stream of DNS queries received by the server. The Levenshtein distance is calculated based on a domain and its predecessor

3. Machine learning classification:-

Following algorithms will be applied on feature obtained above.

- Decision tree:-

  It calculates entropy and information gain and output generated but has problem of over fitting. We will generate module with the selected feature.
- ANN:- it's a Artificial neural network

  Here we give input layer, hidden layer and output layer. Then with the feature we calculate output.
- SVM:- support vector machine

It’s a good binary classifier .we will train with feature and model will be generated.
We are using sk learn python library

- Multiple Logistic regression:-, the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.
- Naive Bayes: - it calculates probability of occurring certain class. Model will be generated using pickle and stored
- Random forest: - Random forest avoids over fitting problem and model will be generated, stored into pickle.

All this machine learning model will generated and its

i. precision
ii. recall
iii. f1 score
iv. Accuracy will be calculated.

- Clustering:-
  Dbscan used for outlier’s detection
  Outliers are specific entries in dataset that are different than other point and don’t play vital role in classification.
  In statistics, an **outlier** is an observation point that is distant from other observations.
  In this domain name will be clustered based on

i. Cryptolockereg. nxgbdtnvrfker.ru
ii. TOVAReg.:- gppwkpxyremp.net
iii. Dyreeg:- q2aa41a5b31294e5e6f28d1adcf48a54b.tk
iv. normalDomaineg:- easypdfcombine.com

4. <u>Time series prediction</u>

We use every domain cluster to train a separate HMM model. Each HMM data record represents a series of domain observations. First, a sequence of domain names are processed by a feature extractor and each of these feature vectors is used as a training record. Then, similar sequences are clustered as a group of DGA domain names with certain outcomes.

Chapter 8.

Details of Hardware and Software

Hardware Requirements:

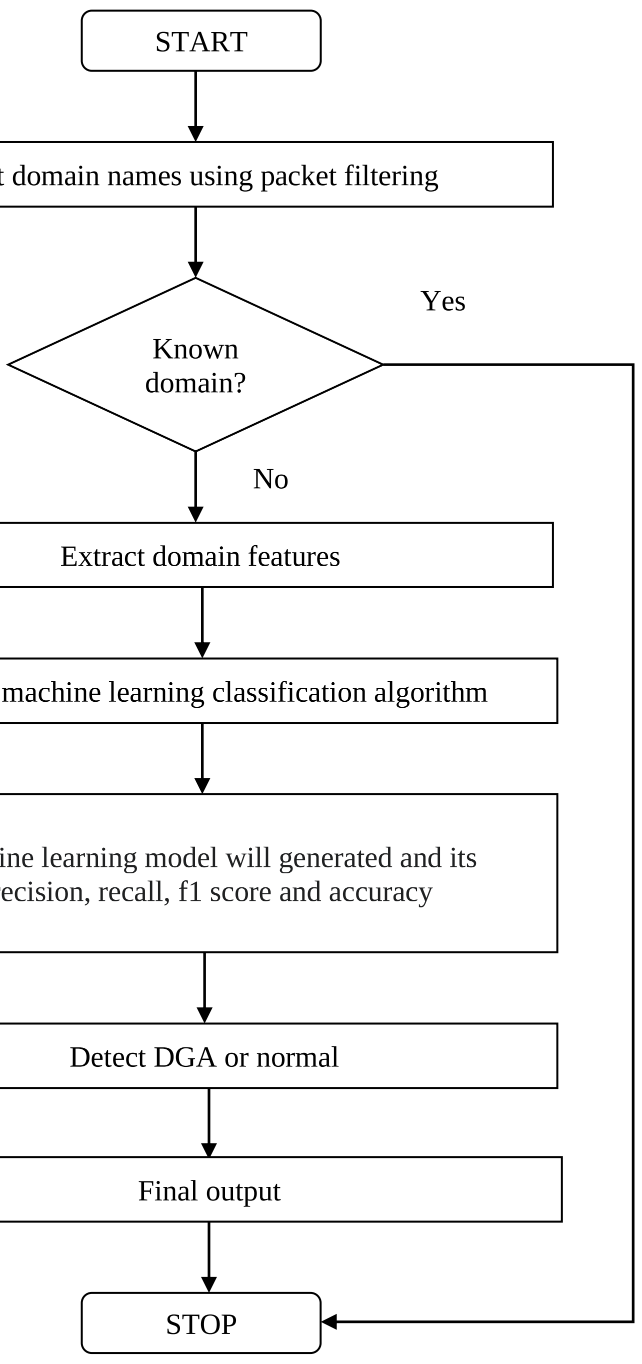
1. Processor: Intel Core i3 or more.
2. RAM: 4GB or more.
3. Hard disk: 250 GB or more.

Software Requirements:

1. Operating System : Windows10, 7, 8.
2. Python.
3. Anaconda.
4. Spyder, Jupyter notebook, Flask.
5. MYSQL.

Chapter 9.
Implementation

Flow chart of the system:



START
Get domain names using packet filtering
Known domain?
Yes
No
Extract domain features
Apply machine learning classification algorithm
Machine learning model will generated and its precision, recall, f1 score and accuracy
Detect DGA or normal
Final output
STOP

Fig. 9.1 - Flowchart of the System

Use case of the system:

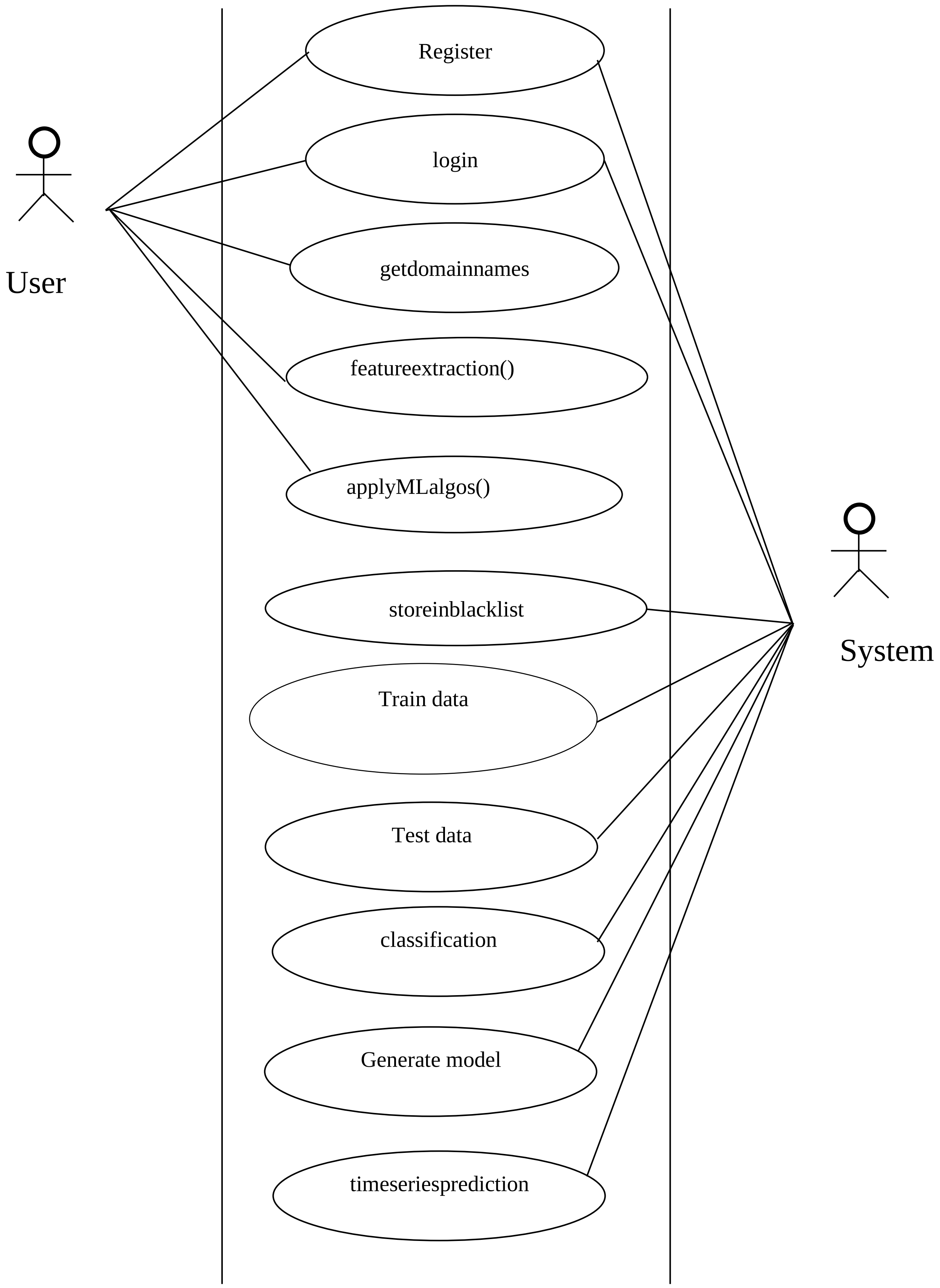


Fig. 9.2 - Use case Diagram of the System

Activity Diagram of the system:

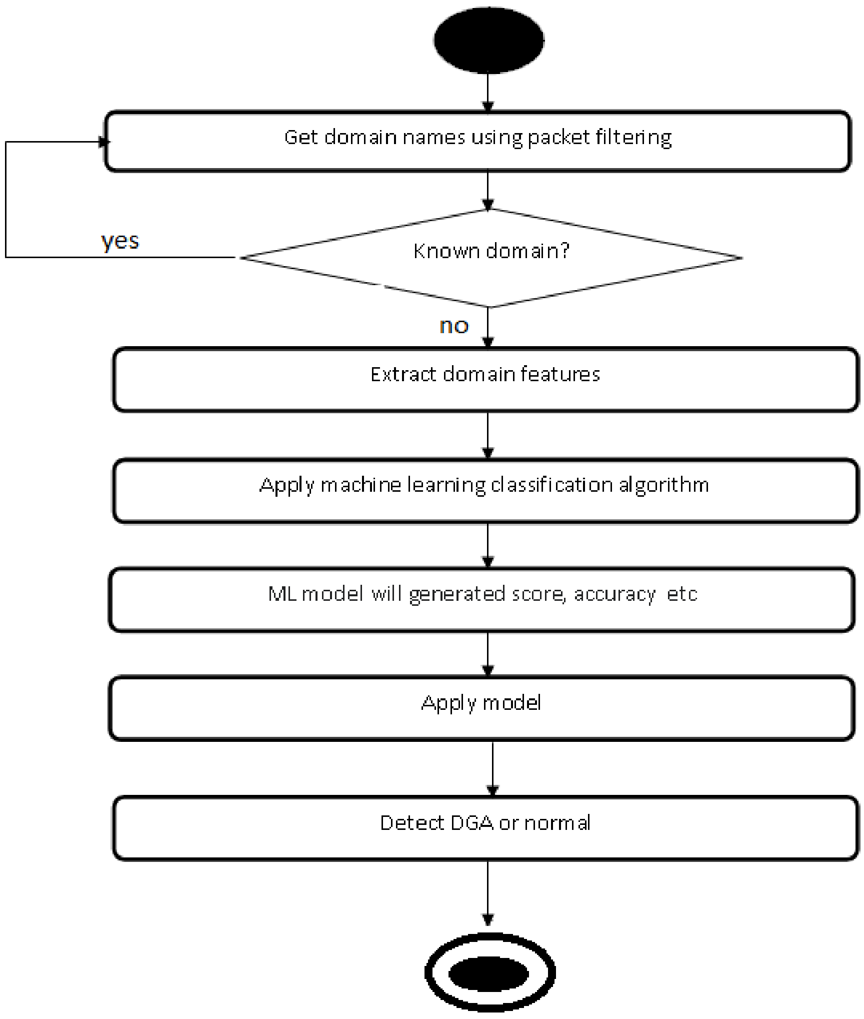
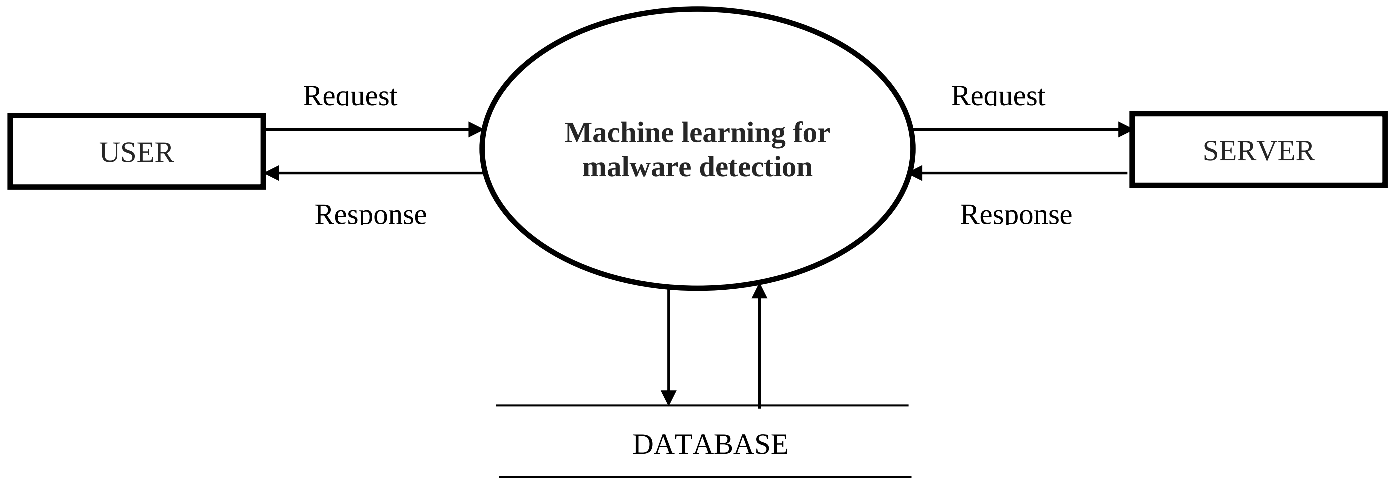

Get domain names using packet filtering
yes
Known domain?
no
Extract domain features
Apply machine learning classification algorithm
ML model will generated score, accuracy etc
Apply model
Detect DGA or normal

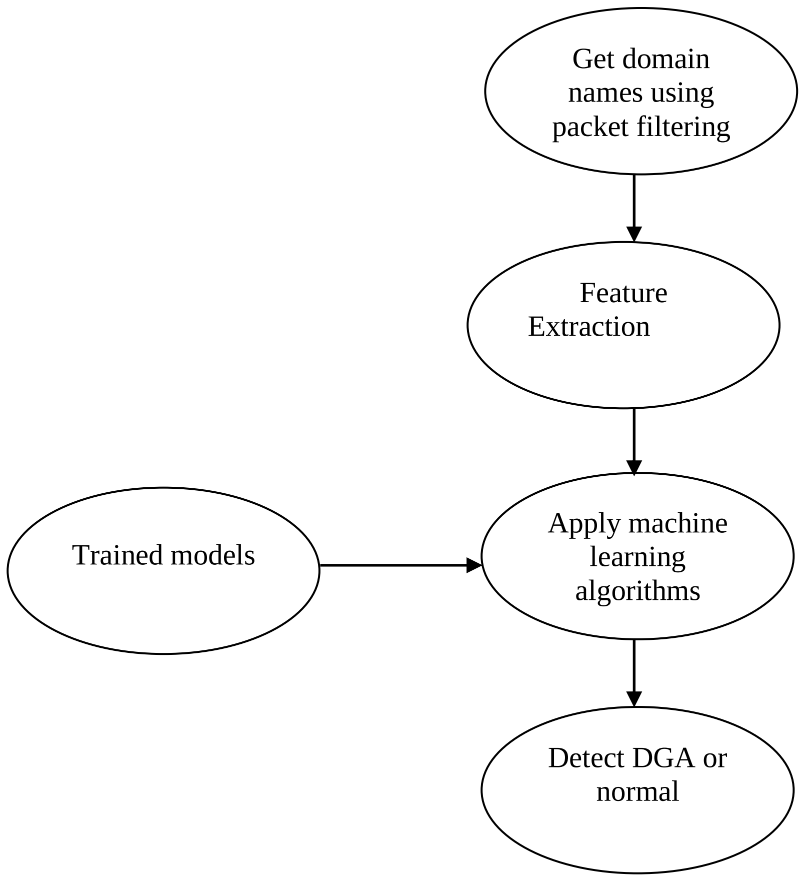
Fig. 9.3 - Activity Diagram of the System

Data-flow diagram (DFD):

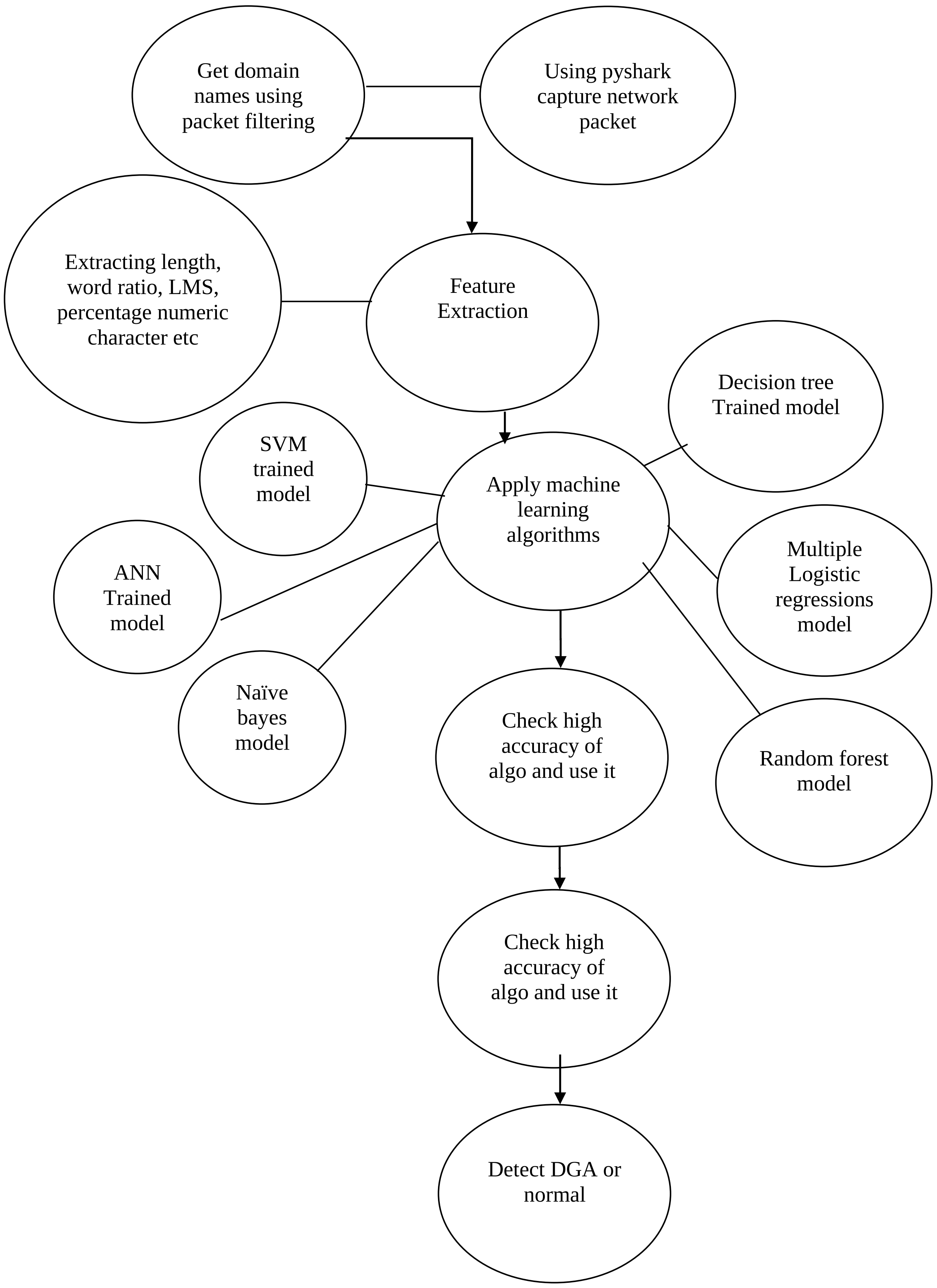
DFD level 0:



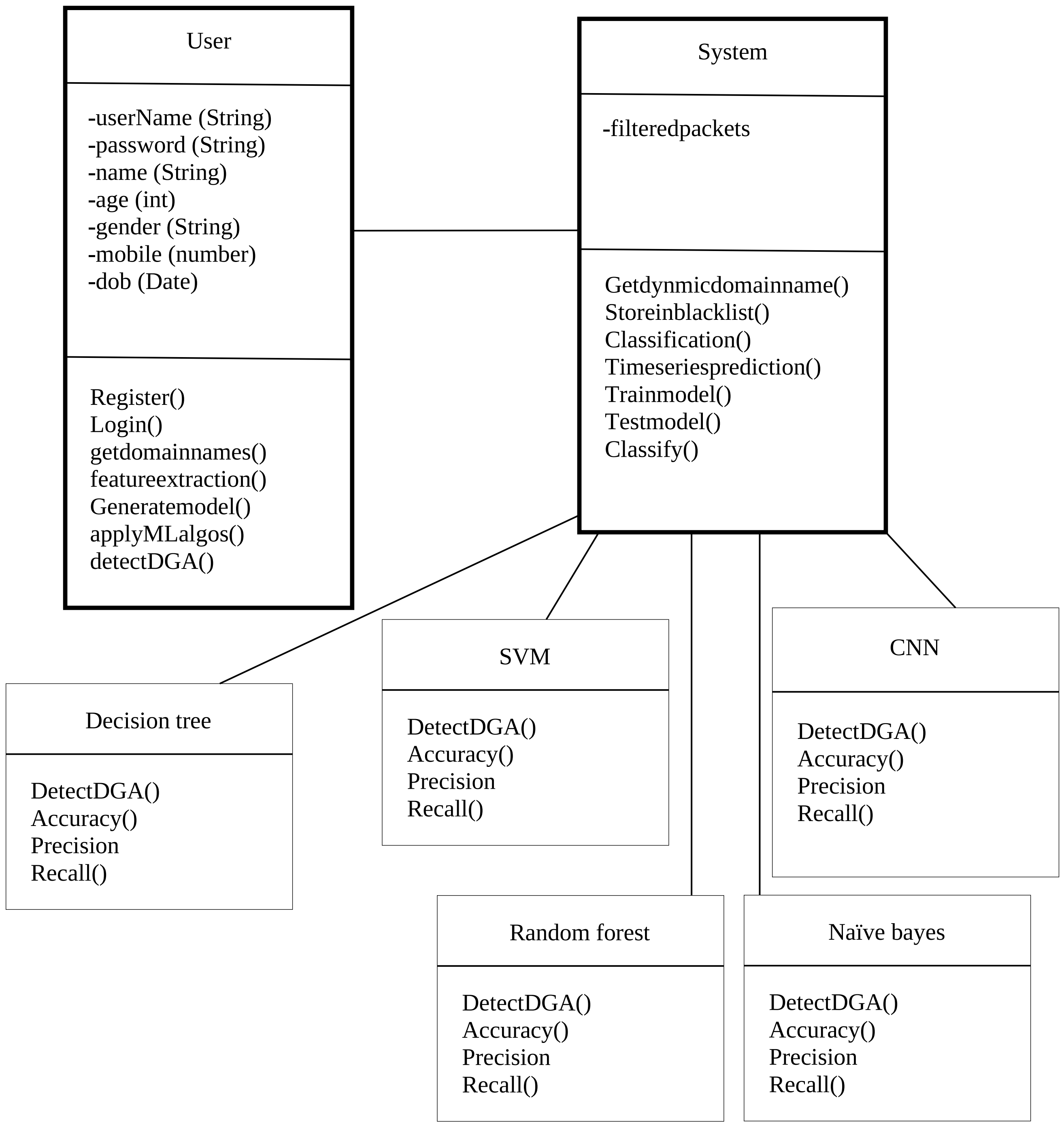
DFD level 1:


Get domain names using packet filtering
Feature Extraction
Trained models
Apply machine learning algorithms
Detect DGA or normal

DFD level 2:


Get domain names using packet filtering
Using pyshark capture network packet
Extracting length, word ratio, LMS, percentage numeric character etc
Feature Extraction
Decision tree Trained model
SVM trained model
Apply machine learning algorithms
ANN Trained model
Multiple Logistic regressions model
Naïve bayes model
Check high accuracy of algo and use it
Random forest model
Check high accuracy of algo and use it
Detect DGA or normal

Class Diagram:



TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with intent of finding software bugs (errors or other defects).

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- Meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs,
- Performs its function within an Acceptable time,
- Is sufficiently usable,
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholder’s desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusive) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed; it can illuminate other, deeper bugs, or can even create new ones. Software testing can provide objective, independent information about the quality of software and risk of its failure to user and/or sponsors. Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how the testing is conducted. For example, in a phased process, most testing occurs after the system requirements have been

Defined and then implemented in testable programs. In contrasts, under an Agile approach, requirements, programming, and testing are often done concurrently.

❖ LEVELS OF TESTING:

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic level of testing are:-

- ✓ Unit testing.
- ✓ Integration testing.
- ✓ Regression testing.
- ✓ System testing.
- ✓ Validation testing.

SOFTWARE TESTING LIFE CYCLE :

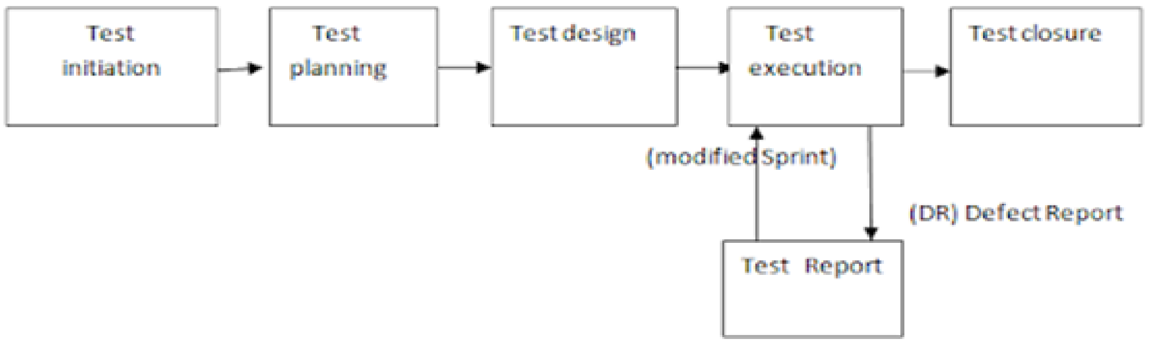


Fig 9.7 - Software Testing Lifecycle

UNIT TESTING

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object- oriented programming, a unit is often an entire interface, such as a class, but could be an individual method, unit tests a short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing.

INTEGRATION TESTING

Integration testing is any type of software testing that seeks to verify the interfaces between components against software design. Software components may be integrated in an iterative way or all together (“big bang”). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules).

REGRESSION TESTING

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regressions, as degraded or lost features, including old bugs that have come back. Such regressions occur whenever software functionality that was previously working correctly, stops working was intended. Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include rerunning previous sets of test cases and checking whether previously fixed faults have re-emerged.

SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

VALIDATION TESTING

Validation Testing ensures that the product actually meets the client’s need. It can also be defines as to demonstrate that the product fulfills its intended use when deployed on appropriate environment

Chapter 10.
Advantages & Disadvantages

ADVANTAGES:

1. Domain Generation Algorithm (DGA), which allows malware to generate numerous domain names until it finds its corresponding C&C server.
2. It is highly resilient to detection systems and reverse engineering, while allowing the C&C server to have several redundant domain names.
3. ML algorithm helps to improve accuracy of result.

DISADVANTAGES:

1. Proposed machine learning framework aims to solve the problem of detecting DGA sequences using machine learning techniques derived from observations in a network.

2. Queries not matching the knowledge are stored in a backlog of the software.

3. The lengthy processing time when directly extended to the multiple string patterns matching.

Chapter 11

Scope

- The most common method to detect malicious URLs deployed by many antivirus groups is the blacklist method.
- Blacklists are essentially a database of URLs that have been confirmed to be malicious in the past. Scope of this project is useful for it helps to prevent malicious activity in cyber world.

Future Modification:

- In future it is intended to improve the system performance on the based on dataset.
- Also use new techniques to get accurate result.

Chapter 12
References

1. Yi Li, Kaiqi Xiong, Tommy Chin, Chengbin Hu, “A Machine Learning Framework for Domain Generation Algorithm-Based Malware Detection” IEEE Access (Volume: 7), 31 January 2019

2. T. Chin, K. Xiong and M. Rahouti, "SDN-based kernel modular countermeasure for intrusion detection", *Proc. 13rd EAI Int. Conf. Secur. Privacy Commun. Netw.*, pp. 270-290, 2019.

3. Suthathira Vanitha N., Professor, Department of EEE, Knowledge Institute of Technology, Tamil Nadu, India, A novel approach in identification of blood group using laser technology, International Journal of Research in Engineering and Technology (2014).
   Available: esatjournals.net/ijret/2014v03/i23/IJRET20140323005.pdf

4. Jose Fernandes, Sara Pimenta, Student Member, IEEE, Filomena O. Soares, Senior Member, IEEE and Graca Minas, Senior Member, IEEE, (2012), A Complete Blood Typing Device for Automatic Agglutination Detection Based on Absorption Spectrophotometry, IEEE Transactions On Instrumentation And Measurement.

5. Nazia Fathima S.M (2013) Classification of blood type by microscopic color images, International Journal of Machine Learning and Computing. Vol. 3, No. 4.
   Available: www.ijmlc.org/papers/342-L472.pdf