**A UNIQUE WEB ASSAULT DETECTION METHOD FOR IOT THAT USES ENSEMBLE CATEGORIZATION**

A “Project Stage II” Report submitted to

JNTU Hyderabad in partial fulfillment

of the requirements for the award of the degree

**BACHELOR OF TECHNOLOGY**

In

***Submitted by***

**AJAY BANDARI 20S11A6902**

**PRAVALIKA NIMMAKAYALA 20S11A6914**

**VAISHNAVI NAINI 20S11A6921**

*Under the Guidance of*

***Ms. B. KOTESWARI***

*B. Tech., M. Tech*

Assistant Professor



***DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING***

***(INTERNET OF THINGS)***

**MALLA REDDY INSTITUTE OF TECHNOLOGY AND SCIENCE**

*(Approved by AICTE New Delhi and Affiliated to JNTUH)*

*Accredited by NBA and NAAC with “A” Grade*

*Maisammaguda, Medchal (M), Hyderabad-500100, Telangana*

**MARCH 2024**

***DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING***

***(INTERNET OF THINGS)***

**MALLA REDDY INSTITUTE OF TECHNOLOGY & SCIENCE**

*(Approved by AICTE New Delhi and Affiliated to JNTUH)*

*(Accredited by NBA & NAAC with “A” Grade)*

*An ISO 9001: 2015 Certified Institution*

*Maisammaguda, Medchal (M), Hyderabad-500100, T. S.*

**MARCH 2024**



***CERTIFICATE***

This is to certify that the “Project Stage II” entitled **“A unique web assault detection method for IoT that uses ensemble categorization”** has been submitted by **Ajay Bandari 20S11A6902, Pravalika Nimmakayala 20S11A6914 and Vaishnavi Naini 20S11A6921** in partial fulfillment of the requirements for the award of **BACHELOR OF TECHNOLOGY**  in **COMPUTER SCIENCE & ENGINEERING (IOT)**. This record of bonafide work carried out by them under my guidance and supervision. **The result embodied in this Project Stage II report has not been submitted to any other University or Institute for the award of any degree.**

**Ms. B. KOTESWARI DR. T. SAI KUMARI**

*Assistant Professor Head of the Department*

*Project Guide*

***External Examiner***

**ACKNOWLEDGEMENT**

The Project Stage II work carried out by our team in the Department of Computer Science and Engineering(IOT), Malla Reddy Instituter of Technology and Science, Hyderabad. ***This work is original and has not been submitted in part or full for any degree or diploma of any other university.***

We wish to acknowledge our sincere thanks to our project guide **Ms. B. Koteswari**, Assistant Professor of Computer Science & Engineering (Cyber Security) for formulation of the problem, analysis, guidance and her continuous supervision during the course of work.

We acknowledge our sincere thanks to **Dr. Vaka Murali Mohan,** Principal and **Dr. T. Sai Kumari**, Head of the Department and Coordinator, faculty members ofCSE(IOT) Departmentfor their kind cooperation in making this Project Stage II work a success.

We extend our gratitude to **Sri. Ch. Malla Reddy**, Founder Chairman MRGI and **Sri. Ch. Mahender Reddy,** Secretary MRGI, **Dr.Ch. Bhadra Reddy**, President MRGI, **Sri. Ch. Shalini Reddy**, Director MRGI, **Sri. P. Praveen Reddy**, Director MRGI, for their kind cooperation in providing the infrastructure for completion of our Project Stage II.

We acknowledge our special thanks to the entire teaching faculty and non-teaching staff members of the Computer Science & Engineering (IOT) Department for their support in making this project work a success.

**AJAY BANDARI HT.NO: 20S11A6902**

**PRAVALIKA NIMMAKAYALA HT.NO: 20S11A6914**

**VAISHNAVI NAINI HT.NO: 20S11A6921**

**INDEX**

**Chapter Page No.**

**ABSTRACT** VI

**LIST OF FIGURES** VII

**LIST OF TABLES** VII

**1. SYSTEM ANALYSIS 1**

1.1 Existing System 2

1.1.1 Disadvantages of Existing System 2

1.2 Proposed System 2

1.2.1 Advantages of Proposed System 2

1.3 Introduction 10

**2. LITERATURE SURVEY 12**

**3. SYSTEM DESIGN 14**

3.1 System Architecture 15

3.1.1 Modules 15

3.1.2 Block Diagram 15

3.2 System Requirements 22

3.2.1 Hardware Requirements 22

3.2.2 Software Requirements 22

**4. INPUT & OUTPUT DESIGN 23**

4.1 Input Design 24

4.2 Output Design 27

**5. SOFTWARE ENVIRONMENT 28**

5.1 Python Technology 29

5.1.1 History of python 29

5.1.2 What can python do 30

5.1.3 Why python 30

5.1.4 Python syntax compared to other 30

programming languages

5.1.5 Uses of Python 30

5.1.6 Python Features 31

5.2 What is Machine Learning 32

5.2.1 types of machine learning 32

5.2.2 applications of machine learning 33

5.2.3 modules used in project

5.3 SDLC 38

**6. SYSTEM STUDY 41**

6.1 Economic Feasibility42

6.2 Technical Feasibility 43

6.3 Social Feasibility 44

**7. SYSTEM TESTING** **46**

7.1 Types of Tests 47

7.1.1 Unit testing 47

7.1.2 Integration testing 47

7.1.3 Functional testing 47

7.1.4 System testing 48

7.1.5 Acceptance testing 49

**8. RESULTS 50**

**9. CONCLUSION & FUTURE ENHANCEMENT 59**

**10**. **BIBLIOGRAPHY 61**

**11. YUKTHI INNOVATION CERTIFICATE 63**

**ABSTRACT**

Internet of Things (IoT) has become one of the fastest-growing technologies and has been broadly applied in various fields. IoT networks contain millions of devices with the capability of interacting with each other and providing functionalities that were never available to us before. These IoT networks are designed to provide friendly and intelligent operations through big data analysis of information generated or collected from an abundance of devices in real time. However, the diversity of IoT devices makes the IoT networks’ environments more complex and more vulnerable to various web attacks compared to traditional computer networks. In this article, we propose a novel ensemble deep learning based web attack detection system (EDL-WADS) to alleviate the serious issues that IoT networks faces. Specifically, we have designed three deep learning models to first detect web attacks separately. We then use an ensemble classifier to make the final decision according to the results obtained from the three deep learning models. In order to evaluate the proposed WADS, we have performed experiments on a public dataset as well as a real-word dataset running in a distributed environment. Experimental results show that the proposed system can detect web attacks accurately with low false positive and negative rate.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Fig. Name** | **Page** |
| figure number 1.4.1 | Proposed Architecture | 03 |
| figure number 1.4.2 | Train – Test Split | 04 |
| figure number 1.4.3 | TF - IDF | 04 |
| figure number 1.4.4 | Architecture of bagging Classifier | 08 |
| figure number 1.4.5 | Architecture of boosting Classifier | 09 |
| figure number 3.1 | Architecture EDL - WADS | 15 |
| figure number 3.2.1 | Use Case Diagram | 17 |
| figure number 3.2.2 | Sequence Diagram | 18 |
| figure number 3.2.3 | Activity Diagram | 19 |
| figure number 3.2.4 | Class Diagram | 20 |
| figure number 3.2.5 | Deployment Diagram | 21 |
| figure number 3.2.6 | Component Diagram | 21 |

|  |  |
| --- | --- |
| **TABLE NAME** | **PAGE.NO** |
| Hardware Requirements | 22 |
| Software Requirements | 22 |

**LIST OF TABLES**

**CHAPTER 1**

**SYSTEM ANALYSIS**

**1. SYSTEM ANALYSIS**

**1.1 EXISTING SYSTEM**

* The method based on deep learning makes full use of the advantages of big data analysis and can detect web attacks more comprehensively and accurately. Ma et al. used static features and evaluated the methods with the Naive Bayes model, support vector machine (SVM), and logistic regression (LR). The results show the deep learning model’s capacity of identifying web attack through these static features.
* Yong et al. [20] proposed a new automatic method to analyze URL requests. Specifically, authors analyzed tokenized URL requests with three-grams and transformed them into vectors based on the likelihood ratio test. This method with the long short-term memory(LSTM) model obtained 90.60% in accuracy.

**1.2 DIS-ADVANTAGES OF EXISTING SYSTEM**

* As for deep learning models, systems proposed in existing studies mostly used single and simple models, such as SVM, LR, CNN, RNN, Naive Bayes, and Random Forest while there is a risk that the system with the single model may be bypassed of detecting specific attacks performed by hackers with new techniques.
* Hence, we conduct our research with automatically dissecting URL requests and utilizing three independent deep learning models for classification

**1.3 PROPOSED SYSTEM**

* We propose EDL-WADS, a novel ensemble deep learning-based system that can detect anomalous queries in which malicious codes are attached in an IoT network.
* We utilize a group of deep learning models to produce different representations of URL requests in order to exploit the advantages from a variety of classification. Al
* An ensemble classifier is utilized in EDL-WADS to improve the detection performance by combining results from different classifiers.
* We compare our proposed approach with several existing approaches deployed in a distributed environment. Our experimental results confirm the effectiveness and superiority.
  1. **ADVANTAGES OF PROPOSED SYSTEM**
* It demonstrates that the comprehensive check and ensemble classifier have the capability of combining results from multiple deep learning models accurately and comprehensively.
* As a result, it helped improve the detection performance of EDL-WADS

This section describes the proposed ensemble machine learning model for the detection of phishing intrusions from URL dataset. Fig. 1.4.1 demonstrate the proposed architecture of Web attack detection using ensemble machine learning model, where the system training phase includes data uploading, preprocessing, building, and training the machine learning model such as logistic regression, and XGBoost, and performance evaluation. Here the performance evaluation is done using the calculation of confusion matrix, and training accuracy. The second phase i.e., prediction involves the test URL data uploading, preprocessing, TF-IDF vectorizer, applying XGBoost model and final prediction of given URL.

**Data Preprocessing in Machine learning**

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.When creating a machine learning project, it is not always a case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So, for this, we use data pre-processing tasks.

Diagram

Description automatically generated

**Fig. 1.4.1. Proposed architecture for web attack detection from URLs using ensemble learning model.**

**Why do we need Data Pre-processing?**

Real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing requires tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

* Getting the dataset
* Importing libraries
* Importing datasets
* Finding missing Data
* Encoding categorical Data
* Splitting dataset into training and test’s

**Splitting the Dataset into the Training set and Test set**

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

A picture containing shape

Description automatically generated

**Fig. 1.4.2 : Train- Test Split**

**Training** **Set**: A subset of dataset to train the machine learning model, and we already know the output.

**Test** **set**: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

**TF-IDF Feature extraction**

TF-IDF which stands for Term Frequency – Inverse Document Frequency. It is one of the most important techniques used for information retrieval to represent how important a specific word or phrase is to a given document. Let’s take an example, we have a string or Bag of Words (BOW) and we have to extract information from it, then we can use this approach.

Diagram

Description automatically generated

**Fig. 1.4.3. TF-IDF block diagram.**

The TF-IDF value increases in proportion to the number of times a word appears in the document but is often offset by the frequency of the word in the corpus, which helps to adjust with respect to the fact that some words appear more frequently in general. TF-IDF use two statistical methods, first is Term Frequency and the other is Inverse Document Frequency. Term frequency refers to the

total number of times a given term t appears in the document doc against (per) the total number of all words in the document and The inverse document frequency measure of how much information the word provides. It measures the weight of a given word in the entire document. IDF show how common or rare a given word is across all documents. TF-IDF can be computed as tf \* idf

TF-IDF do not convert directly raw data into useful features. Firstly, it converts raw strings or dataset into vectors and each word has its own vector. Then we’ll use a particular technique for retrieving the feature like Cosine Similarity which works on vectors, etc.

**Terminology**

t — term (word)

d — document (set of words)

N — count of corpus

corpus — the total document set

**Step 1: Term Frequency (TF):** Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, “Data Science is awesome!” A simple way to start out is by eliminating documents that do not contain all three words “Data” is”, “Science”, and “awesome”, but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency.The weight of a term that occurs in a document is simply proportional to the term frequency.

**Step 2: Document Frequency:** This measures the importance of document in whole set of corpora, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d, whereas DF is the count of occurrences of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

**Step 3: Inverse Document Frequency (IDF):** While computing TF, all terms are considered equally important. However, it is known that certain terms, such as “is”, “of”, and “that”, may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.TheIDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as “is” is present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

Now there are few other problems with the IDF, in case of a large corpus, say 100,000,000 , the IDF value explodes , to avoid the effect we take the log of idf . During the query time, when a word which is not in vocab occurs, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

The TF-IDF now is at the right measure to evaluate how important a word is to a document in a collection or corpus. Here are many different variations of TF-IDF but for now let us concentrate on this basic version.

**ALGORITHMS**

* In EDL-WADS, Deep Learning Models are the key module for detecting web attacks. According to the feature vectors provided in the model of feature learning, we utilized three deep learning models for classification, they are the MRN model, LSTM model, and CNN model, respectively.

**1. MRN (Multi Residual Network)**

* Multiple Residual Networks (MRN) refer to an ensemble of residual networks**.**
* By including skip connections, residual networks (ResNets) solve the vanishing gradient issue and enable the training of very deep networks.

**2. LSTM (Long Term Short Memory)**

* Use LSTMs to capture temporal dependencies in sequential data, such as time-series data representing IoT device behaviors.
* The LSTM model can identify and report instances of anomalies in the IoT data that can be related to a possible online attack.

**3. CNN (Convolution Neural Network) MODEL:**

* CNNs are a useful tool for identifying IoT-based attacks by selecting important features from the dataset.
* CNNs are able to determine which elements are most important for identifying Internet of Things (IoT) attacks by utilizing their capacity to learn complex patterns and representations from collected data

**Ensemble Classifier:**

**Logistic Regression**

* Once the individual models are trained, combine their predictions using a logistic regression ensemble classifier.
* Use logistic regression to learn how to best weigh the predictions of each model to make a final decision.
* The logistic regression ensemble classifier can effectively combine the strengths of each model, potentially improving overall performance and robustness.

**XGBoost Algorithm**

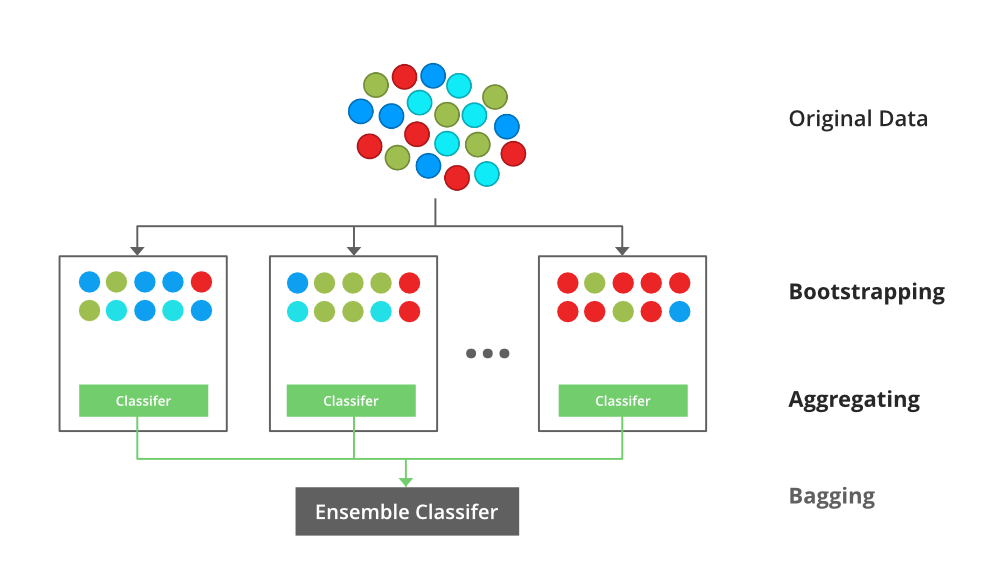
XgBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting. Before understanding the XGBoost, we first need to understand the trees especially the decision tree.

**Decision Tree**

A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

**Bagging**

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.  
Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, examples(or data) from the original training dataset, where is the size of the original training set. The training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out. Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.



**Fig. 1.4.4 : Architecture of bagging classifier**.

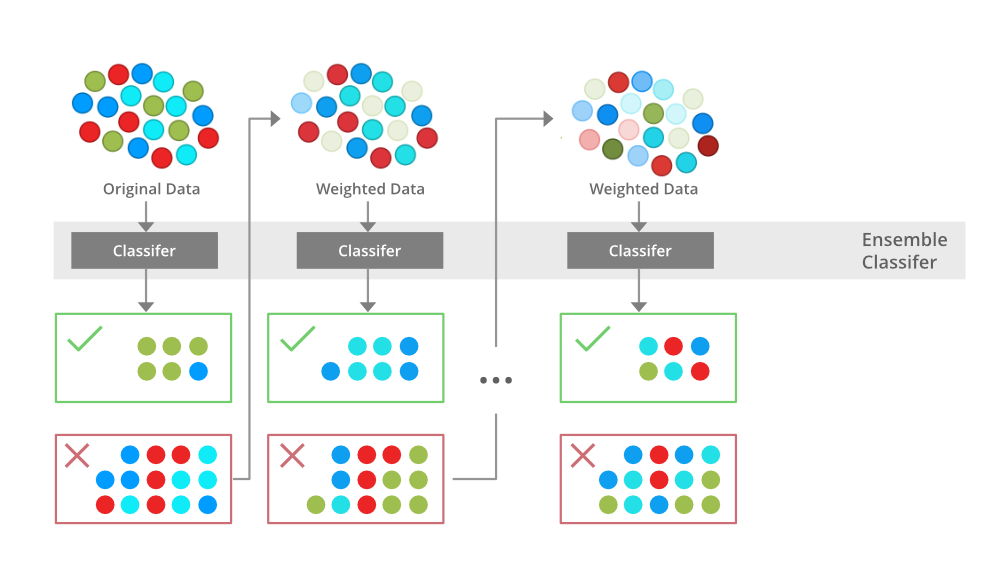
**4.3.3 Random Forest**

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn’t depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

**4.3.4 Boosting**

Boosting is an ensemble modelling, technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.



**Fig. 1.4.5: Architecture of boosting.**

**Gradient Boosting**

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor’s error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels. There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

**XGBoost**

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

**1.3 INTRODUCTION**

AS ONE of the fastest-growing and widely used technologies on the Internet, Internet of Things (IoT) extends the edge of the Internet by connecting additional terminal devices and facilities on the edge of the network. Specifically, IoT contains millions of devices with the capability of interacting with each other and providing great convenience for us. Via IoT technology, smart cities, smart home, smart medical treatment, smart agriculture, and other smart fields are emerging. Our ways of life and work are becoming easier, more efficient, more interesting, and more convenient. There are millions of IoT devices all over the world, some of which are visible to us while others are not. The data collected from these devices and stored in datacenters contain vast amounts of information, which may contain individuals’ private information. More visible and invisible threats are emerging and causing irrecoverable damages. Due to the high concentration of various information, attackers often select storage and service servers as a primary attack target. Once the attackers gain access to the central severs, data breaches are inevitable. Furthermore, the local storage and computing limitations of IoT devices prevent them from detecting and defending against potential web attacks. A minor security threat has the potential to cause severe damage to IoT networks. Therefore, there is no doubt that ensuring the security of IoT networks is of great significance to the success of IoT applications. Compared with traditional computer networks, there are more terminal devices and traffic in IoT networks, which make IoT network security issues more complex and troublesome. Recent works covering web attack detection systems (WADS) have shown a great capacity for the protection of traditional networks. However, these systems have faced severe challenges when utilized in IoT networks. Thus, there is an urgent need for research into more progressive systems to protect IoT networks from various web attacks. As web attacks grow rapidly in sophistication and diversity, researchers of network security are actively exploring new security technologies based on deep learning. While traditional web attack detection technologies show weaknesses in big data environment, the rise of deep learning provides novel solutions to security problems in such environments. Deep learning applications, based on big data analysis, show superior capacity for detecting aggression through massive traffic flow. These deep learning solutions have helped to advance and facilitate the development of IoT network security. In this article, we propose a novel WADS for IoT networks, based on ensemble deep learning. Specifically, the proposed EDL-WADS takes advantage of deep learning models to analyze uniform resource locator (URL) requests in the network traffic and identify anomalous requests within which web attack payloads are attached. In our approach, three deep learning models are employed to each learn relative features hidden in the queries. We use different methods to process and transform URL requests into different types of representations in order to exploit the advantages from a variety of deep learning models. Moreover, we employ an ensemble classifier to conduct a comprehensive analysis of the results of these three deep learning models. The ensemble classifier is designed to allow EDL-WADS to overcome the weaknesses of the individual classifiers and combine their advantages to improve the detection performance.

The contributions of this article are summarized as follows.

* We propose EDL-WADS, a novel ensemble deep Learning-based system that can detect anomalous queries in which malicious codes are attached in an IoT network.
* We utilize a group of deep learning models to produce different representations of URL requests in order to exploit the advantages from a variety of classification.
* An ensemble classifier is utilized in EDL-WADS to improve the detection performance by combining results from different classifiers based on multilayer perceptrons (MLP).
* We compare our proposed approach with several existing approaches deployed in a distributed environment. Our experimental results confirm the effectiveness and superiority of EDL-WADS in detecting IoT web attacks in real time.

**CHAPTER 2**

**LITERATURE SURVEY**

**2. LITERATURE SURVEY**

**[1]. Ma et al. “proposed a system by using static features and evaluated the methods.”** In this paper with the Naive Bayes model, support vector machine (SVM), and logistic regression (LR). The results show the deep learning model’s capacity of identifying web attack through these static features.

**[2]. Kar et al. “proposed a system for web attack detection, in which the method based on statistical characteristics.”** He proposed a system for web attack detection.in which the method based on statistical characteristics is used to represent URL requests, and a novel deep learning model is used to do classification task. The results achieved a high accuracy of 96.37%. Compared with the traditional detection method, deep learning approaches based on statistical characteristics make a significant increase in the result accuracy. However, there are two drawbacks of this method: first, it costs a lot in defining the special dictionary; second, the dictionary cannot include all anomalous words of expressions. Consequently, the hackers can bypass the matching rules with constantly changing payloads. Actually, features extracted by the method based on semantic analysis uses their statistical characteristics.

**[3]. Lee et al. “proposed a novel method to detect SQL injection.”** He proposed a novel method to detect SQL injection with removing values of SQL queries and comparing them with predetermined syntactic rules. Compared with other approaches, the results show that the proposed method is simpler and more effective.

**[4] C. Torrano - Gimenez, H. T. “Nguyen used semantic tools to get a syntax tree from URL requests and defined various of statistical characteristics based on the syntax tree.”** Theyused semantic tools to get a syntax tree from URL requests and defined various of statistical characteristics based on the syntax tree. Experimental results showed that their approach achieved promising performance in web attack detection. Compared with the former method, the second method reduces manual intervention to some extent and overcomes the disadvantage of the first method. However, the second method does not show significant improvement in the performance of web attack detection.

**[5]. Kar et al. “proposed a method based on digraph to analyze and transform URL requests automatically.”** This method results show that the proposed method per formed well and obtained the highest accuracy at 99.63%.

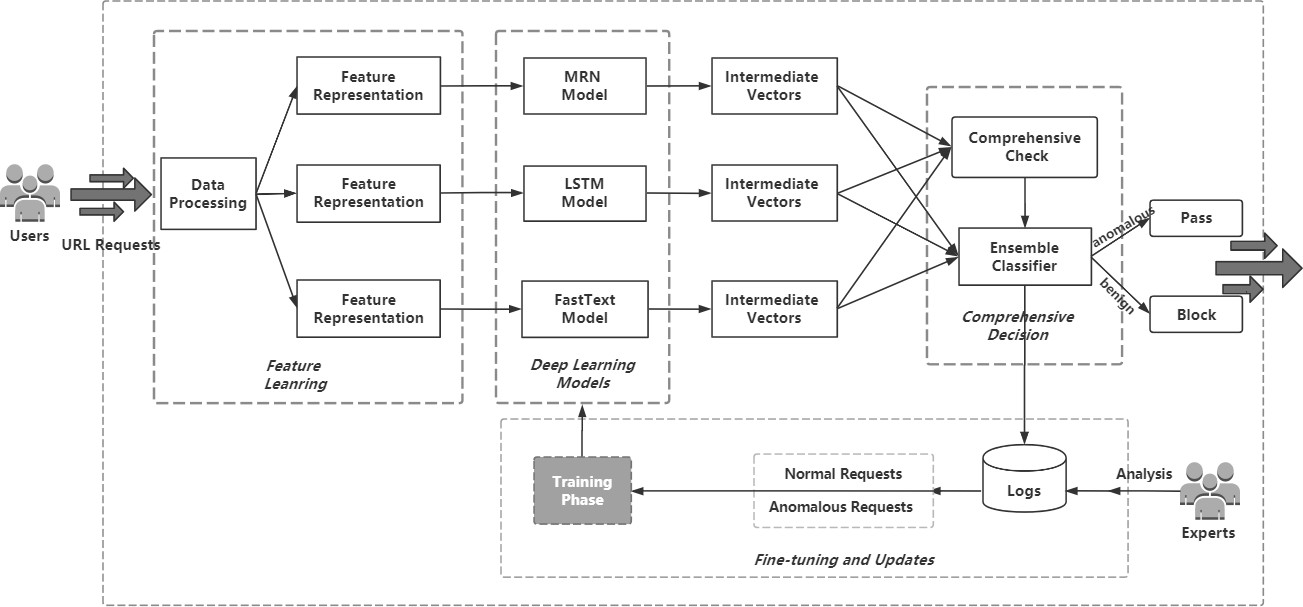
**[6] F. Yong, “proposed a new automatic method to analyze URL requests.”** He proposed a new automatic method to analyze URL requests. Specifically, authors analyzed tokenized URL requests with three-grams and transformed them into vectors based on the likelihood ratio test. This method with the long short-term memory (LSTM)modelobtained98.60%inaccuracy.

**[7]. Saxe and Berlin**, “**described a novel method for automatic analysis.”** which is to add an embedding layer in Convolutional Neural Networks (CNN). The optimal representation for URL requests will be generated through the training for the whole deep learning model. Compare with baseline models, this work performed better and achieved the highest accuracy at 99.3%.

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 SYSTEM ARCHITECTURE**



**Fig. 3.1. Architecture EDL-WADS**

**3.2 MODULES**

* **upload dataset:** using this module we will upload dataset
* **Dataset cleaning:** using this module we will find out empty values in the dataset and replace with mean or 0 values.
* **Train & Test Split:** Using this module we will split dataset into two parts called and training and testing. All machine learning algorithms take 80% dataset to train classifier and 20% dataset is used to test classifier prediction accuracy.

**3.3 BLOCK DIAGRAM:**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:** The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**Use Case Diagram**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:** The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**Use Case Diagram**

## A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram.

Diagram

Description automatically generated

**Fig. 3.2.1: Use Case Diagram**

**Sequence Diagram**

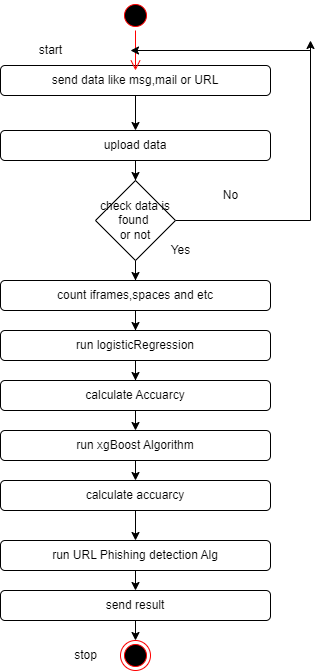
Represent the objects participating in the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

Diagram

Description automatically generated

**Fig. 3.2.2: Sequence Diagram**

**Activity diagram:** Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.



**Fig. 3.2.3: Activity diagram**

**Class Diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

A picture containing box and whisker chart

Description automatically generated

**Fig. 3.2.4: Class Diagram**

**Deployment diagram:** The deployment diagram visualizes the physical hardware on which the software will be deployed.

Diagram

Description automatically generated

**Fig. 3.2.4: Deployment diagram**

**Component diagram:** Component diagram describes the organization and wiring of the physical components in a system.

Diagram

Description automatically generated

**Fig. 3.2.5: Component diagram**

**3.2 SYSTEM REQUIREMENTS**

**3.2.1 Hardware Requirements**

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

|  |  |
| --- | --- |
| Operating system | Windows, Linux |
| Processor | minimum intel i3 |
| Ram | minimum 4 GB |
| Hard disk | minimum 250GB |

**Table No. 1 Hardware requirements**

**3.2.2 Software Requirements**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

|  |
| --- |
| Python IDLE 3.7 version (or) |
| Anaconda 3.7 (or) |
| Jupiter (or) Google colab |

**Table No. 2 Software requirements**

**CHAPTER 4**

**INPUT & OUTPUT DESIGN**

**4.1 INPUT DESIGN**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**Input Stages**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**Error Detection**

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Classified As:**

* User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
* Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**User Initiated Intergfaces**

User initiated interfaces fall into two approximate classes:

* Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
* Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

* The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
* Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**Error Message Design**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system
* The existing system is completely dependent on the user to perform all the duties.

**4.2 OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**CHAPTER 5**

**SOFTWARE ENVIRONMENT**

1. **Software Environment:**

**5.1 What is python**

Below are some facts about python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon,

Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

[Machine Learning](https://www.geeksforgeeks.org/machine-learning/)

GUI Applications (like Kivy, Tkinter, PyQt , etc. )

Web frameworks like Django (used by YouTube, Instagram, Dropbox)

Image processing (like Opencv, Pillow)

Web scraping (like Scrapy, BeautifulSoup, Selenium)

Test frameworks

Multimedia

#### **5.1.1 History of python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's

Influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

**5.1.2 What can python Do?**

Python can build various data visualisations, like line and bar graphs, pie charts, histogrammes, and 3D plots. Python also has many libraries that enable coders to write programs for data analysis and machine learning more quickly and efficiently, like TensorFlow and Keras.

#### **5.1.3 why python**

Python is commonly used for developing websites and software, task automation, data analysis, and data visualisation. Since it's relatively easy to learn, Python has been adopted by many non-programmers, such as accountants and scientists, for a variety of everyday tasks, like organising finances.

#### **5.1.4 Python syntax compared to other**

Python's syntax is similar to English, which makes it much easier to learn. C++, though, is based on object-oriented concepts that deal with memory allocation. Writing the wrong program in C++ can destroy the entire system. Compared to Python, C++ is faster. '

##### **5.1.5 Uses of Python**

**Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

##### **5.1.6 Python Features:-**

Let’s see how Python dominates over other languages.

###### **Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

###### **Extensible**

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

###### **Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language. Improved Productivity

The language’s simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

###### **Simple and Easy**

When working with Java, you may have to create a class to print **‘Hello World’**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

**Object-Oriented**

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

**Free and Open-Source**

Like we said earlier, Python is **freely available.** But not only can you [**download Python**](https://data-flair.training/blogs/install-python-windows/) for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**Portable**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

**5.2 What is Machine Learning**

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting gin machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

**5.2.1 Categories of Machine Leaning**

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

**Need for Machine Learning**

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate, and solve complex problems. On the other side, AI is still in its initial stage and have not surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

**Challenges in Machines Learning**

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

1. Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
2. Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
3. Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.
4. No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
5. Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.
6. Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.
7. Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

**5.2.2 Applications of Machines Learning**

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML.

* Emotion analysis
* Sentiment analysis
* Error detection and prevention
* Weather forecasting and prediction
* Stock market analysis and forecasting
* Speech synthesis
* Speech recognition
* Customer segmentation
* Object recognition
* Fraud detection
* Fraud prevention
* Recommendation of products to customer in online shopping

**How to Start Learning Machine Learning?**

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is the Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So, this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

**How to start learning ML?**

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

**(a) Learn Linear Algebra and Multivariate Calculus**

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

**(b) Learn Statistics**

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!  
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

**(c) Learn Python**

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So, if you want to learn ML, it’s best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It’s best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

**(a) Terminologies of Machine Learning**

* Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
* Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
* Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
* Training – The idea is to give a set of inputs(features) and it’s expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
* Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

**(b) Types of Machine Learning**

* Supervised Learning – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
* Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
* Semi-supervised Learning – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
* Reinforcement Learning – This involves learning optimal actions through trial and error. So, the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

**Advantages of Machine learning**

**1. Easily identifies trends and patterns:** Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

**2. No human intervention needed (automation):** With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

**3. Continuous Improvement:** As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data, you have keeps growing, your algorithms learn to make more accurate predictions faster.

**4. Handling multi-dimensional and multi-variety data:** Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

**5. Wide Applications**: You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

**Disadvantages of Machine Learning**

**1. Data Acquisition:** Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

**2. Time and Resources:** ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

**3. Interpretation of Results:** Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

**4. High error-susceptibility:** Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

**5.2.3 Modules Used in Project**

**TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.‍

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

#### **5.5 SDLC**

Certainly! Developing a web-based graphical password authentication system involves various phases of the Software Development Life Cycle (SDLC). Here's an overview of how you can document each phase of the SDLC for such a system:

1. **Project Initiation:**

**Project Vision:** Define the high-level project vision and objectives, such as creating a more user friendly and secure authentication system using graphical passwords.

Stakeholder Analysis: Identify key stakeholders and their interests in the project.

Feasibility Study: Assess the technical, operational, and economic feasibility of the project.

1. **Planning:**

**Project Scope:** Clearly define what the graphical password authentication system will encompass.

**Project Schedule**: Create a timeline for development, including milestones and deadlines.

**Resource Allocation:** Identify the resources needed, including personnel, hardware, and software.

**Risk Assessment:** Identify potential risks and develop a risk mitigation plan.

1. **System Design:**

**High-Level System Architecture:** Provide an overview of the system's architecture, including components and their interactions.

**User Interface Design:** Describe the graphical elements and layout of the authentication system.

**Security Design:** Explain the security measures in place for protecting graphical password data.

**Database Design:** Outline the structure of the database where user accounts and password information will be stored.

1. **Implementation:**

**Coding:** Detail the actual programming and development work, including the selection of technologies and programming languages.

**Unit Testing:** Describe the testing of individual components to ensure they function as intended.

**Integration Testing:** Explain how different system components are integrated and tested together.

**Quality Assurance:** Document the quality control measures and processes in place.

1. **Testing:**

**System Testing:** Conduct comprehensive testing of the entire system to ensure it works as a whole.

**Security Testing:** Perform penetration testing and vulnerability assessments to identify and mitigate security issues.

**User Acceptance Testing (UAT):** Engage end-users to validate that the graphical password system meets their needs.

1. **Deployment:**

**Deployment Plan:** Describe the process of rolling out the system to production.

**Training:** Explain how training will be provided to system administrators and end-users. Data **Migration:** If migrating from an existing system, detail the data migration process.

1. **Operations and Maintenance:**

**Support and Maintenance Plan:** Outline the ongoing maintenance and support activities, including bug fixes and updates.

**Monitoring and Performance Tuning:** Document how system performance will be monitored and optimized.

1. **Evaluation and Feedback:**

**Post-Deployment Evaluation:** Gather feedback from users and stakeholders to assess the system's performance and user satisfaction.

**Continuous Improvement:** Outline plans for ongoing improvements and feature enhancements.

1. **Conclusion:**

Summarize the key outcomes of the project, including any lessons learned.

1. **Documentation:**

Store all project documentation, including design documents, test cases, and user manuals in a central repository for reference and future update.

**CHAPTER 6**

**SYSTEM STUDY**

**6 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis are:

**•** ECONOMICAL FEASIBILITY

**•** TECHNICAL FEASIBILITY

**•** SOCIAL FEASIBILITY

**6.1 ECONOMICAL FEASIBILITY**

The economic feasibility study demonstrates that investing in the development and deployment of a novel web attack detection system for IoT via ensemble classifier offers substantial benefits in terms of improved security, cost savings, increased efficiency, and competitive advantage. While initial investment costs may be significant, the long-term benefits outweigh the expenses, making it a viable and economically feasible solution for organizations seeking to enhance cybersecurity measures for IoT deployments.

**Development Costs:**

**Personnel:** Hiring skilled developers, data scientists, and cybersecurity experts for system design, implementation, and testing.

**Infrastructure:** Investment in hardware and software resources required for system development and deployment.

**Research and Development:** Costs associated with conducting research, experimentation, and prototyping.

**Operational Costs:**

**Maintenance:** Regular updates, patches, and maintenance to ensure the system's effectiveness and reliability.

**Monitoring:** Continuous monitoring of the system's performance and effectiveness in detecting web attacks.

**Training:** Training programs for IT personnel and end-users to utilize the system effectively.

**Deployment Costs:**

**Integration**: Costs associated with integrating the system with existing IoT infrastructure and networks.

**Deployment:** Expenses related to deploying the system across various IoT devices and networks.

**Improved Security:** Reduction in cybersecurity incidents and vulnerabilities targeting IoT devices.

**Cost Savings:** Potential cost savings associated with preventing security breaches, data breaches, and loss of reputation.

**Increased Efficiency:** Automation of web attack detection processes, leading to faster response times and mitigation of threats.

**Competitive Advantage:** Differentiation in the market by offering robust cybersecurity solutions for IoT environments.

**6.2 TECHNICAL FEASIBILITY**

The technical feasibility study demonstrates that the proposed novel web attack detection system for IoT via ensemble classifier is technically viable and capable of effectively detecting web attacks targeting IoT devices. The system's architecture, data collection, preprocessing, feature extraction, ensemble classifier training, evaluation, scalability, efficiency, integration, and deployment aspects have been thoroughly analyzed, indicating its feasibility for practical implementation in IoT environments.

**System Architecture:**

The proposed system utilizes an ensemble classifier approach to detect web attacks targeting IoT devices.

It comprises multiple classifiers trained on different subsets of data or using different algorithms.

Ensemble methods such as bagging, boosting, or stacking are employed to combine the predictions of individual classifiers.

The architecture includes components for data collection, preprocessing, feature extraction, classification, and decision-making.

**Data Collection:**

The system collects data from various sources, including IoT devices, network traffic, web servers, and sensors.

Data collection mechanisms may involve packet sniffing, log analysis, API monitoring, and anomaly detection techniques.

The collected data is labeled and used for training and testing the ensemble classifier models.

**Preprocessing and Feature Extraction:**

Raw data is preprocessed to remove noise, handle missing values, and normalize the data.

Feature extraction techniques are applied to extract relevant features from the data, including HTTP headers, payload content, user behavior patterns, and network traffic characteristics.

**Ensemble Classifier Training:**

The system trains multiple classifiers using diverse training datasets or algorithm variations.

Common classifiers used in ensemble learning include decision trees, random forests, support vector machines (SVM), and neural networks.

**Evaluation and Testing:**

The trained ensemble classifier models are evaluated using performance metrics such as accuracy, precision, recall, F1-score, and ROC curves.

Testing is conducted on both synthetic datasets and real-world IoT network traffic to assess the system's effectiveness in detecting web attacks.

**Scalability and Efficiency:**

The system's scalability is assessed to ensure it can handle large-scale IoT deployments and high-volume network traffic.

Efficiency metrics such as computational complexity, memory usage, and response time are evaluated to ensure real-time detection capabilities.

Parallel processing, distributed computing, and optimization techniques may be employed to improve system performance and scalability.

**Integration and Deployment:**

The system is integrated into existing IoT infrastructure, network architecture, and security frameworks. Compatibility with different IoT device platforms, communication protocols, and operating environments is ensured.

**6.3 SOCIAL FEASIBILITY**

The social feasibility study demonstrates that the proposed novel web attack detection system for IoT via ensemble classifier has the potential to address societal concerns related to cybersecurity, data privacy, and public safety. By engaging stakeholders, addressing ethical considerations, promoting public awareness, and complying with regulatory requirements, the system can garner social acceptance and support, paving the way for its successful adoption and implementation in IoT environments.

**Stakeholder Acceptance:**

The acceptance and support of stakeholders, including IoT device manufacturers, network operators, cybersecurity experts, regulatory bodies, and end-users, are crucial for the success of the proposed system.

Stakeholder engagement and communication strategies should be implemented to garner support and address concerns regarding privacy, data security, and system reliability.

**Public Perception:**

Public perception of the proposed system will play a significant role in its adoption and implementation.

Education and awareness campaigns can be conducted to inform the public about the importance of cybersecurity in IoT devices and the benefits of the proposed web attack detection system.

**Ethical Considerations:**

Ethical considerations related to data privacy, consent, fairness, and accountability should be addressed in the design and implementation of the system.

Measures should be taken to ensure that the system's operation complies with ethical principles and regulatory requirements governing data protection and cybersecurity.

**Societal Impact:**

The proposed system has the potential to have a positive societal impact by enhancing the security and safety of IoT devices and networks.

By detecting and mitigating web attacks targeting IoT devices, the system can help prevent data breaches, financial losses, and disruptions to critical services, thus safeguarding public safety and trust in IoT technologies.

**Adoption Challenges:**

Adoption of the proposed system may face challenges related to cost, complexity, interoperability, and resistance to change.

Efforts should be made to address these challenges through cost-effective solutions, user-friendly interfaces, compatibility with existing IoT infrastructure, and collaboration with industry stakeholders.

**Regulatory Compliance:**

The proposed system should comply with relevant laws, regulations, and industry standards governing cybersecurity, data protection, and privacy.

**CHAPTER 7**

**SYSTEM TESTING**

**7. SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

##### **7.1 TYPES OF TESTS**

###### **7.1.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

###### **7.1.2 Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

###### **7.1.3 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing.

Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

###### **7.1.4 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

**Features to be tested**

Verify that the entries are of the correct format

No duplicate entries should be allowed

All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

###### **7.1.5 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**CHAPTER 8**

**RESULTS**

1. **RESULTS**

To evaluate the proposed EDL-WADS, we conducted experiments on a synthetic dataset as a benchmark, a real-world dataset as well as a dataset collected in real-time by ourselves when performing attacks to the IoT network using attack tools such as sqlmap, Burpsuite, etc. As part of our experiments, we implemented EDL-WADS in a distributed environment and compared EDL-WADS with several approaches in the literature. A. Datasets and Metrics In order to evaluate EDL-WADS and compare it with existing approaches fairly, we used HTTP CSIC dataset 2010 (commonly referred to as CSIC 2010) [28] as a benchmark dataset. The CSIC 2010 dataset has been broadly used to evaluate IDS. It contains various web attacks including SQL injection, cross-site scripting (XSS), buffer overflow, etc. Significantly, we extract 3329 SQL samples, 2053 XSS samples, and 4812 benign samples and review them manually. Furthermore, we evaluate EDL-WADS on a real-word dataset, which is collected by a security company. There are 276 14 SQL queries, 248 34 XSS queries, and 524 48 benign queries in this dataset. Furthermore, the detection problem is served as a classification problem, and we calculate accuracy, true positive rate (TPR), false positive rate (FPR), precision using TP, TN, FP, and FN defined in [26]. B. Experimental Results and Discussion First, we conduct experiments on the MRN model. As shown in Fig. 4, the structure of the MRN model, four MRN layers are stacked to extract more semantic and statistic features. We set a value for every kernel based on the experience. We then combine these four kernels into six groups. The details of these kernel combination groups are listed in Table VI. In order to achieve the best group of kernels, we carried out experiments with six groups of kernels on CSIC 2010 dataset. The results are summarized in Fig. 8. More specifically, we first set group A based on our experiments and received promising results with accuracy, TPR, and FPR all higher than 98.5%. We then make little changes from group A to group C, the performance increased slowly and achieved the highest in group C. However, the performances of accuracy and precision came to a sharp drop. We come to a conclusion that the kernel with size of 7 × 7 is too wide to extract useful features for the MRN model. The accuracy and precision increased immediately when the kernel of 7 × 7 is replaced. In the feature representation, we map every word in the URL requests to a vector of k-dimension, which is the row of the input matrix. The kernel of m × 1 can extract static features of every word, so that group E comes last with no m × 1 kernel in it ,while group C performs best with two m × 1 kernels. Hence, we apply group C to the MRN model in our EDL-WADS. Next, we implemented experiments for comparing EDLWADS with existing approaches. In order to get a fair comparison, we conduct experiments on a benchmark dataset CSIC 2010. Particularly, three baseline models are tested. 1) A specially designed CNN for web attack detection (SDCNN) [. 2) A web application firewall (WAF) using a character-level CNN (CLCNN) . 3) A deep learning model consists of RNN and LSTM (RALM) . The performances for comparison are listed in Table VII. RALM performs the best with a slight advantage than CLCNN. It achieves 98.56% at accuracy, 98.77% at TPR, and 98.5% at precision. SDCNN comes last among the baseline models. All three models in EDL-WADS performed well, CNN models in EDL-WADS are very similar to CLCNN in all metrics. Actually, the CNN model in EDLWADS and CLCNN both used an embedding layer for feature representation. It seems that the embedding layer is effective in representing URL requests. The results of the LSTM model in EDL-WADS are slightly better than CNN. LSTM performs better than CNN in EDL-WADS, because the URL requests are mapped to vectors before being inputted in deep learning models in LSTM, while the mapping layer is embedded in the deep learning model in CNN and may be influenced in the training phase. The LSTM model in MRN achieves promising results. However, MRN shows better performance in all metrics. There may be two reasons. 1) The MRN and LSTM in EDL-WADS have the same feature representations but different use-patterns of feature vectors. 2) The LSTM model focuses on semantic analysis only, while the MRN model can extract both semantic features and statistic features depending on its special structure. Besides, the EDL-WADS system performs slightly higher than MRN and obtains the highest scores in accuracy, TPR and FPR. It demonstrates that the comprehensive check and ensemble classifier have the capability of combining results from multiple deep learning models accurately and comprehensively. As a result, it helped improve the detection performance of EDL-WADS. Furthermore, because of the limitations of the existing security dataset and the diversity of web attacks, public available datasets that are often used currently are not reliable enough to evaluate a WADS. Further comparisons are carried out to evaluate the capacity of web attack detection ability of the proposed EDL-WADS. More specifically, we conducted experiments on a real-world dataset collected by a security company. Compared with experimental results collected on CSIC 2020 dataset, the value of each metric of each approach is reduced. There are two main reasons according to our analysis. First, the CSIC 2010 dataset was generated in 2010, there are fewer types of web attacks than today. Second, the CSIC 2010 dataset is synthetic and collected in the labs on a network environment that is much simpler than in the real world. Therefore, the decrease in Fig. 9 reflects the importance of using a real-world dataset for evaluating research results in the field of network security. As shown in Fig. 10, MRN performs the best among three individual models and CNN performs the worst. Finally, EDL-WADS outperforms all three individual models. It demonstrates that EDL-WADS is capable of combing MRN, LSTM, and CNN models accurately. A comparison between EDL-WADS and existing approaches, which include DBPF, SDCNN, and CLCNN, is carried out, EDL-WADS achieves superior performance with 99.17% in accuracy, 99.26% in TPR, 99.17% in precision, and 0.93% in FPR. The experimental results demonstrate that EDL-WADS performs better than existing works and can detect web attacks accurately with low false positives and negatives. Eventually, another experiment is conducted to test how EDL-WADS performs in a real-world environment. For this purpose, we used a famous web application DVWA as a target and deployed EDL-WADS in a distributed environment to detect attacks against DVWA. Specifically, we take advantage of several security tools, which include sqlmap, burpsuite, and XSStrike, to launch attacks against DVWA. The experimental results are illustrated in Table VIII. The results obtained show that EDL-WADS achieves the highest accuracy, TPR, and precision as well as the lowest FPR. In this experiment, we collected 6075 anomalous requests from security tools and 4360 normal requests by programs automatically. The EDL-WADS system achieved 100% in TPR, which demonstrates that all web attacks are detected accurately. The other metrics also demonstrated high values. Only two of all requests are detected wrongly: normal requests were detected as malicious ones. The results seem to be unexpectedly ideal. After several rounds of analysis, we found out the reason: these security tools that we have used to perform attacks all use common and simple security rules to scan the target system. EDL-WADS detects such simple and common attacks with very high accuracy. Nonetheless, the EDL-WADS truly demonstrated its effectiveness on real-time web attacks detection, given these attack tools that we have selected are the most commonly used ones on the Internet.

This project describes the concept of phishing intrusion detection and classification using NLP-based Machine learning algorithms such as logistic regression and XGBoost classification. In addition, it will also detect phishing intrusion from test URL.

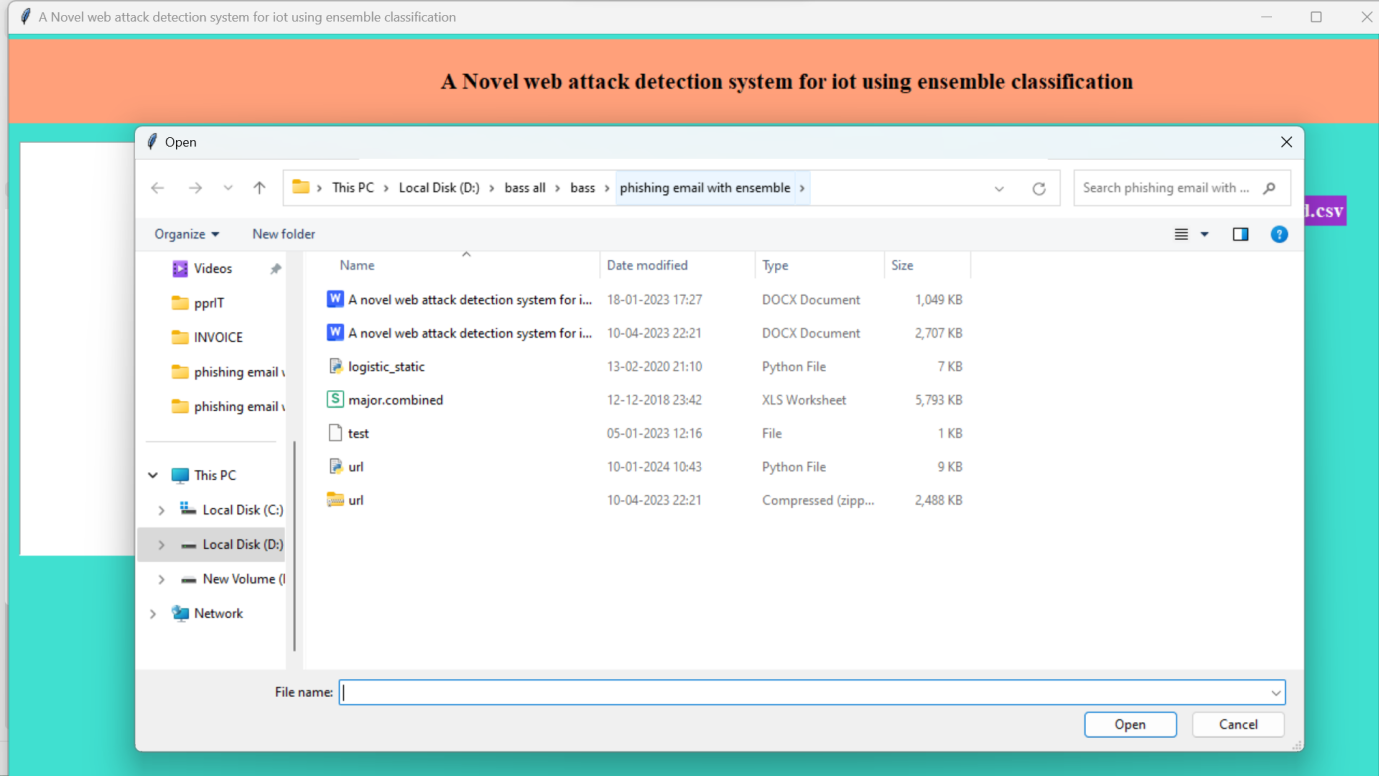


**Screen shot No.1**

Text

Description automatically generated

**Screen shot No.2**



**Screen shot No.3**

Text

Description automatically generated

**Screen shot No.4**

Text

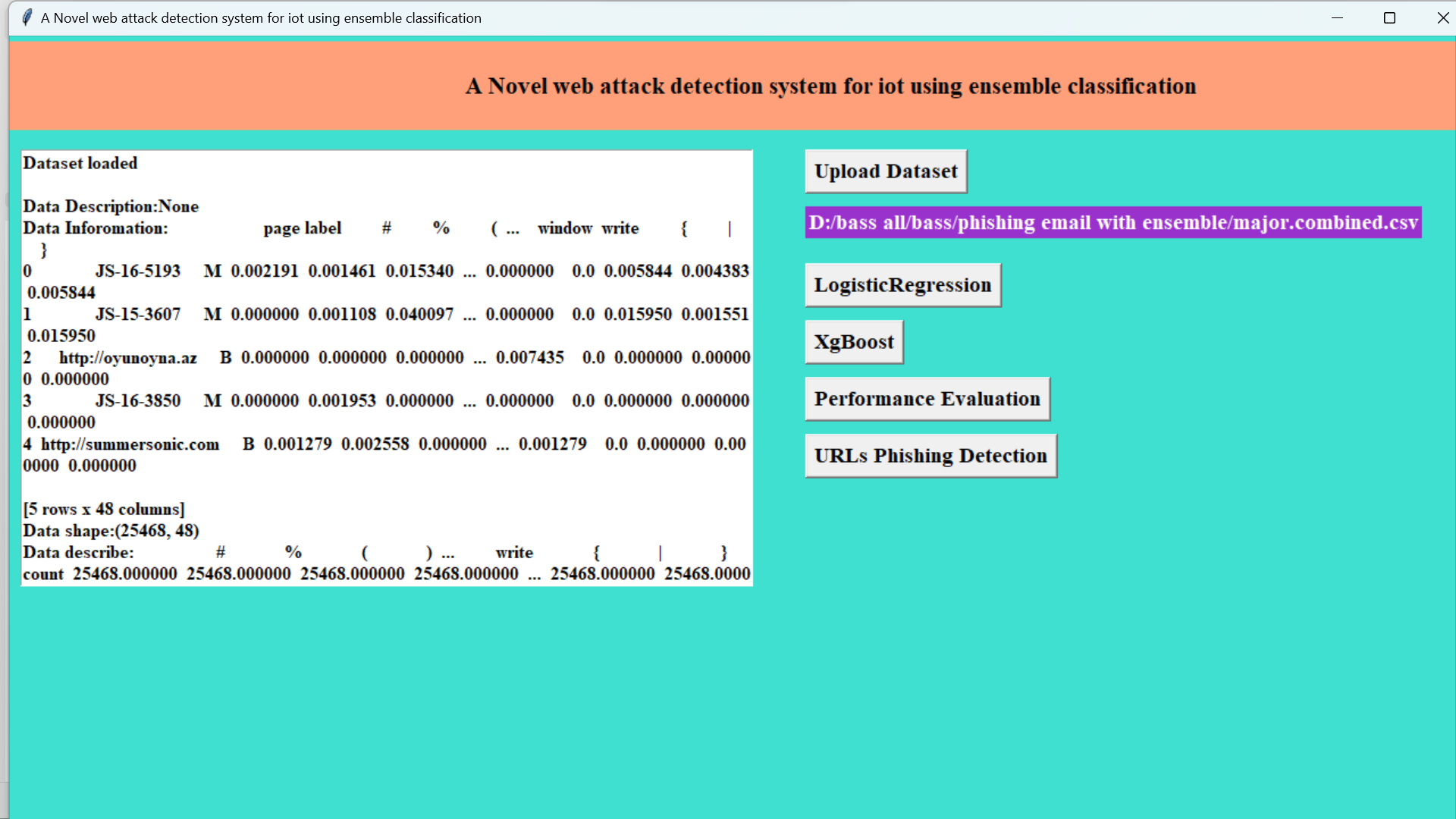
Description automatically generated

**Screen shot No.5**

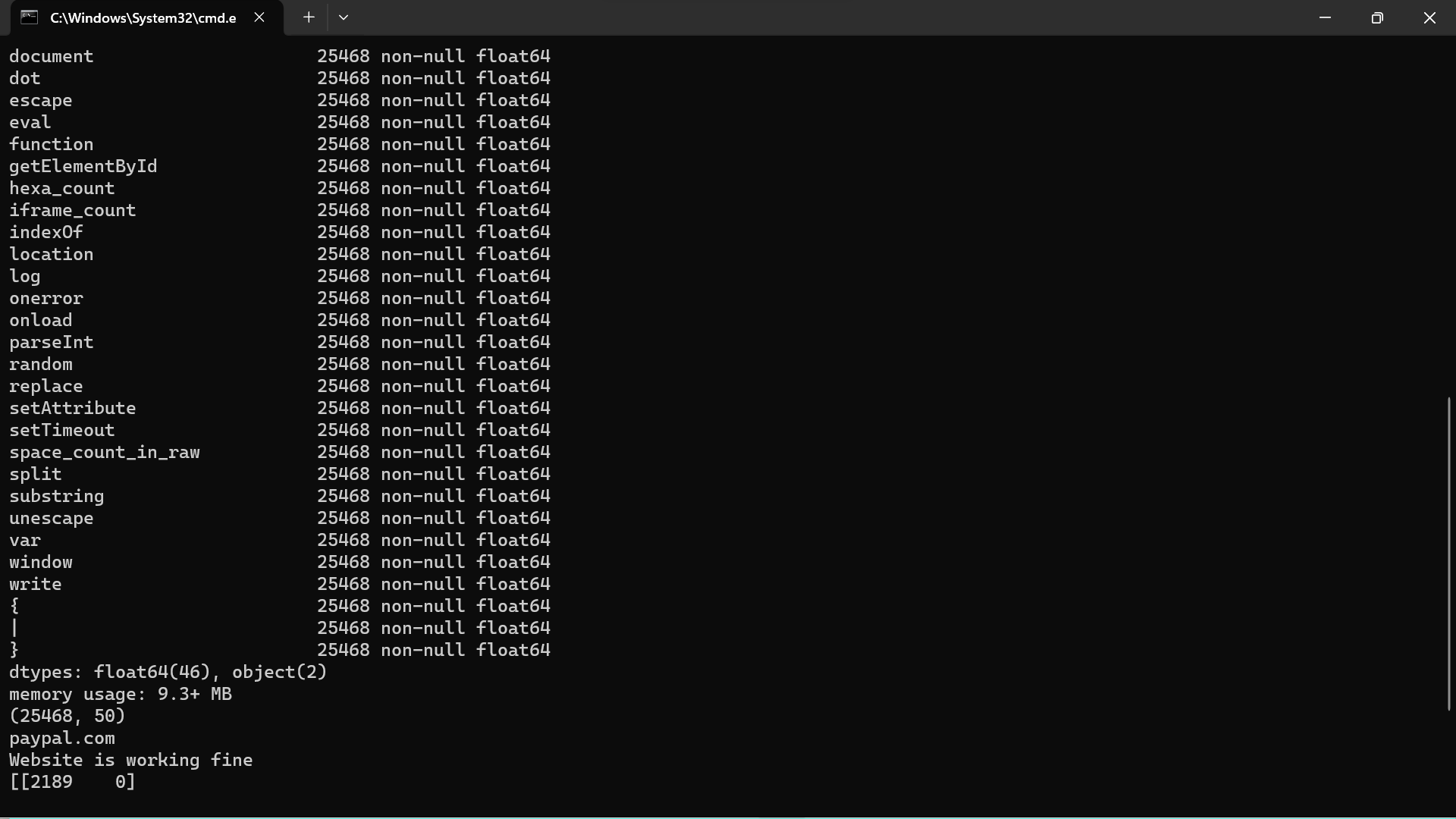
Chart, bar chart

Description automatically generated with medium confidence

**Screen shot No.6**



**Screen shot No.7**



**Screen shot No.8**

Text

Description automatically generated

**Screen shot No.9**

Text

Description automatically generated

**Screen shot No.10**

Text

Description automatically generated

**Screen shot No.11**

**CHAPTER 9**

**CONCLUSION AND FUTURE ENHANCEMENT**

1. **CONCLUSION**

In this article, we proposed a novel WADS, EDL-WADS, for IoTs. Specifically, the EDL-WADS consisted of four modules. 1) A feature learning module for URL request representations. 2) A deep learning module composed of three deep learning models for producing different representations of URL requests in order to exploit the advantages from a variety of classification. 3) A comprehensive decision module for combing the results from the three deep learning models and making the final decision with an ensemble classifier. 4) A fine-tuning and updates module for fine-tuning and updating the three deep learning models in real time. To evaluate the proposed EDL-WADS, we carried out experiments on different datasets. The experimental results on a benchmark dataset CSIC 2010 showed that EDL-WADS outperforms all selected baseline models. The overall performance was 99.47% on accuracy, 99.29% on TPR, and 99.70% on precision, with a low FPR of 0.0033. Furthermore, experiments were carried out on a real-world dataset. The results confirmed that EDL-WADS have a superior performance compared to several existing approaches. However, there were two primary limitations that required further improvement in the future. 1) The current EDL-WADS system can only detect SQL injection and cross-site scripting attacks. 2) The CNN model in EDL-WADS does not perform as well as we had expected; therefore, a more desirable model should replace it in the future. Thus, our future research direction will focus on improving the EDL-WADS for detecting additional types of web attacks (e.g., command injection and file inclusion) and exploring alternative deep learning models to better the performance of the current system.

**Future scope**

In our future work, fishing attacks will be predicted from the logged dataset of attacks by using a convolution neural network (CNN). It will be added as a tool for intrusion detection system and we plan to implement these solutions and develop a robust and generalized intrusion detection model.

**CHAPTER 10**

**BIBLIOGRAPHY**

**10. BIBLIOGRAPHY**

[1] M. Lin, C. Chiu, Y. Lee, and H. Pao, “Malicious URL filtering—A big data application,” in Proc. IEEE Int. Conf. Big Data, 2013, pp. 589–596.

[2] D. Kar, S. Panigrahi, and S. Sundararajan, “SQLiDDS: SQL injection detection using query transformation and document similarity,” in Proc. Int. Conf. Distrib. Comput. Internet Technol., 2015, pp. 377–390.

[3] A. Le, A. Markopoulou, and M. Faloutsos, “PhishDef: URL names say it all,” in Proc. IEEE INFOCOM, 2011, pp. 191–195.

[4] J. Qiu, L. Du, D. Zhang, S. Su, and Z. Tian, “Nei-TTE: Intelligent traffic time estimation based on fine-grained time derivation of road segments for smart city,” IEEE Trans. Ind. Informat., vol. 16, no. 4, pp. 2659–2666, Apr. 2020.

[5] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan, “Dynamic candidate evaluations for automatic prevention of SQL injection attacks,” ACM Trans. Inf. Syst. Secur., vol. 13, no. 2, pp. 398–404, 2010.

[6] C. Luo, S. Su, and Y. Sun, “A convolution-based system for malicious URL requests detection,” Comput. Mater. Continua, vol. 61, no. 3, pp. 399–411, 2019.

[7] M. Li, Y. Sun, H. Lu, S. Maharjan, and Z. Tian, “Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems,” IEEE Internet Things J., vol. 7, no. 7, pp. 6266–6278, Jul. 2020.

[8] Y. H. Hwang, “IoT security & privacy: Threats and challenges,” in Proc. 1st Acm Workshop on Iot Privacy Trust and Security, 2015, p. 1.

[9] A. Jamdagni, Z. Tan, and X. He, “RePIDS: A multi-tier real-time payload-based intrusion detection system,” Comput. Netw., vol. 57, no. 3, pp. 811–824, 2013.

[10] F. Yong, P. Jiayi, L. Liang, and H. Cheng, “WOVSQLI: Detection of SQL injection behaviors using word vector and LSTM,” in Proc. 2nd Int. Conf. Cryptography, Secur. Privacy, 2018, pp. 170–174.

[11] C. Torrano-Gimenez, H. T. Nguyen, G. Alvarez, S. Petrovi´c, and K. Franke, “Applying feature selection to payload-based web application firewalls,” in Proc. 3rd Int. Workshop Secur. Commun. Netw., 2011, pp. 75–81

**CHAPTER 11**

**YUKTHI INNOVATION CERTIFICATE**

**11.YUKTHI INNOVATION CERTIFICATE**

