Identification of IoT Vulnerabilities Using DRL

by

Joy Bordhen
16301121
Tarikul Islam
16301008
Tahmid Al Sakib
16301117
Tanvir Ahmed Joy
16301114
Sara Tasnim
16101165

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2020

Declaration

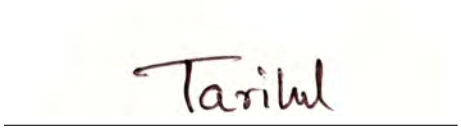
It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

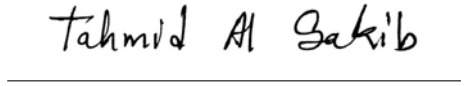
Student's Full Name & Signature:

Joy Bordhen

16301121



Tarikul Islam

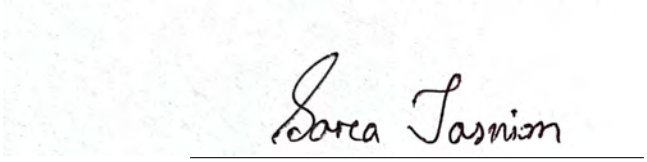


Tahmid Al Sakib

16301117

Tanvir Ahmed Joy

16301114



Sara Tasnim

16101165

Approval

The thesis titled "Identification of IoT Vulnerabilities Using DRL" submitted by

1. Joy Bordhen (16301121)
2. Tarikul Islam (16301008)
3. Tahmid Al Sakib (16301117)
4. Tanvir Ahmed Joy (16301114)
5. Sara Tasnim (16101165)

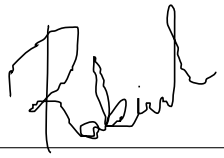
Of Summer, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 17, 2020.

Examining Committee:

Supervisor:
(Member)

Muhammad Iqbal Hossain, PhD
Assistant Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator: (Member)



Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department: (Chair)

Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
Brac University

Mahbubul Alam Majumdar, PhD

Professor and Dean, School of Data and Sciences,

Computer Science and Engineering

Brac University

Ethics Statement

We the members, hereby declare that this thesis is done based on the findings of our extensive research. All the materials and documents which have been used are properly noted and cited in this document. This work, neither in full nor any part has never been submitted by any other person to another university or any institution for the award of any degree.

Abstract

Security issues have been a threat to our modern life even after all the inventions in modern data networks. It has come to a stage where with great innovation comes greater risk. Recently, security issues in IoT devices are increasing day by day with the vast uses of IoT devices in our everyday life. In today's world, the most valuable thing is information, the more information you possess the richer and more powerful one is. Due to the increase in use, the online attackers have been recently targeting IoT devices to gain monetary benefit or acquiring different sensitive information from the crucial sources. Maximum IoT devices are very simple to get hands on illegally. Since IoT devices are not well equipped with proper security measures in order to keep them easily accessible and low cost, the hackers are easily accessing these devices. However, computer and digital devices often appear to become more unstable and vulnerable to malfunctions and vulnerabilities due to cyber attacks. Also, this is economically harmful which is quite frustrating. Securing these have been quite an important issue for many years. Thus, to make sure the information is safe and to overcome these vulnerabilities different measures are taking place to minimize these attacks. Several surveys are put forward to address these IoT based topics including intrusion detection, threat minimization and prevention of these attacks. It is not possible for the users to keep track of the intrusion every time manually. So an autonomous security measure should be taken to prevent the intrusion. In order to do this we are using Deep Reinforcement Learning technique to detect this intrusion and prevent them with the existing data and model as it is highly efficient in tackling complex, diverse and, in particular defense of high-extent cyber attacks. Reinforcement learning (RL) operates in a rotation of sense action goals. Since reinforcement learning acquires information directly from the environment, it is distinct from supervised learning that learns from the examples given. We are trying to detect and prevent malware attacks like viruses, ransomware and everything. For that, patterns need to be found by using deep learning or deep reinforcement learning algorithms. Furthermore, the system will be analyzing multiple attacks from systems that are already infected to do so and finally we will be coming up with a pattern created out of data analysis to prevent further attacks.

Keywords: Deep reinforcement learning, IoT, malware, Q-learning

Dedication

We would like to dedicate this research work to our parents. Without their utmost support, we could not have achieved what we achieved.

Acknowledgement

To commence with, all praise to the God for whom we could have completed our thesis in due time without any major complications in work. However, we would like to thank our honorable supervisor Dr. Muhammad Iqbal Hossain sir for his utmost support, guidance and feedback. He encouraged us to do our research constantly and give guidance whenever we faced any kind of problems. He was always present to offer his most valued help anytime we needed him. Moreover, we also want to thank our Brac university for providing us with the necessary facilities and resources during the whole period of our thesis.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ANN Artificial Neural Network

$DDQN$ Double Deep Q-Network

DL Deep Learning

DNN Deep Neural Network

DQN Deep Q-Network

DRL Deep Reinforcement Learning

IoT Internet of Things

MDP Markov Decision Process

ML Machine Learning

MLP Multilayer Perceptron

NN Neural Network

RL Reinforcement Learning

Chapter 1

Introduction

Deep learning is a machine learning's branch that gradually extracts higher level characteristics from the raw input using multiple layers. It is also called a subfield of machine learning, called artificial neural networks, inspired by the brain's structure and function. Deep learning is an autonomous method of self-teaching in which the current data is used to train the model and algorithms to identify the patterns and predict the new inputs using them. The test data set is always the same here, unless it is modified individually. Reinforcement learning (RL) is an autonomous self-teaching system. There are four important components in this system which are agent, environment, action and reward. An agent interacts with the environment and tries to learn an optimal policy by trial and error indicated by positive or negative reward [1]. RL is not supervised as there is no dataset with labels. It is also not unsupervised learning, as unsupervised learning is based on finding a definite form in collection of data which are unlabeled. Instead of trying to locate a secret structure, RL still seeks to optimize the incentive signal. In RL combinations of states and actions are stored in tables, which is too much information to handle. The integration of Reinforcement learning (RL) and Deep learning is Deep Reinforcement learning. It is useful when it may be too complex for RL to learn from all states and determine the reward path instead of mapping every solution. It creates manageable solution space. The combination of RL and deep learning techniques helps an RL agent to have a good perception of their environment by making use of the resources that deep neural networks can bring [2]. DRL is mainly used where it will learn from the existing test data and apply the best possible outcome. The outcome it

generated will again be added to the existing data dynamically so that the dataset is continuously upgraded with the new outcomes and new problems. For example- in Video games, chess, Go, Atari, Automated cars, Robotics, Finance, IoT etc. sectors DRL is used so that they can be self-autonomous.

1.1 Motivation

The Internet of Things (IoT) is a set of interconnected computer equipment, electrical and digital devices, gadgets with specific user identifiers and the capability to transmit data within a confined network without the need for human to human or human to computer communication. [3]. IoT is a new paradigm that focuses on linking devices, objects or "things" to each other, the Internet and users. The Internet of Things (IoT) nowadays is very common to human life due to its easy accessibility and comparative low cost. IoT devices are set up with different sensors which collect information to be passed to other computing locations and an IoT communication protocol stack has been developed by the IEEE and the IETF to ensure compatibility with other current internet devices [4]. Due to the increasing rate of using IoT devices the number of serious concerns about the dangers is vastly increasing. Data in these communication can range from ambient data (e.g. air quality, humidity level, and temperature) to health related data (e.g. users' heartbeats and peoples habits), and this list goes on, depending on the type of applications and the form of system. Every data is not private so that it has to be made confidential, but it has a growing need to assure that IoT devices are safeguarding information perceived and sent, both during transmission and during unused conditions[5]. At present there are so many companies manufacturing these IoT devices. In order to gain their profit or maximize sales they can compromise on security measures though these devices are not mostly designed with security mechanisms. To verify this the authors in a survey [6] found that some popular customer and commercial level IoT devices are connected with vulnerabilities in aspects of device security. There are different types of malware that seem perfectly normal but when in contact they can cause serious harm to the devices by invading, disabling or damaging the entire system. Another research ranks the physical or insight layer of the architecture in IoT devices under

the maximum risk zone in aspects of security because of different applications used in aggressive and open circumstances[7]. We use Deep Reinforcement Learning to detect and avoid these vulnerabilities, which will train customers about the possible threats of daily IoT based devices. At the initial stage, DRL can map the physical layer's possible security threats. DRL will allow the system to eventually learn the new attack stages after several iterations and will attempt to prevent them from using the conventional technologies.

1.2 Problem Statement and Thesis Objective

During the research we have found that there might be many security risks while using IoT devices. As the IoT devices are mostly made of low cost, the companies might not include better security measures in order to make it more profitable. From research, we have also observed that the percentage of information leakage from these IoT devices are of great numbers. If these types of security concerns raise to a higher volume, people might be discouraged to use these devices in order to keep their personal information safe. The main objectives of our thesis are-

- The main objective of our thesis is to detect malicious attacks on the device.
- The system will analyze different attacks from an infected system and find patterns on how it gathers information or attacks other devices by using various deep learning or deep reinforcement learning algorithms.
- We will create a pattern out of the data gathered from the attacks so that we can identify further attacks in future.

1.3 Thesis Structure

Our entire thesis is separated into several chapters. We have thoroughly outlined our work in each portion. In chapter 2 we have written a background which contains the literature review and algorithm description. Literature review is the summary of published papers of different notable researchers. We have read more than twenty papers and other research works to get a clear idea up to what extent others have

done. Maximum works are related to either Machine Learning or Deep learning. There are very few works based on Deep Reinforcement learning in IoT sectors. Algorithms played a vital role in our thesis as based on these we will get the prediction. Chapter 3 named the proposed model contains the methodology of our work, Dataset description and also a brief discussion on the models we have used in our thesis. The dataset we have used is new and there is not much published work based on the dataset. From chapter 4 our main work, that is experimentation has started. In that chapter we have implemented different Machine Learning Algorithms for the comparison with other algorithms. We have also implemented and found the result of an DL algorithm which is MLP. Among the DRL algorithms we have implemented DQN and DDQN. In the next chapter we have shown a brief result analysis from the experimentation we have done. Lastly we have concluded our thesis with the future plan of our work.

Chapter 2

Background

2.1 Literature review

Different types of IoT vulnerabilities and different theoretical approaches towards the solution applying machine learning techniques is presented in [8]. The paper provided seven attack surfaces. Such as, ecosystem access control and device memory, also there are device web interface,firmware and network services along with administrative interface. Vulnerabilities under the attack surfaces are SQL injection, DoS, weak passwords, account lockout, clear text account credentials, poorly implemented encryption etc. Authors suggested some algorithms to tackle these vulnerabilities and prevent intrusion. According to the paper K-means and DBSCAN could be used for pattern discovery. They proposed using SVM, KNN, Naive Bayes and random forest to discover unusual data points. For malware, intrusion and anomaly detection they suggested using SVM, Random Forest, KNN, ANN and Naive Bayes. This paper did not give any practical solution or comparison between different solutions and algorithms.

Classification of different types of vulnerabilities, solutions, comparison and the result is presented in [9]. Based on machine learning techniques, the authors investigated the IoT systems and reviewed the attack model. This paper has focused on supervised learning, unsupervised learning and reinforcement learning for solution. Solving different type of vulnerabilities are focused in this paper for IoT authentication. This paper proposes to use machine learning to control unwanted access,

detection of malware and privacy of data, secure offloading to use services from servers safely. Authors discussed and emphasized issues that are needed to be addressed in practical IoT devices. According to the paper, techniques of supervised learning like naive Bayes and support vector machine (SVM) along with K-NN which is called K-nearest neighbor, neural and deep neural network (DNN) and random forest which can be used for building the classification or regression model. They proposed applying SVM for locating intrusion of network and spoofing attack, exclusively malware detection they selected K-NN and detection and Dos attract they put neural network in the list. To detect intrusion in the IoT devices Naive Bayes can be used and also for detecting malwares, random forest classifiers can be used . For unsupervised learning authors suggested for detecting DoS attacks, multivariate correlation analysis and for authentication, IGMM. For reinforcement learning techniques authors suggested using Q-learning and Dyna-Q along with PDS which is also called post-decision state and DQN. These techniques enable prime parameters against various attacks through trial-and-error and also it enables IoT devices to decide the security protocols. According to the paper Q-learning could be used to improve authentication, malware detection and anti-jamming offloading. Authors further added that in terms of authentication and detecting malware Dyna-Q and PDS can be applied in IoT devices sequentially and for anti-jamming and offloading DQN can be applied.

Possibilities and possible issues regarding Machine Learning based solution of IoT vulnerability has been discussed elaborately in [10]. Authors classified attacks like buffer overflow attack, malware attack, phishing, exploitation of vulnerabilities of the WebApp, cryptographic attack, side channel attack, Intrusion, DoS, DDoS and man-in-the-middle attack into classes like application layer attacks, network level attacks, multilayered attacks, transport level attack etc. This paper proposed to use supervised, unsupervised, semi-supervised and deep reinforcement learning according to the need. Algorithms like KNN, Q-learning, Naïve Bayes, CNN, SVM, Random Forest, Decision tree etc. are proposed to be used for detecting anomaly and malware, authentication, DDoS etc. This paper also mentioned the limits and challenges of these solutions. It suggests that manipulating ML techniques and

training with wrong data to create vulnerabilities is possible. Authors argue that DL will not be great for real time IoT services as every time some new data is collected, it is needed to be trained again. This paper compared Neural Network with black boxes as it is not possible to know how it is determining the result. Authors mentioned that ML and DL are vulnerable to butterfly effects. A small change in the input can cause a drastic change in the result. It also states that collecting IoT data is important to make the system efficient but it is tough to collect IoT data as IoT data is very sensitive. Authors suggested doing more research on data collecting.

ANN to tackle IoT vulnerabilities used in [11]. In the paper authors used 4000 data points and five layers with three hidden layers within the neural network to get prediction to the accuracy of 99%.

DDoS attacks in IoT systems have been tackled in [12]. Authors performed four different classification of Deep Learning models on the CICIDS2017 dataset. Such as Multilayer perceptron (MLP), 1d-CNN, LSTM and CNN+LSTM. Later they compared their results with ML models. By comparing the results, the paper shows that Deep Learning models performed overall 9% better than Machine Learning models. CNN+LSTM gave the best accuracy of 97.16% amongst both ML and DL models.

Q-learning has been used as a model-free RL technique which can be implemented with low computation complexity in [13]. such that, IoT devices can utilize the Q-learning based offloading to choose their offloading data rates against network attacks, specifically jamming and spoofing. The foundation of offloading policy in terms of Q-function has been divided into several functions which are observing the task importance, the bandwidth of radio channel, formulating the present state by gaining channel and jamming power which is received by the IoT device. The Q-function represents the knowledge gained by the anti-jamming offloading which is generated in the previous state and also an expectation from it is to get discounted long-term reward for every action state pair. Through the Iterative Bellman equation the Q-value updates for each time slot with respect to ongoing offloading

policy. Authors also use IoT devices for assessing some specific functions such as SINL (signal-to-interface-plus-noise) ratio and to predict the utility of the offloading latency in this time slot. To maximize the current Q-function of the IoT device o-greedy algorithm has been applied which chose the offloading policy in that regard. This policy reduces the spoofing rate by half and decreases the jamming rate significantly by 8% where this rate is compared with the benchmark strategy.

Q-learning method can be used to overcome jamming attacks and choose the best channel for communication which is proposed in [14]. Here, The main frequency and bandwidth of radio is taken into account by the system and states are created based on that. The best channel for communication is selected depending on the present state and value of Q-function. Every time the system gets a new result after computing, it changes the values of Q. The authors got 53.8% better reward by using Q learning.

Q-learning algorithm to detect the best sub-band out of the radio spectrum to counter jamming and interference detected from other devices which used in [15]. The system calculates the usage of a frequency and set state values. Based on these the system chooses the best band for communication. It gave 44.3% increased cost of jamming for the proposed model comparing it to other benchmark strategies.

A system to identify software that is pirated and files infected by malware in an IoT network by using a combined deep learning approach proposed in [16]. Using source code plagiarism in order to identify pirated software they proposed to use deep neural networks in TensorFlow. They have suggested detecting plagiarism of source code by using deep learning. To detect noisy data, they suggested using tokenization and weighting the features. To investigate software piracy they collected a dataset from Google Code Jam (GCJ). To identify malicious activity in IoT networks by color image visualization they have used deep convolutional neural networks. For the experiment, they collected malware samples from a mailing dataset.

Attackers have increasingly targeted the IoT framework, while monetary gain and access to sensitive information are the most common for a variety of adversarial operations which is stated in [17]. The paper also mentioned various kinds of attacks like cookie stealing, Cross-Site Scripting (XSS), Structured Query Language (SQL) injection, session hijacking and mostly Distributed Denial of Service (DDoS). He studied a literature review of emerging IoT network developments based on IoT security mechanisms and analyzed different IoT application and communication protocols such as MQTT, XMPP, HTTP REST, AMQP, CoAP, UPnP, JMS. From the survey the paper said that the security issue of the application layer is critical and we need to pay particular attention.

A detailed list of state-of - the-art surveys dealing with different IoT model dimensions which is provided in [18]. Through amalgamating, evaluating and contrasting distributed research outputs, they aimed to promote IoT research efforts. They also developed a detailed taxonomy that outlined IoT vulnerabilities, attack vectors, impacts on different security goals, threats that exploit these vulnerabilities, associated remediation techniques, and offered operational cyber security capability to infer and monitor these vulnerabilities. They proposed a new, data-driven approach based on special, formerly untapped analytical data to produce notions of malice relevant to the IoT model on the Internet, which aims to illustrate the importance of highlighting the value of IoT in customer markets and vital infrastructure domains as it is implemented. The output of the methodology regarding cyber vulnerability information, malicious IoT data and IoT related attack footprint is made accessible to the scientific community as a whole to promote timely IoT security remediation and generally facilitate data-driven analysis through the use of empirical data relevant to IoT.

Machine learning models are often vulnerable to different types of cyber-attacks which is stated in [19]. They proposed to understand DRL vulnerabilities from different perspectives and offered a detailed taxonomy of potential attacks which

carry out the first set of experiments in the taxonomy on the unexplored parts. Besides the latest observation-based attacks on DRL, they proposed the first targeted attacks based on action space and dynamics of the system. Extensively studying the taxonomy of adversarial attacks on DRL systems they evaluated 10 adversarial attacks to gain points in the taxonomy which previously was not examined. Their two practical strategies for carrying out ad adversarial serial attacks are N-attack and online sequential attack which was under limited power consumption. They also suggested two methods to effectively query a model in black-box attacks: the method of adaptive dimension sampling based on finite difference (SFD) and the optimal method of frame selection. They provide a theoretical review of our proposed gradient estimation method and prove to be bound by its efficiency and estimation error and propose the first targeted attack which adversarial disrupts the dynamics of the environment of a DRL system to cause an agent to fail in a particular way, and this method is more realistic in real world applications.

The increasing cyber-attack which has greater impact on network services which plays a significant role in social and economic viewpoint discussed in [20]. Authors proposed to use labeled dataset to build a compact application containing some Deep Reinforce Learning(DRL) algorithms in order to detect intrusions.AWID and NSL-KDD, these two datasets are being used where authors decided to apply Deep Reinforced Learning(DRL) models.For detecting intrusions while running algorithms to find it, these two datasets are more appropriate than any other datasets that has been used for this purpose according to the authors. Authors made some modifications of classic DRL paradigm where they made a significant amount of changes in the environment.Some sampling function regarded cataloged training intuitions has been swapped with the environment. Significant DRL models have been used which are Deep Q-Network(DQN), Double Deep Q-Network (DDQN), Policy Gradient (PG) and Actor-Critic (AC). In terms of results DDQN algorithm stood out among four of these DRL models. Authors show that with their model and changing some parameters DRL provides not only better results but also it shows how it is superior than current Machine learning techniques for intuition detection. The

results showed that in the NSL-KDD dataset in terms of detection score, DQN and DDQN models (considering F1, accuracy and recall) obtain best results. DDQN model stood out for AWID dataset as well considering F1, accuracy and recall.

The decision tree (DT) and back-propagation neural network (NN) classifier's performance to detect attacks in the cloud environment is compared in [21]. Here these classifiers showed similar results in identifying probe attacks and DoS. However, decision tree shows better performance than NN while back propagating. Also, IDS was proposed to identify specified and unspecified attacks in cloud environments with K-means clustering which helps the network to flow into sixty-four clusters and SVM classification algorithms so that it can obtain high accuracy. Later, the researchers applied different models of deep learning(DL) which integrated two deep learning models- multinomial model and binomial classification model to detect an intrusion, and to identify the category of attack. Reinforcement learning is being used by researchers nowadays. Researchers addressed the problem of using reinforcement learning to detect intrusions. A Reinforcement Learning Multi-agent (MARL) framework is used for network detection of DDOS and DOS attacks. It is formed on Q-learning, which is made up of a series of a Decision Hierarchy(DA) agents and Heterogeneous Sensor Agents (SA). For any possible cause, in Q-learning, the result is measured iteratively in that state, that is kept in a form of table. So, to minimize the dynamics of space and the Deep Q Network (DQN) is presented. A state is given in Deep Q networks as an input and the predicted values for the potential acts obtained from the network are produced. As a result of that, complexity decreased.

Q-learning responds imperfectly in some unsettled environment although this is an established reinforced learning algorithm described in [22]. Broad over-estimations of action values play the vital role for this substandard performance. Author imprecise the largest expected value for a somewhat set of random variables which is a substitute way. Overestimated result from a pragmatic bias which inaugurated because Q-learning applied the largest action value for the approximation of largest

awaited action value. So, it is necessary sometimes to misconstrue more than overestimate the largest expected value and double estimator method indicate exactly that. Author added a double estimator method to Q-learning in order to build a new algorithm which is called Double Q-learning algorithm. Author even said that in some cases Q-learning acted inadequate along with elevated discount factors in some introductory experiments whereas Double Q-learning is less uninfluenced. Author experimented with two problems which are "roulette" and "Grid World" where he put a comparison between Q-learning and Double Q-learning to show which performs better and in both cases Double Q-learning shows better results on average reward.

2.2 Algorithms Description

Deep Learning (DL) is a machine learning subdivision that deals with algorithms motivated by the structure. This is also motivated by the brain's functions which is called artificial neural networks. Deep Learning in its every ranking algorithm submits its input as a non-linear variation and utilizes everything it finds during training to give output building an analytical or statistical model. Training iterations proceed until an appropriate degree of precision has been achieved by the output or unless the output is not satisfactory which is mainly the production. Deep learning benefits us constructing features without intervention, the algorithm constructs the feature adjusted by itself and no supervision is needed for that. We have used some algorithms of DL and DRL as the initial of our thesis.

2.2.1 Multilayer Perceptron (MLP)

A perceptron in neural networks is referred to as a unique neural construct that is a precursor to a broad neural network. A MLP(multi layer perceptron) is a perceptron that works with other perceptrons which are stacked in several layers in order to solve complex problems. Feedforward artificial neural networks (ANN) important class is multilayer perceptron thus sometimes rigorously refer to networks composed of multiple layers of perceptron. There are at least three layers in a multilayer perceptron (MLP). The initial layer will be the layer for taking inputs and the final

layer will be giving output and in the middle of them there will be a hidden layer. As the MLPs are totally connected to each other every node in each layer connects with a specific weight w_{ij} to other nodes in the following layer.

In the perceptron learning occurs by modifying the connection weights after every data is processed, comparing on the number of errors in the output result to the expected result.

It is important to understand that "multilayer perceptron" does not refer to a single perceptron which includes multiple layers; rather it refers to many perceptrons that are stacked into layers. According to the comprehending task, the hidden layer can be increased as much as we want in order to make the model more complex. In research multilayer perceptron is useful as it has the ability to solve problems stochastically.

2.2.2 Deep Q-Networks (DQN)

Neural networks are used to estimate the values of Q-function in deep Q-learning. In this algorithm the state is fed as input into the neural network and it generates outputs which are the Q-values of all feasible actions. Q-value function is updated of a particular state related to a particular action by DQN. The procedural steps of deep Q-learning networks (DQNs) are:

1. User saves all the previous knowledge in memory.

2. Next steps are always decided based on the highest values of Q-network.

3. The loss function is calculated here by taking the average of the squares of the error. It is called mean squared error (MSE). Here the error is the difference between the target Q-value–Q* and estimated Q-value. Here,

$$Loss = \frac{\sum_n (q_t - q_{ref})^2}{n} \tag{2.1}$$

   As this is a reinforcement learning based problem, the target Q-value is unknown here. The equation of updating Q-value can be obtained from the Bellman equation and the equation is:

$$Q(S_t, A_t)Q(S_t, A_t) + [R_{t+1} + maxQ(S_{t+1}, a) - Q(S_t, A_t)] \tag{2.2}$$

Though it may seem that the goal is calculating its value itself, reward R is true and not biased. So, the network is able to back-propagate and change the gradient and finally it converges.

Reinforcement Learning applies the Markov Decision Process (MDP). MDP consists of four elements S, T, A and R. Here, S represents the set of states, T represents the probabilities of conversion from every state's action to a new possible state, A represents the set of actions and R represents the function of reward that gives an actual value to every pair of actions. Transition probability of any new possible state is completely dependent on the existing state and action. In MDP, the main objective of this is to determine the maximum optimum policy which delivers the best possible action for each state which leads to an acceptable summation of all rewards. Mainly an MDP is a framework which characterizes an agent that interacts with an existing environment in a process that consecutively makes the decisions. Here, the existing environment executes the R function and T function whereas the model's agent executes the desired policy. Connection between the model's agent and environment is represented according to time-steps where the environment takes a new action from the agents and creates a whole new state transition with a new possible reward. Value functions are the estimation of the value of each state which makes the connection optimal. The value function is either a Q-function or a V-function. Knowing the R function and T function of the Markov decision process we can get an optimal policy. We have used model-free reinforcement learning methods which is also known as Temporal Difference (TD) Learning. This method considers a single transition of the states rather than of all possible transitions with slight modification of the bellman equation. In order to gain the optimum policy exploration of various state action spaces which are based on action-probability and ϵ-greedy is required. The probability in ϵ-greedy is either selected as p or $1 - p$ (random action). To sum up, learning the value or the policy function requires an iterative process where every iteration will generate a function adjustment.

2.2.3 Double Deep Q-Networks (DDQN)

While estimating actions, Q-Network causes overvaluation of the Q values in DQN algorithm. DDQN solves this issue by recommending the use of Q-Network for picking the action and using target Q-Network for estimating the action. In DQN, random experiences are allocated to upgrade one among the two separate value functions. In this way two separate value functions are created and learned. Also it provides two weights, θ and θ'.

One particular weight is responsible for determining greedy policy and another one is for calculating the value. One collection of weights is used with each update to decide the greedy policy, and the other to calculate its value. The target in Q-learning can be written as:

$$Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, argmax\ a\ Q(S_{t+1}, a; \theta_t); \theta_t) \tag{2.3}$$

The error in Double Q Learning is:

$$Y_t^{Double\,Q} = R_{t+1} + \gamma Q(S_{t+1}, argmax\ a\ Q(S_{t+1}, a; \theta_t); \theta'_t) \tag{2.4}$$

Weight θ_t is attributed to the argmax which refers to action selection. From this we get that the greedy policy value is calculated based on the present values as attributed in θ_t. The other set of weights θ_t is used to approximate the significance of this procedure. By exchanging the placement of θ and θ', the scope of the 2nd weights can be corrected balancedly.

Chapter 3

Proposed Model

3.1 Methodology

Firstly, we will take the labeled dataset of IoT devices. From that dataset we will have to exclude the useless data which is completely null. As these data will not affect our prediction result. The other columns will be our feature for our model. If there is both string and integer types of data, we will have to use a label encoder to convert the string value into integer value. We will have to take the dataset into a data frame so that the original csv file of the dataset does not get changed. After fully processing the data we will divide the dataset into tests and train data by 20 and 80 percent respectively, we will train the dataset. Applying the algorithm on our model we will get our result. We will apply different algorithms to find the best one which will give the best performance. We will evaluate our model performance with our expected result. If we do not get our expected result then we will change the parameters in our model and reapply the algorithm until we get our expected result. After minimizing all the factors we will get the best performing model which will be our output prediction.

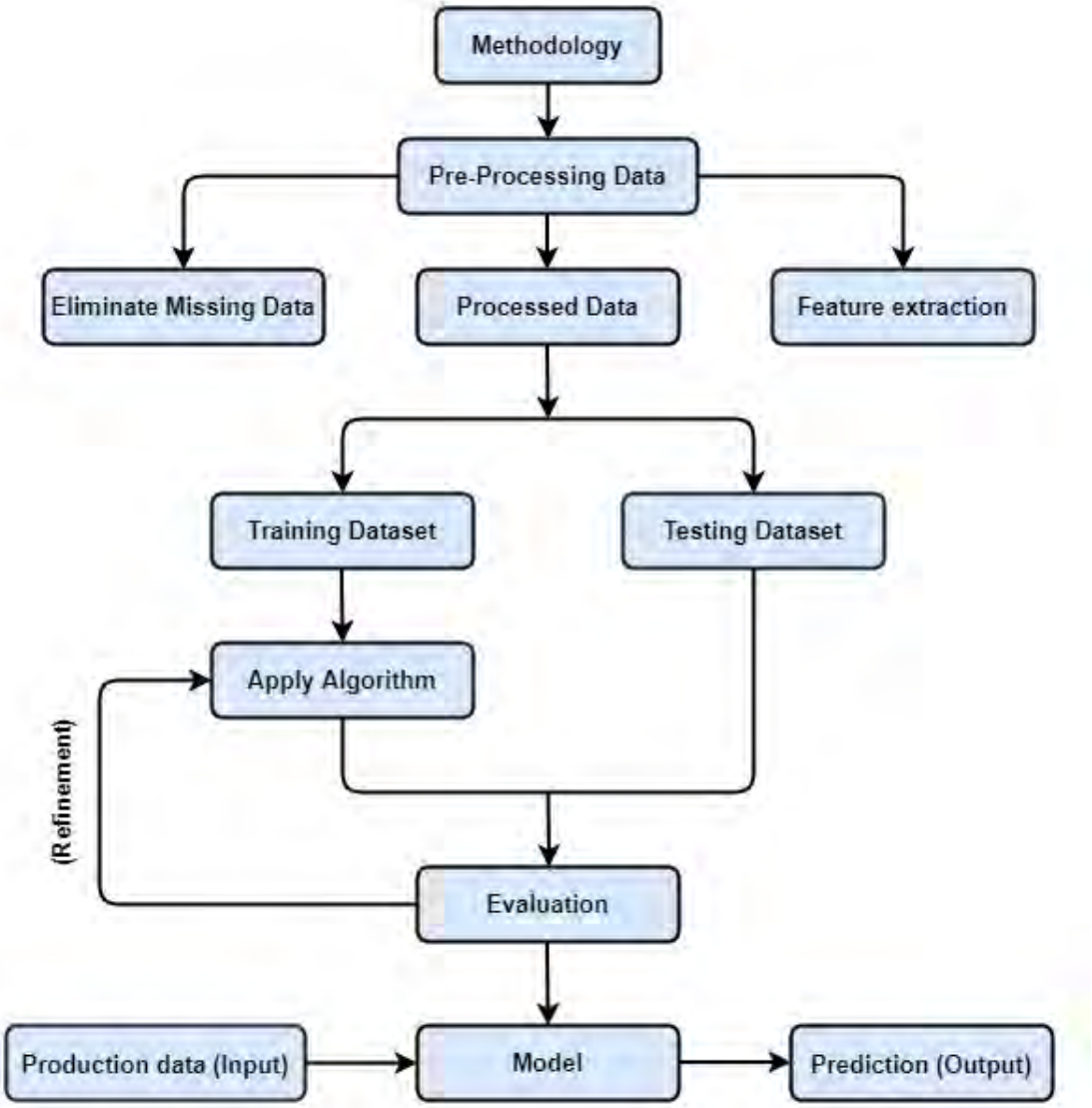


Figure 3.1: Workflow of the model

3.2 Dataset Description

The dataset we have used here is 'IoT-23: A labeled dataset with malicious and benign IoT network traffic' dataset provided by Stratosphere Laboratory and funded by Avast software. The dataset contains network traffics captured in three different IoT devices between 2018 and 2019. The devices are Somfy smart lock, Philips HUE smart LED lamp and Amazon echo. These captures were divided into 23 different sections. Twenty out of these captures represent malicious scenarios and other three represent benign or non-malicious scenarios. The captures were in the

.pcap file system. Then the data were compared with a CSV file containing rules and comparing criteria and the data were labelled accordingly by the original analyst of this dataset. Each labelled file has a labeling column which is a combination of columns tunnel_parents, label and detailed-label. The label column indicates whether a particular traffic flow is Malicious or benign. If the traffic flow is malicious then the detailed-label column specifically indicates what kind of malicious traffic flow it is. These malicious traffic flows are divided into different categories. Detailed information of those categories is given below:

Attack: this label indicates that other hosts were attacked by the infected device using any vulnerable services.

Benign: It shows that the data flow is not harmful.

C&C: This label shows devices which are linked to a Command and control server which can be used to carry out cyber-attacks ranging from DDoS to data stealing.

HeartBeat: this label shows that the device was monitored by the data packet.

DDoS: it shows that the infected device is participating in DDoS attack, or in other words it is being used by attackers to carry out DDoS attack.

FileDownload: it shows that some ill intended files were downloaded to the device from some suspicious source address.

Mirai: It shows that the data packet had similarity with Mirai botnet.

Okiru: It shows that the data packet had similarity with the Okiru botnet.

Torii: It shows that the data packet had similarity with Torii botnet.

PartOfAHorizontalPortScan: This label shows that attackers were using the

network to gather port information to carry out more attacks.

We have used Orange3 data mining tool to analyze the features of our dataset. We applied multiple feature analyzing methods like information gain, gini decrease, ANOVA etc (figure 3.2) to find out the important features. We found out that History is the most important features of all according to most of the methods.

	#	In...in	Gain ratio	Gini	ANOVA	x^2	ReliefF	FCBF
history		0.775	0.690	0.418	123591...	709481....	0.026	0.000
proto		0.774	0.712	0.417	175016...	175391....	0.020	2.894
orig_ip_bytes		0.745	0.456	0.405	120104....	503089....	0.020	0.000
id.orig_p		0.690	0.361	0.377	81311.6...	56687.1...	0.028	0.000
id.resp_p		0.528	0.266	0.297	100688...	516855....	0.240	0.548
orig_pkts		0.125	0.161	0.074	55114.4...	21772.0...	0.019	0.000
duration		0.113	0.105	0.068	43496.3...	231697....	0.002	0.000
resp_ip_bytes		0.013	0.074	0.007	3802.313	3809.689	0.009	0.000
orig_bytes		0.011	0.077	0.006	923.657	643.195	0.003	0.000
resp_bytes		0.009	0.072	0.005	1392.949	2708.587	0.001	0.016
resp_pkts		0.008	0.048	0.004	4085.680	8292.302	0.020	0.000
conn_state		0.007	0.025	0.004	10138.5...	186.618	0.042	0.000
service		0.003	0.093	0.001	2827.769	9.083	0.005	0.000
unknown		0.000	0.000	0.000	345.108	248.269	0.039	0.000
ts		0.000	0.000	0.000	337.247	243.722	0.039	0.000

Figure 3.2: Feature analysis

When we got the dataset we were provided with a conn.log.labeled file for each capture. As CSV file format is easy to handle and we are familiar with it, we have converted each conn.log.labeled file into CSV file format. The datasets included attributes (figure 3.3) like service, protocol, duration, history, conn_state etc. The combination of all the data is so huge that our personal computer could not handle it. So, for now we will only work with the capture CTU-IoT-Malware-Capture-3-1 and capture CTU-IoT-Malware-Capture-1-1 for experimentation.

The CTU-IoT-Malware-Capture-3-1 dataset contains a total of 156,103 flows. From figure 3.4, we see that only 4536 of them are benign and the rest of them are malicious traffic. From table 3.1 we see that 5,962 out of 151,567 malicious traffics are Attacks, 8 of them are C&C and 145,597 of them are PartOfAHorizontalPortScan traffic.

	id.orig_h	id.orig_p	id.resp_h	id.resp_p	proto	service	orig_bytes	resp_bytes	conn_state	history	orig_pkts	orig_ip_bytes	resp_pkts	resp_ip_by
0	192.168.2.5	38792	200.168.87.203	59353	tcp	0	0	0	S0	S	3	180	0	
1	192.168.2.5	38792	200.168.87.203	59353	tcp	0	0	0	S0	S	1	60	0	
2	192.168.2.5	38793	200.168.87.203	59353	tcp	0	0	0	S0	S	3	180	0	
3	192.168.2.5	38793	200.168.87.203	59353	tcp	0	0	0	S0	S	1	60	0	
4	192.168.2.5	38794	200.168.87.203	59353	tcp	0	0	0	S0	S	3	180	0	

Figure 3.3: First five rows of the dataset

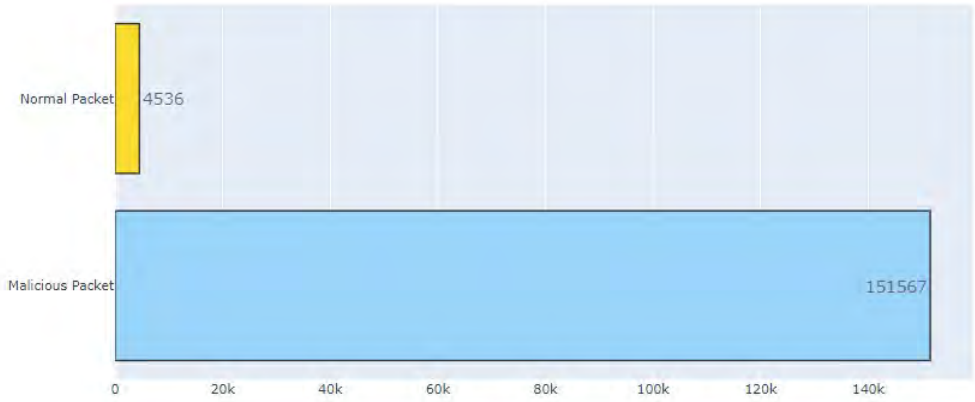


Figure 3.4: Representation of benign and Malicious data in capture-3-1

Table 3.1: CTU-IoT-Malware-Capture-3-1

Label	**Flows**
Attack	5,962
Benign	4,536
C&C	8
PartOfAHorizontalPortScan	145,597

These traffics are generated by Muhstick ransomware. Muhstick ransomware is a type of ransomware which uses a brute force method to crack devices' password and encrypt all the data. There were some empty fields in the dataset. We had to replace them with zero or NaN. We have also created a new column for the label and replaced 'malicious' label with 1 and 'benign' label with 0. Then we have used label encoders to classify multiclass attributes like protocol, service etc. so that the data could be fed to the algorithms.

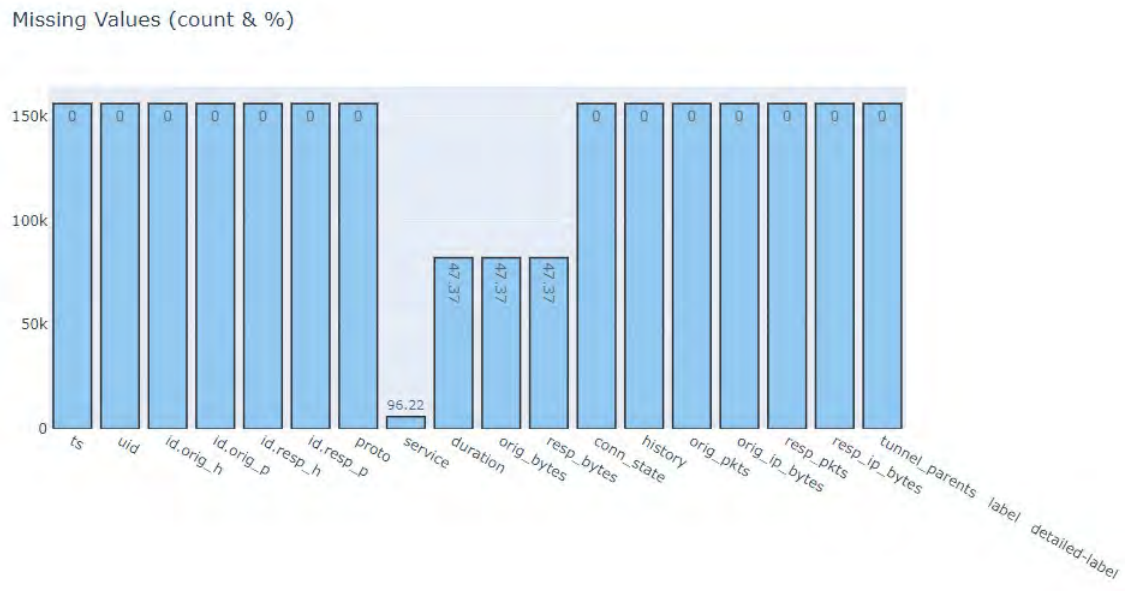


Figure 3.5: Bar chart of missing values of different attributes

As we can see from figure 3.4 that this data is heavily unbalanced, we decided to also experiment on capturing CTU-IoT-Malware-Capture-1-1 dataset. We see that (Table 3.2) this dataset is almost evenly balanced.

Table 3.2: CTU-IoT-Malware-Capture-1-1

Label	**Flows**
Benign	469,275
C&C	8
PartOfAHorizontalPortScan	539,465

Out of total 1,008,748 traffics, 469,275 of them are benign, 539,465 of them are PartOfHorizontalPortScan and 8 of them are C&C (Table 3.2).

Figure 3.6: Representation of benign and malicious data in capture-3-1

3.3 Model Description

3.3.1 DQN Model

A most important task for any models is to properly process the dataset to create mini-batches. Mini-batches are the set of any sample data that will be used by the model. The dataset that is used for training have N samples features of the network and the related intrusion which are labeled with binary values. To take these elements into DRL concept the features of the network are considered as the states and the values of the label as the actions. The DQN mini-batch consists of $n + 1$ randomly selected samples from the dataset. It is generated by random permutation of the dataset before the process is started and after that it chooses $n+1$ number of consecutive data samples which start from an index randomly. We have initialized the episode value by 1,000,000. The epsilon's initial value is 1 and the epsilon decay value is 0.9999. Our mini-batch size was initialized as 1024. The total number of mini-batch is obtained dividing the episode size by the mini-batch size. Initially the max reward was 1. In the model we have used a dataset of one million sizes which was labeled as malicious and benign.

Q function of a model will give the maximum reward under a certain state and an

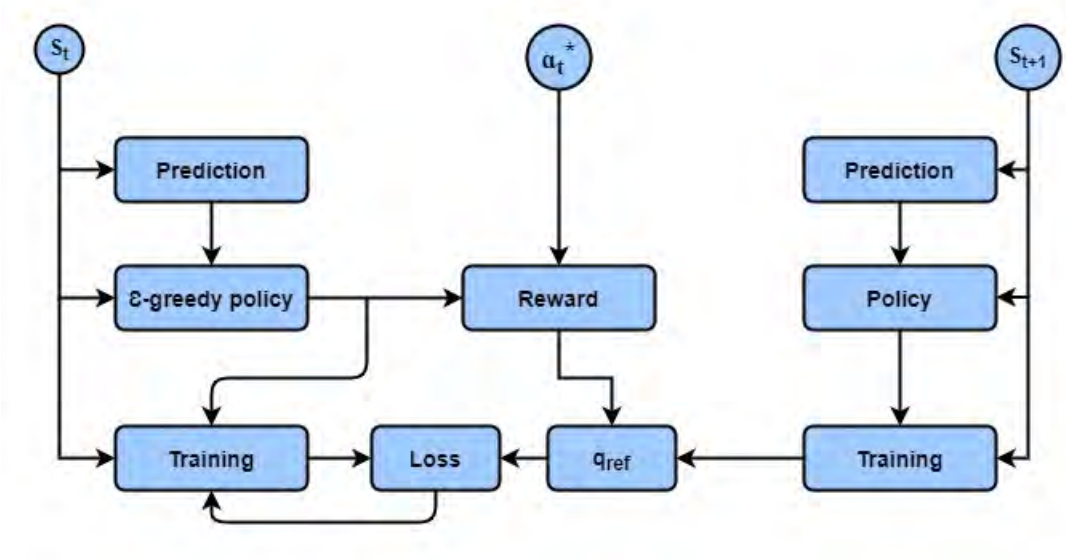


Figure 3.7: DQN Model Flowchart

action. DQN model's main aim is to approximate this Q function value. Obtaining

the Q function we will be able to obtain the value of policy function that selects the step to take for every state. The policy function is obtained from Q function. The formula is:

$$Policy(s) = arg_{\alpha}max(Q(s, \alpha)) \tag{3.1}$$

A collective sample build by a present state value (s_t), a label of ground truth of present state (α_t^*) and the next upcoming state (s_{t+1}) which is the segment of a mini-batch. This mini-batch is also a segment of N samples. Every time there is an iteration by the algorithm on the training set, all the samples from the mini-batch will be processed. Every iteration will generate a mini-batch of new data. A 3 layered neural network is used for the total layers as function approximator for the activation with ReLU activation which ensures that Q-value will be positive. Training iteration is completed with a loss of Mean Square Error linking Q-value approximated by the present state ($\hat{q}_t$) with a referred value: q_{ref}. Referred value is obtained by:

$$q_{ref} = r_t + \lambda * \hat{q}_{t+1} \tag{3.2}$$

Here, r_t is the current reward, q_{t+1} is the Q-value for the next upcoming state and λ is a factor of discount.

For a correct prediction the reward will be 1 and for an incorrect prediction the reward will be 0. When the label of ground truth value α_t^* and the predicted value $\hat{\alpha}_t$ is equal to the reward it is 1 and when not it is 0. Iterating the Q function along with the present state and every label value we obtained the predicted value. After that we select the value of the action that initiated the Q-value into maximum which we got through the iteration $arg_{\alpha}max\ (Q(s_t, \alpha))$ and it is later used in an ϵ-greedy based algorithm. Similarly we get the prediction state for the next state but here we ignored the ϵ-greedy selection. To get the best performance we applied a small value for the discount factor which is 0.5. It helps to learn the current reward rather than the succession of future rewards.

To make our algorithm faster we implemented the Q-function that contains two vectors as input and one output which is a scalar vector. The network is iterated

with every value of action for getting the expected value which helps to get the maximum Q-function. When the model's training is completed, the neural network is used to predict by implementing Q function. Q value is maximized by the predicted action.

3.3.2 DDQN Model

The algorithm that is implemented in this model is Double Deep Q-Learning (DDQN). Both DQN and DDQN are almost alike. However, there is a single difference: in DDQN, two neural networks are executed. One of them executes a target Q-function, whereas another one executes a present Q-function. The target Q-function is the duplicate of the present Q-function. The value of Q for the next upcoming state ($\hat{q}_{t+1}$) can be calculated by using the target Q-function. The moving effect of any target can be avoided while doing gradient drop over $(\hat{q}_t - q_{ref})^2$ if the target Q-function is used. The recursive dependence of q_{ref} on the training network can be avoided.

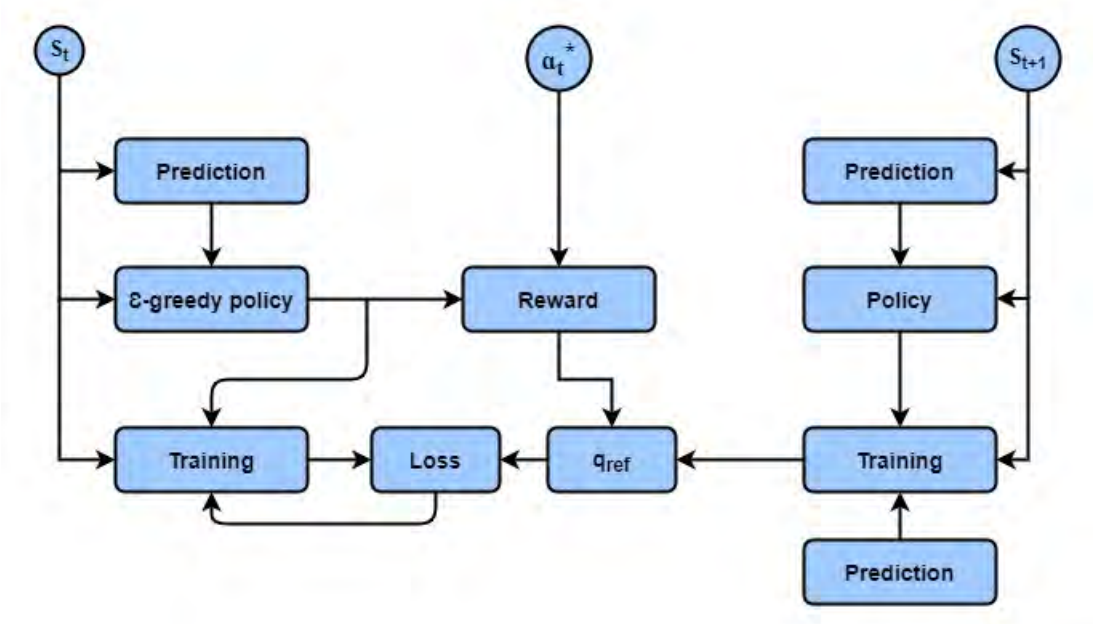


Figure 3.8: DDQN Model Flowchart

Chapter 4

Experimentation

4.1 Implementation of Different ML Algorithms

We have implemented a model to use different Machine Learning (ML) algorithms for the comparison with our main model. We have used Naive Bayes classifier, Decision Tree classifier, KNeighbourClassifier and Logistic Regression on our dataset. We split our dataset into test and train data with a ratio of 1:5. Among the four algorithms, the best result was given by the Decision tree classifier with 100% accuracy and the worst result was given by the Naive Bayes Classifier with 63% accuracy on the model.

Table 4.1: ML Model Test Results

Model	Model Accuracy	Classification report				
			precision	recall	f1-score	support
Naive Bayes Classification	0.6314746	0	0.56	0.95	0.71	112685
		1	0.90	0.35	0.50	129415
		accuracy			0.63	242100
		macro avg	0.73	0.65	0.61	242100
		weighted avg	0.74	0.63	0.60	242100
Decision Tree Classifier	1.0	0	1.00	1.00	1.00	112685
		1	1.00	1.00	1.00	129415
		accuracy			1.00	242100
		macro avg	1.00	1.00	1.00	242100
		weighted avg	1.00	1.00	1.00	242100
KNeighbor Classifier	0.9989095	0	1.00	1.00	1.00	112685
		1	1.00	1.00	1.00	129415
		accuracy			1.00	242100
		macro avg	1.00	1.00	1.00	242100
		weighted avg	1.00	1.00	1.00	242100
Logistic Regression Classifier	0.9093060	0	0.93	0.87	0.90	112685
		1	0.89	0.95	0.92	129415
		accuracy			0.91	242100
		macro avg	0.91	0.91	0.91	242100
		weighted avg	0.91	0.91	0.91	242100

4.2 Implementation of MLP

After preparing the dataset we have split the dataset into train and test data where 20% data are for testing purpose and 80% data are for training purpose. We further divided train data into X_train and y_train, test data into X_test and y_test. X_train and X_test contain the attributes. On the other hand y_train and y_test contain the data of column 'label'.

Then, we have imported the MLP classifier from scikit-Learn library. We have created three hidden layers with the first one having 1000 neurons. Both the second

and third layers have 300 neurons each. We have set 'adam' as the solver parameter which is a stochastic gradient-based optimizer. We have set shuffle to 'False' so that it won't shuffle samples in iterations. The tol (tolerance) parameter is set to a very small number of 0.0001, so that even if the loss score is very low, training will not be stopped. After that we fitted the train data to MLP and got an accuracy of 99.8% with the test data.

As we were not happy with this unbalanced dataset, we have applied this process on CTU-IoT-Malware-Capture-1-1 dataset. After doing all the preprocessing of the data, training (figure 3.1) and testing like we did on the previous dataset, we got an accuracy of 89.9%, which is fairly good. Though this time the accuracy is lower than the previous dataset, we have to keep in mind that the previous dataset was heavily unbalanced. Upon further analysis and applying this algorithm on other balanced datasets we will be able to get a clear idea about the accuracy of this algorithm.

4.3 Implementation of DQN

Using the described process we have predicted our result. In our 1M data we used the DQN algorithm three times by changing the value of the discount factor. When the discount factor is 0.1 the accuracy of the model is 99.02%. Changing the value into 0.5 we got an accuracy of 99.21%. Finally with the value of 0.99 the accuracy stands 99.12%. We got the best accuracy when the value of the discount factor is 0.5 which is 99.21%.

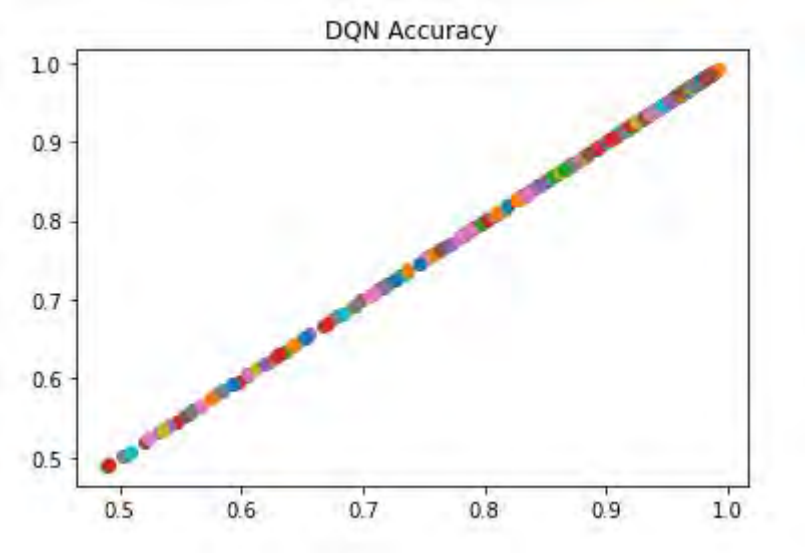


Figure 4.1: Accuracy Scatter plot of DQN model

4.4 Implementation of DDQN

We predicted the accuracy of our model using DDQN in a similar way changing the discount value three times to get the best result. The model gave a test accuracy of 99.32%, 99.12% and 99.31% when the values of the discount factor are 0.1, 0.5 and 0.9 respectively. Among them for the 0.1 value of discount factor we obtained the best accuracy.

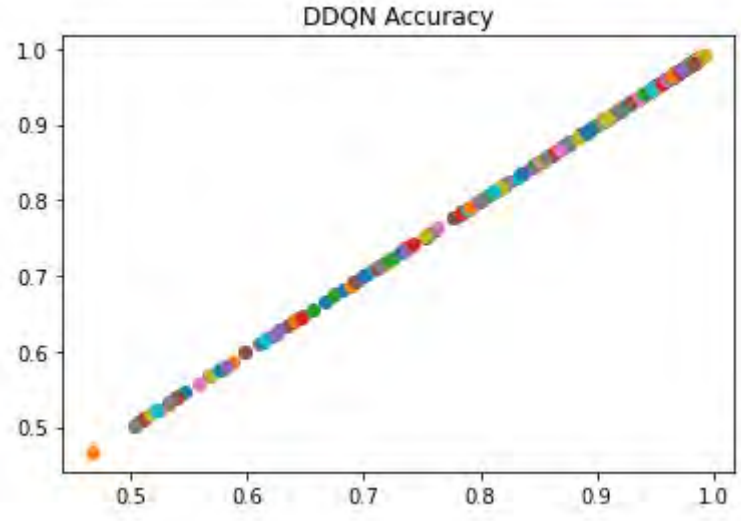


Figure 4.2: Accuracy Scatter plot of DDQN model

Chapter 5

Result Analysis

Table 5.1: Different model result comparison

Model	Model Accuracy	Precision	Recall	f1-score
Naive Bayes Classifier	0.6314746	0.73	0.65	0.61
Decision Tree Classifier	1.00	1.00	1.00	1.00
KNeighbor Classifier	0.9989095	1.00	1.00	1.00
Logistic Regression Classifier	0.9093060	0.91	0.91	0.91
MLP	0.899	0.892	0.884	0.89
DQN	0.9921	0.983	0.991	0.987
DDQN	0.9932	0.994	0.9919	0.993

We have compared the results among the models based on model accuracy, precision, recall, F1-score and confusion matrix. In terms of model accuracy, precision, recall and F1-score Decision Tree Classifier stand out best amongst all. By analyzing its Confusion Matrix, we can see that False positive and False negative values are 0 which means Decision Tree Classifier works best for predicting data. Interestingly we can see that for model accuracy KNeighbour Classifier, DQN, DDQN and Logistic regression gives maximum accuracy with 0.9989095, 0.9921, 0.9931, 0.9093060 respectively except for Decision Tree Classifier. In confusion Matrix KNeighbour Classifier gives the lowest false positive value except for Decision Tree Classifier which is 264 and also the false negative value is 0 for this model. Among all the models that we have worked on Naive Bayes Classifier showed the lowest model

accuracy with 0.6314746.

To compare the results with another paper, in [20] they have got a maximum accuracy of 96.20% using Decision Tree, 90.55% using Naive Bayes, 95.82% using Random Forest, 94.70% using MLP, 95.41% using DQN and 95.70% using DDQN. We got better results in DQN and DDQN by implementing our own model.

After applying multiple algorithms on our dataset, we get different levels of accuracy from those algorithms. For example, Decision Tree Classifier, K-Neighbour Classifier (KNN), Deep Q-Network (DQN), and Double Deep Q-Network (DDQN) achieve high accuracy. On the other hand, Multi-Layer Perceptron (MLP), Logistic Regression Classifier give average accuracy and the Naive Bayes Classifier gives us too low accuracy. As the decision tree algorithm creates a tree representation of data, it achieves high accuracy for binary classifications. KNN is a fast classifier as it does not need to train any data like typical deep learning algorithms. As KNN can not differentiate the importance of labels and our data labels are related to each other, it can create better classifiers than other algorithms like SVM. For our DRL algorithms, DQN and DDQN create a policy where their purpose is to get maximum reward. These algorithms calculate Q-values to predict correct actions for the agent to execute in the environment. As DQN tends to overestimate Q-values, DDQN solves this problem. This is why DDQN has a slightly better result than the DQN algorithm. The logistic regression classifier provides acceptable accuracy for a simple dataset like ours. As it assumes a linear relationship between the dependent variable and the independent variables, it is tough to obtain more complex functions applying logistic regression classifiers. MLP, on the other hand, can solve nonlinear problems unlike the logistic regression classifier but has confusion about global minima. It can not differentiate between global and local minima. Additionally, the user needs to manually set the number of hidden layers which may need to underfitting or overfitting of the model. Lastly, the Naive Bayes algorithm delivers such a bad performance because of its estimation. It takes all labels independent of each other and tries to find the action with the highest probability. This may be correct most of the time but it creates an inaccurate assumption as the features are

not independent of each other. Thus it delivers poor performance.

Chapter 6

Conclusion and Future Work

In the present time application of different Machine Learning algorithms in different sectors has become very common. There is a lot of research based on ML and DRL. In the sector of IoT application of DRL is very few as it is still developing. We have successfully applied the DRL algorithm and predicted vulnerabilities of IoT using labeled and balanced data. The previous works contained smaller datasets compared to the dataset we have used. However, we have only predicted the vulnerabilities, the prevention of these vulnerabilities are yet to be done. We used 7 algorithms for our work and got good accuracy for most of them. We have implemented a model to use different Machine Learning (ML) algorithms for the comparison with our main model. Decision Tree, KNN, DQN and DDQN give us the best results. Decision tree classifier gives 100% accuracy but it only deals with small dataset and can't give good accuracy for large data. KNN gives 99.89% accuracy. It's an instance-based learning; K-NN is a memory-based approach. The classifier immediately adapts as we collect new training data. It allows the algorithm to respond quickly to changes in the input during real-time use but is much time consuming. Then we got the best accuracy for DQN when the value of the discount factor is 0.5 which is 99.21%. DQN is the better version of Q-learning, in DQN we use neural networks to make predictions about q value. So it's much faster and less space consuming. DDQN model gave a test accuracy of 99.32% for the 0.1 value of discount factor and is better than DQN. The worst result was given by the Naive Bayes Classifier with 63% accuracy on the model. The class label associated with the largest probability is produced as a classification by selecting it. Not always the correct class label is

associated with the largest probability, so the accuracy level is low.

Most detection work in IoT sectors has been done using ML, DL, DRL. These experiments are done based on an average dataset. The dataset that we had contained almost 760 millions data. But we could work using only one million data due to our limitations. In future we plan to implement our model using the whole dataset and classifying the different attacks on IoT devices. Moreover, prevention of these vulnerabilities using DRL will help the people to use the IoT devices with more trust.

Bibliography

[1] Y. Li, “DEEP REINFORCEMENT LEARNING: AN OVERVIEW”, arXiv:1701.07274, 26 Nov 2018.

[2] S. S. Mousavi, M. Schukat, E. Howley, ”Deep Reinforcement Learning: An Overview”, Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016 Lecture Notes in Networks and Systems, Vol. 16, 426-440, 2018.

[3] Rouse, Margaret (2019). ”Internet of Things (IoT)”. IoT Agenda. Retrieved 14 August 2019.

[4] J. Granjal, E. Monteiro, and J. Sa Silva, “Security for the internet of things: a survey of existing protocols and open research issues,” IEEE Communications Surveys Tutorials, vol. 17, no.3, pp. 1294–1312, 2015.

[5] T. Alladi, V. Chamola, B. Sikdar, k.k.R. Choo, “Consumer IoT: Security Vulnerability Case Studies and Solutions”, IEEE Consumer Electronics Magazine, vol. 9, issue. 2, pp. 17-25, 2020.

[6] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, “Security analysis on consumer and industrial iot devices” 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, pp. 519–524, 2016.

[7] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, “Evaluating critical security issues of the iot world: Present and future challenges”, IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2483–2495, 2017.

[8] M. Moh, R. Raju, ”Machine Learning Techniques for Security of Internet of Things (IoT) and Fog Computing Systems Learning”, in 2018 International Con-

ference on High Performance Computing Simulation (HPCS), Orleans, 2018, pp. 709-715.

[9] L Xiao, X. Wan, X. Lu, Y. Zhang, D. Wu, "IoT Security Techniques Based on Machine Learning", in IEEE Signal Processing Magazine, vol. 35, issue. 5, pp. 41-49, Sept. 2018.

[10] F. Hussain, R. Hussain, S. A. Hassan, E. Hossain, "Machine Learning in IoT Security: Current Solutions and Future Challenges", IEEE Communications Surveys & Tutorials, vol. 3, issue. 3, pp. 1686-1721, Mar. 2019.

[11] J. Cañedo and A. Skjellum, "Using machine learning to secure IoT systems," presented at 2016 14th Annual Conference on Privacy, Security and Trust (PST), Auckland, 12-14 Dec. 2016.

[12] M. Roopak, G. Yun Tian and J. Chambers, "Deep Learning Models for Cyber Security in IoT Networks," presented at 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7-9 Jan. 2019.

[13] L. Xiao, C. Xie, T. Chen, and H. Dai, "A mobile offloading game against smart attacks", IEEE Access, vol. 4, pp. 2281–2291, 2016.

[14] Y. Gwon, S. Dastangoo, C. Fossa, and H. Kung, "Competing mobile network game: Embracing anti-jamming and jamming strategies with reinforcement learning," presented at. IEEE Conf. Commun. and Network Security (CNS), pp. 28–36, National Harbor, MD, USA, Oct. 14-16, 2013.

[15] M. A. Aref, S. K. Jayaweera, and S. Machuzak, "Multi-agent reinforcement learning based cognitive anti jamming," presented at 2017 IEEE Wireless Commun. and Networking Conf (WCNC), San Francisco,CA,USA, Mar. 19-22, 2017.

[16] F. Ullah, H. Naeem, S. Jabbar, S. Khalid, M.A. Latif, F. Al-turjman, L. Mostarda,, "Cyber Security Threats Detection in Internet of Things Using Deep Learning Approach," IEEE Access, vol. 7, pp. 124379-124389, August 2019.

[17] S.Y. Wani,"Internet of Things (IoT) Security and Vulnerability," 10.13140/RG.2.2.29633.40801.

[18] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum and N. Ghani, "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-scale IoT Exploitations", IEEE Communications Surveys Tutorials, vol. 21, no. 3, pp. 2702-2733, 2019.

[19] C. Xiao, X. Pan, W. He, J. Peng, M. Sun, J. Yi, M. Liu, B. Li, D. Song, Characterizing Attacks on Deep Reinforcement Learning, arXiv:1907.09470, July 2019.

[20] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised model", Expert system with application, vol. 14,p. 112963, march 2019.

[21] K. Sethi, R. Kumar, N. Prajapati, P. Bera, "Deep Reinforcement learning based intrusion and Detection system for Cloud Infrastructure",presented at 2020 International Conference on COMmunication Systems NETworks(COMSNETS),Bengaluru, India, Jan. 7-11, 2020.

[22] Hasselt, H., 2020. "Double Q-Learning", Papers.nips.cc, Part of: Advances of Neural Information Processing System 23 (NIPS 2010), 2010.

[23] Stratosphere Laboratory. "A labeled dataset with malicious and benign IoT network traffic", January 22th. Agustin Parmisano, Sebastian Garcia, Maria Jose Erquiaga.
https://www.stratosphereips.org/datasets-iot23 [Accessed 5 October 2020].