2021

Detecting zero-day attacks using Recurrent Neural Network.

Ndungu, George Muchiri
Faculty of Information Technology
Strathmore University

Detecting Zero-Day Attacks using Recurrent Neural Network

George Muchiri Ndungu

Master of Science in Information Systems Security

2021

Detecting Zero-Day Attacks using Recurrent Neural Network

George Muchiri Ndungu

A Dissertation submitted in partial fulfilment of the requirements of the Degree of Master of Science in Information Systems Security at Strathmore University

Faculty of Information Technology
Strathmore University
Nairobi, Kenya

October 2021

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, this proposal contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Student Name: George Muchiri Ndungu

Sign: 

Date : 3rd September 2021

Approval

This dissertation of George Muchiri Ndungu was reviewed and approved by the following;

Dr. Vincent Omwenga
Research Director,School of Computing and Engineering Sciences
Strathmore University

Dr. Julius Butime
Dean,School of Computing and Engineering Sciences
Strathmore University

Dr. Bernard Shibwabo
Dean,School of Graduate Studies
Strathmore University

Abstract

The development of Information and Communications Technology (ICT) and an increase in the use of mobile technology has enabled organisations to implement and adopt the use of information and management systems to conduct their day to day activities. However, as cyber-attacks against organisations are becoming more frequent and more sophisticated there is a need for advanced measures to help prevent against the known cybersecurity attacks and zero day attacks. In view of the above shortcoming, this study developed an anomaly-based cybersecurity threats detection model using the Recurrent Neural Network (RNN) technique that can be used to detect zero-day vulnerabilities. This approach functions with the assumption that a cybersecurity attack is different from a normal system activity of a legitimate user and can be detected by a system that identifies the differences. The RNN algorithm has a strong modelling ability for anomaly detection, and high accuracy in both binary and multiclass classification. Compared to traditional classification methods its performance includes a higher detection accuracy rate with a low false-positive rate. This research adopts RAD methodology, which heavily emphasizes rapid prototyping and iterative delivery, to develop the RNN system for anomaly detection. This research aimed to develop an RNN model which will be used to detect zero-day vulnerabilities. The predictive model had an accuracy of 93% which was achieved through tests using model demo data. The main objective of the research was met and it proved that the Neural Network Algorithm can be used to detect zero-day attacks in a network.

KEYWORDS

Anomaly detection, Cybersecurity Threats, Zero-Day Attacks, Recurrent Neural Network

Table of Contents

List of Figures

List of Abbreviations

AC:	Accuracy
API :	Application Programming Interface
BN:	Bayesian Network
BNN:	Bidirectional Neural Network
BRNN:	Bidirectional Recurrent Neural Network
DT:	Decision Tree
FN:	False Negative
FN:	False Negative
FP:	False Positive
GRU:	Gated Recurrent Network
ICT:	Information and Communication Technology
ISO/IEC:	International Organisation for Standardization/International Electrotechnical Commission
IT:	Information Technology
KNN:	K- Nearest Neighbour
LSTM:	Long-short Term Memory
NIST:	National Institute of Standards and Security
NN:	Neural Network
RAD:	Rapid Application Development
RNN:	Recurrent Neural Network
SVM:	Support Vector Machines
TN:	True Negative

Acknowledgement

I am grateful for the support received from my supervisor Dr. Vincent Omwenga through his guidance on doing this research. I would like to also acknowledge my colleagues and management from @iLabAfrica for their encouragement and guidance through the research. I also acknowledge my family members for making it easy for me to work on the research by supporting in their special ways.

Chapter 1: Introduction

1.1 Background Study

A zero-day cybersecurity attack is an attack that exploits an undiscovered vulnerability in a network (Frankenfield, 2020). The vulnerability is used to create an exploit and attack a target. The vulnerability continues to be a threat until a patch to fix it is released. Emerging trends and development in ICT have led to the advancement of most institutions adopting the use of technology to enable their clients to have access to their services at any time regardless of location. Although the advancement of technology has offered convenience, it has also opened up cybersecurity challenges that need to be addressed to ensure that is maintained to protect against threats in an organisation.

Cybersecurity refers to the activities and precautions which are designed to offer protection to data availability and integrity across networks (Sun, 2016). Any act that could potentially cause harm and disrupt the normal functioning of a computer system or network is a cybersecurity threat. The function of information security is to protect digital data which is being transferred from one network to the next from disclosure, alteration and destruction, manipulation, and unauthorised access. Cyber-attacks are the intentional interference of computer systems, businesses that are dependent on technology and networks. Hundreds of websites are on the verge of being attacked each day because of the high levels of vulnerability of networks, files, and servers. During cyber-attacks attackers use malicious lines of code to alter the logic of computers systems and code.

According to Kumar (2017), gaining access to databases and computer systems can result in destructive consequences and cyber-crimes such as identity theft and information theft which can harm important data. Cybersecurity threats are classified into three categories according to the intent that an attacker has. An attack can be done for financial gain where the attacker intends to conduct electronic fraud and theft, another category is an attacker whose motivation is to disrupt by having unauthorised access to a system to alter the system resources or affect the operation and the other category is for reconnaissance where an attacker attempts to learn or make use of information from a system but does not affect the system resources but may sell the information to a third party.

A report on the level of Information Technology (IT) security preparedness of businesses in Kenya by the Communications Authority of Kenya (2018), revealed cybersecurity protection challenges of employees' data from cybercriminals. It also mentioned that financial institutions have lost more than Ksh.21 billion due to attacks on systems in 2017 that range from denial of service attacks, brute force attacks, and botnets that led to illegal access to the systems. In the aim to ensure confidentiality, integrity, and availability there is a need to have system frameworks that can be used to monitor cybersecurity systems and prevent cybersecurity attacks.

In detecting cybersecurity malfunctions and destructive consequences, it is important to have detailed information about what kind of cyber-attacks one will be coming across and their respective access and realization points. This can be achieved using a cybersecurity threats monitoring platform. Cybersecurity threats monitoring can be described as the process of continually monitoring a system to prevent any vulnerabilities that may give access to attackers or create room for them to perform malicious activities (Buckner, 2018).

Traditional systems of detecting cybersecurity threats in corporations are becoming obsolete due to the techniques they have used in the identification and prevention of cybersecurity threats. Most platforms use the signature detection method, which is a traditional security monitoring solution, that responds to threats reactively and provides monitoring of only a single vector (Trend Micro, 2018). Signature detection techniques can be described as a process where a unique identifier detects a threat first then use it to identify other future threats of the same kind (Bricata, 2019). This is inadequate since the growing threat attacks pose a challenge to firms and also to monitoring systems because some are only designed to protect against the known attacks. An attacker compromises such systems through new attacks or modifying already existing attacks. Moreover, the developed signatures need to be managed, distributed, and kept up-to-date by security administrators to ensure that they have the required capability to detect attacks early.

Another challenge is the time gap between when an attacker gets unauthorised access to a system and when the breach is detected (Vectra, 2018). An attacker might cause a lot of damage before being identified, which may result in loss of finances or data in a company. A zero-day attack poses a serious threat to Internet security as it exploits zero-day vulnerabilities in computer systems (Singh, 2015). This is because attackers take advantage of the unknown nature of zero-day exploits and use them to create cybersecurity attacks.

However, these factors put a significant constraint on the effectiveness of using this approach. Thus there is a need for enhanced cybersecurity threat detection capabilities that allow early detection of known and zero-day vulnerabilities to provide insights on the best way to prevent it before the threats compromise a network.

An anomaly-based approach using neural networks can be used for early threat detection as it ensures high accuracy and detection rate with low computational overhead (Subba, Biswas, & Karmakar, 2016). A recurrent neural network is a variation of an artificial neural network that continuously uses the output of its previous cell as the input of the current cell along with the input that was applied to this cell. Due to this feature, RNNs can store processed information from the previous input in the hidden state as these can be passed along through the previous cell's output as illustrated in figure 1.1 below. A recurrent neural network can be used to detect zero-day attacks as it can process time-series data by detecting anomalous activity in a network. Some of the exiting techniques that are being used to detect zero-day attacks include the use of a strong web application firewall for monitoring web traffic and filtering any suspicious traffic, use of vulnerability scanning tools that help scan code for any vulnerabilities that might be present

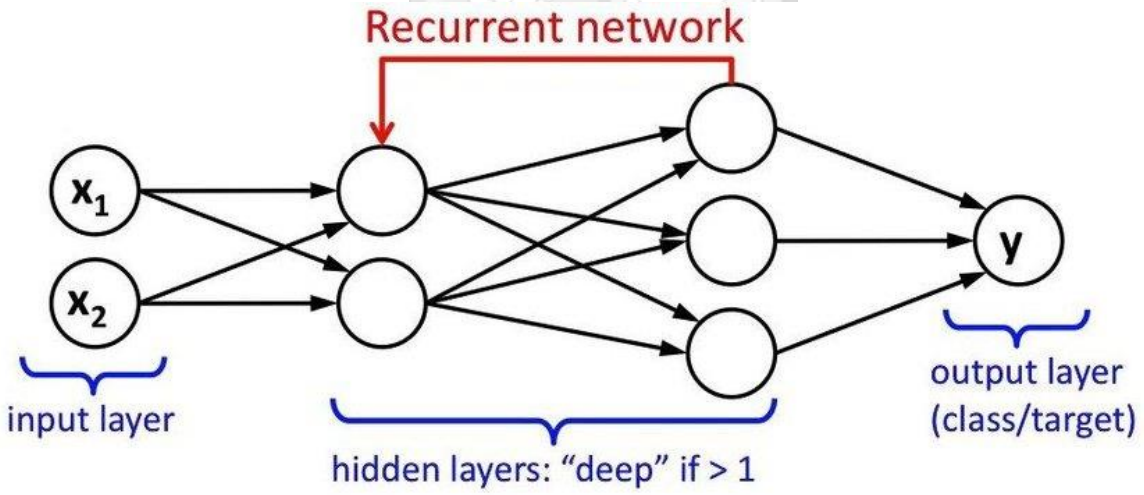


Figure 1.1 RNN Illustration example *(Walters, 2019)*

1.2 Problem Statement

The rapid evolution of network-based cyber-attacks has rendered traditional intrusion detection techniques insufficient for detecting zero-day attacks which may exploit security vulnerabilities that may be present in a network. According to Ayeri et al. (2018), existing techniques in intrusion detection have not been sufficient to protect systems from cybersecurity vulnerabilities, this issue is progressive with the evolution and new development in systems, cybersecurity vulnerabilities, and technology. To prevent zero-day attacks from exploiting such security weakness in a network, there is a need for more proactive techniques to detect known and unknown vulnerabilities.

1.3 General Objective

This project aims to develop an anomaly-based cybersecurity threats detection model using the Recurrent Neural Network technique that is used to detect zero-day vulnerabilities.

1.3.1 Specific Objectives

i. To review cybersecurity threat detection approaches and techniques.
ii. To analyse methods and tools used for detecting zero-day cybersecurity attacks.
iii. To design and develop a RNN model to detect zero-day cybersecurity attacks.
iv. To test the effectiveness of the RNN model to detect zero-day cybersecurity attacks.

1.4 Research Questions

i. What are the threat detection techniques that are being implemented to detect cybersecurity threats?
ii. What are the methods used to detect zero-day vulnerabilities?
iii. Which tools can be used to develop a model to detect zero-day cyber threats?
iv. What is the level of accuracy of the RNN tool in detecting cybersecurity threats

1.5 Justification

Signature-based intrusion detection methods cannot detect zero-day attacks since no signatures exist for such attacks before the attack. Thus there is a need for advanced techniques that can be used to detect a zero-day attack. An anomaly-based approach using neural networks can be used

for early threat detection as it ensures high accuracy and detection rate with low computational overhead. A recurrent neural network can be used to detect zero-day attacks as it can process time-series data by detecting anomalous activity in a network.

1.6 Scope and Limitation

This model is designed to detect zero-day attacks by identifying anomalies in a network that may infer a cybersecurity threat. The model has a high accuracy rate to ensure that it will have reduced overhead while at the same time maintain a high-performance level compared to using other methods. The major Limitation of using RNNs is the difficulty found in training the model to have high a high accuracy rate.

Chapter 2: Literature Review

2.1 Introduction

This research focuses on developing an anomaly-based cyber security threat detection model using the Recurrent Neural Network (RNN) technique that can be used to detect zero-day vulnerabilities. This chapter gives an introduction to cyber-threats detection approaches and it also reviews the various anomaly detection techniques that can be used to detect cybersecurity vulnerabilities.

2.2 Cyber Security Threats Detection Approaches

To develop a cybersecurity threats monitoring platform, a technique for detecting cybersecurity threats needs to be implemented. Cyber threats detection is one of the ways to protect from cybersecurity threats. According to (Naila Belhadj Aissa, 2016) they define intrusion detection is a process of identifying an attack pattern on a network or system that tries to enter without authorization. There are various approaches to detect a cyber-attack; some common methods are discussed below.

2.2.1 Signature-based Approach

This is a method of detecting attacks by use of the signature of already known attacks to generate a detection model. Characteristics of a cyber-attack are analysed and the information gathered is used to create the attack signature. After a cybersecurity attack is identified the signature is added to the database as a recognised cyber-attack (Hao Sun, 2017). This approach depends on the predefined lists of attacks in the model and thus is limiting to the growing landscape of cybersecurity attacks, which makes this method to be less effective. Another limitation of the signature-based approach is that since the attack database is already predefined and known, it makes it easier for attackers to come with new modifications on existing attacks or new attacks that cannot be detected by the signature-based model (Soiri, 2018).

2.2.2 Anomaly Detection

Anomaly detection is the process of identifying unusual behaviour in a system or network (Avi,2021). This approach functions with an assumption that a cybersecurity attack is different from a normal system activity of a legitimate user and can be detected by a system that identifies the differences. Normal user behaviour must first be defined in what is acceptable so that

differences can be distinguished and easily identified (Raiyn, 2014). The anomaly detection approach has an assumption that a cybersecurity attack always has some deviations from normal activity. It can have two approaches dynamic and static anomaly detection.

Static anomaly detection is where there is an area of a system that is being monitored and does not change. The static part of the system under review should have a code that instructs how a given function of the system should be executed. For example, a code describing a login function using a registered username and password from the database. If there is an attempt to login into a system through other means it shows that there is an attempted intrusion method being introduced which is different from the usual normal method. A dynamic anomaly detection approach is used for audit records or traffic data that is being monitored. This approach only focuses on specific areas of interest (Noorossana, Hosseini & Hydrazide, 2018).

2.3 Machine Learning Approach to Anomaly Detection

Machine Learning is a technique that automatically builds a data model using algorithms that analyse and find accurate data with minimal human intervention (Education, 2021)

This approach usually works in three phases, the data model needs to be trained, validated and then its effectiveness is tested through algorithms (Rashmikant, 2018). The main objective of this approach is to classify and predict the availability or absence of any instance learned using the data model that has been trained. Machine learning is used in cybersecurity attack detection to improve the accuracy of detection and reduce false positives (Ayei E. Ibor1, 2018). Machine learning algorithms can be classified into two categories, deep learning, and shallow learning. The two categories can also be further classified into either supervised and unsupervised algorithms.

Supervised learning uses pattern recognition which uses labeled instances for training data with the desired output. This approach can be used to derive a predictive model at the training phase that can be used to classify new datasets. Unsupervised learning uses patterns in unlabeled datasets for training the model to make classification decisions. The use of unsupervised learning in cybersecurity detection provides a technique for categorising a new occurrence through defining an attack and data during the building phase of a model. This research focuses on building a model using an unsupervised learning approach to detect anomalies. Research by (Tandon, 2018) discussed an anomaly detection system that uses an unsupervised learning approach that can automatically tune and optimise the value of parameters to obtain better-categorised instances, that

can represent an attack string or normal traffic. This approach performs the classification of instances after the training phase that comprises clustering, modeling, and filtering. The goal of using machine learning techniques for anomaly detection is to develop a generalization capability from limited training data and to be able to correctly classify future data as normal or abnormal.

2.4 Anomaly Detection Techniques

An anomaly detection technique is a method that can be used in detecting anomalies. Several techniques can be used to detect anomalies but this research will focus mainly on supervised machine learning techniques. A supervised anomaly detection technique requires a labeled training set that contains both normal and anomalous samples to construct a predictive model. They provide a better cyber threat detection rate than unsupervised methods since they have access to more information. The most common supervised algorithms are, Supervised Neural Networks, Support Vector Machines (SVM), k-Nearest Neighbors (KNN), Bayesian Networks, and Decision trees (Ghorbani, Lu, & Tavallaee, 2010). This paper focuses on building a model using a recurrent neural network which is one of the supervised algorithms that can be used for anomaly detection.

2.4.1 K -Nearest Neighbor (KNN)

This is a technique for classifying samples as it calculates the approximate distances between various points on the input vectors, and then assigns the unlabeled point to the class of its K-nearest neighbours (N,2020). In the process of creating a k-NN classifier, (k) is an important parameter and various (k) values can cause various performances. If k is very huge, the neighbours, which will be used for prediction, will use a large classification time and affect the prediction accuracy of determining an anomaly score. The anomaly score of a data instance is defined as its distance to its kth nearest neighbour in a given data set. For a given data instance, the distance to its kth nearest neighbour equals the radius of a hyper-sphere, centered at the given data point, which includes k other instances. Therefore, the distance to the kth nearest neighbour for a given data sentence can be considered as an estimate of the inverse of the density of the instance in the data set.

KNN falls in the supervised learning family of algorithms. Given a labeled dataset consisting of training observations (x,y)(x,y) and would like to capture the relationship between x and the goal is to learn a function $h:X \rightarrow Y$ so that given an unseen observation xx, h(x)h(x) can confidently predict the corresponding output y (Zakka, 2016). In a classification setting, the K-nearest

neighbour algorithm essentially boils down to forming a majority vote between the K most similar instances to a given "unseen" observation. The similarity is defined according to a distance metric between two data points. K -Nearest Neighbor (kNN) can be used for anomaly detection by calculating the differences between data points that are from normal behaviour and anomalous behaviour. The main drawback of this technique is that it is computationally expensive to operate and also has large storage requirements. Figure 2.1 below illustrates how KNN operates.

To recognise different types of motion activities, The KNN classifier will be trained. The idea of the method is to find *k* training samples closest to the sample with an unknown label, and predict the label as a most frequent class among that k.

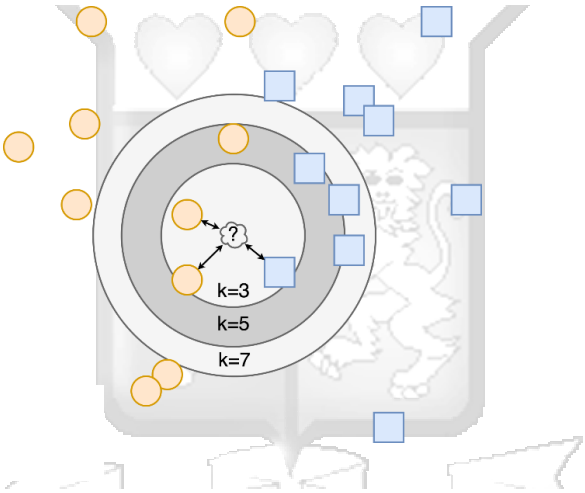


Figure 2.1 KNN Illustration example *(Sosnovshchenko, 2017)*

2.4.2 Support Vector Machines (SVM)

This algorithm can be used for anomaly detection to identify a cyber-attack since it is a classification problem, in which one classifies the normal pattern from the abnormal pattern of a system. It is based on recent advances in statistical learning theory and has been successfully applied in real-world problems such as text categorisation, image classification, and handwritten character recognition (Byun & Lee, 2002). SVM has an assumption that all samples in the training set are independently and identically distributed. However, it also has some limitations in using it for anomaly detection. This is that SVM is sensitive to outliers or noises, it is meant for a two-class problem, it has to be extended for a multiclass problem by choosing a suitable kernel function. The performance of this algorithm depends upon the kernel function (Hoak, 2010). However, SVM

has high detection precision to predict the known attacks and can also detect some unknown attacks.

The research explained that Support Vector Machine (SVM) is deemed as a machine learning technique that could complement the performance of an intrusion detection system, that provides a second line of detection to reduce the number of false alarms (Ghanem, Aparicio-Navarro, G, & Lambotharan). SVM training algorithm will make the decision surface deviate from the optimal position in the feature space. When mapped back to the input space, it results in a highly nonlinear decision boundary. SVM works by mapping data to a high-dimensional feature space so that data points can be categorised, even when the data are not linearly separable. A separator between the categories is found, then the data is transformed in such a way that the separator could be drawn as a hyperplane. Following this, characteristics of new data can be used to predict the group to which a new record should belong.

For example, consider the following figure 2.2 below, in which the data points fall into two different categories:

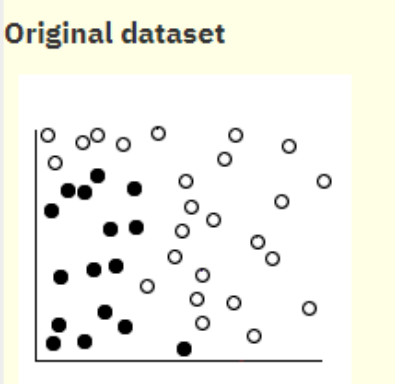


Figure 2.2 SVM Sample Data Set *(IBM, 2012)*

The two categories can be separated with a curve:

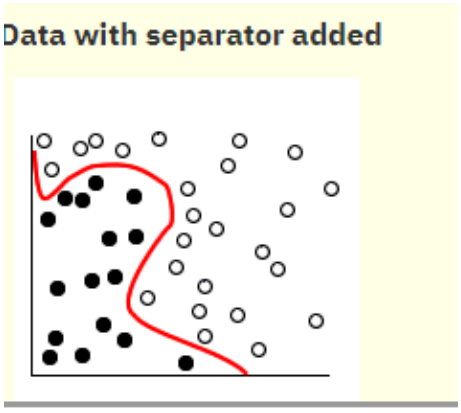


Figure 2.3 Data with Separator *(IBM, 2012)*

After the transformation, the boundary between the two categories can be defined by a hyperplane and you get an output of the transformed data.

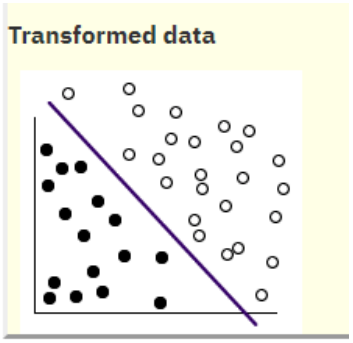


Figure 2.4 Transformed Data *(IBM, 2012)*

2.4.3 Bayesian Networks

Bayesian Network is an algorithm that converts relationships among variables of interest. This technique is generally used for intrusion detection in combination with statistical schemes. It has several advantages, including the capability of converting interdependencies between variables and of predicting events, as well as the ability to incorporate both prior knowledge and data. In research about Model-based Monitoring for Cyber Attack Detection researchers developed an anomaly-based intrusion detection system that employed a naive Bayesian network to perform intrusion detecting on traffic bursts (Valdes & Skinner, 2000). It was developed to improve the inference methods of detecting an attack that used signature methods and statistical anomaly methods. In 2003, two researchers, Kruegel and Mutz, proposed a multisensory fusion approach

using a Bayesian classifier for classification and suppression of false positives that the outputs of different intrusion detection systems sensors were aggregated to produce a single alarm. Their model also was able to solve the problem of distinguishing between anomalous behaviour caused by unusual but legitimate actions and activity that is the manifestation of an attack.

A Bayesian network works as a directed acyclic graph in which each edge corresponds to a conditional dependency, and each node corresponds to a unique random variable. Fig 2.5 below shows an example of how a Bayesian algorithm operates. In the example below, since Rain has an edge going into Wet Grass, it means that (WetGrass|Rain) will be a factor, whose probability values are specified next to the WetGrass node in a conditional probability table.

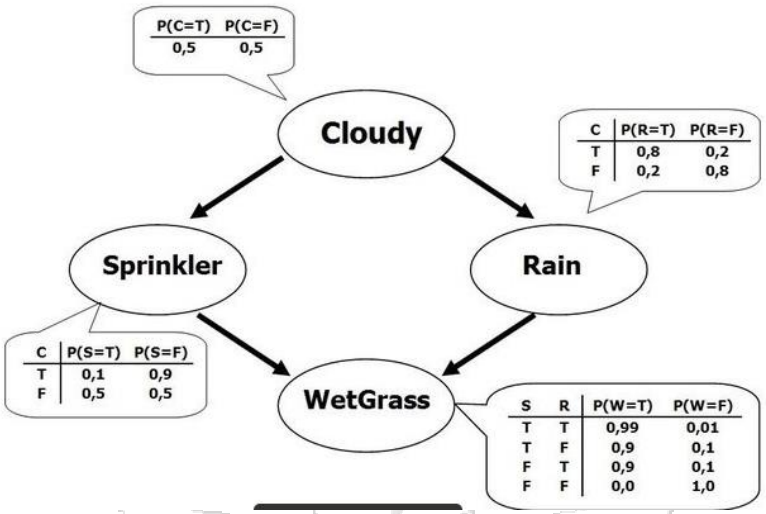


Figure 2.5 Bayesian Network Sample Illustration *(Soni, 2018)*

2.4.4 Decision Tree

A decision tree is a tree that has three main components: nodes, arcs and leaves. Each node is labeled with a feature attribute, which is most informative among the attributes not yet considered in the path from the root (Berrar, 2013). Each arc out of a node is labelled with a feature value for the node's feature, and each leaf is labeled with a category or class. A decision tree can then be used to classify a data point by starting at the root of the tree and moving through it until a leaf node is reached. A decision tree with real values at the leaves is called an uphill decision tree if the values on the leaves are non-decreasing in order from left to right. An Uphill decision tree is similarly a tree-structured solution in which a constant or a relatively simple regression model is fitted to the data in each partition. This algorithm considers the data collected from a system

whether it is normal or an anomaly. The decision tree (DT) is a very powerful and popular data mining algorithm for decision-making and classification problems. It can be represented as a tree-like 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 (Gupta, 2017).

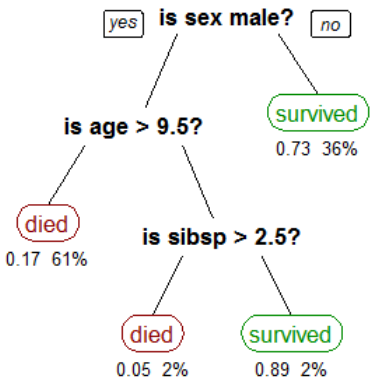


Figure 2.6 Decision Tree Example *(Gupta, 2017)*

In the illustration above the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that does not split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively. The main limitation of this technique is that it may create complex trees that do not classify a problem well and it also makes the learning of some concepts to be hard since the decision tree does not express them easily.

2.4.5 Neural Network

A neural network learns and predicts different user's behaviour in systems. A neural network model that is properly designed and implemented can address many problems encountered by rule-based approaches ("Neural networks," n.d.). The main advantage is their tolerance to imprecise data and uncertain information, and their ability to conclude solutions from data without having prior knowledge of the consistencies in the data. This in combination with their ability to generalize from learning data, has made them a proper approach to anomaly detection. To implement a neural network model, datasets representing attacks and normal user activity have to be introduced to the algorithm to adjust automatically during the training phase (Omar, Ngadi, & Jebur, 2013). This proposed research will use a recurrent neural network to detect anomalies since it overcomes most of the challenges that are faced by other techniques. The Neural Network contains at least three

layers of nodes. The input layer is followed by a hidden layer and an output layer. The input layer consists of the data given to generate the prediction. The layer is called hidden because it does not interact directly with the input or the output layer but works on data transformed by the previous layer. The hidden layer then links to the output layer that displays the final prediction produced by the algorithm. Figure 2.7 below shows an example of how a neural network operates.

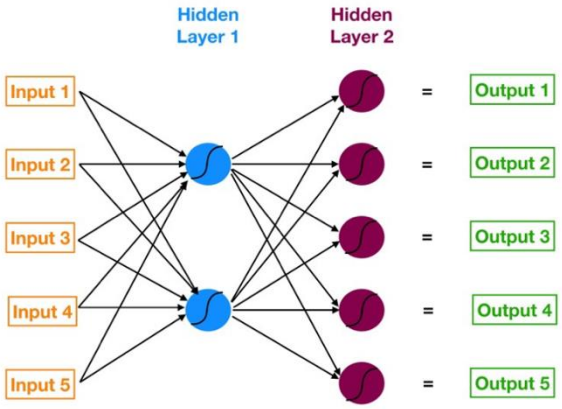


Figure 2.7 Neural Network Illustration *(Yiu, 2017)*

2.5 Zero-day Attacks

"Zero-day" is a broad term that describes recently discovered security vulnerabilities that hackers can use to attack systems. The term is popularized by the notion that developers have 'zero' days or time left to fix the security threat.

Terms such as exploits, vulnerabilities and attacks are used alongside zero-days, also spelt as 0-day, however, it is critical to understand the difference between the three. Zero-day vulnerabilities are those threats unknown to the vendor and thus can be used by hackers to hack a system and are likely to succeed because the vendor is not aware of the threat. Zero-day exploits refer to the methods that hackers use to hack a system once they have identified the zero-day vulnerabilities. Zero-day attacks refer to the use of zero-day exploits to hack a system or server.

Zero-day attacks occur when hackers and malicious system users use an exploit code to attack a system after identifying a vulnerability. By doing this, the victims of the exploit code might experience unusual system behaviour which causes a lot of damage including theft. When the attack is found, it still might be difficult for developers to patch it, due to a lot of research involved, since the threat is only known to the hacker.

Zero-day attacks can be difficult to identify since they might show in different areas such as weak passwords, missing authorization and encryption, broken algorithms and many bugs. Zero-day attacks can however be identified through constant checking of the system to identify suspicious scanning activity, suspicious traffic or unexpected traffic from certain users.

Some of the popular zero-day attacks include The zoom zero-day attack where a hacker could access the files of a user if the user was using an old version of windows. The hacker could take over all their files if the user was an administrator. Another example is the chrome zero-day attack in 2021, where a bug causes a series of updates on chrome. The issue was found to stem from a v8 javascript engine which is normally used in the web browser.

Some of the techniques used include checking the existing databases of known software malware against a system, deploying machine learning algorithms and systems, using third party systems and models.

While the zero-day attacks can be very frightening, there are ways it can be mitigated such as, keeping the software being used up to date, involving this party zero-day attack models that use machine learning to protect the system, using firewalls and using only essential applications.

2.6 Recurrent Neural Networks (RNN)

A recurrent neural network is a variation of an artificial neural network that continuously uses the output of its previous cell as the input of the current cell along with the input that was applied to this cell. Due to this feature, RNNs can store processed information from the previous input in the hidden state as these can be passed along through the previous cell's output. A Recurrent neural network has three-layer which is, an input layer, an output layer and a hidden layer. The RNN model essentially has a one-way flow of information from the input layer to the output layer. The Hidden layer can be regarded as the storage of the whole network, which remembers the end-to-end information that passes through the layers. Recurrent neural networks have a directional loop that is used to memorize the previous information passed and applies it to the next input. The previous output is also related to the current output of a sequence, and this also creates a connection between the nodes. Figure 2.8 below shows depicts an RNN architecture.

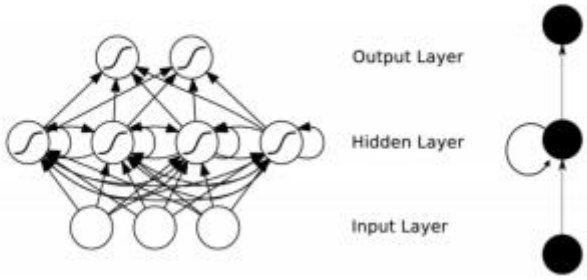


Figure 2.8 RNN Architecture *(Yin C., 2017)*

Figure 2.8 describes the steps the RNN model uses in detecting an anomaly. Data pre-processing is a process of cleaning the raw data where the data is converted to a clean data set. Numericalization is the process of converting the data into numerical values e.g. into binary form which is acceptable for a machine learning model. Normalisation changes the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. Forward propagation is done on a neural network since it allows for a one-way flow of information where the input data is fed in the forward direction throughout the model. Weight updates are calculated by the backpropagation algorithm to improve the network until it can perform the task it is meant to.

In comparing the performance of RNN techniques with other machine learning methods both in binary classification and multiclass classification. The experimental results illustrate that RNN is more suitable for anomaly detection. The performance of RNN is superior to the traditional classification method on the NSL-KDD dataset in both binary and multiclass classification, and it improves the accuracy of anomaly detection and thus reduces the number of false positives. In a research by (Quamar Niyaz, Sun, Javaid, & Alam, 2013), they used a deep learning approach based on a deep neural network for flow-based anomaly detection, and from the experiment, it proved that deep learning can be applied for anomaly detection in software.

Malhotra et al (2015), proposed a model that uses RNNs using the LSTM architecture which proved very useful in finding time series anomalies, which are anomalies over a time frame with multiple actions, instead of single-action anomalies. LSTM architecture excels at this due to its ability to learn which aspects of the previous input data to be remembered and which aspects to forget. They developed a stacked LSTM model, that was trained to recognise regular behaviour, and was tested on 4 different data sets. Through the experiment the model was able to find

collective anomalies where other types of anomaly detection would not find them, being able to link together multiple instances of slightly deviating behaviour into a definitive anomaly.

The RNN model has a strong modelling ability for anomaly detection, and high accuracy in both binary and multiclass classification. Compared with traditional classification methods, such as J48, naive Bayesian, and random forest, the performance obtains a higher accuracy rate and detection rate with a low false-positive rate, especially under the task of multiclass classification as described in analysing the given dataset (Chuanlong, Yuefei, Jinlong, & Xinzheng, 2017).

2.7 Performance Evaluation and Compliance Standards

In this model, the most important performance indicator (Accuracy, AC) of anomaly detection will be used to measure the performance of the RNN anomaly detection model. In addition to the accuracy, detection rate and false-positive rate will also be used. The True Positive (TP) is equivalent to those correctly rejected, and it will denote the number of anomaly records that will be identified as an anomaly. The False Positive (FP) will be the equivalent of incorrectly rejected, and it will denote the number of normal records that will be identified as an anomaly. The True Negative (TN) is equivalent to those correctly admitted, and it will denote the number of normal records that will be identified as normal. The False Negative (FN) is equivalent to those incorrectly admitted, and it will denote the number of anomaly records that will be identified as normal (Chuanlong, Yuefei, Jinlong, & Xinzheng, 2017). The motivation for the model is to obtain a higher accuracy and detection rate with a lower false-positive rate.

To calculate the percentage of the number of records classified correctly versus total records the following computation will be conducted; Accuracy: AC = TP + TN / TP + TN + FP + FN

To calculate the True positive rate that will show the percentage of the number of records identified correctly over the total number of records; True Positive Rate (TPR): TPR = TP / TP + FN

To calculate the False positive rate that will show the number of records rejected incorrectly.

False Positive Rate (FPR): FPR = FP/ FP + TN

The proposed platform will also have built according to the required industry standards to ensure that its functionalities operate as they are meant to. The following are some of the Compliance standards and guidelines that will be used in developing the platform;

2.7.1 NIST Cyber Security Framework

This is a set of best standards, practices and recommendations that can assist an organisation to improve its cybersecurity measures (Brook, 2018). NIST specifies controls and processes that are essential to cybersecurity in an organisation. Figure 2.9 below demonstrates the NIST Framework summary and breakdown.

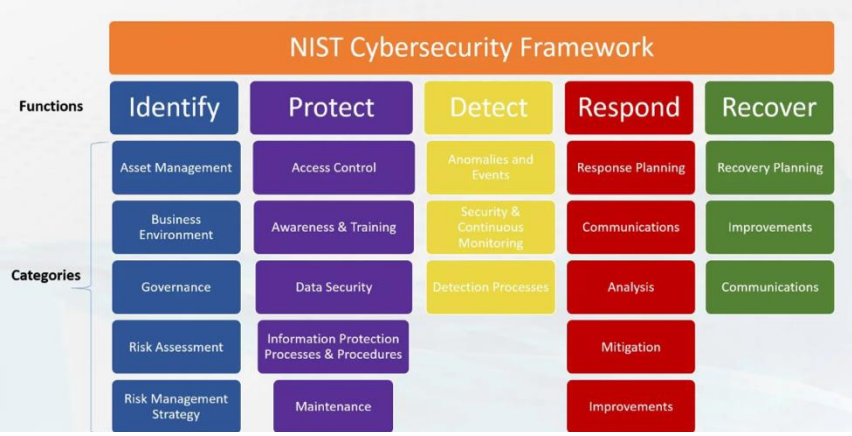


Figure 2.9 NIST Cyber Security Framework (Source: *(Cipher Security, 2018)*)

The NIST cybersecurity framework is organised into five core Functions also known as the Framework Core. Definitions for each Function are as follows:

- **Identify:** Develop the organisational understanding to manage cybersecurity risk to systems, assets, data, and capabilities.
- **Protect:** Develop and implement the appropriate safeguards to ensure the delivery of critical infrastructure services.
- **Detect:** Develop and implement the appropriate activities to identify the occurrence of a security event.
- **Respond:** Develop and implement the appropriate activities when facing a detected security event.
- **Recover:** Develop and implement the appropriate activities for resilience and to restore any capabilities or services that were impaired due to a security event.

NIST helps to prevent any financial loss, reputation damage or any other effect that may affect the normal functioning of an organisation. NIST identifies the critical activities that ensure the smooth

running of an organisation by prioritizing them to ensure cybersecurity and risk management (NIST, 2019). The figure below describes how the NIST framework operates.

2.7.2 GDPR and the Security Monitoring Challenge

Organisations collect sensitive information like physical address location, bank account number, family members contact details and any other personal identification information. General Data Protection Regulation was introduced as a rule that will be used to ensure citizens are protected from data breaches. Access control methods in an organisation whereby each user activity is dependent on the role and permission assigned help to limit the trail audit of a user. Every activity log done by a user in a system is recorded. GDPR specifies that the data monitored or analyse should only be used for the prime purpose it was meant to (Middleton-Leal, 2018).

Cybersecurity threats monitoring is an important aspect of protecting data. It helps to monitor the access to personal data and systems that store and process it. This prevents data breaches and identifies a threat after it has occurred. GDPR specifies that cyber threats monitoring is risk-based and should apply a risk-based approach where through the use of artificial intelligence threats are detected and given a score. Monitoring user activity helps to detect any anomaly that affects the integrity and confidentiality of data (CASSETTO, 2018).GDPR standards will be maintained to ensure that user activity that is being collected or monitored will not be used in an unauthorised way.

2.7.3 ISO/IEC 27001

International Standard for Standardization (ISO) ISO 27001 is a family of standards for managing cybersecurity, it is used in the financial industry and other industries for structuring their internal processes. It is can also be used for verifying the cybersecurity capabilities of vendors (ISO, 2018). It is a standard used to provide requirements for an information security management system (ISMS). The ISMS ensures that confidentiality, integrity and availability of information are maintained. This is achieved through the application of a risk management process and gives assurance that risks are well managed. Organisations need to implement the standards to prevent cybersecurity risks by ensuring that the assets and information are secured and this also gives clients assurance that the firm is protected against the evolving cybersecurity attacks (Gasiorowski-Denis, 2016). Monitoring security activities of another system in real-time to monitor any threats that may affect the system helps in collecting logs and security events from a system or network and gives an analysis of the report by sending an alert if any anomaly is detected

(PAT Research Group, 2018). The key factors to consider when choosing a cybersecurity monitoring platform are as follows:

2.7.3.1 Data Aggregation

Through collecting logs enables the system to aggregate information from different sources and tries to make meaning from any events with common behaviour. Events with common behaviour are most likely to share a source or lead to similar results. Data aggregation facilitates the comparison of information from the various sources and identify the relationship from the obtained information (Cai, 2018)**.**

2.7.3.2 Compliance

In research from Harrington (2021), cybersecurity threats monitoring platform should be designed to process, analyse data and produce reports under a set of required security industry standards. When a platform is designed to comply with the required standards it helps to ensure that the system requirements have been followed.

2.7.3.3 Detecting of Anomalous Activity

The platform should be able to detect any suspicious activity and creates an alert to notify the required personnel to take the required action. Cyber-attacks have evolved to the point where they can pass through technical defences, blend into an environment and remain undetected as long as necessary to carry out their malicious objective. Detecting these covert assaults requires vigilant monitoring and a thorough understanding of what is considered normal behaviour in an IT environment, including both systems and users. Identifying normal vs. abnormal behaviour is the foundation for cybersecurity threats monitoring platform that has created an entire industry within the security profession (Nepal & Jang-Jaccard, 2014). It's worth noting that cybersecurity threats monitoring platforms are only effective when properly configured and have visibility to complete and accurate system information over a while. Without an accurate historical record of normal IT operating activity, it's next to impossible to detect minute variations in performance or behaviour that may subtly signal malicious activity.

2.7.3.4 Intrusion Detection

The intrusion detection technique is the process of detecting malicious patterns in a given data set (Khraisat, 2019). It uses analysis tools to find attacks, valid patterns and relationships in large data sets. Whenever the system finds a security threat it generates an alert to indicate the existence of

intrusion. It also analyses and predicts anomalous activities through the use of various types of parameters that examine the data and include classification, clustering and analysis methods. An intrusion detection algorithm should detect a scanning attack, a denial of service attack and a penetration attack to prevent unauthorised access of data (Kumari & Soni, 2017). A cybersecurity monitoring platform requires an intrusion detection capability that will be used to identify attacks before they can penetrate the system so that necessary countermeasures can be done early enough.

2.7.3.5 Dashboard Reporting

The cybersecurity monitoring platform provides reports on security-related incidents and events, such as successful and failed logins, malware activity and other possibly malicious activities. Dashboard reporting is an event monitoring tool that helps to keep information and data secure (*Security Dashboard - an Overview | ScienceDirect Topics*, n.d.). It facilitates the visibility of the granular details of user activity in an organisation.

2.7.3.6 Real-time Notifications

In research from Winkler (2011), a monitoring platform should have a real-time notification system whenever a critical activity or threat has been discovered in the monitored platform. The mode of notification should also be customizable through a variety of options to try to ease up notifications overload. Real-time notifications in the use of professional events monitoring platforms in case of anomalies help to fix the anomalies or prevent further damage of an act will be received faster compared to working alone without the events monitoring platform. In research from Rosencrance (2020), real-time notifications facilitate effective security monitoring which requires collection, analysis and correlation of data security from across the cloud and on-premises environments to identify threats and intrusions.

2.8 Conceptual Framework

Based on the literature reviewed and the various gaps identified, this project utilises the following conceptual framework to detect anomalies in using the RNN algorithm model that will be in the security threats the monitoring platform. Data that is collected from user activities in the platform that is being monitored goes through an anomaly detection model to learn how the normal user system behaviours and the pattern is (Shimeall & Spring, 2013). A dataset will be used to train the model which will contain actual data that represents different types of attack. The RNN anomaly

detection model will then classify among the activities passed through whether they are normal or an anomaly that is a threat to the system. The results of the classification will be stored in the database for future reference. The monitoring platform will be used to alert and display the results to the system administrators for further action while the normal user activity will be given access to the platform. Fig 2.10 below illustrates the conceptual framework.

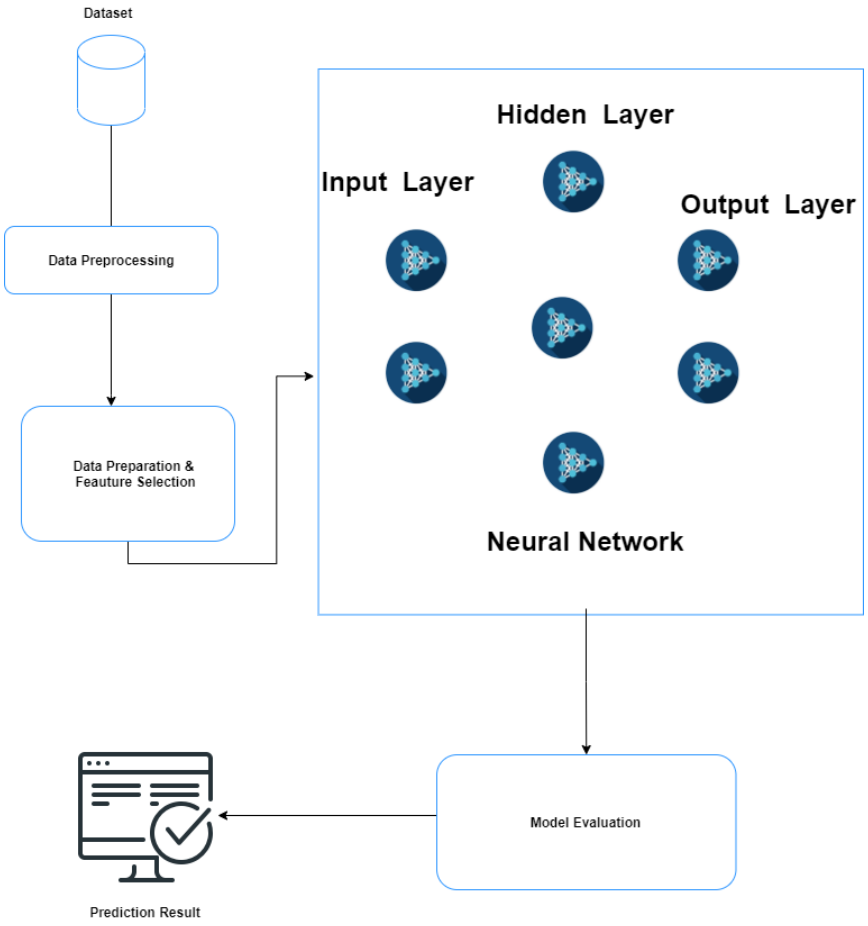


Figure 2.10 RNN Conceptual Framework

Chapter 3: Research Methodology

3.1 Introduction

This chapter describes the methodology to be employed to address the objectives of this research. Through the analysis conducted by the study, cyber security threat detection approaches and techniques were evaluated. Additionally, methods used for detecting zero-day cyber security threats will be addressed following the study requirements. The envisioned model could detect zero-day cyber security threats and validate the effectiveness of the developed model to detect zero-day cyber security threats. The model that we are going to look at is the RNN model. The proposed model is self-adaptive because it can become more refined and better with time.

3.2 Research Design

The research design refers to the general strategy that will be put into use to integrate the various components of a study to develop a logical and coherent evaluation (De Vause, 2006). The quasi-experimental research design was used for this study. Quasi-experimental research designs tested the relations in given environments intending to analyse outcomes of interest-based on given variables. This research design approach used a case of an identified network dataset to test whether the proposed machine learning techniques applied for vulnerability detection will be applicable and effective. Luckily, applied machine learning offers the researcher complete control over the experiment hence one can execute as many trials as possible further enhancing the results of the study.

3.3 Model Methodology

The RAD model is a sharp alternative to the typical waterfall development model, which often focuses largely on planning and sequential design practices (Waters, 2014). this methodology was designed to allow designers to make changes easily after the initial development is complete as compared to the traditional methods of software development. It involves continuous planning, continuous testing and continuous integration to ensure the development of an effective zero-day attack detection model. Figure 3.1 demonstrates the secure RAD life cycle.

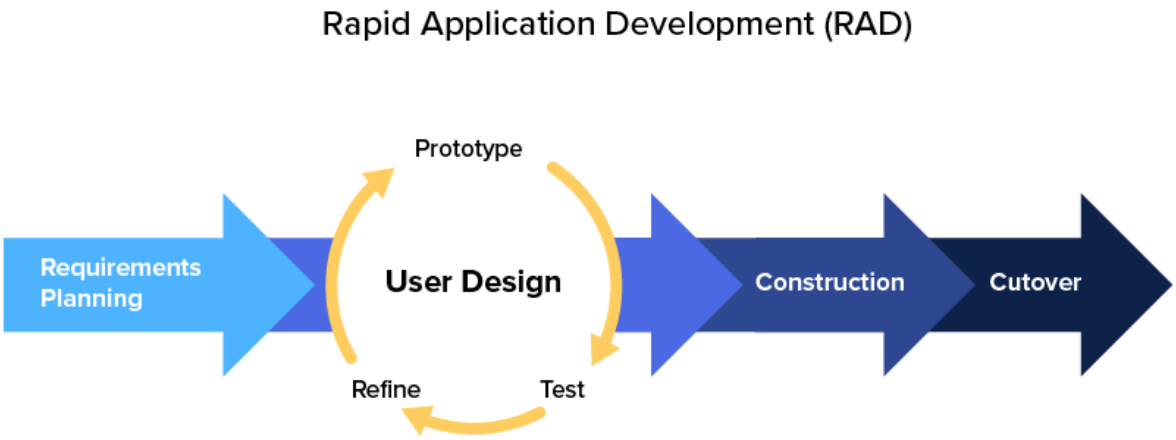


Figure 3.1 RAD lifecycle

3.3.1 Requirements Phase

This was the initial phase that focused on gathering the requirements for the proposed RNN model for detecting zero-day attacks. At this phase, I identified the requirements based on already existing methods of detecting zero-day attacks. The requirement analysis for this research was obtained through a critical analysis of the gaps that had been identified in the existing methods of detecting cyber security threats and detecting zero-day attacks. The outcome of this phase was to define the technical approaches that will be followed to successfully develop the model.

The threat detection approaches and techniques that are already existing are non-machine learning techniques (which are measures such as signature-based approach and anomaly detection) and the machine learning techniques are KNN, Back-propagation Neural Network (BNN), and Decision Tree (DT) among others. In this research, we used machine learning techniques. The best way to detect zero-cybersecurity attacks is through the use of behaviour analysis. The model has been trained using a known data set that contains normal and suspicious traffic. Using deep learning algorithms, the model is then able to learn the behaviour of the network traffic and distinguish normal traffic from any network anomalies that can infer an attack.

3.3.2 System Analysis

System analysis involved identifying ways to break down a large problem by breaking it down into small components which are again decomposed into smaller components. Each of the smaller components can be solved as a single unit, and therefore this is what is referred to as system design (Waldo, 2006). Once components of the platform were identified, it followed those system requirements that were drawn from each of those components which allowed one to describe the

functionalities and come up with the design of the platform under study. Some of the steps followed in system analysis and design include feasibility study, investigation of the current environment, the definition of requirements, technical system options, logical design and physical design.

3.3.3 User Design Phase

The design stage of the model consisted of a detailed analysis of the activities related to the system that acted as a guide to develop a secure design of the model. This involved decomposing the various functions of the system and creating action diagrams to represent the interaction between the user and the system. It also involved the creation of layout screens of the systems and basic functionalities and procedures that can produce a prototype of the model to identify vulnerabilities. The tools that were used for the analysis and design of the data and process included the following;

i. System architecture – The system architecture explains the whole process of how the RNN model is trained and tested for it to be able to detect a zero-day attack
ii. Use Case diagrams – This has been used to show how a user interacts with the platform for detecting a zero-day attack
iii. Sequence diagrams – Has been used to show the whole cycle of training and testing the model
iv. Database schema - Has been used to show the whole database structure layout and relations association

3.3.4 Prototype Development Phase

This phase is where the development of the prototypes of the model began. The development process was iterative to allow better performance of the detection model. The tools used include:

i. Keras. This is a high-level neural networks API, written in python that allows for easy and fast prototyping through user-friendliness, modularity and extensibility. We utilized Keras on top of TensorFlow. Keras was picked as it is intended for quick prototyping and experimentation with a basic API. It permits to design NNs in a secluded manner by joining various layers, actuation and loss functions, and optimizers, and so on. Keras gives great answers for the greater part of the standard deep learning building blocks. Nonetheless, on the off chance that somebody needs to assemble a custom or novel execution, Keras API could be very restricted, and libraries like TensorFlow will be a superior decision. Keras

contains the execution of LSTM with forget doors. There are two significant subtleties, clarified beneath, crucial for seeing how LSTMs are executed in Keras work. In Keras, an adjusted adaptation of BPTT is executed. To unfurl RNN across the whole input sequence comprising of hundreds or even a large number of time steps is computationally wasteful. In this way in Keras RNN is unfurled up to the greatest number of time steps. This parameter is given through the input system. Input data is taken care of as a three-dimensional cluster of shape: (batch_size, lookback, input_dimension). The subsequent sequence, lookback, indicates the number of time steps for which the RNN is unfolded. The input information is isolated into overlapping sequences with a period time frame. Each arrangement has a lookback number of successive time steps and structures one preparing test to the RNN model. During preparing, BPTT is done distinctly throughout individual examples for lookback time steps.

ii. TensorFlow which is a machine learning framework for building machine learning models was used together with the Scikit-learn python library to develop the RNN anomaly detection model.

iii. NumPy and Pandas are among other python libraries used for the development of the algorithm.

3.3.5 Testing

Testing involved validation of the anomaly detection model using the testing dataset that contained test data that was not present during the training phase of the model.

3.4 Data Collection

The primary source of data used in this study was the UNSW-NB15 dataset. This data set is a hybrid of the real world and modern-day activities and contemporary synthetic attack behaviours. The dataset contains a list of attacks, records without anomalies, categorisation of the type of attack and the features available on the dataset (*The UNSW-NB15 Dataset | UNSW Research*, 2021). The description below gives more information on what was entailed in the identified dataset that was used in this study.

i. UNSW-NB15_features.csv - Contains a list of the features available in the dataset

ii. UNSW-NB15_LIST_EVENTS.csv - Contains a list of cyber security events and their categorisation
iii. UNSW_NB15_training-set.csv - Contains a record of normal and attack classes that will be used in training the model to detect anomalies
iv. UNSW_NB15_testing-set.csv - Contains a record of normal and attack classes that will be used in testing whether the trained model will be able to predict cyber security attacks.

3.5 Data Analysis

The UNSW-NB15 dataset contains a large dataset with a wide range of variables. Thus, before using its content, an analysis was done on the general record to generate a specialised dataset for the proposed model. Nonetheless, the majority of UNSW-NB15 dataset records were characterized either as an attack or non-attack. The records had been verified by human experts to facilitate the training of the classification model.

3.5.1 Data Cleaning

Data cleaning was done on the UNSW-NB15 dataset. The matching and selection of data files, as well as example cases, were all part of the cleaning process. Furthermore, the selected records were checked for missing and erroneous data, as these could have a significant impact on the outcomes of the trials. Erroneous data was eliminated using one of two methods: deletion or replacement. To replace the deleted instances, regression imputation was utilized. Missing data were replaced with estimation values in regression imputation. As a result, it aids in the improvement of study quality by reducing the impact of missing and erroneous values. This process ensured all data points were consistent with the objectives of the study and meet the identified classification criteria.

3.5.2 Data Classification

Multivariate statistical approaches were used to do a descriptive analysis first. The association between the various features was established through this analysis approach. To identify any new connections between the variables in the data, the exploratory analysis was added to the general evaluation. These links also aided in redefining the study's and future models' questions. The significance of these data analysis approaches was determined by the necessity to achieve the

research's goals. The first method, which provided the foundation for data evaluation, described the data by broadly characterizing its constituents. The second method then offered the important correlations between the features (variables) in use, while the third method gave a way to forecast future results based on historical data.

3.6 Research Quality

To ensure quality, the research was conducted in an ethical manner and line with the regulations of the university as well as other set standards. The research process was conducted with the platform requirements in mind by ensuring the research questions and objectives have been met. The project was expected to follow quality standards by ensuring that any piece of work from external sources will be properly cited and that the literature review was obtained from reputable sources. Furthermore, the model was developed using verified data, which improved the quality of the final results even further.

3.6.1 Reliability

The UNSW-NB15 dataset that was used for this study was split into two. This was used to provide a testing dataset and a training dataset which helps in assessing how the algorithm will perform In this situation, MSE(Mean Squared Error) validation method was utilized, and a new data set (testing data) x was used. The data set's values were all indexed and labeled appropriately (xi). The xi was compared to another categorized data set, Yi, to assess the model. The error (the difference between f (xi) and y) was computed in this comparison using MSE. After that, the resultant value was squared, and all accessible values were averaged (mean). This can be represented mathematically as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y - \hat{f}\,(x_i)\right)^2$$

3.7 Ethical Considerations

This study was based on the researchers' original ideas and any externally borrowed notion used in the study was explicitly acknowledged and cited in text to acknowledge the data sources and

contribution to the study. Furthermore, the information gathered was secondary in nature and came from the University of South Wales (UNSW) research lab, which does not provide any unique identifiers that might be used to link an individual. Any genuine personal information had already been removed from the file, so there was no risk of a data breach.

In addition, the research was submitted for Ethical approval to the Strathmore university Institutional Ethical Review Committee.

Chapter 4: System Design and Architecture

4.1 Introduction

This chapter will outline the architecture design to detect zero-day vulnerabilities using the RNN. It dives into the functional and non-functional requirements of a system and how the model is expected to function. The system design and architecture followed the methodology guideline in chapter 3 of this research.

4.2 Data Analysis

The developed model would pick out the defining features to classify raw content. Therefore, the data used for the study needed not only to be based on the objectives of the study but also unbiased to any classification. This requirement meant an equal number of data labels for both attacks and non-attack records. The goal for analysis of the UNSW-NB15 dataset was to better understand the data and select the features that would ensure the data is fit for the study. The data analysis steps applied in this study were as follows;

4.2.1 Loading the dataset

The first step was to do a descriptive analysis to confirm the shape,size and columns of the dataset.This was performed to help identify the key elements of the dataset that would form the basis of this study.Figure 4.1 below is a screenshot of the dataset loading process and the process to view the first view of the datset.

Loading Training Dataset

```
import pandas as pd
train = pd.read_csv('/Users/JorgesMuchiri/RNN/train.csv')
```

Python

+ Code + Markdown

```
train.head()
```

Python

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	c
0	1	0.000011	udp	-	INT	2	0	496	0	90909.0902	...	1	2	0	
1	2	0.000008	udp	-	INT	2	0	1762	0	125000.0003	...	1	2	0	
2	3	0.000005	udp	-	INT	2	0	1068	0	200000.0051	...	1	3	0	
3	4	0.000006	udp	-	INT	2	0	900	0	166666.6608	...	1	3	0	
4	5	0.000010	udp	-	INT	2	0	2126	0	100000.0025	...	1	3	0	

5 rows × 45 columns

Figure 4.1 Loading the training dataset

4.2.2 Checking for datatypes in the dataset

The next step involved checking the datatype for each column in the dataset usimg the .info() function. The function also returns about null and not null values present in the dataset.figure 4.2 below shows an illustration of how the function is executed and its output.

```
train.info()
```

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 82332 entries, 0 to 82331
Data columns (total 45 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 82332 non-null  int64
 1   dur                82332 non-null  float64
 2   proto              82332 non-null  object
 3   service            82332 non-null  object
 4   state              82332 non-null  object
 5   spkts              82332 non-null  int64
 6   dpkts              82332 non-null  int64
 7   sbytes             82332 non-null  int64
```

Figure 4.2 Checking Datatypes

4.2.3 Checking for missing values in the dataset

After loading the data the next step involved checking for any missing values to avoid any errors in the final data analysis process.Fig 4.3 below illustrates the isnull().sum() function which is used to check how many missing values are present in each column. In our case, in both the data sets we did not get any of the missing values in any of the columns.

```
train_data.isna().sum()
```

Python

```
id                0
dur               0
proto             0
service           0
state             0
spkts             0
dpkts             0
sbytes            0
dbytes            0
rate              0
sttl              0
dttl              0
sload             0
dload             0
sloss             0
dloss             0
sinpkt            0
dinpkt            0
sjit              0
djit              0
swin              0
stcpb             0
```

Figure 4.3 Checking for missing values on the dataset

4.2.5 Features Encoding

The next step involved ensuring that all the categorical fata have been encoded to numbers so that it can fit well for the model as it requires all the variables to be numeric. Figure 4.4 below shows an illustration of how the feature encoding procedure was done.

```
# import LabelEncoder
from sklearn.preprocessing import LabelEncoder
# Instatniate LabelEncoder
le = LabelEncoder()
# LabelEncode Class column of df
data["service_le"] = le.fit_transform(data["service"])
data["attack"] = le.fit_transform(data["attack_cat"])
# Inspecting encoded df
data.head()
```

(variable) data: Series[Dtype@concat]

Python

	dur	proto	service	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	...	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_sr
29	1.434166	1	-	10	6	530	268	10.459041	254	252	...	0	0	0	
31	1.924287	1	-	10	8	530	354	8.834441	254	252	...	0	0	0	
32	2.773779	1	-	14	8	7954	354	7.570899	254	252	...	0	0	0	
34	1.465041	1	-	10	8	2516	354	11.603770	254	252	...	0	0	0	
35	0.983874	1	http	10	8	816	1172	17.278635	62	252	...	0	0	1	

5 rows × 45 columns

Figure 4.4 Features Encoding

4.2.4 Feature Selection

The next step involved performing a feature selection inorder to reduce the complexity of the model to learn and also to identify features which contribute most to the prediction variable or

output that is in line with this study.Figure 4.5 below shows an illustration of how the feature selection procedure was done.

import library for feature selection

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
```

Python

```
best_features = SelectKBest(score_func = f_classif,k=9)
fit = best_features.fit(X,y)
df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(X.columns)
```

Python

+ Code + Markdown

```
features_scores = pd.concat([df_columns,df_scores],axis =1)
features_scores.columns = ['Feature_Name','Score']
```

Python

```
print(features_scores.nlargest(9,'Score'))
```

Python

```
        Feature_Name         Score
7               sttl  18964.086751
32  ct_dst_sport_ltm  16135.278766
31  ct_src_dport_ltm  12505.863410
1              proto  11096.945966
17              swin  11087.382800
38       ct_srv_dst   8354.924821
28       ct_srv_src   8199.287505
33   ct_dst_src_ltm   8126.081994
20              dwin   7960.170077
```

Figure 4.5 Feature Selection

4.2.6 Standardization and scaling of data

The last step involved scaling and standardization of the data to ensure that the data is in a uniform format and all the variables are on the same scale.This helps in I proving the quality of the data that is being analysed.Figure 4.6 below illustrates the function.

Scaling the features

```
X = (scaler.transform(X))
X
```

Python

```
array([[ 0.77099348, -0.44259086, -0.47371476, ..., -0.67590125,
        -0.56920018,  0.86647113],
       [ 0.77099348, -0.44259086, -0.47371476, ..., -0.67590125,
        -0.56920018,  0.86647113],
       [ 0.77099348, -0.44259086, -0.47371476, ..., -0.76190497,
        -0.56920018,  0.86647113],
       ...,
       [ 0.77099348, -0.44259086, -0.47371476, ..., -0.76190497,
        -0.56920018, -1.15425294],
       [ 0.77099348, -0.44259086, -0.47371476, ..., -0.76190497,
        -0.48605722, -1.15425294],
       [ 0.77099348, -0.44259086, -0.47371476, ..., -0.76190497,
        -0.56920018, -1.15425294]])
```

```
X=np.asarray(X).astype(np.float32)
y=np.asarray(y).astype(np.float32)
```

Python

Figure 4.6 Scaling the features

4.3 Requirements Analysis

This study aimed at developing an anomaly-based cybersecurity threats detection model for detecting cyber security attacks. Based on this research objective the system requirements can be divided into two. The functional requirements and the non-functional user requirements.

4.3.1 Functional Requirements

These describe the users interaction with the system and what the model should essentially do to be deemed efficient.

- The user should be able to enter the URL of the website/system to monitor traffic.
- The tool should detect any anomalies in the traffic that can be flagged as a cyber security attack.
- The tool should update the database with any flagged record for future analysis.
- The tool should be able to send an alert when an attack has been detected.

4.3.2 Non-Functional Requirements

The non-functional requirements specify the properties of the system which will enable it efficiently perform the functional requirements. They include ;

- Reliability: The tool should be able to be relied upon by users by ensuring a high accuracy level in detecting known and unknown cyber security attacks
- User Friendly: The interface has been developed in a way that ensures a good user experience and avoids any kind of complexity
- Performance: the tool should be well trained and optimised to ensure that it functions as expected and gives an output in real-time
- Extensibility: The tool should be able to handle future growth and improvement by having a self-learning capability to improve the model accuracy

4.4 System Architecture

Figure 4.1 below shows the system architecture of the model and illustrates how it will detect zero-day attacks. The first step is preparing the dataset, the pre-processing was the process of cleaning the raw data where the data is converted to a clean data set. The main objective for this step is to ensure that data is fit for the machine learning model. The training dataset is then passed through the model and its accuracy is evaluated before making a prediction.

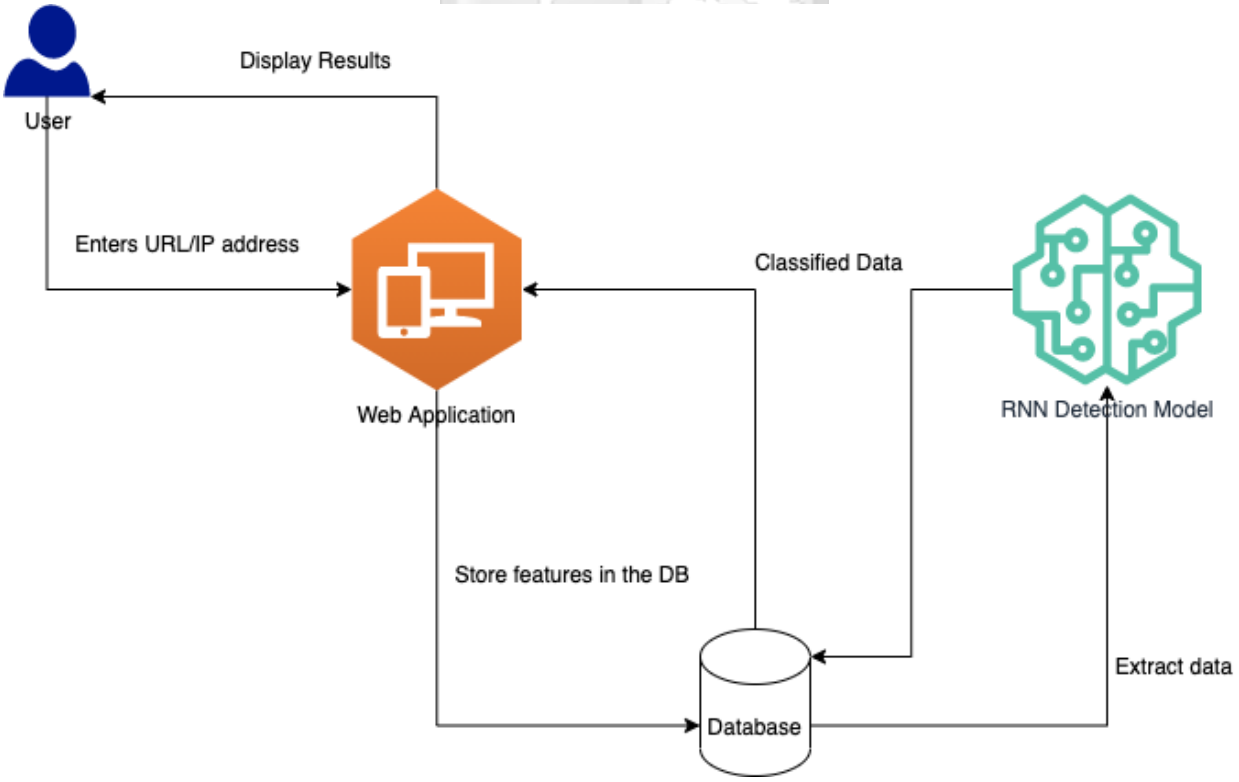


Figure 4.7 System Architecture

4.5 Use Case Diagrams

Figure 4.2 below shows a use case diagram to illustrate how the model operates. The user's input the URL / IP address of the website or system under review and the data is sifted in the input layer for preprocessing and afterwards taken to the hidden layer where the training data set is. The data is classified as normal or suspicious then proceeds to be output. If the anomaly is detected and its score matches that of an attack a positive result is predicted and the score is displayed as the result.

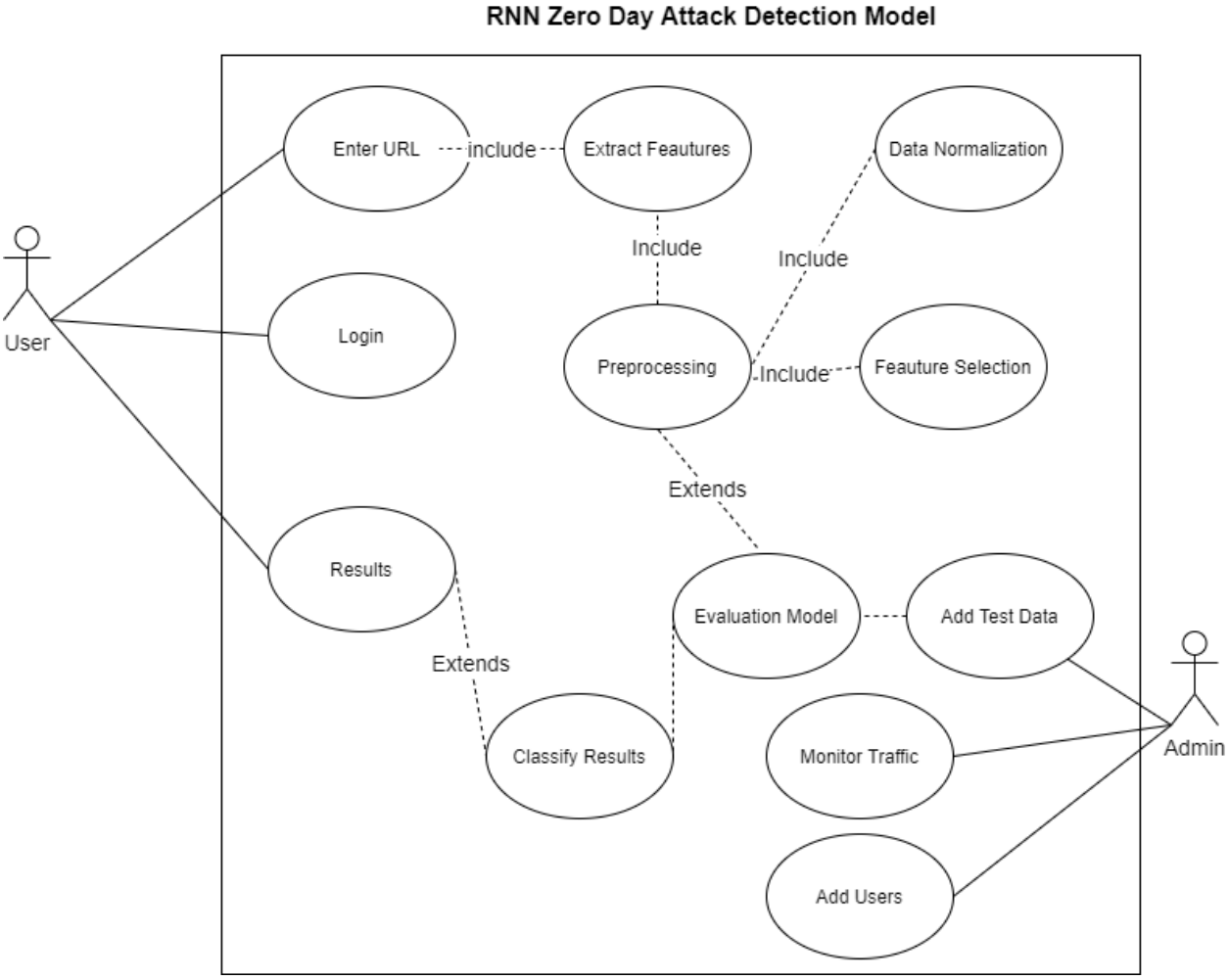


Figure 4.8 Use Case Diagram

4.6 Entity Relationship Diagram

Figure 4.9 below demonstrates the entity relationship diagram which is an abstract model that organized the elements of data exactly how they related to one another and to the properties of other real-world entities.

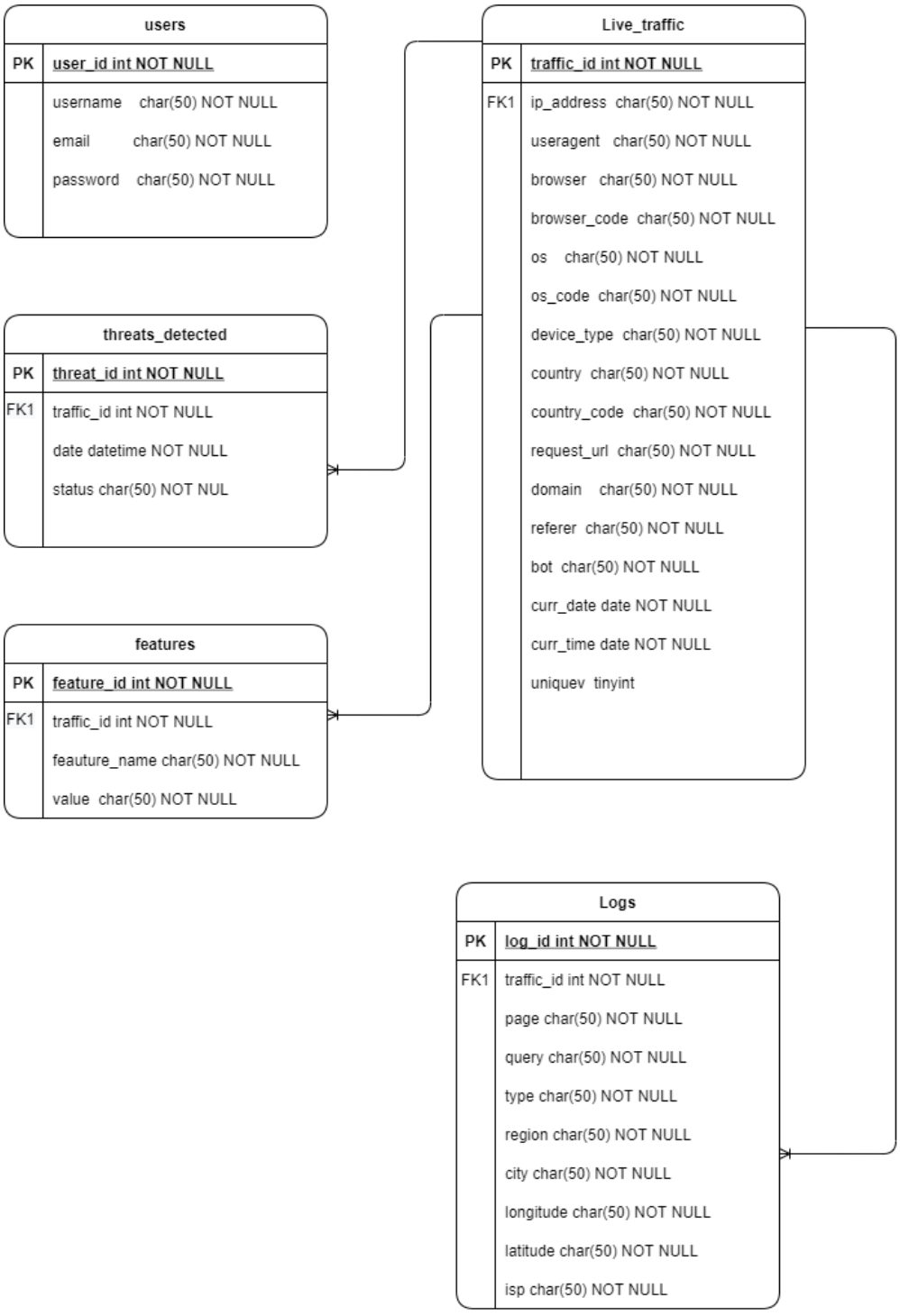


Figure 4.9 Entity Relationship Diagram

4.7 Database Schema

The database schema defines how the data was organized and how the relations among the tables were related. Figure 4.10 below shows an illustration of the database schema.

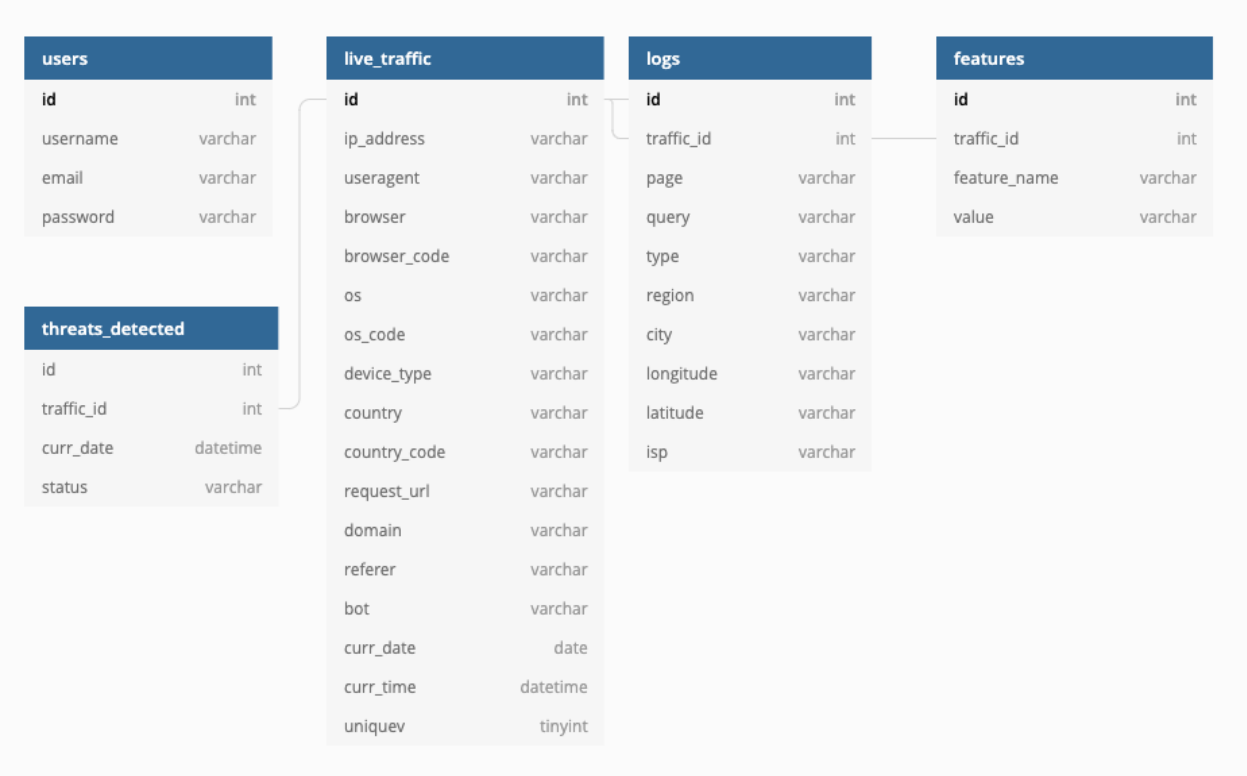


Figure 4.10 Database Schema

4.8 Application Wireframes

4.8.1 Registration Page Wireframe

Every user requires to be registered so that they can have access to the system. Figure 4.11 below shows how the user registration form will be designed.

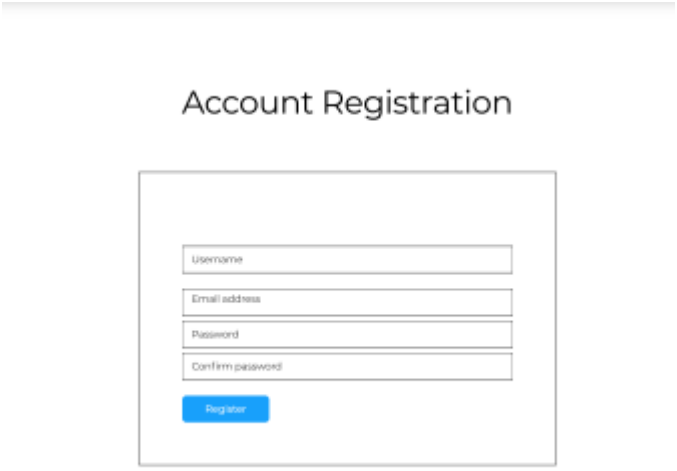


Figure 4.11 Registration Page Wireframe

4.8.2 Login Page Wireframe

After successful registration and verification of a user,they can access the system by entering the username and password on the login page as shown in figure 4.12 below.

Figure 4.12 Login Page Wireframe

4.8.3 Site Information Wireframe

Figure 4.13 below shows the wireframe design of the site information module where a user will be entering the url/ ip address of the web system to be monitored.

Figure 4.13 Site Information Wireframe

4.8.4 Dashboard View Wireframe

Figure 4.14 shows a dashboard overview design where a user of the system will be able to have an oversight statistics of the website being monitored.

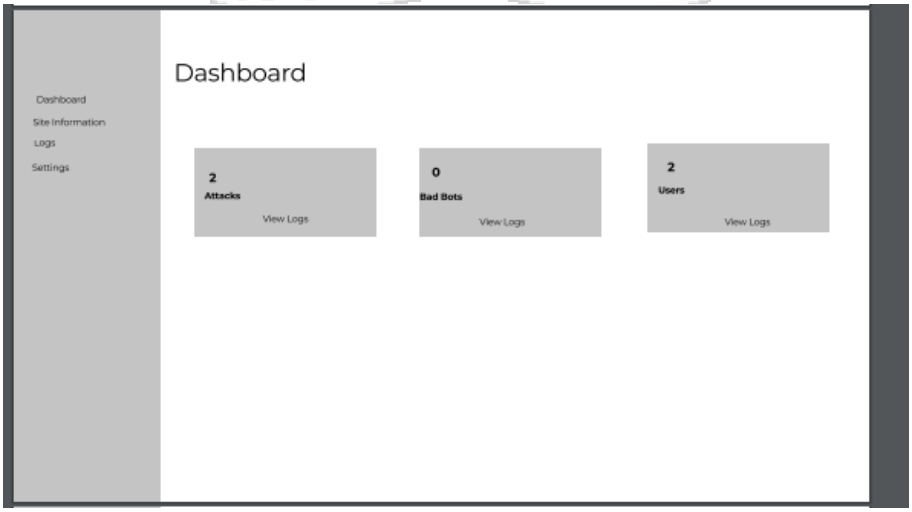


Figure 4.14 Dashboard Wireframe

4.8.5 Logs Summary Wireframe

Figure 4.15 below shows the wireframe design of the logs summary module.Logs that are generated on the site under monitor will be viewed on this page to give a summary of the details being stored.

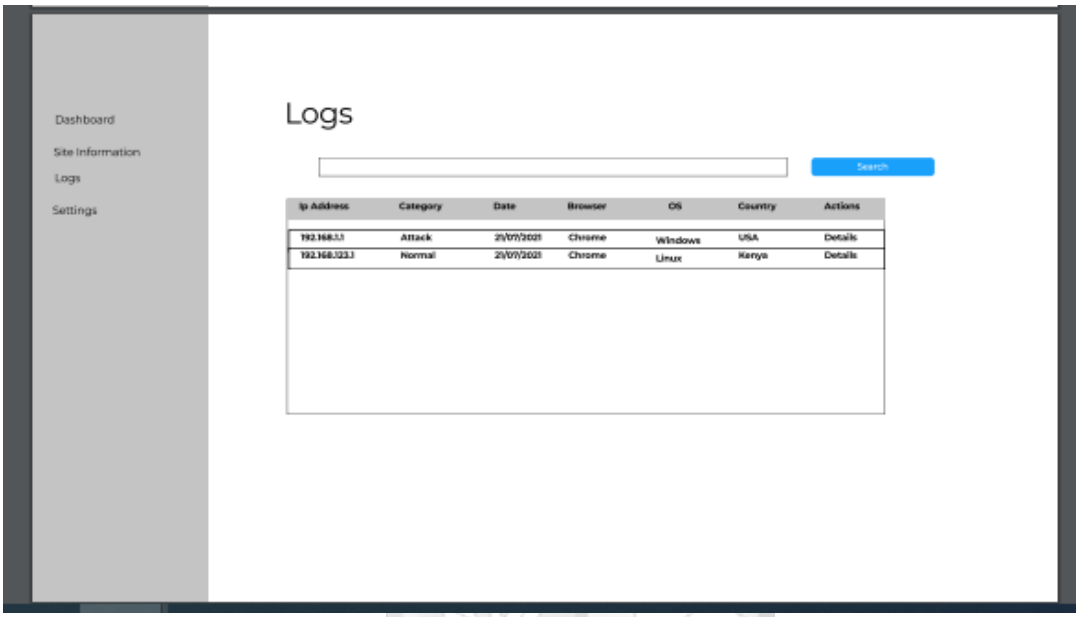


Figure 4.15 Dashboard Wireframe

Chapter 5: System Implementation and Testing

5.1 Introduction

This chapter discusses the implementation of the zero-day attack detection model using Recurrent Neural Network (RNN) and highlights how the model works, its features and how the model was tested, trained and its accuracy validated. This section also includes screenshots of significant features concerning the process of training and testing the model. The model has been implemented using the tools mentioned in chapter three of this research.

5.2 Detection Model Structure

The proposed anomaly detection tool applied a classification model to accurately identify suspicious network traffic based on unique and differentiating traits. Figure 5.1 below illustrates how the whole flow of the model operates. If the specified number of epochs has not been met, a new model is generated, and the weights and training of the model are readjusted. The trained model is then reevaluated for its optimality, and there is convergence, the model is saved as an h5 format file. The system then checks to see if the stopping condition has been met. This process is repeated until the stopping condition is met. The final model is what will be saved. The final model then gives an output based on the result detected.

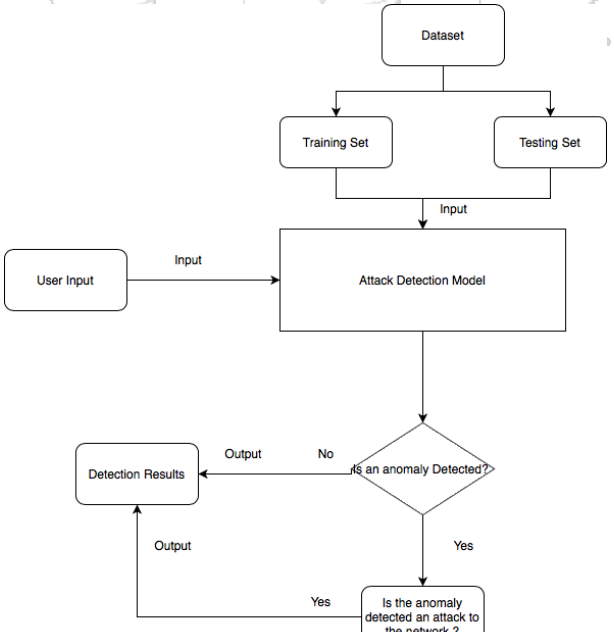


Figure 5.1 RNN Network Anomaly Detection Model

5.2.1 System Components

The developed system contained a section of input from the users where a user would enter data or parameters of interest. The users were expected to enter values that the model would then use to predict security threats to the systems it is embedded in.

5.2.2 Recurrent Neural Network (RNN) Components

The RNN consists of three components the input layer, the hidden layer and the output layer. They all function to predict and produce output based on an input provided by the user.

Its architecture comprises of the following:

i. **Bidirectional Recurrent Neural Networks (BRNN)** - Bidirectional recurrent neural networks (BRNN) works to link two hidden layers of different directions to identical results. In the system creation, BRNN was used as it provides a better prediction on the system to avoid cyber attacks. These networks pull data from the future to predict the user's input.

ii. **Long Short Term Memory (LSTM)** - Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) that can process sequences of data because of feedback connections. This component allows the system to "remember" information to remove the vanishing gradient problem of RNN. They are located in the hidden layer and have an input, output, and forget date to regulate the prediction of the data that is anticipated in the system.

iii. **Gated Recurrent Network (GRU) –** This is a variation of the LSTM and helps solve the gradient problem of RNN, using reset and update gate**.** These are also located in the hidden layer and have two gates. They work mainly to address the short-term memory of the RNN model. They have an update and reset function to regulate data and how much is gained or lost.

5.2.3 RNN Model implementation

A user logs in to a system and then enters the url / ip address of the system or website to monitor. The tool detects the traffic on the pages and asserts if it's normal or an anomaly. The data goes through the RNN detection model to determine this behaviour. If there is abnormal behaviour that infers to an attack the RNN creates an alert of suspicious activity**.**

5.2.4 Recurrent Network Model

The data collected from the generated traffic undergoes the hidden layer which has the training and validation data set. It consists of one input layer with 12 nodes, five hidden layers and one output layer with one node. The input layer provided the input data for the model, the hidden layers were used to enhance the model prediction accuracy while the output layer gives the prediction made by the model. The model is then filled with the training data set and validation data set. 50 number of iterations was the optimum number of times required to test the model. Additional images on the training data set and feature data set can be found in appendices A and B.

Figure 5.2 below illustrates how the model was created using the Keras Library

```
model = keras.Sequential()
model.add(layers.Dense(12, input_dim  = 9, activation="relu"))
model.add(layers.Dense(10, activation ='relu'))
model.add(layers.Dense(8,  activation = 'relu'))
model.add(layers.Dense(6,  activation = 'relu'))
model.add(layers.Dense(4,  activation = 'relu'))
model.add(layers.Dense(2,  activation = 'relu'))
model.add(layers.Dense(1,  activation ='sigmoid'))
```

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['acc'])
```

```
history = model.fit(X_train, y_train, epochs=50, batch_size=64,validation_data=(X_val, y_val))
```

Figure 5.2 Neural Network Model Code Illustration

Figure 5.3 below illustrates a visualization of the ‘val_loss' which is the difference between the predicted value and actual value against the number of iterations. The model was passed through the loss function to determine how effective the model is in predicting the expected value.

Figure 5.3 below illustrates a visualization of the ‘val_loss’

.

```
In [81]:  1 plt.plot(range(50), history.history['val_loss'])
          2 plt.ylabel('val_loss')
          3 plt.xlabel('Epoch')
Out[81]: Text(0.5, 0, 'Epoch')
```

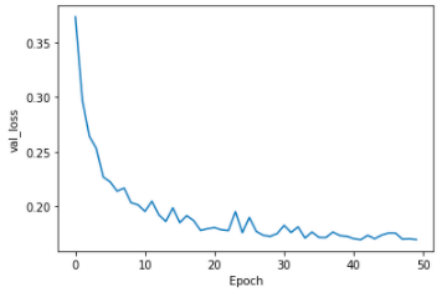


Figure 5.3 Val_loss against the Number of Iterations

5.3 Model Testing and Training

The model was provided with a training data set and a validation data set over 50 iterations. The model passed through a training loss function to measure how fit it was for ensuring a higher accuracy rate.

Figure 5.4 below illustrates the visualization of the comparison between the validation loss against the training loss.[The training loss in blue and validation loss in orange]

```
In [82]:  1 plt.plot(history.history['loss'], label='Training loss')
          2 plt.plot(history.history['val_loss'], label='Validation loss')
          3 plt.legend(frameon=False)
Out[82]: <matplotlib.legend.Legend at 0x7fc6d2886e50>
```

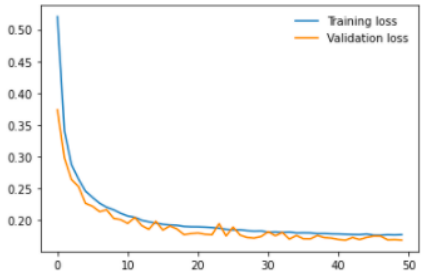


Figure 5.4 Comparison Between the Training Loss and the Validation Loss

Figure 5.5 below data illustrates the training results metrics of 50 number iterations performed on the model.[The Val_loss, Val_acc]

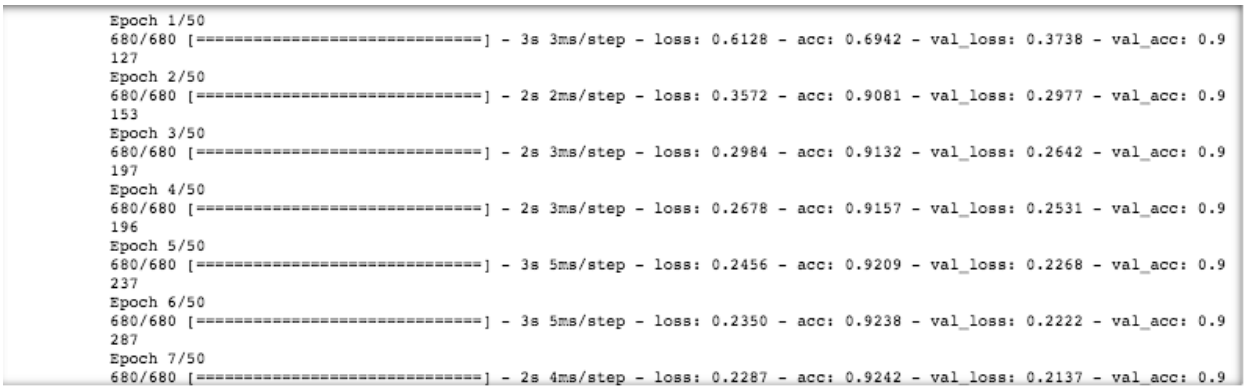


Figure 5.5 Training Results Metrics

The model has a 93% accuracy, precision, recall and f1_score. Figure 5.6 shows the accuracy and precision of the model

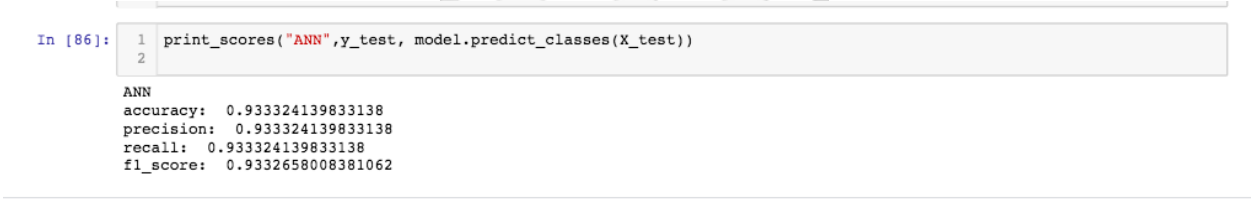


Figure 5.6 Model Evaluation Scores

5.4 Usability Testing

The model was deployed on a web-based platform with a simple user interface that a user could easily input data and interact with the model efficiently. The software used was Flask, deployed in the Streamlight platform.

5.5 Performance Testing

The purpose of this test was to ensure that the system is can perform all its tasks and functionalities correctly as required from beginning to end of a session

Performance Measure	Test	Result
Accuracy	Check if the model has a good accuracy	Successful
Speed	Check if the system responds quickly	Average result
Scalability	Test if the system can do multiple things at a time	Successful

Security	The system checked if an unauthorised user can access it	Successful

5.6 Integration Testing

This stage was to ensure that all the components of the system worked correctly together as one. the integration of the model and the system was tested and the result was properly. The system picked all the input data and performed an analysis to detect the presence of vulnerabilities.

5.7 Model Evaluation

The performance of the model was evaluated by using a separate dataset that has not been exposed to the model during training. This will give an estimate of the predictive performance of the RNN for unseen data in the future. Generally, lower loss means that the model is performing very well since the difference between the predicted output and expected output should be as small as possible. Hence the closer the model loss is to zero the better. On the other hand, the higher the accuracy the better.

Chapter 6 : Discussion

6.1 Introduction

This chapter explains the findings of this research focusing on the set objectives, research questions, and scope, with explanations on the key areas covered. The main objective of the research was to develop an anomaly-based cyber security threats detection model using the Recurrent Neural Network technique.

6.2 Cybersecurity threat detection and approaches

The first objective of this study was to review existing cyber security threat detection approaches and techniques. In the literature review from sections 2.2 to 2.4, there are various cyber security threat approaches and techniques that were discussed. Many research papers were reviewed to see how different scholars have discussed the various methods that can be used for detecting cyber security attacks. The findings show that the main approaches and techniques that are currently in use are the use of the signature-based approach and the use of anomaly detection.

6.3 Methods and tools used for detecting zero-day cyber security attacks

The second objective was to review existing methods and tools used for detecting zero-day cyber security attacks. The literature review highlighted existing methods and tools used for detecting zero-day cyber security attacks. The main methods discussed include the use of threat intelligence, use of SIEM correlation techniques, correlation processes, and behavior analysis in a network, and the use of the machine and deep learning methods. All the methods that have been discussed are in line with the study findings.

The study findings also show that although cyber security solution vendors are growing better at identifying zero-day flaws, the frequency of zero-day assaults and their effectiveness continue to rise. Devices and procedures for creating zero-day attacks, spreading them, and performing zero-day exploits are turning out to be more modern, more inescapable, and simpler to use continuously. Then again, identifying a progressing cyberattack and forestalling the harm is getting a lot harder because of the expanded intricacy of the frameworks that network safety experts are managing and the high-level degree of encryption and confusion utilised by pernicious programming.

This implies that there's an expanded interest for zero-day attacks discovery and avoidance arrangements that can create exact outcomes with fewer false positives (non-occurrences that raise the alert) and false negatives (real episodes that don't raise the caution).

6.4 Design and develop a tool for detecting zero day cyber security attacks

The third objective was to design and develop a RNN tool for detecting zero-day cyber security attacks. The literature review has raised the need to adopt the use of deep learning methods for handling cyber security. To fill that gap in this research focused on how artificial neural networks can be used to detect cyber security threats, this approach was implemented by developing a network anomaly detection model using RNN. The model was designed and developed based on the functional and non-functional requirements that were gathered.

6.5 Testing the tool to detect zero day cyber security attacks

The fourth objective for this research was to test the ability of the tool to detect zero-day cyber security attacks. Testing involved validation of the anomaly detection model using the testing dataset that contained test data that was not present during the training phase of the model. The model passed through a training loss function to measure how fit it was for ensuring a higher accuracy rate and from the function, an accuracy rate of 93% was recorded. This proved that the model was effective to accomplish the objective of this study.

6.6 Summary

This chapter discussed how the research objectives were actualized and how machine learning algorithms can be used to try and solve the challenges of cyber security threats detection of known and unknown attacks. The chapter also discusses how the RNN algorithm can be used to develop the anomaly detection model for detecting zero-day attacks.

Chapter 7: Conclusion, Recommendations and Future Work

7.1 Conclusion

This research aimed to develop an RNN model which will be used to detect zero-day vulnerabilities. The main objective was met by proving that the results generated by the Neural Network algorithm can be used to detect zero-day attacks in a network.

The RNN model was trained and had a 93% accuracy rate in predicting zero-day attacks. Which made the model highly effective in detecting the zero-day attacks based on the dataset that was used for training and testing the model.

The results of the model prove that the developed RNN model can be used in Intrusion detection systems to create alerts whenever a zero-day attack has been detected and prevent the attack from affecting any given network or system.

7.2 Recommendations

The model can be deployed into intrusion detection systems or network monitoring systems to enable detection of not only known cyber security attacks but also zero-day attacks. The real-time monitoring and creating of alerts whenever an attack has been detected will enhance the security of the systems and the network.

The use of the RNN model should also be taken up by as many organisations as possible with an accuracy level of 93%, this shows that chances of not foreseeing an attack are very minimal. The model can be improved to increase the accuracy level and can also be integrated with any system that requires cyber protection against malware and hacking threats.

A combination of multiple deep learning and machine learning algorithms where an output of one algorithm can be used as an input of another can be implemented to have a much accurate and reliable model that can be used in intrusion detection systems.

7.3 Future Work

This research had the objective of developing a deep learning algorithm embedded into a Web App to detect zero-day vulnerabilities. The future work will involve deploying the deep learning model on a network monitoring system so that it can detect known and unknown attacks in real-time as the network is being monitored and also classify them and enforce required measures before it compromises a network or system.

The model will also be expected to have self-learning capabilities based on the data stored from the real-time traffic being monitored to improve its effectiveness and accuracy.

References

Aduol, L. O. (2015). *Sacco Web-Based Management Information System (A Case Study of Kisumu Teachers Sacco).* Retrieved from Sacco Web-Based Management Information System (A Case Study of Kisumu Teachers Sacco: http://sci.uonbi.ac.ke/content/sacco-web-based-management-information-system-case-study-kisumu-teachers-sacco

Alien Vault. (2018). *Detect Advanced Threats with Endpoint Detection and Response (EDR).* Retrieved from Detect Advanced Threats with Endpoint Detection and Response (EDR): https://learn.alienvault.com/c/endpoint-detection-a?utm_internal=threatdetectionlookbook&x=HpcX5z&xs=42488

Astra Web Security. (2018). *Rock Solid Website Security.* Retrieved from Rock Solid Website Security: https://www.getastra.com/features

Ayei E. Ibor1, F. A. (2018). A Survey of Cyber Security Approaches for Attack Detection, Prediction, and Prevention. *A Survey of Cyber Security Approaches for Attack Detection, Prediction, and Prevention.*

B.Chandrashekhar. (2018, December). *Continuous Risk Assessment and Continuous Mitigation*. Retrieved from Continuous Risk Assessment and Continuous Mitigation: https://sanernow.com/blog/continuous-risk-assessment-and-mitigation/

Bouveret, A. (2018). *Cyber Risk for the Financial Sector: A Framework.* Retrieved from Cyber Risk for the Financial Sector: A Framework: https://www.imf.org/~/media/Files/Publications/WP/2018/wp18143.ashx

BRICATA. (2019). *Layers of Cybersecurity: Signature Detection vs. Network Behavioral Analysis.* Retrieved from Layers of Cybersecurity: Signature Detection vs. Network Behavioral Analysis: https://bricata.com/blog/signature-detection-vs-network-behavior/

Brook, C. (2018). *What is the NIST Cybersecurity Framework?* Retrieved from What is the NIST Cybersecurity Framework?: https://digitalguardian.com/blog/what-nist-cybersecurity-framework

Buckner, M. (2018, May). *The Importance of Cyberthreat Security Monitoring*. Retrieved from https://www.pcpc.tech/2018/08/the-importance-of-cyberthreat-security-monitoring/

Bwana, K. M. (2013). Issues in SACCOS Development in Kenya and Tanzania: The Historical and Development Perspectives. *IISTE*.

Byun, H., & Lee, S.-W. (2002). Applications of Support Vector Machines. *Springer-Verlag Berlin Heidelberg* , 213-236.

CASSETTO, O. (2018, April). *GDPR and the Security Monitoring Challenge*. Retrieved from GDPR and the Security Monitoring Challenge: https://www.exabeam.com/information-security/gdpr-and-the-security-monitoring-challenge/

CHUANLONG, Y., YUEFEI, Z., JINLONG, F., & XINZHENG, H. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE*.

Cypher Security. (2018). *A Quick NIST Cybersecurity Framework Summary*. Retrieved from A Quick NIST Cybersecurity Framework Summary: http://blog.cipher.com/a-quick-nist-cybersecurity-framework-summary

Communications Authority of Kenya. (2018, December). *Regulator warns of the rise in cybersecurity threats.* Retrieved from Regulator warns of rising in cybersecurity threats: https://www.businessdailyafrica.com/corporate/tech/Regulator-warns-of-rise-in-cybersecurity-threats/4258474-4927994-c4k826z/index.html

Crossman, A. (2017). *Understanding Purposive Sampling*. Retrieved from Understanding Purposive Sampling: https://www.thoughtco.com/purposive-sampling-3026727

Cynet. (2018). *Advanced Threat Protection.* Retrieved from Advanced Threat Protection: https://go.cynet.com/resources-white-paper-advanced-threat-protection-beyond-the-av

Franciscus, D. (2018, April). *Pros and Cons of an IT Managed Service Provider for a Small Business.* Retrieved from Business News Daily: https://www.businessnewsdaily.com/5115-managed-services.html

Gasiorowski-Denis, E. (2016, December 16). *How to measure the effectiveness of information security*. Retrieved from How to measure the effectiveness of information security: https://www.iso.org/news/2016/12/Ref2151.html

Ghanem, K., Aparicio-Navarro, G, K., & Lambotharan, S. (n.d.). Support Vector Machine for Network Intrusion and Cyber-Attack Detection. *IEEE*, 2017.

Ghorbani, A. A., Lu, W., & Tavallaee, M. (2010). NETWORK INTRUSION DETECTION AND PREVENTION: CONCEPTS AND TECHNIQUES (ADVANCES IN INFORMATION SECURITY). *Theoretical Foundation of Detection Network Intrusion Detection and Prevention". Concepts and Techniques Advances in Information Security. Springer Science, Vol.47.*, 47- 114.

Gupta, P. (2017). *Decision Trees in Machine Learning*. Retrieved from Decision Trees in Machine Learning: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

Hao Sun, X. W. (2017). CloudEyes: Cloud-based malware detection with reversible sketch for resource-constrained internet of things (IoT) devices. *CloudEyes: Cloud-based malware detection with reversible sketch for resource-constrained internet of things (IoT) devices*.

Hoak, J. (2010). *The Effects of Outliers on Support Vector Machines.* Retrieved from The Effects of Outliers on Support Vector Machines: http://www.common-index.com/files/AML_Project.pdf

IBM. (2012). *How SVM Works*. Retrieved from How SVM Works: https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/svm_howwork.htm

INFOSEC. (2013). *Introduction to Secure Software Development Life Cycle*. Retrieved from Introduction to Secure Software Development Life Cycle: https://resources.infosecinstitute.com/intro-secure-software-development-life-cycle

INFOSEC. (2013). *Introduction to Secure Software Development Life Cycle*. Retrieved from Introduction to Secure Software Development Life Cycle: https://resources.infosecinstitute.com/intro-secure-software-development-life-cycle/#gref

Infosec Institute. (2018). *Common Continuous Monitoring (CM) Challenges*. Retrieved from Common Continuous Monitoring (CM) Challenges: https://resources.infosecinstitute.com/common-continuous-monitoring-cm-challenges/#gref

Infosec Institute. (2018). *Common Continuous Monitoring (CM) Challenges*. Retrieved from Common Continuous Monitoring (CM) Challenges: https://resources.infosecinstitute.com/common-continuous-monitoring-cm-challenges/#gref

ISC. (2018). *Cybersecurity Professionals Focus on Developing New Skills as Workforce Gap Widens.* ISC.

Ismail, N. (2018, February). *Securing the future: The evolution of cyber security in the wake of digitalisation*. Retrieved from Securing the future: The evolution of cyber security in the wake of digitalisation: https://www.information-age.com/evolution-cyber-security-wake-digitalisation-123470747/

ISO. (2013). *Information technology -- Security techniques -- Code of practice for information security controls*. Retrieved from Information technology -- Security techniques -- Code of practice for information security controls: https://www.iso.org/standard/54533.html

ISO. (2018). *The ISO/IEC 27000 family of standards*. Retrieved from The ISO/IEC 27000 family of standards: https://www.iso.org/isoiec-27001-information-security.html

Jidiga1, G. R., & Sammulal, D. P. (2013). MACHINE LEARNING APPROACH TO ANOMALY DETECTION IN CYBER SECURITY WITH A CASE STUDY OF SPAMMING ATTACK. *International Journal of Computer Engineering and Technology (IJCET)*, 113-122.

Johansen, K., & Lee, S. (2003). CS424 Network Security: Bayesian Network Intrusion Detection.

Kaimba, E. (2017). African Cyber Security Report. *Demystifying Africa's Cyber Security Poverty Line*.

Karanja, M. (2011, December). Retrieved from https://www.capitalfm.co.ke/business/2011/12/it-good-for-saccos-says-expert/

Kelley Dempsey, A. J. (2011). *Information Security Continuous Monitoring (ISCM) for Federal Information.* Retrieved from Information Security Continuous Monitoring (ISCM) for Federal Information: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-137.pdf

Kruegel, V., & Mutz, C. (n.d.). Bayesian Event Classification for Intrusion Detection.

Kumar, B. (2011). A Study on ISO 9001 Quality Management System Certifications – Reasons behind the Failure of ISO Certified. *Global Journal of Management and Business Research*, 43-50.

Kumar, C. (2017). *Interesting Ways to Watch Cyberattack in Real-time Worldwide*. Retrieved from Interesting Ways to Watch Cyberattack in Real-time Worldwide: https://geekflare.com/real-time-cyber-attacks/

Kumari, U., & Soni, U. (2017). A review of intrusion detection using anomaly-based detection. *2017 2nd International Conference on Communication and Electronics Systems (ICCES).*

KUSSCO. (2018). Retrieved from http://www.kuscco.com/

Luu, T. (2016, November). *https://www.isaca.org/Journal/archives/2015/Volume-1/Pages/Implementing-an-Information-Security-Continuous-Monitoring-Solution.aspx*. Retrieved from https://www.isaca.org/Journal/archives/2015/Volume-1/Pages/Implementing-an-Information-Security-Continuous-Monitoring-Solution.aspx: https://www.isaca.org/Journal/archives/2015/Volume-1/Pages/Implementing-an-Information-Security-Continuous-Monitoring-Solution.aspx

Malhotra, P., Vig, L., & Shroff, G. (n.d.). Long Short Term Memory Networks for Anomaly Detection in Time Series.

Mathew Schultz, E. S. (n.d.). Data Mining Methods for Detection of New Malicious Executables. *Data Mining Methods for Detection of New Malicious Executables*, 1-12.

Middleton-Leal, M. (2018). *How to make your security monitoring and data discovery processes GDPR-Compliant*. Retrieved from How to make your security monitoring and data discovery processes GDPR-Compliant: https://gdpr.report/news/2018/11/29/how-to-make-your-security-monitoring-and-data-discovery-processes-gdpr-compliant/

Mougoue, E. (2017). *SDLC 101: What Is the Secure Software Development Life Cycle?* Retrieved from SSDLC 101: What Is the Secure Software Development Life Cycle?: https://dzone.com/articles/ssdlc-101-what-is-the-secure-software-development

Munguti. (2013). *Information technology security threats in electronic Funds transfer in commercial banks in Kenya.* Retrieved from Information technology security threats in electronic Funds transfer in commercial banks in Kenya: http://erepository.uonbi.ac.ke/handle/11295/63027

MySql. (2017). *System Requirements*. Retrieved from MySql: https://dev.mysql.com/doc/workbench/en/wb-requirements.html

Naila Belhadj Aissa, M. G. (2016). Semi-Supervised Statistical Approach for Network Anomaly Detection. *Semi-Supervised Statistical Approach for Network Anomaly Detection*, 1090-1095.

Nepal, S., & Jang-Jaccard, J. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*.

NIST. (2019, April 23). Retrieved from https://www.nist.gov/cyberframework/new-framework

Noorossana, Hosseini, & Heydarzade. (2018). *An overview of dynamic anomaly detection in social networks via control charts.* Retrieved from An overview of dynamic anomaly detection in social networks via control charts: https://onlinelibrary.wiley.com/doi/full/10.1002/qre.2278

Nyawanga. (2015). *MEETING THE CHALLENGE OF CYBER THREATS IN EMERGING ELECTRONIC TRANSACTION TECHNOLOGIES IN THE KENYAN BANKING SECTOR.* Retrieved from MEETING THE CHALLENGE OF CYBER THREATS IN EMERGING ELECTRONIC TRANSACTION TECHNOLOGIES IN KENYAN BANKING SECTOR: http://erepository.uonbi.ac.ke:8080/bitstream/handle/11295/94405/Nyawanga%2c%20James%20O_Meeting%20the%20challenge%20of%20cyber%20threats%20in%20emerging%20electronic%20transaction%20technologies%20in%20in%20Kenyan%20banking%20sector.pdf?sequence=1&isAllo

Obura. (2018, December 19). *Cases of cyber-attacks in Kenya rise to 3.8 million*. Retrieved from Cases of cyber attack in Kenya rise to 3.8 million: https://www.standardmedia.co.ke/article/2001306810/cases-of-cyber-attack-in-kenya-rise-to-3-8-million

Omar, S., Ngadi, A., & Jebur, H. H. (2013). Machine Learning Techniques for Anomaly Detection: *International Journal of Computer Applications*.

OWASP. (2018, December). *OWASP Secure Software Development Lifecycle Project*. Retrieved from OWASP Secure Software Development Lifecycle Project: https://www.owasp.org/index.php/OWASP_Secure_Software_Development_Lifecycle_Project

Pandora FMS Enterprise. (2018). *Why choose Pandora FMS as a monitoring system?* Retrieved from Why choose Pandora FMS as a monitoring system?: https://pandorafms.com/monitoring-system/

Ponemon Institute. (2018, February). *Cybercrime Costs Financial-Services Sector More Than Any Other Industry*. Retrieved from Accenture: https://newsroom.accenture.com/news/cybercrime-costs-financial-services-sector-more-than-any-other-industry-with-breach-rate-tripling-over-past-five-years-according-to-report-from-accenture-and-ponemon-institute.htm

Quamar Niyaz, Sun, W., Javaid, A. Y., & Alam, M. (2013). A Deep Learning Approach for Network Intrusion Detection System.

Raiyn, J. (2014). A survey of Cyber Attack Detection Strategies. *International Journal of Security and Its Applications*.

Rashmikant, R. (2018). Cyber Security: Threat Detection Model based on Machine learning Algorithm. *Cyber Security: Threat Detection Model based on Machine learning Algorithm*.

Rouse, M. (2018). *Denial of Service Attack*. Retrieved from Denial of Service Attack: https://searchsecurity.techtarget.com/definition/denial-of-service

Serianu. (2018). *Sacco Cyber Security Report*. Retrieved from https://www.serianu.com/downloads/SaccoCyberSecurityReport2018.pdf

Shyu, & Chen. (n.d.). A Novel Anomaly Detection Scheme Based on PCA Classifier," in IEEE Foundations and New Directions of DataMining Workshop. 172-179.

Soiri, H. (2018). A state-of-the-art survey of malware detection approaches using data mining techniques. *Human-centric Computing and Information Sciences*.

Song, J., Takakura, H., Okabe, Y., & Nakao, K. (2013). Toward a more practical unsupervised anomaly detection system. *Information Sciences—Informatics and Computer Science, Intelligent Systems, Applications*.

Soni, D. (2018). *Introduction to Bayesian Networks*. Retrieved from Introduction to Bayesian Networks: https://towardsdatascience.com/introduction-to-bayesian-networks-81031eeed94e

Sosnovshchenko, A. (2017). *Understanding the KNN algorithm*. Retrieved from Understanding the KNN algorithm: https://www.oreilly.com/library/view/machine-learning-with/9781787121515/ea041441-d69c-42a6-ac04-03aa9fd3cab0.xhtml

Tan, X., & Xi, H. (n.d.). Hidden semi-Markov model for anomaly detection. 91-94.

Tandon, G. (2018). *Machine Learning for Host-based Anomaly*. Retrieved from Machine Learning for Host-based Anomaly: https://pdfs.semanticscholar.org/4cb5/d2d8da48c9960f05594de59c06ff85b55176.pdf

Tariq, N. (2018). IMPACT OF CYBERATTACKS ON FINANCIAL INSTITUTIONS. *Journal of Internet Banking and Commerce*.

Techopedia. (2018). Retrieved from https://www.techopedia.com/definition/9269/scalability

Techopedia. (2018). *Botnet Attack*. Retrieved from Botnet Attack: https://www.techopedia.com/definition/29948/botnet-attack

Teoh, C. S., Mahmood, A. K., & Dzazali, S. (2018). Cyber Security Challenges in Organisations: A Case Study in Malaysia. *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*.

Trend Micro. (2018, October). *Bridging Cybersecurity Gaps with Managed Detection and Response*. Retrieved from Bridging Cybersecurity Gaps with Managed Detection and

Response: https://www.trendmicro.com/vinfo/hk-en/security/news/security-technology/bridging-cybersecurity-gaps-with-managed-detection-and-response

Valdes, A., & Skinner, K. (2000). Adaptive, Model-based Monitoring for Cyber Attack Detection. *Springer-Verlag Berlin Heidelberg 2000*.

VECTRA. (2018). *A new threat detection model-new threat detection model.* Retrieved from A new threat detection model: https://www.ontrex.ch/fileadmin/user_upload/www.ontrex.ch/docs/portfolio/enterprise_security/Vectra/Vectra-Ebook-Detection-Model.pdf

Waldo, J. (2006). *On System Design. Burlington, MA: Sun Microsystems, Inc.* Retrieved from On System Design. Burlington, MA: Sun Microsystems, Inc: https://dl.acm.org/citation.cfm?id=1167513

Walters, A. G. (2019, Jan). Retrieved from https://austingwalters.com/classify-sentences-via-a-recurrent-neural-network-lstm/

Wechulia, Wabwoba, F., & Wasike, J. (2017). Cyber Security Challenges to Mobile Banking in SACCOs in Kenya. *Cyber Security Challenges to Mobile Banking in SACCOs in Kenya*, 133-140.

Wekundah. (2015). *The Effects of Cyber-crime on E-commerce; a model for SMEs in Kenya*. Retrieved from The Effects of Cyber-crime on E-commerce; a model for SMEs in Kenya: http://erepository.uonbi.ac.ke/bitstream/handle/11295/95232/Wekunda_%20The%20Effects%20of%20Cybercrime%20on%20E-Commerce%20a%20Model%20for%20SMEs%20in%20Kenya.pdf?sequence=1&isAllowed=y

Wu, X. (2017). *Metrics, Techniques and Tools of Anomaly.* Retrieved from https://www.cse.wustl.edu/~jain/cse567-17/ftp/mttad/index.html

Yin, C. (2017). *A Deep Learning Approach for Intrusion*. Retrieved from A Deep Learning Approach for Intrusion: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8066291

Yin, R. K. (2003). Case Study Research Design and Methods. *Applied Social Research Methods Series, no. 5. 3rd ed. Thousand Oaks, CA.*

Yiu, T. (2017). *Understanding Neural Networks*. Retrieved from Understanding Neural Networks: https://towardsdatascience.com/understanding-neural-networks-19020b758230

Zakka, K. (2016, July 13). *A Complete Guide to K-Nearest-Neighbors with Applications in Python and R*. Retrieved from A Complete Guide to K-Nearest-Neighbors with Applications in Python and R: https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Appendix A: Loading Training Dataset

Loading Training Dataset

In [2]:
```
import pandas as pd
train = pd.read_csv('/Users/mt/Desktop/ANN/train.csv')
```

In [3]:
```
train.head()
```

Out[3]:

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mtl
0	1	0.000011	udp	-	INT	2	0	496	0	90909.0902	...	1	2	0	0	
1	2	0.000008	udp	-	INT	2	0	1762	0	125000.0003	...	1	2	0	0	
2	3	0.000005	udp	-	INT	2	0	1068	0	200000.0051	...	1	3	0	0	
3	4	0.000006	udp	-	INT	2	0	900	0	166666.6608	...	1	3	0	0	
4	5	0.000010	udp	-	INT	2	0	2126	0	100000.0025	...	1	3	0	0	

5 rows × 45 columns

Figure A: Loading Training Dataset Code

Appendix B: Loading Features Dataset

Loading Feautures Data Set

```
In [4]: 1 features = pd.read_csv('/Users/mt/Desktop/ANN/features.csv', sep=",", encoding='mac_roman')

In [5]: 1 features.head()
```

Out[5]:

	No.	Name	Type	Description
0	1	srcip	nominal	Source IP address
1	2	sport	integer	Source port number
2	3	dstip	nominal	Destination IP address
3	4	dsport	integer	Destination port number
4	5	proto	nominal	Transaction protocol

```
In [6]: 1 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 82332 entries, 0 to 82331
Data columns (total 45 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 82332 non-null  int64
 1   dur                82332 non-null  float64
 2   proto              82332 non-null  object
 3   service            82332 non-null  object
 4   state              82332 non-null  object
 5   spkts              82332 non-null  int64
 6   dpkts              82332 non-null  int64
 7   sbytes             82332 non-null  int64
 8   dbytes             82332 non-null  int64
 9   rate               82332 non-null  float64
 10  sttl               82332 non-null  int64
 11  dttl               82332 non-null  int64
```

Figure B.1 Loading Features Dataset Code

import library for feature selection

```
In [62]: 1 from sklearn.feature_selection import SelectKBest
         2 from sklearn.feature_selection import f_classif

In [63]: 1 best_features = SelectKBest(score_func = f_classif,k=9)
         2 fit = best_features.fit(X,y)
         3 df_scores = pd.DataFrame(fit.scores_)
         4 df_columns = pd.DataFrame(X.columns)

In [64]: 1 features_scores = pd.concat([df_columns,df_scores],axis =1)
         2 features_scores.columns = ['Feature_Name','Score']

In [65]: 1 print(features_scores.nlargest(9,'Score'))
```

```
       Feature_Name         Score
7              sttl  18964.086751
32  ct_dst_sport_ltm  16135.278766
31  ct_src_dport_ltm  12505.863410
1             proto  11096.945966
17             swin  11087.382800
38       ct_srv_dst   8354.924821
28       ct_srv_src   8199.287505
33   ct_dst_src_ltm   8126.081994
20             dwin   7960.170077
```

```
In [66]: 1 new_features = features_scores.nlargest(9,'Score')
         2 new_features
```

Figure B.2 Importing Libraries for Feature Selection

Appendix C: Feature Scaling and Data Splitting

Scaling the feauteres

```
In [71]:
1 X = (scaler.transform(X))
2 X
```

```
Out[71]: array([[ 0.77099348, -0.44259086, -0.47371476, ..., -0.67590125,
         -0.56920018,  0.86647113],
        [ 0.77099348, -0.44259086, -0.47371476, ..., -0.67590125,
         -0.56920018,  0.86647113],
        [ 0.77099348, -0.44259086, -0.47371476, ..., -0.76190497,
         -0.56920018,  0.86647113],
        ...,
        [ 0.77099348, -0.44259086, -0.47371476, ..., -0.76190497,
         -0.56920018, -1.15425294],
        [ 0.77099348, -0.44259086, -0.47371476, ..., -0.76190497,
         -0.48605722, -1.15425294],
        [ 0.77099348, -0.44259086, -0.47371476, ..., -0.76190497,
         -0.56920018, -1.15425294]])
```

```
In [72]:
1 X=np.asarray(X).astype(np.float32)
2 y=np.asarray(y).astype(np.float32)
```

Data splitting

```
In [74]:
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state =0)
3 X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=123)
4
```

Figure C.1 Features Scaling and Data Splitting

Appendix D: RNN Model Creation

Defining RNN Model

In [75]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

In [76]:
```
model = keras.Sequential()
model.add(layers.Dense(12, input_dim  = 9, activation="relu"))
model.add(layers.Dense(10, activation ='relu'))
model.add(layers.Dense(8,  activation = 'relu'))
model.add(layers.Dense(6,  activation = 'relu'))
model.add(layers.Dense(4,  activation = 'relu'))
model.add(layers.Dense(2,  activation = 'relu'))
model.add(layers.Dense(1,  activation ='sigmoid'))
```

In [77]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['acc'])
```

In [78]:
```
history = model.fit(X_train, y_train, epochs=50, batch_size=64,validation_data=(X_val, y_val))
```

```
Epoch 1/50
680/680 [==============================] - 3s 3ms/step - loss: 0.6128 - acc: 0.6942 - val_loss: 0.3738 - val_acc:
0.9127
Epoch 2/50
680/680 [==============================] - 2s 2ms/step - loss: 0.3572 - acc: 0.9081 - val_loss: 0.2977 - val_acc:
0.9153
Epoch 3/50
680/680 [==============================] - 2s 3ms/step - loss: 0.2984 - acc: 0.9132 - val_loss: 0.2642 - val_acc:
0.9197
Epoch 4/50
680/680 [==============================] - 2s 3ms/step - loss: 0.2678 - acc: 0.9157 - val_loss: 0.2531 - val_acc:
0.9196
Epoch 5/50
680/680 [==============================] - 3s 5ms/step - loss: 0.2456 - acc: 0.9209 - val_loss: 0.2268 - val_acc:
0.9237
Epoch 6/50
680/680 [==============================] - 3s 5ms/step - loss: 0.2350 - acc: 0.9238 - val_loss: 0.2222 - val_acc:
0.9287
Epoch 7/50
```

Figure D.1 RNN Model Development

Appendix E: RNN Model Accuracy and Score

Evaluating the accuracy and score of the model

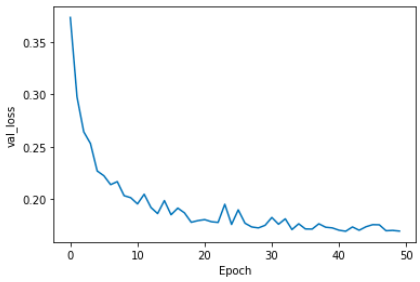
In [ ]:
```
score = model.evaluate(X_test, y_test, verbose =0)
print("Test Score: ", score[0])
print("Test Accuracy: ", score[1])
```

In [80]:
```
score = model.evaluate(X_train, y_train, verbose =0)
print("Training Score: ", score[0])
print("Validation Accuracy: ", score[1])
```

```
Training Score:  0.1725296676158905
Validation Accuracy:  0.9353437423706055
```

In [81]:
```
#the diff between predicted value and actual value
plt.plot(range(50), history.history['val_loss'])
plt.ylabel('val_loss')
plt.xlabel('Epoch')
```

Out[81]: `Text(0.5, 0, 'Epoch')`

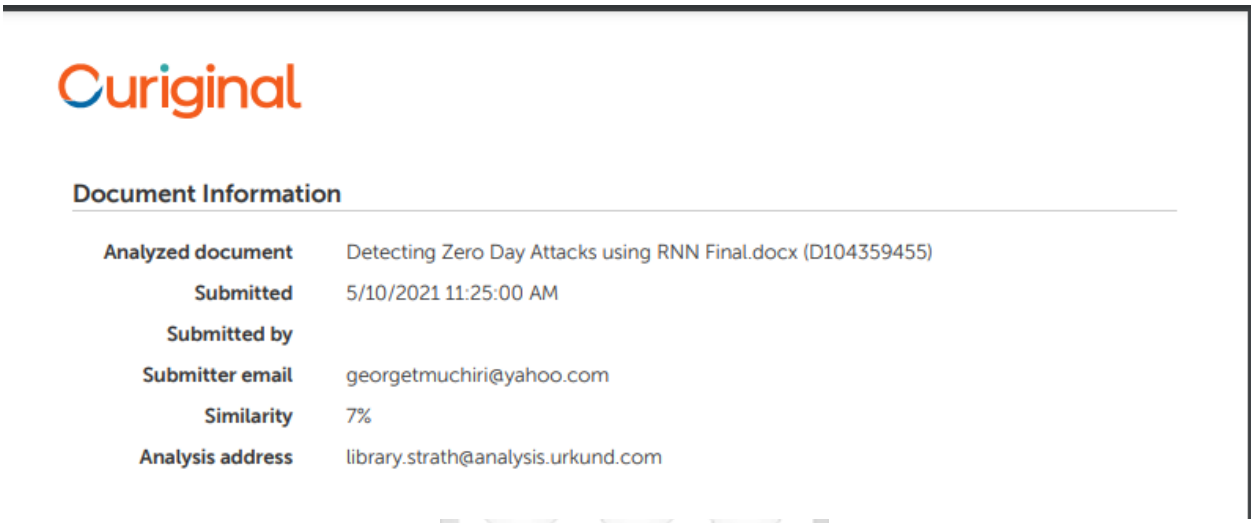


In [82]:
```
#Visualization of comparison between the training and validation loss

```

Figure E.1 RNN Model Accuracy and Score

Appendix F: Similarity Check Report



Curiginal

Document Information

Analyzed document	Detecting Zero Day Attacks using RNN Final.docx (D104359455)
Submitted	5/10/2021 11:25:00 AM
Submitted by	
Submitter email	georgetmuchiri@yahoo.com
Similarity	7%
Analysis address	library.strath@analysis.urkund.com

Figure F.1 Similarity Check Report

Appendix F: Research Ethics Review Report

Completion of Online Research Ethics Review Submission

You have successfully submitted your application for ethics review "Detecting zero day attacks using RNN"
Certificate awarded to: Mr Muchiri, George
Reference number: SU-IERC1090/21
Date and Time: 2021-06-16 16:27:07