東海大學

資訊工程研究所

碩士論文

指導教授: 楊朝棟博士

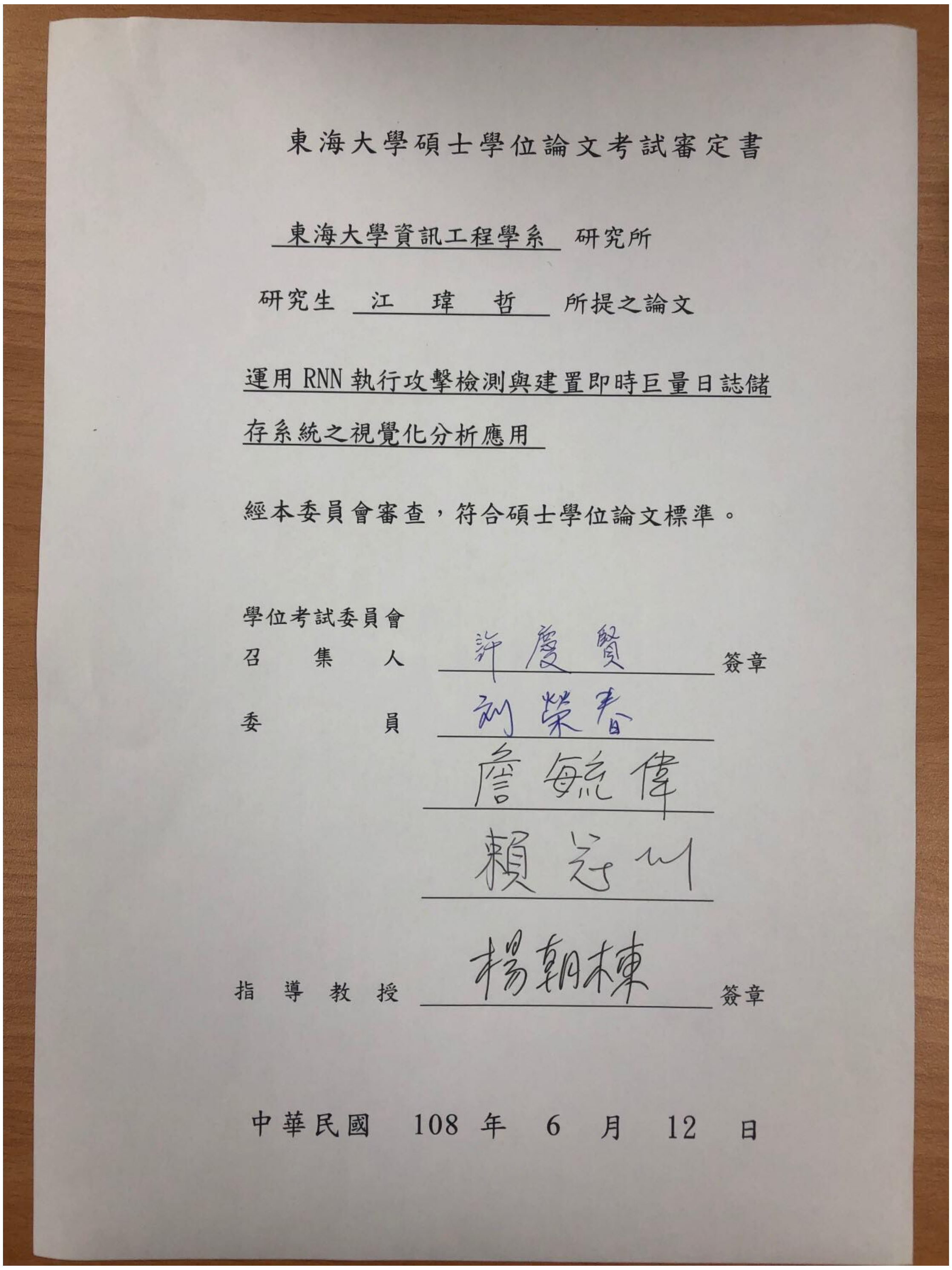
共同指導教授: 劉榮春博士

運用 RNN 執行攻擊檢測與建置即時巨量日誌儲存系統之視覺化分析應用

Using RNN for Cyberattack Detection in a Network Log System with Data Visualization

研究生: 江瑋哲

中華民國一零八年六月



東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 江 瑋 哲 所提之論文

運用 RNN 執行攻擊檢測與建置即時巨量日誌儲存系統之視覺化分析應用

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 許慶賢 簽章

委 員 劉榮春

詹毓偉

賴冠州

指 導 教 授 楊朝棟 簽章

中華民國 108 年 6 月 12 日

摘　要

近年來，資訊安全的問題討論度越來越高，從 OpenSSL Heartbleed 漏洞、美國摩根銀行資訊系統遭駭客入侵、GitHub 遭遇到的 DDoS 的威脅。種種的資訊攻擊事件都透露出雲端資訊安全的重要性已經是現今所不可忽視的議題。正常情況下，每個有使用網路的地方就有其網路日誌資料，而日誌資料對於網路管理人員是非常重要的數據。網路日誌資料包含著種種因素，例如系統錯誤、攻擊警告、流量大小、訊息傳送情形等等。本文的目的是提供一個網路日誌管理系統，可對於各類型的用戶做更進一步的視覺化分析。本系統使用 ELK Stack 技術，資料分析部分則是根據需要的分析目的而去分別對日誌資料做過濾、篩選、分析之類的處理，最後在視覺化呈現在 Web 瀏覽器上。系統運行的服務上主要是分別為 Elasticsearch、Logstash 與 Kibana，藉由數據蒐集、過濾處理與資料分析視覺化的功能，來提供一個網路日誌管理與視覺化分析之服務系統。網路攻擊檢測部分則是採用深度學習模型進行學習訓練，讓模型可以藉由已知的網路攻擊特徵來學習每種攻擊的特徵，然後在跟日誌系統上的分析資訊進行交叉比對，以達到驗證的效果。本文的最終目標是運用視覺化分析呈現各種客製化的 Network Log 相關圖形，並運用校內計算機中心相關資源，分別過濾出重要的網路資訊，例如來源地理位置與網路攻擊相關行為，都有在文內作成果展示，深度學習方面則是運用 RNN 模型對攻擊行為的分類，利用不同的模型進行訓練與測試比較，包含 DNN、LSTM，找出哪一種模型較適合本文的實驗數據。而分類出攻擊行為種類希望能有不錯的準確性，使其能夠跟 ELK Stack 運用相關特徵過濾得來的網路攻擊資訊做交叉比對，讓資訊正確性更為提高。

關鍵字: 網路日誌資料、Elasticsearch、Logstash、Kibana、深度學習、RNN

Abstract

In recent years, information security issues have become more and more discussed, from the OpenSSL Heartbleed vulnerability, the hacking of the US Morgan Bank information system, and the DDoS threats GitHub encountered. The purpose of this paper is to provide a network log management system that allows for further visual analysis of all types of users. The system uses ELK Stack technology, and the data analysis part is to filter, analyze and analyze the log data according to the analysis purpose required, and finally visually present it on the web browser. The services of the system are mainly Elasticsearch, Logstash and Kibana, which provide a network log management and visual analysis service.The network attack detection part uses the deep learning model for learning and training, so that the model can learn the characteristics of each attack by known network attack features. The ultimate goal of this paper is to use visual analysis to present various customized Network Log related graphics, and use the relevant resources of the school computer center to filter out important network information, such as source location and cyber attack related behavior. In the paper, the results of deep learning are the classification of attack behavior using RNN model. Different models are used for training and testing comparison, including DNN and LSTM, to find out which model is more suitable for the experimental data in this paper.

Keywords: Network Log Data, Elasticsearch, Logstash, Kibana, Deep Learning, RNN

致謝詞

能夠完成畢業論文必須感謝很多人，首先謝謝我的指導教授楊朝棟博士，從大三的畢業專題，到大四開始進入 HPC 實驗室，以後最後的研究所時期，老師都對我非常的照顧，藉由不同的訓練與工作事務讓我能更快速的累積自己的實作經驗與處事的應對，謝謝老師在學校的指導，相信這段求學經歷對我的未來的人生會有很大的幫助。

謝謝口試委員許慶賢教授、詹毓偉教授、賴冠州教授、劉榮春教授在百忙之中抽空參加我的論文口試，每個教授提出的寶貴意見都能使我的碩士論文更加具有完整性。另外在碩士的在學期間，時常受到實驗室的學長姐、學弟們的幫助，也非常感謝他們的付出。特別謝謝從大學到研究所的夥伴們，因為有你們才讓我的求學過能更精采與順利。

最後謝謝我的家人，爸爸、媽媽從小對我的教育與栽培，求學期間的支持與鼓勵，都是為了讓我成為一個更好的人，非常謝謝他們，讓我能夠更順利的完成碩士學位，由衷感謝一路上幫助過我的所有人，謝謝你們。

東海大學資訊工程學系 高效能計算實驗室 江瑋哲 一零八年六月

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the past, European Union computers were invaded by botnets, which caused about 3 billions NT dollars in losses. According to statistics from a well-known Zi'an company, the public was more worried about cybercrime than other crimes. Cybercrime accounted for 34%, which was obviously higher than robbery 25% , theft 22%and violent crime 19%. All the signs showed that the development of the Internet also accompanied the problems of cybersecurity. Consortium and enterprise also pay more and more attention to the issue of network security.

In recent years, network attacks have become more advanced. Decentralized Blocking Service (DDoS) attacks are the focus of attention in 2016. Mirai malicious programs are the accomplices behind them. This malicious program converts the victim device into a botnet capable of launching DDoS attacks, while Linux and UNIX systems, that is to say, many Internet of Things (IoT) devices today, are likely to be victims. However, the most notable victim is Dyn, a DNS service manufacturer, whose well-known customers'websites, such as Twitter, Reddit, Spotify and SoundCloud, are not accessible, and the economic losses are astronomical.

Therefore, in this dissertation, our goal is to implement a network log storage and analysis system, and use Deep Laering module to train and detect network attacks, so that managers can grasp more accurate network threat information at any time when analyzing network logs, and can assess the risk of logs and deal with them early.

The main steps of this paper are as follows:

1. Build a huge log management and analysis system to store and process a large amount of network data on the server in real time. Use Logstash to collect and convert network log data into corresponding format fields.

2. Use Elastic search to analyze log data according to field, and then let Kibana visualize the selected field data.

3. Collaborate with the School Computer Center to analyze more detailed information about the campus network, such as dormitories, departments, administrative units and so on, so as to make a clearer classification of the network log data.

4. Collect the historical network attack log data, train the Lucky Deep Learning model and learn the relevant features to detect the attack behavior.

1.1 Motivation

With the progress of science and technology, the problem of network attack is becoming more and more diverse, and the adverse reactions caused by information insecurity are paid more and more attention. As a result, the public pays more attention to the security issues, especially the leaders who want to get accurate, safe and trustworthy information so that they can plan effective countermeasures in advance. The purpose of this system is to provide a system that can monitor log data and analyze it, to provide continuous visual analysis, so that users can control the situation at any time. Attack detection is to use deep learning model to learn and train. Through continuous training, the machine can accurately identify different types of attacks, and the predicted attack detection can be achieved. Make analysis and comparison with the log system to ensure the credibility of the data.

The most vulnerable area of cyberattacks is usually the area with the most frequent network traffic. The university is the most common place to meet this condition, so this study aims to establish a network log management system that can be easily used by managers. By managing the network log data through the system, on the management interface, the required charts can be viewed according to the requirements, and the charts are drawn and sorted by the system, and the data sources are discussed through the teachers of the computer center. The solution is to use the method of regularly uploading the original log data to the database, so that the network log management system can maintain the source of the log data, and the results and abnormalities presented by the system can also be returned to the computer center for relevant response. The instructor also hopes to make corresponding strategies for the abnormal behavior of the network. Finally, this study uses Deep Learning to classify the abnormal behavior of the network to check with the log system to make the data more accurate.

1.2 Thesis Contributions

In this paper, a network log management and analysis system using ELK Stack is built. The system can store and analyze data, and display the results of visual analysis on Dashboard. In addition, the part of attack detection uses deep learning to build and train a model. After importing historical attack data, the model can learn and train. Finally, after the model reaches a certain degree of accuracy, the attack data can be classified as an attack type. Here are the contributions to this article:

1. Use ELK Stack to build a network log data management system, which can analyze and process network log data, present it to users in a more understandable visualization, and then filter out and visualize network attacks. This means that managers can use the system to observe and monitor cyber attacks to handle cyber attacks.

2. The Deep Learning model is used to train and classify attack events. The model can effectively help managers detect suspicious network resources or abnormal behavior in log data, and evaluate and track results based on the results provided to maintain good network security quality.

1.3 Thesis Organization

The structure of this paper is as follows. Chapter 2 mainly introduces the research background and related work. Chapter 3 will analyze how to make various visual analysis of network log data in ELK Stack log management system, as well as how to use Deep Learning to build models to learn to identify attacks in log data. Chapter 4 then details the implementation steps and discusses the evaluation. Finally, in Chapter 5, the conclusions and future prospects of this paper are presented.

Chapter 2

Background Review and Related Works

In the second chapter, this study provides three key points used in this paper: big data, ELK Stack system and Deep Learning. More details will be shown in the next introduction.

2.1 Big Data

Big Data means that the scale of data is so huge that it can not be stored, calculated and analyzed in a certain period of time through traditional methods. According to Matt Aslett, a scientist, he defines Big Data as ”data that was previously ignored because of technological limitations” and discusses data that could not be stored and analyzed before. The data volume of various industries in the world is growing at an alarming rate. It is estimated that the data volume will grow 10 times from 2013 to 2020, and the total amount of data will increase from 4.4 ZB to 44 ZB. Google handles more than 24 gigabytes of data a day, which means it handles thousands of times as much data as all paper publications in the National Library of America. Facebook handles more than 50 billion uploaded photos every day, and people click on the site and leave messages about billions of times every day.

The characteristics of Big Data:

- Volume data: Whether it's science, medicine, the Internet of Things, community interaction, communication, etc., a lot of data are being generated all the time.
- Variety data diversity: From the text, location, voice, image, picture, communication and other structured and unstructured all-encompassing information, they can interact with each other.
- Velocity data instantaneity: Big data also emphasizes the timeliness of data. With a large amount of data feedback generated every second, the data of the past three or four years has been useless.
- Veracity of data: Discuss whether there is any falsification in data collection, whether it can be recorded accurately, whether the data has abnormal values, and how to deal with abnormal situations.

2.2 ELK Stack

ELK Stack is the abbreviation of three open source software, namely, Elastic search, Logstash and Kibana. They are all open source software. In the process of log data processing, log analysis is usually needed, which means that you can search the log file directly to get the information you want. However, in large scale log system, this method is very inefficient. It needs centralized log management to collect and aggregate the logs. The common solution is to set up a centralized log collection system to collect, manage and access logs on all nodes.

Generally, large scale system is a distributed deployment architecture. Different service modules are deployed on different servers. When problems arise, most situations need to locate specific service areas according to the key information in the problem, and construct a centralized log system, which can improve the efficiency of positioning problems.

ELK provides a whole set of solutions, and all of them are open source software, which can be used in conjunction with each other perfectly and efficiently to meet the needs of many occasions. It is a mainstream log system at present.

The following are the characteristics of ELK Stack:

- Collection of log data from multiple sources
- Steady transmission of log data to central system
- Store log data
- Support Customization Analysis

2.2.1 Elasticsearch

Elasticsearch is a free data analysis and search system. It is an open source suite based on Lucene. It supports real-time search and analysis functions. It can be used for reading and writing data, feature filtering and data.

Elasticsearch has the characteristics of real-time search, stability, reliability, fast and easy to install and use.

Elasticsearch's search speed is very fast because it uses inverted index technology to find eligible document lists based on given values and field immediately. Elasticsearch has become one of the main choices of Machine Learning analysis or instant log data processing in recent years due to its high scalability , availability and excellent data analysis efficiency.

2.2.2 Logstash

Logstash is a free open source data system that receives, converts and outputs data from various sources. It supports more than 50 different types of input and output so that users can process various logs or data and define the required data fields according to different sources. Logstash is different from traditional log collection methods mainly in the way of collecting logs. Traditional versions must be written to files through the pipeline provided by internal programs. This method is feasible in a small number of servers, but checking log data in multi-servers can be very cumbersome. Logstash solves this problem. Its structured log greatly improves the convenience of checking log data. More importantly, there are many additional programs that can be output to various data sources for analysis and visual processing.

2.2.3 Kibana

Kibana is an Elasticsearch accessibility tool based on browser pages using the Apache open source protocol. Kibana is an application that uses dashboards as a foundation. Logically, Kibana is very simple, and most processors are at the panel level. Each panel is independent and completes separate data processing and visualization analysis. In addition, Kibana also offers a variety of input features, such as access logs - [YYYY. MM. DD], System Log - [YYYY. MM. DD] so that data from different indexes can be seen on the same panel.

Kibana can display various logs through visual analysis and then build various dashboards through the Elasticsearch search database. Kibana makes it easy to understand large amounts of data through a variety of visual effects and is easy to learn. The web browser-based interface enables beginners to quickly learn to create and share dynamic dashboards and directly display changes to flexible queries. For the filter, the new version of Kibana replaces the facet filter with the filter agg.

2.3 Python

Python and R are the two most important programming languages in the field of Deep Learning and Machine Learning. Python is simple and easy to learn, has a wide range of applications and a smooth learning curve. It is suitable for use as an entry-level programming language. Data analysis can be done through pandas, Numpy, sckikit-learning, Matplotlib and statsmodels. As R is a programming language developed by statisticians, it is good at statistical analysis and chart drawing. Python itself is a general language. Besides deep learning, Python can also be widely used in network development, website construction, game development, web crawler and other fields. When you need to integrate system products and services, you don't need to worry about the processing of different languages anymore. More importantly, Python can also be used as glue language which is very easy and has better performance. In short, Python is a simple, easy-to-learn but powerful programming language worth investing in, and this paper is also a study of Python on Deep Learning.

2.3.1 Anaconda

Anaconda is like a simple use package for Python, in addition to Python itself contains a suite of data analysis, machine learning, deep learning, visualization and computing that Python commonly uses. Anaconda is an open source and free software, but additional acceleration and optimization features are chargeable. Only academic use can apply for a free trial period. In addition, the supported platforms are Linux, Windows and Mac, which are almost large. A system that is often used by some people, and a very convenient feature, is that the Python version can be switched freely, and there is also a jupyter notebook environment in which the overall convenience is improved.

Important suites of Python:

- Matplotlib: basic visual tools, such as long bars, line charts and so on.
- Seaborn: Another visual tool, some users prefer their visual presentation.
- Numpy: Python is a must-have for multidimensional array (matrix) operations, with faster computing speeds than built-in features.

- Pandas: Pandas makes it easy for Python to do almost all of Excel's functions, such as field summing, filtering, analysis tables.
- SciKit-Learn: The Python machine learning model is basically in this suite.
- Jupyter notebook: A lightweight web-base tool for writing Python, which is very popular in the field of data analysis. Although the function is not as powerful as Pycharm and Spyder, it is very convenient to use Jupyter in addition to the case of using a large number of lines of code.

2.3.2 Keras

Keras is an open source code, Python-based Deep Learning library, developed mainly by Francois Chollet and other members of the open source community, authorized by MIT open source code. The main reason why Keras can operate quickly and conveniently is that it has sorted out the input layer, hidden layer and output layer of the training model, which can be used. Users only need to add and fill in the correct parameters, such as the number of neurons, activation function and so on.

The characteristics of Keras deep learning:

- The training model can be built up more quickly by building part of the neural network hierarchy.
- Through the back-end system (Theano, Tensorflow), it can run on CPU and GPU.
- Keras code is concise, easy to maintain and extensible.

2.4 Deep Learning

Deep Learning [1] [2] is a kind of "artificial neural network" which imitates the human neural network and makes such a network deeper. Neural network is an algorithm in the field of machine learning. Machine Learning is the most important technology to realize artificial intelligence.

In 2012, the winning team of ILSVRC, an International Conference on image recognition race, applied deep neuron-like network to make it far ahead of other teams in accuracy, thus making deep neuron-like network become the focus of attention, that is, Deep Learning. In addition, the application of Deep Learning is not only image, but also in various fields, including image generation, natural language generation, automatic translation, robot control and so on. In June 2016, AlphaGo, an artificial intelligence using Deep Learning, defeated the world's top chess players. In every field with human intelligence, it seems that there is a certain feasibility of Deep Learning [3] [4].

Machine Learning is divided into four branches: supervised learning, semi-supervised learning, unsupervised learning and reinforcement learning. Supervised learning is the most common type of machine learning. Supervised learning algorithm is based on a set of samples for prediction. For example, it can use historical prices to estimate future prices. The challenge of supervised learning is that the process of tagging data can be expensive and time-consuming. If the labels are limited, this study can use unlabeled samples to enhance supervised learning. In semi-supervised learning, this research uses both unlabeled data and a small amount of labeled data to improve learning accuracy. When unsupervised learning is performed, the machine obtains completely unlabeled data. Reinforcement learning is to let the model execute directly, and then feedback the results back for training.

2.4.1 DNN

Artificial intelligence is no longer a new topic, and even Deep Neural Networks (DNN) is also a ”old age” technology. In the past, DNN was limited by technological development. Although there were occasional signs of revival, it never lasted for too long. Since 2005, AI has received attention, but there has been no good improvement. It has been a sudden rise in 2012, and it has been the focus of research in the past two years. According to IEK estimates, the global automotive electronics output will reach $ 450 billion in 2023, which is 1.67 times higher than in 2015.

DNN (Deep Neural Networks) is a branch of Machine Learning. It mainly uses supervised or unsupervised learning as a way to train machines to improve the efficiency and accuracy of machine training. The difference between DNN and RNN and CNN is that DNN refers to the fully connected neuron structure, and does not contain convolution units or temporal associations. DNN will also have some problems in use. For example, the upper and lower neurons of a fully-connected DNN can form a connection with each other, which is easy to cause overfitting to lead to regional optimality.

2.4.2 RNN

Recursive Neural Network (RNN) [5] [6] [7] has a feature that the output of each layer in a multi-layer neural network is directly appended to the self-loop of the input. By this architecture, the input before the input of the layer can be memorized. When the input data is a continuous sequence, the input memory before the input can be incorporated into the thinking mode of the next input.

That is to say, the current output is affected not only by the input of the previous layer, but also by the output of the same layer (i.e. the previous one), similar to the statistical Time Series.

In addition, RNN can have many changes and applications:

- One to one: Input and output of fixed length, which is the general Neural Network model.
- One to many: Single input, multiple output, such as image captioning, input an image, want to detect multiple objects in the image, and give titles one by one, which is called ”Sequence output”.

- Many to one: Multiple inputs and single outputs, such as Sentiment Analysis, input a large paragraph to determine whether the sentence is a positive or negative emotional expression, which is called ”Sequence input”.
- Many to many: Multiple inputs and outputs, such as machine translation, input an English sentence and translate it into Chinese, which is called ”Sequence input and sequence output”.
- Another kind of many-to-many: Synchronize’s multiple inputs and outputs, such as Video Classification, input a movie, hoping to produce a title for each frame, which is called Synchronize input sequence and output.

2.4.3 LSTM

LSTM (Long Short-term Memory, LSTM) [8] [9] is a modified RNN, mainly to solve the problem of gradient disappearance and gradient explosion in the process of constant time series. In simple terms, LSTM can perform better in long-term sequence training than a normal RNN because LSTM solves the above-mentioned RNN [10] problem by adopting an improved memory management architecture.

- Forgetting the stage: The forgetting stage is mainly to selectively forget the input value of a node on the team, that is, forgetting the bad value, the choice is better.
- Selecting the memory stage: Selecting the memory stage mainly involves selective memory of the input values, more memory with higher importance, and less memory with lower importance.
- Output phase: The output phase will determine the output value of the current state.

2.5 Grafana

Grafana is an open source visualization tool that can be used on a variety of different materials, but Graphite and Elasticsearch are the most commonly used. Fundamentally, it's an upgraded version of Graphite-web that includes more convenient dashboard features, editing options for more options, and no additional monitoring due to different data sources. software. Users can create a variety of charts through Grafana. Compared to other monitoring software, Grafana has easier to configure configuration settings. These features can greatly reduce the burden on users.

2.6 Related Works

Raghav Rastogi et al. [11] used ELK Stack to build a log management system to filter network failures, information security and error messages. The log system of this article has a reference to his implementation. Pingkan P. I. Langi et al. [12] used an API from Twitter to get the data and uses two methods to enter data into Elasticsearch. The first way is through Twitter River, and the second way is through Logstash, which compares the evaluation of Twitter River and Logstash performance. This article refers to his experimental results and the method of entering data fields.

Gianluigi Ciocca et al. introduced a bigdata log analysis method based on recurrent neural network to predict the most likely future events. Converts the log (unstructured text data) of each component of bigdata to structured data. This article refers to the method of conversion of the data. Carela-Español et al. [13] studied the traffic classification problem of NetFlow data and uses the ML model for training. This article has reference to its method of traffic classification and filtering characteristics. Magdi S. et al. []MAHMOUD2019101 used the RNN model to train and use it for attack detection. This paper refers to the structure and training methods of its training model.

In recent year, the issue of information security [13] [14] has been highly concerned. From loopholes on OpenSSL, Heartbleed, cyberattack on JP Morgan Chase, GitHub threatened by Distributed Denial of Service attack, DDoS. Various cyber-attacks [15] [16] show the importance of storage safety, and this issue should not be overlooked nowadays. In 2018 World Economy Review, the cyber-attack has been ranked in second place. In normal circumstance, the network log data [17] is used whenever the internet is used. Network log data [18] is essential to web administrator, which provides data like system error, cyberattack warning, mobile data gigabytes, message sending status etc. The purpose of this thesis is to provide a network log data management system, which can do visualization analyse for each type of users. The system uses ELK Stack technology, accordingly filter, screen and analyse network log data base on different purpose. And finally apply visualization [19] [20] effects on web browser.

System service is mainly composed by Elasticsearch, Logstash and Kibana [21].The services of the system are mainly composed by Elasticsearch [22], Logstash and Kibana, which provide a network log management and visual analysis service by combining distributed search and analysis services, data collection, filtering processing and data analysis visualization.The network attack detection [23] [24] [25] part uses the deep learning model for learning and training, so that the model can learn the characteristics of each attack by known network attack features [26], and then cross-match the analysis information on the log system to Achieve the effect of verification.

There is also a related study that uses the LSTM model to predict the impact of weather variables on the weather. The prediction model proposed in the paper is an extension of the LSTM model. By adding the intermediate variable signal to the LSTM storage area, it is easier for the model to learn and identify the patterns in the training data set. The architecture also explores various architectures, including single-layer LSTM and multi-layer LSTM. The data set is weather variable data collected by Weather Underground at the airport in Indonesia. At this time, the better model was built in the multi-layer LSTM model, and the experimental results obtained a verification accuracy of 0.8. This study allowed me to recognize the different types of LSTM operations, allowing my experiment to have other attempts.

Chapter 3

System Design and Implementation

This chapter will introduce how to use ELK Stack to visualize the analysis of network log data and implementation, as well as show the deep learning detection model. The network log collected in this paper is a machine from the computer center in the school. Every day, there are more than 8 million pieces of data. According to the amount of data during the school period, the single piece of data is about 2 to 3 G. At present, it has accumulated to 6 TB, and will upgrade the relevant equipment level according to the hardware situation in the future.

3.1 System Architecture

In this article, the study will first introduce how to deploy the entire ELK Stack system, then use the log data of the computer center to import and write the configuration file so that it can read the corresponding log data field for the system to visually analyze. And the paper discusses how to use ELK Stack to build a network log system to make a variety of visual analysis of the campus, and use deep learning model to detect attacks, to assist the reliability of the network log system in information security. The basic environment of the network log system is to build a set of ELK Stack system on the server first, and add related kits to assist, combined with the network resources of the campus computer center, and finally present a variety of visual analysis to provide managers and visitors with a clearer understanding of the campus network information.

The other part is to install Anaconda3 on Windows 10 and use Juypter Notebook as the development environment of Python, then preprocess the log data, then import the deep learning model to train and learn, and detect the attack behavior of other network log data.This section will use different types of deep learning to make comparisons, and refer to the classification accuracy of the experimental data to select a better model.

Finally, the Grafana monitors the performance of Elasticsearch. After the log system is deployed, as the data volume accumulates, the system performance needs to be mastered at any time. Grafana can instantly monitor the data traffic and current performance usage on the system.

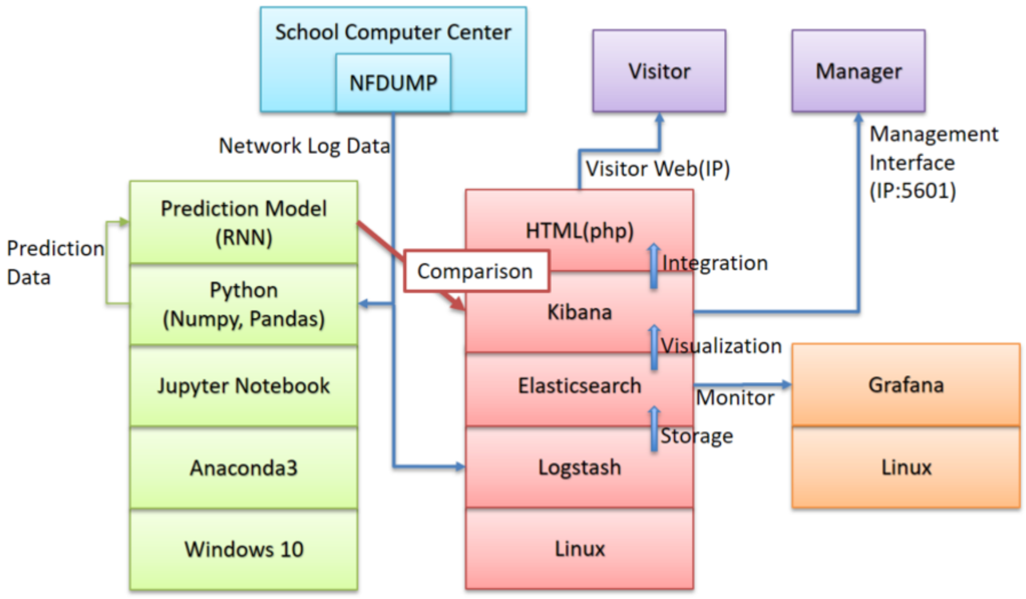


FIGURE 3.1: System architecture

3.2 NetFlow Log System

First, this paper use a shell script to download files from the server, which collects NetFlow log data to the local computer. Then,the paper use ELK Stack for preliminary analysis, Logstash will continue to collect and filter log data, and do format processing. Elastic search stores data passed in by Logstash. Finally, this paper will use Kibana to visualize the log data on the website.

3.2.1 Visualization Analysis of Network Usage

Network log data are constantly updated. If managers want to get the simplest and most understandable information as quickly as possible, the focus is on the application of visual analysis. In the campus, each building has its own network domain, in order to facilitate the management of the network usage of each building, it is necessary to filter out the network domain of each building separately. This paper uses the most commonly used flow analysis to visualize the flow of each building. There are two important elements in the network flow, that is, source and destination. The analysis of IP usage profiles between source and destination presents the relationship charts of IP, port and proto, which makes it easier for administrators to view their relevant information. Geographical location information can also make visual heat maps that can view the frequency of network usage from around the world and locally at once.

3.2.2 Visualization Analysis of Attack Detection

Part of the cyberattack behavior is to use the different characteristics of each attack behavior, and use the filtering method to extract the abnormal data for visual analysis. Features like CodeRed and Worm are relatively fixed values and can be easily analyzed. The characteristics of DDOS are relatively unfixed, so a feature range is set for analysis processing.

3.3 Deep Learning models

In this chapter, the paper will discuss how to use Deep Learning model training and demonstrate the accuracy of cyber attack classification for web log data. This paper experiments and tests the cyberattack behavior of the school with a large amount of data. Because the network log data lacks network attack data, the data source uses the CodeRed, Nimda and Worm attacks with more data to perform the training test.

In the experiment, Keras was used to perform the experiment and test of the model. The Python version used version 3.6, then used Anaconda3's jupyter notebook suite to write Python, and finally compared the accuracy of different models' attack behavior classification.

3.3.1 Network Log Data Preprocessing

As follow Algorithm1, the log data is preprocessed to convert the data into a format that the deep learning model can learn. Then extract the data with network attack behavior, and then classify according to its behavior characteristics, and prepare enough data to make the training and classification accuracy more reliable.

In the case of multiple classifications, in order to make training more effective, try to make the number of different network attacks more average. Our training set is collected from log data at different times, so the model does not learn the same data at the same time. The exact green color of the completed model will be shown in the following sections.

3.3.2 DNN model Training and Prediction

The part of the DNN model is to use the supervised learning method to train. When the preset parameters have not been adjusted at the beginning, it is easy to encounter the overfitting situation, and the result of the test set is not as

Algorithm 1 Data Preprocessing

```
Input: Network Log Source , $NFDump.txt$;
Output: Training, Test Dataset of models;
1: Read file with Python; Dataset = $NFDump.txt$
2: Remove spaces and unnecessary data;
3: Mark three types of attack data;
4: if This is the data with the attack data, Mark as 1, 2 and 3 according to
   different attack types then Delete unmarked data
5: else(There are no attack data in Dataset)
6:     Delete Dataset
7: end if
8: Save file with Python
```

expected. Then, by adjusting the learning rate, neurons and optimizer, it can slowly improve the situation, and the number of training sets can be improved. Finally, the accuracy of the verification set can reach an accuracy of 99.98

In the aspect of the test set, four sets of test data were extracted from different dates and times, and the accuracy rates were 98.88%, 99.97%, 99.47%and 99.91% , respectively, and the average effect could reach 99%or more, representing that the model has Quite high accuracy.

3.3.3 RNN model Training and Prediction

The part of RNN model is also trained by supervised learning. In order to compare the differences of different neural network models, the training set also uses the same training set as other models. In the process of training, the model of RNN is obviously better than DNN. It is easier to achieve high accuracy. The optimizer is also used in the same way as DNN. There is almost no overfitting in the training process. The test set is also using the same four sets of data as DNN, which are from different dates. Time, in order to understand the difference in accuracy more clearly. In the aspect of the test set, four sets of test data were extracted from different dates and times, and the accuracy rates were 100%, 99.98%, 100%and 99.98%, respectively, and the average effect could be close to 100%.

3.3.4 LSTM model Training and Prediction

In the part of the LSTM model, training is also carried out using supervised learning. The training effect is very close to the RNN model. It may be that the long-term sequence is not obvious to the training set, and the RNN test result is close to 100%, making it difficult to distinguish the RNN. The difference between the model and the LSTM model, the accuracy of the verification set is very close to the RNN model.

Finally, the test set is also based on the same four sets of data as the other models, the test results are 100%, 99.98%, 100%and 99.98%, respectively, the same as the RNN model.

Chapter 4

Experimental Results

In this chapter, the study will describe which parts of the log data are analyzed in the web log management system. First, the system operation process is explained. The study uses the Nfdump of the computer center as the data source of the log data, and use the data periodically uploaded to the database to complete the original log data of our network log system. The original log data will be advanced to Logstash. The read and format fields are processed, and then sent to Elasticsearch for storage and waiting for Kibana to send data request signals. After receiving the signal, the corresponding log data is sent to Kibana, and finally to Kibana. The above is a scientific analysis.

The system monitoring part uses another machine to set up the Grafana monitoring system on the machine. It is responsible for linking to Elasticsearch on the log system. Elasticsearch will return the system information in time, including data traffic and system usage. These information can be used by users. In order to get a quick grasp of system usage.

Finally, the model comparison part. After training and testing of the three models, RNN and LSTM are more suitable training models. In terms of classification, all three models can achieve high accuracy, but comprehensively evaluate training time and execution time. And other conditions, the model of the RNN type is considered to be the optimal model for the current data type, and the detailed results of the comparison will be presented below.

4.1 Hardware Environment

This section describes our hardware experimental environment. This paper uses two hosts, one is Linux, as the basic environment of ELK Stack. The other one uses Windows 10 as the operating system and installs Anaconda 3 and Python environment as the model training environment. Hardware devices are shown in Table 4.1.

TABLE 4.1: Hardware specifications

Item	Disk	Core	Ram	OS
Network Log system	8TB	Intel(R)Core(TM) i7-6950X CPU @ 3.00GHz	128G	Ubuntu 18.04.02
Deep Learning	1TB	Intel(R)Core(TM) i7-6700 CPU @ 3.40GHz	16G	Windows 10
Grafana	1TB	Intel(R)Core(TM) i7-6700 CPU @ 3.40GHz	16G	Windows 10

4.2 Experimental Results

4.2.1 Visualization of the Network Log System

Visual analysis of log data is based on different types of needs to make corresponding chart presentations, such as student dormitory, is the most frequent area of traffic anomalies, and there are many reasons for abnormal traffic, computer poisoning, hacker intrusion and the use of plug-in software can lead to abnormal traffic, which can be clearly found through the log data.

In the following results, the meanings and causes of various visualizations will be explained, and detailed explanations and operations will be made separately.

Figure 4.1 tells us an overview of the source and destination IP. The larger the font, the higher the number of connections in the total proportion.This part is visualized using source address and destination address.

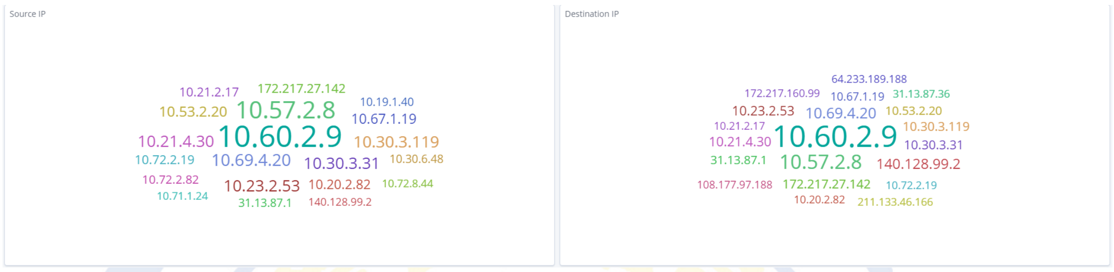


FIGURE 4.1: Source address and destination address

4.2 shows the number of connections between the source address and the destination address, so that the administrator can clearly see that the previous high number of connections is those IP, when abnormal circumstances occur, you can also refer to this figure as a basis.

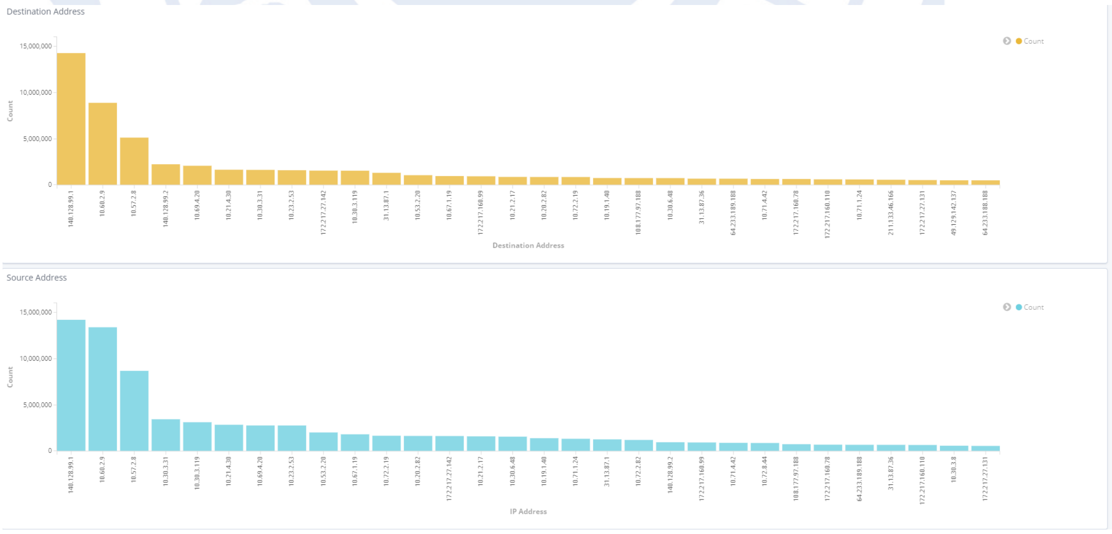


FIGURE 4.2: Source and destination count

Figure 4.3 is the information that summarizes the other two pictures, showing a visual analysis of proportional and counting, and a clearer understanding of various details.

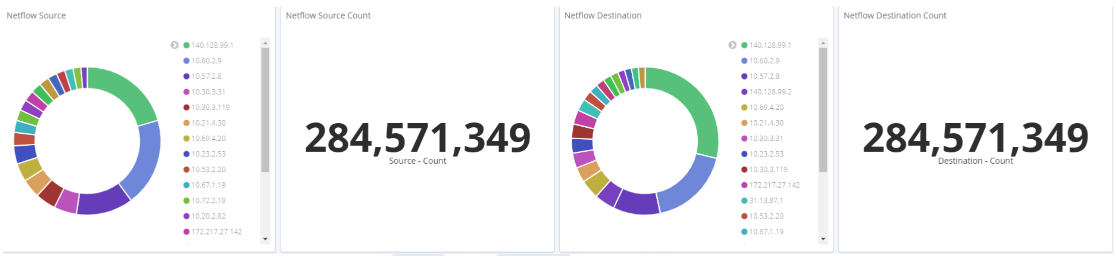


FIGURE 4.3: Count of each source and destination address

Figure 4.4 extracts the number of Packets independently, and some attacks are related to this factor, so the graph is deliberately made for independent display.

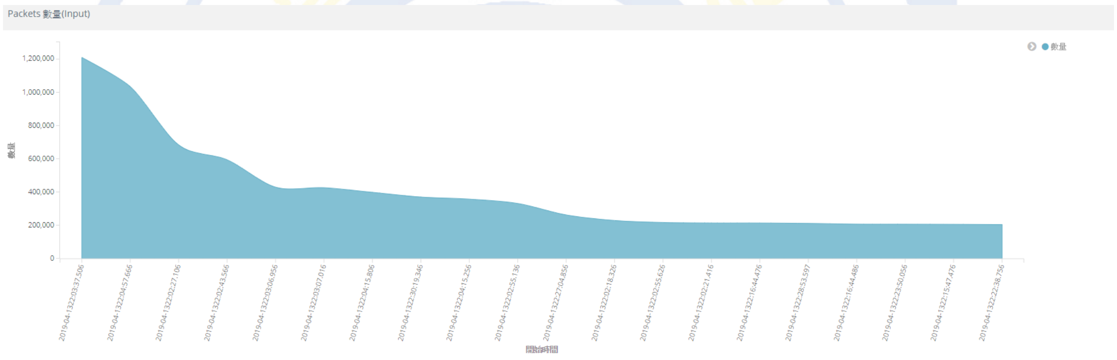


FIGURE 4.4: Count of packet

Figure 4.5 shows the detailed IP usage. The source IP, Port, and network protocols used are all presented. In particular, the IP with the most usage ratio can be used to control the network usage.

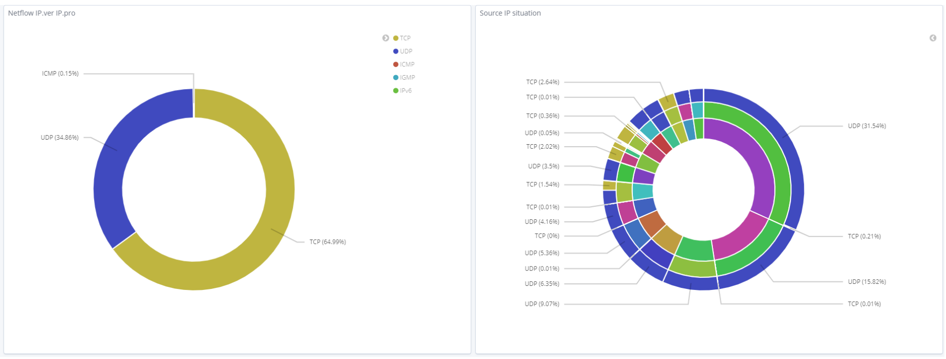


FIGURE 4.5: Source address details

Figure 4.6 shows the use of connections around the world, especially in thermal mode. Figure 4.7 shows the source IP from the world's respective detailed geographic information, from country to city with clear information available for management review.

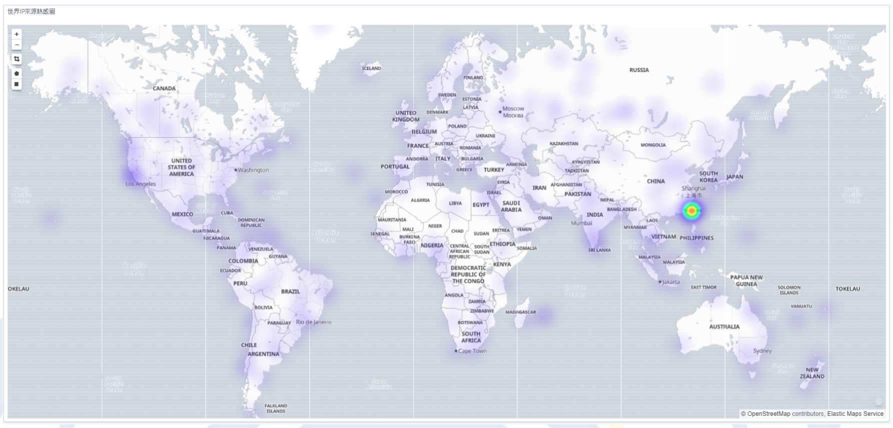


Figure 4.6: World heat map

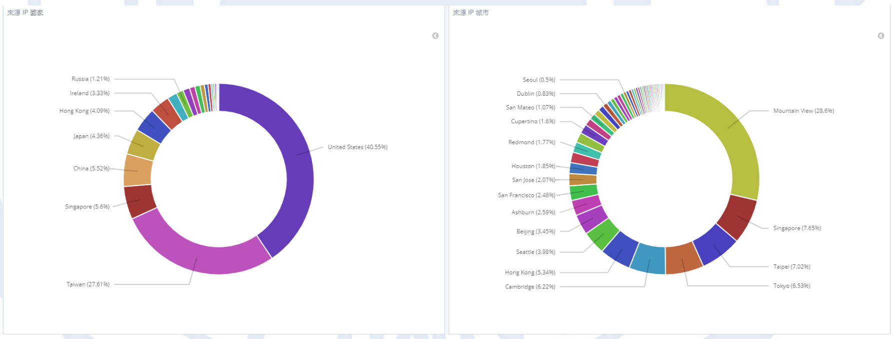


Figure 4.7: Geographic location information of source address

Dormitory network is usually the place where every school is most vulnerable to various emergencies. Each building will be allocated to a separate domain. Managers must first extract and filter the domain separately. After all dormitory domains are filtered out, a complete scale map will be presented, as shown in Fig. 4.8.

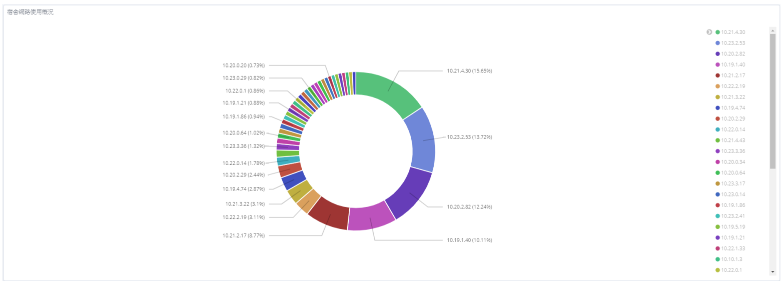


Figure 4.8: Dormitory network usage

In order to facilitate the search, the student network is divided into several regions, which can quickly inspect the use of the network. As shown in Figure 4.9, the use of each student partition is at a glance. This is the most convenient part of visual analysis. It can customize its own charts according to users' needs.

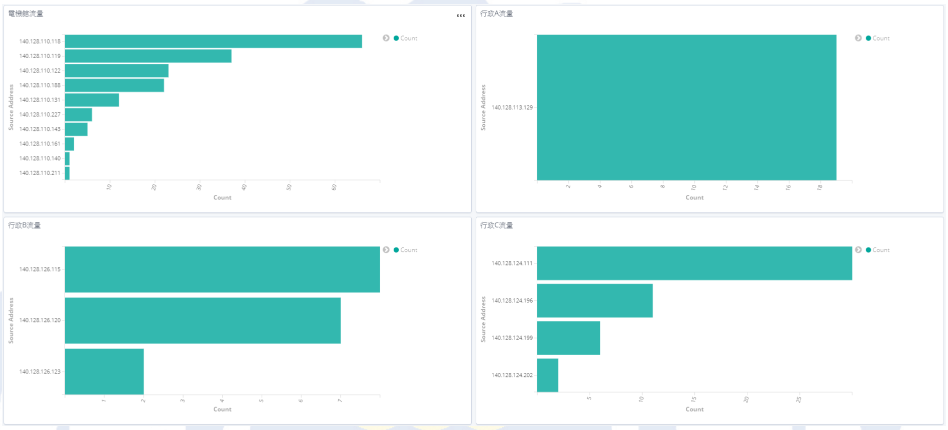


FIGURE 4.9: Dormitory network sources

Figures 4.10 and 4.11 are related features of using network attack behavior, classifying the suspicious source of each network attack, and managing the time of the network traffic to select the time of the attack behavior to be queried. The source of cyber attacks can be more quickly integrated.

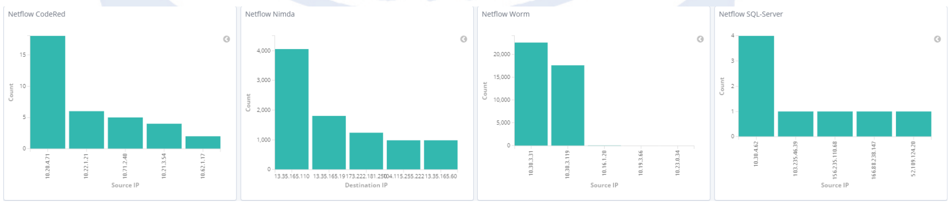


FIGURE 4.10: Cyberattack detection1

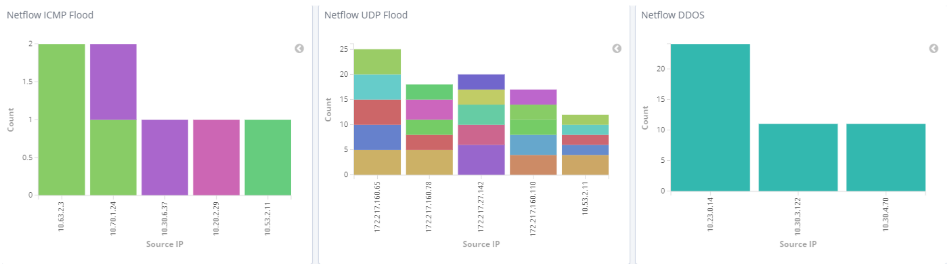


FIGURE 4.11: Cyberattack detection2

4.2.2 System Monitoring

System monitoring is the use of Grafana to link Elasticsearch in the log system, so that the system can immediately transmit system information to Grafana for analysis and processing. As shown in Figure 4.12, Figure 4.13 and Figure 4.14, the system's traffic and packet data can be displayed instantly, which makes it easier for the monitor to grasp the system usage.

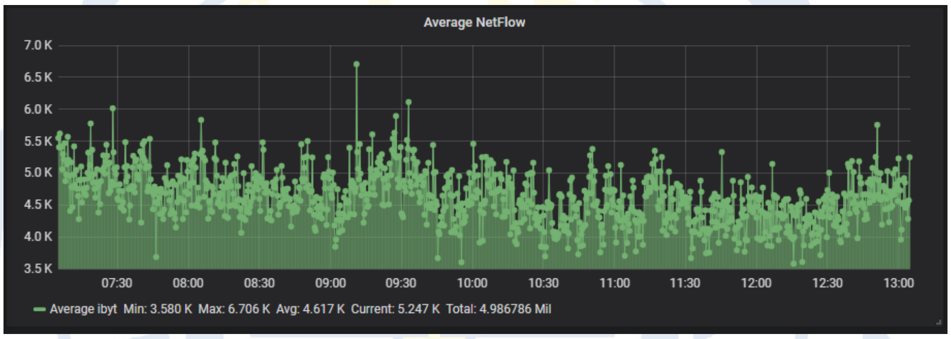


FIGURE 4.12: Average netflow

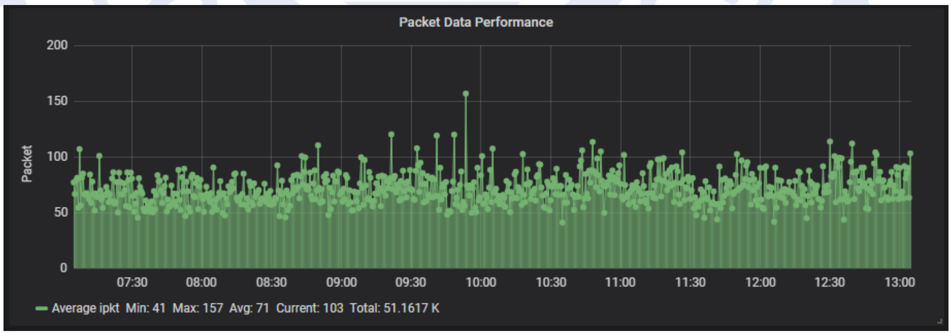


FIGURE 4.13: Packet data performance

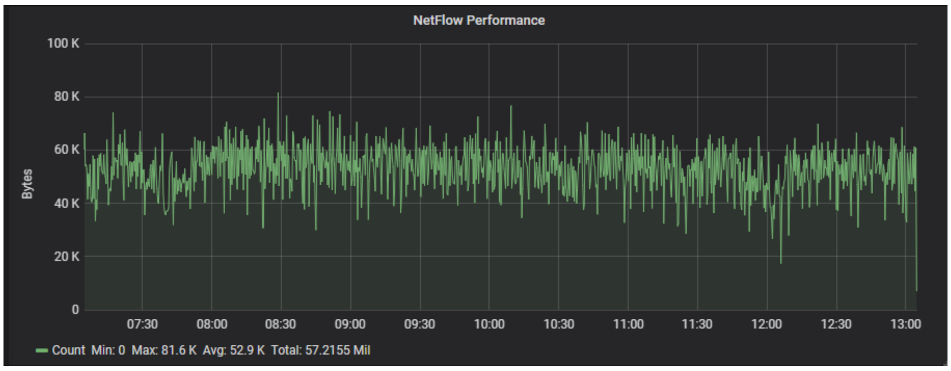
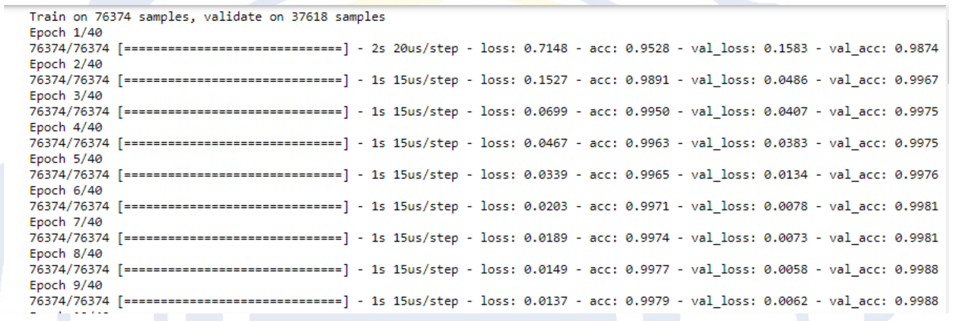


FIGURE 4.14: Netflow performance

4.2.3 Training Models

This section will show the models used for training and testing in this article, and the training of the models, which will be described separately in the following charts.

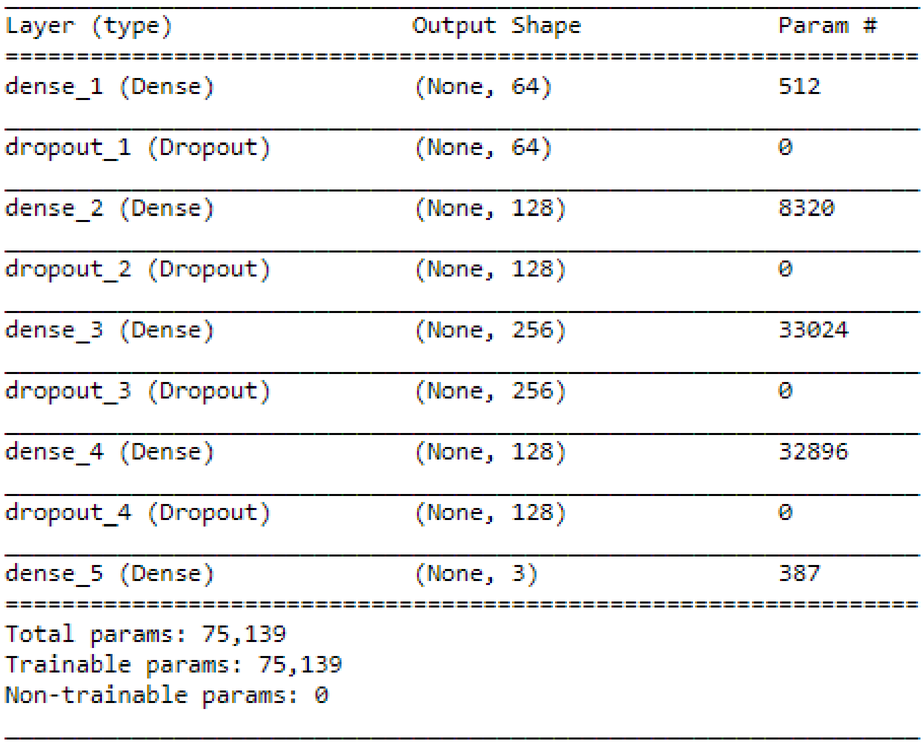
As shown in Figure 4.15, this is the training situation of the dnn model.



```
Train on 76374 samples, validate on 37618 samples
Epoch 1/40
76374/76374 [==============================] - 2s 20us/step - loss: 0.7148 - acc: 0.9528 - val_loss: 0.1583 - val_acc: 0.9874
Epoch 2/40
76374/76374 [==============================] - 1s 15us/step - loss: 0.1527 - acc: 0.9891 - val_loss: 0.0486 - val_acc: 0.9967
Epoch 3/40
76374/76374 [==============================] - 1s 15us/step - loss: 0.0699 - acc: 0.9950 - val_loss: 0.0407 - val_acc: 0.9975
Epoch 4/40
76374/76374 [==============================] - 1s 15us/step - loss: 0.0467 - acc: 0.9963 - val_loss: 0.0383 - val_acc: 0.9975
Epoch 5/40
76374/76374 [==============================] - 1s 15us/step - loss: 0.0339 - acc: 0.9965 - val_loss: 0.0134 - val_acc: 0.9976
Epoch 6/40
76374/76374 [==============================] - 1s 15us/step - loss: 0.0203 - acc: 0.9971 - val_loss: 0.0078 - val_acc: 0.9981
Epoch 7/40
76374/76374 [==============================] - 1s 15us/step - loss: 0.0189 - acc: 0.9974 - val_loss: 0.0073 - val_acc: 0.9981
Epoch 8/40
76374/76374 [==============================] - 1s 15us/step - loss: 0.0149 - acc: 0.9977 - val_loss: 0.0058 - val_acc: 0.9988
Epoch 9/40
76374/76374 [==============================] - 1s 15us/step - loss: 0.0137 - acc: 0.9979 - val_loss: 0.0062 - val_acc: 0.9988
```

Figure 4.15: Training situation of DNN model

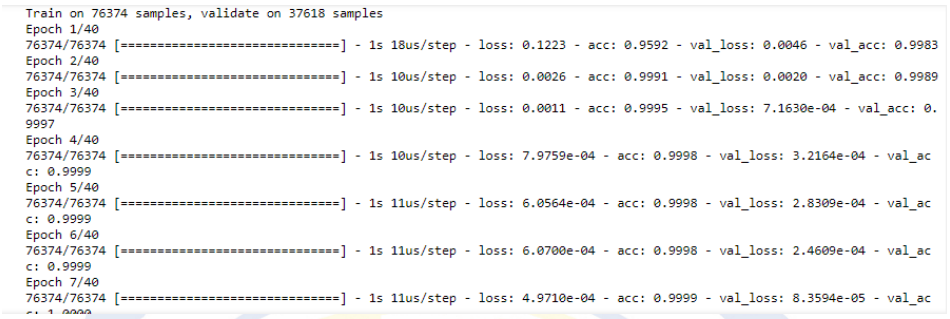
As shown in Figure 4.16, this is the neural layer hierarchy of the model and model of the DNN.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                512
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               8320
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               33024
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 3)                 387
=================================================================
Total params: 75,139
Trainable params: 75,139
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.16: Neuron architecture of DNN model

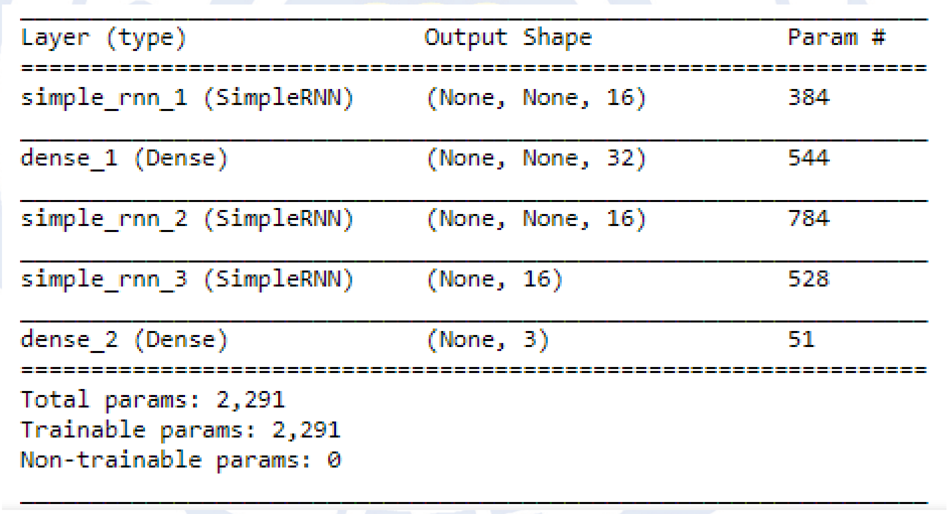
As shown in Figure 4.17, this is the training situation of the dnn model.



```
Train on 76374 samples, validate on 37618 samples
Epoch 1/40
76374/76374 [==============================] - 1s 18us/step - loss: 0.1223 - acc: 0.9592 - val_loss: 0.0046 - val_acc: 0.9983
Epoch 2/40
76374/76374 [==============================] - 1s 10us/step - loss: 0.0026 - acc: 0.9991 - val_loss: 0.0020 - val_acc: 0.9989
Epoch 3/40
76374/76374 [==============================] - 1s 10us/step - loss: 0.0011 - acc: 0.9995 - val_loss: 7.1630e-04 - val_acc: 0.
9997
Epoch 4/40
76374/76374 [==============================] - 1s 10us/step - loss: 7.9759e-04 - acc: 0.9998 - val_loss: 3.2164e-04 - val_ac
c: 0.9999
Epoch 5/40
76374/76374 [==============================] - 1s 11us/step - loss: 6.0564e-04 - acc: 0.9998 - val_loss: 2.8309e-04 - val_ac
c: 0.9999
Epoch 6/40
76374/76374 [==============================] - 1s 11us/step - loss: 6.0700e-04 - acc: 0.9998 - val_loss: 2.4609e-04 - val_ac
c: 0.9999
Epoch 7/40
76374/76374 [==============================] - 1s 11us/step - loss: 4.9710e-04 - acc: 0.9999 - val_loss: 8.3594e-05 - val_ac
```

FIGURE 4.17: Training situation of RNN model

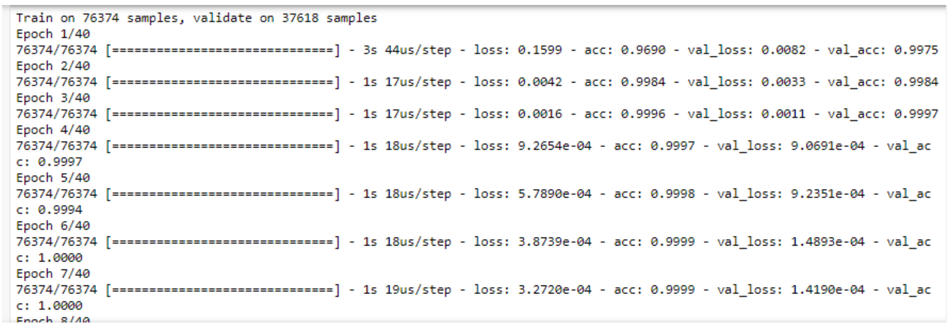
As shown in Figure 4.18, this is the training situation of the dnn model.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_1 (SimpleRNN)     (None, None, 16)          384
_________________________________________________________________
dense_1 (Dense)              (None, None, 32)          544
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, None, 16)          784
_________________________________________________________________
simple_rnn_3 (SimpleRNN)     (None, 16)                528
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 51
=================================================================
Total params: 2,291
Trainable params: 2,291
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 4.18: Neuron architecture of RNN model

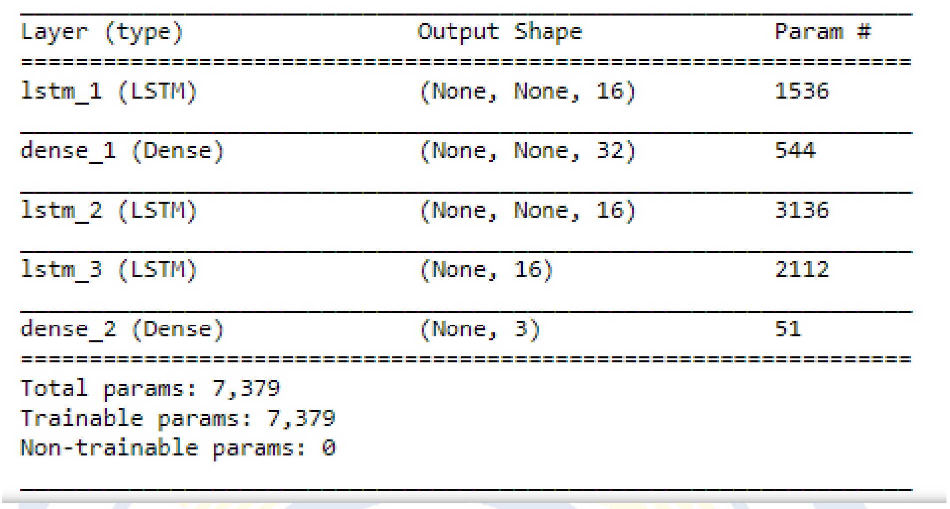
As shown in Figure 4.19, this is the training situation of the dnn model.



```
Train on 76374 samples, validate on 37618 samples
Epoch 1/40
76374/76374 [==============================] - 3s 44us/step - loss: 0.1599 - acc: 0.9690 - val_loss: 0.0082 - val_acc: 0.9975
Epoch 2/40
76374/76374 [==============================] - 1s 17us/step - loss: 0.0042 - acc: 0.9984 - val_loss: 0.0033 - val_acc: 0.9984
Epoch 3/40
76374/76374 [==============================] - 1s 17us/step - loss: 0.0016 - acc: 0.9996 - val_loss: 0.0011 - val_acc: 0.9997
Epoch 4/40
76374/76374 [==============================] - 1s 18us/step - loss: 9.2654e-04 - acc: 0.9997 - val_loss: 9.0691e-04 - val_ac
c: 0.9997
Epoch 5/40
76374/76374 [==============================] - 1s 18us/step - loss: 5.7890e-04 - acc: 0.9998 - val_loss: 9.2351e-04 - val_ac
c: 0.9994
Epoch 6/40
76374/76374 [==============================] - 1s 18us/step - loss: 3.8739e-04 - acc: 0.9999 - val_loss: 1.4893e-04 - val_ac
c: 1.0000
Epoch 7/40
76374/76374 [==============================] - 1s 19us/step - loss: 3.2720e-04 - acc: 0.9999 - val_loss: 1.4190e-04 - val_ac
c: 1.0000
```

FIGURE 4.19: Training situation of LSTM model

As shown in Figure 4.20, this is the training situation of the dnn model.



```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, None, 16)          1536
_________________________________________________________________
dense_1 (Dense)              (None, None, 32)          544
_________________________________________________________________
lstm_2 (LSTM)                (None, None, 16)          3136
_________________________________________________________________
lstm_3 (LSTM)                (None, 16)                2112
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 51
=================================================================
Total params: 7,379
Trainable params: 7,379
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 4.20: Neuron architecture of LSTM model

4.2.4 Model Comparison

In the classification part of the network attack, the three models have good accuracy in the verification set, as shown in Figure 4.21 and Figure 4.22, which is the training and verification of the DNN model.

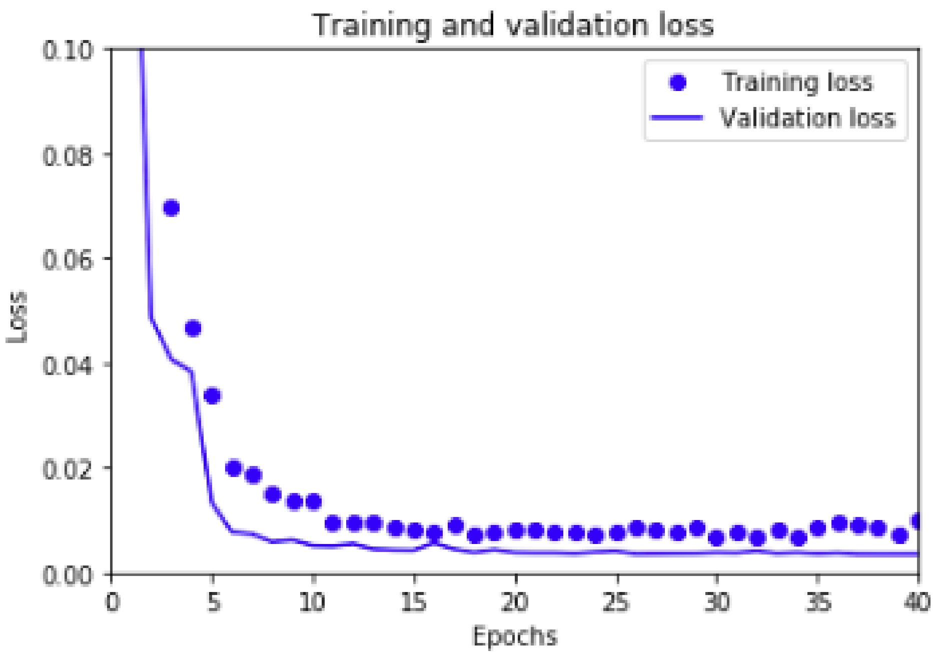


FIGURE 4.21: Training and validation loss of DNN model

Figure 4.23 and Figure 4.24 show the training and verification of the RNN model, which is better than the DNN model.

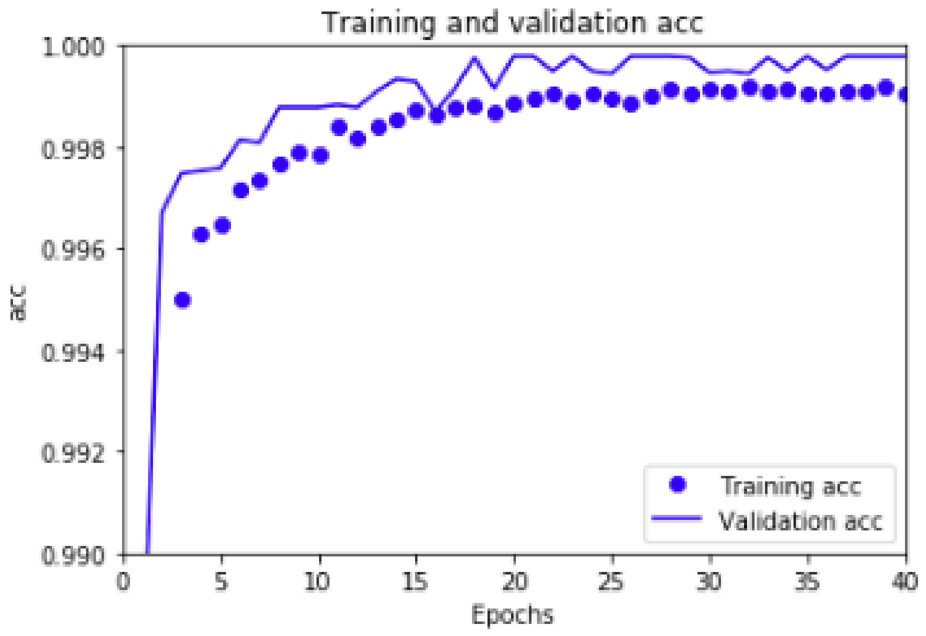


FIGURE 4.22: Training and validation acc of DNN model

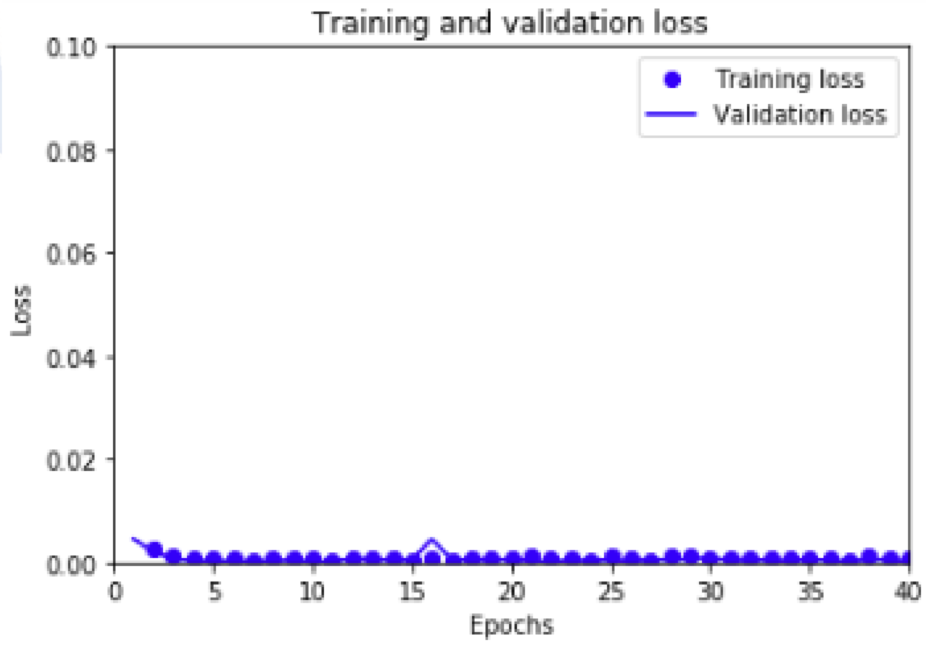


FIGURE 4.23: Training and validation loss of RNN model

Finally, the training and verification scenarios of the LSTM model are shown in Figure 4.25 and Figure 4.26, which is very similar to the RNN model.

As Table 4.2 and Table 4.3 show, the initial training results show that the training results of the RNN model are much better than the other two models, but after stopping the neurons and some parameters, a very high accuracy can be

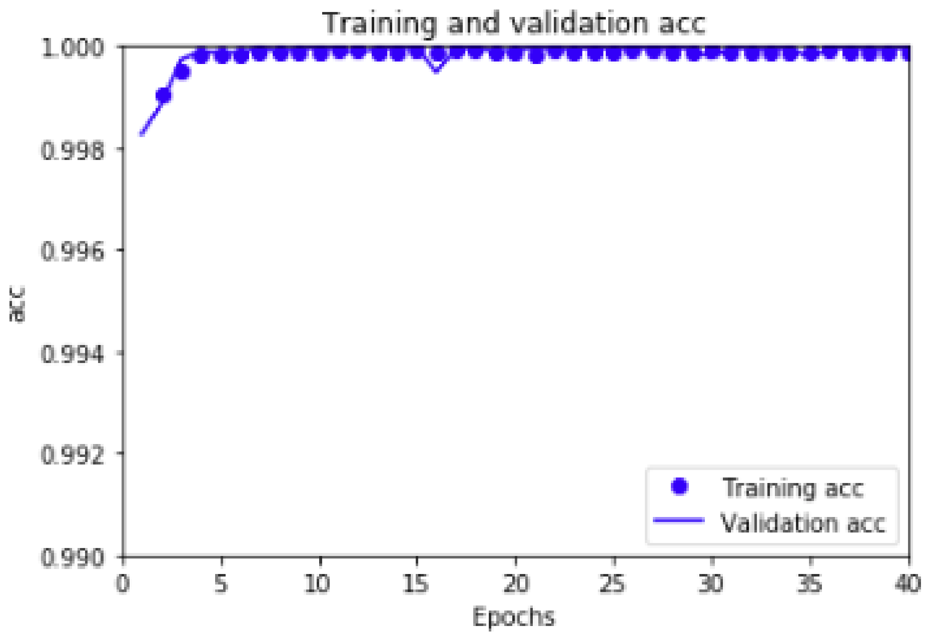


FIGURE 4.24: Training and validation acc of RNN model

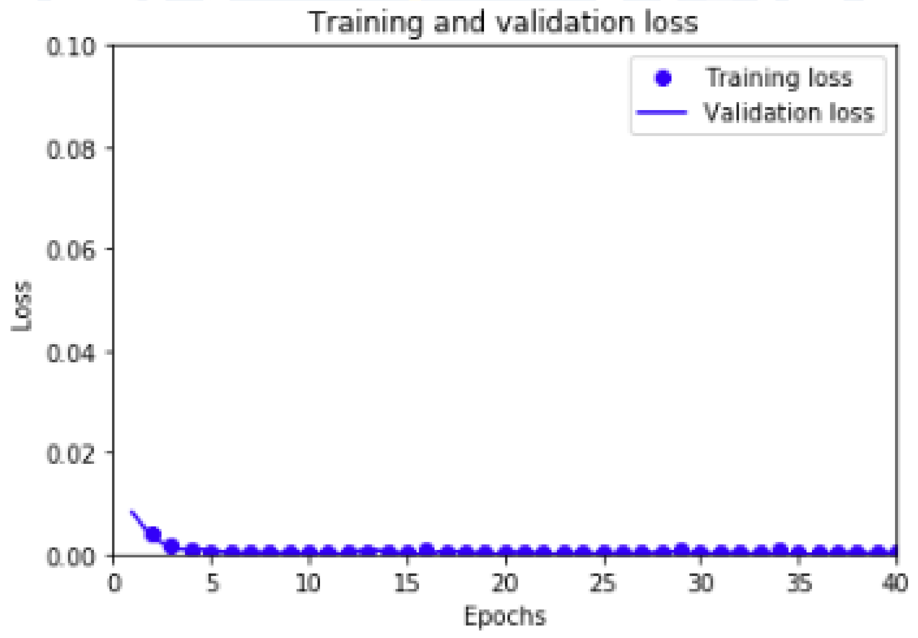


FIGURE 4.25: Training and validation loss of LSTM model

achieved.

Combining the two comparison results, the paper can know that the RNN model is more suitable for the data sources in this paper.

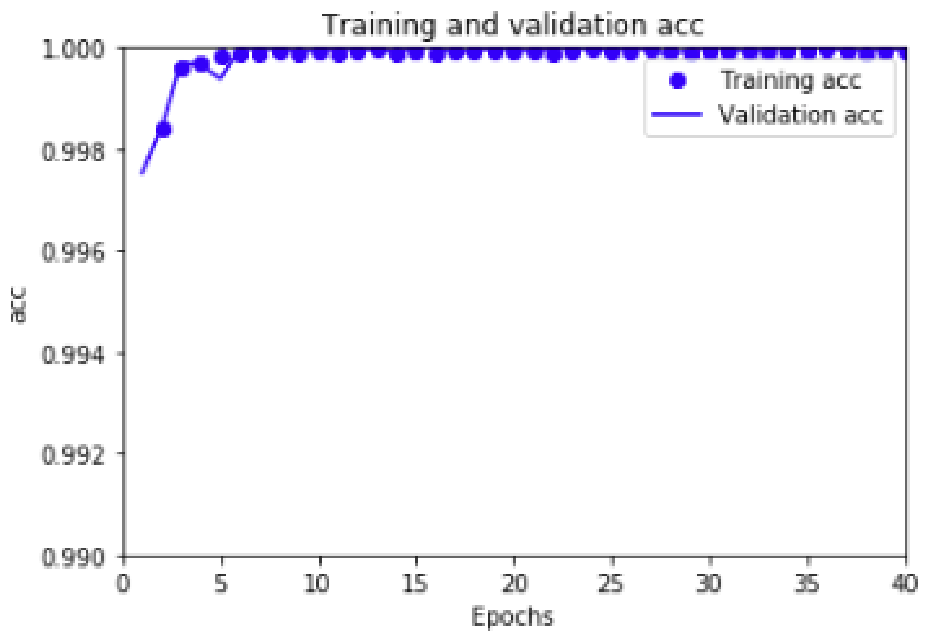


FIGURE 4.26: Training and validation loss of LSTM model

TABLE 4.2: Initial model comparison

	DNN	RNN	LSTM
Verification set accuracy	89.882%	96.947%	96.194%
Test.1 set accuracy	92.84%	99.47%	99.47%
Test.2 set accuracy	90.98%	97.48%	97.48%
Test.3 set accuracy	93.27%	99.98%	99.47%
Test.4 set accuracy	91.85%	100%	98.78%

TABLE 4.3: Final model comparison

	DNN	RNN	LSTM
Verification set accuracy	99.978%	99.997%	99.994%
Test.1 set accuracy	98.88%	100%	100%
Test.2 set accuracy	99.98%	99.98%	99.98%
Test.3 set accuracy	99.47%	100%	100%
Test.4 set accuracy	99.91%	99.98%	99.98%

Chapter 5

Conclusions and Future Work

This chapter is stated for the concluding remarks and the future works of this study.

5.1 Concluding Remarks

In conclution, the paper introduces a network log management and analysis system based on ELK Stack. Users can easily understand the general situation of network usage in the service area through the system. They can make detailed adjustments to the visual analysis of each area. The RNN model is also used for network attack classification, which can achieve more than 98%accuracy and is used for sorting out attack types. The recording system and the attack behavior are mutually comparable so that managers can obtain more accurate information.

5.2 Future Work

In the future, hoping to collect more information related to cyber attacks through the ELK Stack and compare them to the attack classification based on Deep Learning and Kibana's visual analysis. Network usage will provide appropriate visual analysis of different areas, which will make it easier for managers to view configuration files for network log data. This part of deep learning not only hopes to identify more types of attacks, but also hopes to be used to predict attack behavior in the future.

References

[1] Hongyu Liu, Bo Lang, Ming Liu, and Hanbing Yan. Cnn and rnn based payload classification methods for attack detection. *Knowledge-Based Systems*, 163:332–341, 2019.

[2] Imon Banerjee, Yuan Ling, Matthew C. Chen, Sadid A. Hasan, Curtis P. Langlotz, Nathaniel Moradzadeh, Brian Chapman, Timothy Amrhein, David Mong, Daniel L. Rubin, Oladimeji Farri, and Matthew P. Lungren. Comparative effectiveness of convolutional neural network (cnn) and recurrent neural network (rnn) architectures for radiology text report classification. *Artificial Intelligence in Medicine*, 2018.

[3] Olivier Brun, Yonghua Yin, and Erol Gelenbe. Deep learning with dense random neural network for detecting attacks against iot-connected home environments. *Procedia Computer Science*, 134:458 – 463, 2018. The 15th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2018) / The 13th International Conference on Future Networks and Communications (FNC-2018) / Affiliated Workshops.

[4] Rojalina Priyadarshini and Rabindra Kumar Barik. A deep learning based intelligent framework to mitigate ddos attack in fog environment. *Journal of King Saud University - Computer and Information Sciences*, 2019.

[5] Tae-Young Kim and Sung-Bae Cho. Web traffic anomaly detection using c-lstm neural networks. *Expert Systems with Applications*, 106:66–76, 2018.

[6] Qinghan Xue and Mooi Choo Chuah. New attacks on rnn based healthcare learning system and their detections. *Smart Health*, 9-10:144–157, 2018.

[7] Weiling Chen, Chai Kiat Yeo, Chiew Tong Lau, and Bu Sung Lee. Leveraging social media news to predict stock index movement using rnn-boost. *Data & Knowledge Engineering*, 118:14 – 24, 2018.

[8] YiFei Li and Han Cao. Prediction for tourism flow based on lstm neural network. *Procedia Computer Science*, 129:277 – 283, 2018. 2017 International Conference on Identification, Information and Knowledgein the Internet of Things.

[9] Afan Galih Salman, Yaya Heryadi, Edi Abdurahman, and Wayan Suparta. Single layer & multi-layer long short-term memory (lstm) model with intermediate variables for weather forecasting. *Procedia Computer Science*, 135:89 – 98, 2018. The 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life.

[10] Jitendra Kumar, Rimsha Goomer, and Ashutosh Kumar Singh. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters. *Procedia Computer Science*, 125:676 – 682, 2018. The 6th International Conference on Smart Computing and Communications.

[11] T. Prakash, M. Kakkar, and K. Patel. Geo-identification of web users through logs using elk stack. In *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pages 606–610, 2016.

[12] P. P. I. Langi, , W. Najib, and T. B. Aji. An evaluation of twitter river and logstash performances as elasticsearch inputs for social media analysis of twitter. In *2015 International Conference on Information Communication Technology and Systems (ICTS)*, pages 181–186, Sep. 2015.

[13] Valentín Carela-Español, Pere Barlet-Ros, Albert Cabellos-Aparicio, and Josep Solé-Pareta. Analysis of the impact of sampling on netflow traffic classification. *Computer Networks*, 55:1083–1099, 2011.

[14] Meir Kalech. Cyber-attack detection in scada systems using temporal pattern recognition techniques. *Computers & Security*, 84:225–238, 2019.

[15] Rafał Kozik, Michał Choraś, Massimo Ficco, and Francesco Palmieri. A scalable distributed machine learning approach for attack detection in edge computing environments. *Journal of Parallel and Distributed Computing*, 119:18–26, 2018.

[16] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357, 2019.

[17] Chun-Yu Wang, Chi-Lung Ou, Yu-En Zhang, Feng-Min Cho, Pin-Hao Chen, Jyh-Biau Chang, and Ce-Kuen Shieh. Botcluster: A session-based p2p botnet clustering system on netflow. *Computer Networks*, 145:175–189, 2018.

[18] Pin Wu, Zhihui Lu, Quan Zhou, Zhidan Lei, Xiaoqiang Li, Meikang Qiu, and Patrick C.K. Hung. Bigdata logs analysis based on seq2seq networks for cognitive internet of things. *Future Generation Computer Systems*, 90:477–488, 2019.

[19] R. Rastogi, A. S, S. G, P. G, P. D, and A. Singh. Design and development of generic web based framework for log analysis. In *2016 IEEE Region 10 Conference (TENCON)*, pages 232–236, 2016.

[20] M. Moh, S. Pininti, S. Doddapaneni, and T. Moh. Detecting web attacks using multi-stage log analysis. In *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pages 733–738, Feb 2016.

[21] Robert Appleyard and James H. Adams. Using the elk stack for castor application logging at ral. page 027, 03 2016.

[22] S Bagnasco, D Berzano, A Guarise, S Lusso, M Masera, and S Vallero. Monitoring of IaaS and scientific applications on the cloud using the elasticsearch ecosystem. *Journal of Physics: Conference Series*, 608:012016, may 2015.

[23] Rafał Kozik. Distributing extreme learning machines with apache spark for netflow-based malware activity detection. *Pattern Recognition Letters*, 101:14–20, 2018.

[24] Mariam Kiran and Anshuman Chhabra. Understanding flows in high-speed scientific networks: A netflow data study. *Future Generation Computer Systems*, 94:72–79, 2019.

[25] Julio Navarro, Aline Deruyver, and Pierre Parrend. A systematic survey on multi-step attack detection. *Computers & Security*, 76:214–249, 2018.

[26] Magdi S. Mahmoud, Mutaz M. Hamdan, and Uthman A. Baroudi. Modeling and control of cyber-physical systems subject to cyber attacks: A survey of recent advances and challenges. *Neurocomputing*, 338:101–115, 2019.

Appendix A

ELK Stack Installation

Set related kits

```
$ sudo apt-get update
$ sudo apt-get install -y vim ntp curl ssh
```

Install openjdk8

```
$ sudo apt-get install -y openjdk-8-jdk
```

Set openjdk8

```
$ sudo ln -s /usr/lib/jvm/java-8-openjdk-amd64 /usr/lib/jvm/jdk
```

Set .bashrc

```
$ sudo vim .bashrc
$ export JAVA_HOME=/usr/lib/jvm/jdk/
$ source .bashrc
```

Install apt-transport-https & ca-certificates

```
$ sudo apt-get install apt-transport-https ca-certificates -y
```

Set key

```
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |
sudo apt-key add -
```

Set and install Elasticsearch

```
$ echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main"|
 sudo tee -a /etc/apt/sources.list.d/elastic-6.x.list
$ sudo apt-get update
$ sudo apt-get install -y elasticsearch
```

Set elasticsearch.yml

```
$ sudo vim /etc/elasticsearch/elasticsearch.yml
```

Start Elasticsearch

```
$ sudo systemctl start elasticsearch
```

Install Logstash

```
$sudo apt-get install logstash
```

Install Kibana

```
$ sudo apt-get install kibana
```

Set kibana.yml

```
$ sudo vim /etc/kibana/kibana.yml
```

Start Logstash

```
$ sudo systemctl start logstash
```

Start Kibana

```
$ sudo systemctl start kibana
```

Appendix B

Install the Python environment

Set Python environmental location

```
md \pythonwork
cd \pythonwork
```

Set Anaconda environment

```
conda create --name tensorflow python=3.6 anaconda
activate tensorflow
```

Install Tensorflow and Keras

```
pip install tensorflow
pip install keras
jupyter notebook
```

Appendix C

Data preprocessing

```
from pandas import read_csv
from pandas import datetime
from matplotlib import pyplot
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
import re
import pandas
import pandas as pd

series = read_csv('nfcapd.201902151305.txt', engine='python',
 skipfooter=4, header=None, skiprows=[0])
series.columns = ['Date first seen', 'Date last seen',
 'Duration', 'Src IP Addr', 'Dst IP Addr', 'Src Pt',
  'Dst Pt', 'Proto', 'Flags', 'In Pkt', 'In Byte',
  'Out Pkt', 'Out Byte', 'Input', 'Output']
series.to_csv('datapro1.csv', encoding="utf-8-sig")
rowNum=series.shape[0]
colNum=series.columns.size
data = pandas.read_csv('datapro1.csv', engine='python', skipfooter=4)
del data["Flags"],data["Out Pkt"],data["Out Byte"]
if data['In Byte'].dtype != 'int64':
data['In Byte']=data['In Byte'].str.lstrip()
data['In Byte']=data['In Byte'].str.rstrip()
InByte = data['In Byte'].str.strip().str.split('.',0,True)
InByte.columns = ['0', '1']
InByte = InByte.fillna('.0')
InByte['1'] = InByte['1'].replace(' M', '00.0', regex=True)
InByte['1'] = InByte['1'].replace(' G', '00000.0', regex=True)
data['In Byte'] = InByte['0']+InByte['1']
InByte = data['In Byte'].str.strip().str.split('.',0,True)
data['In Byte'] = InByte[0]
```

```
data['In Byte'] = data['In Byte'].astype(int)
if data['In Pkt'].dtype != 'int64':
data['In Pkt']=data['In Pkt'].str.lstrip()
data['In Pkt']=data['In Pkt'].str.rstrip()
InPkt = data['In Pkt'].str.strip().str.split('.',0,True)
InPkt.columns = ['0', '1']
InPkt = InPkt.fillna('.0')
InPkt['1'] = InPkt['1'].replace(' M', '00.0', regex=True)
InPkt['1'] = InPkt['1'].replace(' G', '00000.0', regex=True)
data['In Pkt'] = InPkt['0']+InPkt['1']
InPkt = data['In Pkt'].str.strip().str.split('.',0,True)
data['In Pkt'] = InPkt[0]
data['In Pkt'] = data['In Pkt'].astype(int)
data['Dst Pt'] = data['Dst Pt'].astype(int)
dataflow = data
from sklearn.preprocessing import LabelEncoder
dataflow['Proto'] = dataflow['Proto'].str.lstrip()
dataflow['Proto'] = dataflow['Proto'].str.rstrip()
le = LabelEncoder()
le.fit(['ICMP','IGMP', 'TCP','UDP'])
dataflow['Proto'] = le.transform(dataflow['Proto'])
le.transform(['ICMP','IGMP','TCP','UDP'])
dataflow['Src IP Addr'] = dataflow['Src IP Addr'].str.lstrip()
dataflow['Src IP Addr'] = dataflow['Src IP Addr'].str.rstrip()
dataflow['Dst IP Addr'] = dataflow['Dst IP Addr'].str.lstrip()
dataflow['Dst IP Addr'] = dataflow['Dst IP Addr'].str.rstrip()
CodeRed = dataflow.loc[(dataflow['Dst Pt'].astype(int) == 80 ) &
 (dataflow['In Pkt'].astype(int)==3) & (dataflow['In Byte'].astype(int)==144)]
CodeRed['attack']='0'
if CodeRed.shape[0] > 1:
print(CodeRed['Src IP Addr'].drop_duplicates(keep='first'))
else:
print("no CodeRed attack")
Nimda = dataflow.loc[(dataflow['Dst Pt'].astype(int) == 80 ) ]
from collections import Counter
A = Nimda['Src IP Addr'].value_counts() >=1000
A = Counter(A)[1]
if A !=0 :
for i in range (A):
Nimda_IP=Nimda['Src IP Addr']
Cou_IP=Counter(Nimda_IP)
Top_IP = Cou_IP.most_common()[i][0]
Top_IP_Cou = Cou_IP.most_common()[i][1]
if (Top_IP_Cou >=1000):
Nimda_A = Nimda.loc[(Nimda['Src IP Addr'] == Top_IP )]
Nimda_A['attack']='1'
print(Top_IP + "對外連線次數為%d" % (Top_IP_Cou))
else:
```

```
Nimda_A=None
print('no Nimda attack')
else:
Nimda_A=None
print('no Nimda attack')
from collections import Counter
Worm = dataflow.loc[(dataflow['Dst Pt'].astype(int) == 445 ) &
 (dataflow['Proto'] == 2)]
A = Worm['Src IP Addr'].value_counts() >=1000
A = Counter(A)[1]
Worm_A = None
if A !=0 :
for i in range (0,A):
Worm_IP=Worm['Src IP Addr']
Worm_Cou_IP=Counter(Worm_IP)
Worm_Top_IP = Worm_Cou_IP.most_common()[i][0]
Worm_Top_IP_cou = Worm_Cou_IP.most_common()[i][1]
Worm_A = pd.concat([Worm_A, dataflow.loc[(dataflow['Dst Pt'].astype(int)
 == 445 ) & (dataflow['Proto'] == 2) & (dataflow['Src IP Addr'] ==
  Worm_Top_IP)]])
Worm_A['attack']='2'
if Worm_Top_IP_cou > 1000 :
print ("IP: " + Worm_Top_IP + "在Port445的TCP連接次數為%d" %
 (Worm_Top_IP_cou))

else:
print("no Worm attack")
else:
print("no Worm attack")
CodeRed = CodeRed[0:9]
Nimda_A = Nimda_A[0:9]
Worm_A = Worm_A[0:9]
attack1= pd.concat([CodeRed, Nimda_A, Worm_A])
attack1
attack_data= pd.concat([attack1, attack2])
attack_data
attack_data['Src IP Addr'] = attack_data['Src IP Addr'].str.lstrip()
attack_data['Dst IP Addr'] = attack_data['Dst IP Addr'].str.lstrip()
le = LabelEncoder()
le.fit(attack_data['Src IP Addr'])
attack_data['Src IP Addr'] = le.transform(attack_data['Src IP Addr'])
le.fit(attack_data['Dst IP Addr'])
attack_data['Dst IP Addr'] = le.transform(attack_data['Dst IP Addr'])
attack_data.to_csv('0604.csv')
attack_data
```

Appendix D

Merge CSV files

```
from pandas import read_csv
from pandas import datetime
from matplotlib import pyplot
import pandas as pd
import re

series = pd.read_csv('final.csv', engine='python', skipfooter=0,
 header=None, skiprows=[0])
        del series[0], series[1]
series.columns = ['Date first seen', 'Date last seen',
 'Dst IP Addr', 'Dst Pt', 'Duration', 'In Byte', 'In Pkt'
 , 'Input', 'Output', 'Proto', 'Src IP Addr', 'Src Pt', 'attack']
series.to_csv('datarnn.csv')
series
```

Appendix E

DNN model

```
import numpy as np
from pandas import read_csv
from matplotlib import pyplot as plt
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from keras.layers import Input, SimpleRNN, Activation
from keras import optimizers
from keras.optimizers import RMSprop
from collections import Counter

filename = 'rnndata1.csv'
footer = 1
data = read_csv(filename, header=None, engine='python', skiprows=footer)
del data[0], data[1], data[2], data[5]
A = OneHotEncoder(sparse = False).fit_transform( data[[13]] )
data = data.values
X = data[:,1:8]
Y = A
X_train, X_test, y_train, y_test = train_test_split(X, Y,
 test_size=0.33, random_state=8)
model = Sequential()
model.add(Dense(64, input_dim=7, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
```

```
        model.add(Dense(128, activation='relu'))
        model.add(Dropout(0.3))
        model.add(Dense(3, activation='softmax'))
                model.summary()
        rmsprop = RMSprop(lr=0.001)
        model.compile(optimizer=rmsprop, loss='categorical_crossentropy',
         metrics=['accuracy'])
        history = model.fit(X_train, y_train, epochs=40, batch_size=200,
         validation_data=(X_test, y_test))
        predictions = model.predict(X_test)
print(predictions)

history_dict= history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc = history_dict['acc']
val_acc = history_dict['val_acc']


epochs = range(1,len(loss_values)+1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.axis([0, 40, 0.00, 0.10])
plt.show()
epochs = range(1,len(loss_values)+1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation acc')
plt.xlabel('Epochs')
plt.ylabel('acc')
plt.legend()
plt.axis([0, 40, 0.99, 1.00])
plt.show()
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
print("Accuracy: %.2f%%" % (score[1] * 100.0))

predictions = model.predict(X_test)
np.set_printoptions(threshold=np.inf)
d = np.rint(predictions)
from collections import Counter
def row_counter(d):
lit_of_tups = [tuple(ele) for ele in d]
```

```
return Counter(lit_of_tups)
row_counter(d)
row_counter(y_test)
model.save('attackdnn_model')
filename = '0602.csv'
footer = 1
vdata = read_csv(filename, header=None, engine='python', skiprows=footer)
del vdata[0], vdata[1], vdata[2], vdata[5]
V = OneHotEncoder(sparse = False).fit_transform( vdata[[13]] )
vdata = vdata.values
X_v = vdata[:,1:8]
Y_v = V
from keras.models import load_model
vmodel = load_model('attack_model')
from sklearn.metrics import accuracy_score
score, acc = vmodel.evaluate(X_v, Y_v,batch_size=10)
print('Test score:', score)
print('Test accuracy:', acc)
print("Accuracy: %.2f%%" % (acc * 100.0))
predictions = model.predict(X_v)
np.set_printoptions(threshold=np.inf)
d = np.rint(predictions)
from collections import Counter
def row_counter(d):
lit_of_tups = [tuple(ele) for ele in d]
return Counter(lit_of_tups)
row_counter(d)
def row_counter(Y_v):
lit_of_tups = [tuple(ele) for ele in Y_v]
return Counter(lit_of_tups)
row_counter(Y_v)
def row_counter(Y_v):
lit_of_tups = [tuple(ele) for ele in Y_v]
return Counter(lit_of_tups)
row_counter(Y_v)
print('CodeRed', row_counter(Y_v)[(1.0, 0.0, 0.0)])
print('Nimda', row_counter(Y_v)[(0.0, 1.0, 0.0)])
print('Worm', row_counter(Y_v)[(0.0, 0.0, 1.0)])
```

Appendix F

RNN model

```
    import numpy as np
    from pandas import read_csv
    from matplotlib import pyplot as plt
    import math
    from keras.models import Sequential
    from keras.layers import Dense
    from keras.layers import Dropout
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder
    from sklearn.preprocessing import OneHotEncoder
    from keras.layers import Input, SimpleRNN, Activation
    from keras import optimizers
    from keras.optimizers import RMSprop
    from collections import Counter
    from keras.layers import Input, SimpleRNN, Activation, LSTM

    filename = 'rnndata1.csv'
    footer = 1
    data = read_csv(filename, header=None, engine='python', skiprows=footer)
    del data[0], data[1], data[2], data[5]
    A = OneHotEncoder(sparse = False).fit_transform( data[[13]] )
    data = data.values
    X = data[:,1:8]
    Y = A
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
     test_size=0.33, random_state=8)
    X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
    X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
    model = Sequential()
model.add(SimpleRNN(16, input_dim=7, return_sequences=True))
model.add(Dense(32, activation="relu"))
model.add(SimpleRNN(16, input_dim=7, return_sequences=True))
model.add(SimpleRNN(16, input_dim=7, return_sequences=False))
model.add(Dense(3, activation="softmax"))
model.summary()
    rmsprop = RMSprop(lr=0.001)
    model.compile(optimizer=rmsprop, loss='categorical_crossentropy',
     metrics=['accuracy'])
    history = model.fit(X_train, y_train, epochs=40, batch_size=200,
     validation_data=(X_test, y_test))
    predictions = model.predict(X_test)
    print(predictions)

    history_dict= history.history
    loss_values = history_dict['loss']
    val_loss_values = history_dict['val_loss']
    acc = history_dict['acc']
    val_acc = history_dict['val_acc']


    epochs = range(1,len(loss_values)+1)
    plt.plot(epochs, loss_values, 'bo', label='Training loss')
    plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.axis([0, 40, 0.00, 0.10])
    plt.show()
    epochs = range(1,len(loss_values)+1)
    plt.plot(epochs, acc, 'bo', label='Training acc')
    plt.plot(epochs, val_acc, 'b', label='Validation acc')
    plt.title('Training and validation acc')
    plt.xlabel('Epochs')
    plt.ylabel('acc')
    plt.legend()
    plt.axis([0, 40, 0.99, 1.00])
    plt.show()
    score = model.evaluate(X_test, y_test, verbose=0)
    print('Test loss:', score[0])
    print('Test accuracy:', score[1])
    print("Accuracy: %.2f%%" % (score[1] * 100.0))

    predictions = model.predict(X_test)
    np.set_printoptions(threshold=np.inf)
    d = np.rint(predictions)
    from collections import Counter
    def row_counter(d):
    lit_of_tups = [tuple(ele) for ele in d]
```

```
return Counter(lit_of_tups)
row_counter(d)
row_counter(y_test)
model.save('attackdnn_model')
filename = '0602.csv'
footer = 1
vdata = read_csv(filename, header=None, engine='python', skiprows=footer)
del vdata[0], vdata[1], vdata[2], vdata[5]
V = OneHotEncoder(sparse = False).fit_transform( vdata[[13]] )
vdata = vdata.values
X_v = vdata[:,1:8]
Y_v = V
from keras.models import load_model
vmodel = load_model('attackrnn_model')
from sklearn.metrics import accuracy_score
score, acc = vmodel.evaluate(X_v, Y_v,batch_size=10)
print('Test score:', score)
print('Test accuracy:', acc)
print("Accuracy: %.2f%%" % (acc * 100.0))
predictions = model.predict(X_v)
np.set_printoptions(threshold=np.inf)
d = np.rint(predictions)
from collections import Counter
def row_counter(d):
lit_of_tups = [tuple(ele) for ele in d]
return Counter(lit_of_tups)
row_counter(d)
def row_counter(Y_v):
lit_of_tups = [tuple(ele) for ele in Y_v]
return Counter(lit_of_tups)
row_counter(Y_v)
def row_counter(Y_v):
lit_of_tups = [tuple(ele) for ele in Y_v]
return Counter(lit_of_tups)
row_counter(Y_v)
print('CodeRed', row_counter(Y_v)[(1.0, 0.0, 0.0)])
print('Nimda', row_counter(Y_v)[(0.0, 1.0, 0.0)])
print('Worm', row_counter(Y_v)[(0.0, 0.0, 1.0)])
```

Appendix G

LSTM model

```
import numpy as np
from pandas import read_csv
from matplotlib import pyplot as plt
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from keras.layers import Input, SimpleRNN, Activation
from keras import optimizers
from keras.optimizers import RMSprop
from collections import Counter
from keras.layers import Input, SimpleRNN, Activation, LSTM

filename = 'rnndata1.csv'
footer = 1
data = read_csv(filename, header=None, engine='python', skiprows=footer)
del data[0], data[1], data[2], data[5]
A = OneHotEncoder(sparse = False).fit_transform( data[[13]] )
data = data.values
X = data[:,1:8]
Y = A
X_train, X_test, y_train, y_test = train_test_split(X, Y,
 test_size=0.33, random_state=8)
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
model = Sequential()
model.add(LSTM(16, input_dim=7, return_sequences=True))
model.add(Dense(32, activation="relu"))
model.add(LSTM(16, input_dim=7, return_sequences=True))
model.add(LSTM(16, input_dim=7, return_sequences=False))
model.add(Dense(3, activation="softmax"))
model.summary()
rmsprop = RMSprop(lr=0.001)
model.compile(optimizer=rmsprop, loss='categorical_crossentropy',
 metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=40, batch_size=200,
 validation_data=(X_test, y_test))
predictions = model.predict(X_test)
print(predictions)

history_dict= history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc = history_dict['acc']
val_acc = history_dict['val_acc']


epochs = range(1,len(loss_values)+1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.axis([0, 40, 0.00, 0.10])
plt.show()
epochs = range(1,len(loss_values)+1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation acc')
plt.xlabel('Epochs')
plt.ylabel('acc')
plt.legend()
plt.axis([0, 40, 0.99, 1.00])
plt.show()
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
print("Accuracy: %.2f%%" % (score[1] * 100.0))

predictions = model.predict(X_test)
np.set_printoptions(threshold=np.inf)
d = np.rint(predictions)
from collections import Counter
def row_counter(d):
lit_of_tups = [tuple(ele) for ele in d]
```

```
return Counter(lit_of_tups)
row_counter(d)
row_counter(y_test)
model.save('attacklstm_model')
filename = '0602.csv'
footer = 1
vdata = read_csv(filename, header=None, engine='python', skiprows=footer)
del vdata[0], vdata[1], vdata[2], vdata[5]
V = OneHotEncoder(sparse = False).fit_transform( vdata[[13]] )
vdata = vdata.values
X_v = vdata[:,1:8]
Y_v = V
from keras.models import load_model
vmodel = load_model('attack_model')
from sklearn.metrics import accuracy_score
score, acc = vmodel.evaluate(X_v, Y_v,batch_size=10)
print('Test score:', score)
print('Test accuracy:', acc)
print("Accuracy: %.2f%%" % (acc * 100.0))
predictions = model.predict(X_v)
np.set_printoptions(threshold=np.inf)
d = np.rint(predictions)
from collections import Counter
def row_counter(d):
lit_of_tups = [tuple(ele) for ele in d]
return Counter(lit_of_tups)
row_counter(d)
def row_counter(Y_v):
lit_of_tups = [tuple(ele) for ele in Y_v]
return Counter(lit_of_tups)
row_counter(Y_v)
def row_counter(Y_v):
lit_of_tups = [tuple(ele) for ele in Y_v]
return Counter(lit_of_tups)
row_counter(Y_v)
print('CodeRed', row_counter(Y_v)[(1.0, 0.0, 0.0)])
print('Nimda', row_counter(Y_v)[(0.0, 1.0, 0.0)])
print('Worm', row_counter(Y_v)[(0.0, 0.0, 1.0)])
```