TeleMARS

TECHNICAL REPORT

EXPERIMENT AND IMPROVE REINFORCEMENT LEARNING ALGORITHMS TO ENHANCE ANOMALOUS NETWORK BEHAVIOUR DETECTION

SUMMARY

Cyber security is a significant research area because all of the operations based on government, military, commercial, financial and civilians gather, process, and store tremendous volume of data on computers and others. Cyber-attacks have imposed increasing threats and damages on our modern society. Network Intrusion Detection System (NIDS) is one of the major techniques in preventing cyber-attacks occurred in network traffic. Over the past decade, a lot of research work has been conducted to explore NIDS solutions. The previous studies suggested that AI algorithms have promising potentials in developing effective solutions to detect the growing attacks.

TeleMARS R&D team is committed to advancing AI-based methods, exploring realistic approaches to deploy the research outcomes in real network environment, and supporting on-going research in wider community. The key objectives of this project are to:

- Contribute to the development of NIDS to enhance cyber security capability.
- Contribute to research community in the subject of anomaly detection.
- Establish a practical collaboration framework to enable scientists and IT professionals from diverse background to work together to continuously contribute to NIDS research on various aspects.
- Test and prove TeleMARS operation and technical frameworks, and the team capabilities.
- Inspire and enable the participation of broader research community in cyber security domain supporting gender equality and inclusion.

This project was commenced in September 2020 and finalised in June 2021. The main activities included:

- Literature review and project design.
- Selection of publish datasets and machine learning methods.
- Data analysis and preparation.
- Machine learning model development and experiments.
- Establish evaluation pipelines to partially simulate real application environment.
- Model capability evaluation applying different datasets.

PROJECT IMPLEMENTATION

OVERVIEW

The expected research outcomes of this project include:

- the understanding in the capabilities of Reinforcement Learning method in cyber-attack detection;
- comparison of the performance of a number of popular machine learning (ML) models;
- an evaluation method and framework that can be deployed in real network environment;
- comparison of the robustness of various ML models when feature space changes; and
- a collaboration framework that supports researchers and professionals from various backgrounds.

The in-scope research work involved the following components to achieve the objectives.

- Selection of datasets and the ML models that show strong anomaly detection capability.

This project was supported by APINIC foundation.

- Design and develop various machine learning models using traditional machine learning experiment approach, including
    - Shallow Learning classification-based models;
    - Neural Network Deep Learning models; and
    - A novel Reinforcement Learning model.
- Experiment a novel evaluation method on the trained models.
- Experiment the evaluation of the trained model applying a separate dataset which has different feature space from the training data.

A comprehensive literature review was conducted to refine the project designs including research dataset selection, which is detailed in Project Implementation
Phase I section. Data analysis was conducted to refine the selection and the designs of learning algorithms.

This project selected two public datasets which were collected by different approaches and constructed with different feature spaces. Based on the data analysis and literature review, a number of Shallow Learning and Neural Network Deep Learning methods that were studied in previous research were selected to conduct the comparison against a Reinforcement Learning method.

The models were designed using the selected methods including Reinforcement Learning method, and experimented against the selected datasets. The performance of the models was measured against a set of metrics to conduct comparison and analysis of their capabilities in developing NIDS applications.

Future research in real-time detection requires an effective evaluation approach to measure the overall performance within an environment that simulates real network traffic. The scope of this project does not include real network traffic data collection and construction. An evaluation method was designed to simulate the detection process in real environment and test the consistency and stability of the trained models.

Meanwhile, this project established a collaboration framework providing supportive teamwork environment to support the joint research effort. The collaboration framework aims to enable broader research and IT communities to collaborate and contribute to the on-going research venture in the domain of NIDS applications.

The research activities are carried out in three phases to deliver the above components.

PHASE I SELECTION OF DATASETS AND REFERENCE MODELS

Literature review has been conducted to analyse the previous studies in the field of cyber-attack detection to understand the problems, the approaches studied, the outcomes, the potentials and challenges.

Data collection provides the source data of network traffic for detection model development and implementation. Creating or collecting effective datasets is challenging as it demands designing realistic environments that include wide diversity of normal and attack scenarios, and constructing a comprehensive profile that involves all possible legitimate behaviours.

Real network traffic data collection and processing is excluded in the scope of this project. The public datasets that were produced by networking experts are adopted to support the research work.

DATASET SELECTION

This project was supported by APINIC foundation.

The available public datasets can be classified as network traffic, electrical network-based, android app-based, internet application-based, and IoT-based.

The table below shows a list of available public datasets. [4]

Public dataset	Year of publish	Number of times cited by June 2019
DARPA 1998	1998	1069
KDD Cup 1999	1999	N/A
NSL-KDD	2009	1630
UNSW-NB15	2015	202
DEFCON	2000	12
CAIDAs	2017	18
CDX	2013	8
TWENTE	2014	222
CIC DoS	2017	18
CIC-IDS2017	2017	87
CSE-CIC-IDS2018	2018	N/A
ISCX	2012	453
ADFA2013	2013	147
LBNL	2016	7
ICS cyber attack	2015	124
IEEE 300-bus power test system	N/A	171
Tor-nonTor	2017	18
URL	2016	7
MAWI	2011	182
VPN-nonVPN	2016	49
Android validation	2014	33
Android malware	2018	1
Bot-IoT	2018	2
CTU-13	2013	244
ISOT	2008	98
SSHCure	2014	37

Table 1 Available public dataset

Analysis was conducted on these datasets applying the following criteria:

- Network traffic data including those at routers and links.
- Completeness of network traffic profile including both benign and abnormal behaviours.
- Number of times cited.
- Methods of data collection and preparation.

This project was supported by APINIC foundation.

NSL-KDD and CIC-IDS2017 datasets were selected to conduct machine learning model designs and experiments.

KDD CUP 1999 is considered benchmark data for assessment of intrusion detection systems. The data includes four main categories of attacks that are Denialof-Service (DoS), user-to-root (U2R), Remote to Local Attack (R2L) and Probing Attack. Also, there are three content features and thirty-eight numerical features in the dataset. The features consist of basic features of individual TCP connections, content features within a connection suggested by domain knowledge and traffic features computed using a two-second time window. The NSL-KDD dataset is recommended to solve some of the inherent problems of the KDD'99 dataset. Compared to the original KDD dataset, the NSL-KDD dataset has the following improvements: (1) it does not include redundant records, (2) it does not include duplicate records, (3) the number of selected records is organized as the percentage of records, and (4) the number of records is reasonable.

As NSL-KDD has been heavily used in the studies of machine learning methods, it is used in this project for a benchmark comparison.

The Canadian Institute for Cybersecurity (CIC) conducted a number of projects aiming to overcome the shortcomings of previous datasets, aiming to develop a systematic approach to generate diverse and comprehensive benchmark dataset for intrusion detection based on the creation of user profiles which contain abstract representations of events and behaviours seen on the network. Generating realistic background traffic was the top priority in building this dataset. CIC has used the proposed B-Profile system to profile the abstract behaviour of human interactions and generates naturalistic benign background traffic. For this dataset, the abstract behaviour of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols were built. CIC-IDS2017 dataset comprises both benign behaviour and also details of new malware attacks: such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. This dataset is labelled based on the timestamp, source and destination IPs, source and destination ports, protocols and attacks. A complete network topology was configured to collect this dataset which contains Modem, Firewall, Switches, Routers, and nodes with different operating systems (Microsoft Windows (like Windows 10, Windows 8, Windows 7, and Windows XP), Apple's macOS iOS, and open-source operating system Linux).

The CICFlowMeter tool is used to extract 80 network flow features from the generated network traffic. The flows are tagged using the timestamp, the source and destination ports and IP addresses, and protocol types.

It reproduced comprehensive network traffic conditions and categories of data at both router and application level, providing relatively comprehensive network traffic profile for model training. It has been cited 87 times in recent research work. By using this dataset, this research work could contribute to the research community by adding meaningful reference data and/or lessons for any subsequent research work.

PROPOSED AI APPROACHES AND MODELS

There are two major IDS approach classes for building attack detection model: Signature IDS approach and Anomaly IDS approach.

A Signature IDS monitors network traffic to match observed behaviours with attack signatures logged in a database. It produces higher detection rates and lower false alarm rates for known attacks than other types, but it cannot detect new or even variants of known attacks. This is a significant issue in terms of the computer security required to defend against those attacks. Moreover, a huge effort is necessary to repeatedly update its database that includes various rules for malicious activities, established by network security experts. To address the drawbacks of signature IDS, anomaly IDS approaches have been heavily studied.

This project was supported by APINIC foundation.

An Anomaly IDS creates a normal profile and identifies any variations from it as a suspicious event. It can identify known and zero-day attacks with less effort to construct its profile than a Signature IDS. Figure 1 summaries the major categories of anomaly detection approaches. [2]

Among these categories, both the classification-based shallow learning and deep learning methods have shown promising results using existing public datasets. Based on the data analysis outcomes, the following methods were selected to produce performance reference and comparison: Random Forest (RF), Support Vector Machine (SVM), K Nearest Neighbour (KNN), Multi-Layer Perceptron Neural Network (MLP-NN), Long Short-Term Memory Convolutional Neural Network (LSTM-CNN).

In recent literature, Reinforcement Learning method has been proposed in a few studies to explore how it can improve the capability of deep neural network [8].

This project designed and experimented an Adversarial Reinforcement Learning (ARL) model to explore its ability in detecting emerging attacks.

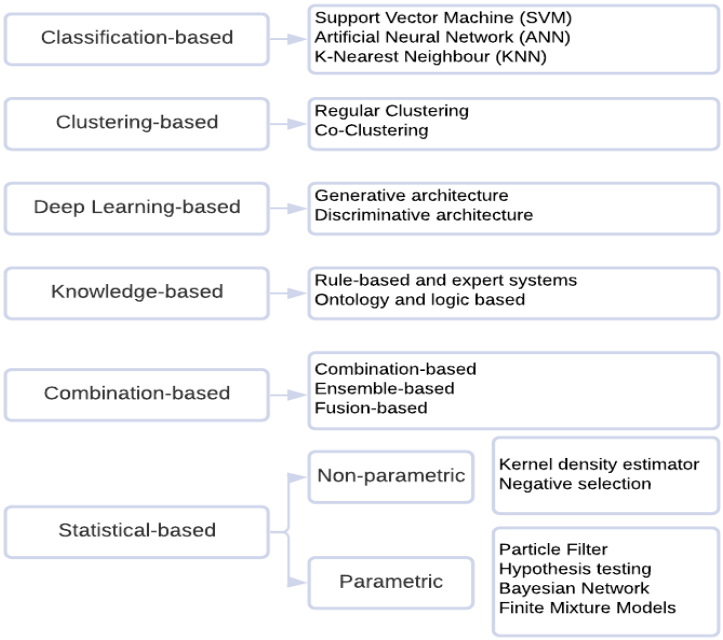


Figure 1 Anomaly Detection Approach Categories

PHASE II DESIGN AND EXPERIMENT VARIOUS MACHINE LEARNING MODELS

EXPERIMENT ENVIRONMENT:

The experiments were carried out in a cloud-based environment. Azure pipeline was used to setup DevOp environment. The bidirectional traceability is established across requirements, stories, repositories, test cases and test results. The environment allows multiple users to access the resources and conduct development collaboratively.

DataOps pipelines were setup to apply various source data ingestion to the process of model training and validation. Researchers can choose suitable DataOps pipeline for each experimentation.

This project was supported by APINIC foundation.

Experiment processes

The experiments were carried out through the following steps.

- Prepared the selected datasets NSL-KDD and CICIDS2017 respectively.
- Designed the machine learning models applying the selected methods and the respective dataset.
- Conducted model training and experiments using the prepared data.
- Designed an Adversarial Reinforcement Learning (ARL) model structure for CIC-IDS2017 dataset.
- Conducted ARL model training and experiments using the prepared CIC-IDS2017 data.
- Analysed the anomaly detection and category classification results.

A typical machine learning model development and experiment process is demonstrated in the diagram below. This project applies the same process to conduct the work.

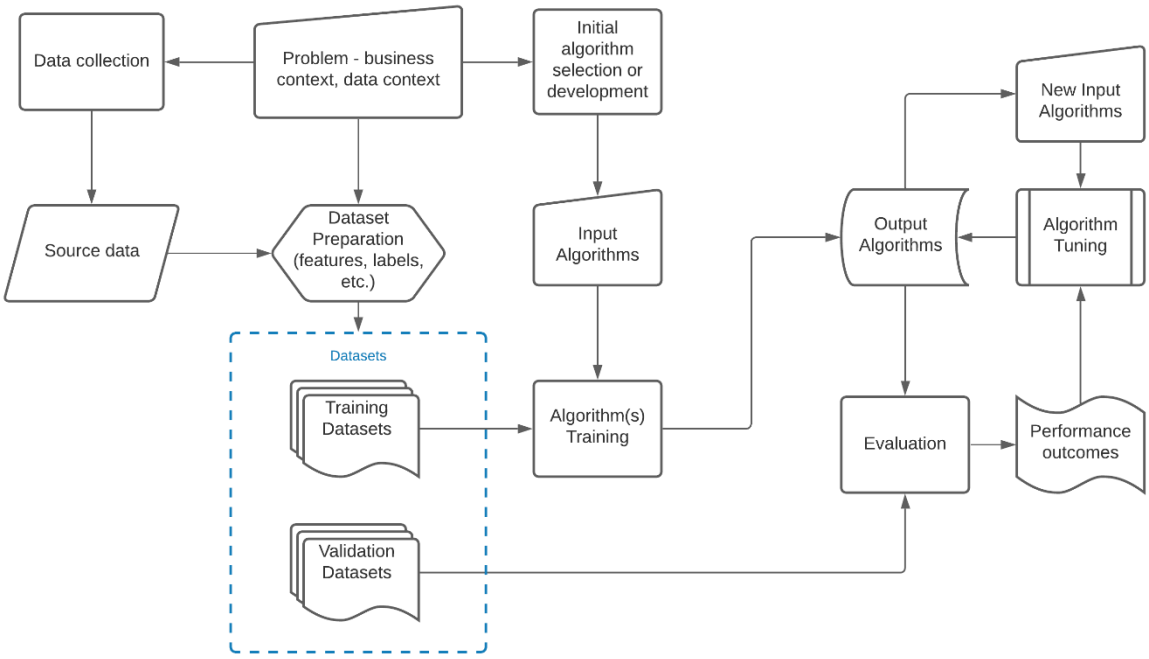


Figure 2 Model Training and Experiment Process

DATASET ANALYSIS AND PREPARATION

Data preparation is a significant step for machine learning methods. The network data extracted from network traffic includes noisy or irrelevant information, missing or duplicated data values, which impact the performance of detection model for detecting anomaly. In order to design and architect the models, the datasets were carefully analysed so that the characters, the feature structures, and the distribution shapes were understood. Suitable data cleansing and processing operations were designed and conducted on the selected datasets respectively.

NSL-KDD Dataset

Data availability

This project was supported by APINIC foundation.

NSL-KDD, which is an updated version of KDD'99 dataset, is downloaded from https://www.unb.ca/cic/datasets/nsl.html. 8 files are available in the dataset:

1. KDDTrain+.ARFF: The full NSL-KDD train set with binary labels in ARFF format
2. KDDTrain+.TXT: The full NSL-KDD train set including attack-type labels and difficulty level in CSV format
3. KDDTrain+_20Percent.ARFF: A 20% subset of the KDDTrain+.arff file
4. KDDTrain+_20Percent.TXT: A 20% subset of the KDDTrain+.txt file
5. KDDTest+.ARFF: The full NSL-KDD test set with binary labels in ARFF format
6. KDDTest+.TXT: The full NSL-KDD test set including attack-type labels and difficulty level in CSV format
7. KDDTest-21.ARFF: A subset of the KDDTest+.arff file which does not include records with difficulty level of 21 out of 21
8. KDDTest-21.TXT: A subset of the KDDTest+.txt file which does not include records with difficulty level of 21 out of 21

ARFF-formatted files include 'attribute' in their header, which are description of columns in the dataset. In this analysis, KDDTrain+.ARFF and KDDTest+.ARFF

1) files are used as training and testing data, respectively.

Data description

There are 125,973 training records in the training data and 22,544 testing records in the testing data. In each dataset, there are 42 columns, where the last column represents labels ('normal'/'anomaly') for the records. 41 attributes were recorded and their descriptions are listed in Appendix 1. List of all attributes included in NSL-KDD data.

In training data, 67,343 records are labelled 'normal' and 58,630 records are 'anomaly', which shows it's a well-balanced dataset. In testing data, there are 12,833 'normal' records and 9,711 'anomaly' records.

Data Preparation

1. Encoding categorical data: Three columns: 'protocol_type', 'service' and 'flag' are categorical data with more than 2 categories in each column. These categories are encoded by One Hot Encoding from Python in training data and testing data, respectively. Due to the fact that fewer categories are available in testing data for 'service' data, columns associated with these 6 missing categories are set to 0 in the testing data. Associated original columns are then removed and replaced by these dummy columns.

2. Normalisation: Histograms of columns with numeric data are illustrated in Figure 3, where it is clear that majority data in these columns are 0. In order to minimize the effect of absolute values to classification model, training data are normalized to standardized data with mean 0 and standard deviation 1.

Testing data are normalized based on mean and standard deviation values estimated from original training data.

This project was supported by APINIC foundation.

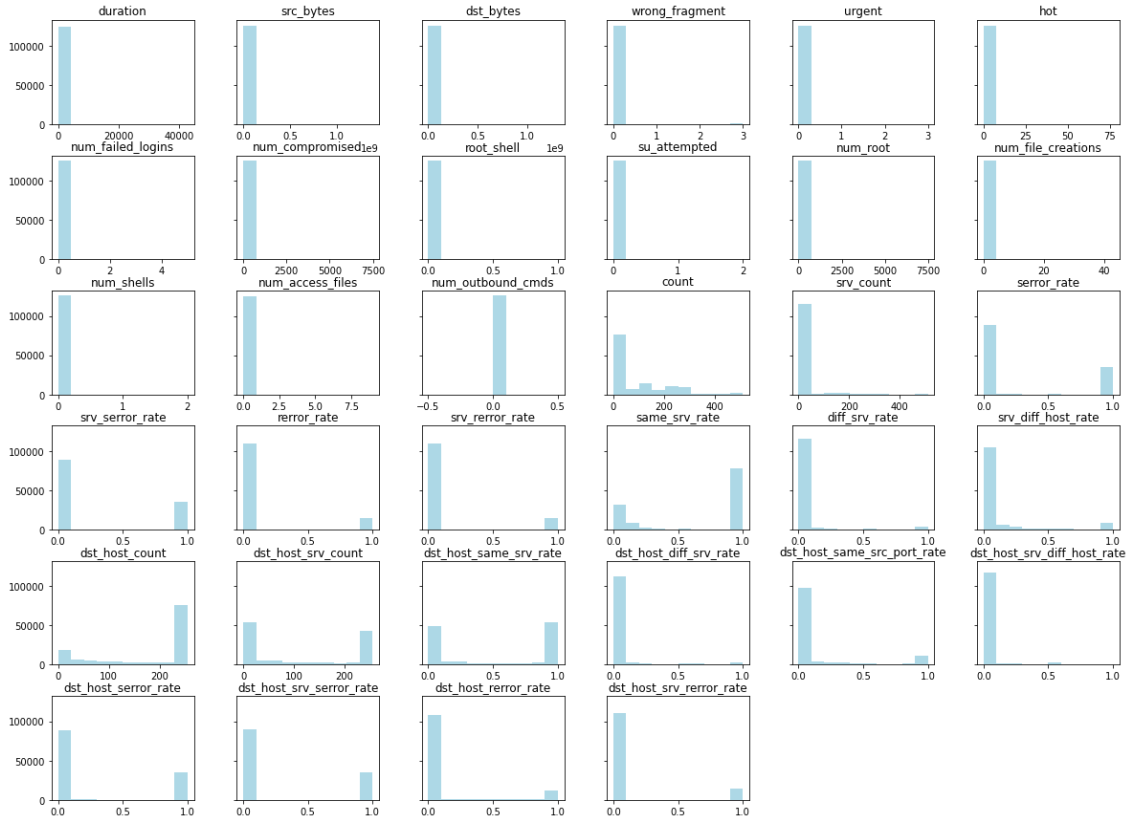


Figure 3 Histogram of numeric columns in NSL-KDD training dataset

CICIDS2017 DATASET

Data availability and description

CICIDS 2017 data consists of 8 data files collected from Monday to Friday in a week. In summary, 14 categories are included in the data, which are

- BENIGN: 2,830,743 records,
- FTP-Patator: 7,938 records
- SSH-Patator: 5,897 records
- DoS Hulk: 231,073 records
- DoS GoldenEye 10,293 records
- DoS slowloris: 5,796 records
- DoS Slowhttptest: 5,499 records
- Heartbleed: 11 records
- Web Attack Brute Force: 1,507 records
- Web Attack XSS: 652 records
- Web Attack Sql Injection: 21 records
- Infiltration: 36 records
- Bot: 1,966 records
- PortScan: 158,930 records
- DdoS: 128,027 records

Details of the distributions of these records in each file are listed in the Table 2 below.

This project was supported by APINIC foundation.

File Name	Category	Number of Records
Monday-WorkingHours.pcap_ISCX.csv	Benign	529918
Tuesday-WorkingHours.pcap_ISCX.csv	Benign FTP-Patator SSH-Patator	432074 7938 5897
Wednesday-WorkingHours.pcap_ISCX.csv	Benign DoS Hulk DoS GoldenEye DoS slowloris DoS Slowhttptest Heartbleed	440031 231073 10293 5796 5499 11
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Benign Web Attack Brute Force Web Attack XSS Web Attack Sql Injection	168186 1507 652 21
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Benign Infiltration	288566 36
Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign Bot	189067 1966
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	PortScan Benign	158930 127537
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	DDoS Benign	128027 97718

Table 2 CICIDS2017 data profile

Data Preparation

1. Sub dataset selection

The full CICIDS2017 dataset is very large in size which makes practical model training and testing very difficult. In this work, we applied stratified random

sampling method to extract 20% of the full dataset. However, anomaly categories have a lot less records compared with benign data. In particular, some

categories only have very small number of records. This may cause data imbalance. In order to resolve the imbalance to support better learning capability, all

the records of anomalies were added to produce sub dataset.

2. Training and Testing data

Stratified sampling method was used to divide the sub dataset into 80% for training and 20% for testing.

This project was supported by APINIC foundation.

3. Handling missing value and columns with zeros

Two columns ‘Flow Bytes/S’ and ‘Flow Packets/s’ have missing values in them. These values are replaced by their means in each group, respectively. Columns

'Bwd PSH Flags','Bwd URG Flags'. 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk','Bwd Avg Packets/Bulk' and 'Bwd

Avg Bulk Rate' have only 0s in them. They are therefore removed in the preparation step.

4. Data normalization

All features were normalized by MinMaxScaler from scikit package in Python. Training data and testing data were normalized separately.

MODEL CONSTRUCTION

The machine learning models were developed to not only detect anomaly behaviours, but also identify the categories of the anomaly behaviours.

Shallow Learning models

Random Forest (RF)

RF is an ensemble method which combines lots of individual decision trees, as illustrated in the Figure 4 below. Each individual tree employs some of the features and spits out a class prediction. The class that receives the most votes becomes RF model’s final prediction. Two RF models were developed using NSL-KDD and CIC-IDS2017 datasets respectively.

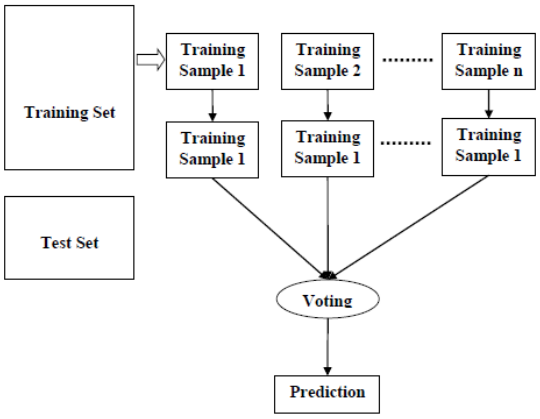


Figure 4 Random Forest Algorithm Steps

Singular Vector Machine (SVM)

SVM works by finding out the separating hyperplane which maximizes the margin between two classes. A SVM model was developed using NSL-KDD to provide comparison data against RF method. A non-linear soft-margin classifier was adopted in this model.

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1-y_i(\mathbf{w}^T\mathbf{x}_i - b)\right)\right] + \lambda\|\mathbf{w}\|^2$$

K Nearest Neighbour (KNN)

This project was supported by APINIC foundation.

KNN is a non-parametric classification method among which an object is classified by a vote of its K nearest neighbours. It is renowned for its classification abilities in high-dimensional problems. Minkowski Distance calculation as the formular below was adopted in model construction. A KNN model was developed using CIC-IDS2017 dataset.

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

Deep learning models

Multi-Layer Perceptron Neural Network (MLP-NN)

MLP-NN is a class of feedforward Artificial Neural Network (ANN). Our models comprise multiple hidden layers of nonlinearly-activating neurons. Learning occurs in the neurons by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. Two MLP-NN models were developed using the NSL-KDD and CIC-IDS2017 datasets respectively.

The diagram Figure 5 illustrates the high-level architecture of a general Artificial Neural Network algorithm.

Convolutional Neural Network Long Short-Term Memory (CNN-LSTM)

The CNN-LSTM model developed in this project is composed of a convolution 1d layer (input), a LSTM layer, a dropout layer and an output layer. It was trained with NSL-KDD dataset to compare with MLP-NN model.

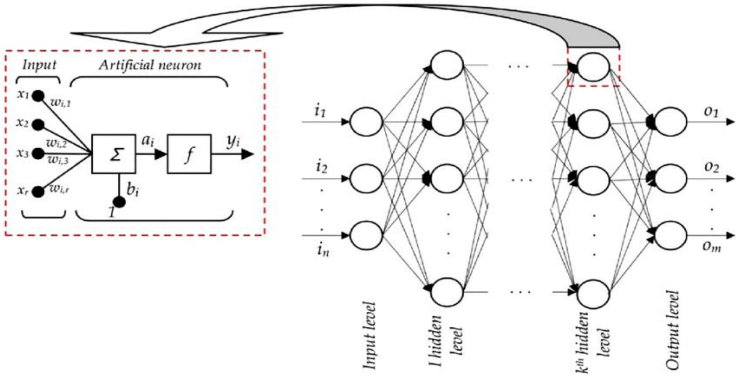


Figure 5 Artificial Neural Network

Reinforcement Learning model

Adversarial Reinforcement Learning (ARL)

An ARL model was developed which includes a classifier agent and an environment agent, both of which are Deep Q Networks (DQN) consisting of MLP-NN models with three hidden layers in the classifier agent and one hidden layer in the environment agent. Each hidden layer has 100 neurons in it. An optimal Q-function was adopted, which obeys the Bellman optimality equation:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

The model works by training both agents at the same time, and iteratively rewards the classifier agent once a correct classification is performed. The ARL model was trained and experimented using the CIC-IDS2017 dataset.

ARL Model Training

This project was supported by APINIC foundation.

During the process of model training, the environment agent used records from CIC-IDS2017 dataset to create attacks while the classifier agent was rewarded when it correctly identified the category of an attack.

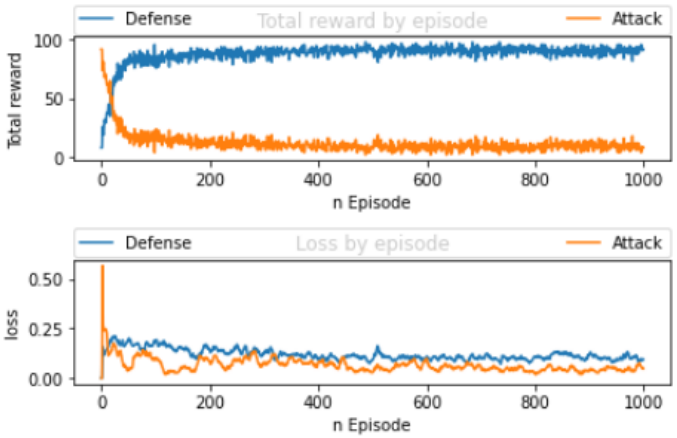
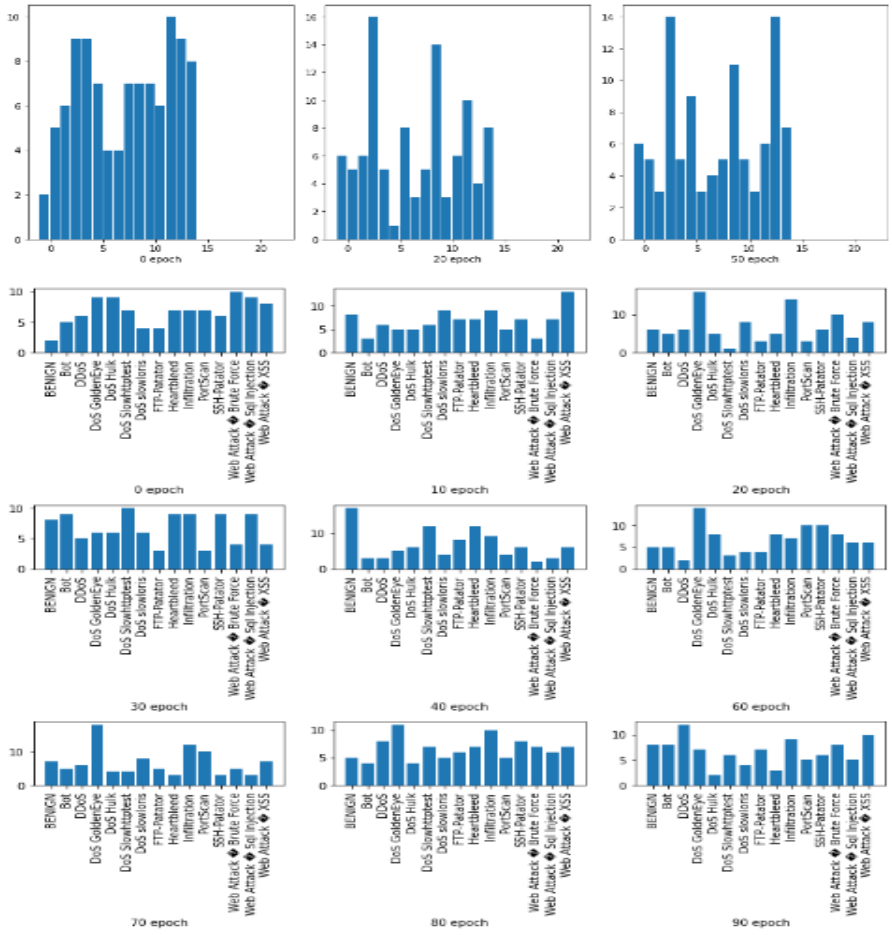


Figure 6 ARL model training - total reward and loss of two agents

Figure 6 shows the classifier agent improving its ability to defend the type of attack each successive episode while the environment agent becomes more unsuccessful making an attack. The classifier and environment agents exhibit expected behaviour of total reward converges to a value. The environment agent used a varied type of attacks for each episode. Figure 7 shows the distribution of the attacks across all the categories in each episode. The distribution shapes varied throughout the process.



This project was supported by APINIC foundation.

Figure 7 Distribution of attacks at different epoch levels

EXPERIMENT RESULTS

The key objective of the model development and experiments is to explore the suitability and potential of Reinforcement Learning method in detecting network emerging attacks.

The models were trained and tested using traditional machine learning experiment approach.

Analysis and evaluation metrics

Parameters

True positive (TP): number of harmful applications correctly classified.

True negative (TN): number of benign applications correctly classified.

False positive (FP): number of benign applications misclassified as harmful. It is regarded as the main drawback of classification methods.

False negative (FN): number of harmful applications misclassified as benign.

Performance metrics

Name	Description
Accuracy (ACC)	Percentage of correct predictions (positive and negative). (TP + TN) / (TP + TN + FP + FN)
Precision	Precision: percentage of correct positives over the total number of positives identified. TP / (TP + FP)
Detection Rate (DR)	The detection rates are evaluated using the Area Under the Curve (AUC) of the receiver operating characteristics (ROC). The detection latency is evaluated by measuring the mean computing time to detect whether a data sample is an intrusion. DR-attack = TP / (TP + FN)
Sensitivity	Sensitivity measures the proportion of attack profiles correctly identified. TP / (TP + FN)
Specificity	Specificity measures the percent of authentic profiles correctly classified, thus providing insight as to the portion of the original authentic profiles that are used for prediction.
F-measures	From precision and recall, this parameter measures the accuracy of the method 2 * {(Precision * Sensitivity) / (Precision + Sensitivity)}
False Alarm Rate (FAR)	False alarms are the benign instances incorrectly classified over the total number of benign samples. FP / TN + FP
False Negative and Positive	False positives over the total number of positives identified FP / TP + FP False negative over the total number of positives identified FN / FN + TN
Miss Rate	Harmful instances incorrectly classified over the total number of harmful samples FN / (TP + FN)

This project was supported by APINIC foundation.

Error Rate	Incorrectly classified instances over the total (FP + FN) / (TP + TN + FP +FN)

This project measures the performance of the developed machine learning models in two scenarios, 1) the classification of each anomaly category, 2) the binary detection of anomalies from benign records. The measurement was scaled between 0 to 1 when 1 was 100%.

Detection performance when using NSL-KDD dataset

The four models designed by Random Forest (RF), Support Vector Machine (SVM), Multi-Layer Perceptron Neural Network (MLP-NN), and Convolutional Neural Network Long Short-Term Memory (CNN-LSTM) methods respectively were trained and tested using NSL-KDD dataset.

The key performance metrics were measured against the binary detection and listed in the table below.

Model	Accuracy	Precision	Specificity	Sensitivity	F-measure
RF	0.79	0.97	0.97	0.68	0.80
SVM	0.79	0.93	0.93	0.69	0.79
MLP-NN	0.79	0.92	0.92	0.69	0.79
CNN-LSTM	0.79	0.95	0.95	0.68	0.80

Table 3 Binary detection performance - NSL-KDD

The results in this table showed the performance of these four models was very similar to each other, among which RF model showed the best overall performance. There are rooms to improve the performance of these models. For the purpose of providing benchmark comparison, these measures were sufficient to show the effectiveness of these methods.

Detection results when using CIC-IDS2017 dataset

The four models designed by Random Forest (RF), K Nearest Neighbour (KNN), Multi-Layer Perceptron Neural Network (MLP-NN), and Adversarial Reinforcement Learning (ARL) methods respectively were trained and experimented using CIC-IDS2017 dataset.

The precision and F-measure of the four models classifying the fifteen anomaly categories were measured. The results are presented in the following table.

Category	KNN		RF		MLP-NN		ARL		Number of cases
	Precision	F1-score	Precision	F1-score	Precision	F1-score	Precision	F1-score	
Benign	0.90	0.94	0.99	0.99	0.79	0.87	0.94	0.85	90924
Bot	0.92	0.73	0.78	0.41	1.00	0.04	0.02	0.04	393
DDoS	1.00	0.99	1.00	1.00	0.99	0.91	0.91	0.59	25606
DoS Goldeneye	0.99	0.96	1.00	0.98	0.91	0.33	0.49	0.58	2059
Dos Hulk	0.99	0.90	1.00	1.00	0.97	0.82	0.69	0.80	46215
Dos SlowHTTPTest	0.90	0.94	0.99	0.96	0.97	0.12	0.25	0.39	1100
Dos SlowLoris	0.99	0.94	1.00	0.98	1.00	0.56	0.21	0.29	1159
FTP-Patator	1.00	1.00	1.00	1.00	0.00	0.00	0.66	0.79	1588
Heartbleed	1.00	0.67	1.00	0.67	0.00	0.00	0.00	0.00	2

This project was supported by APINIC foundation.

Infiltration	0.50	0.22	1.00	0.73	0.00	0.00	0.00	0.00	7
PortScan	0.97	0.97	1.00	1.00	0.91	0.94	0.93	0.95	31786
SSH-Patator	1.00	0.68	1.00	1.00	0.00	0.00	0.39	0.43	1180
Web attack Brute force	0.79	0.59	0.74	0.73	0.00	0.00	0.00	0.00	301
Web attack SQL injection	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4
Web attack XSS	0.43	0.13	0.26	0.32	0.00	0.00	0.00	0.00	130

Table 4 Anomaly category classification performance - CIC-IDS2017

From these results, it is obvious that the Shallow Learning RF and KNN models had better performance while the overall performance of all the models was good.

The binary detection scenarios grouped the data to 'benign' and 'anomaly' only, where the 'anomaly' group included all the categories not belonging to the 'benign' group. Table 5 illustrates the binary detection performance measures of the four models.

It is clear that RF model had the best performance overall. KNN model also showed strong performance which was similar to RF. The performance of both MLP-NN and ARL models under binary scenario was better than classifying each category but less desired compared with RF and KNN models.

Metrics	KNN	RF	MLP-NN	ARL
Accuracy	0.95	1.00	0.87	0.87
Precision	0.91	0.99	0.79	0.96
Specificity	0.90	0.99	0.79	0.94
Sensitivity	0.99	1.00	0.97	0.84
F-measure	0.95	0.95	0.87	0.89
Detection Rate	0.99	1.00	0.97	0.84
False Alarm Rate	0.10	0.01	0.21	0.06
Miss Rate	0.01	0.00	0.03	0.16
Error Rate	0.05	0.00	0.13	0.13
False Positive Rate	0.09	0.01	0.21	0.04
False Negative Rate	0.01	0.00	0.03	0.23

Table 5 Binary detection performance CIC-IDS2017

This project was supported by APINIC foundation.

Meanwhile, the feature importance was measured when developing the RF model. The following Figure 8 includes the 20 most important features that RF model considered in its decision-making process. It shows the 'Packet Length Variance' and 'Packet length Std' were two most important features.

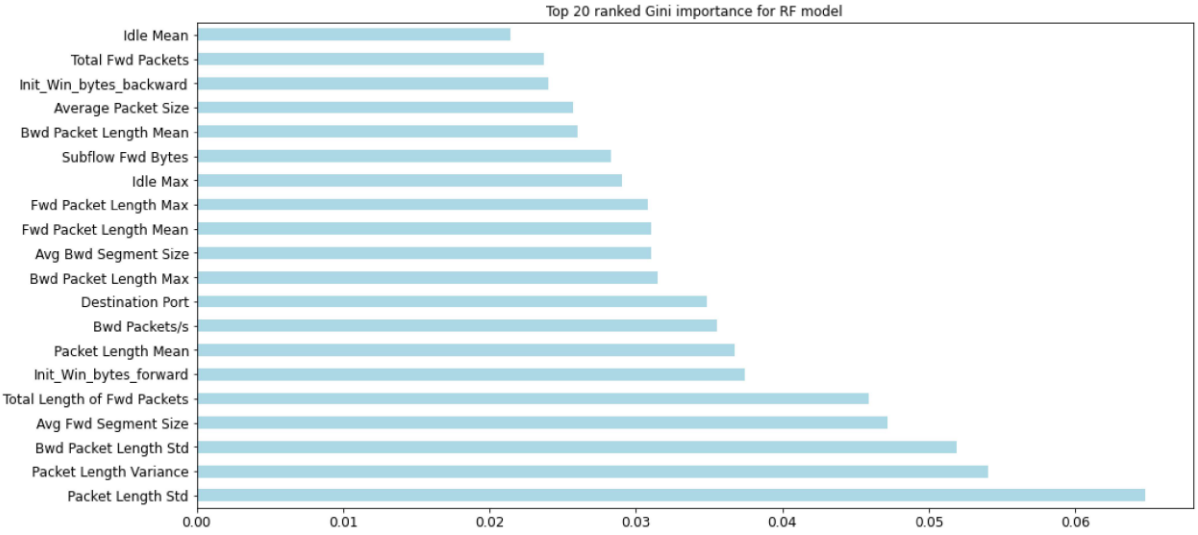


Figure 8 Gini feature importance - RF model

There were two models developed by RF method using two different datasets respectively. The performance of these two models is compared in the Table 6 below.

Dataset	Accuracy	Precision	Specificity	Sensitivity	F-measure
NSL-KDD	0.79	0.97	0.97	0.68	0.80
CIC-IDS2017	1.00	0.99	0.99	1.00	0.95

Table 6 RF model performance against the two datasets

There were two models developed by MLP-NN method using two different datasets respectively. The performance of these two models is compared in the Table 7 below.

Dataset	Accuracy	Precision	Specificity	Sensitivity	F-measure
NSL-KDD	0.79	0.92	0.92	0.69	0.79
CIC-IDS2017	0.87	0.96	0.94	0.84	0.89

Table 7 MLP-NN model performance against the two datasets

Both RF and MLP-NN models showed obvious variations of performance measures between the two datasets. This indicates these machine learning models are sensitive to different data structure.

PHASE III EXPERIMENT A NOVEL EVALUATION METHOD ON THE TRAINED MODELS

EVALUATION APPROACH

The purpose of the evaluation process conducted in this project beyond the traditional ML model experiment processes is to explore an approach which simulates the real environment where a ML model is implemented

This project was supported by APINIC foundation.

to detect network-attacks in real-time. So that the evaluation results can indicate the effectiveness of a ML model in real-time detection.

In real environments the network traffic data streams across network components. However, processing streaming data in real-time as part of system operation can be expensive at the risks of degrading the system performance. The proposed evaluation method assumes the network traffic data and system logs can be captured in batches using the similar method as how the CICIDS2017 was constructed [19]. The network traffic can be extracted and constructed at a certain interval.

An evaluation pipeline was established in a cloud-based environment which allows a simulator to send continuous batches to pipeline endpoints at a configurable interval. The data flows automatically trigger the data preparation processes and execute the trained models. The evaluation pipeline can be deployed in any real network environments. Figure 9 demonstrates the concept of the evaluation pipeline.

Limitation

Within the scope of this project, the real-time network traffic is not collected or constructed. A public dataset that was produced with relatively comprehensive profile and emerging attacks is to be applied. The continuous batch files were generated by applying random sampling method on the adopted public datasets.

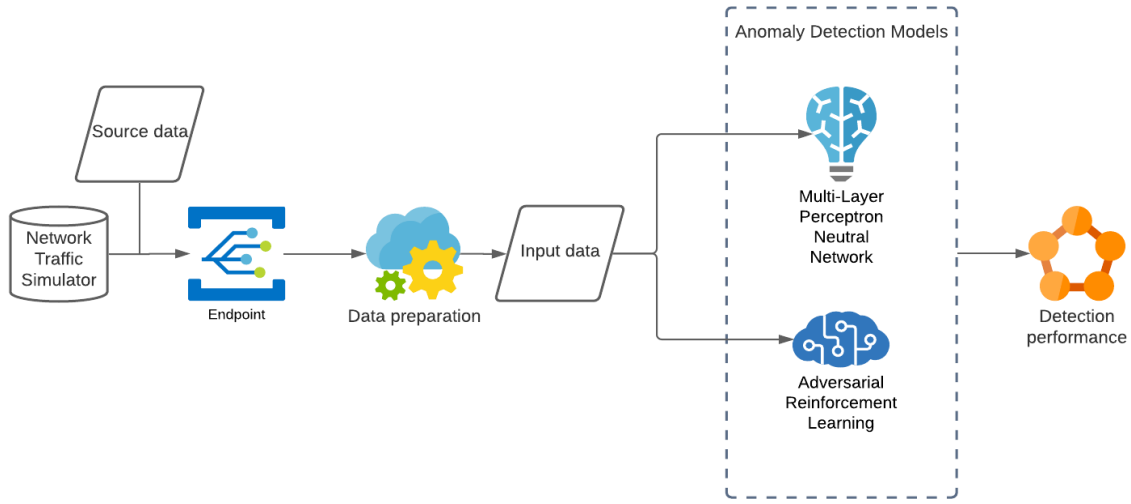


Figure 9 AI anomaly detection model evaluation pipeline

Evaluation Scenario 1

In order to understand how well ML models works in real environments, this evaluation scenario focuses on implementing the trained ML models to detect anomalies in each batch file which has varieties of anomaly density and category distribution.

This evaluation scenario measures the classification and detection performance of the models across continuous small network data batches. The results are compared across the batches to observe the performance consistency and stability.

Evaluation Scenario 2

This project was supported by APINIC foundation.

This evaluation scenario focuses on the comparison of the robustness of the trained ML models when encountering changes or noise in feature space. Ideally, this evaluation could be conducted with real network data noise and uncertainties. However, real network data construction is out of the scope of this project.

The evaluation scenario 2 applied one dataset for training and a different dataset for evaluation. Though this approach imposes the risk of causing the ML models ineffective, it may still compare the different robustness in various models.

Evaluation Datasets

Running anomaly detection models over real labelled network traces with a comprehensive and extensive set of intrusions and abnormal behaviour is the most idealistic methodology for testing and evaluation. This itself is a significant challenge. As network behaviours and patterns change and intrusions evolve, it has very much become necessary to move away from static and one-time datasets towards more dynamically generated datasets, which not only reflect the traffic compositions and intrusions of that time, but are also modifiable, extensible, and reproducible. Within the scope of this project, we applied CICIDS2017 dataset, which is close to realistic network data with zero-attacks, to simulate the realistic network data for evaluation.

Other than CICIDS2017 dataset, another dataset CES-CICIDS2018 was produced by the same organisation - Canadian Institute for Cybersecurity (CIC). This dataset used the notion of profiles to generate datasets in a systematic manner, which contains detailed descriptions of intrusions and abstract distribution models for applications, protocols, or lower-level network entities. Both the CICIDS2017 and CSE-CICIDS2018 datasets include seven different attack scenarios: Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside. The datasets include the captures network traffic and system logs of each machine, along with 80 features extracted from the captured traffic using CICFlowMeter [19].

At the first glance, these two datasets have very similar feature structure. However, when we analysed in more depth on the constructed data, CESCICIDS2018 dataset has different feature and anomaly distribution shape compared with the CICIDS2017 dataset. The detailed analysis of the data extraction and construction processes of these two datasets are not included in the scope of this project.

This project used CICIDS2017 dataset to conduct the evaluation scenario 1, to evaluate the consistency and stability of the trained ML models in cloud-based evaluation pipelines processing the randomly sampled batch flows. CES-CICIDS2018 dataset was adopted to measure the robustness of the four ML models trained by CICIDS2017 dataset to explore the capabilities of these models when facing feature and anomaly changes.

Evaluation Process

The steps of experimenting the evaluation method and pipelines were as the following:

1. Applied random sampling method on CICIDS2017 and CSE-CICIDS2018 data which is outside of the model training data to generate the network data batch files. Each batch file contains 20,000 records.

2. Built a simulator to send the batch network data files at a configurable interval to the evaluation pipeline endpoints.

3. Implemented an evaluation pipeline on the TeleMARS cloud-based platform to automatically

    1. receive the incoming batch flows;
    2. prepare the incoming batch files to get ready to feed the models; and

This project was supported by APINIC foundation.

3. execute the trained machine learning models on the prepared data to detect anomalies in each batch

EVALUATION RESULTS

Performance consistency and stability

The tables below show the performance measures of classifying anomaly categories over randomly picked three batch files, as well as the binary detection performance measures of each model.

Bath file 1

Category	KNN		RF		MLP-NN		ARL		Number of cases
	Precision	F1-score	Precision	F1-score	Precision	F1-score	Precision	F1-score	
Benign	1.00	0.99	1.00	1.00	1.00	0.99	0.99	0.92	8978
Bot	0.88	0.92	0.93	0.91	0.84	0.76	0.11	0.19	46
DDoS	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.99	2549
DoS Goldeneye	1.00	1.00	0.99	1.00	1.00	1.00	0.78	0.87	193
Dos Hulk	1.00	1.00	1.00	1.00	0.99	1.00	0.95	0.96	4560
Dos SlowHTTPTest	0.98	0.99	1.00	1.00	0.98	0.98	0.68	0.80	127
DOS SlowLoris	0.99	1.00	1.00	1.00	0.97	0.98	0.51	0.67	116
PortScan	0.96	0.98	1.00	1.00	0.98	0.99	0.94	0.96	3141
Brute force	0.68	0.75	0.82	0.78	0.78	0.88	0.01	0.02	36
Web attack XSS	0.30	0.30	0.26	0.29	1.00	0.18	0.10	0.18	10

Table 8 Classification of each category in batch file 1

Metrics	KNN	RF	MLP-NN	ARL
Accuracy	0.99	1.00	0.99	0.93
Precision	1.00	1.00	1.00	0.99
Specificity	1.00	1.00	1.00	0.90
Sensitivity	0.99	1.00	0.99	0.90
F-measure	0.99	1.00	0.99	0.94
Detection Rate	0.99	1.00	0.99	0.90
False Alarm Rate	0.00	0.00	0.00	0.01

This project was supported by APINIC foundation.

Miss Rate	0.01	0.00	0.01	0.01
Error Rate	0.01	0.00	0.01	0.07
False Positive Rate	0.00	0.00	0.00	0.01
False Negative Rate	0.02	0.00	0.01	0.14

Table 9 Binary performance measures for batch file 1

Batch file 2

Category	KNN		RF		MLP-NN		ARL		Number of cases
	Precision	F1-score	Precision	F1-score	Precision	F1-score	Precision	F1-score	
Benign	1.00	0.99	1.00	1.00	1.00	0.99	0.99	0.92	9007
Bot	0.97	0.93	1.00	0.94	1.00	0.85	0.08	0.15	38
DDoS	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.99	2500
DoS Goldeneye	0.99	0.99	1.00	1.00	0.99	0.99	0.82	0.89	216
Dos Hulk	1.00	1.00	1.00	1.00	1.00	1.00	0.96	0.96	4530
Dos SlowHTTPTest	1.00	1.00	1.00	1.00	0.99	0.99	0.74	0.84	124
DOS SlowLoris	0.99	1.00	1.00	1.00	0.98	0.98	0.57	0.72	127
Infiltration	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
PortScan	0.97	0.98	1.00	1.00	0.98	0.99	0.94	0.96	3136
Brute force	0.57	0.67	0.77	0.78	0.63	0.77	0.07	0.1	34
SQL injection	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
Web attack XSS	0.21	0.19	0.53	0.50	0.0	0.0	0.16	0.28	17

Table 10 Classification of each category in batch file 2

Metrics	KNN	RF	MLP-NN	ARL
Accuracy	0.99	1.00	1.00	0.93
Precision	1.00	1.00	1.00	0.99
Specificity	1.00	1.00	1.00	0.99
Sensitivity	0.99	1.00	0.99	0.90
F-measure	0.99	1.00	1.00	0.94
Detection Rate	0.99	1.00	0.99	0.90

This project was supported by APINIC foundation.

False Alarm Rate	0.00	0.00	0.00	0.01
Miss Rate	0.01	0.00	0.01	0.10
Error Rate	0.01	0.00	0.00	0.07
False Positive Rate	0.00	0.00	0.00	0.01
False Negative Rate	0.01	0.00	0.01	0.13

Table 11 Binary performance measures for batch file 2

Batch file 3

Category	KNN		RF		MLP-NN		ARL		Number of cases
	Precision	F1-score	Precision	F1-score	Precision	F1-score	Precision	F1-score	
Benign	1.00	0.99	1.00	1.00	1.00	0.99	0.99	0.92	8929
Bot	0.86	0.91	0.91	0.93	0.85	0.77	0.08	0.15	32
DDoS	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.99	2556
DoS Goldeneye	0.98	0.99	0.99	0.99	1.00	0.99	0.76	0.86	218
Dos Hulk	1.00	1.00	1.00	1.00	0.99	1.00	0.96	0.96	4589
Dos SlowHTTPTest	0.99	1.00	0.98	0.99	1.00	1.00	0.73	0.83	106
DOS SlowLoris	0.99	0.97	0.99	0.98	0.97	0.97	0.55	0.69	117
PortScan	0.96	0.98	1.00	1.00	0.98	0.99	0.93	0.96	3148
Brute force	0.75	0.81	0.78	0.72	0.78	0.87	0.04	0.06	37
Web attack XSS	0.27	0.30	0.14	0.17	0.0	0.0	0.10	0.18	9

Table 12 Classification of each category in batch file 3

Metrics	KNN	RF	MLP-NN	ARL
Accuracy	0.99	1.00	0.99	0.93
Precision	1.00	1.00	1.00	0.99
Specificity	1.00	1.00	1.00	0.99
Sensitivity	0.99	1.00	0.99	0.90
F-measure	0.99	1.00	0.99	0.94
Detection Rate	0.99	1.00	0.99	0.90
False Alarm Rate	0.00	0.00	0.00	0.01
Miss Rate	0.01	0.00	0.01	0.10

This project was supported by APINIC foundation.

Error Rate	0.01	0.00	0.01	0.07
False Positive Rate	0.00	0.00	0.00	0.01
False Negative Rate	0.02	0.00	0.01	0.14

Table 13 Binary performance measures for batch file 3

Processing time

The table below records the pipeline data preparation time and model processing time of running a model on a single batch file. The results are compared among the four models over four randomly picked batches.

Model	Batch file no.	Pipeline data preparation duration (mm:ss)	Model processing duration (mm:ss)
RF	1	00:26	00:01
KNN	1	00:17	08:55
MPL-NN	1	00:19	00:02
ARL	1	00:17	00:02
RF	2	00:18	00:01
KNN	2	00:32	08:33
MPL-NN	2	00:42	00:01
ARL	2	00:16	00:02
RF	3	00:20	00:01
KNN	3	00:17	08:33
MPL-NN	3	00:22	00:01
ARL	3	00:27	00:02
RF	4	00:27	00:01
KNN	4	00:18	08:18
MPL-NN	4	00:30	00:01
ARL	4	00:43	00:01

Table 14 Evaluation processing time

Robustness of the ML models

The table below compares the binary detection performance across the four models when sending the batch files generated from CES-CICIDS2018 data through the evaluation pipeline.

Metrics	KNN	RF	MLP-NN	ARL
Accuracy	0.53	0.35	0.64	0.60
Precision	0.38	0.14	0.53	0.57
Specificity	0.35	0.28	0.43	0.36
Sensitivity	0.94	0.84	0.98	0.83
F-measure	0.54	0.24	0.68	0.67
Detection Rate	0.94	0.84	0.98	0.83
False Alarm Rate	0.65	0.72	0.57	0.64

This project was supported by APINIC foundation.

Miss Rate	0.06	0.16	0.02	0.17
Error Rate	0.47	0.65	0.36	0.40
False Positive Rate	0.63	0.86	0.47	0.43
False Negative Rate	0.07	0.07	0.03	0.32

Table 15 Binary detection performance measures for batch file from CES-CIC-IDS2018 data

FINDINGS

OBSERVATION SUMMARY

The performance metrics of RF and MLP-NN models against the two datasets were compared in "Table 6 RF model performance against the two datasets" and "Table 7 MLP-NN model performance against the two datasets". There are obvious variations of performance measures between the two datasets while the overall performance of anomaly detection is good.

The binary detection performance results recorded in "Table 3 Binary detection performance - NSL-KDD" and "Table 5 Binary detection performance - CICIDS2017" showed that the Shallow Learning models such as RF and KNN performed stronger than the MLP-NN and ARL models. RF model had the best overall performance in all the experiments.

The results of the anomaly category classification performance in "Table 4 Anomaly category classification performance" showed the Shallow Learning RF and KNN models were more effective in classifying the anomaly categories with small number of records.

The evaluation results recorded in the tables from Table 8 to Table 13 were consistent cross three randomly selected batch files. There were insignificant fluctuations in the results of classification of a couple of categories which have small number of records. This is expected behaviour of ML models. The results of binary detection performance were highly consistent.

Table 15 shows the models trained with CICIDS2017 data performed poorly in detecting the anomalies in CES-CIC-IDS2018 data. MPL-NN model had better overall performance.

FINDINGS

1. The Shallow Learning RF model showed the best overall performance in detecting emerging attacks using traditional machine learning experiment approach, in particular stronger in classifying the anomaly categories that have only small number of records. KNN model shows similar strong capabilities.

2. The Reinforcement Learning ARL model had good performance but did not show any advantage over other models. However, there are rooms for Deep Learning and Reinforcement Learning models to improve their prediction accuracy and detection sensitivity. Neural Network architecture could be further adapted by modifying the layers, the number of neurons or hyperparameters in each layer, and the dependencies between the neurons and the layers. The architecture reconstruction has chance to lead to the improvements of anomaly detection performance.

This project was supported by APINIC foundation.

3. The differences of performance between the model experiments using NSL-KDD dataset and CIC-IDS2017 dataset respectively showed that ML models are sensitive to data structure.

4. The evaluation results proved the detection performance of the ML models stay consistent and stable when data volume, anomaly density and category distribution change.

5. The evaluation method and the pipelines can be applied to support future real-time detection research.

6. The RF, MPL-NN and ARL models were highly efficient in execution time and resource balance when processing small batch files. However, KNN consumed a lot larger resource and took significantly longer processing time. This indicates the implementation of KNN models in real network environment may be expensive.

7. When the testing data feature space is different from the training dataset, ML models do not perform well in general. So that the ML methods are not effective in this kind of situations.

8. The MPL-NN model showed relatively better robustness in overall detection performance. This indicates that Neural Network method which is regression based is more resilient in dealing with changes in feature space. This method could be adopted to add robustness into anomaly detection models in real network environment.

LESSONS LEARNT AND DISCUSSIONS

The organisations who might benefit from the findings and the lessons learnt include universities, research institutions, network operators, internet or cloud service providers, the organisations or businesses operating in cybersecurity domain. The findings of this project can be used by these organisations to determine their research plan, design research projects, assess cybersecurity solutions, or determine their technology transformation strategy.

Lessons Learnt

- During the process of the project implementation, we have learnt that the strong collaboration between the subject matter experts and data scientists is critical to develop effective and practical AI-based solutions. In the domain of cybersecurity, it is important that cybersecurity experts, network engineers and data scientists work closely together.
- It is worth spending more time on data analysis at all levels including physical layer, raw digital data, processed data, and data in constructed feature space to make sure the structure and characters of data is fully understood. This is critical for ML model development.
- It is also important to control the quality of each step of data processing which impacts the quality of the models.

Next step research problems

This research has proved that the Shallow Learning classification methods are highly effective in anomaly detection using the traditional machine learning experiment approach. The insignificant variations among the performance measures do not indicate the anomaly detection effectiveness in real network environment.

Further performance improvement using traditional machine learning experiment approach is not the focus of future research effort. Instead, the real-time detection should be studied on multiple aspects. The real-time detection related problems include the following:

This project was supported by APINIC foundation.

- training data construction for a specific network environment;
- improvement of the robustness of anomaly detection model;
- dynamic evaluation mechanism;
- dynamic model training, improvement and deployment; and
- lightweight model architecture.

Challenges

However, there are a number of challenges in this process.

1. Firstly, the training data construction requires significant effort from subject matter experts such as cybersecurity engineers and network engineers to work on detailed data analysis to capture and identify comprehensive network behaviours. The subject matter experts need to work closely with data scientists to work out the best way to structure data.

2. Extracting and constructing raw network data into the defined structure in real-time may increase risks in operation performance.

3. The constructed real-time network data will still contain some noise. This requires the ML anomaly detection models to be more robust to remain effective in real environments.

4. The network user behaviours and applications are growing and changing rapidly.

5. The network topology and architecture may change overtime. This may introduce significant changes in network data profile.

6. The cyber-attacks are continuously evolving. A mechanism that promptly identifies any emerging attacks is to be part of the detection strategy to ensure the ML models are up-to-date.

Based on what we have learnt through this project, TeleMARS will establish collaboration with research partners to continue the research in real-time anomaly detection to develop solutions that is practical in real network environment.

BIBLIOGRAPHY

1. Survey of intrusion detection systems: techniques, datasets and challenges; Ansam Khraisat*, Iqbal Gondal, Peter Vamplew and Joarder Kamruzzaman; Khraisat et al. Cybersecurity (2019) 2:20; https://doi.org/10.1186/s42400-019-0038-7
2. A holistic review of Network Anomaly Detection Systems: A comprehensive survey; Nour Moustafa, Jiankun Hua, Jill Slayb
3. A study on Anomaly Detection GAN-based methods on image data; Emanuel H. Silva1, Johannes V. Lochter
4. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study; Mohamed Amine Ferrag a , ∗, Leandros Maglaras b , Sotiris Moschoyiannis c , Helge Janicke b; *Department of Computer Science, Guelma University, Guelma 240 0 0, Algeria b School of Computer Science and Informatics, De Montfort University, Leicester U.K. c Department of Computer Science, University of Surrey, U.K.*
5. A detailed analysis of the KDD CUP 99 dataset, *http://refhub.elsevier.com/S2214-2126(19)30504-6/sbref0070*
6. Fraley, James B., and James Cannady. "The promise of machine learning in cybersecurity." SoutheastCon, 2017. IEEE, 2017.

This project was supported by APINIC foundation.

7. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in ICISSP, 2018, pp. 108–116
8. Chika Yinka-Banjo, Ogban-Asuquo Ugot, “A review of generative adversarial networks and its application in cybersecurity” Artificial Intelligence Review (2020) 53:1721–1736
9. Runa Bhaumik, Bamshad Mobasher and Robin Burke1, “A Clustering Approach to Unsupervised Attack Detection in Collaborative Recommender Systems”
10. Minghui Gao, Li Ma, Heng Liu, Zhijun Zhang, Zhiyan Ning and Jian Xu, “Malicious Network Traffic Detection Based on Deep Neural Networks and Association Analysis” Sensors 2020, 20, 1452
11. Kiran, B. R., Thomas, D. M., and Parakkal, R. (2018). An overview of deep learning based methods for pervised and semi-supervised anomaly detection in videos. arXiv e-prints, page arXiv:1801.03149.
12. Kwon, D., Kim, H., Kim, J., C. Suh, S., Kim, I., and Kim, K. (2017). A survey of deep learning-based network anomaly detection. Cluster Computing.
13. Perera, P., Nallapati, R., and Xiang, B. (2019). OCGAN: One-class Novelty Detection
14. Using GANs with Constrained Latent Representations. arXiv e-prints, page arXiv:1903.08550.
15. Phuc Ngo, C., Aristo Winarto, A., Kou Khor Li, C., Park, S., Akram, F., and Lee, H. K. (2019). Fence GAN: Towards Better Anomaly Detection. arXiv e-prints, page arXiv:1904.01209.
16. Abusitta, A.; Bellaiche, M.; Dagenais, M.; Halabi, T. A deep learning approach for proactive multi-cloud cooperative intrusion detection system. Future Gener. Comput. Syst. 2019, 98, 308–318.
17. Podgorelec, B.; Turkanovic, M.; Karakatic, S. A Machine Learning-Based Method for Automated Blockchain Transaction Signing Including Personalized Anomaly Detection. Sensors 2020, 20, 147.
18. Aechan Kim, Mohyun Park, AND Dong Hoon LEE, AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection. Special Selection on Scalable Deep Learning for Big Data, IEEE Access VOLUME 8, 2020
19. Canadian Institute for Cybersecurity, Datasets https://www.unb.ca/cic/datasets/
20. Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

APPENDIX

APPENDIX 1. LIST OF ALL ATTRIBUTES INCLUDED IN NSL-KDD DATA

Attribute No	Attribute Name	Description	Sample Data
1	Duration	Length of time duration of the connection	0
2	Protocol_type	Protocol used in the connection	Tcp
3	Service	Destination network service used	ftp_data
4	Flag	Status of the connection –Normal or Error	SF
5	Src_bytes	Number of data bytes transferred from source to destination in single connection	491
6	Dst_bytes	Number of data bytes transferred from destination to source in single connection	0
7	Land	if source and destination IP addresses and port numbers are equal then this variable takes value 1 else 0	0

This project was supported by APINIC foundation.

8	Wrong_fragment	Total number of wrong fragments in this connection	0
9	Urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated	0
10	Hot	Number of “hot“ indicators in the content such as: entering a system	0
11	Num_failed_logins	Count of failed login attempts	0
12	Logged_in	Login Status : 1 if successfully logged in; 0 otherwise	0
13	Num_compromised	Number of ‘’compromised'' conditions	0
14	Root_shell	1 if root shell is obtained; 0 otherwise	0
15	Su_attempted	1 if ``su root'' command attempted or used; 0 otherwise	0
16	Num_root	Number of ``root'' accesses or number of operations performed as a root in the connection	0
17	Num_file_creations	Number of file creation operations in the connection	0
18	Num_shells	Number of shell prompts	0
19	Num_access_files	Number of operations on access control files	0
20	Num_outbound_cmds	Numberof outbound commands in an ftp session	0
21	Is_hot_login	1 if the login belongs to the ``hot'' list i.e.	root or admin; else 0
22	Is_guest_login	1 if the login is a ``guest'' login; 0 otherwise	0
23	Count	Number of connections to the same destination host as the current connection in the past two	2
24	Srv_count	Number of connections to the same service (port number) as the current connection in the past two seconds	2
25	Serror_rate	The percentage of connections that have activated the flag (4) s0 s1 s2 or s3 among the connections aggregated in count (23)	0
26	Srv_serror_rate	The percentage of connections that have activated the flag (4) s0 s1 s2 or s3 among the connections aggregated in srv_count (24)	0
27	Rerror_rate	The percentage of connections that have activated the flag (4) REJ among the connections aggregated in count (23)	0

This project was supported by APINIC foundation.

28	Srv_rerror_rate	The percentage of connections that have activated the flag (4) REJ among the connections aggregated in srv_count (24)	0
29	Same_srv_rate	The percentage of connections that were to the same service among the connections aggregated in count (23)	1
30	Diff_srv_rate	The percentage of connections that were to different services among the connections aggregated in count (23)	0
31	Srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)	0
32	Dst_host_count	Number of connections having the same destination host IP address	150
33	Dst_host_srv_count	Number of connections having the same port number	25
34	Dst_host_same_srv_rate	The percentage of connections that were to the same service among the connections aggregated in dst_host_count (32)	0.17
35	Dst_host_diff_srv_rate	The percentage of connections that were to different services among the connections aggregated in dst_host_count (32)	0.03
36	Dst_host_same_src_port_rate	The percentage of connections that were to the same source port among the connections aggregated in dst_host_srv_count (33)	0.17
37	Dst_host_srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections aggregated in dst_host_srv_count(33)	0
38	Dst_host_serror_rate	The percentage of connections that have activated the flag (4) s0 s1 s2 or s3 among the connections aggregated in dst_host_count (32)	0
39	Dst_host_srv_serror_rate	The percent of connections that have activated the flag (4) s0 s1 s2 or s3 among the connections aggregated in dst_host_srv_count (33)	0
40	Dst_host_rerror_rate	The percentage of connections that have activated the flag (4) REJ among the connections aggregated in dst_host_count (32)	0.05

This project was supported by APINIC foundation.