Detection Of Android Malwares Using Recurrent Neural Networks

Thamizhisai.D[1], Praveen Kumar.A[2], Praisen.B[3], Nambiraju.P[4], Kaviarasan.K[5]
[1]Assistant Professor, Dept of CSE
[2, 3, 4, 5]Dept of Computer Science
[1]RAAK college of Engineering and technology Puducherry
[2, 3, 4, 5]Pondicherry university Puducherry

Abstract- *With Android's dominant position within the current smartphone OS, increasing number of malware applications pose a great threat to user privacy and security. Classification algorithms that use a single feature usually have weak detection performance. Although the use of multiple features can improve the detection effect, increasing the number of features increases the requirements of the operating environment and consumes more time. In existing system, a fast Android malware detection framework are preprocessed with the N-Gram technique and the FCBF (Fast Correlation-Based Filter) algorithm based on symmetrical uncertainty is employed to reduce feature dimensionality. Finally, the dimensionality reduced features are input into the CatBoost classifier for malware detection and family classification. In proposed system, the project is expected to show better results by implementing Recurrent Neural Networks (RNN)*

I. INTRODUCTION

1.1 MALWARE

Malware is any software intentionally designed to cause disruption to a computer, server, client, or computer network, leak private information, gain unauthorized access to information or systems, deprive access to information, or which unknowingly interferes with the user's computer security and privacy. Many types of malware exist, including computer viruses, worms, Trojan horses, ransomware, spyware, adware, rogue software, wiper, and scareware. The defense strategies against malware differ according to the type of malware but most can be thwarted by installing antivirus software, firewalls, applying regular patches to reduce zero-day attacks, securing networks from intrusion, having regular backups and isolating infected systems. Malware is now being designed to evade antivirus software detection algorithms [1].

1.2 ANDROID MALWARE

Android malware is malicious software that specifically targets Android devices. As with any type of malware, the intention is to harm the user's device and steal their data. Compared to Apple's App Store, Google's Play Store has less rigid security measures in place. In addition, Android users can download apps from various sources on the internet. This creates an environment in which cyberattacks are possible. Android malware spreads in a variety of ways, including:

Downloading malicious apps

The most common method hackers use to spread malware is through apps and downloads. The apps you download via official stores tend to be safe – although not always – but those which are pirated or downloaded from less legitimate sources are more likely to contain malware. Occasionally an app with malware will make it through to an official app store. These apps are usually discovered and removed quickly, but they underline the need to remain vigilant. If developers use untrusted SDKs – software development kits – then the apps they develop have an increased risk of malware.

Using a device with operating system vulnerabilities

Hackers can exploit any vulnerabilities within your device. Usually, security vulnerabilities are discovered fairly quickly and patched up, but if you don't regularly update software, then your device may be vulnerable. As with your computer, it's essential to keep your mobile device up to date, because hackers can exploit newly discovered vulnerabilities.

Opening clicking on suspicious links in emails or texts

Compromised emails are another way in which hackers install malware on your phone. For example, you may receive an email that says you have won something (a tablet, a vacation, etc). Or you may open an email which appears to be from your bank or another trusted company, asking you to update your details or log in to your account. In both scenarios, if you click on the link, you may be taken to a malicious website which downloads and installs malware on your phone. The data on your phone may then be exposed to the hacker.The same applies to links contained with text messages which appear to come from a legitimate source or

even someone in your contact list if their phone has been hacked. If in doubt, avoid clicking on links or opening attachments.

Using non-secure Wi-Fi/URLs

When you visit insecure websites, you run the risk of exposing sensitive data sent from your device. You're also more vulnerable to man-in-the-middle attacks and being exposed to malware. The browser on your phone could also be a source of vulnerabilities, which could lead to web browser attacks. Make sure you have the most current version of whatever browser you use.

1.3 IMPACT OF ANDROID MALWARE

Besides the irritation of constant ads, mobile malware can access your private information, such as:

- Your banking credentials
- Your device information
- Your phone number or email address
- Your contact lists

Hackers can use this information for a variety of malicious activities, such as committing identity theft using your banking credentials. For example, the Anubis banking Trojan does this by tricking users into granting it access to an Android phone's accessibility features. In turn, this allows the malware to log every app that you launch and the text you enter, including passwords. After you grant initial permission, the malware's activity is invisible on screen, with no sign anything malicious is taking place when you log into your accounts.

Hackers can also use malware to collect and sell your device and contact information, until you're bombarded with robocalls, texts and more ads. They can also send links for more malware to everyone on your contacts list [2].

1.3 OBJECTIVE

The main goal of this project is to develop an efficient deep learning model to classify the android malware.

II. EXISTING WORK

2.1 INTRODUCTION

In the existing approach, a fast Android malware detection framework based on the combination of multiple features: FAMD (Fast Android Malware Detector). Initially, permissions and Dalvik opcode sequences from samples to construct the original feature set has been extracted. Then, the Dalvik opcodes are preprocessed with the N-Gram technique and the FCBF (Fast Correlation-Based Filter) algorithm based on symmetrical uncertainty is employed to reduce feature dimensionality. Finally, the dimensionality reduced features are input into the CatBoost classifier for malware detection. The dataset DS-1, which are collected from various resources and the benchmark dataset Drebin were used in this experiment [7].

2.2 EXISTING METHODS

2.2.1FAST CORRELATION-BASED FILTER (FCBF)

FCBF is a multivariate feature selection method where the class relevance and the dependency between each feature pair are taken into account. Based on information theory, FCBF uses symmetrical uncertainty to calculate dependencies of features and the class relevance [8].

2.2.2CATBOOST CLASSIFIER

CatBoostor Categorical Boosting is an open-source boosting library developed by Yandex. In addition to regression and classification, CatBoost can be used in ranking, recommendation systems, forecasting and even personal assistants [9].

III. PROPOSED WORK

3.1 OVERVIEW

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. Recurrent Neural Networks enable you to model time-dependent and sequential data problems, like stock exchange prediction, artificial intelligence and text generation.

Models under the Recurrent Neural Network are:

- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

3.2 PROPOSED SYSTEM

3.2.1 LONG SHORT TERM MEMORY

Long Short Term Memory (LSTM) is a kind of recurrent neural network (RNN) design applied in deep

learning field. LSTM has feedback connection that is unrelated to standard feed forward neural networks. It cannot only process sole data points but also entire sequences of information. LSTM unit consists of a cell, an input gate, an output gate and a forget gate. The cell recollects values over arbitrary time breaks and therefore the three gates control the data flow into and out of the cell. LSTM networks are compatible for categorizing, handling and producing guesses supported statistic data, subsequently there are often delays of unidentified period between main events during time sequences. LSTMs were established to overcome the discharging and disappearing gradient problems which will be come across when training traditional RNNs.

The LSTM unit contains a memory cell that has three gates described below:

Input gate (i): The input gate computes the sum of input that is allowed to pass through it and is calculated by:

$$i = \sigma (x_t U^i + s_{t-1} W^i) \quad (1)$$

The sigmoid function plots the input value between [0, 1] and this value is multiplied by the weight vector (Ui). This helps the gate manage the quantity of input that is transferred through the input gate.

Forget gate (f): The forget gate helps the network to choose what and how much information from the earlier level to transfer to the succeeding level. The sigmoid function maps the value of this function between 0 and 1. It is given by:

$$f = \sigma (x_t U^f + s_{t-1} W^f) \quad (2)$$

If no input wants to be transferred to the next level, the previous memory is multiplied with the zero vector, that creates the input value zero. Likewise, if the memory at s_{t-1} needs to pass to next level it is multiplied by 1 vector. If only some part of the input is to be passed, then the resultant vector is multiplied with the input vector.

Output gate (o): The output gate, describes the output passed at each step of the network. It is given by:

$$o = \sigma (x_t U^o + s_{t-1} W^o) \quad (3)$$

BiLSTM means bidirectional LSTM, which means the signal transmits backward as well as forward in time.

3.2.2 GATED RECURRENT UNIT

GRU is a type of deep learning algorithm that is enhanced from the LSTM algorithm to minimize the complication of the algorithm by using update gate and reset gate. The update gate is used to regulate hidden state volume to be forwarded to the next state. The reset gate is used to define the consequence of the previous hidden state information.

Update Gate (z): It determines how much of the past information needs to be passed along into the future. It is similar to the Output Gate in an LSTM recurrent unit.

$$z = \sigma (x_t U^z + s_{t-1} W^z) \quad (4)$$

Reset Gate (r): It defines how much of the past information to forget. It is similar to the combination of the Input Gate and the Forget Gate in an LSTM recurrent unit.

$$r = \sigma (x_t U^i + s_{t-1} W^r) \quad (5)$$

BiGRU means Bidirectional GRU's are a kind of bidirectional recurrent neural networks allow for the use of information from both previous time steps and later time steps to make predictions about the current state.

3.3 ADVANTAGES OF PROPOSED SYSTEM

- RNN can process inputs of any length.
- An RNN model is modelled to remember each information throughout the time which is very helpful in any time series predictor.
- Even if the input size is larger, the model size does not increase.
- The weights can be shared across the time steps.
- RNN can use their internal memory for processing the arbitrary series of inputs which is not the case with feedforward neural networks.

IV. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

System architecture is the conceptual model that defines the structure, behaviour and more views of a system.

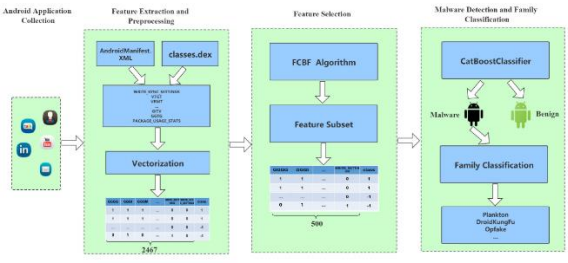

Figure 4.1 Architecture of Existing System

4.2 MODULES

4.2.1 IMPLEMENTATION OF CATBOOST CLASSIFIER WITH NSL-KDD DATASET

```
import sys
import sklearn
import pandas
import numpy
print('Python: {}'.format(sys.version))
print('Scikit-learn: {}'.format(sklearn.__version__))
print('Pandas: {}'.format(pandas.__version__))
print('Numpy: {}'.format(numpy.__version__))
Python: 3.8.16 (default, Dec  7 2022, 01:12:13)
[GCC 7.5.0]
Scikit-learn: 1.0.2
Pandas: 1.3.5
Numpy: 1.21.6
# import essential packages
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

# metrics evaluated
from sklearn import metrics as metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

from google.colab import drive
drive.mount('/content/drive/')
Mounted at /content/drive/

path = 'drive/My Drive'
df = pd.read_csv(path+'/drebin.csv')
df
```

transact	onServiceConnected	bindService	attachInterface	ServiceConnection	...	ACCESS_FINE_LOCATION	SET_WALLPAPER_HINTS	SET_PREFERRED_APPLICATIONS	WRITE_SECURE_SETTINGS	class
0	0	0	0	0	...	0	1	0	0	S
1	0	0	0	0	...	0	1	0	0	S
2	0	0	0	0	...	0	0	0	0	S
3	0	0	0	0	...	1	1	1	0	S
4	0	0	0	0	...	0	0	1	0	S
...	...	...	...	...	...	...	...	...	...	...
15031	1	1	1	1	...	1	1	0	0	B
15032	0	0	0	0	...	1	1	0	0	B
15033	0	0	0	0	...	0	1	0	0	B
15034	1	1	1	1	...	1	1	1	0	B
15035	1	1	1	1	...	1	1	1	0	0

```
15036 rows × 216 columns
print("The dataset shape is {}".format(df.shape))
The dataset shape is (15036, 216)

classes,count = np.unique(df['class'],return_counts=True)
#Perform Label Encoding
lbl_enc = LabelEncoder()
print(lbl_enc.fit_transform(classes),classes)
df = df.replace(classes,lbl_enc.fit_transform(classes))

#Dataset contains special characters like "?' and 'S'. Set them t
o NaN and use dropna() to remove them
df=df.replace('[?,S]',np.NaN,regex=True)
print("Total missing values : ",sum(list(df.isna().sum())))
df.dropna(inplace=True)
for c in df.columns:
    df[c] = pd.to_numeric(df[c])
[0 1] ['B' 'S']
Total missing values :  5

X = df.drop(columns='class',axis=1)
Y = df['class']
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X))

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_si
ze=0.25, random_state=42)   #Splitting train and test data
print('Size of train dataset: ' + str(X_train.shape[0]))
print('Size of test dataset: ' + str(X_test.shape[0]))
Size of train dataset: 11273
Size of test dataset: 3758

from catboost import CatBoostClassifier
cbc = CatBoostClassifier(n_estimators=50, learning_rate=1)
model = cbc.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
print(confusion_matrix(Y_test,Y_pred))
print(accuracy_score(Y_test,Y_pred))
[[2297   53]
 [  79 1329]]
0.9648749334752528
```

4.3 MODULE DESCRIPTION

The execution of the FCBF model with CatBoost classifier was took place on the Python 3.7.5 in Windows 10 64-bit operating system. The data is preprocessed by standard scaling and label encoding. The trained dataset is forwarded into FCBF-CatBoost classifier. Finally, the dataset is separated into 75% of training set and 25% of testing set and classify the target is normal or attack using FCBF-CatBoost classifier model. By implementing this model, it has been achieved with better accuracy of 96.48%.

V.CONCLUSION

In this existing model, a fast Android malware detection framework, FAMD, which combines permission features and Dalvik opcode features from different operation levels to construct feature vectors. To reduce the feature dimensionality and time complexity of the method, the FCBF algorithm is employed for feature selection. As a classifier proposed in recent years, CatBoost is employed in this work to conduct malware detection and family classification. In the experiments, the opcodes with 4-Gram and vectorize the features combined with permissions has been segmented. With the CatBoost as the classifier, the result achieves an accuracy of 96.48% in malware detection.

In future, the project is expected to show better results by implementing Recurrent Neural Networks (RNN) and the performance is compared with existing approaches

REFERENCES

[1] https://en.wikipedia.org/wiki/Malware
[2] https://www.kaspersky.com/resource-center/preemptive-safety/avoid-android-malware
[3] Nan Zhang et al., "Deep learning feature exploration for Android malware detection", Applied Soft Computing Journal, 102 (2021) 107069.
[4] Stuart Millar et al., "Multi-view deep learning for zero-day Android malware detection", Journal of Information Security and Applications, 58 (2021) 102718.
[5] Santosh K. Smmarwar et al., "An optimized and efficient android malware detection framework for future sustainable computing", Sustainable Energy Technologies and Assessments, 54 (2022) 102852.
[6] Vikas Sihag et al., "De-LADY: Deep learning based Android malware detection using Dynamic features", Journal of Internet Services and Information Security (JISIS), volume: 11, number: 2 (May 2021), pp. 34-45.
[7] Hongpeng Bai et al., "FAMD: A Fast Multifeature Android Malware Detection Framework, Design, and Implementation", IEEE Journal, 2020.
[8] https://sci2s.ugr.es/node/331
[9] https://www.geeksforgeeks.org/catboost-ml/