Chapter 73
Android Malware Detection

Shymala Gowri Selvaganapathy, G. Sudha Sadasivam, Hema Priya N, Rajeshwari N, Dharani M, and K. Karthik

Abstract Smartphones and mobile tablets are rapidly becoming essential in daily life. Android has been the most popular mobile operating system since 2012. However, owing to the open nature of Android, countless malwares are intermingled with a large number of benign apps in Android markets that seriously threaten Android security. The botnet is an example of using good technologies for bad intentions. A botnet is a collection of Internet-connected devices, each of which is running one or more bots. The Bot devices include PCs, Internet of Things, mobile devices, etc. Botnets can be used to perform Distributed Denial of Service (DDoS attack), steal data, send spam and allow the attacker access to the device and its connection. To ensure the security of mobile devices, malwares have to be resolved. Malware analysis can be carried out using techniques like static, dynamic, behavioural, hybrid and code analysis. In this chapter, several machine learning techniques and classifiers are used to categorize mobile botnet detection.

Keywords Android · Android security · Botnet · Static · Dynamic · Hybrid · Malware detection · Machine learning techniques

Abbreviation

DDoS	Distributed Denial of Service
CCTree	Categorical Clustering Tree
APK	Android Package
SVM	Support Vector Machine
ELM	Extreme Learning Machine
SLFN	Single-Layer Feedforward Neural Network
CNN	Convolutional Neural Network

S. G. Selvaganapathy (✉) · G. S. Sadasivam · Hema Priya N · Rajeshwari N · Dharani M · K. Karthik
PSG College of Technology, Coimbatore, Tamil Nadu, India

L. Ashok Kumar et al. (eds.), *Proceedings of International Conference on Artificial Intelligence, Smart Grid and Smart City Applications*,
https://doi.org/10.1007/978-3-030-24051-6_73

73.1 Introduction

In the present era of hypersonic technology proliferation, predominantly everyone is gazing with the handheld devices like smartphones, tablets, laptops and that depicts technology's fast-growing pace and interpret the force behind it all. Amidst the various mobile-operating systems, Android stands first as per the statistics [1] in Fig. 73.1. Despite its popularity, there are several security issues in Android which are caused by malwares, software which is specifically designed to disrupt, damage or gain authorized access to a computer system like Virus, Worms, Botnets, etc. So to provide security to mobile devices, malware has to be detected. Mobile botnets are type of botnets that target mobile devices, attempting to gain complete access to the device. Android malware detection and categorization can be done by several machine learning techniques like Support Vector Machine, Extreme Learning Algorithm, Logistics Regression and Convolutional Neural network.

73.2 Literature Survey

Given the potential fallout of installing malware on a personal computer or on a mobile device, there have been many proposed solutions for malware classification. Recently, researchers have been attempting to use machine learning and deep learning models to improve malware classification.

Ahmad Karim et al. [2] has proposed a framework which learns to distinguish applications having C&C functionality from malicious corpus through dynamic analysis of android applications. The framework were purely based on machine learning techniques that can classify applications based on various features collected

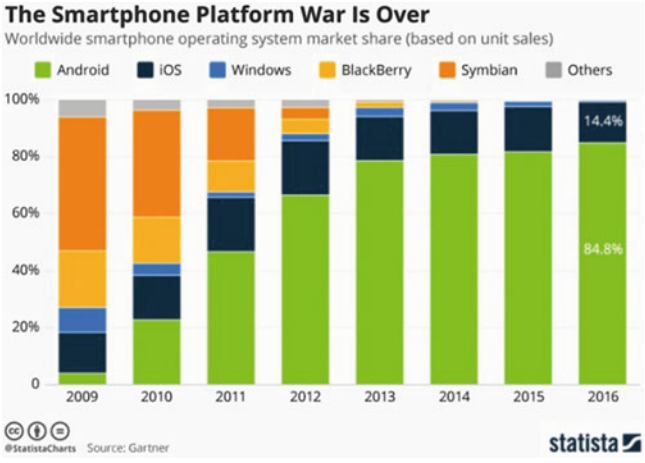

Fig. 73.1 Android statistics

at runtime through dynamic analysis of datasets. They proposed a learning technique by inducing Artificial Neural Networks back-propagation method.

AndiFitriah Abdul Kadir et al. [3] proposed a system of a deep analysis of Command and Control (C&C) and built-in URLs of Android botnets through static and dynamic analysis of the dataset Anubis consisting of 14 families. The dynamic analysis was conducted using Anubis, a web-based malware analysis tool. During the analysis, hidden encryption keys stored within the samples that allow revealing previously unknown features of Android botnets are identified, which are currently used to avoid detection.

DREBIN [4] a lightweight method for detection of Android malware that infers detection patterns automatically and enables identifying malware directly on the smartphone. DREBIN statically inspects a given Android application and extracts different feature sets from the application's manifest and dex code using a dataset of real Android applications. Another limitation which follows the use of machine learning has the possibility of mimicry and poisoning attacks.

Wenyi Huang et al. [5] has modelled a new deep learning malware classification architecture, MtNet, which shows for the first time that deep learning a modest improvement compared to a shallow neural architecture. TheMtNet architecture also employees rectified linear unit (ReLU) activation functions and dropout for the hidden layers. However, MtNet has also vulnerable to the recently reported attack for deep neural networks proposed by Papernot, et al. [6].

Antonio La Marra et al. [7] has proposed MalProfiler, a novel framework based on the Categorical Clustering Tree (CCTree) algorithm, to perform both automated grouping and classification of malicious apps in behavioural classes. The proposed framework clusters large amount of unlabelled malicious applications into smaller homogeneous sets. The analysis is based on a set of static features, extracted directly from the app's APK files, which are representative of both the app structure and of the performed behaviours.

Another interesting behavioural detection framework is proposed in Bose et al. [8] to detect mobile worms, viruses and trojans. The method implies a temporal logic approach to detect malicious activity in time. The ability of this framework to detect new types of malware is still uncertain as it requires a process of specifying temporal patterns for the malicious activities.

73.3 Proposed Work

The proposed system utilizes android botnet samples, perform dynamic analysis extended by machine learning techniques on the application files and categorize them into respective botnet families (Fig. 73.2).

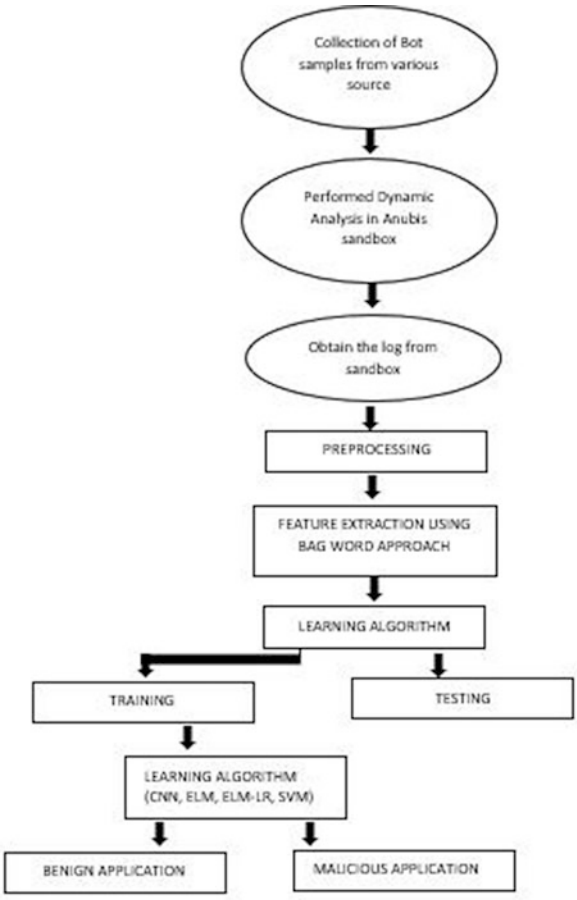

Fig. 73.2 System design of proposed model

73.3.1 Dataset Collection

An integrated dataset consisting of early and mature versions of Android botnets is generated by collecting samples representing 14 botnet families. The bot samples are collected from Android Genome Malware project [11], malware security blog [12], VirusTotal [13] and samples provided by a well-known anti-malware vendor. This dataset includes 1929 samples of Android package (APK) spanning a period from 2010 to 2014. It also covers a large number of existing Android botnets, which reflect the status of Android malware. The bot samples from different sources representing 14 families are stated in Table 73.1 [9] and the various commands and controls (C&C) they use. Dynamic analysis on bot samples was performed using the dynamic analysis toolkit *Anubis* [10]. Results from the dynamic analysis tool are analysed for further evaluations.

Table 73.1 Botnet families

Botnet family	C&C type	Propagation and attack types									
		Backdoor	Data theft	Drive by down load	Exploit technique	Infected SMS	Mobile banking attack	Ransom ware	Repackaged application	Social Engineer	Trojanized Application
Wroba	SMS/ HTTP					YES	YES				YES
Pletor	SMS/ HTTP							YES			YES
Sandroid	SMS						YES	YES			YES
Not compatible	HTTP			YES	YES						
Miso SMS	Email		YES		YES						YES
Bmaster	HTTP		YES		YES				YES		
Root smart	HTTP		YES		YES				YES		
Tiger bot	SMS	YES	YES								YES
Anserver bot	HTTP		YES			YES				YES	
Droid dream	HTTP		YES	YES	YES				YES		YES
Nicky spy	SMS		YES						YES		
Pjapps	HTTP								YES		YES
Geinimi	HTTP		YES	YES					YES		
Zitmo	SMS					YES	YES		YES	YES	

73.3.2 Methodologies

Support Vector Machines Support vector machines, because of their inherent classification capability, have been widely used in classification purposes. The major learning phases in SVM include

(i) Mapping the features onto a large vector space.
(ii) Using the standard methods to determine the separation between the class labels.

SVMs are inherently two-class classifiers. The traditional way to do multiclass classification with SVMs is to build One Vs Rest classifier approach.

OneVsRest classifier creates multiple binary classification models, merges the models and then optimizes the algorithm for each class. One advantage of this approach is its interpretability. Since each class is represented by one classifier only, it is possible to gain knowledge about the classes by analysing its corresponding classifier. This is the most commonly used algorithm for multiclass classification.

Extreme Learning Machines Extreme Learning Machines (ELM) are single hidden-Layer Feedforward Neural Networks (SLFN), where only the output weights are optimized, and all the weights between the input and hidden layer are assigned randomly.

SLFN with N hidden layers and with arbitrarily chosen input weights can learn N distinct observations with arbitrarily small error. Unlike the most practical implementations that all the parameters of the feedforward networks need to be tuned, we may not necessarily adjust the input weights and first hidden layer biases in applications. Some simulations have shown that this method not only makes learning extremely fast but also produces good generalization performance.

ELM learning speed can be thousands of times faster than traditional feedforward network learning algorithms like back-propagation algorithm while obtaining better generalization performance. Different from traditional learning algorithms to the proposed learning algorithm not only tends to reach the smallest training error but also the smallest norm of weights.

ELM Algorithm

Input: A training dataset
Activation function (.), and the number of hidden nodes M;
Output: A weights matrix

1. Randomly assign input weights Wj and biases bj, j = 1, 2, ..., M
2. Calculate the hidden layer output matrix H
3. Calculate output weights matrix $\hat{\beta} = H^{\bullet} T$

Step1: Let S = 0.
Step2: Partition Tinto training set Tr and testing set Te, l.e. T = Tr U Te

Step3: Train a probabilistic SLFN with ELM on Tr.
Step4: While testing accuracy less than a predefined value, do
Step5: For $\forall x \in Te$ Te calculate posterior probabilities $p(\omega_k|\, x)$ with trained SLFN;
Step6: For $\forall a_i \in A$ let the weights Wji = 0 (J = 1, 2, M) retrain new probabilistic SLFN with ELM on Tr;
Step7: probabilities SLFN; For $\forall x \in Te$ recalculate posterior $p(\omega_k|\, x)$ with trained new probabilistic.
Step8: Calculate $R(a_i) = \sum_{k=1}^{c} \sum_{x \in Te} \mid p(\omega_k|x) - p'(\omega_k|x) \mid$;
Step9: Calculate a$*$ = argmax{R (ai)};
Step10: Let S = S U {a $*$}

ELM with its higher scalability and less computational complexity not only unifies different popular learning algorithms but also provides a unified solution to different practical applications like regression, binary and multiclass classifications. The activation function like sine, Gaussian, sigmoidal, etc., can be chosen for hidden neuron layer and linear activation functions for the output neurons.

Logistic Regression Logistic regression is used for a different class of classification problems. Classification is all about portioning the data into groups based on certain features. In regression models, hypothesis function can be written as $Y = C^T X$, where

$$x = \begin{bmatrix} 1 \\ x^1 \\ x^2 \\ .. \\ x^n \end{bmatrix} \text{ and } c = \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ .. \\ \beta_n \end{bmatrix}$$

where Xi is the vector containing ith feature value for all the entries in dataset.

A sigmoid function is applied over the general known hypothesis function to get it into a range of (0, 1). The new activation function becomes like $\mathbf{Sg(y) = sg\,(C^T X) = 1/1 + e^{-CTx}}$. Logistics regression is generally used where the dependent variable is Binary or Dichotomous and it measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities.

Convolutional Neural Network CNN inherits the traits of an ordinary neural network. CNN stacks the neurons in a three-dimensional fashion. To implement for Anubis dataset, the log files are embedded onto vector space using a pertained word to vector model trained on a large corpus of text data. Then, the vector embedding's are given to the convolution layer, which performs convolution using multiple filter sizes. The next layer is the pooling layer, which max-pools the result of the convolution layer into a set of feature vectors and adds dropout regularization. The final layer is the classification layer which uses a softmax activation function to perform the classification task among multiple layers.

73.4 Result Analysis

The above four methodologies are used to evaluate the performance on the application. The multiclass SVM was trained for 70% as training set; the algorithm was predicted for 30% as testing set; and the accuracy score was obtained. In order to overcome the drawbacks of multiclass SVM in optimizing the various parameters to obtain a better accuracy, SLFN algorithm with only three-layered architecture is employed where the accuracy is much reduced. So ELM with logistic regression is applied then the accuracy rate on classification is increased. Finally, the dataset is tested with CNN algorithm and accuracy is calculated and shown in Table 73.2. The flow graph visually represents the performance of the elm algorithm pipelined with the logistic regression approach. The output of SVM and ELM are represented as graph visually (Fig. 73.3, 73.4, 73.5 and 73.6).

Table 73.2 Comparison of accuracy scores

Algorithm	Activation function used	Accuracy score
Multiclass SVM	Linear SVC	90.1%
ELM with SLFN	Hyperbolic tangent function	36.5%
ELM with logistic regression	Sigmoid	93.2%
CNN	Softmax	96.52%

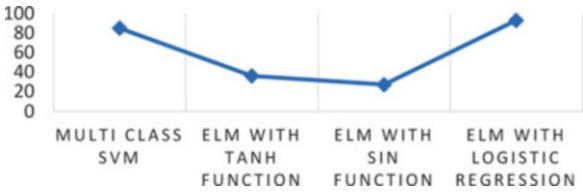

Fig. 73.3 Comparison of performance of SVM and ELM algorithms

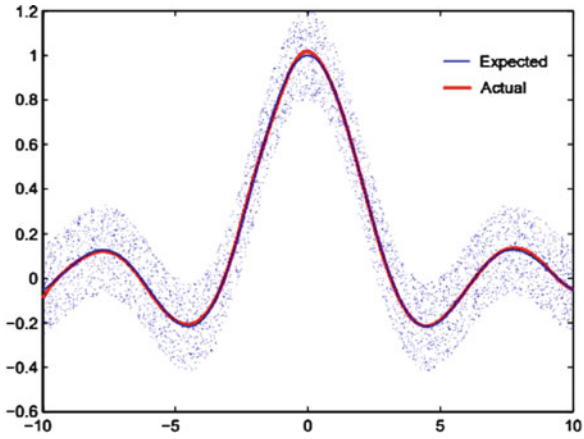

Fig. 73.4 Accuracy for ELM

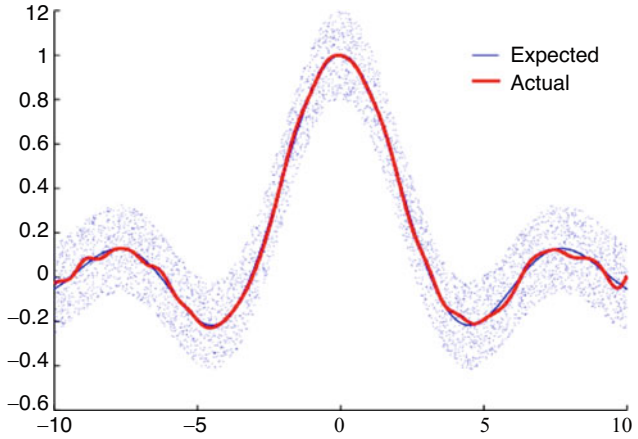

Fig. 73.5 Accuracy for SVM

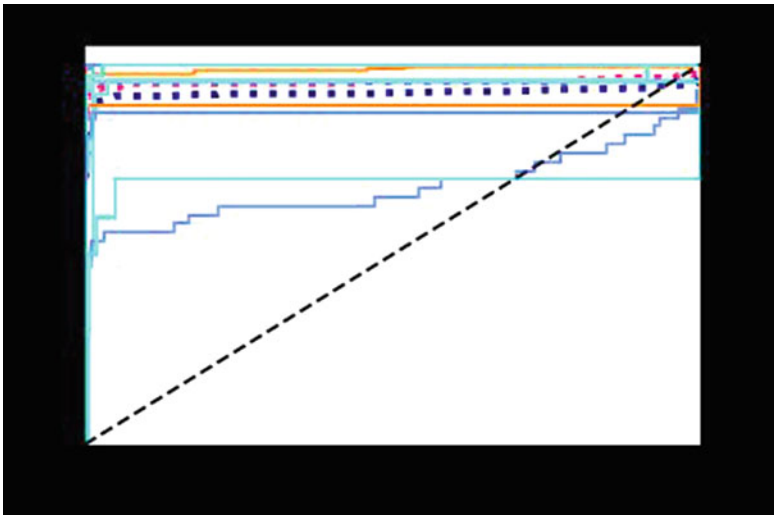

Fig. 73.6 ROC Curve of SVM

73.5 Conclusion

A framework is developed to analyse and detect Android-based mobile botnet application through dynamic analysis, augmented by machine learning technology. Extreme Learning Machines (ELM) are used for representation learning, and logistic

regression is used for multiclass classification. SVM and CNN are ML algorithms used to evaluate the proposed methodology. The future work can be to develop a hybrid on-device analysis for better categorization. For this purpose, we need to construct and implement the own sandbox with rich UI capabilities providing deep code coverage which can ultimately avoid all the deficiencies inherited from traditional dynamic analysis sandboxes.

References

1. https://www.statista.com/chart/4112/ smart phone-platform-market-share/
2. Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swamix A (2016) The limitations of deep learning in adversarial systems. In: IEEE European symposium on security and privacy
3. Kadir AFA, Stakhanova N, Ghorbani AA (2015) Android botnets: what URLs are telling us. in Springer
4. Bose A, Hu X,Shin KG, Park T (2008) Behavioral detection of malware on mobile handsets
5. MtNet: A Multi-Task Neural Network for Dynamic Malware Classification
6. MalProfiler: Automatic and Effective Classification of Android Malicious Apps in Behavioral Classes
7. http://scikit-learn.org/one-vs-rest-classifier
8. http://medium.com/towards-data science/activation-functions-and-its-types-which-is-better
9. http://www.unb.ca/cic/datasets/android-botnet.html
10. Karim A, Salleh R, Shah SAA (2015) DeDroid: a mobile botnet detection approach based on static analysis. In IEEE
11. Liu L (2008) BotTracer: execution-based bot-like malware detection. In: ISC '08 proceedings of the 11th international conference on information security
12. Divita J, Hallman RA (2017) An approach to botnet malware detection using Nonparametric Bayesian Methods. In ACM