ISSN: ISSN: 0974-5823 Vol. 7 No. 2 February, 2022

International Journal of Mechanical Engineering

Self-Attention based Bidirectional LSTM and Machine Learning Approaches for Android Malware Classification

Hena Iqbal[a], Navdeep Singh[b], Deepak Chahal[c], Krishna Kumar Singh[d]

[a]*School of Engineering and Technology, AL Dar University College, Dubai, UAE.*

[b]*Department of Computer Science & Engineering Punjabi University, Patiala, Punjab, India.*

[c]*Department of Computer Science, Jagan Institute of Management Studies, Rohini, New Delhi, India.*

[d]*Symbiosis Centre for Information Technology, Hinjawadi, Pune, Maharashtra, India.*

[a]dr.hena.iqbal@gmail.com[b]emailnavdeepsony@gmail.com,
[c]deepak.chahal@jimsindia.org,[d]emailsinghkrishna80@gmail.com

Abstract

Android-based mobile and other electronics devices dealing with medical and security applications bear huge amount data processing at heterogeneous environment causing a serious problem of security threats and system crash. To meet the business requirements, the android-based OS maintains much anticipated and smart environment for creating the developer's comforts zone. Therefore, from the prospective of maintaining personal information secured, it is utmost important to focus on safety and security of digital devices. In this paper, we gave the assurance for better malware classifiers utilizing the popular machine learning and deep learning based methods. In particular, performing the extensive set of experiments using machine learning classifiers on two dataset, one is created by API call and another Prem++ is created permission over APK. The API call dataset contains 1156 features, whereas prem++ contains 130 features. The methodology adopted with self-attention based features which are processed by setting of bidirectional LSTM. The noticeable advantages of our approach are that the system follows simple static approach and track the malware without dynamic analysis. More generally, APIs and permission settings are part of every android device which votes our approach satisfactorily applicable.

Keywords: Malware detection, Random forest, Self-attention, Machine learning, LSTM, API Toolkit.

1.0 Introduction

The word 'Malware' is derived from malicious software by connecting mal from malicious and ware from software. It basically refers to the threatening activities which have major target to access the unauthorized resources. In other words malware may refer to unethical practices to misuse the unauthorized software and fulling the interest for legitimate activities. The variants of its applications include viruses, ransomware and spyware, worm, adware, Trojan horses, and malware etc. [4, 7]. Malware usually consists of cyber-attacker-developed codes which are intended to inflict significant data and device harm or to obtain unauthorized access to a network. Malware can interpret any computing machine from small PC to big work station [5]. Connecting over internet causes the serious with misusing the personal via login. In general, malware is sent by email in the form of a connection or file which needs the recipient to select the link or open the file to execute the malware. Everybody uses in connected in daily life with some processing machine like smart mobile and PC to accomplish the required the task for their living. Therefore, corresponding to the growth of population, poor activities are encouraged by digital media. Hence, it is utmost important as well as challenging problem to keep safe the individual information in the huge mass of data server. The major proliferation of malware activities is noticed for accomplish the legitimate targets [6]

Cybersecurity analysts and researchers have two approaches: static and dynamic analysis of malware detection. Recent revolutions in digital media and highly of growing corresponding applications created very hazardous situation due to every devices is connected via several applications. Therefore, modern malwares are capable to evade both dynamic and static approaches via many available apps. Static analysis technique is adopted via virus scanning in which sequence of malicious pattern is followed to match the patterns in dataset. The matching score determines the program is malicious. But plenty of evasion approaches do not follow the tools for detecting malware. The popular approach of evasion, obfuscation is adopted in case of malware updates its polymorphism in run-time mode to hide its identity. The malware techniques are polymorphic and metamorphic and discussed in detail by Baysa et al. (2011) [8]. But the major issues with signature based malware detection are reported they do not maintain the semantic of the malware behavior [9]. From the detailed comparison of the static and dynamic approach [10], Moser et al. [11] introduced that a special opaque constant with obfuscation transformation can lead the evasion of malware to break the static analysis approach.

To create shielding against the loose points in signature base approach, Egele et al. (2008)[12] followed dynamic approach for

malware detection and developed systems ad refried behavior based malware detection. According to Yalew et al. (2017) [13], user-spaced malware takes OS services at large scale in which dynamic approach support to create a behavioral file of the intercept between OS and malware. The intercept of generated system calls falls into four categories for dynamic analysis of

alware detection: 1) Emulation, 2) Hyper-supervision, 3) Hooking, 4) Bare-metal approach. Hooking approach introduced in [13] overwrites the API code directly from te memory of the process in online modes. This monitors the record of any API calls from windows. Another important approach discussed by Bayer et al.(2006)[14] in which no emulation is entertained for tracking the API calls. The method based on emulation approach is bit sallower which results poor detection of malware.

Several popular tools for malware analysis are discussed in [15] al of which follows hyper supervision methods. The malware detection API tools used by Lengyel et al. (2014)[16], provide satisfactory performance without creating any interference with OS process and stands significantly better than emulation approach.

From the motivation of recent digital evolution AI-based applications covers majority of the problem space to provide better solution. Several examples from the state-of-the-arts ensured, machine learning and deep learning impact the domain of security at large scale. Feature engineering is key component of machine learning. In highly qualitative research discussed on feature engineering for malware detection, it recommended the rich set of the knowledgebase features to train the model [17, 18, 19]. To tackle the behavior of constantly developing malware, it is burdensome to maintain the drastic updates in dataset frequently. The problem to tackle such behavior of malware, deep learning provides better framework to experiment with large amount data and processing abrupt changes with the corresponding features.

In the contribution aspect, our work introduced self-attention based bidirectional LSTM (SA-BilSTM) resulting highly recommend performance in comparison to several existing state-of-the-arts like random forest, decision tree etc. For setting the experimental framework, we used two datasets. One is generated based on API Calls and another dataset is Permission which also referred as Prem++. The sets of malwares used to perform the experiments includes: TeslaCrypt, Vawtrak, Zeus, DarkComet, CTB-Locker, CyberGate, Xtreme, Locky Ransomware and Dridex.

1.2 Scope and Motivation

Android based operating systems are highly responsible to provide friendly services at large scale in heterogeneous environments. In the situation of several applications running in parallel, making the system secured and robust is toughest a challenge. Detecting malware can play vital role to deal such a troublesome situations of system crashed. The limitations of existing malware detection techniques, static analysis bears economical side effects via time, power and other resources. As working strategy is static analysis follows signature based scheme, it fails to determine the malwares at runtime. Another issues towards solving the problem, it is switched to dynamic approaches. But it requires huge amount of features to understand the changing behavior of the malware. Therefore, lacking of suitable and sufficient amount of feature samples, machine learning approach is discouraged. Developing a sustainable malware detection system is utmost important to keep the safe the growing digital media.

2.0 Literature Review

Shankarapani et al. (2010) present algorithms for identification that can enable the antivirus community to guarantee a version of a known malware without having to establish a signature, it can always be detected. By study of comparisons (based on specific quantitative measures) a matrix of similarity scores that can be generated is performed to determine the likelihood that a piece of code under inspection contains a particular malware. Authors present two methods- SAVE and MEDiC.

MEDiC uses analysis assembly calls and SAVE uses API calls for analysis (Static API call series and Static API call set). Authors illustrate where assembly can be superior to API calls. This provides a more rigorous comparison of executables. On the other hand, API calls may be superior to Assembly for Its speed and smaller signature. Better detection efficiency can be given by both of proposed techniques against obfuscated malware.

Alazeb et al. (2011) Zero-day Identification of Malware based on Supervised Learning Algorithms the API functions were used for feature representation, again and again. With the Help Vector Machines algorithm, the best result was obtained with normalized poly-kernel. 97.6 percent accuracy was reached, with a false-positive rate of 0.025. Amin Kharraz *et al.* (2015) proposed" A Look Under the Hood of Ransomware Attacks" studied ransomware attacks between 2006 and 2014. It tells that we can detect and stop zero-day ransomware attacks by keep view of I/O requests and securing the MFT (Master File Table) in the NTFS. The authors suggest mitigating ransomware attacks, system need real time monitoring.

Sgandurra et al. (2016) have suggested EldeRan tool, which checks Characteristic signatures of ransomware by examining a collection of Actions in the initial phases of the kill-chain assault flow. EldeRan detects and categorizes Ransomware dynamically by evaluating tasks such as registry operations Key operations, Windows API calls, directories, and files Operations of a machine. Logical Regression by EldeRanuses To identify each user's classifier algorithm and ML algorithm, Application, which has additional features for defining and identifying For as yet unknown ransomware, build signatures.

Carlin et al. (2017) highlighted the low-level study of both Dynamic and static opcodes to detect malware on 1,000 samples of labels in the runtime dataset to influence the typical AV labels. They obtained the dataset from VirusShare. The reviewer chose the scale and facility modality. There are 180,000 malware records, and these records are called by MD5 hash with no other metadata. Highest accuracy is 98.4% percent. kumar R et al. (2017) proposed "Evaluating Shallow and Deep Networks for Ransomware Detection and Classification" tells about supervised machine learning method of detection of ransomware. Multi-layer perceptron (MLP) used for ransomware detection and classification. It proposed a method using API calls for ransomware

detection. As a feature for classification it uses 131 API calls which are used as input for MLP architecture. Takeuchi et al. (2018) introduced Ransomware Detection using Support Vector Machine (SVMs). There are 588 samples in the dataset, which have 312

benign and 276 Ransomware. VirusTotal is used to obtain these samples. The authors developed the same vector symbols with different sequences of API calls. Author checked and educated the classifier of the SVM data type. The normal vector symbol accuracy is 93.52 percent, and 97.48 percent is the best SVM accuracy.

Greg et al. (2018) recommends technique of network management for data from traffic so that features can be extracted from it. Those features are used in the classification of ransomware and the used algorithm is Random Forest Binary Classifier. They say the rate of detection is 86 percent.

Research Methodology

We purposed method in which we use different machine learning algorithm for classification of different malware. In this section, we will describe the architecture of our proposed system. Our proposed detection system is attempting to improve the performance of malware detection using API call data. This system has main modules of raw feature extraction and applying learning algorithm for classification.

Feature Extraction and classification:

Feature selection creates the road map for machine learning classification. Every data item exhibits some specific features which gives identification to that item. For instance, in real estate estimating the price of a house requires knowledge as a multidimensional matrix, where the attributes are represented by the columns and rows presents in the matrix for the properties of numerical values. In case of an image, it is considered a pixel and interpreted as an RGB color. These features are referred to as traits, and the matrix is known as vector of functions. The data extraction method from the files is referred to as the feature extraction. The purpose of extracting features is to acquire a collection of insightful data that help to remove redundancy. It is crucial to understand that characteristics can reflect the important and valid details regarding our dataset, because we plan to develop for the specific experiments. In real-life scenario, exact forecasting is an ideal case. It makes extracting features a non-obvious assignment. Extracting feature is most complex task as per subject of research as it involves a lot of study and statistics. Additionally, it is quite domain-specific, but generic approaches apply badly here. Non-redundancy is another major prerequisite for a good feature set. Getting redundant characteristics, i.e., characteristics that outline the same data as well as redundant attributes of knowledge, which depend closely on each other's will skew the algorithm and thus have an incorrect one Outcome. Furthermore, if the input data is too large to be fed into the algorithm (has too many characteristics), so it can be translated to a reduced vector function (vector, having a smaller number of features). The phase of diminishing the measurements of the vector is referred to as function collection. At the completion of this operation, the chosen features are supposed to detail the related data from the Initial set so that, without any precision loss, it can be used instead of initial data.

It is essential to express raw data in some meaningful forms so that we can use it. We use cuckoo sandbox for feature extraction. Cuckoo sandbox support virtual system and monitor the file. We identified the registry key processes ip address and API call as feature. We select API call as a feature because it full fills all requirements. API call outlines everything happening to the operating system. Any action we do in file may be viewed as API call. After taking API call sequences of various files including malware we use these data as a training dataset of machine learning model. We used four machine learning algorithm which is suitable for classification. These classifiers include naïve Bayes, Regression, J48 and Random Forest.

Result Discussion and Analysis

All the experiment is done on Intel (R) Core i5-4210U CPU machine with 1.70 GHz processor, 32.00 GB of memory. The Microsoft Windows 10 pro is installed on this machine. Ransomware and benign software is analyzed in the Virtual Machine for 20 seconds. We choose small running time of 20 seconds for selecting relevant feature before infection of ransomware in the machine.

Datasets and Experimental Framework

API CALL and Permission: For preparing the dataset, we collect 3000 files in which 2000 files are malicious and 1000 files are benign. Malicious files include dridex, ctb-locker cyber-gate, teslacrypt, zeus, vawtrak, darkcomet xtreme and locky. Another dataset Permission dataset is created using APK toolkit [2]. In root direction of APK toolkit, there is manifest.xml file for providing essential guide to applications. The detailed designing of API toolkit is described in [3]

Cuckoo Sandbox

Cuckoo Sandbox is the platform for investigating open-source malware that makes it easy to get any file or URL has a comprehensive behavioral analysis in a matter of seconds. We use it in virtual environment for extracting features.

Classification

After feature selection of data we trained that data with the help of weka 3.8.5 data mining framework. After that we took test dataset of 1156 instances in which 173 samples are benign. From the extracted features of data samples, confusion is generated. As we have used five classifiers for each corresponding confusion matrices are represented in Table-1, Table-2, Table-3, Table-4, and Table-5.

Naïve Bayes classifier: It classifies data with assumption of independence among predictors based on Bayes' Theorem. A Naive Bayes classifier believes, in basic terms, that the inclusion of a certain function in a class is unrelated to the existence of any other feature. The major application of Naïve Bayes Classifier is tuned to forecast the trend analysis in a given time seriespattern.

Random Forest Classifier: Supervised learning ensemble methods which create a forest with n numbers of decision tree. It is developed for both classification and regression analysis. It shows highest accuracy in comparison of other ensemble methods.

Regression**:** It is supervised class of machine learning which predicts resultant value based of intendent variables based on dependent variables. The class of regression may be referred linear and non-linear depending on the correspondence of independent and dependent variables.

J48 Decision Tree: J48 is improvements of ID3 algorithms (Iterative Dichotomiser-3) developed by WEKA. For constructing decision tree, it uses both greedy and top down approach. The algorithm J48 has its significant over decision tree to utilize the missing values. The major drawbacks of J48 algorithm is concerned with overfitting, empty an insignificant braches.

Self Attention based Bidirectional LSTM (SA-BiLSTM): Long-short-Term-Memory (LSTM) is class of recurrent neural network which is capable to resolve the problem of long term calls of the information. Bidirectional-LSTM (BiLSTM) works in forward and backward directions which incorporate to develop an efficient mechanism for extracting the feature from multichannel matrix. The architecture of Self-attention based BiLSTM (SA-BiLSTM) is explained in [1] which is basically used for sentiment classification.

4.5 Weka: Weka is knowledge analysis environment developed by university of Waikato. Weka can classify data by machine learning in one click simply. We can apply weka algo on direct dataset by weka preprocess option in weka explorer or we can call these algorithms by our java program code. Java is used to write weka, so as java is machine independent it is also an open source and we can use weka in any platforms.

Table-1: Confusion matrix of Naïve Bays

a	b	c	d	e	f	g	h	i	j	Naïve Bayes Class
112	12	0	0	18	4	0	3	6	18	a=Beign
2	113	1	0	0	2	1	0	0	6	b=Dridex
2	0	81	0	0	1	0	0	1	2	c=Locky
0	1	0	110	0	1	0	0	0	1	d=Teslacrypt
5	11	10	1	46	1	0	2	2	6	e= Vawlrak
7	4	0	1	4	82	1	0	2	15	f=Zeus
1	0	0	0	0	2	128	0	0	0	g=DarkComet
0	1	0	0	0	0	7	120	0	1	h=CyberGate
3	0	0	1	0	2	3	0	112	0	i=Xtreme
2	0	1	0	0	0	0	0	0	76	j=CTB-Locker

Table-2: Confusion matrix of Random Forest

a	b	c	d	e	f	g	h	i	j	Random Forest Class
170	0	0	0	0	0	1	0	0	0	a=Beign
0	113	2	2	6	1	0	0	0	1	b=Dridex
0	0	90	0	1	0	0	0	0	2	c=Locky
0	1	0	110	1	0	0	0	0	1	d=Teslacrypt
0	2	1	1	65	4	0	1	0	0	e= Vawlrak
0	3	0	0	6	102	3	1	0	1	f=Zeus
0	1	0	0	1	1	128	0	0	0	g=DarkComet
0	1	0	0	0	3	6	119	0	0	h=CyberGate
0	0	1	0	0	0	0	1	119	0	i=Xtreme
0	0	0	0	2	0	0	0	0	77	j=CTB-Locker

Table-3: Confusion matrix of Regression Analysis

a	b	c	d	e	f	g	h	i	j	J48 DT Class
173	0	0	0	0	0	1	0	0	0	a=Beign
0	114	0	0	3	7	0	1	0	0	b=Dridex
0	1	87	1	2	1	0	1	2	0	c=Locky
0	0	0	111	0	0	1	0	0	1	d=Teslacrypt
0	4	1	1	62	4	0	1	0	1	e= Vawlrak
0	6	3	2	4	95	1	2	1	2	f=Zeus
0	0	0	1	0	0	127	3	0	0	g=DarkComet
0	2	0	0	0	1	5	120	0	0	h=CyberGate
0	0	0	0	1	0	0	0	120	0	i=Xtreme
0	0	0	0	2	1	0	0	0	76	j=CTB-Locker

Table-4: Confusion matrix of Regression Analysis

a	b	c	d	e	f	g	h	i	j	Regression Class
171	1	0	0	0	1	0	0	0	0	a=Beign
2	116	0	2	4	0	0	0	0	1	b=Dridex
0	0	92	0	2	0	0	1	0	0	c=Locky
0	0	0	112	1	0	0	0	0	0	d=Teslacrypt
2	0	0	0	72	0	0	0	0	1	e= Vawlrak
3	2	1	0	5	104	1	0	0	2	f=Zeus
1	0	0	0	0	0	130	0	0	0	g=DarkComet
1	0	0	0	0	1	7	120	0	0	h=CyberGate
1	0	0	0	0	0	0	0	120	0	i=Xtreme
1	1	0	0	0	0	0	0	0	77	j=CTB-Locker

Table-5: Confusion matrix of Self Attention Bidirectional LSTM

	b	c	d	e	f	g	h	i	j	SA-BiLSTM Class
179	1	0	0	0	1	5	0	1	0	a=Beign
2	106	0	2	0	0	0	1	0	1	b=Dridex
0	0	102	0	2	0	0	1	0	0	c=Locky
0	0	0	111	1	0	0	0	1	0	d=Teslacrypt
2	1	0	0	109	0	0	0	0	1	e= Vawlrak
0	2	1	0	5	104	1	0	1	2	f=Zeus
1	0	0	1	0	0	129	0	0	0	g=DarkComet
1	0	4	0	0	1	7	110	0	0	h=CyberGate
1	0	2	0	4	0	0	0	100	0	i=Xtreme
1	1	0	0	1	0	0	2	0	99	j=CTB-Locker

5- Result and Discussion

Out of 1156 instances 968 instances are classified accurately while 168 instances classified incorrectly. On the given related input feature matrix, the accuracy is received 0. 8373.

For Naive Bayes classifier, out of 1156 instances 968 instances are classified accurately while 168 instances classified incorrectly. Hence accuracy is the classifier 83.73%. Similarly, in case regression, out of 1156 instances 1094 instances are classified accurately while 62 instances classified incorrectly. Hence accuracy is 94.63%. Another important classifier based on decision tree, In J48, out of 1156 instances 1086 instances are classified accurately while 70 instances classified incorrectly. Hence accuracy is 93.94%. Random forest classifier, out of 1156 instances 1114 instances are classified accurately while 42 instances classified incorrectly. Hence accuracy is 96.36%.

While classifying with SA-BiLSTM, out 1156, it is received 1149 correctly classified samples and results high satisfactory performance with accuracy of 99.394%. Table 6 summaries the performance of all the classifier with their accuracy and other performance measures average True-Positive-Rate (TPR), average False-Positive-Rate, and F-measure the classifiers for malware detection. From accuracy graph presented in Figure 1, it is observed SA-BilSTM performed better 99.39% and 97.80% accuracy on API Calls and Permission dataset.

Table-6: Comparative Performance based Machine Learning and Deep Learning Classifiers

Datasets	Classifier	TPR	FPR	F-Measure	Accuracy
Permission	Naïve Bayes	0.84	0.03	84.20	81.02
	Random Forest	0.95	0.05	95.21	95.43
	Regression	092	0.08	90.02	91.29
	J48 Decision Tree	0.92	0.07	91.99	92.89
	SA-BiLSTM	0.97	0.04	96.88	97.80
API-Calls	Naïve Bayes	0.85	0.02	85.30	83.73
	Random Forest	0.96	0.04	96.40	96.36
	Regression	0.94	0.05	94.70	94.63
	J48 Decision Tree	0.93	0.06	93.90	93.94
	SA-BiLSTM	0.98	0.03	98.02	99.39

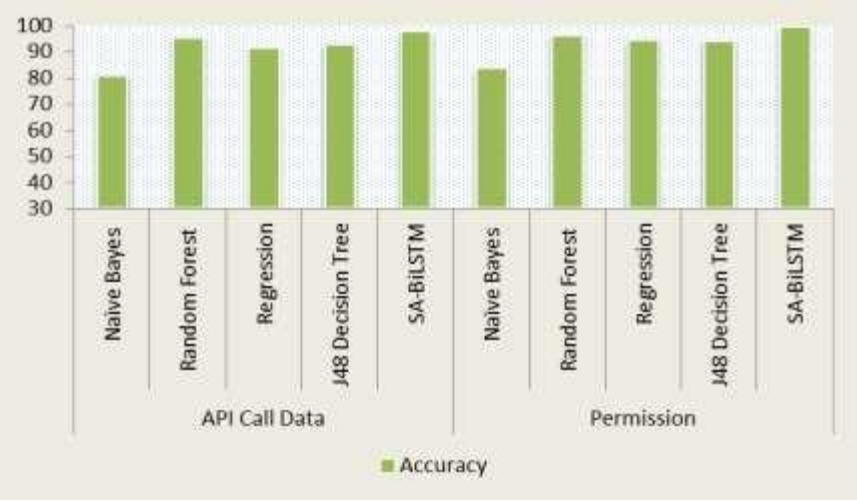

Figure-1: Accuracy on Machine Learning Classifiers on API Call and Permission Datasets

6- Conclusion and Future Recommendation

Malware detection and its avoidance is utmost important task for cyber security analyst. We have proposed detection of malware and classification with higher accuracy. We have used four classification algorithms to detect malware. In our approach, the permission prem++ is extracted from the information of every APP's profile whereas API is extracted from the file of App package. By using jointly, perm++ and API call features, our approach can classify any the malware of any potential class. From the experiments, it is also justified the success rate of different machine learning algorithms. Random Forest, Naïve Bayes, Regression and J48 classifier are used for detection which shows accuracy of 96.36%, 83.73%, 94.63%, and 93.94% respectively. The accuracy based on SA-BiLSTM is 99.39% which is highest on all the classifiers on both the datasets Permission and API-Calls. J48 Decision Tree and Naïve Bayes classifier shows higher accuracy than previous work and Random Forest find almost equal accuracy. We have classified different types of malwares which are dridex, locky, teslacrypt, vawtrak, zeus, darkcomet, cybergate, xtreme and CTB-locker. Thus, we have become successful to detect malware like dridex, vawtrak, dark-comet and cybergate using this method. Our method is suitable to classify and detect polymorphic behavior of malware.

6.0 Reference:

1. Li, W., Qi, F., Tang, M., & Yu, Z. (2020). Bidirectional LSTM with self-attention mechanism and multi-channel features for sentiment classification. Neurocomputing, 387, 63-77.
2. Android developer page for Android Manifest Permission group http://developer.android.com/reference/packages.html.
3. Peiravian, N., & Zhu, X. (2013, November). Machine learning for android malware detection using permission and api calls. In 2013 IEEE 25th international conference on tools with artificial intelligence (pp. 300-305). IEEE.
4. Saeed, I. A., Selamat, A., & Abuagoub, A. M. (2013). A survey on malware and malware detection systems. International Journal of Computer Applications, 67(16).
5. Zhou, Y., & Jiang, X. (2012, May). Dissecting android malware: Characterization and evolution. In 2012 IEEE symposium on security and privacy (pp. 95-109). IEEE.
6. Farivar, F., Haghighi, M. S., Jolfaei, A., & Alazab, M. (2019). Artificial intelligence for detection, estimation, and compensation of malicious attacks in nonlinear cyber-physical systems and industrial IoT. IEEE transactions on industrial informatics, 16(4), 2716-2725.
7. Hull, G., John, H., & Arief, B. (2019). Ransomware deployment methods and analysis: views from a predictive model and human responses. Crime Science, 8(1), 1-22.
8. Baysa, D., Low, R. M., & Stamp, M. (2013). Structural entropy and metamorphic malware. Journal of computer virology and hacking techniques, 9(4), 179-192.
9. Feng, Y., Anand, S., Dillig, I., & Aiken, A. (2014, November). Apposcopy: Semantics-based detection of android malware through static analysis. In Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering (pp. 576-587).
10. Damodaran, A., Troia, F. D., Visaggio, C. A., Austin, T. H., & Stamp, M. (2017). A comparison of static, dynamic, and hybrid analysis for malware detection. Journal of Computer Virology and Hacking Techniques, 13(1), 1-12.

11. Moser, A., Kruegel, C., & Kirda, E. (2007, December). Limits of static analysis for malware detection. In Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007) (pp. 421-430). IEEE.
12. Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2008). A survey on automated dynamic malware-analysis techniques and tools. ACM computing surveys (CSUR), 44(2), 1-42.
13. Yalew, S. D., Maguire, G. Q., Haridi, S., & Correia, M. (2017, August). T2Droid: A TrustZone-based dynamic analyser for Android applications. In 2017 IEEE Trustcom/BigDataSE/ICESS (pp. 240-247). IEEE.
14. Bayer, U., Kruegel, C., & Kirda, E. (2006). TTAnalyze: A tool for analyzing malware (pp. 180-192). na.
15. Zaidenberg, N. J. (2018). Hardware rooted security in industry 4.0 systems. Cyber Defence in Industry 4.0 Systems and Related Logistics and IT Infrastructures, 51(135).
16. Lengyel, T. K., Maresca, S., Payne, B. D., Webster, G. D., Vogl, S., & Kiayias, A. (2014, December). Scalability, fidelity and stealth in the DRAKVUF dynamic malware analysis system. In Proceedings of the 30th annual computer security applications conference (pp. 386-395).
17. Mohamed Shakeel, P., Baskar, S., Sarma Dhulipala, V. R., Mishra, S., & Jaber, M. M. (2018). Maintaining security and privacy in health care system using learning based deep-Q-networks. Journal of medical systems, 42(10), 1-10.
18. Azmoodeh, A., Dehghantanha, A., Conti, M., & Choo, K. K. R. (2018). Detecting crypto-ransomware in IoT networks based on energy consumption footprint. Journal of Ambient Intelligence and Humanized Computing, 9(4), 1141-1152.
19. Kumar, A., Kim, J., Lyndon, D., Fulham, M., & Feng, D. (2016). An ensemble of fine-tuned convolutional neural networks for medical image classification. IEEE journal of biomedical and health informatics, 21(1), 31-40.
20. Takeuchi, Y., Sakai, K., & Fukumoto, S. (2018, August). Detecting ransomware using support vector machines. In Proceedings of the 47th International Conference on Parallel Processing Companion (pp. 1-6).
21. Vinayakumar, R., Soman, K. P., Velan, K. S., & Ganorkar, S. (2017, September). Evaluating shallow and deep networks for ransomware detection and classification. In 2017 international conference on advances in computing, communications and informatics (ICACCI) (pp. 259-265). IEEE.
22. Song, S., Kim, B., & Lee, S. (2016). The effective ransomware prevention technique using process monitoring on android platform. Mobile Information Systems, 2016.
23. Shaukat, S. K., & Ribeiro, V. J. (2018, January). RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning. In 2018 10th International Conference on Communication Systems & Networks (COMSNETS) (pp. 356-363). IEEE.
24. Shankarapani, M. K., Ramamoorthy, S., Movva, R. S., & Mukkamala, S. (2011). Malware detection using assembly and API call sequences. Journal in computer virology, 7(2), 107-119.
25. Scalas, M., Maiorca, D., Mercaldo, F., Visaggio, C. A., Martinelli, F., & Giacinto, G. (2019). On the effectiveness of system API-related information for Android ransomware detection. Computers & Security, 86, 168-182.
26. Scaife, N., Carter, H., Traynor, P., & Butler, K. R. (2016, June). Cryptolock (and drop it): stopping ransomware attacks on user data. In 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS) (pp. 303-312). IEEE.
27. Salman, T., Bhamare, D., Erbad, A., Jain, R., & Samaka, M. (2017, June). Machine learning for anomaly detection and categorization in multi-cloud environments. In 2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud) (pp. 97-103). IEEE.
28. Rai, M., & Mandoria, H. (2019). A study on cyber crimes cyber criminals and major security breaches. Int. Res. J. Eng. Technol., 6(7), 1-8.
29. Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525-41550.
30. Pundir, S. L. (2013). Feature Selection using Random Forest in intrusion detection system. International Journal of Advances in Engineering & Technology, 6(3), 1319.
31. Ligh, M. H., Case, A., Levy, J., & Walters, A. (2014). The art of memory forensics: detecting malware and threats in windows, linux, and Mac memory. John Wiley & Sons.
32. Alazab, M., Layton, R., Venkataraman, S., & Watters, P. (2010). Malware detection based on structural and behavioural features of api calls.
33. Shukla, M., Mondal, S., & Lodha, S. (2016, October). Poster: Locally virtualized environment for mitigating ransomware threat. In proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 1784-1786).
34. Zhao, M., Ge, F., Zhang, T., & Yuan, Z. (2011, October). AntiMalDroid: An efficient SVM-based malware detection framework for android. In International conference on information computing and applications (pp. 158-166). Springer, Berlin, Heidelberg.
35. Rhode, M., Burnap, P., & Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. computers & security, 77, 578-594.

36. Cusack, G., Michel, O., & Keller, E. (2018, March). Machine learning-based detection of ransomware using SDN. In Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization (pp. 1-6).

37. Haddadi, F., Khanchi, S., Shetabi, M., & Derhami, V. (2010, April). Intrusion detection and attack classification using feed-forward neural network. In 2010 Second international conference on computer and network technology (pp. 262-266). IEEE.

38. Sgandurra, D., Muñoz-González, L., Mohsen, R., & Lupu, E. C. (2016). Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020.

39. Carlin, D., Cowan, A., O'kane, P., & Sezer, S. (2017). The effects of traditional anti-virus labels on malware detection using dynamic runtime opcodes. IEEE Access, 5, 17742-17752.

40. Chen, J., Wang, C., Zhao, Z., Chen, K., Du, R., & Ahn, G. J. (2017). Uncovering the face of android ransomware: Characterization and real-time detection. IEEE Transactions on Information Forensics and Security, 13(5), 1286-1300.

41. Cabaj, K., Gregorczyk, M., & Mazurczyk, W. (2018). Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics. Computers & Electrical Engineering, 66, 353-368.

42. Brewer, R. (2016). Ransomware attacks: detection, prevention and cure. Network Security, 2016(9), 5-9.

43. Alhawi, O. M., Baldwin, J., & Dehghantanha, A. (2018). Leveraging machine learning techniques for windows ransomware network traffic detection. In Cyber threat intelligence (pp. 93-106). Springer, Cham.

44. Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., & Kirda, E. (2015, July). Cutting the gordian knot: A look under the hood of ransomware attacks. In International conference on detection of intrusions and malware, and vulnerability assessment (pp. 3-24). Springer, Cham.

45. Andronio, N., Zanero, S., & Maggi, F. (2015, November). Heldroid: Dissecting and detecting mobile ransomware. In international symposium on recent advances in intrusion detection (pp. 382-404). Springer, Cham.