A survey of recent publications on the use of Machine Learning for malware identification

Markku Leppälä
markku.leppala@aalto.fi

Tutor: Alexey Kirichenko

Abstract

The modern antivirus software relies heavily on static and dynamic analysis to detect malware in the systems. The use of machine learning models, based on static and dynamic features such as code structure or CPU usage, has been continuously attracting the attention of the researchers. These models help to maximize True Positive Rate when constraining False Positive Rate to be close to zero. This survey examines nine recent publications on developing a malware identification model for Windows or Android platforms with machine learning classification. The main goals are to review different approaches, used machine learning frameworks, and claimed results in the selected publications.

KEYWORDS: malware detection, machine learning, Windows, Android

1 Introduction

Signatures represent a specific pattern that is created from attributes collected by analyzing the static properties of the malware [1]. Malware detection via comparison of signatures to files is often ineffective for zero-day malware and also due to the obfuscation technologies such as polymorphism or encryption [2, 3]. These signatures also have a long overhead for

reverse-engineering the malware and distributing the newest database to end-users [4].

The static and dynamic analysis combined with machine learning models has been proven an effective and scalable solution in detecting the unseen malware or modified version of previously known one [5]. Such models can be created with the help of static analysis or dynamic analysis, which analyzes the traces of running applications, or even the combination of these, a hybrid model [6].

This survey analyzes nine recent publications on malware detection with machine learning models. The aim is to examine the approaches for creating the models and detecting the malware. The next section describes the methodology of the analysis in detail.

2 Methodology

This survey is a literature review of the selected publications. These papers cover the two platforms with the highest amount of malware: Windows and Android. Other key criteria are maturity and credibility of the paper. These papers should not be older than four years and have high-quality data used. These papers and the selection criteria are listed in the Appendix I. A review of a paper consists of five sections: *summary*, *data*, *feature engineering*, *approach*, and *frameworks*.

The summary analyzes the problems addressed by the publication and the main contributions claimed by the authors, such as novel methods. This section also indicates which is the intended platform for the solution.

The data section examines the quality and quantity of the datasets used for training and testing the models. Important factors regarding the data and data labels are the source, gathering method, amount of benign and malicious samples, and novelty of the data.

The feature engineering inspects the data extraction approaches, which can be static, dynamic, or hybrid, techniques used for reducing the dimensionality, and selecting the features.

The approach describes the machine learning approaches used. These include methods for data construction, training, and validation methods used in order to achieve the classification. This part also describes the results achieved by the chosen methods.

The frameworks sections inspect the software used for data extraction and machine learning tools or libraries used in the approach.

3 Smart malware detection on Android

Smart malware detection on Android addresses an issue that commercial antivirus solutions running on the Android platform are using static signatures on malware detection [4]. Thus, the performance of these solutions lacks against variants and zero-day attacks. The main contributions include extensive behavioral data collection, dynamic and static analyses combination, favorable detection performance, and zero-day malware detection. The solution outperforms other antivirus software in higher detection performance and lower resource consumption.

3.1 Data

The paper [4] uses 400 benign applications and 400 malware samples in training and evaluation processes in total. The paper halves applications into logistic classifier training and evaluation of the trained model. Behavioral data is collected by monitoring the application for 5 minutes. Benign applications are downloaded from Google Play and SlideMe Market, whereas malicious data is from VirusShare, AVCaesar, Contagio Mobile, and Android Malware Genome Project.

3.2 Feature Engineering

The work in [4] creates a hybrid model combining metadata from Google Play, such as ratings and a number of downloads and the extracted behavioral data, such as CPU usage and a number of sent SMSs. The publication does not report dimensionality reduction.

3.3 Approach

Four hundred unseen samples, 200 benign and 200 malicious, validate the results described in Table 1.

Table 1. Detection performance

Detection Metric	Value
True positive rate	0.980
False negative rate	0.020
True negative rate	0.990
False positive rate	0.010
Accuracy	0.985
Precision	0.989
F-measure	0.985

3.4 Frameworks

The paper [4] extracts dynamic data with built-in hooks in the system. The authors use the LingPipe Java library for classifier training with a logistic regression algorithm.

4 SigPID: Significant Permission Identification for Machine Learning Based Android Malware Detection

Every ten seconds, a new malicious Android app is released into the wild [7]. Based on the publication, no malware detection tools are available that would be suitable for a large bundle of applications. The publication introduces SigPID [5], scalable malware detection. The authors develop 3-levels of feature pruning to reveal the most significant permissions to detect malware.

4.1 Data

The approach uses 310 926 benign and 54 694 malicious samples. The benign apps are downloaded from Google Play Store in June 2013. The malicious apps are obtained from Google Play and Anzhi Store between Jan 2012 and Dec 2014.

4.2 Feature Engineering

The work in [5] uses Android applications' requested permissions as features for static malware analysis. The paper uses novel 3-level feature pruning reducing features by 84% from original 135 distinctive to 22 significant ones. The permissions are ranked to find the ones displaying the highest support.

4.3 Approach

The approach of the paper [5] uses multiple ML algorithms to find the best performing one. Functional Tree algorithm performs the best for selected permissions. The author validates the results of the approach against models created with a different number of permission, such as a list of dangerous permissions provided by Google with the best performing algorithm. Table 2 describes the results.

Table 2. Detection performance

Detection Metric	Value
True positive rate	0.9362
False positive rate	0.0236
Accuracy	0.9563
Precision	0.9754
F-measure	0.9554

4.4 Frameworks

The paper [5] uses machine learning models from the Weka library.

5 Machine learning aided Android malware classification

Milosevic et al. [8] stress that widespread adoption of Android devices and their capability to access significant private and confidential information have resulted in these devices being targeted by malware developers. This paper proposes two models for malware detection in Android. The first one is a lightweight model based on app permissions. The second one is a novel approach performing code analysis, capable of revealing more granular app behaviors.

5.1 Data

The paper [8] utilized 178 malicious and 190 benign Android samples for training and testing from the M0Droid dataset [9].

5.2 Feature Engineering

The work in [8] uses static extraction for features, which are permissions requested by the application and source code analysis. The requested permissions are extracted from the manifest file. The source code analysis is achieved by tokenizing decompiled Java files into unigrams that are used as a bag-of-words.

5.3 Approach

The paper [8] uses different approaches for the aforementioned feature sets. For the permission-based analysis, several machine learning algorithms are used. The models trained with these algorithms are evaluated using several clustering techniques and ten-fold cross-validation. SVM with Sequential Minimal Optimization (SMO) produced the best result with an F-measure of 87.9%. Training the model and classifying is fast; thus, it is suitable for real-time monitoring.

For the code-based classification, SVM with SMO produced the best results with an F-measure of 95.1% and 95.2% precision. No false positive rate is reported.

Both approaches are using ensembles of multiple classifiers. These produced slightly better results with a drawback of increased training time.

5.4 Frameworks

The paper [8] uses Weka, Dex2jar, and Procyon Java decompiler.

6 Scorpion: A Metagraph2vec Based Malware Detection System

Gotcha - Sly Malware! Scorpion: A Metagraph2vec Based Malware Detection System [10] proposes a solution to detect malware in a continuously evolving landscape of sly malware. Current and future malware are using sophisticated encryption, obfuscation, and polymorphism to avoid detection. The paper introduces Scorpion, a heterogeneous information network (HIN) based detection system, to model relations among archives, files, APIs, and DLLs for malware detection.

6.1 Data

The paper [10] uses sample collection from Comodo Cloud Security Center to train and evaluates the model. In total, 59 794 Portable Executable (PE) files are used, of which 23 178 are malicious, and 36 571 are benign. The files are labeled by Comodo.

6.2 Feature Engineering

The paper [10] uses dynamic features. These features are content-based features such as Windows API calls and relation-based features such as semantic file-machine relations or the creation of a new file. The features are extracted from the uploaded PE file in the Comodo's online server.

6.3 Approach

The extracted features are mapped into a meta-graph scheme built over HIN using meta-graph guided random walk and skip-gram. The low-dimensional vectors obtained from metagraph2vec, the meta-graph scheme, are fed to SVM (Support Vector Machine) to train the classification model. The model is validated with ten-fold cross-validation. The results are described in Table 3.

Table 3. Detection performance

Detection Metric	Value
True positive rate	0.974
False positive rate	0.01
Accuracy	0.983
F-measure	0.977

6.4 Frameworks

The paper [10] uses Comodo Cloud Security Center to extract the features and label the samples. SVM from LibSVM is used to train the classification model.

7 Malware classification using self organising feature maps and machine activity data

Burnap et al. [11] point out that cyber attacks are becoming increasingly complex with obfuscation of network traffic and system-level interactions. This paper aims to use a hard-to-obfuscate machine data footprint to classify malware. This paper builds a machine classifiers and implements a two-map Self Organizing Feature Map (SOFM) to improve the classification accuracy for 'fuzzy-boundaries.'

7.1 Data

The paper [11] uses 594 malware and 594 benign 32-bit PE samples for training the model. The labels are obtained from VirusTotal.

7.2 Feature Engineering

Data is extracted from the PE samples by running the software for 5 minutes in a Cuckoo sandbox environment. In addition to the Cuckoo, traces are recorded with a script developed by the P. Burnap et al. This environment runs Windows 7 operating system. These recorded dynamic execution traces are mapped into two-map SOFM. The dimensionality is reduced to 50 x 50, which enables faster training time with a minimum drop in accuracy.

7.3 Approach

The two SOFMs are combined with a classifier based around a Logistic Regression model and used the Best Matching Unit output from the SOFM. The classifier is validated with a k-fold cross-validation method. The findings of this suggest that a neural-type model for learning, combined with SOFM, outperforms other models for detecting polymorphic malware and APTs. Results are listed in Table 4. False positive rate is not reported.

Table 4. Detection performance

Detection Metric	Value
Accuracy	0.9376
Precision	0.946
F-measure	0.938

7.4 Frameworks

Data is gathered from Cuckoo with combined custom scripts. Weka ML libraries are to develop a number of supervised classifiers.

8 A comparison of static, dynamic, and hybrid analysis for malware detection

Damoran [6] compares the effectiveness of malware detection techniques based on static, dynamic, and hybrid analysis. These two hybrid models are trained with static features and classified with dynamic features and vice versa. The aim of this paper is to compare the detection rate on different malware families and the robustness against code obfuscation. Hidden Markov Models (HNNs) are used to classify malware based on features from API calls and opcode sequences.

8.1 Data

The work in [6] uses 745 malware samples in total from six different families [12]. Forty different Windows System 32 files are used as benign samples. The imbalance problem in the dataset ratio is discussed.

8.2 Feature Engineering

For training and scoring, the paper [6] uses opcode sequences and API calls. These features are extracted with a static and dynamic approach. The static opcode is extracted with IDA Pro from the disassembly and dynamic opcode with the 'tracing' feature. API call names are obtained statically with IDA Pro and dynamically using Buster Sandbox Analyser (BSA). Features are reduced by only using mnemonic opcodes and names of the API calls.

8.3 Approach

The paper [6] uses four different approaches for training and testing, including static, dynamic, and hybrid approaches. For each approach, five-fold cross-validation is used. The scores from a given experiment are used to form a scatterplot, from which a ROC curve is generated. This area under the ROC curve is serving as our measure of success. The claimed results suggest that for API calls and opcode sequences, a fully static strategy is generally the most efficient approach. The hybrid approach is inconsistent and inferior compared to dynamic and static approaches.

8.4 Frameworks

The paper [6] uses IDA Pro and BSA feature extraction.

9 Binary malware image classification using machine learning with local binary pattern

Luo and Lo [13] propose a malware classification approach based on image processing and convolutional neural network. The paper argues that static analysis is ineffective against code obfuscation, and dynamic analysis fails when the malware has an alternating behavior in two different environments. Thus, the proposed method is based on analyzing features extracted from malware binary code with a local binary pattern (LBP), which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

9.1 Data

The dataset [14] used includes 32 malware families and 12 000 greyscale malware images in total.

9.2 Feature Engineering

The approach is static as the behavior of the malware is not observed. This approach uses the binary code of the malware mapped into a greyscale image. These images are reorganized into three by three grids of which the features are extracted using LBP. This feature extraction is a visual descriptor; thus, it is taking into account the value of the pixel and comparing it to the value of surrounding pixels.

9.3 Approach

The aim of the paper [13] is to recognize the family of the malware. This approach uses TensorFlow for the LBP features classification. TensorFlow is used for training and testing. 20% of each malware family dataset is used for training and the rest for testing. The classification is performed with two convolutional filters and a flattening layer. The results are compared against approaches using SVM or k-nearest neighbors (KNN) algorithm with either LBP or GIST [15] features of which the combination of TensorFlow and LBP performed the best with the accuracy of 93.17%. No benign samples are reportedly used to evaluate the false positives.

9.4 Frameworks

The paper [13] uses TensorFlow library as classifier.

10 Automatic malware classification and new malware detection using machine learning

Traditional antivirus systems based on signatures fail to classify unknown malware into their corresponding families and are unsuccessful in the detection of new kinds of malware programs. Therefore, Liu et al. propose a machine learning-based malware analysis system [16] using three sources for features and a novel feature extraction method.

10.1 Data

21 740 Windows malware samples belonging to nine families are used to train and test the classification. The dataset is collected and labeled by Kingsoft, ESET NOD32, and Anubis. Benign samples are not included.

10.2 Feature Engineering

The proposed solution deals with three kinds of data extracted with static analysis: grayscale images, opcode n-gram, and import functions, which are employed to extract the features of the malware. Grayscale images are constructed from the binary code obtained with an interactive disassembler (IDA). Opcode n-gram is extracted with IDA Pro from the control flow graph (CFG) using a 3-gram model. Import functions are extracted by analyzing the appearance of different DLL's in import functions. This approach uses a novel two-step strategy to reduce dimensionality based on frequency and discrimination.

10.3 Approach

The proposed decision-making system uses multiple classifiers with a weighted voting scheme. The shared nearest neighbor (SNN) method is used to cluster points in the high-dimensional space. The results of this paper show that the system can effectively classify the family of unknown malware with the best accuracy of 98.9% and successfully detects 86.7% of the new malware. These results are achieved with Random Forest classifying algorithm combined with grayscale images and 3-gram opcode.

10.4 Frameworks

The paper [16] extracts opcode features using IDA Pro. Scikit-learn machine learning library is used to obtain the classification algorithms.

11 Classification of malware families based on runtime behaviors

Pektas and Acarman [3] argue that currently, most malware samples are derived from existing ones, and with the obfuscation, technologies can easily evade the common security solutions. This paper addresses the challenge of classifying malware samples by using runtime artifacts while being robust to obfuscation. The authors present an online classifying system usable on a large scale with distributed architecture.

11.1 Data

The testing malware dataset is obtained from the VirusShare Malware Sharing Platform [17]. The dataset included 17 900 PE files belonging to 51 malware families, which are labeled with VirusTotal.

11.2 Feature Engineering

Dynamic features of the malware samples are extracted with running the PE file in two sandboxes. Extracted features include API calls and changes made to the OS. After running the samples, labels are acquired with VirusTotal. N-gram is used to create feature vectors of the API calls. These API calls are group into 15 different categories such as system, registry, and network activities in order to reduce the feature space. Each feature is weighted with frequency and invert document frequency weights.

11.3 Approach

The approach of the paper [3] uses the Jubatus distributed online ML framework for online classification. Multiple algorithms are used to train and test the model, such as Confidence Weighted Learning and Adaptive Regularization of Weight. The obtained results are validated with ten-fold cross-validation. The approaches testing accuracy is reached at 91.8%. A false positive rate is not reported. The paper reported that the Windows API call sequence is not changed by obfuscation techniques.

11.4 Frameworks

The features are extracted by running the malware in Cuckoo and Virmon sandboxes. VirusTotal is used for acquiring labels of the samples. Jubatus' online ML framework is used for classifying.

12 Conclusions

This paper presents nine different papers on malware detection in Android and Windows environments. The methods proposed in examined papers provided consistent detection of unseen malware with high rates.

The papers have various feature extraction methods: static, dynamic, and hybrid. Static and dynamic analyses were used often, whereas the hybrid approach was rare. Damodaran [6] states that the complexity of such hybrid techniques makes it difficult to discern the actual benefit of any one particular aspect of the technique. Regardless, hybrid models will probably be researched in the future due to the potential of combining data of various features as shown in [4] by Gheorghe et al. In general, static models are faster on classifying the malware, thus are more suitable for real-time detection even in mobile devices [8]. However, static models are at risk of being bypassed by obfuscation technologies [2].

The samples and datasets of these papers vary a lot. Thus, conducting an absolute ranking the classification methods is not possible based on the comparably small and unbalanced datasets. The performance of the classifiers is connected to the nature of the dataset. For example, in the paper by Milosevic [8], Bayesian algorithms usually require much larger datasets than SVM to train the model with higher accuracy for the specific set of features.

In malware model training, the feature and dimensionality reduction is needed to improve the accuracy of the model and reduce the training time. These papers researched use varying, often novel, approaches in order to reduce the features.

Ensemble classifiers are used to vote for the classification label [8, 16]. Based on these papers, using multiple classifiers produced slightly better results with the cost of longer training time, as multiple models need to be trained. However, ensemble classifiers are a potential addition to online training or non-real-time applications.

Using machine learning for malware detection and classification provides promising results with F-measures up to 98.5%. As the environments of detection vary, the proposed solution should also be modified to meet the needs. Based on the reviewed papers, future research includes the evaluation of hybrid models. Also, in order to compare the previously proposed models, these approaches need to be validated with a significantly bigger labeled and balanced datasets.

References

[1] U. Aijaz, A. Patra, A. Siddiq, B. Chatterjee, and M. Khan. Malware detection on server using distributed machine learning. *Perspectives in Communication, Embedded-systems and Signal-processing - PiCES*, 2(7):172–175, Nov. 2018.

[2] A. G. Kakisim, M. Nar, N. Carkaci, and I. Sogukpinar. *Analysis and Evaluation of Dynamic Feature-Based Malware Detection Methods*, page 247–258. Springer International Publishing, 2019.

[3] A. Pektaş and T. Acarman. Classification of malware families based on runtime behaviors. *Journal of Information Security and Applications*, 37:91–100, Dec 2017.

[4] L. Gheorghe, B. Marin, G. Gibson, L. Mogosanu, R. Deaconescu, V. Voiculescu, and M. Carabas. Smart malware detection on android. *Security and Communication Networks*, 8(18):4254–4272, Sep 2015.

[5] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye. Significant permission identification for machine-learning-based android malware detection. *IEEE Transactions on Industrial Informatics*, 14(7):3216–3225, Jul 2018.

[6] A. Damodaran, F. D. Troia, C. A. Visaggio, T. H. Austin, and M. Stamp. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, Dec 2015.

[7] G DATA. 8,400 new android malware samples every day. `https://www.gdatasoftware.com/blog/2017/04/29712-8-400-new-android-malware-samples-every-day`, 2017. Accessed: 26.10.2019.

[8] N. Milosevic, A. Dehghantanha, and C. K. Raymond. Machine learning aided android malware classification. *Computers and Electrical Engineering*, 61:266–274, Jul 2017.

[9] M. Damshenas, A. Dehghantanha, K. Choo, and R. Mahmud. M0droid: An android behavioral-based malware detection model. *Journal of Information Privacy and Security*, 11(3):141–157, Jul 2015.

[10] Y. Fan, S. Hou, Y. Zhang, Y. Ye, and M. Abdulhayoglu. Gotcha - sly malware! In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18*. ACM Press, 2018.

[11] P. Burnap, R. French, F. Turner, and K. Jones. Malware classification using self organising feature maps and machine activity data. *Computers & Security*, 73:399–410, Mar 2018.

[12] A. Nappa, M. Z. Rafique, and J. Caballero. *Driving in the Cloud: An Analysis of Drive-by Download Operations and Abuse Reporting*, page 1–20. Springer Berlin Heidelberg, 2013.

[13] L. Jhu-Sin and L. Chia-Tien. Binary malware image classification using machine learning with local binary pattern. pages 4664–4667. IEEE, 2017.

[14] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath. Malware images. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security - VizSec '11*. ACM Press, 2011.

[15] A. Oliva and A. Torralba. *Chapter 2 Building the gist of a scene: the role of global image features in recognition*, page 23–36. Elsevier, 2006.

[16] L. Liu, B. Wang, B. Yu, and Q. Zhong. Automatic malware classification and new malware detection using machine learning. *Frontiers of Information Technology & Electronic Engineering*, 18(9):1336–1347, 2017.

[17] VirusShare. Malware sharing platform. `https://virusshare.com/`. Accessed: 27.11.2019.

Appendix I: Selected papers, platform, year, and citations

Paper name	Plarform	Year	Citations
Smart malware detection on Android	Android	2015	9
Significant Permission Identification for Machine Learning Based Android Malware Detection	Android	2017	109
Machine learning aided Android malware classification	Android	2017	87
Scorpion: A Metagraph2vec Based Malware Detection System	Windows	2018	17
Malware classification using self organising feature maps and machine activity data	Windows	2018	29
A comparison of static, dynamic, and hybrid analysis for malware detection	Windows	2015	93
Binary malware image classification using machine learning with local binary pattern	Windows	2017	14
Automatic malware classification and new malware detection using machine learning	Windows	2017	25
Classification of malware families based on runtime behaviors	Windows	2017	19