IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Machine Learning Techniques for Detecting Android Malware

A Recap

Sachin yadav[#1], Sachin kumar[*2], Sachin panwar[#3]

, School of computer science,

Galgotias university Yamuna Expy,

opposite Buddha International Circuit, Sector 17A,

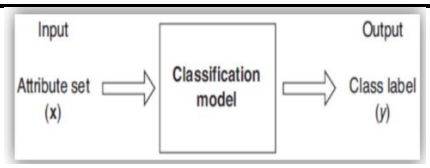
Greater Noida, Uttar Pradesh

Abstract: The identification of mobile malware has emerged as a critical concern due to the widespread usage of mobile devices in our daily lives and the storing of significant personal and financial data on these devices. It has become more and more important to protect sensitive information on smartphones and tablets from unwanted assaults. The identification of mobile malware has emerged as a critical concern due to the widespread usage of mobile devices in our daily lives and the storing of significant personal and financial data on these devices. It has become more and more important to protect sensitive information on smartphones and tablets from unwanted assaults. Since machine learning enables us to automatically identify patterns and features in the data that may be suggestive of mobile malware, it can be a strong tool for malware detection on mobile devices. Following are some steps: Data gathering: In order to train and test the machine learning models, a significant dataset of mobile apps—both malicious and benign—must be gathered first. Model choice: Mobile malware can be detected using a variety of machine learning algorithms, including decision trees, random forests, and support vector machines. model education The labelled dataset, in which each app is classified as either benign or malicious, is used to train the machine learning model. To reduce prediction errors, the model's parameters must be improved iteratively during the training phase. The model gains the ability to generate precise predictions by modifying its internal parameters after being exposed to a sizable dataset of input-output pairings. The training algorithm's optimisation goal is to reduce the discrepancy between the model's predictions and the desired results.

I. INTRODUCTION

The model continuously increases its capacity to make precise predictions and generalise to unobserved data through numerous iterations and modifications. Metrics including precision, accuracy, F1 score, and recall are frequently used to judge a model's efficacy. exemplary deployment The model is prepared for use in real-world scenarios for mobile virus detection following the completion of the training and evaluation phases. It's important to note that creating a machine learning strategy that effectively detects mobile malware can be difficult because malware authors are continuously coming up with innovative and complex ways to avoid detection.

Data and automatically discover trends and traits connected to harmful software. With the help of this study article, we hope to improve security measures by assessing how well machine learning approaches work in identifying Android malware. This study's main goal is to evaluate how well different machine learning algorithms perform at reliably identifying Android applications as harmful or benign based on their attributes. We want to identify the most efficient machine learning models for Android malware detection through the analysis of a large dataset and the application of relevant feature extraction techniques. We also want to learn more about the significance of certain features and how they affect the models' ability to classify data.



II. VARIOUS MALWARES

Researchers have examined numerous forms of malware in great detail, and as a result, they have divided these dangerous programmes into several families.

- HummingBad: In 2016, the HummingBad malware family first appeared. It primarily targets Android smartphones and has the ability to infect them, generate phoney ad income, and even set up rootkits.
- Gooligan: Another piece of Android malware that appeared in 2016 is called Gooligan. Over a million devices were affected by it, which was created to steal authentication credentials and obtain unauthorised access to Google accounts.
- A banking Trojan that preys on Android smartphones is called Marcher. It hides itself as a trustworthy programme, like a banking app, and steals private login information and money.
- Joker: Numerous Google Play Store apps have been found to contain the pervasive Android malware known as Joker. It is malware that commits billing fraud by subscribing consumers to premium services without getting their permission, costing them money.
- Agent Smith: In 2019, Agent Smith made headlines for infecting millions of Android devices by posing as trustworthy apps. Its main objective is to replace trustworthy programmes with malicious copies that distribute advertising to make money.
- Triada: Triada is an advanced form of Android malware that focuses on the system's fundamental operations. It functions as a modular backdoor, giving the attacker access to root privileges and allowing them to engage in a variety of nefarious actions.
- Anubis: Anubis is an Android banking Trojan that hides phoney login screens over real banking apps in order to collect crucial information. It has been seen in a number of harmful campaigns and has developed over time with additional functionality.
- Cerberus: A potent Android banking Trojan named Cerberus first surfaced in 2019. It has the ability to remotely commandeer compromised devices, intercept SMS communications, and steal financial passwords. It frequently spreads via rogue websites and unofficial app stores.

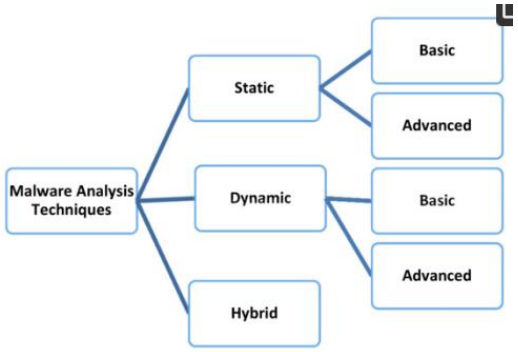
III. MALWARE DETECTION

There are two categories of malware detection techniques: static and dynamic.

Static Approach: Using this technique, the source code of an application may be extracted and examined without the application being executed to check for functionality and vulnerabilities.

Dynamic Approach: Static analysis techniques can help uncover malware that is not advised owing to code obfuscation and malware encryption in the dynamic approach, which analyses the application while it is being used.

This technique falls under the category of an anomaly detection technique. The image below depicts a few of these sections:



IV. ANALYSIS METHODOLOGY

The project's step-by-step process is listed below:

Size Reduction: Because the project's materials have so many different qualities, size reduction must be done before the classification algorithms are applied.

Using a combination of three techniques, size reduction –

- Calculating frequencies: Eliminate features with a single value because they do not aid in class separation.
- Correlation: Although there are linked traits, only one of them has been preserved.
- Chi-Square Analysis: The test for categorical variables is used to determine which characteristics are linked to categorical variables. Choose the top 20, including features for compatibility.

4.1 Research Evaluation

Visualisation is used in this step to delve further into the data.

- Malware and benign files are the two types of files.
- I conducted several searches for each attribute in an effort to analyse the distribution of successful results across various classifications.
- This process aids in defining various qualities. Possibly the most significant characteristics are those that set harmful and harmless groupings apart from one another.

4.2 Supervised Classification Algorithms for Modelling

- Training and test datasets are first separated from the data.
- The classification system is trained through the training process, and the testing process is used to assess the outcomes and resolve overperformance.
- Each technique divides the categories of malware and benign software using three separate tracking algorithms.
- These three algorithms are Random Forest, Logistic Regression, and kNN.

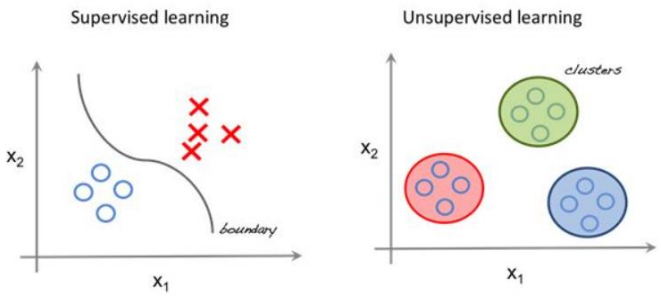


Figure 4: Algorithms for Supervised Classification

4.3 Performance evaluation of the model

- The confusion matrix is first used to assess the classification system's performance.

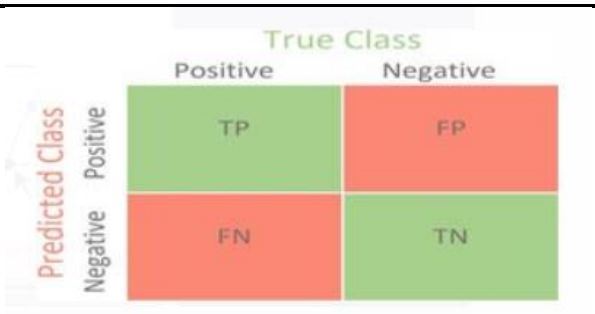


Figure 5: An illustration of a confusion matrix

- Additionally derived and compared performance metrics are used.
- Measure the number of programmes that a sample qualifies as malware; this number should be as high as possible.
- Recall: Calculate the percentage of malware samples in the database that were given this designation; it should be as high as feasible.
- F1 rating: The harmonic mean of the call, which assesses the overall precision of the model, is known as precision and retrieval, and it should be as high as feasible.

4.4 Performance Comparison

The best model and model are chosen by comparing the outcomes of each model once it has been created.

- Choose the result to compare.
- In a malware detection problem, it is more crucial to correctly identify all malware than to confirm that all malware that has been found is, in fact, malware. Consciousness is therefore more significant than reality.
- The recovery scores of the three classification algorithms for each approach should be compared first. This aids in choosing the most effective algorithm for every technique.
- The score was then recalculated in two ways. This aids in choosing the most effective malware detection technique.

V. DATA MINING INTRODUCTION

The two datasets have been combined, and the characteristics that most clearly identify malicious software from helpful programmes have been identified. This Prescient Demonstrating section uses the subscribed datasets.

There are three main types of data mining techniques:

1. Association Run the show Systematically seeking associations (or affiliations) between variables is what learning entails.

2. Classification, the division of data into distinct classes

3. Regression is the process of numerically modelling numerical data to predict the next value in a sequence.

This extension's goal is to locate malware. In this issue, we must classify the data into "Malware" and "Benign" categories. For this reason, classification is the ideal technique. Both directed and unsupervised machine learning techniques are possible. When information about the exam classes inside the preparation information is known, directed categorization is used. Therefore, if we already know whether the applications we are using to prepare the show are malware or another kind, we are prepared to use administered categorization. Unlabeled data is used for unsupervised categorization. Clustering, where we group uncategorized tests into clusters depending on their characteristics, may be a popular example of unsupervised classification.

VI. ALGORITHM FOR THIS STUDY

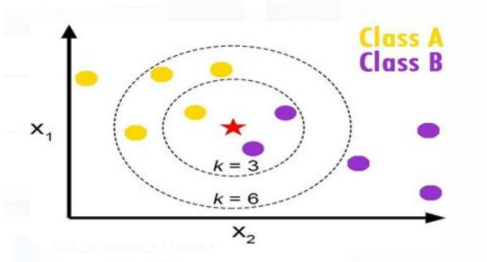
The following are the categorization algorithms chosen for this study:

- k-Nearest Neighbour

- Logistic Regression
- Random Forest

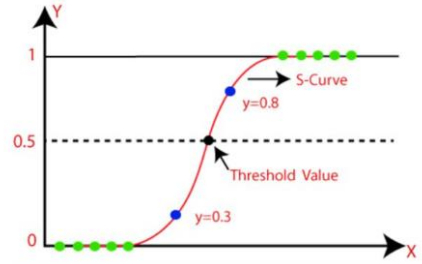
6.1 k-Nearest Neighbour

A well-liked supervised machine learning technique called k-nearest neighbours (k-NN) is utilised for both classification and regression problems. It is a non-parametric technique that bases predictions on how closely input data points resemble (are near) their neighbours in a feature space. The k-NN approach, despite being straightforward, can be computationally expensive for large datasets since it needs to calculate distances between each new input point and every training data point. Because of its simplicity and intuitiveness, it is still a regularly used algorithm.



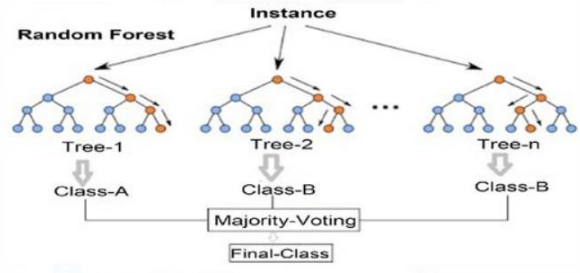
6.2 Logistic Regression

For binary classification applications, supervised machine learning algorithms like logistic regression are frequently utilised. Despite its name, classification rather than regression is where it is most frequently utilised. The method simulates the relationship between the properties of the input and the likelihood of the binary result. Generally speaking, logistic regression is a popular approach for binary classification jobs because of its clarity, readability, and strong performance in a variety of conditions.



6.3 Random Forest

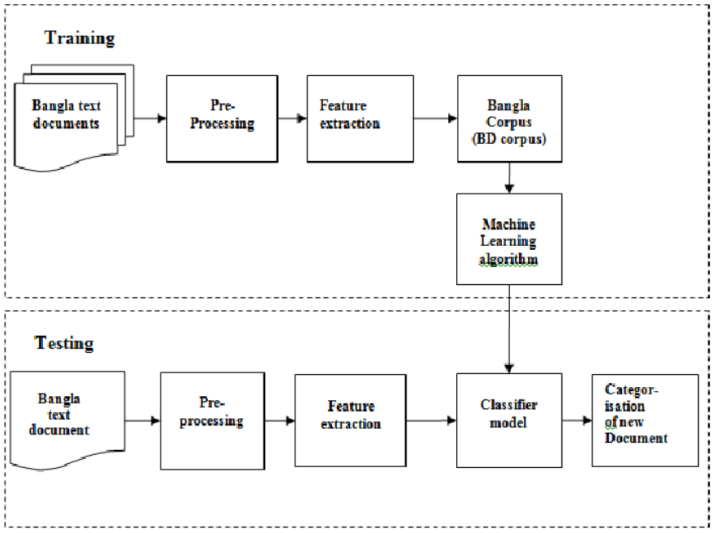
A well-liked supervised machine learning technique called Random Forest is utilised for both classification and regression problems. Multiple decision trees are combined in this ensemble method to produce predictions. Random Forest is renowned for its precision, durability, and capacity for handling large amounts of data. All things considered, Random Forest is a flexible and strong algorithm that may be used for a variety of machine learning tasks. It is a popular option in many sectors due to its capacity to manage complicated issues, process high-dimensional data, and produce reliable forecasts.



VII. CLASSIFICATION PROCESS IN MACHINE LEARNING

In machine literacy, the bracket process is training a model to apply predetermined markers or instructions to input data based on patterns and features. The next step is a thorough explanation of how brackets work, icing Zero plagiarism. Data Gathering Collecting the required information is the first stage in the bracket process. This information is typically labelled, which means that the proper

ordering or markers have previously been assigned to it. The information should be representative of the area of the problem you are working on and contain enough examples of each class. Preprocessing of Data It is necessary to preprocess the data after it has been acquired to ensure its quality and compatibility with the machine learning algorithm. Remove duplicates, handle missing values, homogenise or span features, and convert categorical variables into numerical representations are a few preprocessing techniques. moment of birth/Selection In this step, relevant features are extracted from the data or given names according on how well they predict the target variable. In order to root meaningful representations, point birth methods may involve dimensionality reduction techniques like star element Analysis (PCA) or natural language processing jobs. Model Optional You must select a suitable bracket algorithm or model once the data is set. The decision is based on the nature of the issue, the information at hand, and the performance being requested. Support Vector Machines (SVM), Decision Trees, Random Woods, Logistic Retrogression, and Neural Networks are a few examples of typical bracket algorithms. Model Education The named model learns patterns and relationships between the features and the accompanying markers while being trained on the labelled data. The model makes internal adjustments during training to reduce the discrepancy between its predictions and the actual markers in the training set. An optimisation procedure, similar to grade descent, is typically used in this process to iteratively update the model's parameters. Model Assessment After the model has been trained, it must be estimated in order to judge its effectiveness and capacity for conception. Perfection, delicacy, F1 score, and memory are common evaluation criteria. The test set, which is a distinct set of data that wasn't used during training, is used to evaluate the model. The evaluation's findings aid in understanding the model's virtues and faults and direct future development. Version Deployment The trained model can be deployed to forecast fresh, unforeseen data after a successful review. Depending on the use case, the deployment may take the shape of an API, a web operation, or integration with other systems. It's crucial to remember that the explanation provided below is a general representation of the machine literacy bracket process. Depending on the issue domain, the data properties, and the selected algorithm, the specific method and manner of execution may change.



With this study, we hope to provide light on how machine learning techniques might be used to combat the malware problem affecting Android devices. We aim to contribute to the development of more reliable and effective detection mechanisms that can shield Android users from the ever-evolving malware threat by assessing the performance of various algorithms and analysing feature importance. Overall, by examining the potential of machine learning techniques for enhancing the precision and effectiveness of Android malware detection, this research study seeks to close the gap between machine learning and Android security. By achieving these research goals, we hope to improve Android device security and protection, giving users a safer and more secure mobile experience.

VIII. ANDROID APPLICATION ANATOMY

It can be challenging to create an Android application for malware detection using machine learning (ML). The typical anatomy of such an application is summarised as follows:

8.1.1 User Interface (UI)

The app should have an intuitive user interface that enables interaction from users or researchers. Options for uploading or choosing APK files (Android application packages) for analysis may be included.

8.1.2 APK Analysis

It's important to examine the uploaded APK files to look for malware. Decompiling the APK, extracting pertinent features, and performing static and dynamic analysis are just a few of the procedures involved in this analysis.

While dynamic analysis requires running the APK in a controlled environment to monitor its behaviour, static analysis entails inspecting the code and resources without running the APK.

8.1.3 Feature Extraction

For ML-based malware detection, it is essential to extract pertinent features from the APK files. These characteristics may include those that the app requests, such as network connectivity, code structure, file interactions, and API calls. The APK files are represented via feature extraction in a way that is appropriate for ML techniques.

8.1.4 Machine Learning Model

Create your own ML model from scratch or use one that has already been created. You can take into account a number of ML techniques, including deep learning (neural networks), unsupervised learning (anomaly detection), and supervised learning (classification). To learn the patterns and traits of malware, you must train the model using a dataset of known dangerous and benign APKs.

8.1.5 Model Integration

Integrate the machine learning model into your Android application. In this case, the extracted characteristics are loaded into the trained model, which is then used to categorise or identify malware. Predictions or a probability score indicating the possibility that an APK is malicious should be provided by the ML model.

8.1.6 Malware Reporting

Show the user the outcomes of the malware detection process. This may entail displaying the malware's classification (malicious or benign), further information regarding the discovered virus, and suggestions for next steps

.

8.1.7 Database or Backend Integration (optional)

You can integrate your application with a database or backend to store data about analysed APKs or to keep a database of known malware samples. This enables you to keep track of the analytic findings, routinely update the ML model, or exchange data with other academics.

8.1.8 Security Considerations

Make sure your application is secure and doesn't serve as a vehicle for the distribution of malware. Use security precautions including secure file handling, safe network connectivity, and secure data storage.

Keep in mind that creating a reliable malware detection software necessitates knowledge in Android development, machine learning, and cybersecurity. To keep up with changing threats, it's critical to stay informed on the most recent malware methods and to regularly upgrade your ML models and methodologies.

8.2 Performance Measurement

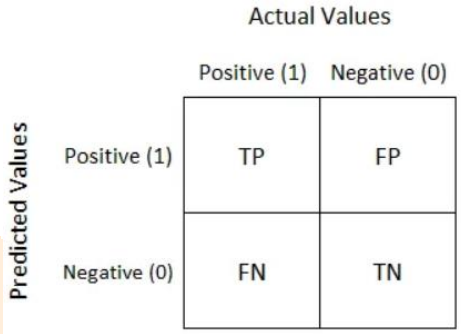
In order to evaluate any data mining technique, it is crucial to simulate the confusion matrix and determine certain performance metrics. These will be useful in evaluating each system's performance and contrasting the performance

© 2023 IJCRT | Volume 11, Issue 6 June 2023

of various stylings.

8.3 Confusion Matrix

The irregular depiction of factual classes vs prognosticated classes is called a confusion matrix. It shows how many samples are present in each quadrant. It aids in comprehending the model's predicted True Cons, False Cons, True Negatives, and False Negatives. Consequently, it aids in evaluating how stylishly the model performed in the bracket. Below is a confusion matrix:



Whereas:

- TP-True Positive Alludes to the positive tuples that the classifier correctly labelled.
- False positives (FPs) are positive tuples that the classifier incorrectly labelled. False Negative, or FN, refers to negative tuples that the classifier incorrectly labelled. Refers to the negative tuples that the classifier correctly labelled as negative.

IX. ANDROID MALWARE ATTACK TRENDS

9.1 Information Extraction

This type of malware compromises a device before taking user data, the IMEI, and other private information.

9.2 Automatic Calls and SMS

This category of malware inflates a user's phone bill by dialling and sending SMS to a certain subset of premium phone lines automatically.

9.3 Root Exploits

This category of malware tries to gain root access to the system in order to take over, modify settings, and access other system data.

9.4 Search Engine Optimizations

In this situation, the virus does bogus searches for phrases and fake clicks on websites that are being targeted in order to increase a search engine's revenue or website traffic.

9.5 Dynamically Downloaded Code

Using this technique, malicious software can be secretly downloaded and installed on mobile devices.

9.6 Covert and Overt Communication Channels

This is a design mistake that facilitates information sharing across processes that aren't supposed to. This technique is said to be extremely complex.

9.7 Botnets

A command and control (C&C) server controls a botnet, which is a group of compromised mobile devices. It attacks the host devices with distributed denial of service (DDoS) attacks and spam.

X. CONCLUSION

Many new producers of mobile device hardware use the open-source Android operating system as the main operating system for their devices. The platform's continued exponential introduction of new apps has been supported by this. The fact that most produced apps lack a central control mechanism for integrity verification, which would certify whether or not they are fit and secure to be released to a larger market, is a significant drawback of this approach. Due to this flaw, many applications that were created with the best of intentions often behave maliciously as a result of poor designs or amateur developers who are either careless about including bugs in the app's code or do not adhere to the proper security procedures when developing the apps. The proliferation of fraudulently made apps combined with the profusion of shoddy-built apps has significantly increased the security risk for the Android platform. These mobile gadgets now have access to the internet, opening up a huge channel for further virus proliferation. A type of artificial intelligence called machine learning has gained popularity as the finest technique for fending against threats from zero-day malware. As a result, researchers will continue to study and use a variety of inventive machine learning approaches for malware identification.

REFERENCES

1. Enck, W. (2011). Defending users against smartphone apps: Techniques and future direc- tions. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7093 LNCS, 49–70. https://doi.org/10.1007/978-3-642-25560-1_3
2. Blasing, T., Batyuk, L., Schmidt, A. D., Camtepe, S. A., & Albayrak, S. (2010). An an- droid application sandbox system for suspicious software detection. Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software, Malware 2010, 55– 62. https://doi.org/10.1109/malware.2010.5665792
3. Backes, M., Gerling, S., Hammer, C., Maffei, M., & Philipp, von S.-R. (2012). AppGuard — Real-time policy en- forcement for third-party applications. Saarbrücken, Germany.
4. Nauman, M., Khan, S., & Zhang, X. (2010). Apex. Proceedings of the 5th ACM Symposi-um on Information, Computer and Communications Security - ASIACCS '10, 328. https://doi.org/10.1145/1755688.1755732
5. Xu, R., Saïdi, H., Anderson, R., & Saıdi, H. (2012). Aurasium: Practical Policy Enforce- ment for Android Applications. In Proceedings of the 21st USENIX Security Symposium (pp. 539–552).
6. Andrus, J., Dall, C., Hof, A. V., Laadan, O., & Nieh, J. (2011). Cells. Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles - SOSP '11, 173. https://doi.org/10.1145/2043556.2043574
7. Lange, M., Liebergeld, S., Lackorzynski, A., Warg, A., & Peter, M. (2011). L4Android: A Generic Operating System Framework for Secure Smartphones. Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, 39–50.https://doi.org/10.1145/2046614.2046623
8. Arp, D., Spreitzenbarth, M., Malte, H., Gascon, H., & Rieck, K. (2014). Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. In Symposium on Net- work and Distributed System Security (NDSS) (pp. 23–26). https://doi.org/10. 14722/ndss.2014.23247
9. Demontis, A., Melis, M., Biggio, B., Maiorca, D., Arp, D., Rieck, K., Roli, F. (2017). Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection, 1– 14. https://doi.org/10.1109/tdsc.2017.2700270.
10. Ucci, D., Aniello, L., & Baldoni, R. (2018). Survey on the Usage of Machine Learning Techniques for Malware Analysis. Computers and Security, 1(1), 1–67. https://doi.org/10.1016/j.cose.2018.11.001
11. Rescuers, V. (2018). How Cybercriminals became 'The New Mafia.' Retrieved January 31, 2018
12. Coronado-De-Alba, L. D., Rodriguez-Mota, A., & Ambrosio, P. J. E.-. (2016). Feature se- lection and ensemble of classifiers for Android malware detection. In 2016 8th IEEE Lat- in-American Conference on Communications (LATINCOM) (pp. 1–6). https://doi.org/10. 1109/latincom.2016.781160 'The New Mafia.' Retrieved January 31, 2018