HYBRIDROID: Detecting Malicious Applications on Linux Powered Smartphone's by Hierarchical Machine Learning Algorithms.

Mr. Rahul Pawar[1]* and Dr. C Mahesh[2]

[1]Research Scholar, Veltech University Avadi, Chennai [2] Department of CSE, Veltech University, Chennai

Email ID of corresponding Author: rahulpawar8087@gmail.com

Abstract - From the last two decades mobile phones are a very important part of our lives and Android is known to be the most popular working framework for mobile phones. Android is having a quickly expanding scenario that has pulled in the intruders for creating malwares. Android permits downloading and installation of applications from android market and third party websites. This offers malware engineers a chance to put repackaged malicious code in the legitimate applications. Many malware detection techniques are proposed and various discovery frameworks have been created which utilizes static and dynamic investigation techniques. However, the existing models are not sufficient for accurately detecting malicious applications in the Android Smartphones. Many Malware Detection Techniques are good in specific areas like selection of maximum and informative features, Accuracy of Detection, on device analysis and Minimum Performance Overhead. To increase the ac- curacy and efficiency of Malware detection we proposed multilayer models using fusion of algorithms in the field of machine learning. The proposed model is a complete solution by considering all best possible algorithms in each area maintaining efficiency and accuracy in the Detection System. The research contributes in multiple layers. First, In Feature extraction Maximum features are considered including user level and Kernel Level. At Feature Selection most informative Features selected for accurate results. A novel fusion approach is proposed by using various machine learning algorithms for classification of applications according to the behavior. This provides high efficiency and accuracy by combining Static and dynamic analysis of Malware detection. Experimental results have shown high accuracy with great efficiency in terms of memory as well as Power.

KEYWORDS: Malware detection, Android Permissions, Deep learning, Fast algorithm, ID3, PRNR

1 INTRODUCTION

Android smartphones are known as the most popular smartphones as far as operating systems are concerned. As per Gartner's 2019[1] report, Android captured 74 percent of its market share and dominated all other competitors. Figure I shows Android's market share of 2019. And as a result, many Smartphone vendors are using the android operating system on their smart phones. With this popular nature, attackers create malware and result in the rapid growth of Android malware. Figure II shows an increasing rate of Android malware in 2019.To protect devices from these Android Malwares many antimalware systems are proposed to tackle this malware. Such techniques can be useful in detecting the malicious codes attached as a payload when a specific application is executed. Since anti-malware is static, it is not enough to identify the dynamic and encrypted malware.

FIG 1: Android Market share

Analytical fusion technique is applied by integrating static detection and dynamic machine learning techniques to achieve high precision. By applying this strategy, the re- liability of the mobile system is compromised. Existing static and dynamic monitoring antimalware methods are being applied on the Smartphone. Whereas, some of complex analytics techniques are applied on the server end. Rather than testing on the machine end the android application will submit the extracted features on the host machine fusing three algorithms of machine learning. It will not run overall applications on the computer and android Smartphone and this saves the power and memory of the computer and increases the overall system's performance.

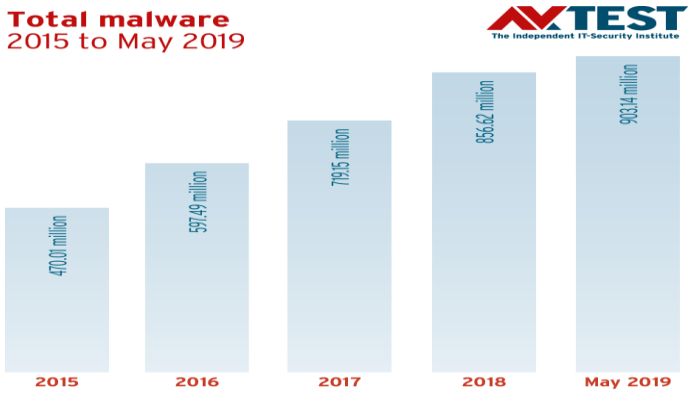

FIG 2: Number of Android Malwares detected year wise

In this paper, a Hybrid model of malware detection in android is proposed as HYBRIDROID. Three levels are combined into a Hybrid approach for malware detection:

1. Features extraction and selection using PRNR
2. Local analysis & Remote Host
3. Machine Learning Intelligence

In the Deep learning stage, First features are extracted from android applications and given as an input to feature selection algorithms. Selection of features is done to determine the appropriate features that provide good and valuable knowledge about the application

behavior. Sets of user-level and kernel-level features are used to this end. Device system calls are taken out during run-time for detailed malware analysis. For feature extraction and selection Different machine learning algorithms are used; combining their efficiency to achieve a more precise output when we have used the fusion technique. The proposed system helps in overcoming the limitations of existing systems also resource efficiency of the existing malware detection systems can be increased. Through HYBRIDROID, we provide solutions to Android Smartphone users by selection of extracted features of the installed applications. HYBRIDROID also assists Android mobile users to make them aware of their various applications and their complex behavior. Chapter II includes a survey of literature of the latest malware detection approaches. In addition, the document flow is as follows. In Chapter III, proposed solution is explained namely HYBRIDROID, which overcomes the limitations of previous anti malware systems and high accuracy is obtained by low resource consumption. Chapter IV contains the experiments performed and results obtained during development of HYBRIDROID and in the Last Chapter conclusion and future scope is described.

2 SURVEY OF LITERATURE

There are two malware detection techniques static and dynamic, as per analysis both approaches have limitations in terms of power and memory due to which work to be done towards the development of fusion techniques for antimalware development. Although the use of fusion techniques in detection has increased the accuracy in malware detection more work needs to be done. Following are the categories of the existing Android malware detection techniques:

1) Static Malware detection.

2) Dynamic Malware Detection

3) Machine learning for Malware Detection

2.1 STATIC MALWARE DETECTION

This group includes methods for Android malware detection that use static and dynamic hybrid analytics to achieve a high degree of accuracy.

A sandbox of Android devices was implemented by Bläsing et al.[2], which can detect suspicious activity in Android applications by static and dynamic detection. The Android program is first decompressed and then decomposed with the “Baksmali tool”. The decompiled “smali” files are then screened to remove static patterns. The Monkey frame is used for running random inputs of users including moves, browsing and clicks. In the kernel space, AA Sandbox uses the fully-controlled program execution kernel module to exe- cute the program and logs by depriving system calls. The static logs of behavior and the dynamics are the basis of mathematical action vectors. Such vectors may be used either manually or automatically to identify suspicious application behavior, and to classify applications. Various Experiments have shown, through implementing handwritten software called fork bombs using Runtime Exec generating a vector of static behavior and a vector of dynamic behavior,

ensuring the proper functioning of the proposed mechanism. The machine produced performance clearly demonstrates that the software uses the principal weakness of Runtime.

Zhou et al.[3] introduced Droid Ranger in popular Android markets to detect known and zero-day malware. They have acquired 204,040 Android apps including 75 % from play store [4], and 25 % from four alternate Android app markets. Droid Ranger uses static features to detect known malware in two stages: it uses initially vital permissions, which malware needs to perform the desired functionality, to document malicious applications. The number of applications required for further study in the second stage is greatly reduced by this method. Once the permit is updated, the behavior is filtered as the system's activity is determined from manifest file information and calls to the API. Such specifics are mapped to the malware recognizing and detecting code of conduct.

Android makes use of the Dex Class Loader [5] to dynamically load the java code. Through analysis, authors discovered 1055 applications using this class. For Plank-ton, the zero day form of malware detection, it's heuristic. The other heuristic is locally linked to the dynamic load of native code and is not likely to occur in a non-default directory application which contains native code. This heuristic helps to recognize the malware which uses the root privileges of the OS kernel to access it. That was the way to discover Robot Dream Light [6], an official Android malware of zero days. In the second stage, it performs complex heuristic-based execution tests for the evaluation of applications' runtime behavior. To detect malicious behavior of the dynamically loaded java code, API calls and arguments are registered, and system calls are monitored to detect malicious activity by dynamically loading native code of the application. Within 204,040 Droid Ranger applications, 211 infected applications have been successfully identified, with 40 applications containing zero-day malware and 171 applications infested with well-known malware.

2.2 DYNAMIC MALWARE DETECTION

This category includes anti malware that is hybrid from static and dynamic analyzes, and techniques of machine learning. Droid-Dolphin has proposed using static or dynamic malware detection by Wu et al .[7], the cloud-based Droid-Dolphin detective software. The scheme proposed consists of four phases: pre-processing, emulation and control, feature extraction and computer education. The pre-processing stage is to track malicious API calls by using API-Monitor[8]. This approach helps detect malicious API calls by tracking call logs while an API call is being triggered by the application. During this process, the authors logged 25 API calls. The emulation and control applications were installed on virtual Android computers with APE[9, 10] and Droid-Box. Total 13 Execution events reported by Droid-Box. The proposed software allows APE to simulate a GUI-based scenario, and helps navigate the code path of the application to detect malicious behavior. Apps are represented using the Template N-gram. The functions API monitor and APEBOX can be used as SVM input [11] and LIB-SVM [12] to implement models for malware detection. The authors used large-scale training datasets to test the efficiency of the proposed software, which included 32000 malicious and 32000 benign devices. The study spectrum contains 1000 safe services, and 1000 negative ones. Experimental findings show that the 86.1 % accuracy with low

performance.

Wang et al.[13] proposed a hybrid detection scheme for malware that detects proven malwares and their variants through signature-based malware and zero-day malware detection. Dynamic functions for static analysis are extracted from the Manifesto, and dex-files are disassembled using the Android Asset Packaging Tool [14]. Cuckoo Droid [15] and a robot simulation tool [16] are used to improve Cuckoo Droid to remove unstable features. Such static and dynamic features then become vector space where the value of every dimension is 0 or 1.Specific screening methods for harassment and the identification of anomalies are implemented to increase accuracy and performance. After the functions have been selected the Linear SVC classifier [17] is implemented. The software will then test whether the code is labeled as malware whether it is typical malware or a new form of malware. This categorizes the malware based on the signature similarity and updates the training database. In comparison, if the detection of malicious activities cannot be done by the system, an error would be detected. It uses the SVM classifier One-Class [18]. When any unusual activity is detected, the system is identified as a hazard of zero days, and the training log is updated. The results indicate that misuse detection can detect a true positive rate of 98.79%. And the detector of anomalies can detect a true positive rate of 98.76%.

Shijo and Salim[19] suggested a hybrid malware detection strategy that combined static analysis with machine-learning dynamic analysis. PSI is extracted from the binary code as static features during static analysis. Cuckoo's malware analyzer is used by running the program to uninstall complex features. API call logs and series extracts are used to differentiate between malicious and friendly applications. The call sequences of the API are analyzed with the n-gram method that creates grams for the API for each file and specifies the incident frequency. The API call gram is omitted and the rest of the call grams used to construct vector functions are omitted under a threshold value. The proposed scheme should combine these inputs into two distinct classification techniques for each file; to build an integrated function vector. The vectors are static and dynamic. The participants Weka tool [20] and the two SVM [21] machine learning methods [22] and Random Forest [23] were employed in assessing the efficacy of the approach proposed.The tests indicate unknown programs with 98,7% accuracy.

Droid-Detector is an Android online malware identification program [24]. This con- ducts dynamic and static analyzes on remote servers to differentiate between malware and legitimate applications. Allow and fast API calls are used for static analysis. These static functions are extracted using Baksmali[2] and TinyXml[25] tools like 7-zip[26] A XML-Printer2[27]. The Droid-Box platform for dynamic analysis removes complex behavior. Each Android application runs in the Droid-Box for a particular period of time, and its complicated behavior. The derived characteristics are then transformed into the vectors of both the static and dynamic functions. In the deep learning model, these features are given as inputs for malicious application detection. Authors used 20,000 legal Google Play store applications and Contagious Web malware software, 500 applications, and Genome project Droid-Detector, 1260 smartphones, to check out. Experimental findings show that 96.7 per cent accuracy of malware can be detected. Dynamic processing on the remote server takes place

for a certain time. The key downside is that malware which is not malicious can escape detection during the monitoring process.

3 PROPOSED SYSTEM

HYBRIDROID is a platform for Android apps for hybrid malware detection at 4 stages. It is hybrid between the following four stages for analyzing and detecting malware.

3.1 STATIC AND DYNAMIC ANALYSIS ON REMOTE HOST

At Stage 1, the hybrid static and dynamic tests have a high degree of precision in integrating two analytical techniques. This checks the program code by using a static analysis and measures the actions of the malicious software. Static characteristics from the manifest code and program dex code are derived from a static analysis method. Static analyses with a little change motivated by Drebin[28] use the same series of statics to achieve high detection accuracy. Table 1 shows different classes identified for on device analysis.

TABLE 1

Different Classes for Dynamic analysis

Sr.No	Class Category	Details of Class
1	CLASS A	Hardware Components Required
2	CLASS B	Permissions Required
3	CLASS C	Components for the application
4	CLASS D	Calls with suspect API
5	CLASS E	Calls with limited API

Android Product Packaging Tool and Baksmali tool are used to remove all the static features above. The call tracing system analyzes applications based on legal Android smartphones. Device calls are used as complex functions. These machine calls enable us to overcome the limitations of static analysis and analyze the application's actions in real time. Next Level is a combination of a local and a remote host. The remote host conducts extensive static analyses to achieve more precise results. The complex analysis is carried out at the local host to get practical feedback rather than using any coded Monkey Runner device that creates changed, unrealistic inputs. User awareness enables the production and trans- mission of service call logs to a remote server. Remote repository tests machine actions continuously on the basis of missing logs and static functions.

3.2 APPLYING MACHINE LEARNING ALGORITHMS

Stage 2 gives feedback to the machine learning intelligence platform for the identification and analysis of suspicious activity in unknown apps with the feature vectors built out of the evaluated code. Both programs are either malicious or good natured. In HY- BRIDROID,

identification is performed on remote hosts such that all of the training data collection is stored in a resource-rich storage memory. In the end, this reduces overhead capacity.

Figure 3 displays the architecture of HYBRIDROID. The main elements of this model are as follows.

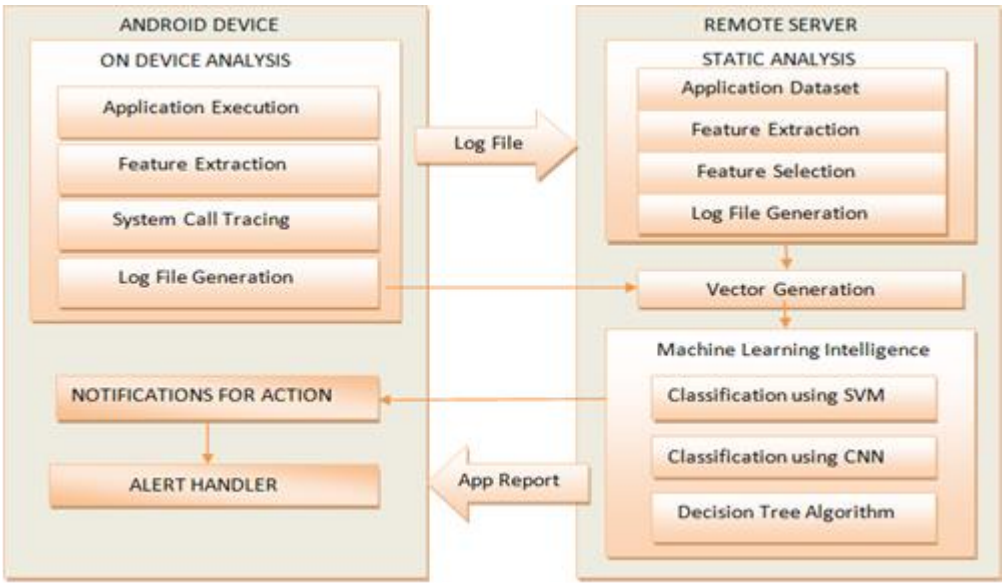

FIG 3: Architecture of HYBRIDROID

3.3 ANDROID CLIENT

An Android interface app was created for customer ending of HYBRIDROID. It provides a user-friendly interface which includes all the currently installed applications on a computer. This interface enables users to use any program through the client framework of HYBRIDROID. The program runs seamlessly while a user opens and every other program runs via HYBRIDROID. HYBRIDROID tracks context processes and does not affect any software programs. If a system app is running, it does not have to test for maliciousness or not, because the developer is inserting system applications into the computer. In comparison, HYBRIDROID links the S-trace module for that program and monitors the user's call method while this application is installed by the user from every app store. For eg, Messaging app is a device program, so HYBRIDROID will not connect to S-trace when run by a user and if Subway Surf runs HYBRIDROID would automatically hook the S-trace when installing a user program from the Google Play Store. S-trace only records computer calls, instead of intercepting them.

HYBRIDROID keeps track of the program's system calls and generates a log. The log provides a list of system calls including call names and counts for every system call. The code identifiers for that program are also forwarded to the server for instant start of the static analysis. Application Identifiers includes the name, version name and business name of the program. The count for each system call is sent to the server when the log file is created. If no call method in the log file is identified, the '0' value is passed on. The SQL server files are managed on the server where device call logs are stored in different apps from various App stores downloaded.

3.4 REMOTE SERVER

The HYBRIDROID resource is efficiently translated to memory and Android managed devices using a full static analysis on a remote host. The method of static analysis is shown in Figure 4. After acquiring HYBRIDROID device applications IDs the first time the system scans the program using the name of the product in its catalog of previously listed software. Where the program is located in a database, the classification summary will be submitted to the customer order. If the program package name is not identified, the software will be downloaded from the app-store. The static functions are removed from the program kit until it is downloaded. The kit is decompressed by Android Resource Packaging Software. This module opens and unpacks the object file to .dex. The software gathers all necessary privileges, application parameters, filtered intents and hardware features from a manifest file used by the client.

3.5 FEATURE SELECTION (Features and accelerated test Algorithm)

Feature selection algorithm is implemented for features like Permissions and System calls. In this Feature selection algorithm, use of graph clustering is done with the application function and it is grouped using the graph clustering technique (GCT) [29]. From GCT by grouping the APK into different clusters as per usability of permission the cluster obtained.

From the Matrix (1), here x_{ij} represents the binary value of i^{th} apk for j^{th} permission. FAST algorithm-introduce the concept of membership of i^{th}APK as δ_i. Where, $\delta_i \in \{C_1, C_2, \ldots, C_k\}$, is the membership value of i^{th} APK over k cluster. C_r Represents the cluster of APK which has common or permission access. Our Target is now narrow to check the APK having malicious permission or maximum permissions. Thus, this feature selection algorithm generates a high possibility, self-directed and useful feature. Minimal spanning decision tree is utilized for selecting the relevant permissions from extracted.

-Remove unrelated features.

-Create MST from the linked once

-Selection relevant feature is done by partition MST

Table 2 : Fast Algorithm

Algorithm: FAST

Initialize : S=∅ , sum[1…I, 1..Y] = 1, Z[1..I] = Y

Gain computation :

[axGain = 0

or f *in feature space* F *do*

$\hat{\alpha} = arg\ maxG_{SUf}(\alpha)$

$\hat{g} = maxG_{SUf}(\alpha)$

if $MaxGain < \hat{g}$ *then*

$MaxGain = \hat{g}$

$f^* = f$

$\propto^* = \propto^{\wedge}$

Feature selection:

$S = S \cup \{f^*\}$

If termination condition is met, then stop

Model adjustment :

or instance, i such that there is y&&$f^*(x_{i,}y) = 1$ do

[i] - $= sum[i,y]$

$sum[i,y]X^{=}\exp(\propto^*)$

[i] + $= sum[i,y]$

o to step 1.

Table 3: List of Permission Selected after the Execution of FAST algorithm

P_j *is only used in MAPK* **(Output =1) and BAPK (Output =-1)***					
1	READ_CALENDAR	10	READ_PHONE_STATE	19	PROCESS_OUTGOING_CALLS
2	WRITE_CALENDAR	11	READ_PHONE_NUMBERS	20	BODY_SENSORS
3	CAMERA	12	CALL_PHONE	21	SEND_SMS
4	READ_CONTACTS	13	PHONE_CALLS	22	RECEIVE_SMS
5	WRITE_CONTACTS	14	READ_ LOG	23	READ_SMS
6	GET_ACCOUNTS	15	WRITE_ LOG	24	RECEIVE_WAP_PUSH
7	ACCESS_ LOCATION	16	ADD_VOICEMAIL	25	RECEIVE_MMS

8	ACCESS_COARSE_LOCATION	17	USE_SIP	26	READ_EXTERNAL_STORAGE
9	RECORD_AUDIO	18	READ_PHONE_STATE	27	WRITE_EXTERNAL_STORAGE

***MAPK: Malicious and BAPK: Benign**

Table 3 shows 27 selected permissions after implementation of FAST algorithm on total 324 permissions; the important and informative permissions only are obtained from total number of permissions. Same Applied for System Calls and 10 System calls are selected after implementation of the algorithm from total 43 System calls.

3.6 MACHINE LEARNING INTELLIGENCE

Features are input for system classification. Machine learning algorithms are used to train distinguishing malicious and benevolent programs automatically. Machine learning refers to the recognition rules for isolated elements that are manually designed. V.3.6 is the method of Weka used in the classification of machine learning. In the analysis of malicious programs HYBRIDROID uses SVM classifiers. SVM applies to vectors for both a specific purpose, a static vector and a dynamic vector and is suitable for malicious applications and legal applications on the Drebin dataset. The program is marked as legal or malicious by SVM.SVM can have three possibilities, since it is implemented separately to static and dynamic function vectors.

For both static and dynamic vectors, application is defined as real. In both static and dynamic vectors, the program is marked as malicious. The classification results of all research, for example, include a paradox. Application is regarded as true for dynamic function vector static vector and malware and vice versa. The final classification decision is then taken. The program is marked as ’valid’ because all the outcomes of static analysis and dynamic analysis show a valid program. Program is marked as ’malware,’ as results in evaluation are considered malicious on all analyzes. The application is called "risky," where either objective analysis or legal second analysis declares the application to be malicious. Such results are communicated to the consumer client whether the program is malicious or dangerous and the user receives an alert.

3.7 APPLICATION BEHAVIOR EXPLANATION

Once the server results have been made, the client application will be forwarded to the true Android device. HYBRIDROID informs users of the program behavior, under review, without interrupting any user activity. This also describes the framework to users for device behavioral understanding of the apps. Not only does the program distinguish between the applications ’legitimate and malicious activities, it also gives Android users sufficient information about the actions performed by the application. It includes the name of your program, the name of your package, permission, hardware and facilities, version number, date of start of the program and last update date. In order to decide whether to use the

application or to take a call for uninstalling the application, users also can see information about programs.

4 EXPERIMENTATION & RESULTS

Different experiments and the results are discussed in this section. Three machine learning algorithms are used during the development of the method proposed, and the fusion effect is applied to increase precision and achieve performance. Several rubrics can be used for measurement parameters and for the efficiency of these machine learning algorithms.

Accuracy is the number of samples that a classifier correctly detects, divided by the number of all malware and Benign (i.e., non-malicious) applications

Accuracy ACC,

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} * 100 \quad (1)$$

Malicious applications in the dataset are known as optimistic instances in this research since we are involved in detecting malicious applications. The truly positive rate is therefore the relation between the correctly defined number of malicious applications and the number of malicious applications in general.

Precision is the ratio of predicted malware that are correctly labeled a malware, and is defined as follows:

Precision PR,

$$P = \frac{TP}{TP + FP} * 100 \quad (2)$$

Recall or detection rate is the ratio of malware samples that are correctly predicted, and is defined as follows:

Recall RC,

$$R = \frac{TN}{TN + FN} * 100 \quad (3)$$

F-Measure is the harmonic mean of precision and recall, and is defined as follows: F-Measure F-M,

$$F - M = \frac{2 * TP}{2 * TP + FP + FN} \quad (4)$$

4.1 DATASET

To evaluate the proposed model, 4509malware samples from Virus Share [30] and 115 samples from Malgenome [31] were used. In addition, 23335 Non malicious samples are downloaded from android market [32] in between March 2018 and April 2019was used. Total number of samples used in our experiment is 27959. Data set or sample details are shown in table 4. All APKs are tested with virus total's internet malware scanning service tools (VTIMSST). After Implementation the result of the proposed system shows that the APK's downloaded from the android market are benign APK's. And APK's which are downloaded from the malgenome are detected as malicious APK's.

Table 4: Android Application Dataset

Sr. No	**Number of APK**	
Benign Apk	23335	[32]
Malware samples	4509	[30]
Mal genome	115	[31]
Total	27959	

4.2 PERFORMANCE ANALYSIS

In order to make a static test module successful, an experiment is performed in HY-BRIDROID, to find the most useful features against malicious applications. The use of permission in maximum applications is a subset of those required by the applications during the selection of features. The function set is selected to eliminate the use permission function set as requested. Network addresses used in the software are also those found in the Google Play Shop. There is therefore no consistent way of distinguishing remote Host network addresses that are malicious or legit for any application which is built by different developers. As a result, a function set for network addresses is also removed. The following six collections of features are used for static analysis. A test is carried out to ensure that the selected six feature sets achieve high precision in comparison to the 8 feature sets.

The first trial applies a SVM Algorithm to a dataset. The data set is broken up by random means in known (72%) and unknown (28%) partitions. This repeats the same loop ten times and calculates the average results obtained for each sprint. The average accuracy at 0.008 or 0.7% was false positives at 96.97%. The evaluation of the average positive and negative values by the eight datasets used by Drebin shows in Table 5.It also displays an overall malware detection of HYBRIDROID. Classifiers can be shown to be more robust as TPR shows improvement as compared to MADAM and SAMADROID. In addition, the FPR of HYBRIDROID is low, 0.5% compared with that of Drebins

Table 5:

Comparison of Detection Systems based on Hit Rate

Detection System	TPR	FPR
Drebin	94	1

SAMADROID	98.5	0.5
HYBRIDROID	98.6	0.4

Second experiment is carried out in order to check the efficiency of other classifiers on the dataset of six feature sets. Machine learning algorithms such as SVM, Decision tree and Naive Bayes are implemented. 72% of the data is used as a training set and 28% is used to track. HYBRIDROID was found to be 92.3 percent accurate at an incredibly low 0.4 percent false positive rate. Though SVM is more reliable than ID3, it offers the lowest genuinely positive rate at cost. HYBRIDROID has the highest true positive score, which shows a classifier's ability to correctly identify the malicious software.

Table 6:

Performance Analysis of Machine Learning Algorithms

Methods	**Accuracy**	**Precision**	**Recall**	**F-Measure**
ID3	72.6	73.6	70.5	78.6
SVM	82.4	85.6	81.3	87.6
Naive Bayes	65.2	69.3	67.2	71.6
HYBRIDROID	92.3	94.7	91.8	95.4

HYBRIDROID is assessed based on systems calls in this experiment. As an input into the machine learning method, Weka, vectors for the device call frequency function are provided. To the system calls dataset are added relevant machine-learning classifiers such as the SVM and Decision Tree. The data set for each classifier receives a 5-fold cross-validation and the results are compared. The Accuracy, Precision, Recall and the F-Measure of these classifiers are listed in Table 6 and Figure 4.

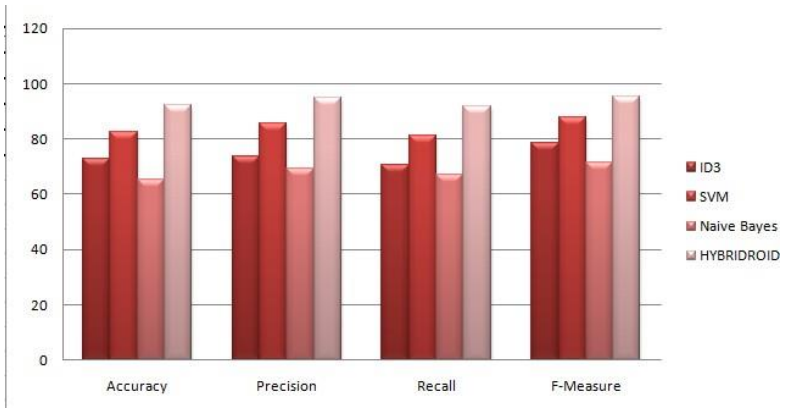

FIG 4: Performance Analysis of Machine Learning Algorithms

4.3 PERFORMANCE OVERHEAD

This section addresses two tests performed on real Android devices to determine the over- head deterioration caused by HYBRIDROID. Overhead production caused by MADAM is high compared to HYBRIDROID, the results reveal. This experiment shows the over- head created by the ARMOD ROID client program. HYBRIDROID's overhead efficiency is calculated by means of a common test tool, i.e. A Standard Edition Quadrant Application. Benchmark testing was carried out on the Samsung Galaxy Grand Prime (CPU Quad-core 1.2GHz Cortex-A53, RAM 1 GB). The Mobile runs Android 5.1 on Lollipop. Table 7 shows the performance overhead caused by the Proposed Malware Detection System

TABLE 7

Results -Performance Overhead of HYBRIDROID

Test	Normal Case	With HYBRIDROID	Performance Overhead
CPU	165680	168980	1.95%
Memory	53005	53155	0.28%
I/O	110564	111584	0.91%
Total	45356	45628	0.59%

4.4 MEMORY CONSUMPTION

The HYBRIDROID client program's memory use is comparable to many well-known cell phone antivirus programs. Figure 7 shows the Memory required for every application on the Android device prior to any scanning and detection process. Table 8 shows the CPU and Memory required for every application respectively on the Android device prior to any scanning and detection process.

TABLE 8

Memory and CPU Consumption Results

Anti-Malware System	CPU Consumption	Memory Consumption
McAfee	11.5	46.4
AVG	12.26	58.6
360 Security	15.58	65.5
Quick Heal	12.77	62.4
HYBRIDROID	10.65	39.5

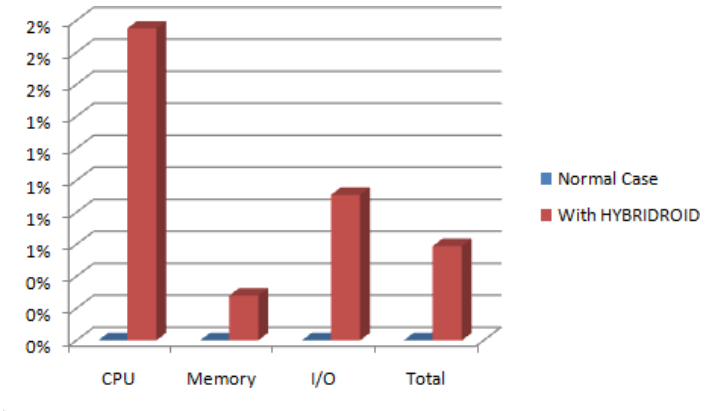

FIG 5: Chart-Performance Overhead of HYBRIDROID

4.5 POWER CONSUMPTION

Second, the malicious data set of Drebin has been used to build a classification model that does not contain any new variants in the malware category. In the future, we expect to extend HYBRIDROID's malicious portfolio so that HYBRIDROID can protect Android devices against new applications effectively. Results indicate that HYBRIDROID consumes moderate power as opposed to the other Anti-malwares. It comes in because HY- BRIDROID only tracks running programs. The other anti-malware applications search for all applications whether they are running background or not. Another explanation for our safety system's low power consumption is its distinctive feature that only user applications are evaluated and the devices are not monitored. Because computer manufacturers add system applications to the device, and are not malicious, HYBRIDROID does not scan for system applications for this purpose. On the other hand, other anti-malware checks all consumer and system applications on which they are using much power. HYBRIDROID also analyzes a dynamic system and performs static server analysis, while the other anti- malware listed in Table 8 only analyzes the user's run time behavior. Figure 6 Shows the CPU Consumption by each anti malware software after running it over the Android.

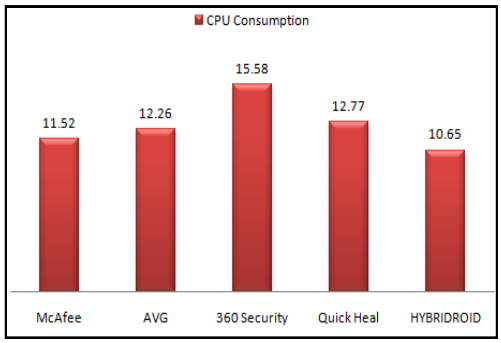

Fig 6 : CPU Consumption of Various Antimalware

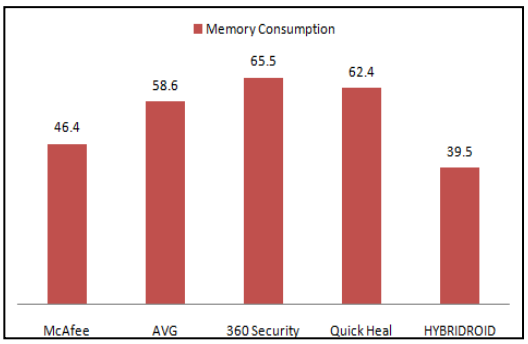

Fig 7 : Memory Consumption of Various Antimalware

5 DISCUSSION & FUTURE WORK

Although HYBRIDROID offers high detection precision and low resource consumption, there are some constraints. Next, the entire machine is connected to the server. Applications shall be classified on the server and the reports shall be submitted to Android for security purposes. In the local host no identification is made of malicious behavior i.e. A Google phone. So, if the network connection falls or the channel congestion is caused by the Android computer not being able to contact with the server, the performance of HYBRIDROID will be decreased Second, Drebin 's malicious data set was used to train a classification model that does not include the new malware forms variants. We plan to extend HYBRIDROID's malware portfolio to include new malware in the future so that HYBRIDROID can effectively defend Android devices from new applications.

6 CONCLUSION

This study focuses on developing a malware detection system that detects malware on Android devices while maintaining low use of resources. Many of the existing techniques for the detection and prevention of malware have been thoroughly examined during this research from 8 years from 2012 to 2020. Based on the advantages and disadvantages of current antimalware techniques, we asked whether current work is detecting Android malware in a precise manner and ensuring limited use of Android devices' hardware re- sources. Therefore, for the Android operating system, we introduced a 3 stage, hybrid malware detection platform. There is no hybrid 3-level malware detection method in our best knowledge. Therefore, HYBRIDROID is a modern tool for malware detection, combining the advantages of static analysis with dynamic analysis and artificial intelligence learning. It also runs on a local host and remote host to achieve the resource output. For Android devices, HYBRIDROID Client App is created. It analyzes and interacts with the dynamic system and with the server for documenting static analysis and detection. Re- mote server supports machine learning and static analysis-based recognition. We showed that HYBRIDROID is more effective than Drebin for static analyses through extensive analysis. Remember also that HYBRIDROID makes over-the-top production on Android very negligible. In terms of use of hardware resources HYBRIDROID is also successful.

7 REFERENCES

1. (2019). Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2019. Available: https://www.gartner.com/reviews/market/endpoint-protection platforms/vendor/malwarebytes
2. T. Bläsing, L. Batyuk, A.-D. Schmidt, S. A. Camtepe, and S. Albayrak, ''An Android Application Sandbox system for suspicious software detection,'' in Proc. 5th IEEE Int. Conf. Malicious Unwanted Software., Malware, Oct. 2010, pp. 55 62.
3. Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, ''Hey, you, get off of my market: Detecting malicious apps in of cial and alternative Android markets,'' in Proc. 19th Annu. Netw. Distrib. Syst. Secur. Symp., 2012.
4. Android Apps on Google Play. Accessed: Aug. 30, 2016. [Online]. Available: https://play.google.com/store?hl=en

5. DexClassLoader | Android Developers. Accessed: Aug. 30, 2016. [Online]. Available: https://developer.android.com/reference/dalvik/system/DexClassLoader.html
6. Update: Security Alert: DroidDreamLight, New Malware From the Developers of DroidDream | Lookout Blog. Accessed: Aug. 30, 2016. [Online]. Available: https://blog.lookout.com/blog/2011/05/30/security-alert-droiddreamlight-new-malware-from-the-developers-of-droiddream/
7. W.-C. Wu and S.-H. Hung, ‘‘DroidDolphin: A dynamic Android malware detection framework using big data and machine learning,’’ in Proc. Conf. Res. Adapt. Convergent Syst., Oct. 2014, pp. 247 252.
8. API Monitor: Spy on API Calls and COM Interfaces (Freeware 32-Bit and 64-Bit Versions!) | Rohitab.Com. Accessed: Aug. 22, 2016. [Online].
9. DroidBox.Accessed:Aug.22,2016.[Online].Available:https://github.com/pjlantz/droidbox
10. S. Chang, ‘‘APE: A smart automatic testing environment for Android malware,’’ Dept. Comput. Sci. Inf. Eng., Nat. Taiwan Univ., Taipei, Taiwan, Tech. Rep., 2013.
11. A. Ng, Support Vector Machines for Machine Learning. Stanford, CA, USA: Stanford Univ., 2008
12. LIBSVM A Library for Support Vector Machines. Accessed: Aug. 17, 2016. [Online]. Available: http://www.csie.ntu. edu.tw/~cjlin/libsvm/
13. X. Wang, Y. Yang, Y. Zeng, C. Tang, J. Shi, and K. Xu, ‘‘A novel hybrid mobile malware detection system integrating anomaly detection with mis-use detection,’’ in Proc. 6th Int. Workshop Mobile Cloud Comput. Services, Sep. 2015, pp. 15 22.
14. Android Aapt eLinux.Org. Accessed: Aug. 13, 2016. [Online]. Available: http://elinux.org/Android_aapt
15. CuckooDroid Book CuckooDroid V1.0 Book. Accessed: Aug. 19, 2016.
16. [Online]. Available: http://cuckoo-droid.readthedocs.io/en/latest/
17. 10 Open Source Mobile Test Automation Tools. Accessed: Aug. 19, 2016.
18. [Online]. Available: http://www.testingexcellence.com/open-source-mobile-test-automation-tools/
19. S. R. Gunn. (1998). University of Southampton Support Vector Machines for Classi cation and Regression. Accessed: May 16, 2017. [Online]. Available: http://ce.sharif.ir/courses/85-86/2/ce725/resources/root/LECTURES/SVM.pdf
20. K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu, ‘‘Improving one-class SVM for anomaly detection,’’ in Proc. Int. Conf. Mach. Learn., Nov. 2003, pp. 3077 3081.
21. P. V. Shijo and A. Salim, ‘‘Integrated static and dynamic analysis for mal-ware detection,’’ Procedia Comput. Sci., vol. 46, pp. 804 811, Jan. 2015.
22. Weka 3 Data Mining With Open Source Machine Learning Software in Java. Accessed: Dec. 16, 2015. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/
23. A. Andrew, N. Cristianini, and J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge, U.K.: Cambridge Univ. Press, 2000.
24. A. Andrew, ‘‘An introduction to support vector machines and other kernel-based learning methods,’’ in Kybernetes. Cambridge, U.K.: Cambridge Univ. Press, 2013.

25. D. Dittman, T. M. Khoshgoftaar, R. Wald, and A. Napolitano, ‘‘Random forest: A reliable tool for patient response prediction,’’ in Proc. BIBMW, Nov. 2011, pp. 289 296.
26. Z. Yuan, Y. Lu, and Y. Xue, ‘‘Droiddetector: Android malware characteri-zation and detection using deep learning,’’ Tsinghua Sci. Technol., vol. 21, no. 1, pp. 114 123, Feb. 2016.
27. TinyXML Download | SourceForge.Net. Accessed: Jun. 10, 2017. [Online].
28. 7-Zip. Accessed: Jun. 10, 2017. [Online]. Available: http://www.7-zip.org/
29. AXMLPrinter2 | Android Tales. Accessed: Jun. 10, 2017. [Online]. Available: http://android.amberfog.com/?tag=axmlprinter2
30. D. Arp, M. Spreitzenbarth, H. Malte, H. Gascon, and K. Rieck, ‘‘DREBIN: Effective and explainable detection of Android malware in your pocket,’’ in Proc. Symp. Netw. Distrib. Syst. Secur. (NDSS), 2014, pp. 23 26.
31. Update: Security Alert: DroidDreamLight, New Malware From the Developers of DroidDream | Lookout Blog. Accessed: Aug. 30, 2016. [Online]. Available: https://blog.lookout.com/blog/2011/05/30/security alert-droiddreamlight-new-malware-from-the-developers-of-droiddream/
32. https://virusshare.com/research
33. http://www.malgenomeproject.org/policy.html
34. https://play.google.com/store/apps/details?id=com.google.android.finsky&hl=en&gl=US

AUTHORS

First Author – **Rahul Y. Pawar**, Pune, India has completed M.Tech Computer from university of Pune. Currently Pursuing PhD from Veltech University, Chennai and Working as Assistant Professor, D Y Patil College of Engineering,Pune. He has awarded with “KrushiMauli Award 2020” Author holds two patents and published many articles and papers in reputed journals.

Second Author –Dr.C.Mahesh has more than 20 years of experience in the field of teaching. He was awarded B.E. in electrical & electronics engineering from Madras University, Chennai. He was awarded M.E. Computer science and engineering from Anna University, Chennai. He was awarded with a doctorate in computer science in the year 2016. Currently he is working as associate Professor in Veltech Institute of science and Technology, Chennai. His area of interest includes neural networks and natural language processing

Annexure I
List of All permission in Android Operating System

ACCESS_ALL_DOWNLOADS	CHANGE_WIFI_STATE	MASTER_CLEAR	SET_ACTIVITY_WATCHER
ACCESS_BLUETOOTH_SHARE	CHANGE_WIMAX_STATE	MEDIA_CONTENT_CONTROL	SET_ALWAYS_FINISH
ACCESS_CHECKIN_PROPERTIS	CONFIGURE_WIFI_DISPLAY	MODIFY_AUDIO_SETTINGS	SET_KEYBOARD_LAYOUT
ACCESS_DOWNLOAD_MAN	CONNECTIVITY_INTERNAL	MODIFY_CELL_BROADCAST	SET_ORIENTATION
ACCESS_FM_RADIO	CONTROL_LOCATION_UPDATES	MODIFY_NETWORK_ACCOUNTING	SET_POINTER_SPEED
ACCESS_MOCK_LOCATION	CONTROL_VPN	MODIFY_PARENTAL_CONTROLS	SET_PREFERRED_APPLICATIONS
ACCESS_MTP	CONTROL_WIFI_DISPLAY	MOUNT_FORMAT_FILESYSTEMS	SET_PROCESS_LIMIT
ACCESS_NETWORK_CONDITIONS	FACTORY_TEST	MOUNT_UNMOUNT_FILESYSTEMS	SET_SCREEN_COMPATIBILITY
ACCESS_NOTIFICATIONS	FLASHLIGHT	NOTIFY_PENDING_SYSTEM_UPDATE	SET_TIME
ACCESS_WIFI_STATE	FORCE_STOP_PACKAGES	PACKAGE_USAGE_STATS	SET_TIME_ZONE
ACCESS_WIMAX_STATE	FREEZE_SCREEN	PACKAGE_VERIFICATION_AGENT	SET_WALLPAPER
ACCOUNT_MANAGER	GET_ACCOUNTS_PRIVILEGED	PACKET_KEEPALIVE_OFFLOAD	SHUTDOWN
ASEC_ACCESS	GET_APP_GRANTED_URI_PERMISSIONS	PROCESS_CALLLOG_INFO	SIGNAL_PERSISTENT_PROCESSES
ASEC_CREATE	GET_DETAILED_TASKS	PROCESS_PHONE_ACCOUNT_REGISTRATION	START_ANY_ACTIVITY
ASEC_DESTROY	GET_PACKAGE_IMPORTANCE	PROVIDE_TRUST_AGENT	START_PRINT_SERVICE_CONFIG_ACTIVITY
BACKUP	GET_PACKAGE_SIZE	READ_BLOCKED_NUMBERS	STORAGE_INTERNAL
BATTERY_STATS	GET_TASKS	READ_DREAM_STATE	SUBSCRIBED_FEEDS_READ
BIND_CONNECTION_SERVICE	GET_TOP_ACTIVITY_INFO	READ_FRAME_BUFFER	SUBSCRIBED_FEEDS_WRITE
BIND_KEYGUARD_APPWIDGET	GLOBAL_SEARCH	READ_INPUT_STATE	SUBSTITUTE_NOTIFICATION_APP_NAME
BIND_MIDI_DEVICE_SERVICE	GRANT_RUNTIME_PERMISSIONS	READ_INSTALL_SESSIONS	SYSTEM_ALERT_WINDOW
BIND_NFC_SERVICE	HARDWARE_TEST	READ_PROFILE	TRANSMIT_IR
BIND_NOTIFICATION_LISTENER_SERVICE	INJECT_EVENTS	READ_SEARCH_INDEXABLES	TRUST_LISTENER
BIND_PRINT_SERVICE	INSTALL_PACKAGES	READ_SOCIAL_STREAM	UPDATE_CONFIG
BIND_REMOTEVIEWS	INTENT_FILTER_VERIFICATION_AGENT	READ_SYNC_SETTINGS	UPDATE_DEVICE_STATS
BIND_REMOTE_DISPLAY	INTERNAL_SYSTEM_WINDOW	READ_SYNC_STATS	UPDATE_LOCK
BIND_SCREENING_SERVICE	INTERNET	READ_USER_DICTIONARY	UPDATE_LOCK_TASK_PACKAGES
BIND_TELECOM_CONNECTION_SERVICE	KILL_UID	REAL_GET_TASKS	USER_ACTIVITY
BIND_TEXT_SERVICE	LAUNCH_TRUST_AGENT_SETTINGS	REBOOT	USE_CREDENTIALS
BIND_TV_REMOTE_SERVICE	LOCAL_MAC_ADDRESS	RECEIVE_BLUETOOTH_MAP	VIBRATE
BIND_VOICE_INTERACTION	KILL_UID	RECEIVE_BOOT_COMPLETED	WRITE_MEDIA_STORAGE
BIND_VPN_SERVICE	LAUNCH_TRUST_AGENT_SETTINGS	RECEIVE_DATA_ACTIVITY_CHANGE	WRITE_PROFILE
BLUETOOTH	LOCAL_MAC_ADDRESS	REGISTER_CONNECTION_MANAGER	WRITE_SECURE_SETTINGS
BLUETOOTH_STACK	MANAGE_ACCOUNTS	REGISTER_SIM_SUBSCRIPTION	WRITE_SETTINGS
BROADCAST_CALLLOG_INFO	MANAGE_ACTIVITY_STACKS	REGISTER_WINDOW_MANAGER_LISTENERS	WRITE_SMS
BROADCAST_NETWORK_PRIVILEGED	MANAGE_APP_OPS_RESTRICTIONS	REMOVE_TASKS	
BROADCAST_PACKAGE_REMOVED	MANAGE_DEVICE_ADMINS	REORDER_TASKS	
BROADCAST_PHONE_ACCOUNT_REGISTRATION	MANAGE_DOCUMENTS	REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	
BROADCAST_SMS	MANAGE_NETWORK_POLICY	REQUEST_INSTALL_PACKAGES	
CAMERA_SEND_SYSTEM_EVENTS	MANAGE_NOTIFICATIONS	REVOKE_RUNTIME_PERMISSIONS	
CAPTURE_AUDIO_OUTPUT	MANAGE_USERS	SCORE_NETWORKS	
CHANGE_CONFIGURATION	MANAGE_SOUND_TRIGGER	SEND_CALL_LOG_CHANGE	
CHANGE_NETWORK_STATE	MANAGE_USB	SEND_DOWNLOAD_COMPLETED_INTENTS	

Annexure II
List of APK used in our experiment analysis

Sl. No	Name of APK	Positive / Negative	Version &Year	Devices Used
1	Bhratgas	Negative	2.2.3 (2019)	Sony Xperia , Samsung Galaxy , Ras Berry Pie Kit,

2	Call log monitor	Negative	3.1.1 (2018)	Sony Xperia , Ras Berry Pie Kit,
3	Chrome	Negative	77.0.3865.92 (2018)	SDK Simulator, Ras Berry Pie Kit, Samsung Galaxy
4	Crick buzz	Negative	4.5.015 (2018)	Ras Berry Pie Kit, Sony Xperia
5	ES File Explorer	Negative	4.2.0.3.4 (2019)	Samsung Galaxy, SDK Simulator, Ras Berry Pie Kit,
6	Google Pay	Negative	43.0.001 (2019)	Samsung Galaxy , SDK Simulator, Ras Berry Pie Kit,
7	IRCTC Rail Connect	Negative	2.1.54 (2018)	Samsung Galaxy , SDK Simulator, Ras Berry Pie Kit,
8	Phone Pay	Negative	3.4.03 (2019)	Samsung Galaxy , SDK Simulator, Ras Berry Pie Kit,
9	Photos	Negative	4.24.1.268654418 (2018)	Samsung Galaxy , Ras Berry Pie Kit, Sony Xperia
10	Step Set Go	Negative	0.8.0 (2019)	Sony Xperia , Ras Berry Pie Kit, Samsung Galaxy
11	Truecaller	Negative	10.48.10 (2018)	Sony Xperia , Ras Berry Pie Kit, Samsung Galaxy
12	TryIT	Positive	1.0 (2019)	Samsung Galaxy , SDK Simulator, Sony Xperia
13	UTS	Negative	14.0.17 (2019)	Samsung Galaxy , Sony Xperia
14	Whatsapp	Negative	2.19.258 (2019)	SDK Simulator, Ras Berry Pie Kit, Sony Xperia
15	Where is my Train	Negative	6.1.8 (2018)	SDK Simulator, Ras Berry Pie Kit, Samsung Galaxy
16	XECH FIT	Positive	1.0.5 (2019)	SDK Simulator, Ras Berry Pie Kit, Sony Xperia
17	JS.Miner.11	Positive	2.1.1 (2018)	SDK Simulator, Ras Berry Pie Kit, Sony Xperia, Samsung Galaxy
18	Trojan.Script.Generic 1	Positive	1.0.0 (2018)	Sony Xperia, SDK Simulator, Samsung Galaxy
19	Coinminer_COINHIV E.SMF1-JS	Positive	2.0.7 (2018)	Samsung Galaxy, Ras Berry Pie Kit, Sony Xperia
20	Application.CoinMine r.AY	Positive	3.1.0 (2019)	Samsung Galaxy , Ras Berry Pie Kit, Sony Xperia
21	Calculator	Negative	7.1.2 (2019)	Samsung Galaxy , Ras Berry Pie Kit, Sony Xperia
22	Daily life	Negative	3.3.1 (2018)	SDK Simulator, Ras Berry Pie Kit, Sony Xperia
23	Files	Negative	7.1.2 (2019)	SDK Simulator, Ras Berry Pie Kit, Sony Xperia, Samsung Galaxy
24	Flashlight	Negative	8.10.5 (2019)	SDK Simulator, Ras Berry Pie Kit, Sony Xperia, Samsung Galaxy
25	FM Radio	Negative	7.1.2 (2019)	SDK Simulator, Ras Berry Pie Kit, Sony Xperia, Samsung Galaxy
26	Launcher3	Positive	7.1.2 (2019)	SDK Simulator, Sony Xperia, Samsung Galaxy
27	Slides	Negative	1.19.352.05 (2019)	Samsung Galaxy, SDK Simulator, Sony Xperia
28	You tube	Negative	14.37.54 (2019)	Samsung Galaxy , SDK Simulator, Sony Xperia