



ISSN: 2321-2152

IJMECE

*International Journal of modern
electronics and communication engineering*

E-Mail

editor.ijmece@gmail.com

editor@ijmece.com

www.ijmece.com

NOVEL MACHINE LEARNING METHOD FOR ANDROID MALWARE DETECTION USING CO-EXISTING FEATURES

G VISWANATH¹, T V NIRMALA², A DHANASEKHAR REDDY³

¹Associate Professor, Department of CSE(AI ML), Sri Venkatesa Perumal College of Engineering & Technology, Puttur, Email: viswag111@gmail.com, ORCID: <https://orcid.org/0009-0001-7822-4739>

²P.G Scholar, Department of MCA, Sri Venkatesa Perumal College of Engineering & Technology, Puttur, Email: nirmalatv05@gmail.com

³Assistant Professor, Department of MCA, Sri Venkatesa Perumal College of Engineering & Technology, Puttur, Email: ghanasekhar918@gmail.com

Abstract: The venture breaks down Android malware detection using Drebin, Malgenome, and CIC_MALDROID2020. Rich Programming interface and Authorization information from these data sets permits full examination. Logistic Regression, Support Vector Machine, K-Nearest Neighbors (KNN), Random Forest, Decision Tree, and a Stacking Classifier that incorporates Random Forest, MLP, and LightGBM have been utilized for classification. The undertaking requires broad dataset planning, model training, and execution assessment. This complete technique looks to make strong Android malware detection models. Effective model results work on versatile security by further developing Android malware discovery and relief. Portable security specialists and experts benefit from the drive. To further develop highlight extraction, we made a Stacking Classifier that joins Random Forest, Multi-Layer Perceptron (MLP), and LightGBM models. Ensemble learning significantly increments prediction accuracy. We have made a simple to-utilize Flask framework with SQLite for secure information exchange, signin, and testing. This improved on strategy works with info and forecast recovery, making the undertaking stronger and easy to understand.[42]

Index terms - co-existence, FP-growth, machine learning, malware.

1. INTRODUCTION

Cell phone deals are taking off. The ICD study [1] predicts that cell phone deals will surpass 351 million units by 2024. Android is the most well known portable working framework, with 2.5 billion clients in 190 nations [2]. Cell phones' huge assortment of abilities and expanded use of online entertainment, web banking, and gaming have raised significant stresses over gadget security and protection. Since Android is open-source, malware designers may effortlessly send off assaults and produce destructive Android malware applications. In current civilization, Android malware is turning out to be more huge [3], [4].

The Google Play Store contains over 3.43 million applications as of January 2021 [5]. New application commercial centers like AppBrain and AppChina permit purchasers to download applications. Outsider market applications are possible perilous and inconspicuous. Allix et al. [6] saw as 22% of Google Play applications and half of AppChina applications hurtful.

Numerous safeguards have been executed to stop maverick applications. Signature-based malware detection is an early technique that looks at presented projects to new ones. Despite the fact that it turns out really for known malware, obscurity strategies and new malware just keep away from it. Zhou and Jiang [7] inspected four portable security applications against more than 1200 Android applications to survey signature-based enemy of malware scanners. Muddled or repackaged malware programs are imperceptible by current enemy of malware programming. Scott [8] jumbled eleven noxious applications from unmistakable families. Nobody could perceive rebel applications after obscurity.

ML is utilized to recognize Android malware and recognize it from harmless ones without contrasting examples. Intelligent machine learning approaches use "training data" to foster a model that makes predictions or decisions without being expressly modified. Utilizing ML strategies, malware detection might be static or dynamic. In static examination, an Android application is assessed without running. Conversely, dynamic examination breaks down application conduct in a controlled climate.

The manifest (application consents), source code (Programming interface calls), and purposes are utilized in static examination [19,21]. A few recovered characteristics might be problematic. Malware and harmless projects look for various authorizations. Different feature selection techniques are utilized to distinguish the most pertinent highlights for malware application classification. A predefined feature selection strategy scores the connection or reliance among info and result or class factors utilizing measurable techniques. Most calculations rank every

trademark independently. All things considered, this study detects Android malware through existing together properties.[44]

2. LITERATURE SURVEY

A great many people right now convey a web associated PC [4]. The web is influencing our regular routines. Notwithstanding PCs and cell phones, detached contraptions are presently web associated with become brilliant. SCADA in urban communities, emergency clinics, and different organizations is associated with the web to make it more astute. Development of web makes life simpler.

Specialists are investigating ML calculations for malware location in immense applications. Numerous strategies have shown guarantee in the writing, and many are being tried in the lab. [6]This study returns to the objective of malware location to check whether lab approval circumstances precisely reflect malware finder execution in the field. We made various AI classifiers utilizing CFG highlights to do this. More than 50 000 Android applications from best in class sources are utilized in our assortment. Our method beats AI based approaches in the lab. This elite presentation doesn't switch over completely to ludicrous execution. A key inquiry emerges from the exhibition difference we noticed — F-measures plunging from over 0.9 in the lab to beneath 0.1 in nature. How do state of the art strategies work practically speaking?

Versatile malware, prominently on Android, has multiplied quickly because of advanced mobile phone use [10,15,18]. Because of its rising ascent, powerful arrangements are required. Our guarded capacities is

frustrated by our low comprehension of creating portable malware and the shortfall of convenient examples. [7] Our objective in this study is to arrange or depict Android malware. After nearly 12 months, we gathered north of 1,200 malware tests from August 2010 to October 2011 from most of Android malware families [7,13,38]. We additionally break down their establishment, enactment, and unsafe payloads. The characterisation and advancement based examination of agent families show that they are changing quick to stay away from versatile enemy of infection programming discovery. We tried four example versatile security devices and found that the best case recognizes 79.6% of them and the most pessimistic scenario 20.2%. These discoveries propose further developing cutting edge enemy of portable malware frameworks.[46]

As billions of cell phone clients consistently save individual and delicate information on their gadgets, programmers have a wide stage to take this information [10]. We propose utilizing consents and Programming interface to distinguish android malware. We made normal and incorporated feature vectors. Logistic regression yielded 97.25% exactness for normal and 96.56% for blended attributes. To limit arrangement preparing and testing time, we refined the features to 131 by erasing low change includes and acquired 95.87% accuracy.

Pernicious applications compromise Android security. The extending number and assortment of these applications makes ordinary security futile, leaving Android handsets powerless against new contaminations. [13] This article presents DREBIN, a lightweight Android malware detection approach that recognizes risky applications straightforwardly on the

cell phone. Since limited assets forestall run-time program checking, DREBIN plays out an exhaustive static examination to catch whatever number qualities as could be expected under the circumstances. Since these qualities are coordinated in a common vector space, normal malware examples might be naturally found and used to make sense of our technique's decisions. DREBIN beats different past calculations and distinguishes 94% of malware with negligible misleading problems in a test utilizing 123,453 applications and 5,560 malware tests [7,13,38]. The clarifications feature key malware highlights. Normal examination time on five normal cell phones is 10 seconds, making it OK for confirming downloaded applications right on the gadget.

3. METHODOLOGY

i) Proposed Work:

A particular ML model for Android malware detection shows how coinciding consents and APIs separate malware from harmless applications. It beats Drebin, CIC_MALDROID2020, and Malgenome models in accuracy [9,13,25]. It intends to further develop Android security and privacy. The Stacking Classifier in this review utilizes Random Forest, Multi-Layer Perceptron (MLP), and LightGBM models to further develop feature extraction. Ensemble advancing incredibly increments prediction accuracy. We have made a simple to-utilize Flask framework with SQLite for secure information exchange, signin, and testing. This improved on technique works with info and expectation recovery, making the undertaking stronger and easy to understand.

ii) System Architecture:

The framework configuration starts with a dataset of Android applications with properties created through conjunction combinations[4,5,7]. This dataset is then parted into training and test sets. AI models (knn, svm, rf, dt, lr, and augmentation stacking classifier) begin with the preparation set. These calculations figure out how to recognize unsafe and harmless applications. From that point onward, the test set is utilized to assess these models' capacity to sort new applications.

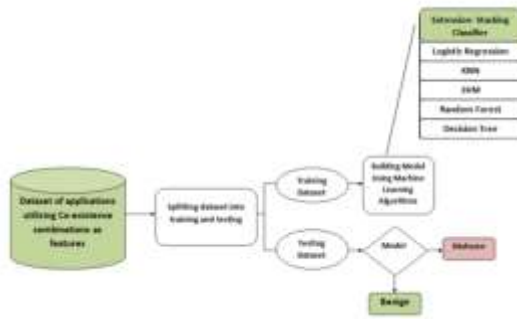


Fig 1 Proposed architecture

iii) Dataset collection:

DREBIN

- Drebin is a well known Android malware collection. It has a few Android applications, both great and terrible. To make compelling models for Android malware detection, its scale and assortment make it a famous benchmark dataset [9,13,25].
- We used the Drebin dataset with these feature mixes.
- Top 5 lines of information with each feature mix are displayed underneath. No. segments are noticeable.




Fig 2 Drebin dataset

MALGENOME

- Android applications in Malgenome target malware tests. Numerous Android malware models are incorporated. It gives Android malware tests to research and show working to help Drebin.
- We utilized the Malgenome dataset with a few feature combinations [917,36]. We give the main 5 lines of data so that each feature combination could see how much sections.



Fig 3 Malgenome

CIC_MALDROID2020

- The Canadian Institute for Cybersecurity (CIC) gives CIC_MALDROID2020, which is enormous, later, different, and careful.
- CIC_MALDROID2020 dataset containing these feature combinations was used.
- Top 5 columns of data with each feature combination are displayed beneath. No. segments are noticeable.



The image shows a screenshot of the CIC_MALDROID2020 dataset. It displays three different feature combinations for the dataset: 'API+Permission', 'Only API', and 'Only permission'. Each combination shows a list of features with their corresponding values. The 'API+Permission' combination includes features like 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', and 'android.permission.INTERNET'. The 'Only API' combination includes features like 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', and 'android.permission.INTERNET'. The 'Only permission' combination includes features like 'android.permission.ACCESS_NETWORK_STATE', 'android.permission.ACCESS_WIFI_STATE', and 'android.permission.INTERNET'.

Fig 4 CIC_MALDROID2020

iv) Data Processing:

Data processing transforms raw information into business-helpful data. Information researchers accumulate, sort out, clean, check, break down, and orchestrate information into diagrams or papers. Data can be handled physically, precisely, or electronically. Data ought to be more significant and decision-production simpler. Organizations might upgrade activities and settle on basic decisions quicker. PC programming improvement and other mechanized information handling innovations add to this. Big data can be transformed into significant bits of knowledge for quality administration and independent direction.

v) Feature selection:

Feature selection chooses the most steady, non-repetitive, and pertinent elements for model turn of events. As data sets extend in amount and assortment, purposefully bringing down their size is significant. The fundamental reason for feature selection is to increment prescient model execution and limit processing cost.

One of the vital pieces of feature engineering is picking the main attributes for machine learning algorithms. To diminish input factors, feature selection methodologies take out copy or superfluous elements and limit the assortment to those generally critical to the ML model. Rather than permitting the ML model pick the main qualities, feature selection ahead of time enjoys a few benefits.[48]

vi) Algorithms:

Logistic Regression is a classification technique that predicts input classification. It utilizes the sigmoid capability to move input qualities to a likelihood score somewhere in the range of 0 and 1, then, at that point, applies an edge to sort the contribution to at least two classifications. The model learns coefficients during preparing to fit information and group precisely.

```
from sklearn.linear_model import LogisticRegression
#from sklearn.pipeline import Pipeline

# instantiate the model
log = LogisticRegression()

# fit the model
log.fit(X_train,y_train)
y_pred = log.predict(X_test)
```

Fig 5 Logistic regression

A **Support Vector Classifier (SVC)** is a ML model that recognizes the ideal hyperplane to divide

information classes while expanding edge. The principal support vectors it sees as empower accurate binary and multi-class groupings.

```
# Support Vector Classifier model
from sklearn.svm import SVC
svc = SVC()

# fit the model
svc.fit(X_train,y_train)
y_pred = svc.predict(X_test)
```

Fig 6 SVM

K-Nearest Neighbors (KNN) is an adaptable classification and regression strategy. It predicts by means of larger part voting or normal from the K nearest data focuses to an input. Non-parametric and simple to build, KNN is delicate to K and may not perform well in high-layered information without preprocessing [30].

```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)

# fit the model
neigh.fit(X_train,y_train)
y_pred = neigh.predict(X_test)
```

Fig 7 KNN

Random Forest is an ensemble learning approach that predicts utilizing a few decision trees. Training decision trees on arbitrary information subsets and averaging their expectations works. This ensemble strategy further develops classification and regression accuracy, wipes out overfitting, and performs well.

```
# Random Forest Classifier Model
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
forest = RandomForestClassifier(n_estimators=10)

# fit the model
forest.fit(X_train,y_train)
y_pred = forest.predict(X_test)
```

Fig 8 Random forest

A **Decision Tree** is a ML model that characterizes or predicts results by recursively isolating information into subgroups relying upon the main trait. It constructs a tree-like design with hubs addressing features and branches addressing elective choices, making it interpretable and gainful for different errands.

```
# Decision Tree Classifier model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth=30)

# fit the model
tree.fit(X_train, y_train)

y_pred = tree.predict(X_test)
```

Fig 9 Decision tree

A **Stacking Classifier** is a ensemble learning approach that utilizes many base models like Random Forest (RF), Multi-Layer Perceptron (MLP), and LightGBM to further develop prediction accuracy. It further develops forecast execution by joining these models' assets. Base models are prepared on the training data, and their predictions are used as info features for a meta-student, which figures out how to consolidate them to create the last prediction. Stacking

works on conjecture exactness and is used in many ML applications.[50]

```
estimators = [{"X": RandomForestClassifier(), "y": LRClassifier(y_train_state), "X_test": LRClassifier()}]
stack = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression())
# fit the model
stack.fit(X_train, y_train)
# predict the model
y_pred = stack.predict(X_test)
```

Fig 10 Stacking classifier

4. EXPERIMENTAL RESULTS

Precision: Precision estimates the level of positive cases or tests precisely sorted. Precision is determined utilizing the recipe:

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} = \frac{TP}{TP + FP}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

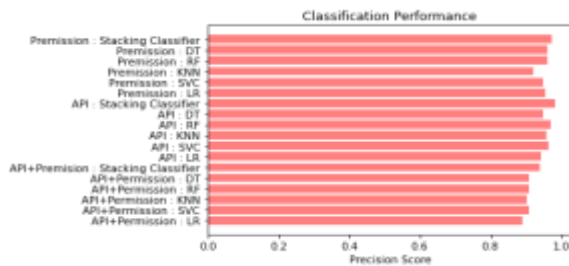


Fig 11 Precision comparison graph

Recall: Machine learning recall assesses a model's ability to perceive all significant examples of a class. It shows a model's culmination in catching occasions of a class by contrasting accurately anticipated positive perceptions with complete positives.

$$\text{Recall} = \frac{TP}{TP + FN}$$

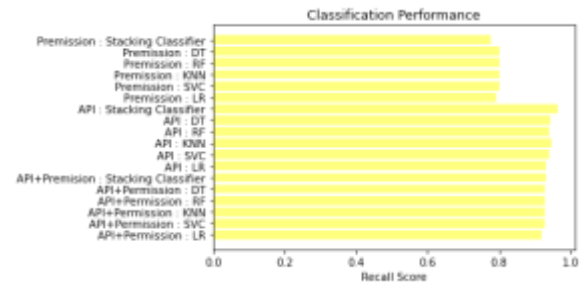


Fig 12 Recall comparison graph

Accuracy: A test's accuracy is its ability to recognize debilitated from sound cases. To quantify test accuracy, figure the small part of true positive and true negative in completely broke down cases. Numerically, this is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

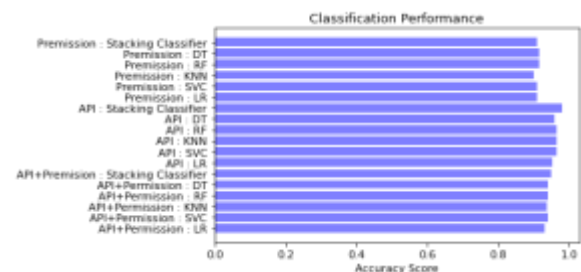


Fig 13 Accuracy graph

F1 Score: Machine learning model accuracy is estimated by F1 score. Consolidating model precision and recall scores. The accuracy measurement estimates how frequently a model anticipated accurately all through the dataset.

$$F1 \text{ Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$

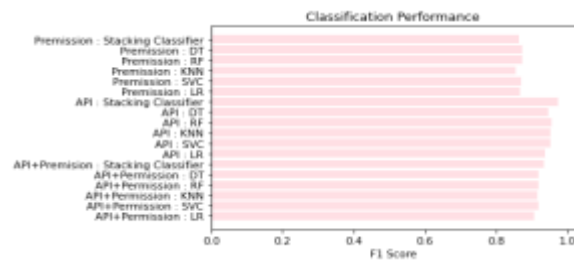


Fig 14 F1Score

	Model	Accuracy	Precision	Recall	F1Score
1	API+Permission - LR	0.92	0.92	0.92	0.92
2	API+Permission - SVC	0.92	0.92	0.92	0.92
3	API+Permission - KNN	0.92	0.92	0.92	0.92
4	API+Permission - RF	0.92	0.92	0.92	0.92
5	API+Permission - DT	0.92	0.92	0.92	0.92
6	API+Permission - Stacking Classifier	0.95	0.95	0.95	0.95
7	API - LR	0.90	0.90	0.90	0.90
8	API - SVC	0.90	0.90	0.90	0.90
9	API - KNN	0.90	0.90	0.90	0.90
10	API - RF	0.90	0.90	0.90	0.90
11	API - DT	0.90	0.90	0.90	0.90
12	API - Stacking Classifier	0.92	0.92	0.92	0.92
13	Permission - LR	0.88	0.88	0.88	0.88
14	Permission - SVC	0.88	0.88	0.88	0.88
15	Permission - KNN	0.88	0.88	0.88	0.88
16	Permission - RF	0.88	0.88	0.88	0.88
17	Permission - DT	0.88	0.88	0.88	0.88
18	Permission - Stacking Classifier	0.90	0.90	0.90	0.90

Fig 15 Performance Evaluation

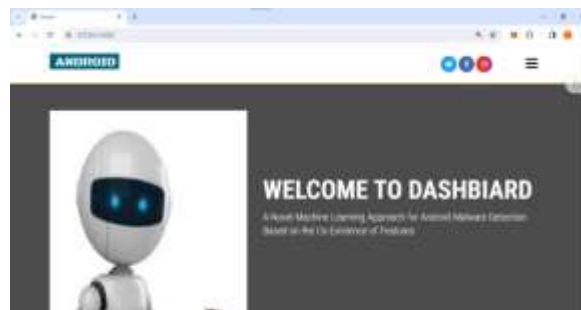


Fig 16 Home page

+SignIn

Already have an account? [Sign in](#)

Fig 17 Signin page

+SignIn

Register here! [Sign Up](#)

Fig 18 Login page

KILL_BACKGROUND_PROCESSES:

CHANGE_NETWORK_STATE:

transact:

Ljava.lang.Class.getCanonicalName:

Ljava.lang.Class.getMethods:

Landroid.content.Context.registerReceiver:

getBinder:

createSubprocess:

Fig 19 User input



Fig 20 Predict result for given input

5. CONCLUSION

The investigation demonstrated ML models can detect Android malware. These models were viable at detecting hazardous applications, protecting Android users. The stacking classifier beat individual calculations. Consolidating many models further developed detection accuracy, it learning's capability to ensemble learning's potential. Flask with SQLite's easy to understand interface increments openness. The

plan further develops convenience and acknowledgment by means of client testing, input approval, and consistent model predictions. The venture focused on the need of Programming interface Consent joining. Multistatic attributes are vital to malware detection, and this blend further developed identification. AI models performed contrastingly on Drebin, Malgenome, and CIC_MALDROID2020 [36]. Creating fruitful location calculations requires careful dataset choice and information. The models adjusted precision and misleading up-sides. This keeps typical projects from being distinguished as dangers while recognizing malware, diminishing client inconvenience. This drive makes colossal impacts. Security specialists might support network safety with these better recognition strategies. Designers can all the more likely safeguard their applications, and buyers can get further developed security against Android malware, making versatile more secure.

6. FUTURE SCOPE

Further review could screen and examine dynamic qualities to further develop the proposed framework's constant detecting abilities. This would further develop framework reaction to changing Android malware dangers. Investigation to pick the main unique attributes for malware detection can work on model effectiveness and accuracy. Conceivable feature selection techniques incorporate common data and recursive element expulsion. Growing the framework to recognize Android application conduct anomalies helps support security. Recognizing takeoffs from regular way of behaving may show malware. [7] New Android malware may require the framework to overhaul its models and detection strategies. Long haul adequacy requires standard

danger knowledge updates and framework refreshes. A more complete portable security arrangement can incorporate cross-stage malware detection for iOS and other versatile working frameworks.

REFERENCES

- [1] H. Menear. (2021). IDC Predicts Used Smartphone Market Will Grow 11.2% by 2024. Accessed: Oct. 30, 2022. [Online]. Available: <https://mobile-magazine.com/mobile-operators/idc-predicts-usedsmartphone-market-will-grow-112-2024?page=1>
- [2] D. Curry. (2022). Android Statistics. Accessed: Oct. 30, 2022. [Online]. Available: <https://www.businessofapps.com/data/android-statistics/>
- [3] O. Abendan. (2011). Fake Apps Affect Android Os Users. Accessed: Oct. 30, 2022. [Online]. Available: <https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/72/fake-apps-affect-android-osusers>
- [4] C. D. Vijayanand and K. S. Arunlal, “Impact of malware in modern society,” J. Sci. Res. Develop., vol. 2, pp. 593–600, Jun. 2019.
- [5] M. Iqbal. (2022). App Download Data. Accessed: Oct. 30, 2022. [Online]. Available: <https://www.businessofapps.com/data/app-statistics/>
- [6] K. Allix, T. Bissyand, Q. Jarome, J. Klein, R. State, and Y. L. Traon, “Empirical assessment of machine learning-based malware detectors for android,” Empirical Softw. Eng., vol. 21, pp. 183–211, Jun. 2016.
- [7] Y. Zhou and X. Jiang, “Dissecting Android malware: Characterization and evolution,” in Proc. IEEE Symp. Secur. Privacy, May 2012, pp. 95–109.
- [8] J. Scott. (2017). Signature Based Malware Detection is Dead. Accessed: Oct. 30, 2022. [Online]. Available: <https://icitech.org/wpcontent/uploads/2017/02/ICIT-Analysis-Signature-Based-MalwareDetection-is-Dead.pdf>
- [9] Q. M. Y. E. Odat. Accessed: Dec. 27, 2022. [Online]. Available: <https://github.com/esraa-cell28/a-novel-machine-learning-approach-forandroid-malware-detection-based-on-the-co-existence>
- [10] S. R. Tiwari and R. U. Shukla, “An Android malware detection technique based on optimized permissions and API,” in Proc. Int. Conf. Inventive Res. Comput. Appl. (ICIRCA), Jul. 2018, pp. 258–263.
- [11] (2018). Dex2jar—Tools To Work With Android.dex & Java.Class Files. Accessed: Oct. 30, 2022. [Online]. Available: <https://kalilinuxtutorials.com/dex2jar-android-java/>
- [12] Androzoo. Accessed: Jul. 30, 2022. [Online]. Available: <https://androzoo.uni.lu/>
- [13] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, “Drebin: Effective and explainable detection of Android malware in your pocket,” in Proc. NDSS, Feb. 2014, pp. 23–26.
- [14] Virusshare. accessed: Jul. 30, 2022. [Online]. Available: <https://virusshare.com/>

- [15] H. Cheng, X. Yan, J. Han, and C.-W. Hsu, “Discriminative frequent pattern analysis for effective classification,” in Proc. IEEE 23rd Int. Conf. Data Eng., Apr. 2007, pp. 716–725.
- [16] M. Parkour. Contagio Mini-Dump. accessed: Jul. 30, 2022. [Online]. Available: <http://contagiomindump.blogspot.it/>
- [17] Malgenome Project. accessed: Jul. 30, 2022. [Online]. Available: <http://www.Malgenomeproject.org>
- [18] C.-F. Tsai, Y.-C. Lin, and C.-P. Chen, “A new fast algorithms for mining association rules in large databases,” in Proc. IEEE Int. Conf. Syst., Man Cybern. San Francisco, CA, USA: Morgan Kaufmann, Oct. 1994, pp. 487–499.
- [19] A. Lab. (2017). Amd Dataset. Accessed: Oct. 30, 2022. [Online]. Available: <https://www.kaggle.com/datasets/blackarcher/malware-dataset>
- [20] V. Avdiienko, “Mining apps for abnormal usage of sensitive data,” in Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng., vol. 1, May 2015, pp. 426–436.
- [21] Y. Aafer, W. Du, and H. Yin, “Droidapiminer: Mining api-level features for robust malware detection in android,” in Security and Privacy in Communication Networks, T. Zia, A. Zomaya, V. Varadharajan, and M. Mao, Eds. Cham, Switzerland: Springer, 2013, pp. 86–103.
- [22] V. M. Afonso, M. F. de Amorim, A. R. A. Grégio, G. B. Junquera, and P. L. De Geus, “Identifying Android malware using dynamically obtained features,” J. Comput. Virology Hacking Techn., vol. 11, no. 1, pp. 9–17, 2015.
- [23] S. K. Dash, G. Suarez-Tangil, S. Khan, K. Tam, M. Ahmadi, J. Kinder, and L. Cavallaro, “DroidScribe: Classifying Android malware based on runtime behavior,” in Proc. IEEE Secur. Privacy Workshops (SPW), May 2016, pp. 252–261.
- [24] H. Cai, N. Meng, B. G. Ryder, and D. Yao, “DroidCat: Effective Android malware detection and categorization via app-level profiling,” IEEE Trans. Inf. Forensics Security, vol. 14, no. 6, pp. 1455–1470, Jun. 2019.
- [25] A. F. A. kadir, N. Stakhanova, and A. Ghorbani, “An empirical analysis of Android banking malware,” Tech. Rep., Nov. 2016.
- [26] M. Sun, M. Zheng, J. C. S. Lui, and X. Jiang, “Design and implementation of an Android host-based intrusion prevention system,” in Proc. 30th Annu. Comput. Secur. Appl. Conf. New York, NY, USA: Association for Computing Machinery, 2014, pp. 226–235, doi: 10.1145/2664243.2664245.
- [27] Google Play Store. Accessed: Jul. 30, 2022. [Online]. Available: <https://play.google.com/store/games>
- [28] D. Son. (2019). Apktool—Tool For Reverse Engineering Android Apk Files. Accessed: Oct. 30, 2022. [Online]. Available: <https://securityonline.info/apktool-reverse-engineering-android-apkfiles/>
- [29] McAfee. Accessed: Jul. 30, 2022. [Online]. Available: <https://www.mcafee.com/>

- [30] G. Baldini and D. Geneiatakis, "A performance evaluation on distance measures in KNN for mobile malware detection," in Proc. 6th Int. Conf. Control, Decis. Inf. Technol. (CoDIT), Apr. 2019, pp. 193–198.
- [31] Y. Zhang, Y. Yang, and X. Wang, "A novel Android malware detection approach based on convolutional neural network," in Proc. 2nd Int. Conf. Cryptogr., Secur. Privacy, Mar. 2018, pp. 144–149.
- [32] M. Amin, T. A. Tanveer, M. Tehseen, M. Khan, F. A. Khan, and S. Anwar, "Static malware detection and attribution in Android byte-code through an end-to-end deep system," *Future Gener. Comput. Syst.*, vol. 102, pp. 112–126, Jan. 2020, doi: 10.1016/j.future.2019.07.070.
- [33] M. Lindorfer, M. Neugschwandtner, and C. Platzer, "MARVIN: Efficient and comprehensive mobile app classification through static and dynamic analysis," in Proc. COMPSAC, vol. 2, Jul. 2015, pp. 422–433.
- [34] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and efficient behavior-based Android malware detection and prevention," *IEEE Trans. Depend. Sec. Comput.*, vol. 15, no. 1, pp. 83–97, Jan./Feb. 2018.
- [35] S. Chen, M. Xue, Z. Tang, L. Xu, and H. Zhu, "StormDroid: A streamglized machine learning-based system for detecting Android malware," in Proc. 11th ACM Asia Conf. Comput. Commun. Secur. York, NY, USA: Association for Computing Machinery, May 2016, pp. 377–388, doi: 10.1145/2897845.2897860.
- [36] (2020). CIC_MALDROID2020 Dataset, Canadian Institute for Cybersecurity. Accessed: Oct. 30, 2022. [Online]. Available: <https://www.unb.ca/cic/datasets/malroid-2020.html>
- [37] S. Y. Yerima and S. Sezer, "DroidFusion: A novel multilevel classifier fusion approach for Android malware detection," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 453–466, Feb. 2019.
- [38] H. L. Thanh, "Analysis of malware families on Android mobiles: Detection characteristics recognizable by ordinary phone users and how to fix it," *J. Inf. Secur.*, vol. 4, no. 4, pp. 213–224, 2013.
- [39] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for Android malware detection using deep learning," *Digit. Invest.*, vol. 24, pp. 48–59, Mar. 2018.
- [40] Z. Aung and W. Zaw, "Permission-based Android malware detection," *Int. J. Sci. Technol. Res.*, vol. 2, no. 3, pp. 228–234, 2013.
- [41] G. Viswanath, "Hybrid encryption framework for securing big data storage in multi-cloud environment", *Evolutionary intelligence*, vol.14, 2021, pp.691-698.
- [42] Viswanath Gudditi, "Adaptive Light Weight Encryption Algorithm for Securing Multi-Cloud Storage", *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol.12, 2021, pp.545-552.
- [43] Viswanath Gudditi, "A Smart Recommendation System for Medicine using Intelligent NLP Techniques", 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), 2022, pp.1081-1084.

[44] G.Viswanath, “Enhancing power unbiased cooperative media access control protocol in manets”, International Journal of Engineering Inventions, 2014, vol.4, pp.8-12.

[45] Viswanath G, “A Hybrid Particle Swarm Optimization and C4.5 for Network Intrusion Detection and Prevention System”, 2024, International Journal of Computing, DOI: <https://doi.org/10.47839/ijc.23.1.3442>, vol.23, 2024, pp.109-115.

[46] G.Viswanath, “A Real Time online Food Ordering application based DJANGO Restfull Framework”, Juni Khyat, vol.13, 2023, pp.154-162.

[47] Gudditi Viswanath, “Distributed Utility-Based Energy Efficient Cooperative Medium Access Control in MANETS”, 2014, International Journal of Engineering Inventions, vol.4, pp.08-12.

[48] G.Viswanath,“ A Real-Time Video Based Vehicle Classification, Detection And Counting System”, 2023, Industrial Engineering Journal, vol.52, pp.474-480.

[49] G.Viswanath, “A Real- Time Case Scenario Based On Url Phishing Detection Through Login Urls ”, 2023, Material Science Technology, vol.22, pp.103-108.

[50] Manmohan Singh,Susheel Kumar Tiwari, G. Swapna, Kirti Verma, Vikas Prasad, Vinod Patidar, Dharmendra Sharma and Hemant Mewada, “A Drug-Target Interaction Prediction Based on Supervised Probabilistic Classification” published in Journal of Computer Science, Available at: <https://pdfs.semanticscholar.org/69ac/f07f2e756b79181e4f1e75f9e0f275a56b8e.pdf>