IMPLIMENTATION OF MACHINE LEARNING TO DETECT ANDROID MALWARE WITH SIGNIFICANT PERMISSION

R. Murugadoss
Professor, Department Artificial Intelligence,
V.S.B. College of Engineering Technical Campus,
Coimbatore, Tamil Nadu, India .
Corresponding mail : drrmdcse@gmail.com

***Abstract*: Malicious app proliferation has reached alarming levels, posing a severe risk to the developing digital ecosystem. According to the most recent study, a new dangerous Android app is created every ten seconds. Researchers need a modular malware detection system that can quickly locate and identify the most important bad programmes in order to stop this massive strike. There are many ways to identify malware, including system- and network-based techniques. On the other side, it is still difficult to expand detection for many different programmes. Researchers developed SigPID, a malware-finding method based on authorization usage analysis, to combat the fast increase in Android infections. Instead of gathering and evaluating all Android rights, this study proposes three rounds of pruning by mining permission data to locate the most significant permissions that can be helpful in differentiating between benign and harmful apps. Then, SigPID categorises numerous families of benign and malicious applications using techniques based on machine learning. Only 21 permissions, according to our analysis, are essential. The effectiveness of our approach, which only considers 23 rights, is then compared to a baseline technique that considers all permits. The results show that employing an SVM as the classifier can lead to over 90% accuracy, recall, reliability, and F-measure, which are comparable to those discovered using the baseline technique, while having research that is 4–32 times less expensive than when using all rights. With a detection rate of 94 percent for viruses and 92 percent for unidentified malware specimens in the data, SigPID exceeds other cutting-edge methods.**

Keywords: **malware, Android, mining, SigPID, precision, F-measure**

I. INTRODUCTION

Like mobile transactions, SMS, Internet shopping, etc., all these people need phones, which have gradually become essential things in life, but the most frequent problem with these mobile phones is safety. The extremely untroubled and advantageous environment to write malicious code using a threat and safety fault of the Android system because of the open-resource nature of the Android operating system was the cause for the quickly increasing count of malicious code in the Android system. For sending detectable SMS, network traffic, theft of the user's data, downloading many malware apps, remote control, etc., including the malicious behaviour of Android malware, terrifying the personal rights and asset safety of mobile phone users. The number of malicious Android apps is rapidly increasing. In particular, malicious software frequently uses obscure technology. Have obtained many research solutions, and many researchers have settled the issues of malicious discernment using machine learning methods in recent years.

The approach for finding Android malware is one that researchers are constantly upgrading and improving. Static examination and dynamic examination are the two main categories of malware investigation innovation, and the detection method is derived from ordinary machine-profound learning calculations. After decompiling the APK record, the static examination technique suggests removing malicious components through semantic analysis, consent analysis, and other methods. High proficiency and quickness are the main advantages of static recognition; nonetheless, it is challenging to identify polymorphic distortion innovations and jumbled codes. The required APK consents might help with developing awareness of the risks.

[1] launched a consent-based APK reviewer that makes use of static analysis to categorise Android apps as benign or malicious. In order to ascertain whether Android engineers use the most advantageous strategy when requesting authorizations, WHY-PER, a system that uses natural language processing (NLP) methods to find terms that indicate the requirement for a given

consent in an application representation, was launched in [2].

For Android malware static identification, presented a list of capabilities that included authorizations and API calls, and classifiers that made use of the suggested highlight set outperformed those who only used the consents. Dynamic research strategies based on social attributes can effectively address this issue as more Android malware avoids static locations through tactics like repackaging and code fuzzing.

[3] Taint Droid was proposed as a capable, system-wide, unique pollutant monitoring and investigation framework equipped to work while pursuing numerous sources of sensitive information. By utilising Android's virtualized execution environment, Corrupt Droid provides continuous testing with little CPU overhead.

[4] proposed the ntLeakSemaic structure, which naturally detects odd, touchy organisation signals from a variety of applications. In comparison to current pollution examination approaches, it can achieve improved accuracy with fewer false positives. For cell phone crime scene investigations and post-mortem examinations, [5] proposed the Droid Scraper framework to recover significant runtime information constructs for use in programming by listing and reconstructing the articles in memory. Malware attempts to evade detection by imitating the security-sensitive activities of secure apps and hiding its payload to reduce the likelihood of being found. Because of the setups that lead to behaviours that are security-sensitive.

App Context is a static programme investigation methodology that introduced. It removes the settings of security-sensitive practises to aid application examination in differentiating harmful and harmless practises. Although antivirus software can reliably identify Android malware, it must physically erase the marking code and update the client side after obtaining the malware test. While this technology is highly accurate at locating known malware, it also has a number of drawbacks. For instance, it is impossible to tell apart malware handled using confusion techniques from esoteric malware that has never been seen before. Recently, researchers have used AI computations to identify malware and increase location precision.

[6] promoted a SecureDroid framework. They developed a group learning approach by adding up the individual classifiers and introduced a novel component determination mechanism to make the classifier more difficult to avoid. The method for illustrating malware as a language tested the viability of finding semantics in instances of that language using the two security and communication networks, and they organised malware records by using the KNN. The SVM calculation is widely used in traditional AI calculations for Android malware finding, and it typically has a good arrangement influence.

Correlation data cannot be used by ML algorithms to characterise Android malware due to their weak frameworks. As a result, researchers tried to distinguish Android malware using machine learning techniques. The learning method has a wide range of applications in language processing, speech recognition, and picture processing, and because of its excellent nonlinear relationship fitting ability, it has a promising application possibility in malware identification. Other machine learning algorithms like DBN, LSTM, and stacked autoencoder [7] are frequently used.

As a result of deep learning's success in identifying photos, malware may now be converted into images for the purpose of training and detection [8]. [9] converted the malware into grayscale photographs, which were then identified and categorised by a CNN that could extract features from the properties of the infected images. After deactivating the Android APK and translating the opcodes, API modules, and high-level dangerous API functionalities into 3 channels of an RGB output, used a CNN to identify the features of the malware family. Using a balanced dataset to improve deep neural learning for benign versus harmful implementation paths, [10] proposed a machine learning method for identifying mobile malware that considers all possible execution pathways.

[11] developed an engine for identifying internet-based mobile malware. Some researchers have proposed malware detection systems that use various deep learning techniques to increase detection accuracy and make use of different artificial neural networks. An Attention-CNN-LSTM-based multi-model deep learning-based method for Android threat analysis and detection was proposed by [12]. After comparing the recurrence and CNN-based deep learning models, created an autonomous extraction features component and a hybrid convolutional classification model.

[13] provides a methodology for detecting Android malware using Rock Hyrax Swarm Optimisation and deep learning (RHSODL-AMD). The method described comprises identifying the most important permissions and Application Programming Interfaces (API) calls, which produces effective differentiation between legitimate software and malicious software. [14] suggest and demonstrate a variety of techniques for

Android malware detection. The construction of an in-process detection system with data analytics. It might examine your present app collection using the detection mechanism to seek for any harmful software so you can get rid of it. This is made possible by machine learning-based models. [15] illustrate that permissions can produce robust and efficient malware detection systems when concept drift is addressed. The ability of various sets of characteristics to discriminate is also examined. With an average F1 score of 0.93 over data spanning seven years, we discovered that the baseline set of permissions, as described in Android 1.0 (API level 1), was adequate to create a reliable detection model.

II. PROPOSED METHOD

2.1. SigPID

The goal of the SigPID system is to examine the fewest number of permits while still accurately and quickly identifying malware. Our technology uses software programmes' access lists to accomplish this, but instead of focusing on all rights, SigPID focuses on authorizations that potentially boost virus detection rates. This significantly lessens the requirement to review licences that don't really affect how well attacks are detected. In a nutshell, SigPID uses multi-level data trimming to reduce permissions with low influence on detection efficiency, which comprises three primary components: (i) permission rating with a negative interest rate; (ii) authorization extraction using frequent patterns; and (iii) support-driven licence ranking. To detect suspected mobile malware, SigPID uses supervised learning classification techniques after trimming. Figure 1 depicts the system assessment process, which is divided into three parts: Data Pre-Processing, Building Systems to Detect and Recognise Analysis,

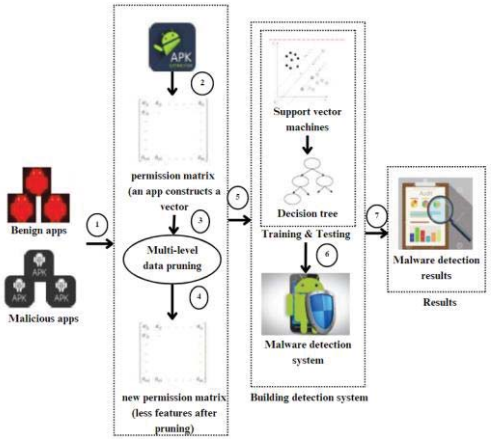

Fig. 1: Overview of the System

2.2MLDP

The multi-level database pruning procedure that SigPID employs to condense our authorization database is an essential part of the system. Additional permissions for mobile apps can total up to 129 in total. All 129 permissions are not typically requested by programmes. In the back of manifest.xml, the components that an application needs are listed. When reviewing a large number of applications, the total number of rights requested by all applications may be astronomically large, necessitating a lengthy investigation. This significant processing overhead may reduce the effectiveness of malware identification because it reduces analyst productivity. Use three phases of data trimming procedures to remove rights that have no bearing on malware detection to address this problem.

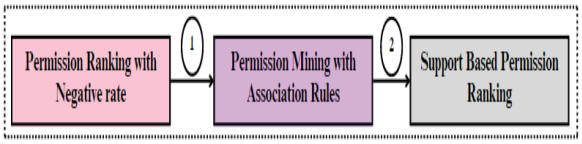

Fig. 2: Multi-Level Data Pruning

As a result, they could be securely eliminated without compromising malware detection performance. Figure 2 shows the entire method in detail. After that, researchers go over each stage of the pruning procedure.

2.2.1 Negative Rate Permission Ranking

Each permission specifies a certain action that an app is allowed to perform. The programme's Internet connectivity is indicated, for instance, by the licence "Internet." Various types of helpful and harmful apps may request different permissions depending on how they need to function. Creators think there are distinct groupings of requirements for dangerous apps. There has been research that looks for malicious activity using permissions. As a result, this study might not need to look at all 134 rights in order to build a trustworthy malware recognition system if they do fall into comparable subgroups.

Prioritising permissions based on how hazardous and innocent apps use them is the next step. Ratings are not an entirely new idea. Finding high-risk rights has previously been done using a general authorization ranking method like mutual data. Their methods, on the other hand, frequently ignore no-risk privileges, which are crucial permissions in our method, and tend to concentrate only on high-risk rights. An approach called PRNR generates results that are easy to interpret and have

a clear ranking. As a result, it was start by figuring out their ratio using the formula below:

size (Mj) denotes the number of rows in M, while size (Bj) denotes the number of rows in B. Adaptable "offers the difference between the two vectors' sizes. In this project, "due to a huge number of benign applications, is extremely small. The number of "policies" depending on multiple sorts of datasets could be manually specified.

2.2.2 Permission Mining with Association Rules

This section provides an explanation of the MLDP's second pruning procedure. After eliminating 56 of 129 non-influential rights using PRNR and PIS, the authors now wish to find strategies to further minimise non-influential rights. Researchers can see that some rights always appear together in an application by checking the limited access list. For instance, the WRITE SMS and READ SMS permissions are always combined. Both of them are listed as having dangerous permissions. This study can provide the entity with additional assistance from its buddy as a result. In this approach, a single authorization might reflect both of them. Researchers can disable the "Write SMS privilege in this situation.

2.3 Integration of Numerous Lists and X-Values in Enhanced MLDP

When researchers use MLDP for large permissions, SIGPID has two noticeable inefficiencies. The system uses two PIS for MLDP, hence the existence of two rating lists. Each PIS will probably require some time to process. In order to get over these two MDLP constraints, this study propose a novel method called FML and a novel characteristic called X-value. After that, researchers will discuss each one separately.

2.3.1 Multiple List Fusion

The first technique rates each list individually in the MLDP before combining multiple lists into a single one. FML allows us to add PIS to the final list just once, as opposed to the two times required by MLDP before applying this alteration. The procedure can be improved by making this change because it takes time to analyse PIS. The fusion process consists of three phases.

Step 1 of the approach involves combining the +ive and -ive rate lists. When determining the R (Pj), each authorization's rate varies from -1 to 1. However, by calculating the absolute value of all rates R(Pj) for each permission.

Step 2: Standardise the scores of each list to the ideal value after the initial data reduction. Each new score will therefore range from 0 to 1.

Step 3: Select a function for multiple normalised lists.

Step 4: Make a table with two normalised scores from the frequency list and the positive-rate list for each permit. S2i is the normalised score from the recurrence list for the ith permission, while S1i is the normalised score from the positive-rate list for the ith permission in the table, shown in equation 1:

$$S_i = \frac{(S1_i + S2_i)}{2} \quad (1)$$

For ith approval, equation 2 and 3 examines the entire score of S1i and S2i. When S2i is a very large score, even though S1i is a tiny score, ith permission might still be deemed major permission.

$$S_i = S1_i * S2_i \quad (2)$$

$$S_i = 1/2 * (S1_i + S2_i) * (S1_i * S2_i) \quad (3)$$

2.3.2 X-value

To identify the PIS permission with the fewest permissions, researchers use the X-value. The X-value, or mean gap between recall, f-measure, and precision, can be used to determine the minimum number of rights necessary to get a high and stable f-measure. Accuracy first decreases before progressively increasing. Recall increases initially before declining. Accuracy, f-measure, and recall all approach a critical point at which f-measure becomes stable, suggesting that changes only occur within specific ranges. Table 1 displays the PRNR scores.

Table 1: PRNR Rankings

Permission Ranking with Negative rate	
Benign List	Malicious List
READ_SOCIAL_ STREAM	SEND_SMS
BIND_ ACCESSIBILITY _SERVICE	RECEUVE_SMS
WRITE PROFILE	BRICK

READ_PROFILE	RECEIVE_MMS
NFC	BROADCAST_PACKAGE_REMOVED
SET_ALARM	READ_SMS
READ_CALL_LOG	WRITE_APN_SETTINGS

III. RESULT AND DISCUSSION

3.1. Evaluation

The SigPID system's spam detection performance is assessed. To create our malware sensor, researchers used 10,988 apps in our analysis. Only 33 essential permissions are identified by trimming, and when these rights are utilised to detect malicious activity, our approach achieves 89.1 percent accuracy. Following that, this work will go over the significant findings from our performance review. The results of multi-level data filtering are shown in Table 2.

Table 2: Multi-Level Data Pruning Results

Number of features	**34**	**60**	**66**	**135**
STD	1.99756	1.98647	1.4789	1.74668
Accuracy	93.09	82.09	98.45	65.86
Status	99.08	87.67	78.98	100.67
Precision	98.98	89.56	98.23	98.86
Recall	95.09	95.78	108.55	88.09
F-measure	94.99	98.78	96.98	99.45

1. The three main components of multi-level reduction of data are support-based permission ranking, association rule-based authorization mining, and a negative permission rate. This work evaluate malware detection by turning on each of these levels to confirm how much each level of data mining methodology contributes to performance improvement. In terms of runtime, researchers also evaluate the effectiveness of multi-level data reduction. The SVM algorithm is used to identify malware.
2. The Capability of Different Deep Learning Approaches to Detect Malware and put the SigPID mechanism to the test using a range of information mining and machine learning techniques.
3. Evaluation of Alternative Methodologies were compare the outcomes of methods using various permission ranking algorithms, like Mutual Data, to SigPID's categorization effectiveness.

3.2 Data Set

The researcher will demonstrate how to generate a permission database in this section. While benign programmes are gathered into a single group, malicious apps are separated into 177 families. The requested authorization list is thereafter created by extracting the access requests from each programme listed in the AndroidManifest file. A value of 1 indicates that the app requests the permissions, whereas a value of 0 indicates that it does not. The permission data is then transformed into a binary dataset. The integration of permission files from both malicious and helpful programmes creates an extensive database for the processing of data.

3.3 Pruning data at multiple levels

Every element of a multi-level data reduction system is evaluated by researchers for effectiveness. The efficacy of the recognition system is first evaluated after allowing authorization ratings with a negative rate. There is also another rating system for authorization that uses mutual data. Second, it assesses the efficacy of detection when association rules-based data mining has been enabled. The detection effectiveness is then assessed when the support-based approval rating has been activated. Finally, it examines the efficacy of multi-level data pruning by merging all three strategies.

3.3.1 Using a Negative Rate to Evaluate the Permission Rating

First, create the good and bad permission ranking lists using PRNR. Then, according to the ranking lists, researchers incrementally increase privileges using the PIS. Two permissions from each category are added to the primary permission list for each phase.

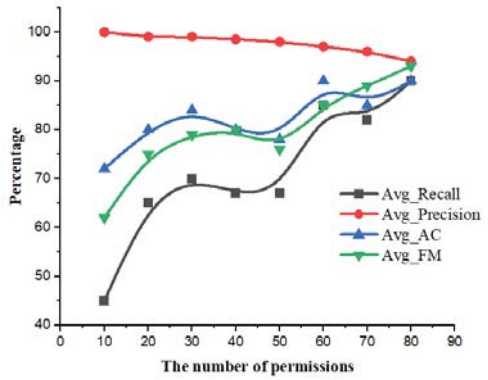

Fig. 3: Permission Incremental System Malware Detection Performance

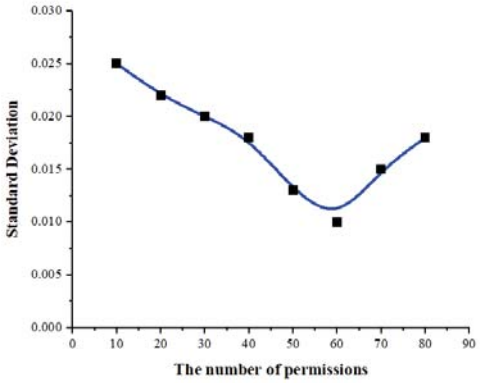

Fig. 4: The Standard Error with Increasing Permissions

According to Figures 3 and 4, when the number of permissions increases, the prediction performance, retention, and F-measure improve. Meanwhile, the precision drops slightly with each round, although it always remains over 95%, as seen in Fig. 3. One interesting finding is that after the level of permissions hits 66, precision, F-measure, and recall are all at the same level. The standard error of the F-measure is also shown in Fig. 4. The outcomes of using 67 ML algorithms are shown in Table 3.

Table 3: Findings from 67 ML Algorithms

Number of features	Precision	Recall	F-measure	ROC	Training (seconds)	Testing (seconds)	Total
45	88.23%	99.73%	96.94%	92.34%	1.55%	3.23%	3.56%
144	93.76%	100.89%	93.43%	95.67%	5.34%	23.43%	23.45%

3.3.2 Using Association Rules to Evaluate Permission Mining

The system is examined following the application of the PRNR technique, and the results are contrasted with those of the model with 135 permissions. Both techniques produce F-measure accuracy of roughly 90%, as shown in Table 3. The revised malware model with 66 rights has a better recall rate but a lower precision rate than the outdated classification technique with 135 permissions, according to a detailed examination of the data. Or, to put it another way, while the new technology accurately detects more malware, it sacrifices accuracy when detecting genuine programmes. However, the most recent model performs just marginally better in terms of precision and F-measure.

3.4 Using Different Machine Learning Algorithms to Evaluate Malware Detection Efficiency

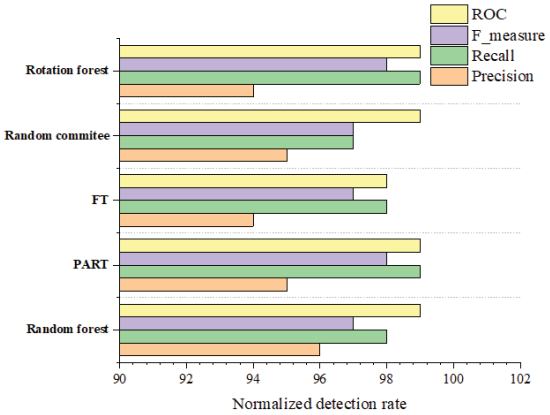

Fig. 5: Top 5 Machine Learning Algorithms Outcomes

Figure 5 shows the detection performance with 33 licences normalised by the model's performance with 135 permissions. In terms of reliability, durability, ROC, and F-measure, the malware detection model with a complete access list surpasses the detection algorithm with 34 privileges, but the difference is minimal. Table 3 lists the top 5 malware identification models with an F-measure of more than 89 percent and other performance metrics of more than 85 percent, demonstrating that the detection approach can still accurately identify benign and malicious programmes.

Figure 6 shows the completion time of a system with 34 permissions when compared to a version with 135 permissions. The malware sample identification model takes around half as long as the one with the full permission list. Meanwhile, a dataset with fewer characteristics can save a large amount of memory.

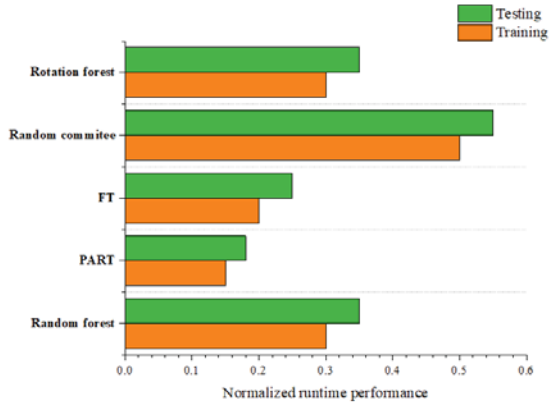

Fig. 6: Top 5 Deep Learning Algorithms' Runtime Efficiency

Five of the top ten most efficient methods use tree architectures. Our MLDP can assist considerably in increasing the operating time productivity of the virus detection system depending on tree topologies, as tree structure-based methods often consume a lot of time and space to execute the categorization process.

3.5 MLDP Improvement

3.5.1 MLDP's scalability

Researchers showed that MLDP can work successfully by using 2,765 hazardous apps and a selection of 2,765 benign apps from a corpus of 342,675 benign applications. This found that 34 of the 135 permissions are important. Our technology detects malware with 91.9 percent reliability when these 34 rights are utilised. Our proposed method outperformed the whole authorization set in terms of both f-measure and precision.

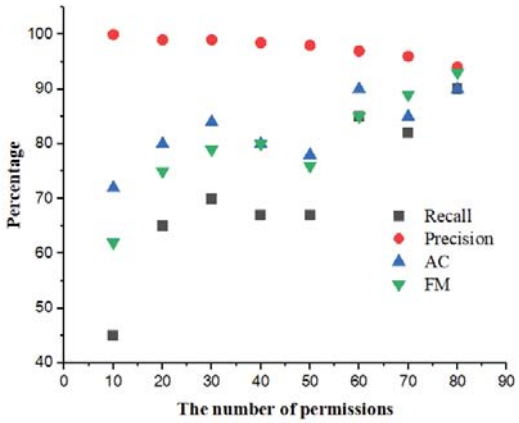

Fig. 7: 1st Step: Efficacy of PIS in Malware Analysis in PRNR

To establish a stable system depending on PRNR, at least 89 rights are required, as shown in Fig. 7. This discrepancy is to be expected, as various datasets might provide various PIS stopping points. The graph remains stable fairly quickly after complete the SPR, as shown in Fig. 8.

Table 4 provides more information on the f-measure and accuracy. When researchers use 25 authorizations in the second PIS with SPR, and get the best f-measure.

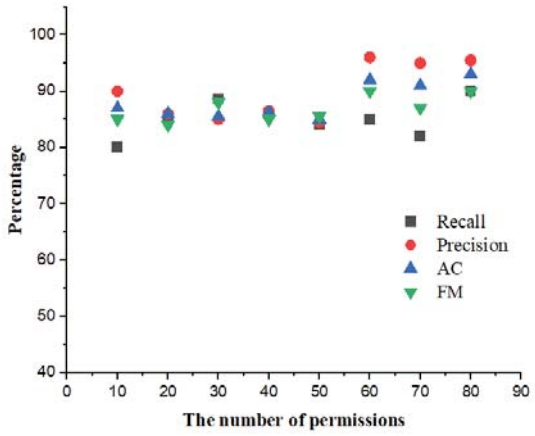

Fig. 8: 2nd Step: Efficacy of PIS in Malware Analysis in SPR

Tab. 4: Efficacy of PIS in SPR Malware Analysis

Number of feature	10	20	30	40	50	60
X-Value	3.48666	1.6657	1.977	1.99	1.89	1.686
FM	102.9	86.56	105.6	94.63	92.7	97.56
Recall	99.36	103.14	92.26	97.59	88.6	99.89
Precision	92.63	89.63	83.36	88.36	83.4	105.4
ACC	98.32	103.56	99.36	101.1	102	104.3

IV.CONCLUSION

This work, demonstrate that mobile virus detection can be extremely accurate and efficient while reducing the amount of licences that need to be checked. SigPID uses a three-level trimming technique to extract only significant permits. Based on our database of over 1000 malware attacks, and can increase runtime performance by 85.6 percent and detection accuracy by over 90% by only evaluating 34 of 135 permissions. The results show that employing an SVM as the classifier can lead to over 90% accuracy, recall, reliability, and F-measure, which are comparable to those discovered using the baseline technique while having a research that is 4-32 times less expensive than when using all rights. With a detection rate of 94 percent for viruses and 92 percent for unidentified malware samples in the data, SigPID exceeds other cutting-edge methods. Researchers need a modular malware detection system that can quickly locate

and identify the most important bad programmes in order to stop this massive strike. There are many ways to identify malware, including system- and network-based techniques. On the other side, it is still difficult to expand detection for many different programmes.

REFERENCES

[1] Arora, S. K. Peddoju, and M. Conti, "PermPair: android malware detection using permission pairs," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1968–1982, 2020. https://doi.org/10.1109/TIFS.2019.2950134

[2] W. Wang, J. Wei, S. Zhang, and X. Luo, "LSCDroid: malware detection based on local sensitive API invocation sequences," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 174–187, 2020. https://doi.org/10.1109/TR.2019.2927285

[3] H. Cai, N. Meng, B. Ryder, and D. Yao, "DroidCat: effectiveAndroid malware detection and categorization via app-level profiling," *IEEE Transactions on Information Forensics & Security*, vol. 14, no. 6, pp. 1455–1470, 2018. https://doi.org/10.1109/TIFS.2018.2879302

[4] Ali-Gombe, S. Sudhakaran, A. Case, and G. G. Richard, "DroidScraper: a tool for Android in-memory object recovery and reconstruction," in *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses*, pp. 547–559, Beijing, China, October 2019.

[5] M. A. Kadri, M. Nassar, and H. Safa, "Transfer learning for malware multi-classification," in *Proceedings of the 23rd International Database Applications & Engineering Symposium (IDEAS'19)*, pp. 1–7, Athens, Greece, June 2019. https://doi.org/10.1145/3331076.3331111

[6] W. C. Kuo, T. P. Liu, and C. C. Wang, "Study on Android hybrid malware detection based on machine learning," in *Proceedings of the IEEE International Conference on Computer and Communication Systems (ICCCS 2019)*, pp. 31–35, Singapore, February 2019. https://doi.org/10.1109/CCOMS.2019.8821665

[7] X. Yuan, J. Zhou, B. Huang, Y. Wang, C. Yang, and W. Gui, "Hierarchical quality-relevant feature representation for soft sensor modeling: a novel deep learning strategy," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 3721–3730, 2020. https://doi.org/10.1109/TII.2019.2938890

[8] Y. Wang, Z. Pan, X. Yuan, C. Yang, and W. Gui, "A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network," *ISA Transactions*, vol. 96, pp. 457–467, 2020. https://doi.org/10.1016/j.isatra.2019.07.001

[9] F. Mercaldo and A. Santone, "Deep learning for image-based mobile malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 16, no. 2, pp. 157–171, 2020. https://doi.org/10.1007/s11416-019-00346-7

[10] Pektaş and T. Acarman, "Learning to detect Android malware via opcode sequences," *Neurocomputing*, vol. 396, pp. 599–608, 2020. https://doi.org/10.1016/j.neucom.2018.09.102

[11] S. Luo, Z. Liu, B. Ni, H. Wang, H. Sun, and Y. Yuan, "Android malware analysis and detection based on Attention-CNNLSTM," *Journal of Computers*, vol. 14, no. 1, pp. 31–43, 2019.

[12] H. Safa, M. Nassar, and W. A. R. A. Orabi, "Benchmarking convolutional and recurrent neural networks for malware classification," in *Proceedings of the 15th International Wireless Communications & Mobile Computing Conference (IWCMC 2019)*, pp. 561–566, Tangier, Morocco, June 2019. https://doi.org/10.1109/IWCMC.2019.8766515

[13] Albakri, Ashwag, et al. "Metaheuristics with Deep Learning Model for Cybersecurity and Android Malware Detection and Classification." *Applied Sciences* 13.4 (2023): 2172. https://doi.org/10.3390/app13042172

[14] Akhtar, Muhammad Shoaib. "Analyzing and comparing the effectiveness of various machine learning algorithms for Android malware detection." *Advances in Mobile Learning Educational Research* 3.1 (2023): 570-578. https://doi.org/10.25082/AMLER.2023.01.005

[15] Guerra-Manzanares, Alejandro, Hayretdin Bahsi, and Marcin Luckner. "Leveraging the first line of defense: A study on the evolution and usage of android security permissions for enhanced android malware detection." *Journal of Computer Virology and Hacking Techniques* 19.1 (2023): 65-96. https://doi.org/10.1007/s11416-022-00432-3