International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

Malicious Android Application Package(APK) Detection Using Deep Learning

Syed Shoeb Ali[1] , Syed Parvez Uddin[2] , Mohammed Faiz Khan[3] , Mohammed Rahmat Ali[4]

[1,2,3]B.E Students, Department Of CSE, ISL Engineering College HYD India,

[4]Assistant Professor, Department Of CSE, ISL Engineering College HYD India,

ABSTRACT

As of late, the uses of advanced mobile phones are expanding relentlessly and furthermore development of Android application clients are expanding. Because of development of Android application client, some gate crashers are making vindictive android application as instrument to take the delicate information and data for fraud and misrepresentation portable bank, versatile wallets. There are such a large number of malevolent applications discovery instruments and programming's are accessible. Be that as it may, a viably and productively vindictive application recognition device expected to handle and deal with new complex pernicious applications made by interloper or programmers. This paper Utilizing Machine Learning approaches for distinguishing the malignant android application. First, dataset of past pernicious applications has to be obtained with the assistance of Help vector machine calculation and choice tree calculation make up correlation with preparing dataset. The prepared dataset can foresee the malware android applications up to 93.2 % obscure/new malware portable application

The main steps performed through this framework are sketched as follows:

1. A set of features is computed for every binary file in the training or test datasets, based on many possible ways of analyzing a malware.
2. A Deep learning system based firstly on one-sided perceptron's, and then on feature mapped one-sided perceptron's and a kernelized one-sided perceptron's, combined with feature selection based on the F1 and F2 scores, is trained on a medium-size dataset consisting of clean and malware files. Cross-validation is then performed in order to choose the right values for parameters. Finally, tests are performed on another, non-related dataset. The obtained results were very encouraging.

INTRODUCTION

The rapid growth of mobile devices and remarkable advances in 4G/5G mobile networking technologies have both inspired and facilitated a plethora of mobile applications (apps). In support of these mobile devices, various mobile operating systems have been developed, such as Android, iOS, and BlackBerry 10, etc. Among these mobile operating systems, Android has become the most popular one because of numerous mobile apps it provides. Unfortunately, smartphones running Android have been increasingly targeted by attackers and infected with malicious apps: according to the mobile threat report released by F-Secure in 2014 Q1, over 95% of malicious apps were distributed on the Android platform. Moreover, as the major sources of Android apps, third-party markets contain a lot of cracked or tampered apps, and there is no sign to indicate whether the apps have been checked for security risks or not in these third-party markets. The lack of security inspection on Android apps intensifies the spreading of malicious apps (malware) on the Android platform. Zhou et al. performed some Android malware analysis by popular security software tools, and the detection rate was only between 20.2% and 79.6%, which clearly indicates an urgent and rapidly growing demand on malware detection for Android applications. To address the increasingly wide-spread security risks, the Android platform itself provides several security solutions that harden the installation of malware, such as the Android permission system and Google ―Bouncer‖. To perform certain tasks on Android devices, such as sending a SMS message, each Android app has to explicitly request the corresponding permission from the user during the installation process. However, many users tend to arbitrarily grant permissions to unknown Android apps without even looking at what types of permissions they are requesting and thereby significantly weaken the protection provided by the Android permission system. As a result, it is very difficult to limit the propagation of malicious apps by the Android permission system in practice. On the other hand, Google ―Bouncer‖ is a service added to Google Play, the official Android market, in 2012, which aims to automatically scans apps (both new and previously uploaded ones) and developer accounts in Google Play with its reputation engine and cloud infrastructure. Even if Bouncer adds another line of defense for Android security, it still has many limitations. First, Bouncer can only scan Android apps for limited time, which means a malicious app can easily bypass it by not doing anything malicious during the scan phase. Second, no malicious code needs to be included in the initial installer when it gets scanned by Bouncer. In this case, the malicious app can have an even higher probability to evade Bouncer's detection. Once the application passes

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP184-191

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

Bouncer's security scan and gets installed on a real user's Android device, then the malicious app can either download additional malicious code to run or connect to its remote control and command (C&C) server to upload stolen data or receive further commands. Recently, there have been some research efforts on detecting Android malware by using various machine learning algorithms. Drebin extracts permissions, APIs and IP address as features and uses the Support Vector Machine (SVM) algorithm to learn a classifier from the existing ground truth datasets, which can then be used to detect unknown malware. DroidMat alternatively applies KNN (K-Nearest Neighbor) algorithm to permissions and intents to identify malware. In addition, DoridAPIMiner focuses on providing several lightweight classifiers based on the API level features. However, the exiting malware detection approaches have some limitations that have to be addressed. For instance, the recall value of DroidMat is significantly lower than its precision value, which indicates that some malware cannot be correctly detected. In addition, the high accuracy of DroidAPIMiner owes to the fact that its testing dataset contains far more benign pps than malware. Moreover, it generally takes a large amount of time for Drebin to build the classifier because of the high-dimensional feature vector, which contains over 545,000 features. Thus, it is critical to develop an effective and efficient approach to detect malware for Android platform. In this paper, we propose a malware detection scheme for Android platform using the SVM-based approach. In this scheme, we first calculate the similarity scores between malware and benign applications in terms of suspicious API calls, and use these similarity scores as features in the feature vector. Then, we also use risky permission requests as additional features when we train the SVM classifier. Experimental results on real benign and malicious application dataset show that the proposed scheme can accurately identify Android malware.

LERATURE REVIEW

Many research studies on android malware detection have been performed in recent years, in which static analysis and dynamic analysis are the two major directions among these research works. It gives an overview of Android malware regarding its dangerous behaviors and vulnerabilities. static Analysis Based Malware Detection The first approach for detecting Android malware is inspired by concepts from static program analysis. Several methods have been proposed that statically inspect mobile apps and disassemble the code. Decompiling and data flow tracking are two main techniques in most of the static analysis methods. For instance, Kirin checks the permission of Android apps for indications of malicious activity. Similarly, Stowaway analyzes API calls to detect over-privileged apps, and Risk Ranker statically identifies Android apps with different security risks. There are also some common open-source tools for static analysis such as Smali and Anroguard, which enable dissecting the content of apps with little effort. In addition, Android Leaks focuses on finding privacy or sensitive data leaks for Android apps using decompiled code. Flow Droid aims to identify malware by building precise model of Android's lifecycle. Leak Miner is a tool that detects leakage of sensitive information on Android with static taint analysis.

Dynamic Analysis Based Malware Detection

Dynamic analysis focuses on detecting Android malware when Android apps are being executed, and most of them monitor the behaviors of apps in terms of accessing private data or using restricted API calls. For instance, both TaintDroid and DroidScope dynamically monitor Android apps when they are actually executed, where the former focuses on taint analysis and the latter aims to introspection at different layers of Android system

In general, dynamic analysis cannot be performed on the smartphones themselves to directly identify malware because they usually incur a large amount of computational overhead. Thus, they are mainly used to detect Android malware offline via scanning and analyzing a lot of Android applications. For example, malware detection schemes such as AppsPlayground [17] and DroidRanger [18] have been proposed to dynamically analyze Android apps and detect the possible malicious behaviors they may perform. **Malware Detection Using Machine Learning Algorithms**

It is generally difficult to manually specify and update detection patterns for Android malware in static analysis process. Therefore, some recent research efforts have been focusing on using various machine learning algorithms to automatically extract app features efficiently distinguish malicious apps from benign ones by models without manual operation. Drebin , DroidMat , and DroidAPIMiner build models with different machine learning algorithms on features, including permissions, API calls, and so on.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

- System : Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz
- Hard Disk : 1 TB.
- Input Devices : Keyboard, Mouse
- Ram : 8 GB.

SOFTWARE REQUIREMENTS:

- Platform : Anaconda

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP184-191

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

- Front End :Flask Framework
- Operating system : Windows 10.
- Coding Language :Python
- Tool :Anaconda
- Algorithms sed : Neural Network Deep Learning Algorithm
  Support Vector Classifier Machine Learning Algorithm

THEORETICAL BACKGROUND

What Is OpenCV?

OpenCV [OpenCV] is an open source (see http://opensource.org) computer vision library available from http://SourceForge.net/projects/opencvlibrary.

OpenCV was designed for computational efficiency and having a high focus on real-time image detection. OpenCV is coded with optimized C and can take work with multicore processors. If we desire more automatic optimization using Intel architectures

[Intel], you can buy Intel's Integrated Performance Primitives (IPP) libraries [IPP]. These consist of low-level routines in various algorithmic areas which are optimized. OpenCV automatically uses the IPP library, at runtime if that library is installed.

One of OpenCVs goals is to provide a simple-to-use computer vision infrastructure which helps people to build highly sophisticated vision applications fast. The OpenCV library, containing over 500 functions, spans many areas in vision. Because computer vision and machine learning oft en goes hand-in-hand,

OpenCV also has a complete, general-purpose, Machine Learning Library (MLL).

This sub library is focused on statistical pattern recognition and clustering. The MLL is very useful for the vision functions that are the basis of OpenCV's usefulness, but is general enough to be used for any machine learning problem.

What Is Computer Vision?

Computer vision is the transforming of data from a still, or video camera into either a representation or a new decision. All such transformations are performed to achieve a particular goal. A computer obtains a grid of numbers from a camera or from the disk, and that's that. Usually, there is no built in pattern recognition or automatic control of focus and aperture, no cross-associations with years of experience. For the most part, vision systems are still fairly naïve.

The Origin of OpenCV

OpenCV came out of an Intel Research initiative meant to advance CPU-intensive applications. Toward this end, Intel launched various projects that included real-time ray tracing and also 3D display walls. One of the programmers working for Intel at the time was visiting universities. He noticed that a few top university groups, like the MIT Media Lab, used to have well-developed as well as internally open computer vision infrastructures—code that was passed from one student to another and which gave each subsequent student a valuable foundation while developing his own vision application. Instead of having to reinvent the basic functions from beginning, a new student may start by adding to that which came before.

OpenCV Structure and Content

OpenCV can be broadly structured into five primary components, four of which are shown in the figure. The CV component contains mainly the basic image processing and higher-level computer vision algorithms; MLL the machine learning library includes many statistical classifiers as well as clustering tools. HighGUI component contains I/O routines with functions for storing, loading video & images, while CXCore contains all the basic data structures and content.

Why OpenCV?

Specific OpenCV was designed for image processing. Every function and data structure has been designed with an Image Processing application in mind. Meanwhile, Matlab, is quite generic. You can get almost everything in the world by means of toolboxes. It may be financial toolboxes or specialized DNA toolboxes.

Speedy Matlab is just way too slow. Matlab itself was built upon Java. Also Java was built upon C. So when we run a Matlab program, our computer gets busy trying to interpret and compile all that complicated Matlab code. Then it is turned into Java, and finally executes the code.

If we use C/C++, we don't waste all that time. We directly provide machine language code to the computer, and it gets executed. So ultimately we get more image processing, and not more interpreting.

After doing some real time image processing with both Matlab and OpenCV, we usually got very low speeds, a maximum of about 4-5 frames being processed per second with Matlab. With OpenCV however, we get actual real time processing at around 30 frames being processed per second.

EXISTING SYSTEM:

Suleiman et al proposed a classification approach based on parallel machine learning to detect android malware. Depends on real malware samples and benign applications have derived from it, a total of 179 training features were extracted and divided into api calls and commands related: 54 features. App permissions: 125. A composite classification model was developed from a parallel set of heterogeneous classifiers, namely simple logistic,

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP184-191

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

naïve bayes, decision tree, part, and ridor.

DISADVANTAGES OF EXISTING SYSTEM:

Accuracy of these Models are less and prediction of results are not correct All types of features are not taken as input.
Algorithm detects genuine or fake only.

PROPOSED SYSTEM:
The schematic representation of the architecture of this work shows the first phase starting with the mixture of malware and benign Android Application Package (APK) files taken from a dataset that is publicly available to everyone. Then, we extract the static features by our own Python script written in Jupiter notebook environment. The features are API-calls and permissions. These features are formatted and stored in Comma Separated Values (CSV) file as a data frame that serves the training process. Finally, we test all classifiers using 10-folds cross-validation, in addition to calculating the appropriate metrics for the evaluation

ADVANTAGES OF PROPOSED SYSTEM:
Accuracy of the model is high deep learning methods are used to train model Prediction of malicious or not is also possible form this model.
Time taken for prediction is less.

Identifying Android Malware Using SVM Algorithm

Matlab uses just way too much system resources. With OpenCV, we can get away with as little as 10mb RAM for a real-time application. Although with today's computers, the RAM factor isn't a big thing to be worried about. However, our drowsiness detection system is to be used inside a car in a way that is non-intrusive and small; so a low processing requirement is vital.
Thus we can see how OpenCV is a better choice than Matlab for a real-time drowsiness detection system.

Real – Time Streams

Media Pipe calculator graphs are often used to process streams of video or audio frames for interactive applications. Normally, each Calculator runs as soon as all of its input packets for a given timestamp become available. Calculators used in real-time graphs need to define output timestamp bounds based on input timestamp bounds in order to allow downstream calculators to be scheduled promptly. See Real-time Streams for details.
Support Vector Machine

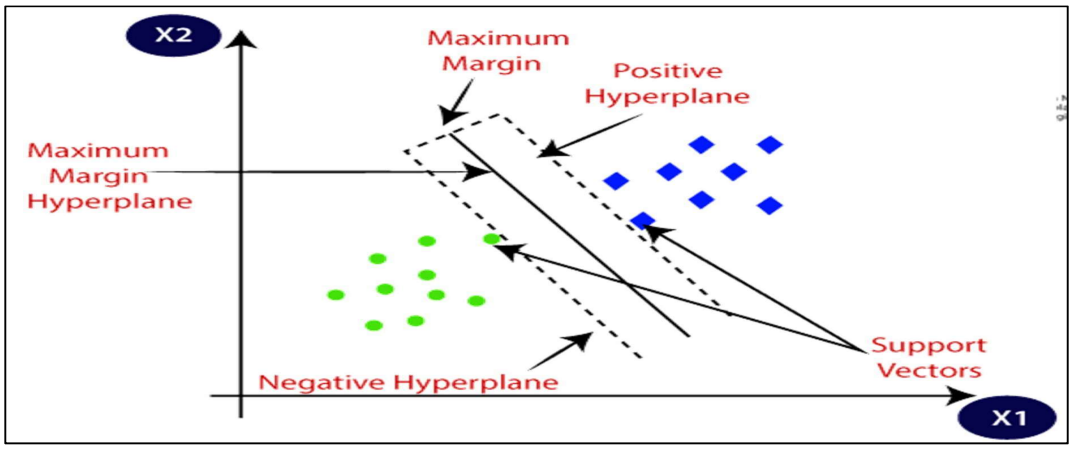
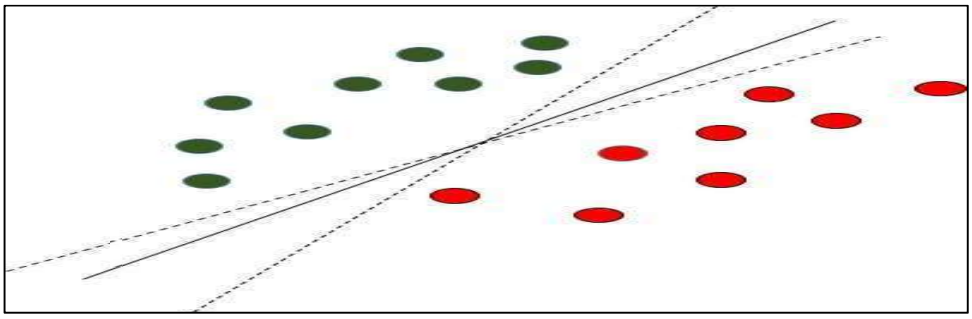


Fig 2.2 SVM Hyper Plane



DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP184-191

ISSN 2347-3657
International Journal of Information Technology & Computer Engineering
Volume 13, Issue 2s, 2025

Fig. 2.3 Multiple Decision Boundaries

A support vector machine (SVM) is a type of supervised machine learning classification algorithm which outputs an optimal hyperplane that categorizes new examples given labeled training data. SVMs were introduced initially in 1960s and were later refined in 1990s. However, it is only now that they are becoming very popular, owing to their ability to achieve outstanding results.

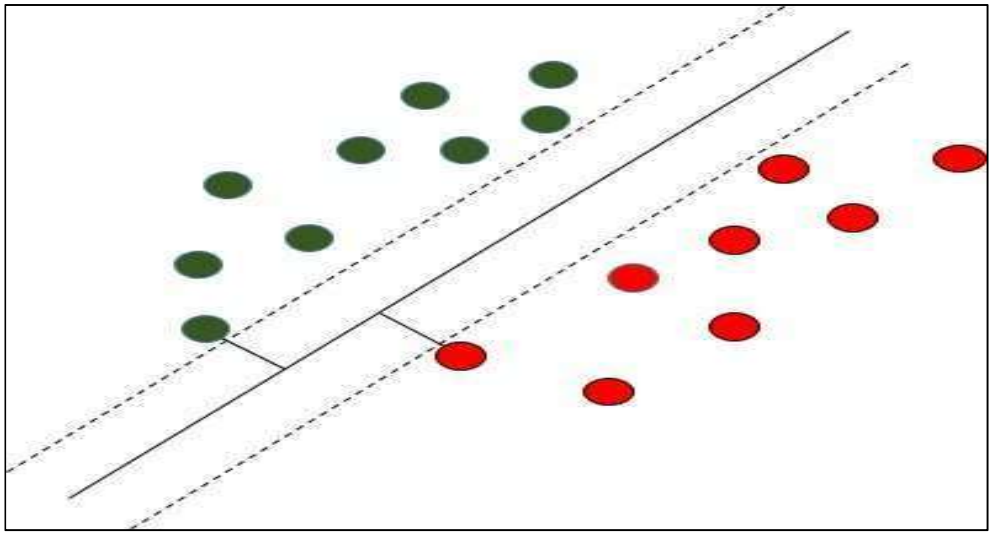


Fig. 2.4 Decision Boundary with Support Vectors

SVM differs from the other classification algorithms in the way that it chooses the decision boundary that maximizes the distance from the nearest data points of all the classes. An SVM doesn't merely find a decision boundary; it finds the most optimal decision boundary.

and Figure 2.7 it was shown how the simple SVM algorithm can be used to find decision boundary for linearly separable data. However, in the case of non-linearly separable data, such as the one shown in Figure 2.8, a straight line cannot be used as a decision boundary.

Kernel SVM: In the previous two figures Figure 2.6

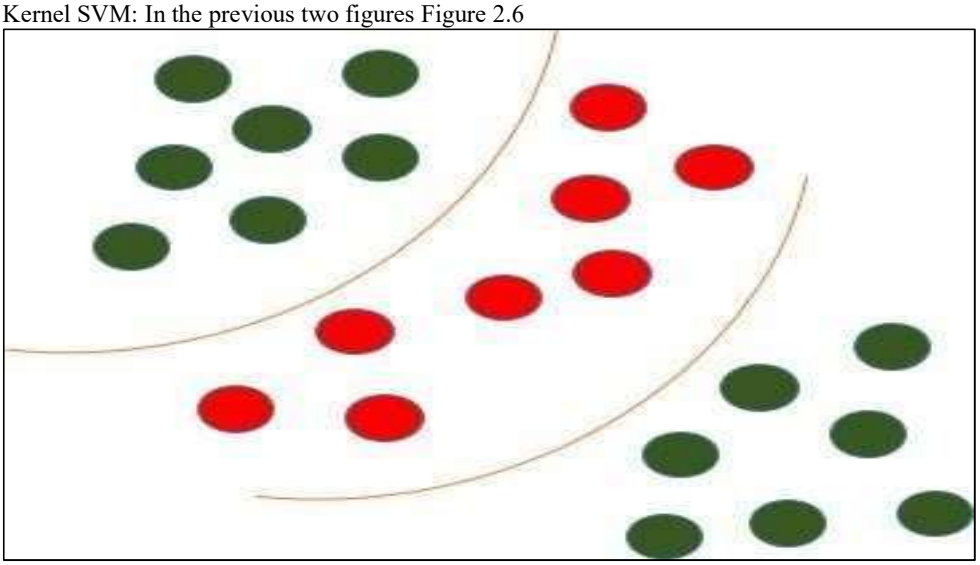


Fig. 2.5 Non-linearly Seperate able Data

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP184-191

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

In case of non-linearly separable data, the simple SVM algorithm cannot be used. Rather, a modified version of SVM, called Kernel SVM, is used. Basically, the kernel SVM projects the non-linearly separable data lower dimensions to linearly separable data in higher dimensions in such a way that data points belonging to different classes are allocated to different dimensions.

Neural Network:

An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.
We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

THE ARCHITECTURE OF AN ARTIFICIAL NEURAL NETWORK:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Let us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:

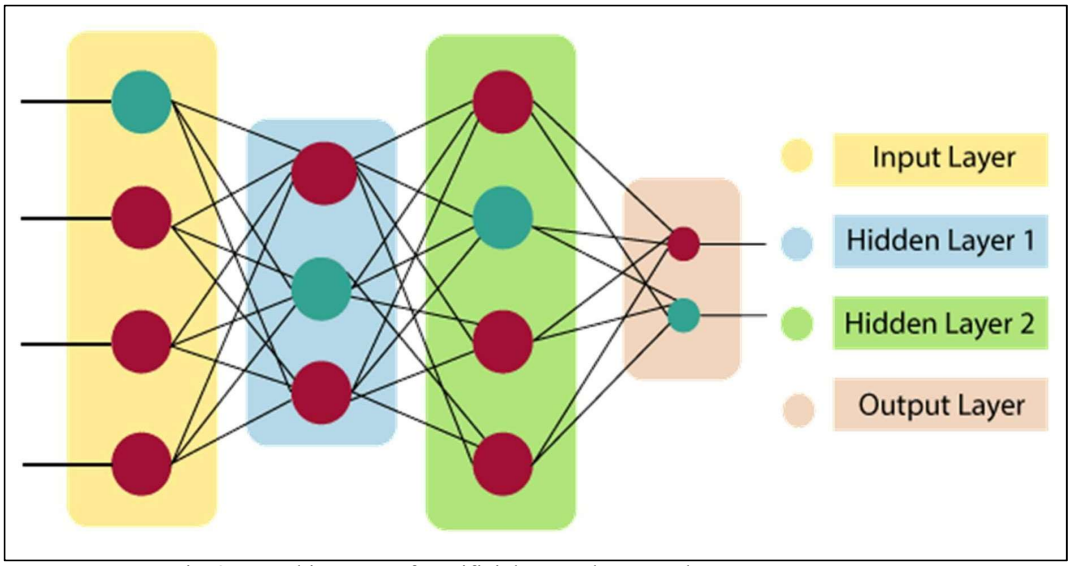


Fig 2.5 Architecture of Artificial Neural Network **Input Layer:**

As the name suggests, it accepts inputs in several different formats provided by the programmer.

Hidden Layer:

The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

Output Layer:

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP192-203

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system. Organized in a way that supports reasoning about the structures and behaviors of the system.

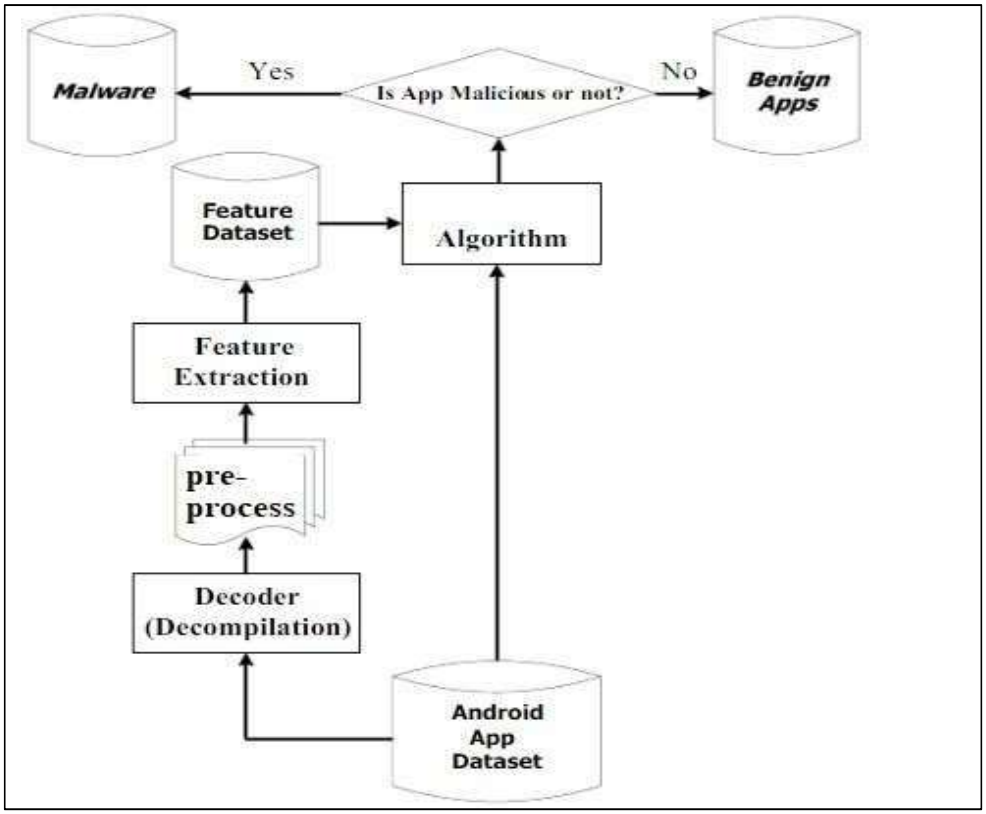


Figure 5. 1 System Architecture

3 -Tier Architecture:

The three-tier software architecture (a three-layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP192-203

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

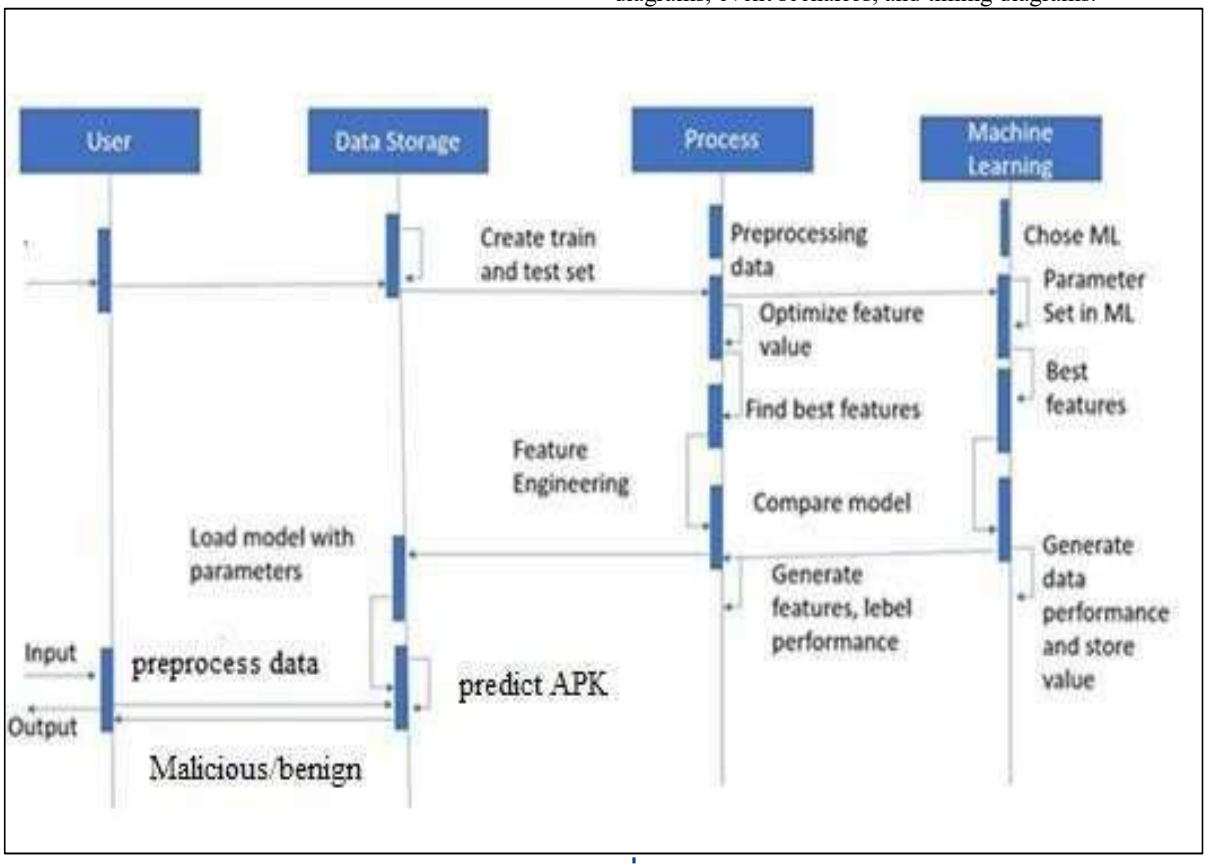
Volume 13, Issue 2s, 2025

Advantages of Three-Tier:

- Separates functionality from presentation.
- Clear separation – better understanding.
- Changes limited to well define components.
- Can be running on WWW.
- Effective network performance.

SEQUENCE DIAGRAMS:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



SYSTEM IMPLEMENTATION

To conduct studies and analyses of an operational and technological nature, and To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

INPUT AND OUTPUT DESIGNS

LOGICAL DESIGN

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

PHYSICAL DESIGN

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP192-203

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor.
The H/S personal specification is developed for the proposed system.

INPUT & OUTPUT REPRESENTATION

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.
Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

a. Select methods for presenting information.
b. Create document, report, or other formats that contain information produced by the system.

CONCLUSION

In this work, we propose a SVM and Neural network-based malware detection scheme for Android platform, and use both dangerous API calls and risky permission combinations as features to build an SVM classifier and Neural network, which can automatically distinguish malicious Android apps (malware) from legitimate ones. Experiment results show that the proposed scheme is able to identify malware in an accurate manner.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP192-203

FUTURE WORK

The future scope of the project on deep learning-based malicious APK detection is vast and promising, primarily driven by the rapid evolution of mobile malware and the increasing sophistication of cyber threats. Future research could explore more advanced neural network architectures, such as transformer models, which have shown exceptional capabilities in various domains. Additionally, integrating federated learning could enhance privacy by allowing models to be trained on decentralized data sources, mitigating the risk of data breaches. The incorporation of explainable AI (XAI) techniques can make the detection process more transparent, helping cybersecurity professionals understand and trust the model's decisions. Furthermore, expanding the dataset to include a wider variety of malware samples from diverse sources can improve the robustness and generalizability of the detection system. Real-time detection capabilities and the development of lightweight models suitable for deployment on mobile devices are also critical areas for future enhancement, ensuring the approach remains practical and scalable in real-world applications.

REFERENCES

1 DATA Report, ―8,400 new android malware samples every day‖.

2 Ch.-Y. Huang, Y.-T. Tsai, and C-H. Hsu, ―Performance evaluation on permissionbased detection for android malware,‖ Advances in Intelligent Systems and Applications, vol. 2, pp. 111-120, 2013.

3 Zarni, and W. Zaw, ―Permission-based android malware detection,‖ International Journal of Scientific and Technology Research, vol. 2, no. 3, pp. 228-234, 2013.

4 D. Luke, V. Notani, and A. Lakhotia, ―Droidlegacy: Automated familial classification of android malware,‖ In Proc. of the ACM SIGPLAN on Program Protection and Reverse Engineering Workshop, pp. 3, 2014.

5 M. Zhang, Y. Duan, H. Yin, and Z. Zhao, ―Semantics-aware Android malware classification using weighted contextual API dependency graphs,‖ In Proc. of the ACM Conference on Computer and Communications Security (CCS), pp. 1105-1116, 2014.

6 Y. Zhongyang, Z. Xin, B. Mao, L. Xie, ―DroidAlarm: an all-sided static analysis tool for Android privilege-escalation malware,‖ In Proc. of the 8th ACM SIGSAC symposium on Information, computer and munications security, pp. 353-358, 2013.

7 E. Chin, A. P. Felt, K. Greenwood, D. Wagner, ―Analyzing interapplication communication in Android,‖ In Proc. of the international conference on Mobile systems, applications, and services, pp.239-252, 2011.

8 L. Lu, Z. Li, Z. Wu, W. Lee, G. Jiang, ―Chex: statically vetting android apps for component hijacking vulnerabilities,‖ In Proc. of the ACM conference on Computer and communications security, pp. 229-240, 2012.

9 P. PF Chan, L. CK Hui, SM. Yiu, ―Droidchecker: analyzing android applications for capability leak,‖ In Proc. of the ACM conference on Security and Privacy in Wireless and Mobile Networks, pp. 125-136, 2012.

10 K. Lu, Z. Li, V. P. Kemerlis, Z. Wu, L. Lu, ―Checking More and Alerting Less: Detecting Privacy Leakages via Enhanced Data-flow Analysis and Peer Voting,‖ In Proc. of the Network and Distributed System Security Symposium (NDSS), 2015

11 Ijteba Sultana, Dr. Mohd Abdul Bari ,Dr. Sanjay,‖ *Routing Performance Analysis of Infrastructure-less Wireless Networks with Intermediate Bottleneck Nodes*‖, International Journal of Intelligent Systems and Applications in Engineering, ISSN no: 2147-6799 IJISAE,Vol 12 issue 3, 2024, Nov 2023

12 Md. Zainlabuddin, "*Wearable sensor-based edge computing framework for cardiac arrhythmia detection and acute stroke prediction*‖, Journal of Sensor, Volume2023.

13 Md. Zainlabuddin, "*Security Enhancement in Data Propagation for Wireless Network*‖, Journal of Sensor, ISSN: 2237-0722 Vol. 11 No. 4 (2021).

14 Dr MD Zainlabuddin, "*CLUSTER BASED MOBILITY MANAGEMENT ALGORITHMS FOR WIRELESS MESH NETWORKS*‖, Journal of Research Administration, ISSN:1539-1590 | E-ISSN:2573-7104 , Vol. 5 No. 2, (2023)

15 Vaishnavi Lakadaram, " Content Management of Website Using Full Stack Technologies‖, Industrial Engineering Journal, ISSN: 0970-2555 Volume 15 Issue 11 October 2022

16 Dr. Mohammed Abdul Bari,Arul Raj Natraj Rajgopal, Dr.P. Swetha ,‖ *Analysing AWSDevOps CI/CD Serverless Pipeline Lambda Function's Throughput in Relation to Other Solution*‖, International Journal of Intelligent Systems and Applications in Engineering , JISAE, ISSN:2147-6799, Nov 2023, 12(4s), 519–526

17 Ijteba Sultana, Mohd Abdul Bari and Sanjay,‖ *Impact of Intermediate per Nodes on the QoS*

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP192-203

Provision in Wireless Infrastructure less Networks‖, Journal of Physics: Conference Series, Conf. Ser. 1998 012029 , CONSILIO Aug 2021

18 M.A.Bari, Sunjay Kalkal, Shahanawaj Ahamad," *A Comparative Study and Performance A nalysis of Routing Algorithms*‖, in 3rd International Conference ICCIDM, Springer - 978- 981-10-3874-7_3 Dec (2016)

19 Mohammed Rahmat Ali,: BIOMETRIC: AN e-AUTHENTICATION SYSTEM TRENDS AND FUTURE APLLICATION‖, International Journal of Scientific Research in Engineering (IJSRE), Volume1, Issue 7, July 2017

20 Joshua, S.C.; Garber, N.J. Estimating Truck Accident Rate and Involvements Using Linear and Poisson Regression Models. Transp. Plan. Technol. 1990, 15, 41–58. [CrossRef]

21 Mohammed Rahmat Ali,: BYOD A systematic approach for analyzing and visualizing the type of data and information breaches with cyber security‖, NEUROQUANTOLOGY, Volume20, Issue 15, November 2022

22 Mohammed Rahmat Ali, Computer Forensics -An Introduction of New Face to the Digital World, International Journal on Recent and Innovation Trends in Computing and Communication, ISSN: 2321- 8169-453 – 456, Volume: 5 Issue: 7

23 Mohammed Rahmat Ali, Digital Forensics and Artificial Intelligence ...A Study, International Journal of Innovative Science and Research Technology, ISSN:24562165, Volume: 5 Issue:12.

24 Mohammed Rahmat Ali, Usage of Technology in Small and Medium Scale Business, International Journal of Advanced Research in Science & Technology (IJARST), ISSN:2581-9429, Volume: 7 Issue:1, July 2020.

25 Mohammed Rahmat Ali, Internet of Things (IOT) Basics - An Introduction to the New Digital World, International Journal on Recent and Innovation Trends in Computing and Communication, ISSN: 2321- 8169-32-36, Volume: 5 Issue: 10

26 Mohammed Rahmat Ali, Internet of things (IOT) and information retrieval: an introduction, International Journal of Engineering and Innovative Technology (IJEIT), ISSN: 2277-3754, Volume: 7 Issue: 4, October 2017.

27 Mohammed Rahmat Ali, How Internet of Things (IOT) Will Affect the Future - A Study, International Journal on Future Revolution in Computer Science & Communication Engineering, ISSN: 2454-424874 – 77, Volume: 3 Issue: 10, October 2017.

28 Mohammed Rahmat Ali, ECO Friendly Advancements in computer Science Engineering and Technology, International Journal on Scientific Research in Engineering(IJSRE), Volume: 1 Issue: 1, January 2017

29 Ijteba Sultana, Dr. Mohd Abdul Bari ,Dr. Sanjay, —*Routing Quality of Service for Multipath Manets, International Journal of Intelligent Systems and Applications in Engineering"*, JISAE, ISSN:2147-6799, 2024, 12(5s), 08–16;

30 *Mr. Pathan Ahmed Khan, Dr. M.A Bari,: Impact Of Emergence With Robotics At Educational Institution And Emerging Challenges", International Journal of Multidisciplinary Engineering in Current Research(IJMEC), ISSN: 2456-4265, Volume 6, Issue 12, December 2021,Page 43-46*

31 Shahanawaj Ahamad, Mohammed Abdul Bari, Big Data Processing Model for Smart City Design: A Systematic Review —, VOL 2021: ISSUE 08 IS SN : 0011-9342 ;Design Engineering (Toronto) Elsevier SCI Oct : 021

32 Syed Shehriyar Ali, Mohammed Sarfaraz Shaikh, Syed Safi Uddin, Dr. Mohammed Abdul Bari, —Saas Product Comparison and Reviews Using Nlp‖, Journal of Engineering Science (JES), ISSN NO:0377-9254, Vol 13, Issue 05, MAY/2022

33 *Mohammed Abdul Bari, Shahanawaj Ahamad, Mohammed Rahmat Ali," Smartphone Security and Protection Practices", International Journal of Engineering and Applied Computer Science (IJEACS) ; ISBN: 9798799755577 Volume: 03, Issue: 01, December 2021 (International Journal,U K) Pages 1-6*

34 .A.Bari& Shahanawaj Ahamad, —Managing Knowledge in Development of Agile Software‖, in International Journal of Advanced Computer Science & Applications (IJACSA), ISSN: 2156-5570, Vol: 2, No: 4, pp: 72-76, New York, U.S.A., April 2011

35 Imreena Ali (Ph.D), Naila Fathima, Prof. P.V.Sudha ,—Deep Learning for Large-Scale Traffic-Sign Detection and Recognition‖, Journal of Chemical Health Risks, ISSN:2251-6727/ JCHR (2023) 13(3), 1238-1253

36 Imreena, Mohammed Ahmed Hussain, Mohammed Waseem Akram‖ An Automatic Advisor for Refactoring Software Clones Based on Machine Learning‖, Mathematical Statistician and Engineering ApplicationsVol. 72 No. 1 (2023)

37 Mrs Imreena Ali Rubeena,Qudsiya Fatima Fatimunisa —Pay as You Decrypt Using FEPOD Scheme and Blockchain‖, Mathematical Statistician and Engineering Applications: https://doi.org/10.17762/msea.v72i1.2369 Vol. 72

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP192-203

No. 1 (2023)

38 Imreena Ali , Vishnuvardhan, B.Sudhakar,‖ Proficient Caching Intended For Virtual Machines In Cloud Computing‖, International Journal Of Reviews On Recent Electronics And Computer Science , ISSN 2321- 5461,IJRRECS/October 2013/Volume-1/Issue-6/1481-1486

39 S. Hao, B. Liu, S. Nath, W. G. Halfond, and R. Govindan, ―PUMA: Programmable UI-automation for Large-scale Dynamic Analysis of Mobile Apps,‖ In Proc. of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys), pp. 204-217, 2014.

40 Heena Yasmin, A Systematic Approach for Authentic and Integrity of Dissemination Data in Networks by Using Secure DiDrip, INTERNATIONAL JOURNAL OF PROFESSIONAL ENGINEERING STUDIES, Volume VI /Issue 5 / SEP 2016

41 Heena Yasmin, Cyber-Attack Detection in a Network, Mathematical Statistician and Engineering Applications, ISSN:209 4-0343, Vol.72 No.1(2023)

42 Heena Yasmin, Emerging Continuous Integration Continuous Delivery (CI/CD) For Small Teams, Mathematical Statistician and Engineering Applications, ISSN:20940343, Vol.72 No.1(2023)

43 W. Zhou, Y. Zhou, X. Jiang, P. Ning, ―Detecting repackaged smartphone applications in third-party android marketplaces,‖ In Proc. of the ACM conference on Data and Application Security and Privacy, pp. 317-326, 2012.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP192-203