**ANDROID MALWARE CHARACTERIZATION AND DETECTION USING DEEP LEARNING**

**ABSTRACT**

Now a day’s mobile phones and tablets are becoming integrated part of our life. One of the most important operating system for these devices is android. But, due to open nature of android, thousands of bugs are hidden in more number of applications that seriously effects its security and performance. This can be avoided by using Deep learning, which is new concept in machine learning. In this concept, we are combing the characteristics of static analysis with that of dynamic analysis of android apps and characterize the bug using this technique. In this technique, we are implementing an online android malware detection engine which can automatically detect whether the app is malware or not. Previously by using deep learning technique thousands of android apps has been tested in order characterize malware. This can achieve 95% of detection accuracy. More number of popular anti-virus software has been evaluated which shows the urgency in malware detection.

**INTRODUCTION**

**Problem statement:**

My problem statement in this project is how to protect our open source android devices from malware and how to achieve proper security and how to increase performance of our android device.

**Objectives:**

The main objective of this project is to design an application to protect our open source android from malware and to secure our data from malicious theft and to avoid hanging of our phone by adding android malware detection engine that can automatically detect whether the app has malware or not.

**Research questions:**

How to avoid these malware attacks and protect our data?

How to make android as secured open source?

How to reduce thefts and personal information of the users?

How to make filtered applications in android?

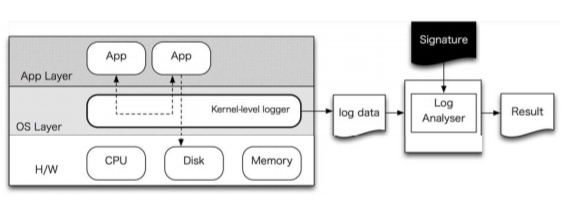
How to trust an application weather it contains malware or not.

**Literature survey**

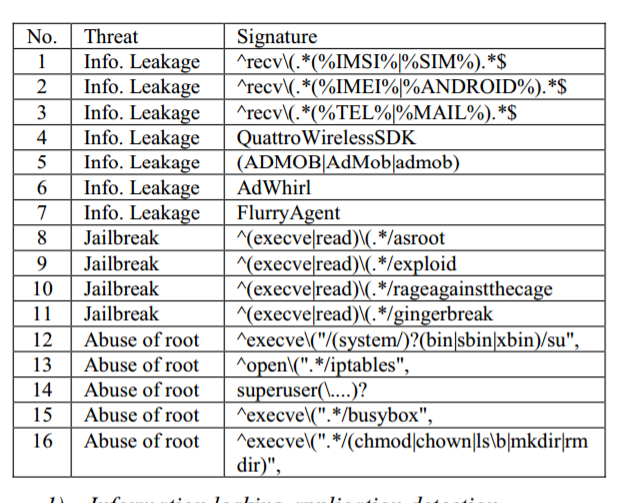
**Malware detection:**

The author Takamasa isohara published a paper in computational intelligence and security 2011 in order to avoid malicious theft of personal information by some of the malware applications like geimini and droid dream. In this they have used a frame work called log cat which is mainly used to monitor the behavior of android applications because it is important to know behavior of all apps for security reasons. After that they have proposed a system, which contains log collector and log analyzer. Log collector is used to notice all calls related to systems and it is acts as a filter for the events conducted by target applications. log analyzer is mainly used to match the activities of all applications with signatures that are described by regular expressions in order to detect the malware activity. After that they have implemented a prototype and conducted experiments with more than 230 applications. The overall result is effective only up to some extent. After geimini more than 50 malwares has been detected by android system and this prototype failed to detect the characteristics of that malwares[1].

The system architecture for his study is as fallows

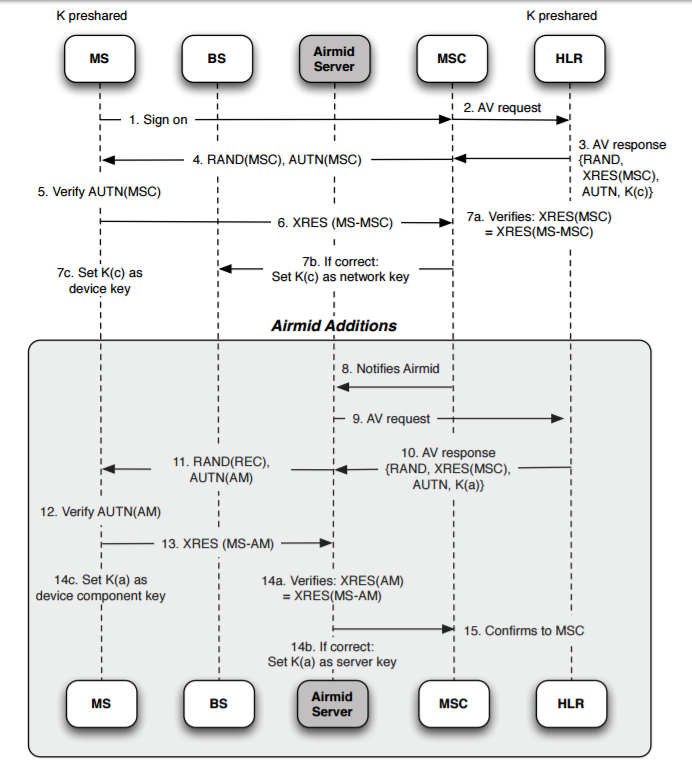


The signatures for his malware study is as fallows

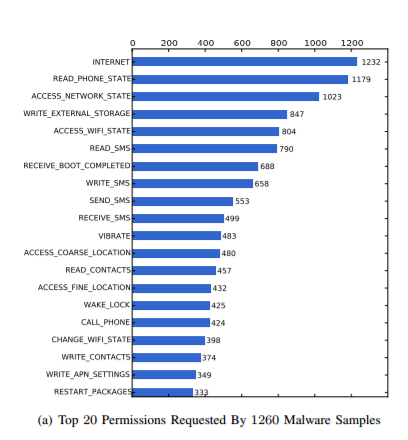


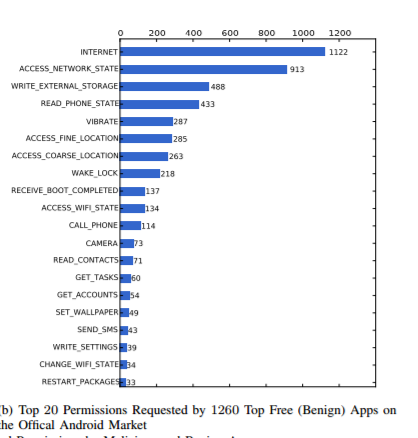
After that Nadji,Giffin and traynor conducted a research on same thing .They have developed w an infrastructure which will detects the viruses and bugs in an application based on their network behavior. In this we developed an prototype airmid ,which uses the cooperation between in network sensors and smart devices to identify the malware traffic. After that we have developed a simple malicious application which is exceeding the malware capabilities that are considered as serious threat to our devices .Airmid is mainly used to show a range of automated recovery responses starting from on device firewall to application removal[2].

The authentication flow for his work is as below



After that similar work, has done by Zhou and jiang nd conducted work on more than 1200 samples. After that we analyzed them in every possible aspect including their installation and activation methods in order to know the exact nature of malware payload. The subsequent evaluation based study of these malware families reveal that they are evolving rapidly and the need of mobile anti-virus software in today’s world. Based on the evolution with four mobilerepresentative security software, our analysis shows that the worst case detects only 20.2% and our best case detects 79.6% of our dataset. These analyses clearly shows the need for development of anti-mobile malware solutions for our future generations[3].

His related graphs for top 20 malwares is as fallows

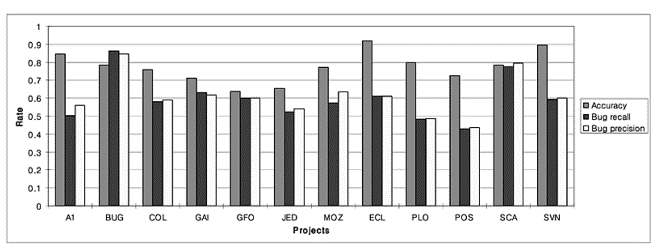


In most of the papers all the authors try to avoid bugs by concentrating only on source code to predict but they are not concentrating on other properties like log , metadata ,file .some of the papers which concentrate on only bugs are as fallows

Gyimothy et al model has used only source code to avoid bugs and to characterize bugs with class level of granularity. While examining the source code, the granularity associated with that code is very high[4].

Brun and Ernst proposed a model only to find the hidden code errors[5].

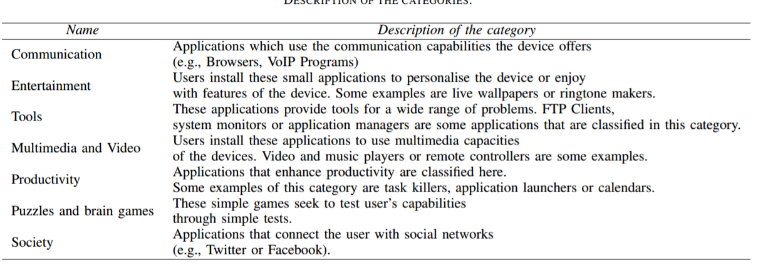
After that Sunghun , Whitehead , proposed better work rather than previous 2 papers by using change classification. It concentrates on all properties like metadata. But this too fail for some software changes because it gives maximum of only 70% accuracy. After that he characterize malware as[6]



**Android malware characterization**

Heloise pieterse and martin proposed a paper in order to characterize the android malware and to avoid botnets from our android mobiles. some of the malwares like Geinimi , RootSmart shows all botnet characteristics. So it is important to identify and study all the characteristics related to this botnet functionalities and this can be identified by comprehensive literature study of most popular android malicious applications. All identified features of these botnets are studied in terms of android botnet discovery process and also its development model. All these identified features and characteristics will helps us to identify all functionalities of android botnet[7].

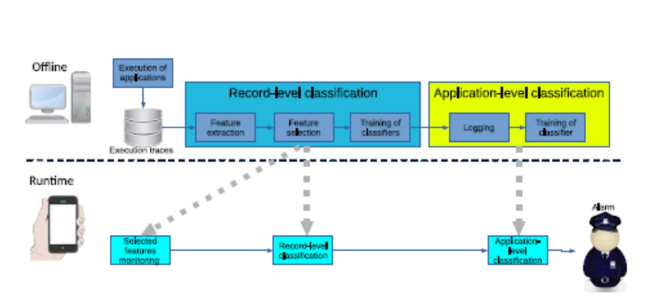
A related work has been done by Sanz and santos in order to characterize android malware [8]. They proposed an approach which is used to characterize the android malware application through machine learning. In this we will extract the fallowing details from all android applications. It will extract all the permissions associated with the particular application along with its permissions from android market. This approach will automatically characterize the android malware[9].

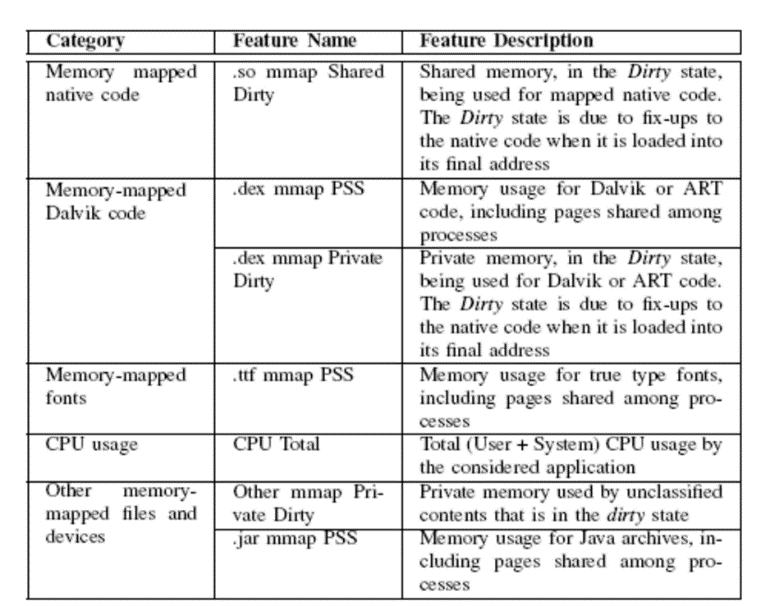


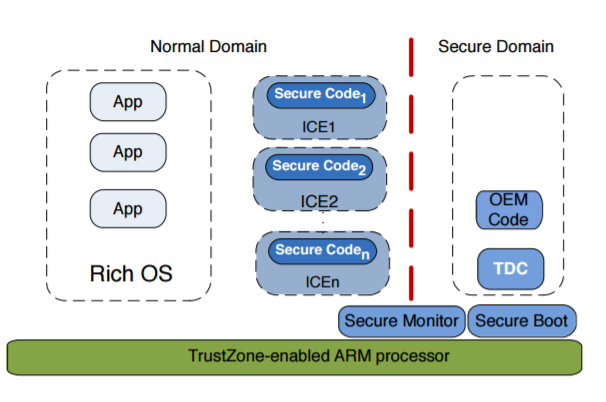
There is a tremendous increase in the usage of android in a span of 3 years the sales are tripled from 40 million to 120 million in last three years. There is also increase in 400% of malware during 2010. This is need to be avoided inorder to make android a trusted platform.

The goals and contribution of this paper are three- firstly we need to characterize android malware from 49 different android families and these are collected. Secondly after collection of malwares, these are made with time line analysis and their characterization is observed depending on the break down including installation, activations and payloads. Thirdly an evolution based study is made which shows antivirus methods are not working properly. It is proposed by Yazin joh and Xuxian [10].

Effective characterization of malware is very important for securing the upcoming generation of smart devices. Static detection, which is the best preferred technique which is being used, this is not powerful one to defeat malware, which is even less effective in the near future. Dynamic malware characterization guarantees better protection since it operates at run-time and can identify also unknown malware[11]. He uses fallowing architecture to characterize this



He has extracted fallowing features 

Mobile devices perform important transactions. Always to protect secure code from a malicious mobile OS is a big challenge. ARM TrustZone is a new technology that can protect secure code in protected domain from a domain which is untrusted. Since the attack surface of the secure domain will increase along with the size of secure code, it becomes arduous to negotiate with OEMs to get new secure code installed. It is proposed by He Sun and Kun Sun[12]. They uses fallowing architecture to characterize this****

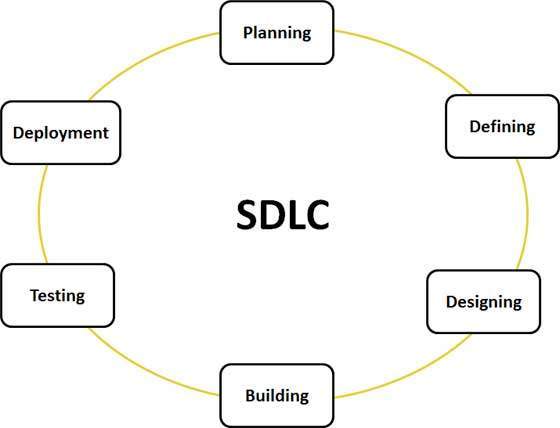
**METHODOLOGY**

Methodology is nothing but sequence of steps and set of methods/operations used in a project in order to complete our project. The main goal of these methodologies is to make the process more predictable and efficient[5]. In this project, we are using LINEAR-SEQUENTIAL LIFE CYCLE MODELin software development life cycle

**Software Development Life Cycle :**

Software development life cycle is used mostly in all software projects in order to keep track of design of project, development of project and finally to test the software for reliable output.

It is nothing but a structure followed by team in a project. software development life cycle contains the detailed plan of project like how to develop a specific software, How to maintain it and how to replace it. The simple snap of SDLC is as shown below[1]



There are different models in SDLC like Waterfall model, V-Shaped model, Big bang model , Spiral model and Incremental model.

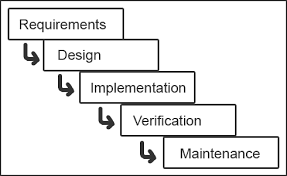
In these we considered only the basic model, that is waterfall model. The detailed explanation for this is as shown below

**Waterfall Model :**

It is also known as linear sequential life cycle model. It is the basic model in SDLC. Each phase must be completed before the next phase can begin and there is no overlapping in the phases[2].

This model explains the SDLC in a linear fashion. Due to this it is also known as linear sequential life cycle model. Waterfall model is mainly used, when the project is small and also changes in the project is less [3]

There are mainly 5 phases in waterfall model. They are Requirements, design, implementation, verification and maintenance.[4]



The different phases of waterfall model is explained as fallows.

**1.Planning:**

It is the first step in SDLC. In this step we have to think about what is the aim of our project, what is the need of that project and also about all the resources required to our project. After that we have to analyses the cost and benefits.

The main aim of our project is to detect the malware in android mobile device and how to remove it. The need in this is most of our data has been stolen from our mobile devices without our knowledge due to open nature of android software. So, my aim is to protect our device from that type of data sharing.

**2.System Analysis and Requirements:**

In this phase, all the requirements for end user are decided by our team. In this we have to examine all the functional requirements. Here we will focus on problems, need for the solution and team has to analyses the best solution for that problem. The programmer has to understand the clear picture of what are the business needs and how can we meet them. The programmer is responsible to assign work to remaining team mates.

**3.System Design:**

This step illustrates the activities, features, operations that fulfill the functional requirements and the requisite descriptions in depth. Here, in this step the end user studies their needs for their proposed system. The important elements of hardware and software are observed.

**Class Diagram:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information. In class diagram attributes and operations which are involved are included and given in the diagrammatic format[6].



**Use Case Diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**Sequence Diagram:**

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. This object interaction allows the messages to be performed in a sequence among the different objects[7].

**4.Developement:**

It is the phase our application development starts. In this step, we will implement our project that is all the theoretical work we have done so far is converted in to program. Here we need all tools and design specifications to complete this phase successfully. At this phase programmers, and database developers are needed for building testing and preparing technical surroundings for the system.

**5. Integration & Testing**:

After the completion of program development, we have test that program to avoid all errors and also to ensure that the application is working properly. The system is tested for integration with previous model which is needed to communicate. This testing process is conducted repeatedly until we have removed all errors. Here the application is verified for successful completion. In our project we are going to do fallowing testings

***Unit Testing:***

Unit testing is conducted usually as part of a combined code and unit phase test of the software lifecycle. The main objectives of unit testing is to be familiar with all the field entries are working properly or not and all the pages must be activated from the identified link and finally, we have to make sure that the entry screen, input and output responses must not be delayed.

Features to be tested

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# *Integration Testing*

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications.

***Acceptance Testing***

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. In this step end user has to be satisfied. Until then we have to make changes to our application up to user gets satisfied

**6.Implementation:**

In this phase, we are placing our newly created system into target environment and it is the basic key of this phase and it can be done by cut over. This is somewhat risky. Cut over occurs only in peak timings and thus it minimizes the risk. Training on system development is the best way for good implementation.

**7. Operations and Maintenance**

Operation and maintenance is the final phase in SDLC. This phase mainly requires maintenance and updates for our newly created system. We have to monitor the system performances like backup, planning, audit, and contingency.

**Advantages of waterfall model:**

1. It is simple to use and easy to understand
2. It is the best model for small projects and different phases of the project are completed in time
3. In this model all phases are documented well and so that it is easy to refer
4. Water fall model is easy to manage due to its rigidity

**Disadvantages of waterfall model:**

1. It is not good for complex and larger projects
2. In this model, we cannot go back a step to previous phase because things get complicated in implementation phase
3. If client realizes the need of more specifications and demand change which can impact the budget of the project and product with quality assured product.

It does not allows us to change the work, which can arise in the lots of rework, because of late testing there can be unpredictable software quality.

**CONCLUSION**

By studying both android malware detection and malware characterization, if we design an app by clubbing both the features we can protect our android device and can avoid malware attacks and hanging of our mobile. This system mainly focuses on data security and to increase our mobile performance. This cannot be avoided totally up to 100%. But we will protect our devices up to some extent. Studies are still going on to reduce this problem completely.

**FUTURE WORKS:**

In this we are characterizing only the static and dynamic behavior of applications .In future we are trying to update this on the basis of all permissions , API calls etc by after downloading an application to our device, based on the code of app we are trying to get all the bugs and after that we have to inform weather the app is risky or not.

We are trying to build even an IOS app for this.

**REFERENCES:**

[1]. <http://ieeexplore.ieee.org/abstract/document/6128277/?section=abstract>

[2]. <http://www.cc.gatech.edu/~ynadji3/docs/pubs/repair11.pdf>

[3]. <http://nieh.net/teaching/e6998/papers/OAKLAND12.pdf>

[4]. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3853189/>

[5]. <http://ieeexplore.ieee.org/document/5359599/>

[6] http://ieeexplore.ieee.org/document/4408585/

[7]. <http://ieeexplore.ieee.org/document/6320432/>

[8].<http://paginaspersonales.deusto.es/claorden/publications/2012/Sanz_2012_CCNC_Android_Apps_Categorisation.pdf>

[9]. <http://ieeexplore.ieee.org/document/6181075/>

[10]. <http://ieeexplore.ieee.org/document/6234407/citations?tabFilter=papers>

[11]. http://ieeexplore.ieee.org/document/7588859/

[12]. <http://ieeexplore.ieee.org/document/7266865/>