Detecting Software Vulnerabilities in Android Using Static Analysis

R.Dhaya[1] ,M.Poongodi[2]
[1]Faculty, [2]Student,
[1,2]Dept. of CSE, Velammal Engg. College,Chennai
[1]dhayavel@yahoo.co.in,[2]poongs.swamy@gmail.com

***Abstract-* Now a day's mobile devices like Smartphone, tablets and Personal Digital Assistants etc. were playing most essential part in our daily lives. A high-end mobile device performs the same functionality as computers. Android based smart phone has become more vulnerable, because of an open source operating system. Anyone can develop a new application and post it into android market. These types of applications were not verified by authorized company. So it may include malevolent applications it may be virus, spyware, worms, etc. which can cause system failure, wasting memory resources, corrupting data, stealing personal information and also increases the maintenance cost. Due to these reasons, the mobile phone security or mobile security is very essential one in mobile computing. In the existing system is not able to detect new viruses, due to the limitation of updated signatures. The proposed system aims to motivate static code analysis based malware detection using search based machine learning algorithm which is called N-gram analysis and it detects the unnoticed malicious characteristics or vulnerabilities in the mobile applications.**

Keywords - Malware, Android, Static Analysis, N-Gram, SVM, Vulnerability, CVSS

I. INTRODUCTION

Mobile devices such as Smartphone, tablets and Personal Digital Assistant are well liked gadget in recent years. Once a mobile device grows simultaneously their functional and technical complexity also increases. Latest Smartphone proffers a lot of services and applications than services gave by the computers. Most of the data transmission like Net banking, online shopping etc. will be done through mobile phones i.e. smart phones. The result of the mobile phone usage is to raise the criminals to get the benefit of these activities for unlawful gains. So data should be secured from any kind of electronic attacks. A mobile malware like virus, worms, Trojan etc is a malicious software program that aims to damage the mobile devices such as tablets, smart phones, Personal Digital Assistants, etc. Malwares broadcast via Short Messaging Service (SMS), Multimedia Messaging Service (MMS), and Bluetooth etc. Mobile viruses can spread from PC networks into mobile networks or vice versa. Because of this mobile security is vital one for the above. The major part of the mobile companies is using the Android Platform. Android platform is becoming very popular today and it is getting more vulnerable because it is an open source and easy to develop the applications freely. A Malware developer takes this one as an advantage to write malware programs. Because of these malwares, Android based Smart phones are easily attacked and it performs malicious activities such as theft the sensitive data, drain the battery without user's knowledge. Different security countermeasures are being developed and applied to Smartphone to mitigate the security threats. This paper illustrates N-gram and static analysis based malware prevention techniques for Android based mobile phones. This method is used to categorize the malware/benign mobile applications.

Many antivirus companies uses signature based detection algorithm but is not able to identify new viruses. So we designed a software program that uses machine learning based algorithm (N-gram) to detect the given mobile applications is having malware or not. For this we divided our dataset into two types.1.Training set – was used by N-gram to create a classifier to classify in the past unknown features of source code as malware or benign. 2. Test set- is an element of dataset that does not have any instances of previous one which are trained by N-gram analysis. Test set is used to check the performance and accuracy of the algorithm over unnoticed instances.

The rest of the paper is sorted as follows. A section 2 talk about an overview of the proposed system and Section 3 describes the implementation of proposed system. Section 4 describes the analysis part of the proposed system and section 5 contains the conclusion of the proposed system.

II. SYSTEM OVERVIEW

A. Existing System

Most of the commercial antivirus companies used a signature based detection methods to identify malwares. Signature is the binary of pattern of the machine code of a particular virus [1]. It may be strings; binaries. It checks the content of the file dictionary of malware signatures [3]. This method lists the following disadvantages,

ISBN No. 978-1-4799-3914-5/14/$31.00 ©2014 IEEE

- It needs the huge database to store the malware signatures [2].
- It fails to identify an unknown malwares because a new malware may not contain a known signature of malwares [4].

B. Objective of the Proposed System

This paper describes the static or source code analysis of an android application package files. Static approach assists to recognize the vulnerabilities such as SQL injection, Data Manipulation Language (DML), Password, Cookie poisoning, etc and join together with the software and evaluate the complexity necessitated in it using Common Vulnerability Scoring System. Because of this we are finding bugs in the mobile applications, before delivering it to the customer and detecting new vulnerabilities also. Previous system supports signatures (character strings, binary) to detect malwares. Our system considers signatures are Source code for this we are applying N-gram concept and these signatures are used to find the new security vulnerabilities in the applications.

Main Concepts are used in proposed system. They are,

- Source code analysis or Static analysis

It is used to pre checks the code to find bug in the application before it becomes to an issue [5].

- N-gram analysis

It is a one type of probabilistic model; it uses the concept of Markov chain model. N-gram is a contiguous sequence of n times from a given sequence of input data and it is used to predict the next item [10].

Advantages of proposed system

- Quick and proactive
- Cheap in economical wise, Flexible and easily automated

C. 2.3 Proposed System Architecture Diagram

This paper mainly focuses on identification of malwares in mobile phone application using search based malware detection algorithm namely, N-gram analysis. If malware is present in any mobile application, it performs malicious activities like sending SMS to address book, etc. Static code analysis is used to reverse the code without executing the android application file (apk) and convert it into the source code of the file and extracting the features of the application and use the machine learning tool to create the learning model database which is based on malware and benign applications to classify the malwares. We need to check whether the given test application is having malware or not. For this, compare the N-gram signatures of the test application and signature which is already stored in the files. The result will be the malicious or benign code of the given test application.

The proposed system explains the steps in the following block diagram shown in figure 1.

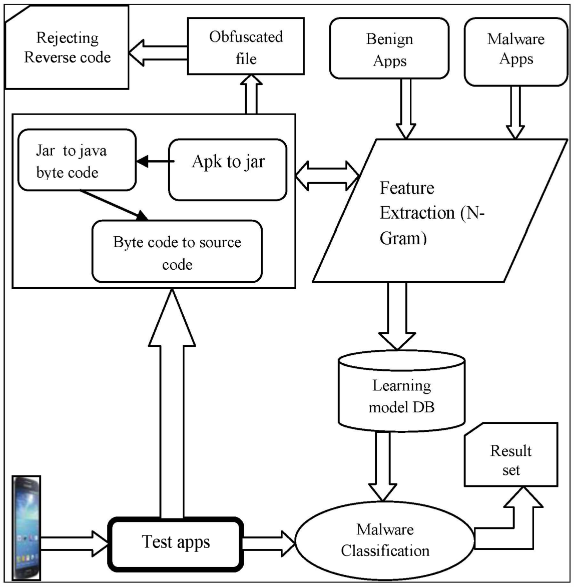


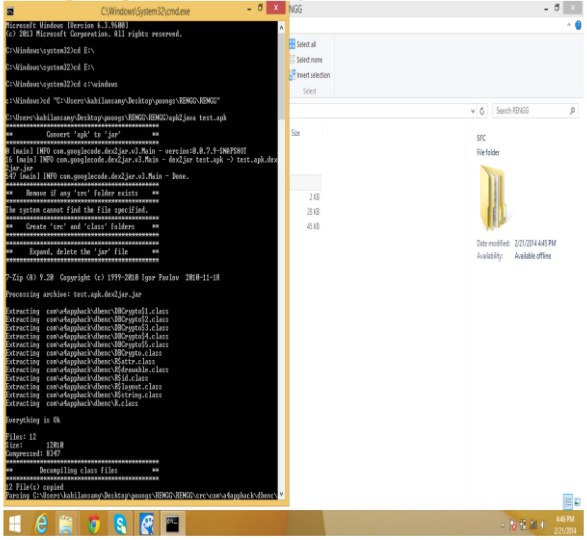
Figure1 :Architecture Diagram

III. IMPLEMENTATION

The proposed system contains four modules. They are followed by,

A. Reverse Engineering Automation

Reverse engineering is a method which is used to analyze an existing code in order to inspect the vulnerabilities or malicious characteristics in the software. This method has the capability to create source code from an executable files. Many tools are used to perform the reverse engineering and static analysis methods in android mobile applications to find the malwares. Android application package files (.apk) are not allowed to execute in Java Virtual Machine (JVM), So Dalvik Virtual Machine (DVM) executes the applications in dalvik executable format (.dex). Classes.dex is a main component in android applications which is not able to view.Dex2jar is a tool which is used to convert dex files into .class file format. To convert the apk file with jar file using dex2jar tool [6] these file are converted into byte codes. It will be decompiled by JD-GUI. Java Decompiler-Graphical User Interface (JD-GUI) is a tool [8] to get the source code of the application as a java code. Apktool is used to decompile [7] and recompile the apk files. It contains manifest.xml file which includes the needed permissions of an androidapplications.Run a batch file to get the source code. Batch file is a set of commands which is used to perform a repeated task. Apk2java.bat filename.apk is used [9] to perform the reverse engineering process in the command prompt is shown in figure 2.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd E:\

C:\Windows\system32>cd E:\

C:\Windows\system32>cd c:\windows

c:\Windows>cd "C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG"

C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG>apk2java test.apk
*********************************************
**          Convert 'apk' to 'jar'         **
*********************************************
0 [main] INFO com.googlecode.dex2jar.v3.Main - version:0.0.7.9-SNAPSHOT
16 [main] INFO com.googlecode.dex2jar.v3.Main - dex2jar test.apk -> test.apk.dex
2jar.jar
547 [main] INFO com.googlecode.dex2jar.v3.Main - Done.
*********************************************
**    Remove if any 'src' folder exists    **
*********************************************
The system cannot find the file specified.
*********************************************
**    Create 'src' and 'class' folders     **
*********************************************
*********************************************
**     Expand, delete the 'jar' file       **
*********************************************

7-Zip (A) 9.20  Copyright (c) 1999-2010 Igor Pavlov  2010-11-18

Processing archive: test.apk.dex2jar.jar

Extracting  com\a4apphack\dbenc\DBCrypto$1.class
Extracting  com\a4apphack\dbenc\DBCrypto$2.class
Extracting  com\a4apphack\dbenc\DBCrypto$3.class
Extracting  com\a4apphack\dbenc\DBCrypto$4.class
Extracting  com\a4apphack\dbenc\DBCrypto$5.class
Extracting  com\a4apphack\dbenc\DBCrypto.class
Extracting  com\a4apphack\dbenc\R$attr.class
Extracting  com\a4apphack\dbenc\R$drawable.class
Extracting  com\a4apphack\dbenc\R$id.class
Extracting  com\a4apphack\dbenc\R$layout.class
Extracting  com\a4apphack\dbenc\R$string.class
Extracting  com\a4apphack\dbenc\R.class

Everything is Ok

Files: 12
Size:       12010
Compressed: 8347
*********************************************
**        Decompiling class files          **
*********************************************
12 File(s) copied
Parsing C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\src\com\a4apphack\dbenc\
```

Figure 2: apk2source Conversion

B. *Benign/Malware Feature Extraction*

Training set was created based on Benign/Malware application's features like API call, Call flow, Device memory are extracted from android packages and Open Web Application Security Project (OWASP), torrent. The training set contains the vulnerabilities like SQL injection, Password, DML, Print stacktrace and its feature vectors. It is gathered from the public resources. After extracting the source code, N-Gram is applied in the extracted features. Instead of text or strings, source code is considered as N-Grams signatures. The following table 1 lists the example of vulnerabilities and extracted features in android.

TABLE 1: EXTRACTED FEATURES

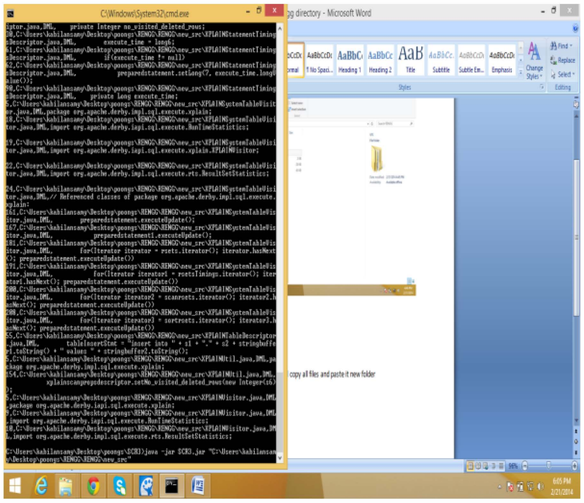
Vulnerabilities	Feature Vectors

Connection_String_Injection	CxList con = All.FindByName ("*get Connection");
	CxList inputs = Find_Interactive_Inputs();
	CxList sanitize = Find_General_Sanitize() + Find_Integers();
	Result= con.Influenced ByAndNotSanitized (inputs, sanitize);

SQL_Injection	CxList db = Find_DB();
	CxList read = Find_Read_NonDB();
	CxList outputs = Find_XSS_Outputs();
	CxList sanitize = Find_XSS_Sanitize();
	result = All.FindXSS (db + read, outputs, sanitize);

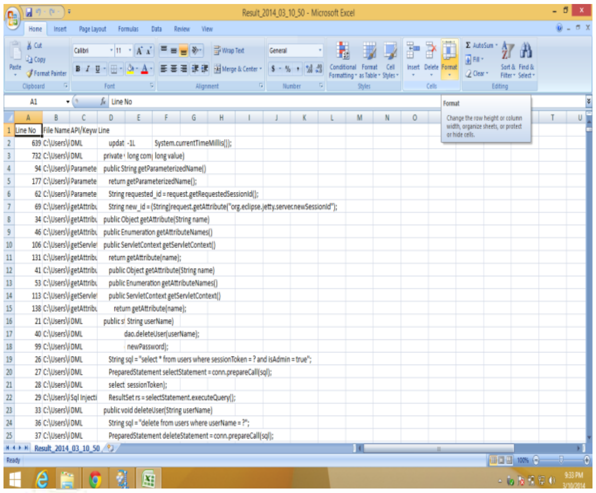
C. *Constructing File Database and classifying the malware*

N-gram source codes signatures are not able to store the sqllite database or other databases because huge space is need to store N-gram signatures of source code. So we are maintaining these signatures as Comma Separated value (CSV) file. A CSV file is any file containing text that is separated with a comma, or any other character.Apk2java command is used to get the all java code and it is running from the scratch. Create a java file and compile it .Using this create a jar file and run a Java Archive (JAR) file for source code review using java –jar filename.jar command. After executing this command one CSV file is created in the folder .In that file marked programs have some issues and check it. Reverse engineering process will create the N-gram for test set application and it is used to extract the most frequent n-gram signatures in the given database. It will be classify the application based on database files whether it is malware or benign applications. After executing above all steps and take the marked java file and import it using eclipse and check the source code has the issues or not is shown in figure 3 and figure 4..



```
C:\Windows\System32\cmd.exe
iptor.java,DML,    private Integer no_visited_deleted_rows;
30,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINStatementTiming
sDescriptor.java,DML,        execute_time = long6;
61,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINStatementTiming
sDescriptor.java,DML,        if(execute_time != null)
62,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINStatementTiming
sDescriptor.java,DML,            preparedstatement.setLong(7, execute_time.longV
alue());
98,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINStatementTiming
sDescriptor.java,DML,    private Long execute_time;
5,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVisit
or.java,DML,package org.apache.derby.impl.sql.execute.xplain;
18,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVisi
tor.java,DML,import org.apache.derby.iapi.sql.execute.RunTimeStatistics;

19,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVisi
tor.java,DML,import org.apache.derby.iapi.sql.execute.xplain.XPLAINVisitor;

22,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVisi
tor.java,DML,import org.apache.derby.impl.sql.execute.rts.ResultSetStatistics;

24,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVisi
tor.java,DML,// Referenced classes of package org.apache.derby.impl.sql.execute.
xplain:
161,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVis
itor.java,DML,        preparedstatement.executeUpdate();
167,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVis
itor.java,DML,            preparedstatement1.executeUpdate();
181,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVis
itor.java,DML,        for(Iterator iterator = rsets.iterator(); iterator.hasNext
(); preparedstatement.executeUpdate())
191,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVis
itor.java,DML,            for(Iterator iterator1 = rsetsTimings.iterator(); iter
ator1.hasNext(); preparedstatement.executeUpdate())
200,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVis
itor.java,DML,        for(Iterator iterator2 = scanrsets.iterator(); iterator2.h
asNext(); preparedstatement.executeUpdate())
208,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINSystemTableVis
itor.java,DML,        for(Iterator iterator3 = sortrsets.iterator(); iterator3.h
asNext(); preparedstatement.executeUpdate())
55,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINTableDescriptor
.java,DML,        tableInsertStmt = "insert into " + s1 + "." + s2 + stringbuffe
r1.toString() + " values " + stringbuffer2.toString();
5,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINUtil.java,DML,pa
ckage org.apache.derby.impl.sql.execute.xplain;
154,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINUtil.java,DML,
            xplainscanpropsdescriptor.setNo_visited_deleted_rows(new Integer(s6)
);
5,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINVisitor.java,DML
,package org.apache.derby.iapi.sql.execute.xplain;
9,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINVisitor.java,DML
,import org.apache.derby.iapi.sql.execute.RunTimeStatistics;
10,C:\Users\kabilansamy\Desktop\poongs\REMGG\REMGG\new_src\XPLAINVisitor.java,DM
L,import org.apache.derby.impl.sql.execute.rts.ResultSetStatistics;

C:\Users\kabilansamy\Desktop\poongs\SCR3>java -jar SCR3.jar "C:\Users\kabilansam
y\Desktop\poongs\REMGG\REMGG\new_src"
```

Figure 3: Execution of Source Code Review file



A1 Line No

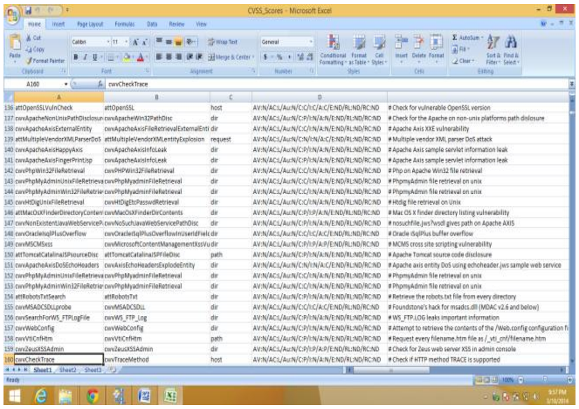
	A	B	C	D	E	F
1	Line No	File Name	API/Keyw	Line		
2	639	C:\Users\	DML	updat	-1L	System.currentTimeMillis());
3	732	C:\Users\	DML	private ·	long com	long value)
4	94	C:\Users\	Paramete	public String getParameterizedName()		
5	177	C:\Users\	Paramete	return getParameterizedName();		
6	62	C:\Users\	Paramete	String requested_id = request.getRequestedSessionId();		
7	69	C:\Users\	getAttribu	String new_id = (String)request.getAttribute("org.eclipse.jetty.server.newSessionId");		
8	34	C:\Users\	getAttribu	public Object getAttribute(String name)		
9	46	C:\Users\	getAttribu	public Enumeration getAttributeNames()		
10	106	C:\Users\	getServlet	public ServletContext getServletContext()		
11	131	C:\Users\	getAttribu	return getAttribute(name);		
12	41	C:\Users\	getAttribu	public Object getAttribute(String name)		
13	53	C:\Users\	getAttribu	public Enumeration getAttributeNames()		
14	113	C:\Users\	getServlet	public ServletContext getServletContext()		
15	138	C:\Users\	getAttribu	return getAttribute(name);		
16	21	C:\Users\	DML	public s	String userName)	
17	40	C:\Users\	DML		dao.deleteUser(userName);	
18	99	C:\Users\	DML		newPassword);	
19	26	C:\Users\	DML	String sql = "select * from users where sessionToken = ? and isAdmin = true";		
20	27	C:\Users\	DML	PreparedStatement selectStatement = conn.prepareCall(sql);		
21	28	C:\Users\	DML	select	sessionToken);	
22	29	C:\Users\	Sql Injecti	ResultSet rs = selectStatement.executeQuery();		
23	33	C:\Users\	DML	public void deleteUser(String userName)		
24	36	C:\Users\	DML	String sql = "delete from users where userName = ?";		
25	37	C:\Users\	DML	PreparedStatement deleteStatement = conn.prepareCall(sql);		

Result_2014_03_10_50

Figure 4 :Result of the apk file

IV. PERFORMANCE EVALUATION

Common Vulnerability Scoring System (CVSS) is a tool which is used to assess the vulnerabilities' severity level in software applications and it will be available in the internet and use it freely. This method uses the [12] CVSS to check the severity levels of vulnerabilities in the android application package files which is shown in figure 5. To find the vulnerabilities in the android environment, source code analysis is used. After using this take the corrective action against vulnerabilities in the apk files to reduce the malware activities in the android based mobile devices [11].



A160 cwvCheckTrace

	A	B	C	D	
136	attOpenSSLVulnCheck	attOpenSSL	host	AV:N/AC:L/Au:N/C:C/I:C/A:C/E:ND/RL:ND/RC:ND	# Check for vulnerable OpenSSL version
137	cwvApacheNonUnixPathDisclosur	cwvApacheWin32PathDisc	dir	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# Check for the Apache on non-unix platforms path dislosure
138	cwvApacheAxisExternalEntity	cwvApacheAxisFileRetrievalExternalEnti	dir	AV:N/AC:L/Au:N/C:C/I:N/A:N/E:ND/RL:ND/RC:ND	# Apache Axis XXE vulnerability
139	attMultipleVendorXMLParserDoS	attMultipleVendorXMLentityExplosion	request	AV:N/AC:L/Au:N/C:N/I:N/A:C/E:ND/RL:ND/RC:ND	# Multiple vendor XML parser DoS attack
140	cwvApacheAxisHappyAxis	cwvApacheAxisInfoLeak	dir	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# Apache Axis sample servlet information leak
141	cwvApacheAxisFingerPrintJsp	cwvApacheAxisInfoLeak	dir	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# Apache Axis sample servlet information leak
142	cwvPhpWin32FileRetrieval	cwvPHPWin32FileRetrieval	dir	AV:N/AC:L/Au:N/C:C/I:N/A:N/E:ND/RL:ND/RC:ND	# Php on Apache Win32 file retrieval
143	cwvPhpMyAdminUnixFileRetrieva	cwvPhpMyadminFileRetrieval	dir	AV:N/AC:L/Au:N/C:C/I:N/A:N/E:ND/RL:ND/RC:ND	# PhpmyAdmin file retrieval on unix
144	cwvPhpMyAdminWin32FileRetrie	cwvPhpMyadminFileRetrieval	dir	AV:N/AC:L/Au:N/C:C/I:N/A:N/E:ND/RL:ND/RC:ND	# PhpmyAdmin file retrieval on unix
145	cwvHtDigUnixFileRetrieval	cwvHtDigEtcPasswdRetrieval	dir	AV:N/AC:L/Au:N/C:C/I:N/A:N/E:ND/RL:ND/RC:ND	# Htdig file retrieval on Unix
146	attMacOsXFinderDirectoryConten	cwvMacOsXFinderDirContents	dir	AV:N/AC:L/Au:N/C:P/I:P/A:N/E:ND/RL:ND/RC:ND	# Mac OS X finder directory listing vulnerability
147	cwvNonExistentJavaWebServiceP	cwvNoSuchJavaWebServicePathDisc	dir	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# nosuchfile.jws?wsdl gives path on Apache AXIS
148	cwvOracleIsqlPlusOverflow	cwvOracleISqlPlusOverflowInUseridFielc	dir	AV:N/AC:L/Au:N/C:C/I:C/A:C/E:ND/RL:ND/RC:ND	# Oracle iSqlPlus buffer overflow
149	cwvMSCMSxss	cwvMicrosoftContentManagementXssVu	dir	AV:N/AC:L/Au:N/C:P/I:P/A:P/E:ND/RL:ND/RC:ND	# MCMS cross site scripting vulnerability
150	attTomcatCatalinaJSPsourceDisc	attTomcatCatalinaJSPFileDisc	path	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# Apache Tomcat source code disclosure
151	cwvApacheAxisDoSEchoHeaders	cwvAxisEchoHeadersExplodeEntity	dir	AV:N/AC:L/Au:N/C:N/I:N/A:C/E:ND/RL:ND/RC:ND	# Apache axis entity DoS using echoheader.jws sample web service
152	cwvPhpMyAdminUnixFileRetrieva	cwvPhpMyadminFileRetrieval	dir	AV:N/AC:L/Au:N/C:C/I:N/A:N/E:ND/RL:ND/RC:ND	# PhpmyAdmin file retrieval on unix
153	cwvPhpMyAdminWin32FileRetrie	cwvPhpMyadminFileRetrieval	dir	AV:N/AC:L/Au:N/C:C/I:N/A:N/E:ND/RL:ND/RC:ND	# PhpmyAdmin file retrieval on unix
154	attRobotsTxtSearch	attRobotsTxt	dir	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# Retrieve the robots.txt file from every directory
155	cwvMSADCSDLLprobe	cwvMSADCSDLL	dir	AV:N/AC:L/Au:N/C:C/I:C/A:C/E:ND/RL:ND/RC:ND	# Foundstone's hack for msadcs.dll (MDAC v2.6 and below)
156	cwvSearchForWS_FTPLogFile	cwvWS_FTP_Log	dir	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# WS_FTP.LOG leaks important information
157	cwvWebConfig	cwvWebConfig	dir	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# Attempt to retrieve the contents of the /Web.config configuration fi
158	cwvVtiCnfHtm	cwvVtiCnfHtm	path	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# Request every filename.htm file as /_vti_cnf/filename.htm
159	cwv2eusXSSAdmin	cwv2eusXSSAdmin	dir	AV:N/AC:L/Au:N/C:P/I:P/A:P/E:ND/RL:ND/RC:ND	# Check for Zeus web server XSS in admin console
160	cwvCheckTrace	cwvTraceMethod	host	AV:N/AC:L/Au:N/C:P/I:N/A:N/E:ND/RL:ND/RC:ND	# Check if HTTP method TRACE is supported

Sheet1 Sheet2 Sheet3

Figure 5: Analysis using CVSS

V. CONCLUSION

Rapid growth of the malwares in every year in android based mobile devices. It is a hot topic to find the malwares for research scholars and antivirus companies. Signature based detection method gives the protection when the threat is identified before itself. This method does not achieve to find unnoticed malwares. Our effective method is built to find malwares in mobile applications using static analysis approach. This classification method uses N-gram concept and it is applied to the source code. The test set application's codes are compared with training set application which is already trained to the machine and it contains many examples collected from the public resources. CVSS is used to find the Vulnerabilities' relentlessness in the apk file. It is used to achieve the true positive and false positive rates.

REFERENCES

[1] Kirti Mathur,Saroj Hiranwal,"A Survey on Techniques in Detection analyzing malware executables" International Journal of Advanced Research in Computer Science and Software Engineering", Vol.3,Issues 4,pp 422-428,2013.

[2] P.Vinod,V.Laxmi, M.S.Gaur , "Survey on Malware Detection Methods,3rd Hackers" Workshop on Computer and Internet Security,Department of Computer Science Engineering, Prabhu Goel Research Centre for Computer and Internet Security,IIT,Kanpur,PP.74-79,2009.

[3] A.Bose, X.Hu Kang,G.Shin and T.Park,"Behavioral Detection of Malware on Mobile Handsets", IEEE International Conference on Mobile Systems, Applications, Services , pp 225-238,2009.

[4] Marwa M.A.Elfattah,Aliaaa A.A.Yousif and Ebada sarhan amhed,"Handsets Malware Threats and Facing Techniques" International Journal of Advanced Computer Science and Applications , Vol.2,No.12,pp 42-48,2011.

[5] Aubrey-Derrick Schmidt, Rainer Bye, Hans-Gunther Schmidt, Jan Clausen, Osman Kiraz , Kamer Ali Yuksel, Seyit Ahmet Camtepe, and Sahin Albayrak, "Static Analysis of Executables for Collaborative Malware Detection on Android", IEEE International Conference on Communications,pp: 1-5, 2009.

[6] Dex2Jar [Online] Available: https://code.google.com/p/dex2jar/downloads/list Date Accessed: 2014 January.

[7] Android APKTOOL [Online], Available: http://code.google.com/p/android-apktool Date Accessed: 2014 January.

[8] Java Decompiler [Online], Available: http://java.decompiler.free.fr/?q=jdgui, Date Accessed: 2014 January.

[9] "APK2JAVA"[Online],Available:https://code.google.com/apk2java/downloads/detail?name=apk2java_v_1.0.zip , Date accessed:2014 January.

[10] Igor Santos,Yoseba P.Kenyas,Jamie Devesa and Pablo G.Bringas,"N-gram Based File Signatures for Malware Detection ", International Conference on Enterprise Information Systems(ICEIS), pp 317-320,2009.

[11] Common Vulnerability Scoring System (CVSS) [Online] Available: https://nvd.nist.gov/cvss.cfm?calculator&adv&version=2&vector

[12] Assad Ali, Pavol Zavarsky, Dale Lindskog, Ron Ruhl, "A New CVSS-Based Tool to Mitigate the Effects of Software Vulnerabilities", International Journal for Information Security Research (IJISR), Volume 1, Issue 4, and pp: 178-182, 2011.