Android Anti-Virus System for Malware Mutation in Networking

Chimeleze Collins Uchenna, Mardeni Bin Roslee[*], Prince Ugochukwu Nmenme

Centre for Wireless Technology, Faculty of Engineering, Multimedia University, **MALAYSIA**

[*]Corresponding Contact:
Email: mardeni.roslee@mmu.edu.my

ABSTRACT

Nowadays, the rapid evolution in the mobile phone industry has attracted lots of consumers around the world while smartphones being the trend of the phone with the highest demand by a large margin. Recent research has shown that Android Operating System has accounted for 88% of the mobile phone market which has led to the production of different varieties malware targeted mostly on Android Phones. Furthermore, recent research has also revealed that there is high negligence to this great threat where by Android Antimalware software only counter trivial attacks posed by malware or viruses. This paper supports most of the theories and in fact, focuses on one of the most typical vulnerabilities of Android Antimalware which is the mutation attacks. In this paper, the best in class mobile antimalware for Android were assessed and tested how safe they are against different normal obfuscation strategies even with known malware and the results were not up to a satisfactory level. Furthermore, the scope of this research also spans to the implementation of a proposed antimalware which detects and counters mutation attacks using static detection of Android malware using Integrity Check Technique. The feedbacks were analyzed using SPSS 2.0. Analysis of respondents' feedbacks shows that there is even little or no knowledge of malware threats or proper antimalware by mobile phone users. This brings great concerns and this work shows why assessment of this subject matter is and essential considering the rapid growth of smartphone usage. This paper is to evaluate the efficacy of Anti Malware tools on Android in the face of various evasion techniques while developing a system that counters this evasion technique.

Key words:
Android anti-virus, Malware, CRC3 Algorithm

12/27/2018 **Source of Support:** None, **NoConflict of Interest:** Declared

INTRODUCTION

Mobile devices such as smartphones, iPod, and tablets is gradually becoming famous. Sadly, this draws the attention of malicious developers too. Currently, malware have become a matter of interest in the world of mobile computing today. Mostly, it has been stated in the android OS, which is one of the famous mobile OS, malicious software's has been rapidly growing, and android is the obvious target of today for malware authors (Labs, 2014). At the rate which android malware are growing, the platform has also evolved in the production of antimalware tools for the OS, offered to all Android with a scope of free and paid categories, available at the android app Google Play store.

In this paper, we assess the adequacy of anti-malware apparatuses on Android OS even with different avoidance methods. For instance, polymorphism is a typical confusion strategy that has been broadly utilized by a malware to sidestep recognition software by changing a malware in various structures ("transforms") yet with a similar code structure. Moreover, the expression "change" comprehensively is utilized, to allude to different polymorphic or transformative changes.

Overview of Work

In this work, an AV-Test.org, an antivirus assessment lab has been reviewed which appraised antiviruses for Android for the culmination of their recognition. This research is impertinent to their studies or research in that antivirus's execution in identifying polymorphic variations of known virus are assessed. The greater part of the apparatuses (9/10) examined is in this paper are appraised as "excellent" by them. This Provides motivations to trust that the apparatuses were not examined won't have any better protection from polymorphism likewise considered the vigorousness of antiviruses against Android viruses as of late (Zheng et al., 2012). They execute a subset of the mutations, utilize them to create a few virus variations, and test these on VirusTotal, a web service that checkmates submitted tests against more than forty antiviruses. Their outcomes however just demonstrate the adjustment in general detections rates as the mutations are connected. The consequences of this exploration are exceptionally strong demonstrates that all hostile to malware instruments really surrender for all malware tests tried. Also, the shortcomings and qualities of a portion of the items were additionally found. VirusTotal filter was not utilized because detection rates for some antivirus to virus, (for example Dr. Web and AVG) are boundlessly unique for the portable adaptation and the VirusTotal (maybe desktop-based) rendition. Christodorescu and Jha in 2004 directed a research like this research on desktop antivirus to malware applications eight years back (Fredrikson et al., 2010). They likewise landed at the conclusion that these applications have low strength against virus jumbling. This research investigation depends on Android antivirus, and there was incorporation a few perspectives in this research study that are extraordinary to Android. Moreover, this research dates after many research works (see beneath) on mutating strong detection, and it is normal that the proposed procedures to be promptly coordinated into new plug items. At long last, there are many works in the business about the assessment of desktop antivirus instruments on measurements, for example, signature fulfilment, ease of use et cetera (Rubenking, 2012). The other paper focuses on the investigation and implementation of a neural network binary malware classifier that can classify an unseen file as malicious or benign. The scope has been narrowed down to classify Windows Portable Executable (PE) files based on their imported library function calls (Rad et al., 2018). Last but not least, the other work focuses on analyzing the malware in a restricted environment and how information can be preserved. So, in other to address the negative effects of malicious software, the author discussed some of the malware analysis methods which was used to analyze the software in an effective manner and helped to control them (Eze and Chukwunonso E., 2018).

THE SURVEY

In the quest of investigating the testing of android anti-virus response to malware mutation, a survey was conducted. The objectives of the survey are under listed:

- To evaluate and test all the best anti malware and access their weakness and strength
- To develop and system that effectively counters virus mutation attacks which is a major problem for existing Android Anti Malware.
- To Safeguard the Android devices from newest forms of malware attacks
- To develop a system that counters transformation attacks

Survey Design

The questionnaire was designed with both closed ended and open ended questions so as to provide a more convenient way for the respondents to provide feedback. The questionnaire contained 5-point Likert explanations, requiring no classified subtle elements of respondents in this way making it simple for the respondents to take part in the survey. The Scope of questionnaire involves statistic questions correlated to the research.

SURVEY ADMINISTRATION

The questionnaire was distributed amongst both the staff members and students of FTMS College, Cyberjaya Campus, Malaysia. The staff and students who participated were allowed to take the questionnaires home in other to enable them research further on the topic.

Findings of the Survey

There was a sum total of 150 questionnaires distributed in this research. From the data collected, only 102 responses were received. From this respondents, 83% own android mobile devices and 17% do not own android devices. 67% of respondents use an anti-virus on their android mobile devices, while 33% of users do not make use of an antivirus. A total number of 60 persons (69%) of the respondents makes use of the AVG, 21 persons (20%) makes use of Norton anti-virus and 21 persons made use of the ESET anti-virus. Respondents who positively rated the effectiveness of their current anti-virus were 51 (50%) and those who negatively rated theirs were also 51 (50%). Also, 50% of the users believed that their current anti-virus provided complete solution and detection against mutual attack, the other 50% did went contrary with their choice. 67% feel more secured with an anti-virus on their android devices, while 33% did not feel secured having an anti-virus software. 83.3% of respondents were satisfied with the level of protection that was being offered by their android anti-viruses and 16.7% of them were not satisfied. 16.7% of the respondent do not find the application easy to use. 50% could not really decide if it's easy to use nor difficult to use, while 33.3% found the application easy to use. 16.7% disagree to the statement that android anti-virus tends to slow down phone's performance by exhausting the RAM space, 33.3% were not able to ascertain if the statements is true. While 33.3% agreed to it, and the remaining 16.7% strongly agreed to the question. 16.7% do not agree that their current anti-viruses require certain updates. 16.7% of the respondents also were not able to decide. While 33.3% agreed that updates to their android anti-virus is quite paramount. 66.7% of the participants strongly agree to the owners of the application to be able to access/ for the Android Anti viruses to showcase the level of privacy that is being accessed by applications on their devices.

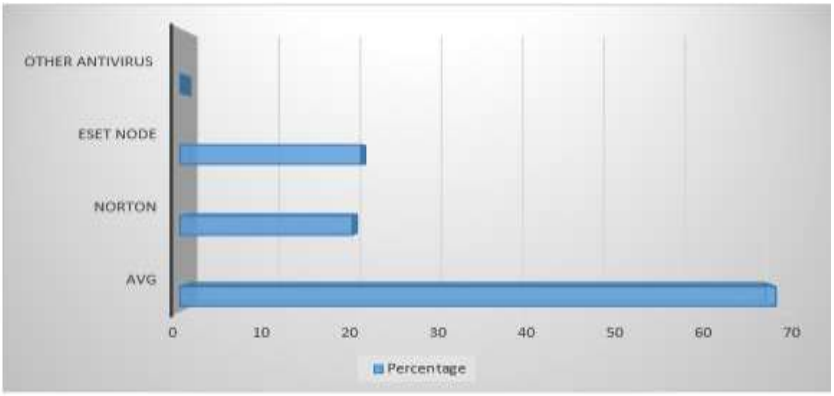


Figure 1: Anti-virus currently used by respondents

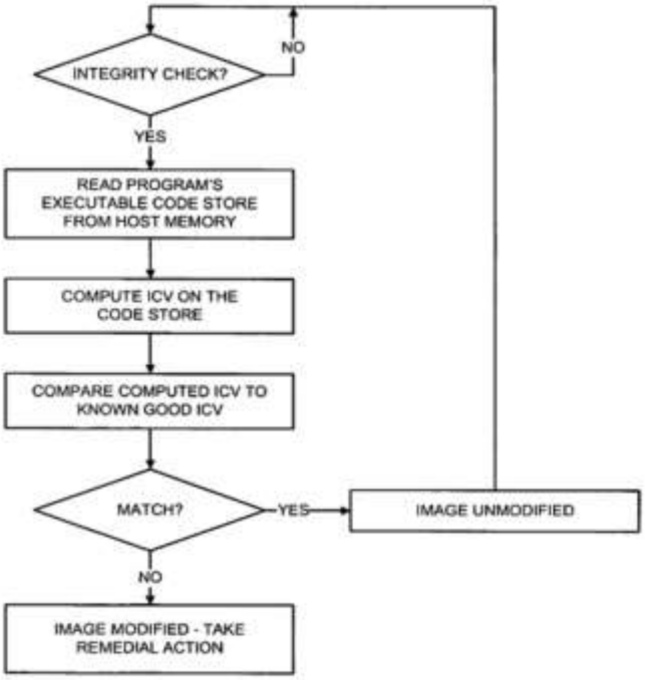
THE EXISTING AND PROPOSED SYSTEM

The method of virus detection method used in proposed system is called the Integrity Check Technique. Integrity checking is on the most popular method of detecting viruses. This method identifies the existence of malwares by matching the hash values of a file with the hash value of its uninfected version. A file is deemed to be uninfected If no difference is found between the two hash values (Thengade et al., 2014). Therefore, integrity check technique does not scan for existence of malicious code. It however, checks whether the code is uninfected as compared to its initial state.

At first, a framework is thought to be uninfected. At this phase the unique characteristics (fingerprint) of each file on the system is figured and put away in a safe area in the storage. The unique characteristics can be computed utilizing a few distinct calculations, for example, an MD4, MD5, or even a simple CRC32 (Thengade et al., 2014). The proposed system was conditioned to similar algorithmic pattern to Integrity Check Virus Detection Technique whilst imploring Cyclic Redundancy Check calculation during file integrity check process. This method is quite old and widely used in most antivirus software, because it is still very reliable (Kane, 2014). This calculation makes unique tables for storing records in the CRC sum. Simultaneously it drives through a folders tree and special tables, on the off chance that it finds new unregistered file then antivirus computes its CRC aggregate and spares it in these tables, if record is now enrolled in the tables. At that point antivirus contrasts its CRC values and their previous value (Koopman et al., 2015).

Using this above stated strategy the proposed system was designed using same structure whereby the antivirus scans new files, checks if its unregistered file using the *AppData class*, calculates its integrity and carries out file permissions checks and its activities before declaring such file as safe or dangerous to the device.

The following flowchart is a representation of the Data Integrity Check Process. This process comprises of an Integrity Check with a Yes or No condition. If the integrity check is fulfilled (Yes), it then reads the Program's executable code store from the host memory before then computing the ICV to known good ICV. A Match condition block comes after the Compare Computed ICV to Known Good ICV and if it does not match (YES), image remains unmodified and is taken back to the Integrity Check condition to repeat the process. Whereas, if it matches (NO), the image is modified and remedy action is being taken.


INTEGRITY CHECK?
NO
YES
READ PROGRAM'S EXECUTABLE CODE STORE FROM HOST MEMORY
COMPUTE ICV ON THE CODE STORE
COMPARE COMPUTED ICV TO KNOWN GOOD ICV
MATCH?
YES
IMAGE UNMODIFIED
NO
IMAGE MODIFIED - TAKE REMEDIAL ACTION

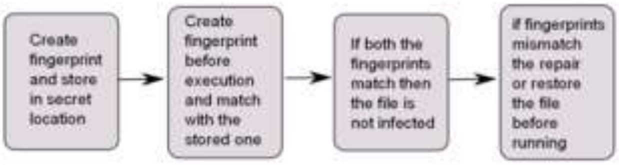

Create fingerprint and store in secret location
Create fingerprint before execution and match with the stored one
If both the fingerprints match then the file is not infected
if fingerprints mismatch the repair or restore the file before running

Figure 2: Data integrity check process

Table 1: Data Integrity Checking Technique

Vendor	Product	Package name	version
AVG	Antivirus Free	com.antivirus	3.1
Dr. Web	Dr. Web anti-virus Light	com.drweb	7.00.3
Lookout	Lookout Mobile Security	com.lookout	8.7.1
ESET	ESET Mobile Security	com.eset.ems	1.1.995
Dr. Web	Dr. Web anti-virus Light	com.drweb	7.00.3
Kaspersky	Kaspersky Mobile Security	com.kms	9.36.28
Trend micro	Mobile Security Personal Ed.	com.trendmicro.tmmspersonal	2.6.2
ESTSoft	ALYac Android	com.estsoft.alyac	1.3.5.2
Zoner	Zoner Antivirus Free	com.zoner.android.antivirus	1.7.2
Webroot	Webroot Security & Antivirus	com.webroot.security	3.1.0

Currently, most anti-virus products are embodied with integrity checking techniques in such a comprehensive way. There are a lot of things that are still not being checked. Older integrity checkers are seen as simply too slow or hard to use to be that effective (Computer Know, 2017).

Data integrity is defined as a fundamental attribute of information security. It is the accuracy and consistency of data which is stored in a data mart, data warehouse, database or any other construct (Veracode, 2017).

Prior to choosing an integrity checker, the following features must be contained:

- Ease of use with a clear and concise report and built-in help.
- An integrity checker has to be fast.
- The integrity check computation must be more sophisticated than a mere checksum.
- It should be able to check and restore the boot and partition sectors.
- It should possess some built-in safety features for protection. Such as: the ability to define the signature information file name and save it on an external media file.

Table 2: Dataset

Famliy	Package name	SHA-1 code	Date Found	Remarks
DroidDream	com.droiddream. bowlingtime	72adcf43e5f945ca9f72 064b81dc0062007f0fbf	03/2011	Root exploit
Fakeplayer	org.me.androidapplication1	1e993b0632d5bc6f0741 0ee31e41dd316435d997	08/2010	SMS trojan

This area portrays the antivirus items and the virus tests which were utilized for this research. Ten antivirus products were assessed, which are recorded in Table 1. There are many free and paid against malware. The most mainstream antivirus; Kaspersky and Trend Micro, which were then not exceptionally prominent but rather are settled merchants in the security business were incorporated.

The malware set utilized as a part of this examination traverses more than two distinctive virus types. Droid-Dream is a malware with root exploit characteristics. DroidDream tries to get root benefits utilizing two diverse root abuses, seethe against the enclosure, and exploit misuse. Fakeplayer the principal known malware on Android, sends SMS messages to premium numbers, in this way costing cash to the client.

Table 3: Malware samples used for testing anti-malware tools

Code	Technique
P	Repack
A	Disassemble and assemble
RP	Rename package
EE	Encrypt native exploit or payload
RI	Rename identifiers
RF	Rename files
ED	Encrypt strings and array data
CR	Reorder code
CI	Call indirection
JN	Insert junk code

All transformations contain P

All transformations except P contain A

Result and Analysis

As has already been discussed, malware samples using various techniques discussed in earlier sections were transformed and passed through anti-malware tools for evaluation. 4 and 5, depicts the series of transformations applied to DroidDream and Fakeplayer samples and the detection results on various anti-malware tools.

The blank cells in the tables represent positive detentions which were positive while cells with 'X' show that the antivirus failed to identify the virus test after the given mutations were implemented to virus sample. The tables mirrors a universal approach for this research. Furthermore testing with insignificant mutations were completed and after that continue with mutations that are more unpredictable. Every mutation is implemented into a virus sample and the mutated sample is tested on an antivirus. On the off chance that detection breaks with minor mutations, the process is then brought to a halt. Next, the utilization of all the DSA mutations. On the off chance that detection still does not break mutation, blends of DSA mutation is also applied

Generally there is no characterized arrangement in which mutations ought to be connected (now and again a heuristic works; for instance, viruses that incorporate native exploitation are probably going to be detected in view of those endeavors). However, in this paper, there was no use of more than two mutations to break identification. Results with all the virus tests are outlined in Table 6. This table gives the insignificant mutations relevant to avoid detection for malware sets. For instance, DroidDream requires both endeavor encryption and call indirection to sidestep Dr. Web's discovery. These negligible mutations additionally give knowledge into what sort of detection marks are being utilized. See Table 4 and Table 5 for key.

Table 4: DroidDream mutation against antiviruses. ('X' indicates failure in detection)

	AVG	Symantec	Lookout	ESET	Dr. Web	Kaspersky	Trend M.	ESTS Of t	Zoner	Web root
P			x							
A			x						x	
RP	x		x					x	x	
EE			x						x	
RI		x	x						x	x
ED			x						x	
CR			x						x	
CI			x						x	
JN			x						x	
RI+EE		x	x	x					x	x
EE+ED			x			x			x	
EE+RF			x				x		x	
EE+CI			x		x				x	
RP+RI+EE+ED+RF+CI	x	x	x	x	x	x	x	x	x	x

Table 5: Fakeplayer mutation against antiviruses. ('x' indicates failure in detection. EE transformation does not apply for lack of native exploit or payload in Fakeplayer)

	AVG	Symantec	Lookout	ESET	Dr.Web	Kaspersky	Trend M.	ESTsoft	Zoner	Webroot
P										
A							x		x	
RP							x	x	x	x
RI				x		x	x		x	
ED							x		x	
CR							x		x	
CI					x		x		x	
JN							x		x	
RP+RI	x	x	x	x		x	x	x	x	x
RP+RI+CI	x	x	x	x	x	x	x	x	x	x

METHOD OF VIRUS DETECTION

The method of virus detection used in proposed system is called the Cyclic Redundancy Check. This method is quite old and widely used in most antivirus software, because it is still very reliable (Kane, 2014). A cyclic redundancy check (CRC) is commonly used in storage devices and digital network as an error-detecting to detect accidental changes in raw data (Wadhe et al., 2012). This algorithm creates special tables for storing files CRC sum. It goes through a folders tree and special tables simultaneously, if it finds new unregistered file then antivirus calculates its CRC sum and saves it in these tables, if file is already registered in the tables. Then antivirus compares its CRC value with their previously calculated value (Koopman et al., 2015). Using this above stated strategy the proposed system was designed using same structure whereby the antivirus scans new files, checks if its unregistered file using the *AppData class*, calculates its CRC and carries out file permissions checks and its activities before declaring such file as safe or dangerous to the device.

Algorithm (CRC32)

CRC32 is a checksum/hashing algorithm that is very commonly used in kernels, and for Internet checksums. It is very similar to the MD5 checksum algorithm. Start with a 32bit checksum with all bits set (0xffffffff). This helps to give an output value other than 0 for an input string of "0" bytes. In a loop: Look up a "polynomial" (actually just a 32bit value) in a table, based on the next piece of input data (usually a byte), and the low N bits of the previous CRC value. (Where N is the size of the data you are operating on -- usually 8 bit bytes.). Shift the previous 32bit CRC value down by N bits. Exclusive-OR the "polynomial" together with the shifted CRC value to produce a new value. The proposed system algorithm is seen below.

Unsigned int crc32 (unsigned char *message)

```
{
 int a,b;
 unsigned int byte, crc;
 a = 0;
 crc = 0xFFFFFFFF;
 while (message[a] != 0)
 {
 byte = message[a]; // for next byte.
 byte = reverse(byte); // reversal 32 -bit.
 for (b = 0; b<= 7; b++)
 { // eight times.
 if ((int)(crc ^ byte) < 0)
 crc = (crc << 1) ^ 0x04C11DB7;
 else crc = crc << 1;
 byte = byte << 1; // Ready for next msg bit.
 } a = a + 1;
 } return reverse(~crc);
}
```

At the end of the loop, exclusive-OR the calculated CRC value with 0xffffffff again (this is identical to doing a binary NOT on the CRC value). This is the final CRC32 result. See Table 6 for key.

Table 6: Evaluation summary ('+' indicates the composition of two transformations)

	DroidDream	Fakeplayer
AVG	RP	RP + RI
Symantec	RI	RP + RI
Lookout	P	RP + RI
ESET	RI + EE	RI
Dr. Web	EE + CI	CI
Kaspersky	EE + ED	RI
Trend M.	EE + RF	A
ESTSoft	RP	RP
Zoner	A	A
Webroot	RI	RP

First Finding

All the antiviruses tested in this research are vulnerable against mutation attacks. Every one of the mutations showing up in Table 6 are anything but difficult to create and apply, reclassify just certain syntactic properties of the virus, and are basic approaches to mutate a malware or virus. Mutations like identifier renaming and information encryption are effortlessly accessible utilizing free and business apparatuses. Endeavor and payload encryption is additionally simple to accomplish.

Second Finding

No less than 43% signatures are not founded on code level relics. That is, these depend on file names, checksums (or paired successions) or data effortlessly acquired by the Package Manager API. Likewise all AVG marks were observed to be gotten from the substance of Android Manifest just (and subsequently that of the PackageManager API). If there should be an occurrence of AVG, the marks depend on application part classes or bundle names or both. Moreover, this data is gotten from Android Manifest as it were. This cases was affirmed by putting a phony Android Manifest in malware bundles and amassing them with whatever remains of the bundle kept as it seems to be. This Android Manifest did not have any of the segments or bundle names declared by the virus. The outcomes were that identification was negative for all the antivirus tests.

Result and Output

The proposed system, Norton, and AVG Antivirus were tested and result were stated below.

Table 7: Test Scenario 1

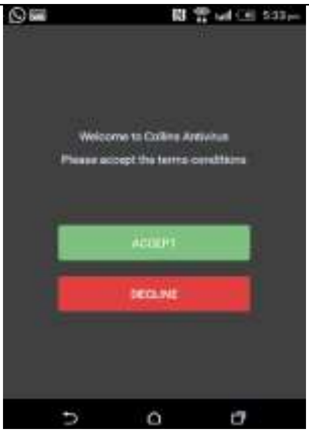
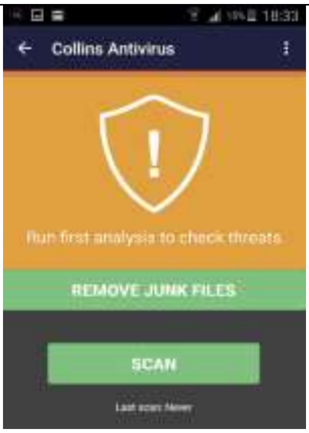
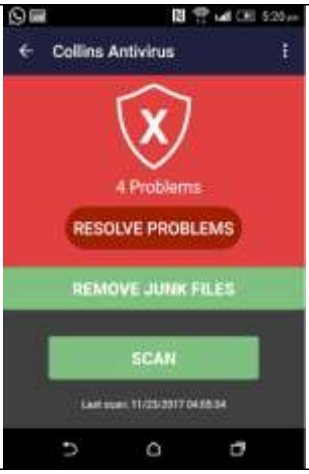
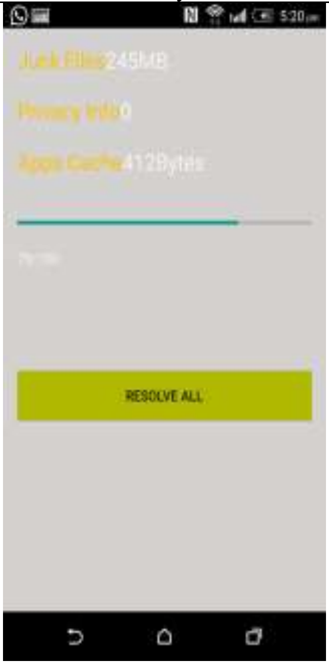
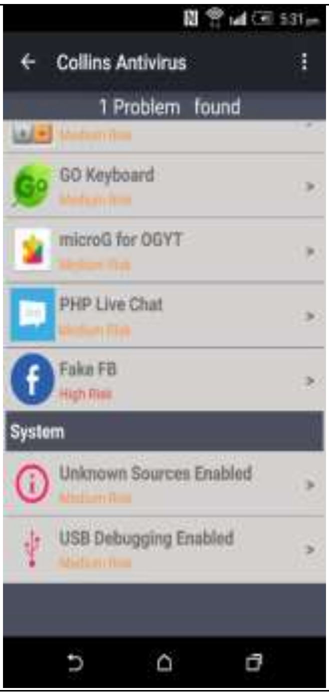
S/N	Expectation	Result
1.	The proposed system is expected to do a background scan on the device upon which it is installed. If the device has not been scanned by the. Then the user upon first usage of the system will be asked to accept the terms and conditions of the before making use of the antivirus. **Result:** This worked perfectly fine as the proposed system treated the device as an unfamiliar device hence the behaviour of the system as portrayed by the image on the right.	 Welcome to Collins Antivirus Please accept the terms-conditions
2.	Further verification to ascertain if the proposed system is able to identify familiar devices. The antivirus is designed to show if the device has been scanned before and show date or history of last scan if the device was scanned before. **Result:** The image on the right shows that device is able to display expected result upon first installation	 Run first analysis to check threats. Last scan: Never
3.	The image on the right side of this table section further proves that the proposed system is able to recognize and device to which it is stored and also able to keep history of scans carried out on the device	 4 Problems Last scan: 11/23/2017 04:55:34 5:20 pm Collins Antivirus RESOLVE PROBLEMS REMOVE JUNK FILES SCAN

Table 8: Test Scenario 2

S/N	Action	Verify if
1.	As displayed on the image in this sections, it is was verified that the proposed system was able to scan the device memory fragments created by junk file in the device and help resolve such issue	 Junk Files 245MB Privacy Info 0 Apps Cache 412Bytes RESOLVE ALL
2.	This was the most important and critical aspect the proposed system. It is regarded as the primary functionality of the proposed system. Based on the context and scope of this research the device used to carry out tests for this research was infected with a virus application which was mutated to avoid detection. **Result:** The proposed system (antivirus) was able to discover vulnerabilities in the device and furthermore detect the mutated virus installed in the device	 Collins Antivirus 1 Problem found Medium Risk GO Keyboard Medium Risk microG for OGYT Medium Risk PHP Live Chat Medium Risk Fake FB High Risk System Unknown Sources Enabled Medium Risk USB Debugging Enabled Medium Risk

3RD Scenario involves the testing of the Norton Antivirus as against mutated droidreams virus (Fake Facebook), and its results are shown below.

Table 9: 3RD Scenario

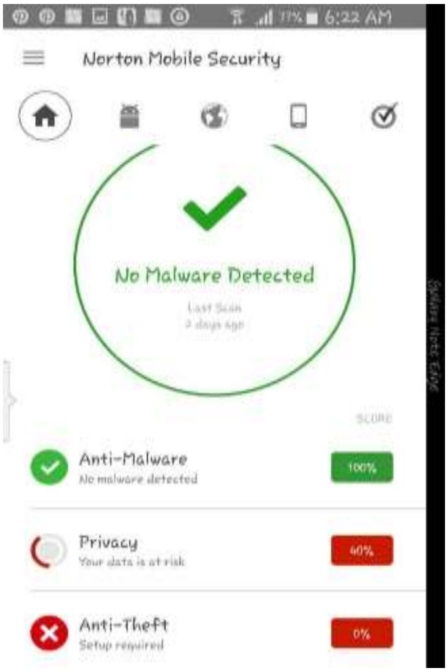
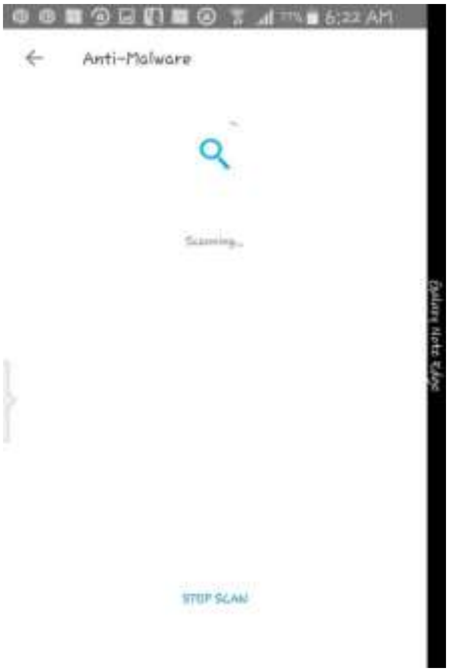
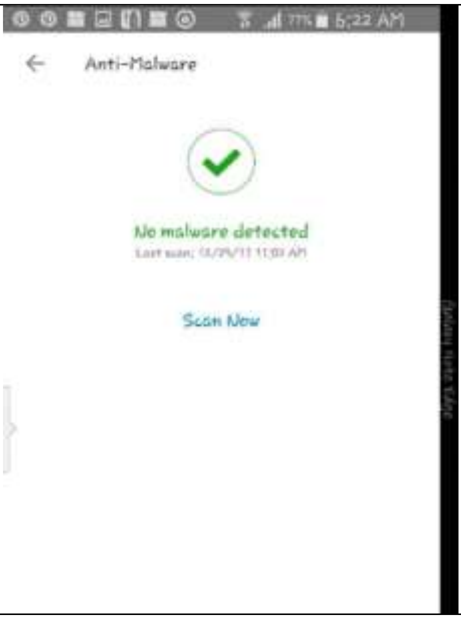
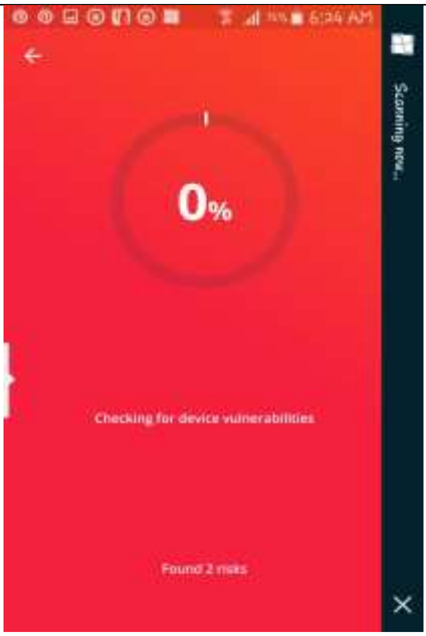
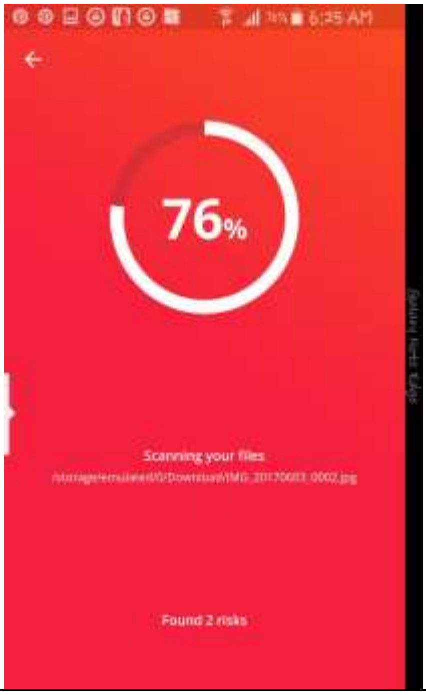
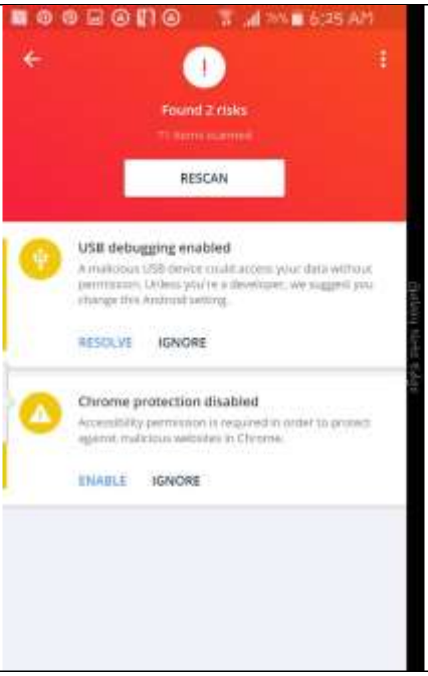
S/N	Expectation
1.	The existing system (Norton Antivirus) is expected to do a background scan on the device upon which it is installed. The user will be ask to choose the options, scan **Result:** The Norton will scanning the whole ,and which is quite slow, and less effective at the end  6:22 AM Norton Mobile Security No Malware Detected Last Scan 2 days ago SCORE Anti-Malware No malware detected 100% Privacy Your data is at risk 40% Anti-Theft Setup required 0% Galaxy Note Edge  6:22 AM Anti-Malware Scanning... STOP SCAN Galaxy Note Edge
2.	Result: Further verification to ascertain if the Norton system is not able to identify malware device (Mutated Fake Facebook). It could not detect the Fake Facebook(Mutated Droidreams Malware ,which shows "No Malware Detected" after effective scanning  6:22 AM Anti-Malware No malware detected Last scan: Scan Now Galaxy Note Edge

Table 10: 4th Test Scenario

AVG is testing against mutated Droidreams malware (Fake Facebook), observations and results were shown below

S/N	Expectation	Result
1.	The existing system (AVG Antivirus) is expected to do a background scan on the device upon which it is installed. The user will be ask to choose the options, scan **Result:** The Norton will scanning the whole ,and which is quite fast when compare to the Norton Antivirus, and less effective at the end	 6:24 AM 0% Checking for device vulnerabilities Found 2 risks Scanning now... 

2.	Result: Further verification to ascertain if the AVG system is not able to identify malware (Mutated Fake Facebook). It could not detect the Fake Facebook(Mutated Droidreams Malware ,but found two non- significant risks	 6:25 AM Found 2 risks 71 items scanned RESCAN USB debugging enabled A malicious USB device could access your data without permission. Unless you're a developer, we suggest you change this Android setting. RESOLVE IGNORE Chrome protection disabled Accessibility permission is required in order to protect against malicious websites in Chrome. ENABLE IGNORE

TRIVIAL TRANSFORMATIONS REPACKING

Those codes which do not require code level changes. The following transformation falls into these categories.

Unzipping and Repacking

All the android packages are signed jar files. Using the zip utilities, we can unzip and then it can be repacked using the android SDK. During this process, these applications are signed with the different key. Detection signatures that matches the original developer key or the checksum of entire package becomes in effective by this transformation (Kalaiarasi et al., 2015) and (Sankareswari and Jothi, 2015).

Disassembling and Reassembling

In this, we disassemble the classes.dex of the application packages and then reassembled again. Due to this the classes, methods, instruction order and strings in the dex file are reassembles in more than one way and the compiled program will be represented in more than one form so that the signature that matches the entire classes.dex file are evaded by this transformation (Kalaiarasi et al., 2015) and (Sankareswari and Jothi, 2015).

Changing Package Name

In android, each application has a unique package name. In the given malicious application, we change the package name so that we can evade the anti-malware products (Kalaiarasi et al., 2015) and (Sankareswari and Jothi, 2015).

CONCLUSION

As a conclusion, due to the architecture of the proposed system which is created about the limitations and scope of this research, it can be further advanced and be protracted with additional functionalities. Some of the features which would be included in the subsequent version of the proposed system comprise of the reduction of false positive produced by the antivirus during device scan, the use of cloud storage for the profiling of virus of signatures which will include frequent fresh updates and the system would be deployed to Android lower level API level 14 covering more android phones and also to iOS mobile operating system. Finally, this paper is to evaluate the efficacy of Anti Malware tools on Android in the face of various evasion techniques has been successfully achieved.

REFERENCES

Computer Know (2017), Integrity Checking. [Online] Available from: <https://www.cknow.com/cms/vtutor/integrity-checking.html [Accessed on: 25th December, 2017].

Eze, A.O. and Chukwunonso E.C. (2018) Malware Analysis and Mitigation in Information Preservation, IOSR Journal of Computer Engineering (IOSR-JCE) ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 20, Issue 4, Ver. I.

Fredrikson, M.; Jha, S.; Christodorescu, M.; Sailer, R. and Yan, X. (2010) "Synthesizing near-optimal malware specifications from suspicious behaviors," in Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, pp. 45–60.

Kalaiarasi, P. Rovina, F. Sowdeeswari, R. and Roshmi, A. (2015), EETA: Enhancing and estimating the transformation of attacks in android application, 4(2).

Kane, J.P. (2014) *System and method for reducing antivirus false positives*. Ca, Inc., U.S. Patent 8,713,686.

Koopman, P., Driscoll, K. and Hall, B. (2015). Selection of Cyclic Redundancy Code and Checksum Algorithms to Ensure Critical Data Integrity.

Labs, M. (2014). McAfee Labs. 5 November, pp. https://www.mcafee.com/hk/resources/reports/rp-quarterly-threat-q3-2014.pdf.

Rad, B.B.; Nejad, M.K.H. and Shahpasand, M. (2018), Malware Classification and Detection Using Artificial Neural Network, Journal of Engineering Science and Technology, pp.14 – 23.

Rubenking, N. J. (2012) "PCMag. The Best Antivirus for 2012," http://www.pcmag.com/article2/0,2817,2372364,00.asp.

Sankareswari, K. and Jothi, S.A. (2015), Hybrid Approach for Securing Biometric Templates Using Visual Cryptography, 3 (9).

Thengade, A., Khaire, A., Mitra, D. and Goyal, A. (2014). Virus Detection Techniques and Their Limitations. *International Journal of Scientific & Engineering Research*, *5*(10).

Veracode (2017). https://www.veracode.com/state-software-security-2017

Wadhe, A., Suryawanshi, R. and Mahajan, N. (2012). Novel Approach for Worm Detection using Modified Crc32 Algorithm.

Zheng, M.; Lee, P. and Lui, J. (2012) "Adam: An automatic and extensible platform to stress test android anti-virus systems," DIMVA.

--0--

ISSN: 2409-3629

Online Archive Link: https://abc.us.org/ojs/index.php/ei/issue/archive