Malware in the Mobile Device Android Environment

Hintea, D, Bird, B & Walker, A

Annual ADFSL Conference on Digital Forensics, Security and Law 2016

May 25th, 10:30 AM

Malware in the Mobile Device Android Environment

Diana Hintea
Coventry University, School of Computing, Electronics and Maths, diana.hintea@coventry.ac.uk

Robert Bird
Coventry University, School of Computing, Electronics and Maths, robert.bird@coventry.ac.uk

Andrew Walker
BAE Systems Applied Intelligence Guildford, walker65@uni.coventry.ac.uk

Scholarly Commons Citation

Hintea, Diana; Bird, Robert; and Walker, Andrew, "Malware in the Mobile Device Android Environment" (2016). *Annual ADFSL Conference on Digital Forensics, Security and Law*. 6.
http://commons.erau.edu/adfsl/2016/wednesday/6

MALWARE IN THE MOBILE DEVICE ANDROID ENVIRONMENT

Diana Hintea
Coventry University
School of Computing, Electronics and Maths
Coventry, CV1 2JH
diana.hintea@coventry.ac.uk

Robert Bird
Coventry University
School of Computing, Electronics and Maths
Coventry, CV1 2JH
robert.bird@coventry.ac.uk

Andrew Walker
BAE Systems Applied Intelligence
Guildford, GU2 7RQ
walker65@uni.coventry.ac.uk

ABSTRACT

Consequent to the world wide increase of smartphone use, the incidence of malware developed to exploit smartphone operating systems has exponentially expanded. Android has become the main target to exploit due to having the largest install base amongst the smartphone operating systems and owing to the open access nature in which application installations are permitted. Many Android users are unaware of the risks associated with a malware infection and to what level current malware scanners protect them. This paper tests how efficient the currently available malware scanners are. To achieve this, ten representative Android security products were selected and tested against a set of 5,560 known and categorized Android malware samples. The tests were carried out using a digital-forensically rigorous testing framework and methodology, which ensures the scientific validity of the results. The detection rates of the tested malware scanners varied widely with half unable to detect any samples at all during initial testing. The malware scanners that were able to detect the samples scored highly with the top four between 97-99% and a fifth scanner scoring 87%. The results emphasise the need for more complex detection mechanisms and protections in future versions of Android and the next generation of malware scanners.

Keywords: malware, mobile forensics, Android

1. INTRODUCTION

The Android operating system and its users are a significant target for malware developers; however, many Android users are unaware of what malware is and the threats associated with it. The first objective of this paper is to establish whether Android is more susceptible to malware than other mobile operating systems and if so, why. Android users that are more security-conscious will often download and use malware scanners in the expectation that it will protect them from malware, but the effectiveness of these scanners is yet to be investigated. The second part of this paper answers this concern by testing a representative set of malware scanners against known malware, making use of both static analysis and virtual devices.

The paper is structured as follows: Section 2 reviews the associated literature review, while Section 3 describes the research methodology. Section 4 presents the setup procedure, while Section 5 provides details on the testing methodology. The results obtained through testing are given in Section 6. Finally, Section 7 concludes the paper.

2. LITERATURE REVIEW

This section describes the related literature in three different areas: i) Android operating system features and malware, ii) the threat of Android malware, and iii) the effectiveness of malware scanners in Android.

2.1 Malware and the Android operating system

Android is a mobile operating system based on the Linux kernel and developed in part by Google. The operating system was launched in September 2008 (Morrill, 2008) and has since grown rapidly. As of 2012, Android has the largest installed base of any mobile platform, powers hundreds of millions of mobile devices and has more than 1 million new device activations each day (Android Open Source Project, 2012).

Having the largest market share makes Android a prime target for malware developers as they seek to spread infection promiscuously and to maximise their impact. Moreover, Motive Security Labs identified in their H2 2014 report what they consider to be key reasons why malware continues to be a significant problem in Android, whilst also performing a comparison with other mobile platforms. The reasons mentioned are the following:

1. Android apps can be downloaded from third-party app stores and web sites.
2. There is no control of the digital certificates used to sign Android apps.
3. Android apps are usually self-signed and can't be traced to the developer.
4. It is easy to hijack an Android app, inject code into it and re-sign it.

In terms of other smartphones, such as iPhone, Blackberry or Windows Mobile, they make up less than 1% of the infections observed. The iPhone and Blackberry have a more controlled app distribution environment and are thus less of a target (Motive Security Labs, 2014). Motive Security noted that Android phones and tablets are responsible for around 50% of the malware infections observed. These results are based on analysis of all platforms, including both desktop and mobile operating systems.

2.2 The threat of Android malware

To understand why malware is so prolific in Android, it is important to understand what it seeks to do and the unique opportunities, exploiting a mobile operating system present to malware developers. Zhou and Jiang (2012) discuss the threat of Android malware. In the paper, they focus on the Android platform and aim to systematize or characterize existing Android malware. In order to achieve this, they collected 1,200 malware samples that cover the majority of existing Android malware "families." Although this appears to be a small number of samples by statistical standards, at the time these samples formed the first large collection of Android malware sample and represented the majority of existing Android malware.

Their findings determine that 86% of the samples are repackaged versions of legitimate applications with malicious payloads and can therefore be broadly categorised as Trojans. Android users should be aware of this statistic and should avoid downloading applications from third parties even if they appear to be genuine applications. Another finding of this report is that 36.7% of the collected malware samples leverage root-level exploits. This result shows how important it is for Android users to keep their operating systems "patched" or up to date. Older versions of Android are potentially vulnerable to root-level exploits that have since been patched.

The report found that 45.3% of the samples have the built-in support of transmitting background short messages (to premium-rate numbers) or initiating phone calls without user awareness. This category of malware could cause financial penalty to a user if they are infected as they could incur a large phone bill that they are required to pay to their provider. This result also enforces the previous observation that the Android operating system should be patched regularly. This is because in Android 4.2 onwards, a security mechanism was introduced whereby the operating system will alert the user before sending a premium rate SMS message (Eitzen, 2012). The final finding of this report is that 51.1% of the malware samples are harvesting users' information, including user accounts and short messages stored on the phones (Zhou and Jiang, 2012). The information gathered by this type of malware could be personally or commercially compromising and could lead to crimes such as identify theft for the individual, or intellectual property and theft of commercial secrets for a business.

This paper identifies the key threats of Android malware - information theft and fraud through the misuse of premium rate texts and calls. Mobile operating systems, such as Android, store significant amounts of personal and business information that is potentially valuable to malware developers. Secondly, it finds that mobile operating systems frequently run on devices that contain SIM cards, making it possible for malware developers to obtain money directly through fraudulent phone charges.

2.3 The effectiveness of malware scanners in Android

A recent paper that discusses the effectiveness of malware scanners in Android is: "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket" (Arp et al. 2014); hence-forth referred to as the DREBIN paper. In the DREBIN paper ten "anti-virus" scanners were tested "(AntiVir, AVG, Bit-Defender, ClamAV, ESET, F-Secure, Kaspersky, McAfee, Panda, Sophos)" against a set of 5,560 malware samples; the same malware set used in this research. The paper discusses the creation of an Android application called DREBIN which is a lightweight method for detection of Android malware that enables identifying malicious applications directly on the smartphone.

The application was tested on Android and detects 94% of the malware; however, the other "anti-virus" products' detections rates were checked by uploading the samples to the VirusTotal website (VirusTotal, 2015) and recording the result returned by the service. This disparity of methods renders comparing results accurately impossible. This is because VirusTotal does not differentiate between different products from the same vendor. For example, the AVG product for Windows desktop and Android may work differently and report a different detection rate, but VirusTotal only outputs a detection result for AVG as a single result. Further, VirusTotal does not recommend that its service is used for this purpose. Under the heading "BAD IDEA:

VirusTotal for antivirus/URL scanner testing" on their website's about page they state:

"At VirusTotal we are tired of repeating that the service was not designed as a tool to perform antivirus comparative analyses..." Those who use VirusTotal to perform antivirus comparative analyses should be aware that they are introducing many implicit errors in their methodology, the most obvious being:

1. VirusTotal's antivirus engines are command line versions, so depending on the product, they will not behave exactly the same as the desktop versions: for instance, desktop solutions may use techniques based on behavioural analysis and count with personal firewalls that may decrease entry points and mitigate propagation, etc."
2. Some of the solutions included in VirusTotal are parametrized (in coherence with the developer company's desire) with a different heuristic/aggressiveness level than the official end-user default configuration (VirusTotal, 2015).

For the above stated reasons, the authors decided to test the detection rates of malware scanners available on the Android operating system using the same malware samples. Using this method will give an accurate representation of the detection rates users would experience if they were to come into contact with that malware on their Android device.

3. RESEARCH METHODOLOGY

The key questions this research addresses are the following:

1. If Android is a particular target of malware, are there any protections against malware within the operating system itself?
2. What are the threats it poses? This section of the question seeks to understand the goals of malware created for Android and the threat it poses to users, their devices and data.
3. How effective are malware scanners in detecting and addressing those threats? This section of the question focuses on malware defense and the malware products created by vendors to protect against malware.

The first section is best addressed by using the literature review. Sources that discuss the evolution of the Android operating system and malware created were used. The authors also believe that a literature review is most appropriate for the second section of the question. Sources that analyse the goal of Android malware and the threat it poses will be used. For the third section the authors chose to use a testing methodology. We test a known set of malware samples against ten representative Android malware scanning products using static analysis. The results will be used to determine a detection rate for each product. The testing methodology was chosen for the following reasons:

1. The Android operating system is constantly updated. It is difficult to find sources that test malware in the latest versions of Android.
2. Malware scanners and their definitions are continuously updated. New malware scanners that may not have previously been tested against a dataset may have become available since the source published, or a product may have improved its detection rate.
3. Any source that carried out testing previously is quickly out of date as definitions in malware scanners are updated; in most cases daily. By

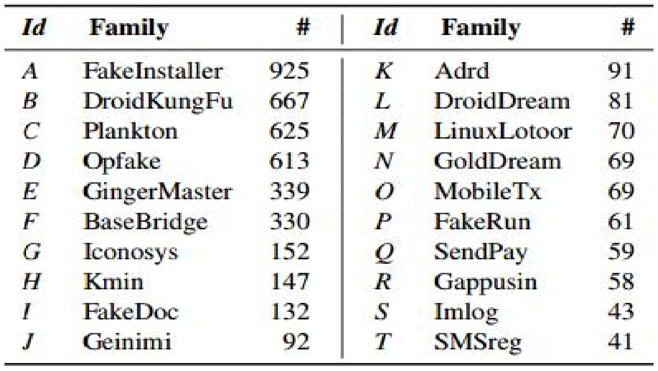
carrying out our own testing we will have an accurate, up-to-date snapshot of the latest products and their definitions.

4. Using data from third party sources such as VirusTotal may not give accurate results.

4. SETUP PROCEDURE

4.1 Selection of malware samples

The authors used the DREBIN malware dataset (Arp et al., 2014) for testing. This is a well-defined and ample database of malware samples. The DREBIN dataset contains 5,550 malware samples collected in the period from August 2010 to October 2012. Each sample was uploaded to the VirusTotal (VirusTotal 2015) service and checked against the ten most common malware scanners, namely AntiVir, AVG, BitDefender, ClamAV, ESET, F-Secure, Kaspersky, McAfee, Panda, and Sophos. Applications were categorised as malicious only if they were detected as so by at least two of the products. The dataset also includes all 1260 samples from the Android Malware Genome Project (Zhou and Jiang, 2012). Additionally, any samples that were identified as adware were excluded from the dataset. The dataset contains malware from 178 different identified malware families with 81.15% of the malware belonging to the top twenty families in the dataset.



Id	Family	#	*Id*	Family	#
A	FakeInstaller	925	*K*	Adrd	91
B	DroidKungFu	667	*L*	DroidDream	81
C	Plankton	625	*M*	LinuxLotoor	70
D	Opfake	613	*N*	GoldDream	69
E	GingerMaster	339	*O*	MobileTx	69
F	BaseBridge	330	*P*	FakeRun	61
G	Iconosys	152	*Q*	SendPay	59
H	Kmin	147	*R*	Gappusin	58
I	FakeDoc	132	*S*	Imlog	43
J	Geinimi	92	*T*	SMSreg	41

Figure 1. DREBIN Dataset, top malware families (Arp et al. 2014)

The authors selected the DREBIN dataset as it contains a large number of samples across a substantial number of malware families, facilitating a comprehensive comparative test. The dataset was collected over a long period of time and is therefore likely to contain samples targeting devices running both newer and older versions of Android. The DREBIN paper itself states that "To the best of our knowledge, this is one of the largest malware datasets that has been used to evaluate a malware detection method on Android" (Arp et al., 2014). When creating the malware set, a number of steps have been taken to make the samples more reliable by requiring the samples to be categorised as malicious by more than one of the tested antivirus products and by excluding adware as it exists in a "twilight zone between malware and benign functionality" (Arp et al., 2104).

Table 1
The Android Application Package install files.

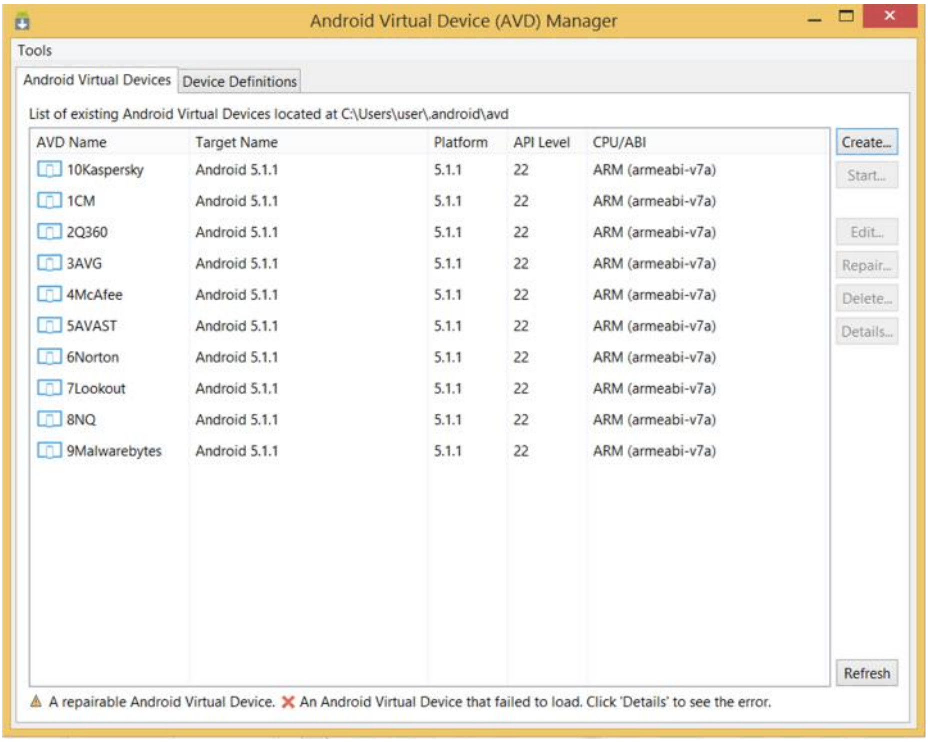
Reference	APK Name	Version	APK MD5
1CM	com.cleanmaster.security.apk	2.4.9	114a64167a7e59edc94d708ac2795c01
2Q360	com.qihoo.security-67.apk	3.2.1	fa8a0eab6bd5979a8cc13db83d7ffcfe
3AVG	com.antivirus.apk	4.3	75d2ee1597a6ed056e9fdee297774fa5
4McAfee	com.wsandroid.suite.apk	4.4.0.392	c44c8dc58fb85fa4d935abb0f17370b7
5AVAST	com.avast.android.mobilesecurity.apk	4.0.7880	b9a8e4a1ca3fa4414f4b192bb7f09e35
6Norton	com.symantec.mobilesecurity	3.10.0.2360	ca366bd072ef1e71598b32a81f8ca8d2
7Lookout	com.lookout	9.15-26a3b5f	900cc7207cd4bc44aadf1184cfa2b60e
8NQ	com.zrgiu.antivirus	7.3.12.02	7820cbc011ad8a99b76166557b8e2ed3
9Malwarebytes	org.malwarebytes.antimalware	1.05.1.1000	8740049b991021df6c42f10ea1102aa6
10Kaspersky	com.kms.free-84.apk	11.8.4.474	7c7eef4f24aed445d97f6d21c5319559

4.2 Selection of malware scanners

The authors selected applications to test from the Google Play Store based on the following criteria:

- Any application that identifies itself as an antimalware or antivirus product and appears in Google's Play store as one of their "Top Apps." Google lists 540 such applications.
- Any application that identifies itself as an antimalware or antivirus product and appears in Google's Play Store as one of the top applications for a category, for example "Top Tools Apps."

This selection process was limited to the top 100 applications in each category. The Google Play store contains the following categories: Books & Reference, Business, Comics, Communication, Education, Entertainment, Finance, Health & Fitness, Libraries & Demo, Lifestyle, Live Wallpaper, Media & Video, Medical, Music & Audio, News & Magazines, Personalisation, Photography, Productivity, Shopping, Social, Sports, Tools, Transport, Travel & Local, Weather, and Widgets.



List of existing Android Virtual Devices located at C:\Users\user\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
10Kaspersky	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)
1CM	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)
2Q360	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)
3AVG	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)
4McAfee	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)
5AVAST	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)
6Norton	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)
7Lookout	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)
8NQ	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)
9Malwarebytes	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)

⚠ A repairable Android Virtual Device. ✕ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Figure 2 -Creating Android Virtual Devices

Table 2
MD5 and SHA1 hashes

	MD5	**SHA1**
8GB SD – No malware	2bd9ab33caeb1a4a395fbeeb5c3ef66b	12418e0c1428dcff432b4dbbb0a46923a6a74d7f
8GB SD - With DREBIN malware Samples	ed3c60da24fa4f57def5e27a339316ad	872f15461eacb225d56f1ae665960f84f540f373

Table 3
Malware applications selected

Reference	**Application Title**	**Developer**	**Version**
1CM	CM Security Antivirus AppLock	Cheetah Mobile (AntiVirus & AppLock)	2.4.9
2Q360	360 Security - Antivirus FREE	Qihoo 360 (NYSE:QIHU)	3.2.1
3AVG	AntiVirus Security - FREE	AVG Mobile	4.3
4McAfee	Security & Antivirus -FREE	McAfee (Intel Security)	4.4.0.392
5AVAST	Antivirus & Security	AVAST Software	4.0.7880
6Norton	Norton Security and Antivirus	NortonMobile	3.10.0.2360
7Lookout	Security & Antivirus | Lookout	Lookout Mobile Security	9.15-26a3b5f
8NQ	Antivirus Free-Mobile Security	NQ Security Lab	7.3.12.02
9Malwarebytes	Malwarebytes Anti-Malware	Malwarebytes	1.05.1.1000
10Kaspersky	Kaspersky Internet Security	Kaspersky Lab	11.8.4.474

The rationale behind this selection process is that users will browse the "Top Apps" when selecting a malware scanning product. Another method would be to use search terms within the Play Store, such as "antivirus" and "malware". This is problematic due to the naming conventions of applications, key words and the search algorithm used by the Play Store. This would also introduce bias, as key words and terms we associate with malware scanners are likely to differ from that of the average user. The applications listed in Table 3 were selected.

The APK (Android Application Package) install files were then downloaded for the selected applications from the Google Play Store.

4.3 Using AVD (Android Virtual Device)

For testing, the authors decided to use virtualised Android phones by using AVD (Android Virtual Device) which is included in the Android SDK. A virtual Android phone was used for testing each malware scanner. Each phone was configured in AVD as illustrated in Figure 2.

Each virtual device is named with the reference of the scanner to be tested on that device. The Nexus 5 device template was chosen and the device was set to run the latest version of Android (5.1.1 13/04/2015). For the CPU architecture ARM was chosen so as to replicate the architecture found on the physical Nexus 5 phone. Cameras for the device were emulated, 2048MB of RAM (default) and 200MB of internal storage (default) were allocated. The virtual SD card (Section 6.4) containing the malware samples for each device is attached.

4.4 Virtual SD Card

Once the decision to use virtual devices for testing was made, the authors decided to create an SD card image for each device using the Android SDK tool "mksdcard" in order to load the malware samples onto each virtual device. The authors created a blank 8GB SD card using the command: *"mksdcard.exe 8G 8GBSD"* and mounted the image in Ubuntu and copied all 5560 malware samples onto the SD card image, followed by un-mounting the image. The created image served as the master file for all the virtual device's SD cards and was not used in testing. An SD card image was made for each virtual device and named after its reference number used in the test. A virtual SD card was used, as it was easy to verify all the samples to be loaded by checking the hash of the SD card image. Furthermore, using a virtual SD card is a lot faster than booting the phone and copying each malware sample individually.

5. TESTING

The following steps were followed to test each malware scanner. It is important that testing occurs in a forensically-sound environment so that a third party performing the same tests would achieve the same results. When carrying out the testing, the authors adhered to the UK ACPO (Association of Chief Police Officers) principles (ACPO, 2012) which provides guidelines on the handling of digital evidence.

Although these principles are aimed at law enforcement and admissibility of evidence in court, they serve as a solid framework for this type of testing. ACPO Principle 3 will be followed and states: "An audit trail or other record of all processes applied to digital evidence should be created and preserved. An independent third party should be able to examine those processes and achieve the same result" (ACPO 2012).

Steps to reproduce:

1. Verify the SD card image to be tested. The MD5 and SHA1 hash should match that of the original SD card image.
2. Configure the Android Virtual device and attach the applicable virtual SD card containing the malware samples.
3. Select the device in AVD manager and select "Start" to power on the device.
4. Locate "adb.exe," installed as part of the Android SDK.
5. At the command line, call "adb.exe shell." This will provide shell access to the device.
    a. At the shell, navigate to the mounted SD card using "cd /storage/sdcard."
    b. Use the ls command to list the mounted files "ls –la." If the samples are listed, the SD card has mounted successfully.
    c. Exit the shell "exit."
6. Install the malware scanning product to be tested.
7. Within the emulator window, find and launch the application.
8. Within the application, ensure that the most aggressive scanning option is enabled and that that the definitions are up to date. This will vary between applications.
9. Put the device in Airplane mode to prevent further updating.
10. Begin the scan.
11. Record the result. The detection rate is calculated based on how many of the 5560 samples are detected.

As it can be seen from Table 4, five of the malware scanners were unable to detect any malware samples. The authors decided to carry out further testing to investigate the possible cause by appending the .apk extension to a small set of the samples on the SD card. Removing the file extension is common practice when sharing malware samples as it is thought to help prevent accidental installation of the malware. The samples selected were:

- 000a067df9235aea987cd1e6b7768bcc1053e640b267c5b1f0deefc18be5dbe1.apk
- 000e0948176bdec2b6e19d0f03e23f37910676a9b7e7709954614bac79269c36.apk
- 00c8de6b31090c32b65f8c30d7227488d2bce5353b31bedf5461419ff463072d.apk
- 00ceaa5f8f9be7a9ce5ffe96b5b6fb2e7e73ad87c2f023db9fa399c40ac59b62.apk
- 00cf11a8b905e891a454e5b3fcae41f3ed405e3c5d0f9c1fce310de4a88c42d0.apk

The five samples were chosen alphabetically when the malware set was listed. The results on this additional testing are shown in Table 4.

Table 4
Main results

Reference	**Detected Malware**	**Detection Rate (%)**
1CM	5532	99.49640287769784
2Q360	0	0
3AVG	5546	99.74820143884892
4McAfee	5468	98.34532374100719
5AVAST	4846	87.15827338129496
6Norton	0	0
7Lookout	0	0
8NQ	0	0
9Malwarebytes	0	0
10Kaspersky	5420	97.48201438848921

Table 5
Drebin testing results

	AV1	**AV2**	**AV3**	**AV4**	**AV5**	**AV6**	**AV7**	**AV8**	**AV9**	**AV10**
Product	AntiVir	AVG	Bit-Defender	ClamAV	ESET	F-Secure	Kaspersky	McAfee	Panda	Sophos
Detection rate	96.41	93.71	84.66	84.54	78.38	64.16	48.50	48.34	9.84	3.99

6. RESULTS AND ANALYSIS

The results obtained are illustrated in Table 4. Both AVG and CM scored over 99%; only failing to detect 16 and 28 pieces of malware respectively out of the 5560 samples. McAfee and Kaspersky had lower, but still high detection rates. Of the malware scanners that detected the malware, AVAST scored the worst, failing to detect over 1 in 10 (87%) of the malware samples. Q360, Norton, Lookout, NQ and Malwarebytes were unable to detect any of the malware samples.

The DREBIN paper tested ten antivirus scanners "(AntiVir, AVG, Bit-Defender, ClamAV, ESET, F-Secure, Kaspersky, McAfee, Panda, Sophos)" (Arp et al., 2014) against the same set of malware samples; their detection rate findings for these products are summarised in the table below. It is important to note that the VirusTotal (VirusTotal 2015) website was used to calculate detection rates in this case.

Direct comparisons to the DREBIN results cannot be made as VirusTotal does not differentiate between vendors' offered products. Malware scanning engines from AVG, Kaspersky, and McAfee were, however, tested in both experiments. The results are compared in Table 6 for completeness.

Table 6
Additional testing

Product	**AVG/3AVG**	**Kaspersky/10Kaspersky**	**McAfee/4McAfee**
DREBIN Paper	93.71	48.50	48.34
Experiment Result	99.74820143884892	97.48201438848921	98.34532374100719
Difference +/-	+6.03820143884892	+48.98201438848921	+50.00532374100719

Further testing, using a set of malware samples appended with the .apk extension, gave the results presented in Table 7 when tested against the malware scanners that were unable to detect any of the malware samples (2Q360, 6Norton, 7Lookout, 8NQ and 9Malwarebytes).

Table 7
Testing using a set of malware samples appended with the .apk extension

Reference	**Detected Malware**	**Detection Rate (%)**
2Q360	0	0
6Norton	4	80
7Lookout	5	100
8NQ	0	0
9Malwarebytes	5	100

As Table 7 shows, two out of the five malware scanners (2Q360, 8NQ) showed no improvement in their detection rate when tested against the "apk malware samples." Three out of the five tested malware scanning products (6Norton, 7Lookout, 9Malwarebytes) began detecting the samples as malware once the apk extension had been added to the samples' filename. This demonstrates that these products are not scanning the samples that do not have an apk file extension. This is a critical security risk as Android does not require a file extension to install an application. With some basic testing using Android virtual devices, the authors discovered that when a user opens an application file without an extension, the Android operating system prompts the user to choose an application to launch that file. The top (and default option) in this case is the Android package installer. It appears as though Android uses the file header rather than the extension to ascertain this. As a recommendation, the developers of these products should carry out the following to remedy the situation:

- Malware scanning products should scan all files by default and examine the file header to ascertain the file type. The filename and extension should not be relied upon.
- Malware scanning products should check these files against a set of known malware signatures; again the filename and extension should not be relied upon to do this as they are easily changed.

Previous major work (DREBIN) had relied on the use of VirusTotal, a third party analysis site, which the site owners themselves state is not fit for the purpose used. In regards to the static analysis we performed, in which the malware is not executed, the only factor affecting the detection rate would be the anti-malware application malfunctioning when run on a virtual device as opposed to a physical one. In all cases the applications did not report errors when installed, executed, or when performing scanning on the virtual device.

7. CONCLUSIONS AND FUTURE WORK

This paper presents the threats of malware in the Android environment. The results show that half of the Android malware scanners were initially unable to detect any of the 5560 malware samples. Further testing proved that the reason for this in three of those products was that the file extension was used, in part, to regulate the scanning. This is a substantial security risk for two reasons: i) a file extension is easily changed and ii) Android does not require a file to have an .apk extension to install it as an application. This paper provides a snapshot of the current detection rates of the tested products and identifies a number of key flaws that could be abused by malware creators.

In terms of future work, it would be useful to investigate those products that showed no improvement in detection rate during the second part of the experiment. The authors would also like to further analyse the results of the top five malware scanners tested in this paper. It would be interesting to see if there are any commonalities amongst the samples that those scanners were unable to detect.

Finally, this paper focused on static analysis of the malware samples. Static analysis is the first line of defense against malware before it is installed or has the chance to be propagated by users. Many of the products tested claim that they will perform analysis of samples at runtime - detecting and blocking malicious behaviour. By testing both the static and dynamic capabilities of malware scanners, a more comprehensive detection rate could be calculated.

REFERENCES

[1] ACPO (2012) *ACPO Good Practice Guide for Digital Evidence*. [online] available from <http://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf>

[2] Android Open Source Project (2012) *Android, the World's Most Popular Mobile Platform* [online] available from <http://developer.android.com/about/index.html>

[3] Arp, D., Spreitzenbarth, M., Malte, H., Gascon, H., and Rieck, K. (2014) 'Drebin: Effective and Explainable Detection of Android Malware in Your Pocket.' *Symposium on Network and Distributed System Security (NDSS)* 23–26

[4] Eitzen, C. Von (2012) *Android 4.2 Warns against Malicious Apps and Premium Rate Texts* [online] available from <http://www.h-online.com/security/news/item/Android-4-2-warns-against-malicious-apps-and-premium-rate-texts-1744110.html> [20 April 2015]

[5] Morrill, D. (2008) *Announcing the Android 1.0 SDK, Release 1* [online] available from <http://android-developers.blogspot.in/2008/09/announcing-android-10-sdk-release-1.html> [10 April 2014]

[6] Motive Security Labs (2014) *Motive Security Labs Malware Report – H2 2014* [online] available from <https://resources.alcatel-lucent.com/asset/184652>

[7] University of Göttingen (2014) *The DREBIN Dataset* [online] available from <http://user.informatik.uni-goettingen.de/~darp/drebin/> [10 April 2015]

[8] VirusTotal (2015) *About VirusTotal* [online] available from <https://www.virustotal.com/en/about/> [27 April 2015]

[9] VirusTotal (2015) *VirusTotal* [online] available from <https://www.VirusTotal.com/> [10 April 2015]

[10] Zhou, Y. and Jiang, X. (2012) 'Dissecting Android Malware: Characterization and Evolution.' *Proceedings - IEEE Symposium on Security and Privacy* (4), 95–109