An Effective Online Scheme for Detecting Android Malware

Shuang Liang, Xiaojiang Du and Chiu C. Tan
Dept. of Computer and Information Science,
Temple University, Philadelphia, PA 19122, USA
{shuang.liang2012, dux, cctan}@temple.edu

Wei Yu
Dept. of Computer and Information Science,
Towson University, Towson, MD 21252, USA
wyu@towson.edu

***Abstract*—The growing popularity of Android-based smartphones have led to the rise of Android based malware. In particular, profit-motivated malware is becoming increasingly popular in Android malware distribution. These malware typically profit by sending premium-rate SMS messages and/or make premium-rate phone calls from infected devices without user consent. In this paper, we investigate the telephony framework of the Android operating system and propose a novel process user-identification (UID) based online detection scheme. Our scheme can effectively detect premium-rate and background SMS messages as well as premium-rate phone calls initiated by malware. We implemented our detection system on a Samsung Google Nexus 4 running Android Jelly Bean and tested the effectiveness of detecting real malware from Android markets. The experimental results show that our scheme is efficient and effective in detecting background messages and premium-rate messages and phone calls. Our scheme can detect and block all the background and premium-rate SMS messages and phone calls initiated by popular malware.**

***Keywords*—Android; smartphone; malware detection; security**

I. Introduction

Android has become the world's most popular mobile platform operating system [1]. As of the second quarter of 2013, the Android operating system (OS) occupies 79.3% of the total smartphone operating system market share [2]. The rapid growth brings a significant profit boost for Android smartphone vendors such as Samsung and LG [3]. Unfortunately, the popularity of the Android mobile platform has also attracted malware developers to design Android-based malware. According to the statistics of the computer security company F-Secures mobile threat report [4], the third quarter of 2013 has seen a fast growth of profit-motivated threats, which typically make monetary profit by sending premium-rate SMS messages from infected mobile devices without user consent. As reported by Android malware researchers and anti-virus companies, a large number of malware families have been spreading in official and unofficial Android markets. Example of some popular Android malware includes Geinimi [5], DroidKungFu [6], and AnserverBot [7]. Also in existence are SMS-sending trojan families such as FakeInst [8], OpFake [9], and SmsSend [10]. Such malware can cause financial loss and privacy leakage for mobile users.

Most of the current Android malware detection schemes are *offline* [11], [12]. This means that they identify malware by examining the source code and/or the permission file of Android application (App) package. If the source code is not available, these schemes have to obtain the source code by reverse engineering. The limitation of this approach is that some malware is starting to use code obfuscation, java reflection functions, or native libraries to circumvent the vetting process [13].

In this paper, we consider a different approach and develop an *online* malware detection tool named *Droid Sentinel*, which can detect and block background SMS messages and phone calls initiated by malware. Droid Sentinel sends an alarm to the user when background SMS messages, background phone calls, or premium SMS messages or phone calls are detected. Through our experimental study on commercial Android phones, we found that in order to send SMS messages using the system SMS application or to make phone calls using the system dial application, the sequence of the system service UIDs (program user ID) is fixed for a specific Android OS version. Based on this finding, we propose the following detection mechanism: we record the UID sequences of normal usage for sending SMS and making phone calls. Then, the normal UID sequences are used to detect background/premium SMS messages and phone calls initiated by malware, which have different UID sequences. Our experimental data using real malware shows that our developed system is effective and efficient. In our experiments, Droid Sentinel can detect and block all background/premium SMS messages and phone calls. Droid Sentinel can also send the user an alarm for background/premium SMS and background phone calls.

The contributions of this paper are summarized as follows:

- We propose a novel UID-sequence-based online malware detection scheme, which can effectively detect and block background/premium SMS messages and phone calls initiated by malware.
- Our proposed online malware detection scheme is orthogonal to existing offline detection schemes. Our proposed scheme may be used in addition to the offline schemes. Furthermore, our proposed scheme cannot be circumvented by code obfuscation and other tricks played by malware authors.
- We implement the online malware detection scheme in the Android Jelly Bean version. Then, we conduct experiments with real malware samples on Android smartphones and real-world cellular networks.
- The experimental data shows that our detection scheme is effective and efficient.

The rest of this paper is organized as follows: In Section II, we discuss popular premium-rate SMS message and phone call related malware families and their threats to mobile users.

978-1-4799-3572-7/14/$31.00 ©2014 IEEE

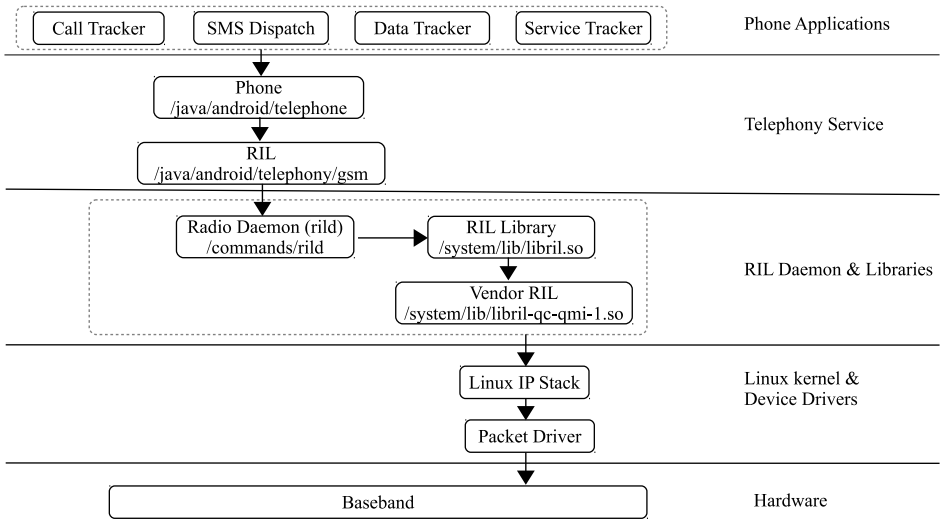


Fig. 1. Android Telephony System Components Interactions

In Section III, we present our online malware detection scheme Droid Sentinel. In Section IV, we describe the evaluation of our online detection scheme which used real Android platforms. We summarize lessons learned from Droid Sentinel and some recommendations for dealing with Android malware in Section V. We discuss related work in Section VI and conclude the paper in Section VII.

II. Threats from SMS and Phone Call Related Malware

From our investigation of different types of Android malware, we found that most of the malware that can cause financial loss are premium-rate SMS related. This observation is consistent with the data published by Zhou *et al.*[14]. For the 1,260 malware samples collected by [14], 571 malware samples sign up for premium services by sending SMS messages, 315 malware samples block incoming SMS confirmation messages, and 138 malware samples steal personal information through SMS messages. In the real world, SMS and phone call related malware and Trojan applications spread widely and upgrade quickly [15], [16], [17], [18]. This kind of malware can send premium SMS messages in the background and intercept incoming SMS messages from service providers without being noticed by the user. We analyzed most of the premium-SMS-message related malware and found that they share some common features:

- They need a user to push a button to trigger the SMS sending event.
- Most of the malware requests SMS related permissions.
- They inject their malicious code into popular Android Apps.
- They intercept incoming confirmation messages such that the event is not noticed by the user.

New malware variations are developed rapidly to avoid being detected by existing security schemes. Some malware also adopted obfuscation (transforming program code to conceal its purpose or logic), which makes it more difficult for static analysis. For example, Android.Gamex [19] uses a trivial encryption (byte XOR with 0x12) to hide a package in assets/logos.png. As described by Collberg et al. [20], a number of automatic code obfuscation methods have been developed, showing that the analysis of code and reverse engineering become challenging. Rastogi *et al.* evaluated several commercial mobile anti-malware products for Android and tested how resistant they are against various common obfuscation techniques [21]. The experimental data showed that all the anti-malware products evaluated would be susceptible to common evasion techniques. To address this urgent issue, in this paper, we propose a new approach for detecting SMS related malware. We designed and implemented Droid Sentinel - an online tool that runs in real time to detect and block SMS messages and phone calls initiated by malware. The details of Droid Sentinel are given in the following sections.

III. Droid Sentinel - An Online Real-time Malware Detection Scheme

A. Architecture Overview

Android is an open-source software stack created for a wide array of devices with different forms of factors [22]. Figure 1 illustrates the interactions of Android telephony system components. The Android system is built on the Linux kernel. It does not include the full feature set of standard Linux kernel and the native windowing system. In addition, there is no standard glibc support. As shown in Figure 2, on top of the underlying hardware, a typical Android system consists of the Applications (Apps) on the top, the Application Framework, the Native Libraries, and the Linux Kernel on the bottom. Our developed tool, the Droid Sentinel, has two components spanning the Applications layer and the Libraries layer. The component residing on the Applications layer is the Droid Sentinel App that is used to obtain the recent process UIDs. It then determines whether the captured event, by the corresponding component in the Libraries layer, is background/premium SMS messages or phone calls. The other component is the one

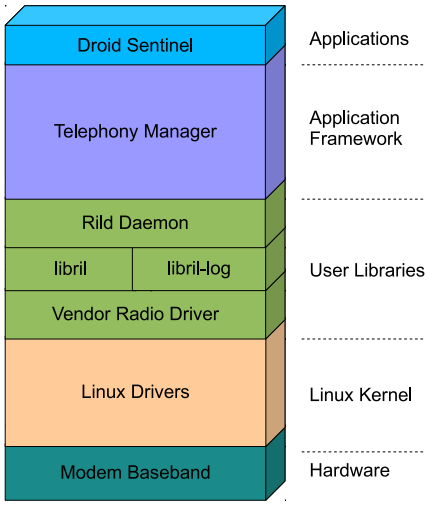


Fig. 2. The Architecture of Droid Sentinel

found in the Libraries layer. We modified the radio library and embedded our event detection component in that layer. When the telephony events (sending SMS messages, making phone calls) happen, the Libraries-layer program component reports the event to the corresponding Application layer part.

We implemented Droid Sentinel in the application layer of an Android system. Android telephony system components are shown in Figure 1. We extended the Radio Interface Layer (RIL) Daemon and the Libraries with 'libril-log' to support SMS and phone call detection and reporting. As shown in Figure 1, all telephony related phone Apps such as Dialer, Call Tracker, SMS, etc., go from the top Telephone Service layer to the RIL Daemon and Libraries layer. From there, the Apps go to the Linux Kernel and Device Driver layer, and finally, end at the modem hardware. All of the phone Apps are started during the system boot up phase. These Apps are tied up with the Android telephony framework services. The telephony framework provides APIs to access the Phone services. All queries from the application layer through APIs are directed to the RIL of the Android system by these services.

Generally speaking, RIL is the bridge between the Android phone framework services and the hardware. It consists of two basic components: the RIL daemon (RILD) and vendor specific RIL library. The RILD is initialized during the system boot up phase. It detects the running environment to check whether it is running in an emulator or on a real phone. *libreference-ril.so* is loaded if it is running in an emulator; otherwise, the vendor RIL library is loaded. Because we implemented our online detection scheme on real-world hardware instead of the virtual emulator, we used the vendor specific RIL library. For the Google Nexus 4 phone, it is the Qualcomm modem library *libril-qc-qmi-1.so*. In the end, the procedure *RIL_startEventLoop* is invoked and the program control goes to the dynamically linked library *libril.so*. We extended the dynamically linked library to support event logging and reporting with an additional library, referred to as the libril-log, that we built. Before sending requests from high layer telephony services to the modem hardware driver, libril-log sends them to its counterpart service in the Android application

```
AID_ROOT             0  /* traditional unix root user */
AID_SYSTEM        1000  /* system server */
AID_RADIO         1001  /* telephony subsystem, RIL */
AID_BLUETOOTH     1002  /* bluetooth subsystem */
AID_GRAPHICS      1003  /* graphics devices */
AID_INPUT         1004  /* input devices */
AID_AUDIO         1005  /* audio devices */
AID_CAMERA        1006  /* camera devices */
AID_LOG           1007  /* log devices */
AID_COMPASS       1008  /* compass device */
AID_MOUNT         1009  /* mountd socket */
AID_WIFI          1010  /* wifi subsystem */
AID_ADB           1011  /* android debug bridge (adbd) */
AID_INSTALL       1012  /* group for installing packages */
AID_MEDIA         1013  /* mediaserver process */
AID_DHCP          1014  /* dhcp client */
```

Fig. 3. Snippet of Android UIDs and GIDs

layer (i.e. Droid Sentinel in Figure 2). Droid Sentinel then analyzes the requests and determines whether it is a malware initiated SMS/phone call.

B. Online Scheme for Detecting Background SMS and Phone Calls

Linux uses process UIDs to prevent one process from accessing the space of another process. From a security point of view, Android also takes advantage of this Linux user-ID based protection mechanism to identify and isolate application resources. This approach is different from other operating systems (even different from the traditional Linux configurations) where multiple applications run with the same user permission [23].

We designed our online UID-based detection scheme by exploiting the fact that each Android application or service has its own unique user ID (UID). This is used to isolate one application/service from another. In an Android system, UIDs less than 10,000 are reserved for system users, devices, or daemons with 10,000 being the first application UID [24]. Figure 3 lists some of the reserved UIDs. "AID_RADIO" is the UID for the telephony subsystem which is our focus.

Our detection scheme may be extended to use other reserved Android system UIDs for detecting other malware families, including malware that can take photos from the background without user notice. Through experiments on real phones, we found that in order to complete one action such as dialing, sending SMS messages, or taking photos, a number of system services and applications will be called. The UIDs of these services and applications for dialing and sending SMS messages are unique after the applications are installed. For making phone calls and sending SMS messages, we use the most recent two UIDs (from the UID sequence) as the signature UID sequence. We obtain the normal UID sequence for dialing and sending SMS events by using our self-written Application which automatically detects and records the invoked UID sequences when sending SMS messages and making phone calls on a Google Nexus 4 phone. Table I shows the normal UID signatures for dialing and sending SMS events and their corresponding Android package names. The column UID1 is the UID of the most recent invoked program or service and its corresponding App name. For different Android system versions, the signatures may be different. Hence, when a

different version of Android is used, the normal UID signature profile should be defined first.

TABLE I. NORMAL UID SIGNATURES FOR DIALING AND SENDING SMS MESSAGES

Event	UID1	UID2
Dial	1001 (App Phone)	10000 (App Contacts)
SMS	10022 (App Messaging)	10018 (App Launcher)

Any UID sequence inconsistent with the normal UID signatures will be considered as background (without interacting with users) SMS messages or phone calls. Table II shows the abnormal UID signatures captured during execution of our self-written test bench. For the dialing event, the UID2 is different from that in Table I. For the SMS event, UID1 is different from that in Table I. When these events are detected, an alert will be shown on the user's phone to draw attention. Note that the alert message shows the destination number and the name of the App that initiated the operation. At the same time, the outgoing phone call will be blocked automatically and SMS messages will be held waiting for the user's approval or denial.

TABLE II. ABNORMAL UID SIGNATURES CAPTURED IN TEST BENCH

Event	UID1	UID2
Dial	1001 (App Phone)	1001 (App Phone)
SMS	10044 (App TestBench)	10018 (App Launcher)

In addition, if a background SMS sending or call dialing event is detected, the destination number is checked against premium number prefixes. Our current implementation considers only premium numbers in the United States, which have the form 1-900-###-#### or 900-###-####. Note that our designed system can be easily extended to include other premium numbers. Droid Sentinel will notify the phone owner about a premium-number event that may be more harmful than the background events. In the following, we will demonstrate the implementation of our proposed UID-based detection scheme and the algorithm in detail.

C. Implementation on Android System

As discussed in the architecture overview in Section III-A, our system consists of two components that span from the phone applications layer to the user libraries layer of Android. These two components communicate with each other using the socket interface for system programs provided by Android [25]. We now discuss the implementation details and how our developed online malware detection scheme works. In the following description, we use Droid Sentinel to refer to the App component at the phone applications layer of Android and we use RIL-log to refer to the phone call/SMS message logging and reporting component located at the user libraries layer of Android.

The detection activity is started by Droid Sentinel by initiating the socket connection and sending "START" command to RIL-log. As shown in the flow diagram in Figure 4, Droid Sentinel first connects to the RIL-log by using socket communications in the UNIX domain. After getting the connection request, RIL-log obtains the file handler of this socket and starts receiving commands from Droid Sentinel. After receiving the "START" command, RIL-log enables the logging and reporting feature. After successfully enabling RIL-log, Droid Sentinel starts the socket monitoring thread and

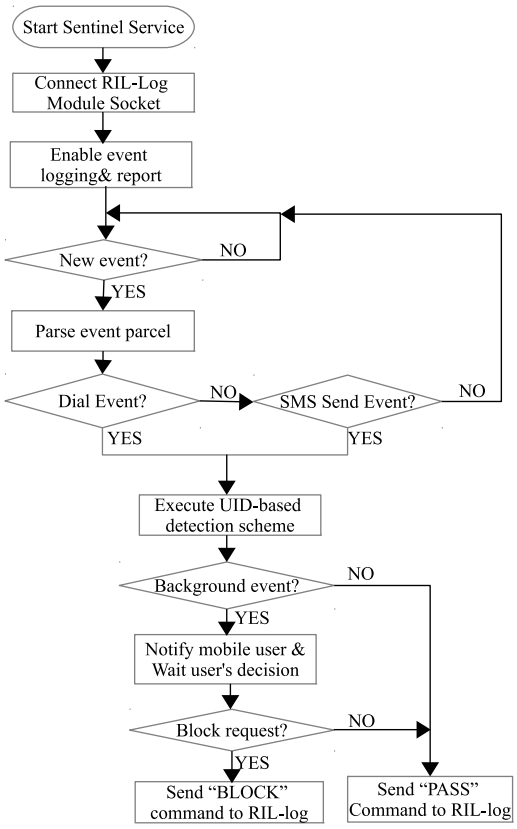


Fig. 4. Flow Diagram of Droid Sentinel

waits for the phone events from RIL-log. Upon receiving the phone event, Droid Sentinel extracts the event type code from the message by parsing the Parcel message container sent from RIL-log. Parcel is designed as a high-performance IPC (Inter-Process Communication) transport and can be used for data serialization [26].

In our current implementation, Droid Sentinel supports detecting two types of events: dialing and sending SMS messages. If it is a dialing event, Droid Sentinel employs the UID-based detection scheme (discussed in the previous subsection) to check whether it is a background event. If this event is a background dialing event, Droid Sentinel extracts the target phone number and determines whether it is a premium phone number by checking the prefix of the number. Droid Sentinel then notifies the phone user about the background dialing threat with the number showing in the message. In addition, the phone call is automatically blocked. On the other hand, if it is not a background phone call, Droid Sentinel sends a "PASS" command to RIL-log that allows the phone call to continue. If it is an SMS-sending event, Droid Sentinel applies the same UID-based detection scheme and decides whether it is a background SMS message. Different from the background dialing event, Droid Sentinel asks the user whether he/she wants to block the SMS message instead of automatically blocking the message. If the user still wants to send this message, Droid Sentinel sends a "PASS" command to let the message be sent; otherwise, the message is blocked.

For the aforementioned UID-based detection scheme, we design it as a two-step algorithm. The first step is to obtain

the UID sequences for malware detection on the current platform. Because the system-reserved UIDs may be different for different Android OS versions, our algorithm first makes a profile of the new system environment. The profiling process is done by Droid Sentinel. We first send an SMS message using the normal Android system SMS-sending App (as well as other non-malicious SMS-sending Apps) and Droid Sentinel records the normal UID sequence after detecting the SMS-sending event. For the background SMS message sending sequence, we launch popular background SMS-sending malware and our test bench to capture the UID sequences. Afterwards, we apply the UID-based detection scheme as described in Algorithm 1 to obtain a UID feature vector which we obtain by comparing the normal UID sequence with the UID sequences generated by the malware.

In the UID-based detection algorithm, we start the normal UID-signature detection with a signature length of 1 (denoted as $sign\_len$ in the algorithm). For all of the malicious UID sequences (denoted as $obtained\_malware\_uid\_seqs$) that we obtained, we compare them with the normal UID sequence of length $sign\_len$. We also try different signature lengths to find the shortest length signature that can distinguish between all malware UID-sequences and the normal UID sequence. We use the notation $sign\_weak$ to indicate that the current signature length is not long enough to distinguish malicious UID sequences from the normal UID sequence. The same algorithm is used to obtain background phone-call UID feature vectors. The second step is to use the obtained UID feature vectors as signatures to detect background/premium phone calls and SMS messages.

The other component of our detection scheme is RIL-log which is implemented as a user library in the library layer of Android. First, RIL-log is registered by the system daemon 'rild' and waits for a socket connection from Droid Sentinel. After RIL-log receives the "START" command from Droid Sentinel, it starts the monitoring process and reports SMS sending and phone dialing requests to Droid Sentinel. We insert our code before the telephony requests are sent to the Android processing functions. Hence, when RIL-log receives the requests, it parses the request data and extracts the request number, target phone number, and the message from the SMS-sending request. Next, RIL-log wraps the data in a single packet and sends it to Droid Sentinel and then waits for the response from Droid Sentinel before moving forward. There are two types of responses from Droid Sentinel: "BLOCKING" or "PASS". On receiving a "BLOCKING" command, RIL-log blocks the SMS sending/dialing request. If the response is "PASS", RIL-log dispatches the pending request to the corresponding processing function. Figure 5 illustrates the flow diagram of the RIL-log request logging and reporting process.

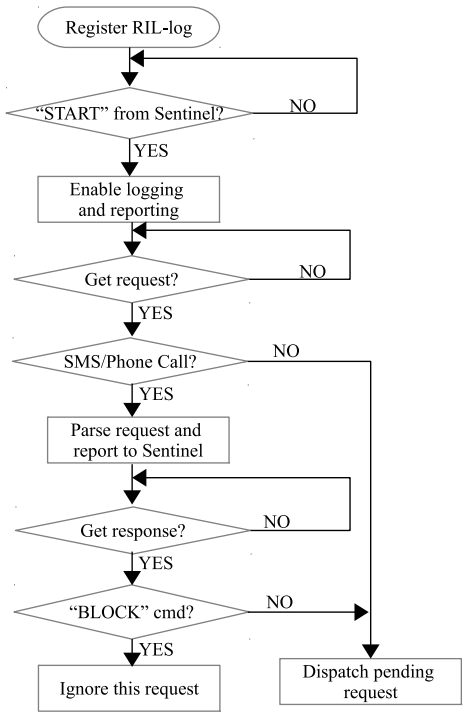


Fig. 5. Flow Diagram of RIL-log

Algorithm 1 UID-based Detection Scheme Algorithm

```
1: sign_len ← 0
2: sign_weak ← false
3: for sign_len = 1 to 10 do
4:    for all malicious_seq ∈ obtained_malware_uid_seqs do
5:       if malicious_seq[sign_len] = normal_seq[sign_len] then
6:          sign_weak ← true
7:          break
8:       end if
9:    end for
10:   if not sign_weak then
11:      break
12:   end if
13: end for
14: if not seq_to_test[sign_len] = normal_seq[sign_len] then
15:    return background_event = true
16: else
17:    return background_event = false
18: end if
```

IV. Performance Evaluation

A. Experimental Environment

We conducted two sets of experiments: one with our self-written test bench and the other with real-world malware downloaded from Contagio [27]. We installed and tested our online malware detection scheme on Android 4.2.2 (Jelly Bean). The test bench, Droid Sentinel, and the real-world malware were installed on a Google Samsung Nexus 4 Android phone. We used the Android Debug Bridge (adb) tool [28] to set up the Android phone experimental environment and show debugging information on a laptop. The experimental setup is shown in Figure 6. In the experiment, we used a Nexus 4 phone with number 267-403-**** to send SMS messages and make phone calls to an iPhone 5 with number 215-301-****. The SMS messages and phone calls went through real cellular networks, including T-Mobile and AT&T.

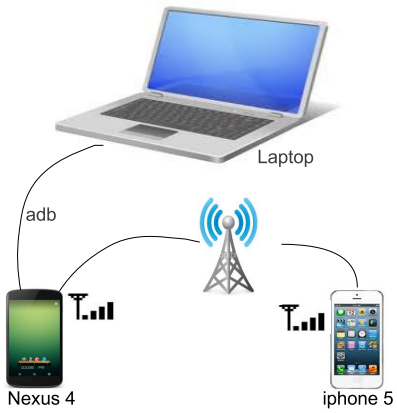
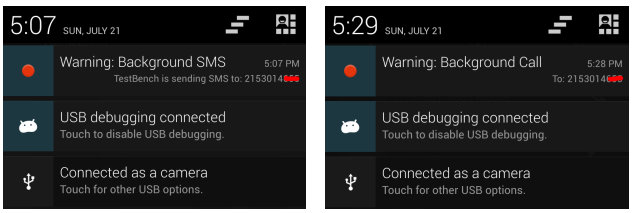


Fig. 6. Online Scheme Experiment Environment

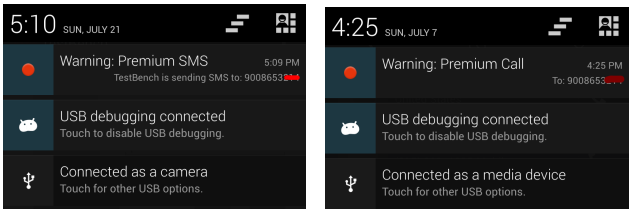
B. Experiments with Self-Written App

In order to test the capability of detecting background SMS messages and phone calls, we developed an Android App as a test bench. Our test bench app can initiate background SMS messages and phone calls. A warning message will pop up on a user's phone as a notification if background SMS or phone calls are detected. Figure 7 shows notifications of a background SMS and phone call. For premium numbers, we test our detection scheme by calling some numbers that start with 900 and 1-900. As shown in Figure 8, the alerts for premium SMS messages and phone calls are shown as notifications on the user's phone. When a background or premium SMS message is detected, the user can press the notification button and a dialog pops up asking the user whether to send or block the SMS message. This is shown in Figure 9.



(a) Background SMS (b) Background Phone Call

Fig. 7. Background SMS and Phone Call Notifications

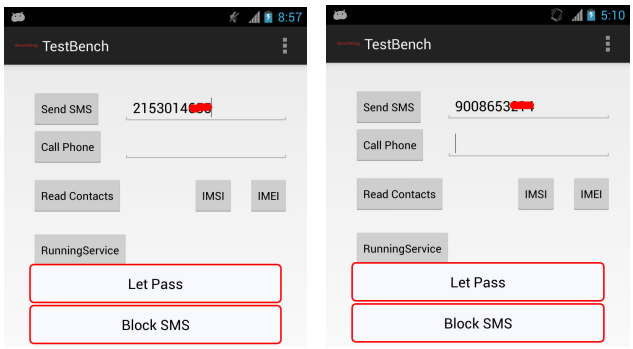


(a) Premium SMS (b) Premium Phone Call

Fig. 8. Premium SMS and Phone Call Notifications

C. Experiments with Real Malware Sending Premium SMS

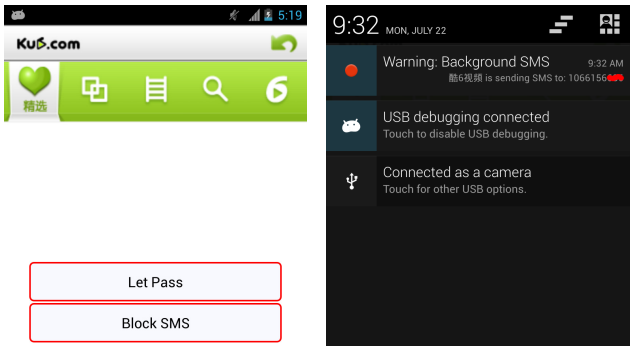
We also ran experiments with some real-world malware that send premium SMS messages. The details are given below.



(a) Background SMS (b) Premium SMS

Fig. 9. Blocking Background/Premium SMS

1) HippoSMS: HippoSMS is a premium SMS-sending malware family listed by Jiang's research team at NCSU [15]. This malware family can cause additional phone charges by sending SMS messages to a hard-coded premium number (1066******) located in China. Jiang *et al.* tested the malware using several leading mobile Anti-Virus software. However, none of them detected this malware. Even though it was published in the alternative Chinese Android App markets, HippoSMS could affect many Android users in China because most users in China download free Apps from third party Android markets. We tested our framework with one sample of this malware family. Our experimental data shows that Droid Sentinel can detect and block SMS messages sent by HippoSMS. Figures 10a and 10b demonstrate the malware and the notification when premium SMS messages are sent in the background.

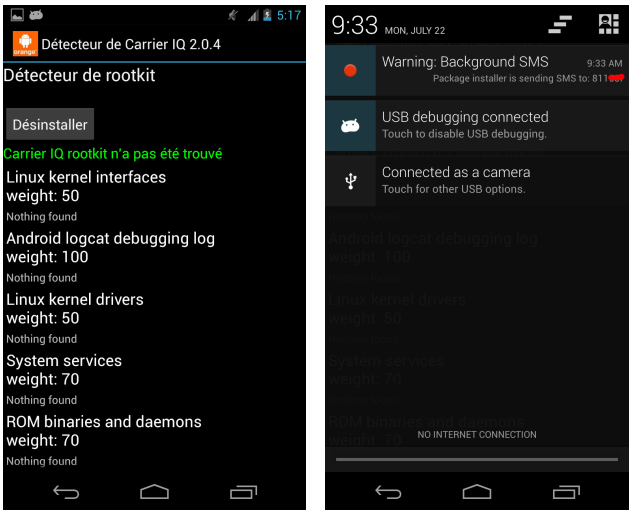


(a) Hippo Malware Sample (b) Hippo Initiated SMS Notification

Fig. 10. Experiment with a Premium SMS Malware - HippoSMS

2) Android.Qicsomos: Android.Qicsomos is a modified version of an open source project which tends to detect Carrier IQ on a device. It has additional code to send SMS to a premium-rate number located in France. Analysis of the blog from Symantec gives some details about the malware [16]. The malicious code will be triggered when the user presses the "uninstall" button to remove the App. Once pressed, SMS messages are sent to 81168 which is a premium-rate number. The Trojan follows up by executing an uninstall routine to remove the App. Figures 11a and 11b demonstrate the malware and the corresponding notification generated by

our tool indicating that the background premium number SMS is sent. In our experiment, the notification appears immediately after the uninstall button is pressed. This is consistent with the analysis from Symantec [16]. In other words, the malware immediately sends premium SMS messages when the uninstall button is clicked.



(a) Qicsomos Malware Sample (b) Qicsomos Initiated SMS Notificaton

Fig. 11. Experiment with a Premium SMS Malware - Android.Qicsomos

V. Lessons Learned

In this paper, we designed an efficient and effective online malware detection scheme based on Android's UID mechanism. Through the implementation of our scheme and the experiments on existing premium SMS-sending malware, we learned lessons from both the system and the application aspects. Below are some of the lessons that we learned:

- Some system APIs such as SMS sending APIs can be misused by malware authors to perform malicious activities. Hence, API designers and/or Android system developers should add more constraints for using these powerful APIs.
- We found that nearly all of the premium SMS sending malware requests some form of SMS related permissions. Hence, a user should be cautious when he/she installs Apps that request SMS related permissions such as WRITE_SMS and SEND_SMS. Special care must also be taken when Apps are installed from alternative Android Apps Markets because these may be Trojans that can send premium messages.
- We found that most SMS sending actions are triggered by pressing buttons such as "INSTALL" or "PROCEED" . Hence, a user should be careful when he/she presses buttons on newly installed Apps.

VI. Related Work

In the following, we review related literature. There are a number of detection approaches to defend against Android malware. For example, Zhou *et al.* proposed a permission-based behavioral footprint scheme and a heuristics-based filtering scheme to detect both new samples of known Android malware families and unknown malware families [29]. Their proposed schemes were implemented in DroidRanger which is a scalable and efficient offline analysis tool. DroidRanger uses permissions to filter out some good Apps that do not request specific permissions. In their investigation, in order to send premium SMS messages without a user's knowledge, malware needs to monitor received SMS messages and remove billing-related messages. Hence, they used the SEND_SMS and RECEIVE_SMS permissions to filter out most of the SMS related malware. Nevertheless, this permission and behavioral footprint based detection scheme can be circumvented by malware with root permissions, native code, or obfuscated code. Differently, our UID-based detection scheme deployed on real mobile platforms generates immediate notifications when malicious activities are detected. We also implemented the online detection scheme on real Android smartphones. Our experimental data shows that our developed scheme can detect and block background/premium SMS messages and phone calls in real time, leading to the protection of users from financial loss and privacy leakage. In addition, our work belongs to a real time scheme, which is orthogonal to offline schemes (e.g. [29]).

Most of the static malware detection techniques are permission and/or signature based. More complex analysis by information flow and data flow analysis on the App source code can also protect the data and enhance the security of the Android system. Examples that fall in this category are TaintDroid [30] and D2Taint [31]. In these proposed schemes, in order to keep track of the data, data needs to be marked and monitored by using tags to detect information leakage at run time. Because of this, these schemes are not energy-efficient and are not suitable for resource-constrained platforms such as mobile phones and tablets. Differently, our detection scheme is triggered only when SMS and phone call requests are received from the telephony service of the Android system, leading to less energy consumption.

For online malware detection, Enck *et al.* proposed Kirin [32] to conduct security protection during application installation time. Particularly, Kirin injects security rule checking into the Android Application Installer and supplies certificates during installation time. Instead of using general rules, Kirin relies on well-constructed security rules. Nevertheless, defining security rules for Kirin requires security experts with a thorough understanding of threats and existing protection mechanisms in Android systems. Differently, our scheme uses UID-sequence based rules and does not need experts to define the rules. The signatures can be automatically generated by the profiling process of Droid Sentinel.

As shown in [33], more and more vulnerabilities in the Android kernel have been disclosed. Hence, it is urgent and crucial to protect Android at the system level. To this end, CrowdDroid [34] presented a framework that keeps monitoring Linux system calls and uses these calls as detection features. They used k-means clustering algorithms to differentiate benign Apps from malicious Trojan Apps. Note that monitoring system calls consumes copious amounts of energy and resources from a mobile device. Isohara, Takemori *et al.*

proposed a kernel log analysis framework to detect malicious activities [35]. They used behavioral signatures to match log information. Nevertheless, there are only a limited number of signatures. Differently, we use a generic feature of Android systems, the UID-sequence, to generate signatures and use signatures to carry out malware detection online.

VII. Conclusions

In this paper, we investigated background/premium SMS message and phone-call related malware on the Android platform. By exploiting Android's application isolation mechanism that assigns unique user IDs (UID) to each Application and system service, we designed a UID-sequence based online malware detection scheme. We implemented our online detection system (Droid Sentinel) on real Android platforms and smartphones. Droid Sentinel is an online detection tool that detects and blocks background/premium SMS messages and phone calls in real time. Our experimental data on real phones, real cellular networks, and real malware demonstrated that Droid Sentinel can effectively detect/block background/premium SMS messages or phone calls. In our experiments, Droid Sentinel detected and blocked all the malware samples that we downloaded from the Internet. As ongoing work, we are extending our developed system to detect other types of malware such as those that take photos in the background.

Acknowledgment

This research was supported in part by the US National Science Foundation (NSF) under grants CNS-1022552, CNS-1065444, and CNS-1156574.

References

[1] "Android," http://www.android.com/.

[2] I. D. Corporation, "Apple cedes market share in smartphone operating system market as android surges and windows phone gains," http://www.idc.com/getdoc.jsp?containerId=prUS24257413.

[3] S. Analytics, "Samsung captures 95 percent share of global android smartphone profits in q1 2013," https://www.strategyanalytics.com/default.aspx?mod=pressreleaseviewer&a0=5362.

[4] I. D. Corporation, "Mobile threat report july-september 2013," http://www.f-secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q3_2013.pdf.

[5] "Geinimi," http://www.symantec.com/security_response/writeup.jsp?docid=2011-010111-5403-99.

[6] X. Jiang, "Droidkungfu," http://www.csc.ncsu.edu/faculty/jiang/DroidKungFu.html.

[7] X. Jiang, "Anserverbot," http://www.csc.ncsu.edu/faculty/jiang/AnserverBot/.

[8] JoeSecurity, "Analyzing "android-trojan/fakeinst": Plug & play premium sms fraud," http://joe4mobile.blogspot.com/2013/10/analyzing-android-trojanfakeinst-plug.html.

[9] Symantec, "Malware charges a fee for free apps on google play," http://www.symantec.com/connect/blogs/malware-charges-fee-free-apps-google-play.

[10] A. ThreatLabs, "Android/smssend," http://www.avgthreatlabs.com/virus-and-malware-information/info/android-smssend/.

[11] R. Johnson, Z. Wang, C. Gagnon, and A. Stavrou, "Analysis of android applications' permissions," in *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*. IEEE, 2012, pp. 45–46.

[12] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, "Droidmat: Android malware detection through manifest and api calls tracing," in *2012 Seventh Asia Joint Conference on Information Security*. IEEE, 2012, pp. 62–69.

[13] Symantec, "Obfuscating embedded malware on android," http://www.symantec.com/connect/blogs/obfuscating-embedded-malware-android.

[14] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, may 2012, pp. 95 –109.

[15] X. Jiang, "Security alert: New android malware – hipposms," http://www.csc.ncsu.edu/faculty/jiang/HippoSMS/.

[16] I. Asrar, "The day after the year in mobile malware?" http://www.symantec.com/connect/blogs/day-after-year-mobile-malware.

[17] "Detailed analysis of android.arspam," http://forensics.spreitzenbarth.de/2011/12/22/detailed-analysis-of-android-arspam/.

[18] X. Jiang, "Security alert: New beanbot sms trojan discovered," http://www.csc.ncsu.edu/faculty/jiang/BeanBot/.

[19] Symantec, "Android.gamex," http://www.symantec.com/security_response/writeup.jsp?docid=2012-051015-1808-99.

[20] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," Department of Computer Science, The University of Auckland, New Zealand, Tech. Rep., 1997.

[21] V. Rastogi, Y. Chen, and X. Jiang, "Droidchameleon: evaluating android anti-malware against transformation attacks," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, 2013, pp. 329–334.

[22] "Android open source project," http://source.android.com/source/index.html/.

[23] "Android security overview," http://source.android.com/devices/tech/security/.

[24] "Android uids and gids," http://android-dls.com/wiki/index.php?title=Android_UIDs_and_GIDs.

[25] "Localserversocket," http://developer.android.com/reference/android/net/LocalServerSocket.html.

[26] "Parcel — android developers," http://developer.android.com/reference/android/os/Parcel.html.

[27] "Contagio," http://contagiominidump.blogspot.com/.

[28] "Android debug bridge," http://developer.android.com/tools/help/adb.html.

[29] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets," in *Proceedings of the 19th Annual Network and Distributed System Security Symposium*, 2012.

[30] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, 2010, pp. 1–6.

[31] B. Gu, X. Li, G. Li, A. C. Champion, Z. Chen, F. Qin, and D. Xuan, "D2taint: Differentiated and dynamic information flow tracking on smartphones for numerous data sources," Technical Report, Tech. Rep., 2012.

[32] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 235–245.

[33] X. Hei, X. Du, and S. Lin, "Two vulnerabilities in android os kernel," in *Proceedings of the IEEE International Conference on Communications*, 2013.

[34] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011, pp. 15–26.

[35] T. Isohara, K. Takemori, and A. Kubota, "Kernel-based behavior analysis for android malware detection," in *Proceedings of the 2011 Seventh International Conference on Computational Intelligence and Security*, 2011, pp. 1011–1015.