Dissecting SMS Malwares in Android

Anoop Joseph Babu, Rahul Raveendranath, Venkiteswaran Rajamani
College of Engineering, Trivandrum, India
{manoopjoseph, rahul92, venkit07}@gmail.com

Soumya Kanti Datta
EURECOM
Biot, France
Soumya-Kanti.Datta@eurecom.fr

Abstract—Android is the most widely used operating system which spans variety of smartphones, tablets and wearable devices. Since it is open source, developers can take full advantage of the extensive number of APIs in the framework. But the popularity and openness of the android made it a favorite target of malware authors. This paper focuses on the impact of some design decisions in framework which contributes in making Android applications vulnerable. A proof of concept SMS malware is presented to analyze the working of most threatening SMS malwares in the wild. This malware sends service messages to telecom operators and incurs charges or transfer of funds. It uses the vulnerability in ordered broadcast intent system to remains stealthy by intercepting and aborting possible notifications from telecom operators. Countermeasures to mitigate this security leaks are also discussed.

Index Terms—Android; Security threats; Countermeasures; SMS Malware; Permissions.

I. INTRODUCTION

Smartphones are becoming increasingly ubiquitous. According to a research report by IDC, out of a total base of 211.6 million smartphone units shipped during the first quarter of 2014, Android accounted for 81.0% of all smartphone shipments [1]. As a result of this rising popularity of the platform, android has become one of the favorite platforms for cyber criminals. Cyber criminals often create and distribute Trojan viruses that infect a victim's mobile phone, so that the phone makes calls or sends text messages without the user's knowledge. The malware directs these unauthorized calls to premium-charge numbers or to chargeable text services that are operated by the criminal. By infecting large number of phones and getting each phone to make several calls or send several texts, the criminal can generate a significant revenue stream.

In this paper we have created a proof of concept SMS malware, RogueSMS which depicts the working of a generic SMS malware. We demonstrate how android-based smartphones can be exploited by deploying a malware that uses the SMS service as its medium of operation. The vulnerabilities of the telecom operator as well as the Android subsystem are used to harness revenue from these malwares. A good number of mobile operators use text messaging to transfer units/credits between two mobile users without requiring any form of validation/authorization beyond the message being sent from the phone. The units/credits refer to the user balance or airtime that can be used to make phone calls, to send/receive SMS messages, or to access the Internet. For example, a user who wants to transfer units builds a structure-defined text by entering two elements: the amount they want to transfer from their balance and the mobile number of the beneficiary. After drafting the message, the user sends it to a specific number and the transaction is completed by the operator through balance transfer. This can be extended to other potential financial transactions that are made through SMS as well.

The rest of the paper is structured as follows. In section II, we study the Android system and its vulnerabilities. Thereafter, an overview of the proof of concept SMS malware, RogueSMS, is presented in section III. This is followed by Section IV which deals with possible countermeasures. The section V gives a brief outline of SMS malwares in the wild. Thereafter, the paper concludes in Section VI.

II. VULNERABILITY STUDY

Android is an open development platform. It offers developers the capability to build rich and innovative applications. Developers are free to have superiority of the device hardware, access location information, run background service, set alarm and so on. Developers have full access to same framework and the core APIs. This characteristic also makes the android system vulnerable.

A. Android application components

Android defines four component types –

- **Activities:** It is the components which define the user interface of application. Typically, an application developer defines one activity per screen. Activities start each other, possibly passing and returning values. Only one activity on the system has keyboard and processing focus at a time, all others are suspended.
- **Services**: This component performs background processing. When an activity needs to perform some operation that must continue after the user interface disappears (an example is playing of music in background), it commonly starts a service specifically designed for that action. Usually user cannot keep track of the services running in an instance. There will be plenty of services running in background. This helps malware authors to write malicious parts of app which stay alive in background, and is hidden from user.

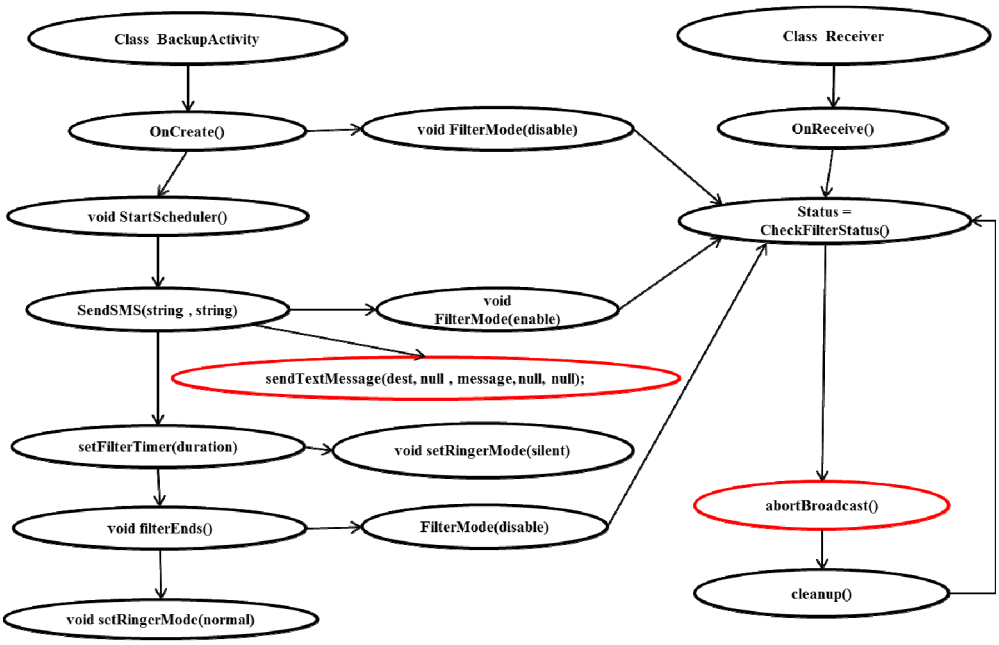

Figure 1: Partial data dependence graph for malware part of RogueSMS

- **Content Providers:** Content provider components store and share data using a relational database interface. Each content provider has an associated "authority" describing the content it contains. Other components use the authority name as a handle to perform SQL queries (such as SELECT, INSERT, or DELETE) to read and write content. Any application can access content providers. The only security measure is that it has to register its affiliation in manifest file. Since content providers are global service to all apps, any changes made are visible system wide. Like deleting an SMS from the SMS content provider will flush it from other SMS apps. Thus enabling malware authors to selectively delete notification SMS sent by service providers.
- **Broadcast Receiver**: A broadcast receiver is a component that responds to system-wide broadcast announcements. They act as mailboxes for messages from other applications. Commonly, application code broadcasts messages to an implicit destination. Broadcast receivers thus subscribe to such destinations to receive the messages sent to it. The two major classes of broadcasts are – (i) **Normal broadcasts** are completely asynchronous. All receivers of the broadcast are run in an undefined order, often at the same time and (ii) **Ordered broadcasts** are delivered to one receiver at a time. As each receiver in ordered broadcast executes in turn, it can propagate a result to the next receiver, or it can completely abort the broadcast so that it won't be passed to other receivers. The order receivers run in can be controlled with the *android: priority* attribute of the matching intent-filter; receivers with the same priority will be run in an arbitrary order. The `SMS_RECEIVED` intent is an ordered broadcast. So manipulation of intent is easily possible since the intent flows through app as per the priority order defined in the Android manifest file. Thus this mechanism enables the high priority applications to edit or abort the contents of intent before passing it to receivers with lower priority. This feature of ordered broadcast is used widely by malware authors to modify original data before it reaches the legitimate applications.

III. Threat Model

A. Application Design

RogueSMS is a proof of concept SMS malware which can exploit the permission subsystem and ordered broadcast subsystem of android. The malware sends SMS to premium numbers or performs fund transfer and when notification arrives, it deletes the notification so that user is unaware of the attack. Notification by operator stating the deduction of balance or fund transfer is the only way in which user gets to know about the attack. This is taken care of by intercepting the SMS intent and aborting it so that other applications such as messaging will not receive this. The RogueSMS application has a main activity, a sender service and a sms filter service.

- **Main Activity** – The main activity acts as a backup and restore activity. This forms the graphical interface of the application. The main activity triggers the sender service and listener service at least once. The sender service is then scheduled to run at predefined intervals. This is achieved using handlers and runnables.
- **Sender Service** - The sender service automatically runs on the fixed time intervals scheduled by the Main Activity. This service sends SMS to the predefined number requesting for activating a premium service or fund transfer. The behavior is predefined in the malicious code. After sending the SMS, it starts the sms filter service to monitor and delete the reply sent by the operator notifying the fund deduction.

```
<receiver
    android:name=".filter1"
    android:exported="true" >
    <intent-filter android:priority="999" >
        <action android:name="android.provider.Telephony.SMS_RECEIVED" >
        </action>
    </intent-filter>
</receiver>
```

Figure 2: Snapshot of AndroidManifest.xml

- **Filter Service** - This service runs in background for few minutes monitoring all the messages received by the device. The service is registered with highest priority on `SMS_RECEIVED` intent. Figure 1 shows that filter service is registered with maximum possible priority. Since `SMS_RECEIVED` intent is an ordered broadcast, the filter service gets the intent soon after the SMS reaches the radio. The filter service then extracts the message body and checks for the degree of occurrence of characteristic words that define the notification SMS. It flags the message as red or white by checking if the received SMS is a notification message sent by the operator or not. If flagged red, the filter service will abort the intent. When an ordered intent is aborted by the highest priority receiver, it will not reach other receivers who have registered for the intent. Thus the SMS will not reach default messaging application which stores it in the content provider. So the aborted SMS will not be populated in the sms content provider. If the message is flagged white, the filter service will pass the intent to next receiver without modification. The lifespan of filter service is 10 mins after its triggering. This interval is more than enough to receive possibly all service messages from operator resulted due the malicious activity of sender service.

B. Target Analysis

The RogueSMS can use the broadcast receiver and permission subsystem of android. But on Android 4.4, only one app can receive the new SMS_DELIVER_ACTION intent, which the system broadcasts when a new SMS message arrives. This is usually the default messaging application selected by the user. Other apps receive the new SMS through the SMS_RECEIVED_ACTION broadcast intent when it arrives. This new changes to the SMS intent prevents the filter service from intercepting the message. But this change is not passed to the predecessors of Android 4.4. Referring to the Fig 2.2, only 13.6% of devices run Android 4.4. The rest runs on lower versions which are vulnerable to this attack.

Version	Codename	API	Distribution
2.2	Froyo	8	0.8%
2.3.3 - 2.3.7	Gingerbread	10	14.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	12.3%
4.1.x	Jelly Bean	16	29.0%
4.2.x		17	19.1%
4.3		18	10.3%
4.4	KitKat	19	13.6%

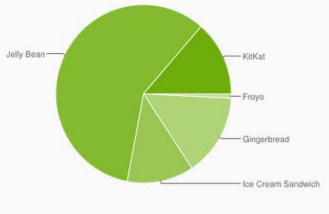

Figure 3: Data collected during a 7-day period ending on June 4, 2014 from Google Play store

C. Characteristics

The permissions required by the malicious sender and filter services are SEND_SMS and RECEIVE_SMS. These, along with READ_SMS and WRITE_SMS required for the backup and restore activity, are the total permission required by the application. The users see it as default permissions requested by any message related application. This human tendency is highly favorable to malicious apps.

In order to remain stealthy, the sender service does take necessary precautions. After sending the malicious SMS, the sender service clears any USSD messages that may appear notifying the user about SMS charges. The service then triggers the filter service. The filter service then sets the device to silent mode and listens to possible reply from the operator about fund transfer. When it receives the required message, the message is discarded and the device is reverted back to its normal mode. The victim will only know about the balance deduction through their monthly bill, if they do a close inspection. In order to increase stealthiness, fund transfer of small amount is done.

IV. Counter Measures

In this section we propose practical measures to overcome RogueSMS and other SMS based malwares.

A. Clustering of SMS based malwares

Clustering of SMS based applications using their permissions [2] can reveal if the application is requesting permissions other than those in the same cluster. If so, it may be deploying some malicious code. Dong-Jie et al [3] discuss a static analysis of detecting malware behaviour. Static behaviour is extracted from permission requested through manifest files, Intents, Inter-Component communication & API calls from byte code. K-means and EM clustering algorithms are used to classify the nature of app. Schmidt et al [4] extract function calls from binaries of applications, and apply their clustering mechanism, called Centroid, for detecting unknown malware.

B. Behaviour based detection

Behaviour based detection uses dynamic methods such as testing the app in an actual environment, using sandboxing or emulation etc. Bose et al[5] present a behavioral detection framework. Instead of the signature-based solutions, they detect mobile malware by observing the logical ordering of an application's actions. They discriminate the malicious behaviour of malware from the normal behaviour of applications by training a classifier based on Support Vector Machines (SVMs). Also, sandboxing the application and performing dynamic analysis reveals the malicious behaviour of the application. AASandbox system proposed by Thomas et al[6] is a sandboxing technique which does static and dynamic analysis of application.

C. Other methods

The problem of ordered broadcast receivers can be overcome by using a trusted middleware application which has the highest priority to SMS_RECEIVED intent. When new SMS arrives, the middleware app supplies the intent to the

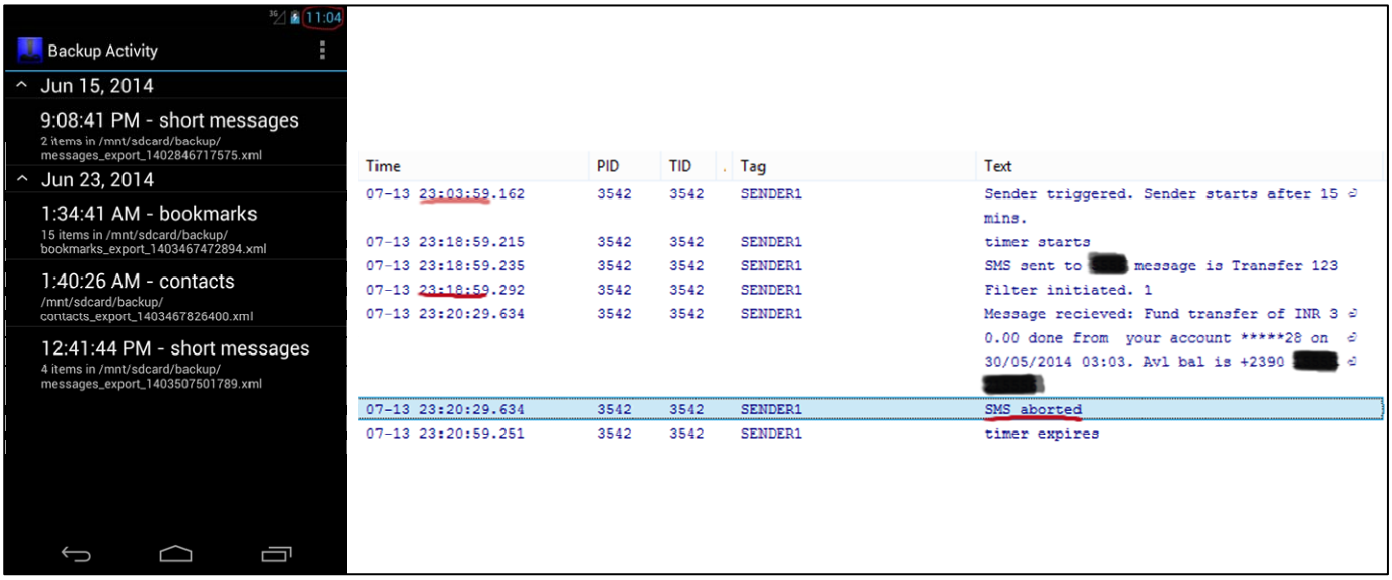

Figure 4: Shows starting the backup activity triggers the malware. The malicious SMS is scheduled to send 15mins after triggering. Soon after deploying SMS, the filter starts to delete notifications. The time lapse of the filter is predefined to 2 minutes. Android logcat view of the attack is given above.

default messaging app which then stores the SMS in content provider. The intent is then passed normally to the next application registered for SMS_RECEIVED intent. Moreover, the user has to take cautions about installing applications in his device. It is necessary to understand the authenticity of permission requested by the application. Also, it is not safe to install applications from third party app stores. They may contain repackaged versions of legitimate apps with malicious code embedded [7]. We can also group apps in a specific store as blacklist or whitelist using crowdsourcing. Burguera et al [8] presents Crowdroid which is a crowd sourced malware profiling system. The end user can consult this list to decide if he is installing a malicious application or not.

V. State-of-the-art

William Enck et al [9] evaluated the security impact of the SMS interface on the availability of the cellular phone network. Many operators provide internet based SMS delivery which can be used to launch denial of service attacks. FakePlayer is another example that masquerades as a movie player but does not provide the advertised functionality. Rather it sends SMS to premium numbers. Another well seen exploit is spamming the user contacts. Sending SMS spam is illegal in many countries. Thus, sending from a compromised device reduces the risk of the spammer being caught. Another piece of malware, Geinimi, was set up to send premium SMS messages to numbers specified by remote commands [10]. As per Trend Micro [11] fake versions of Flappy Bird circulate online. Most of these fake versions were premium service abusers, which sent messages to premium numbers, causing affected users to incur unwanted billing charges. HippoSMS [12] is another prominent malware found in Chinese app stores on repackaged apps. It sends SMS messages to a hardcoded premium number without the user's knowledge. Spam-SMS floods predefined SMS to user contacts. Also usage pattern based malwares have been demonstrated in [13]. SMS based malware emulating that method would be almost impossible to detect by current countermeasures.

VI. Conclusion

The RogueSMS discussed in this paper tried to exploit the vulnerabilities of the Android platform and telecom operators that are harnessed by those malwares mentioned in section V. We tried to put on the shoes of malware authors and suggested countermeasures for the attacks. Our analysis revealed that the broadcast receiver system and the permission system of the android is the root cause of vulnerability. The intent delivery of new SMS received can be easily intercepted and manipulated. Here, we were able to hide the notification SMS sent by the telecom operator about the fund transfer. To prevent these attacks, the user has to be cautious about the permission they grant to the applications they install. Using crowd sourced malware profiling systems and dynamic analysis through sandboxing can help user identify malwares. Moreover, a middleware which can monitor the intent delivery is necessary.

VII. References

[1] "Android Pushes Past 80% Market Share While ... - IDC." 2013. 10 Sep. 2014 http://www.idc.com/getdoc.jsp?containerId=prUS24442013

[2] Jiang, Danyang et al. "A security assessment method for Android applications based on permission model." Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on 30 Oct. 2012: 701-705.

[3] Wu, Dong-Jie et al. "Droidmat: Android malware detection through manifest and API calls tracing." Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on 9 Aug. 2012: 62-69.

[4] Schmidt, A-D et al. "Detecting symbian os malware through static function call analysis." Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on 13 Oct. 2009: 15-22.
[5] Bose, Abhijit et al. "Behavioral detection of malware on mobile handsets." Proceedings of the 6th international conference on Mobile systems, applications, and services 17 Jun. 2008: 225-238.
[6] Blasing, Thomas et al. "An android application sandbox system for suspicious software detection." Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on 19 Oct. 2010: 55-62.
[7] Zhou, Yajin et al. "Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets." Proceedings of the 19th Annual Network and Distributed System Security Symposium Feb. 2012: 5-8.
[8] Burguera, Iker, Urko Zurutuza, and Simin Nadjm-Tehrani. "Crowdroid: behavior-based malware detection system for android." Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices 17 Oct. 2011: 15-26.
[9] Enck, William et al. "Exploiting open functionality in SMS-capable cellular networks." *Proceedings of the 12th ACM conference on Computer and communications security* 7 Nov. 2005: 393-404.
[10] Felt, Adrienne Porter et al. "A survey of mobile malware in the wild." *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices* 17 Oct. 2011: 3-14.
[11] Trend Micro Threat Encyclopedia | Latest information on ..." 2012. 13 Jun. 2014. http://aboutthreats.trendmicro.com/us/mobile/monthly-mobile-review/2014-04-malware-in-apps-clothing
[12] Elish, Karim O, Danfeng Yao, and Barbara G Ryder. "User-centric dependence analysis for identifying malicious mobile apps." Workshop on Mobile Security Technologies May. 2012.
[13] S. K. Datta, C. Bonnet and N. Nikaein, "Usage pattern based security attacks for smart devices," In International Conference on Consumer Electronics – Berlin (ICCE-Berlin 2014) [Accepted].