International Journal of Advance Engineering and Research Development
Scientific Journal of Impact Factor (SJIF): 4.72
Special Issue SIEICON-2017,April -2017
e-ISSN : 2348-4470 p-ISSN : 2348-6406

New Tracking Rootkit at Application Layer in Android

Inzmamul Huq Navab[1], Nikita Bhangadiya[2], Parth Pateliya[3], Pooja Tandel[4] , Kisori Shekokar[5]

[1]*Computer Engineering, Sigma Institute of Engineering*
[2]*Computer Engineering, Sigma Institute of Engineering*
[3]*Computer Engineering, Sigma Institute of Engineering*
[4]*Computer Engineering, Sigma Institute of Engineering*
[5]*Computer Engineering, Sigma Institute of Engineering*

Abstract — *Nowadays protecting Data is most preventing issue in mobiles devices. All computer related thing is going to adopt portability. There is greater chance to be stolen mobile devices. We are going to constitute Rootkit which will work in System App of mobile device. We will attempt to send location coordinate (GPS signals) using that Rootkit and Thief's inserted SIM card number to predefined destination number.*

Keywords*-Tracking; GPS; covert channel; mobile phone; sms; rootkit; covertly; SystemUI; Injection; compilation; decompile*

I. INTRODUCTION

Android is open sources and uses on daily basis. According to analysis, if thief stoles mobile, he always replacing his own SIM card. In this research, we have designed such a mechanism of rootkit which detect new SIM card of thief and send GPS coordinates and mobile number of thief via SMS using that SIM card covertly. A Destination address for SMS will be predefined in rootkit.

Before Discussing Actual research, we have to go through basic knowledge of Android OS. Android source are available at Git version control repository.

A. Android Architecture

Android Architecture is based on layered architecture. Each of depth level of layers have own permissions and access. Android uses Java API Framework contains all necessary functionalities required by Application layer. At lower layer, android uses Linux drivers and Linux Kernel module. It also has native C and C++ library support and Android runtime ART for effective compilation of java and xml integration in executable manner.

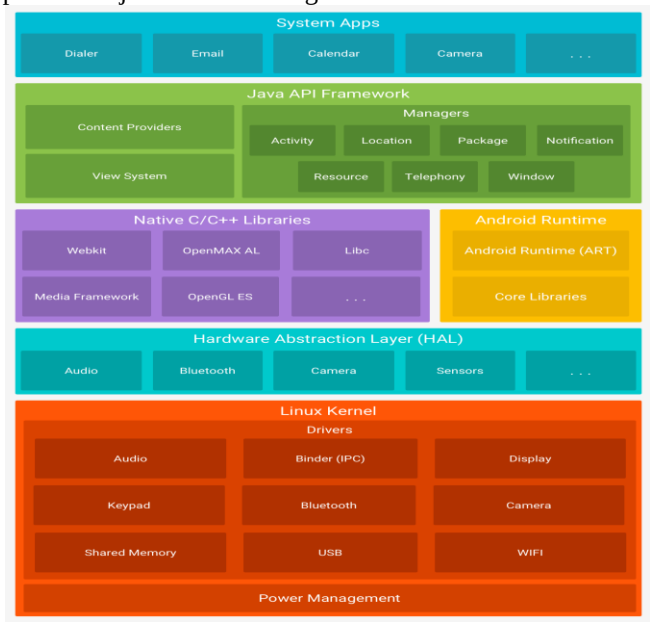


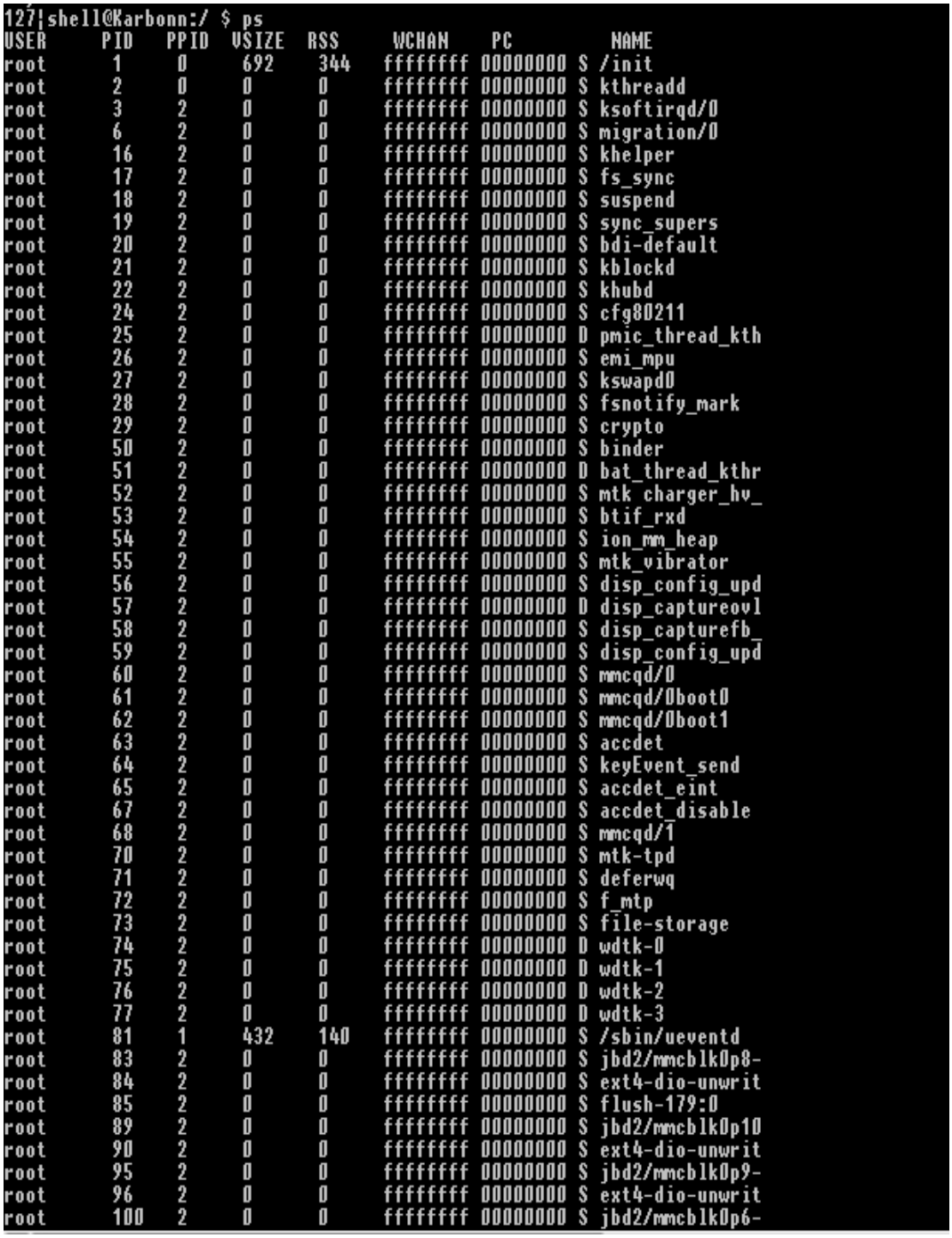
Figure 1: ***Android Architecture [9]***

B. Android Permission and Linux Sandbox

Android uses Linux based sandbox .In Linux, Each processes have particular GID (Group ID) or UID (User ID). Android is slightly different from Linux, it uses Aid (Android application id).As shown in Fig 2, PS shell command gives all process which currently running in android. Here PID and PPID refer as process id and parent process id respectively. Here if PID and PPID are same then both are refers from same process ID or we can say as same GID.

Android protects hardware resources, to achieve this goal it divides whole system in access privileges. This access privileges are known as permissions. There are mainly three type of permissions.1.normal 2.dangorous 3.Signature.As shown in fig 3, we have listed all permissions using pm list permission command.

In android, there is ANDROID_FILESYSTEM_CONFIG.h (fig 4) file which defines all type of static and dynamic PIDs. In example, every system daemons and system services have predefined AID in android and All application which installs by users have start AID from 10,000.



```
127|shell@Karbonn:/ $ ps
USER      PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      1     0     692    344   ffffffff 00000000 S /init
root      2     0     0      0     ffffffff 00000000 S kthreadd
root      3     2     0      0     ffffffff 00000000 S ksoftirqd/0
root      6     2     0      0     ffffffff 00000000 S migration/0
root      16    2     0      0     ffffffff 00000000 S khelper
root      17    2     0      0     ffffffff 00000000 S fs_sync
root      18    2     0      0     ffffffff 00000000 S suspend
root      19    2     0      0     ffffffff 00000000 S sync_supers
root      20    2     0      0     ffffffff 00000000 S bdi-default
root      21    2     0      0     ffffffff 00000000 S kblockd
root      22    2     0      0     ffffffff 00000000 S khubd
root      24    2     0      0     ffffffff 00000000 S cfg80211
root      25    2     0      0     ffffffff 00000000 D pmic_thread_kth
root      26    2     0      0     ffffffff 00000000 S emi_mpu
root      27    2     0      0     ffffffff 00000000 S kswapd0
root      28    2     0      0     ffffffff 00000000 S fsnotify_mark
root      29    2     0      0     ffffffff 00000000 S crypto
root      50    2     0      0     ffffffff 00000000 S binder
root      51    2     0      0     ffffffff 00000000 D bat_thread_kthr
root      52    2     0      0     ffffffff 00000000 S mtk charger_hv_
root      53    2     0      0     ffffffff 00000000 S btif_rxd
root      54    2     0      0     ffffffff 00000000 S ion_mm_heap
root      55    2     0      0     ffffffff 00000000 S mtk_vibrator
root      56    2     0      0     ffffffff 00000000 S disp_config_upd
root      57    2     0      0     ffffffff 00000000 D disp_captureovl
root      58    2     0      0     ffffffff 00000000 S disp_capturefb_
root      59    2     0      0     ffffffff 00000000 S disp_config_upd
root      60    2     0      0     ffffffff 00000000 S mmcqd/0
root      61    2     0      0     ffffffff 00000000 S mmcqd/0boot0
root      62    2     0      0     ffffffff 00000000 S mmcqd/0boot1
root      63    2     0      0     ffffffff 00000000 S accdet
root      64    2     0      0     ffffffff 00000000 S keyEvent_send
root      65    2     0      0     ffffffff 00000000 S accdet_eint
root      67    2     0      0     ffffffff 00000000 S accdet_disable
root      68    2     0      0     ffffffff 00000000 S mmcqd/1
root      70    2     0      0     ffffffff 00000000 S mtk-tpd
root      71    2     0      0     ffffffff 00000000 S deferwq
root      72    2     0      0     ffffffff 00000000 S f_mtp
root      73    2     0      0     ffffffff 00000000 S file-storage
root      74    2     0      0     ffffffff 00000000 D wdtk-0
root      75    2     0      0     ffffffff 00000000 D wdtk-1
root      76    2     0      0     ffffffff 00000000 D wdtk-2
root      77    2     0      0     ffffffff 00000000 D wdtk-3
root      81    1     432    140   ffffffff 00000000 S /sbin/ueventd
root      83    2     0      0     ffffffff 00000000 S jbd2/mmcblk0p8-
root      84    2     0      0     ffffffff 00000000 S ext4-dio-unwrit
root      85    2     0      0     ffffffff 00000000 S flush-179:0
root      89    2     0      0     ffffffff 00000000 S jbd2/mmcblk0p10
root      90    2     0      0     ffffffff 00000000 S ext4-dio-unwrit
root      95    2     0      0     ffffffff 00000000 S jbd2/mmcblk0p9-
root      96    2     0      0     ffffffff 00000000 S ext4-dio-unwrit
root      100   2     0      0     ffffffff 00000000 S jbd2/mmcblk0p6-
```

Figure 2: ***PS command using ADB SHELL***

Figure 3: ***List of Permissions in android***


```
#ifndef _ANDROID_FILESYSTEM_CONFIG_H_
#define _ANDROID_FILESYSTEM_CONFIG_H_

#include <stdint.h>
#include <sys/cdefs.h>
#include <sys/types.h>

#if defined(__ANDROID__)
#include <linux/capability.h>
#else
#include "android_filesystem_capability.h"
#endif

#define CAP_MASK_LONG(cap_name) (1ULL << (cap_name))

/* This is the master Users and Groups config for the platform.
 * DO NOT EVER RENUMBER
 */

#define AID_ROOT 0 /* traditional unix root user */

#define AID_SYSTEM 1000 /* system server */

#define AID_RADIO 1001          /* telephony subsystem, RIL */
#define AID_BLUETOOTH 1002      /* bluetooth subsystem */
#define AID_GRAPHICS 1003       /* graphics devices */
#define AID_INPUT 1004          /* input devices */
#define AID_AUDIO 1005          /* audio devices */
#define AID_CAMERA 1006         /* camera devices */
#define AID_LOG 1007            /* log devices */
#define AID_COMPASS 1008        /* compass device */
#define AID_MOUNT 1009          /* mountd socket */
#define AID_WIFI 1010           /* wifi subsystem */
#define AID_ADB 1011            /* android debug bridge (adbd) */
#define AID_INSTALL 1012        /* group for installing packages */
#define AID_MEDIA 1013          /* mediaserver process */
#define AID_DHCP 1014           /* dhcp client */
#define AID_SDCARD_RW 1015      /* external storage write access */
#define AID_VPN 1016            /* vpn system */
#define AID_KEYSTORE 1017       /* keystore subsystem */
#define AID_USB 1018            /* USB devices */
#define AID_DRM 1019            /* DRM server */
#define AID_MDNSR 1020          /* MulticastDNSResponder (service discovery) */
```

Figure 4: ***Reserved AID for process***

C. Android Development environment

Android development environment is not so developed even there is not rigid guide or documentation about android Source. Even android application development is rich and documented. ADB is android debugging bridge uses as communication between android mobile and computer via USB. ADB provides facility about shell and file system access or debugging facility. ADB have PULL or PUSH command. Hence we can easily insert file or retrieve file from android. Android studio is another development IDE which helps as to compile packages and debug facility provided by GOOGLE cooperation.

D. System Application and SystemUI.apk

System applications are resides in /system/apps path in android. This apps are inbuilt by manufacturers, basically this applications can access all permissions as defined in android policy. As shown in fig 6 system apps are responsible to handle all of basic need of android such as calling, messaging, showing GUI for widget, Handling Resources and much more.

In this entire collection of app, one app called as SystemUI.apk. This app is responsible for handling all GUI level interaction from users. SystemUI manages screens and handle activates. It resides on top of application layer. SystemUI is activity which always starts after System boot up. After android 3.1 services and boot receivers not starts in absence of activity. SystemUI is only activity which runs until android lifecycle; hence it is very suitable to implement tracking rootkit in this apk file at application level.

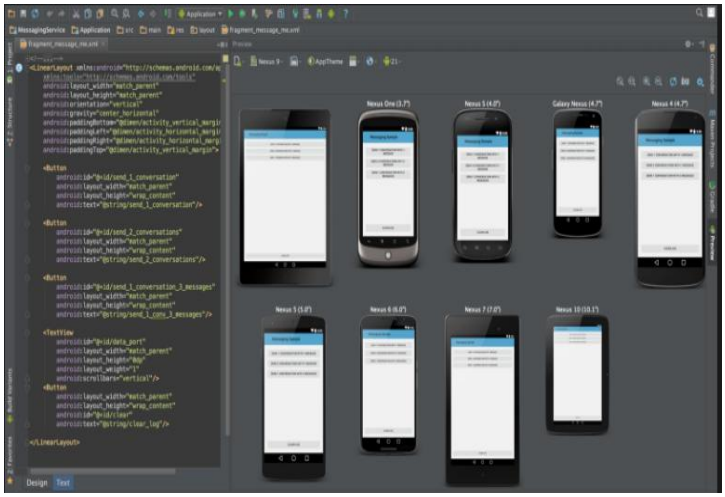


Figure 5: ***Android Studio for development [14]***

```
C:\adb>adb shell cd /system/app;ls
APKInstaller.apk
APKInstaller.odex
ApplicationGuide.apk
ApplicationGuide.odex
ApplicationsProvider.apk
ApplicationsProvider.odex
BackupRestoreConfirmation.apk
BackupRestoreConfirmation.odex
BaiduNetworkLocation.apk
BasicDreams.apk
BasicDreams.odex
BatteryWarning.apk
BatteryWarning.odex
Books.apk
CDS_INFO.apk
CDS_INFO.odex
CP_AudioElectric.apk
CP_AudioElectric.odex
CP_BrowserUI.apk
CP_BrowserUI.odex
CP_CButton.apk
CP_CButton.odex
CP_Calculator.apk
CP_Calculator.odex
CP_CalendarProvider.apk
CP_CalendarProvider.odex
CP_CalendarUI.apk
CP_CalendarUI.odex
CP_CallRecord.apk
CP_CallRecord.odex
CP_ContactsProvider.apk
CP_ContactsProvider.odex
CP_ContextMode.apk
CP_ContextMode.odex
CP_CoolShow.apk
CP_CoolShow.odex
CP_Cooperation.apk
CP_Cooperation.odex
CP_DMP.apk
CP_DMP.odex
CP_Dial.apk
CP_Dial.odex
CP_DownloadProvider.apk
CP_DownloadProvider.odex
CP_EngMode.apk
CP_EngMode.odex
CP_FackProvider.apk
CP_FackProvider.odex
CP_FactoryPattern.apk
CP_FactoryPattern.odex
CP_FactoryPatternHelper.apk
CP_FactoryPatternHelper.odex
CP_FactoryTest.apk
CP_FactoryTest.odex
```

Figure 6: ***System apps installed in my android***

II. THEORY OF BACKGROUND

Basically, there are thousands of ways to implement rootkit in android. In each release of android from 2.0 to 7.0(latest), Google has improved android architecture and security. Google has restricted permission level more tightly. Android became more robust.

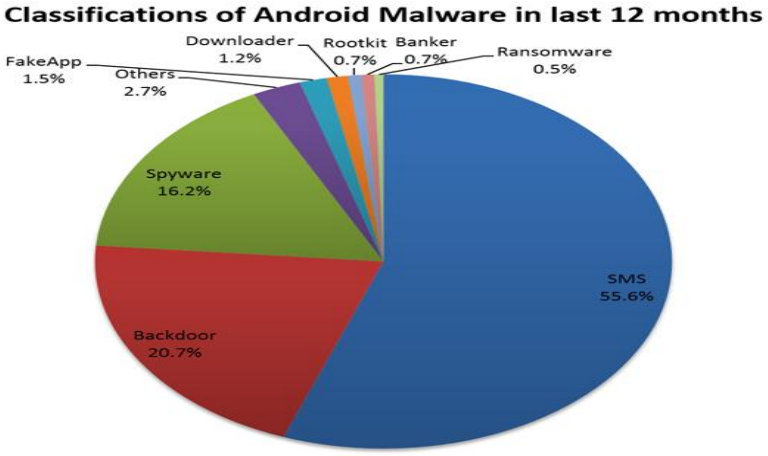


Figure 7: ***rootkit and malware analysis [10]***

A. about rootkits

Android is open source hence exploit of its vulnerability is very easy. For expert, it may easy to exploit android vulnerability.

Rootkits are chunks of program or can be program, application, system program, script, malware (Trojan).Most of root kits are injected in system level. In example, Modem Injection, TCP/IP layer hacking, Framework hijacking.

B. Types of rootkits

Rootkits are designed for different purpose. Basically all rootkit have one purpose about covey data. As shown in figure 8, Rootkits uses BOOT, SMS, CALL and most of system resources without user permissions. According to Survey most prevalent rootkit is Droid kung fu. Droid kung fu injects itself in very low level of android context.

	Installation				Activation								
	Repackaging	Update	Drive-by Download	Standalone	BOOT	SMS	NET	CALL	USB	PKG	BATT	SYS	MAIN
ADRD	√				√		√	√					
AnserverBot	√	√			√	√	√		√		√	√	
Aeroot				√									
BaseBridge	√	√			√	√	√				√	√	
BeanBot	√					√		√					
BgServ	√				√	√							√
CoinPirate	√				√	√							
Crusewin				√	√	√							
DogWars	√												
DroidCoupon	√				√		√	√		√			
DroidDeluxe				√									
DroidDream	√												√
DroidDreamLight	√				√			√					
DroidKungFu1	√				√						√	√	
DroidKungFu2	√				√						√	√	
DroidKungFu3	√				√						√	√	
DroidKungFu4	√				√						√	√	
DroidKungFuSapp	√				√						√	√	
DroidKungFuUpdate	√	√											
Endofday	√				√	√							
FakeNetflix				√									
FakePlayer				√									
GamblerSMS				√	√								
Geinimi	√				√	√							
GGTracker			√	√	√	√					√		
GingerMaster	√				√								
GoldDream	√			√	√	√		√					
Gone60				√									
GPSSMSSpy				√		√							
HippoSMS	√				√	√							√
Jifake	√		√										
jSMSHider	√									√			√
KMin				√	√								
Lovetrap				√	√	√							
NickyBot				√	√	√							
Nickyspy				√	√								
Pjapps	√				√	√						√	
Plankton		√		√									
RogueLemon				√		√							
RogueSPPush				√		√							
SMSReplicator				√		√							
SndApps				√	√								
Spitmo			√	√		√		√					
TapSnake				√	√								
Walkinwat				√									
YZHC				√	√								
zHash				√	√								
Zitmo			√	√		√							
Zsone	√					√							√
number of families	25	4	4	25	29	21	4	6	1	2	8	8	5
number of samples	1083	85	4	177	1050	398	288	112	187	17	725	782	56

Figure 8: ***Android rootkit and analysis [11]***

III. Methodology

As we discussed earlier, we are going to design Rootkit in application Layer. Since release of Android 3.2 developers had block start of Broadcast receiver and Service without Activity. Hence it became nearly impossible to implement rootkit at Application layer. But there is always way to knock .We found that there is activity named SystemUI which always runs in android context so it is possible to implement Broadcast Receiver and Start Service from SystemUI activity.

A. Basics of SystemUI and Package Manager

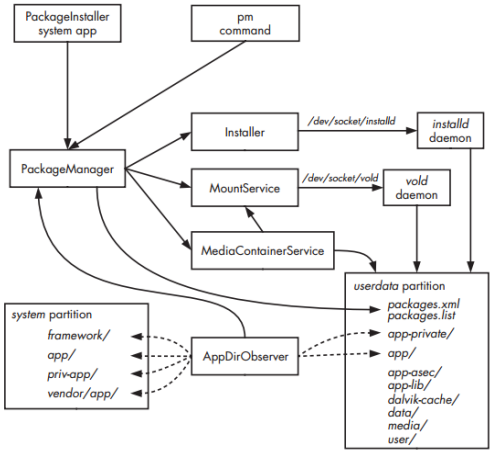


***Figure 9:** Package Manager in action [12]*

Package Manager is responsible for installing and uninstalling Application. Package Manager have daemons such as installd, voLd .which manages User apps and system apps.

```
shell@Karbonn:/ $ su
root@Karbonn:/ # grep 'com.android.systemui' /data/system/packages.list
com.android.systemui 10018 1 /data/data/com.android.systemui
```

***Figure 10:** systemUI in packages.list runs in 10018 AID*

SystemUI is package installed in /system/apps/ path as SystemUI.apk and SystemUI.odex.SystemUI manages initial launcher Activity. It manages media,screenshot,settings,status,usb UI .fig 11 shows SystemUI packages structure from Android open source project(AOSP).In AOSP,it resides in framework/packages/SystemUI.

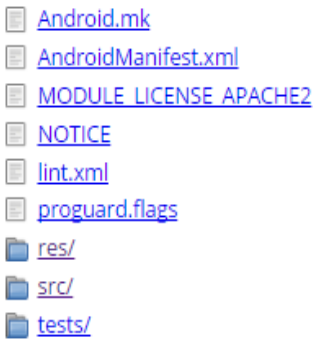


***Figure 11:** AOSP SystemUI package*

***Figure 12:** AOSP /src/android/SystemUI*

B. GPS and Telephony API

Telephony API provides facility such as call, message to user. Telephony Manager provide methods, service and state information of Telephony layer. To use android telephony service we need to use call system telephony service, In example Context.getSystemService(Context.TELEPHONY_SERVICE). Another class in Android Telephony is SMS manager which can be used as SMS sending, encoding and configuring.
sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent,PendingIntent deliveryIntent),this method is uses as to send text message. Destination address is about where to send. scAdress about from which address you need to send. A text parameter defines text message to send.

A GPS (Global positioning system) is widely uses in System. It receives signal from different gps satellite and calculate location.In android, LocationManager and LocationListner are used for gps location gaining and processing. Location Listner also use for listen updated current location. requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener listener, Looper looper) method is reserved for location receiver. We need to just register location listener to this gps provider.LocationListner listens for location update, provider unable, disable.

C. Retrieving SystemUI.apk using ADB

Now begins fun part. SystemUI retains in /system/apps as discussed before. To retrieve SystemUI we will take help of ADB.ADB PULL command help to retrieve SystemUI from android.

```
C:\adb>adb pull /system/app/SystemUI.apk
7218 KB/s (2306210 bytes in 0.312s)
```

Figure 13: ***SystemUI pull from android to PC***

D. Decompilation of SystemUI.apk and SystemUI.odex

We have SystemUI in android Context now time to decompile it. As shown in figure Fig 14 apk file is package which contains resources such as anim ,layout, menu, drawable. However it doesn't contain any compiled classes like classes.dex.

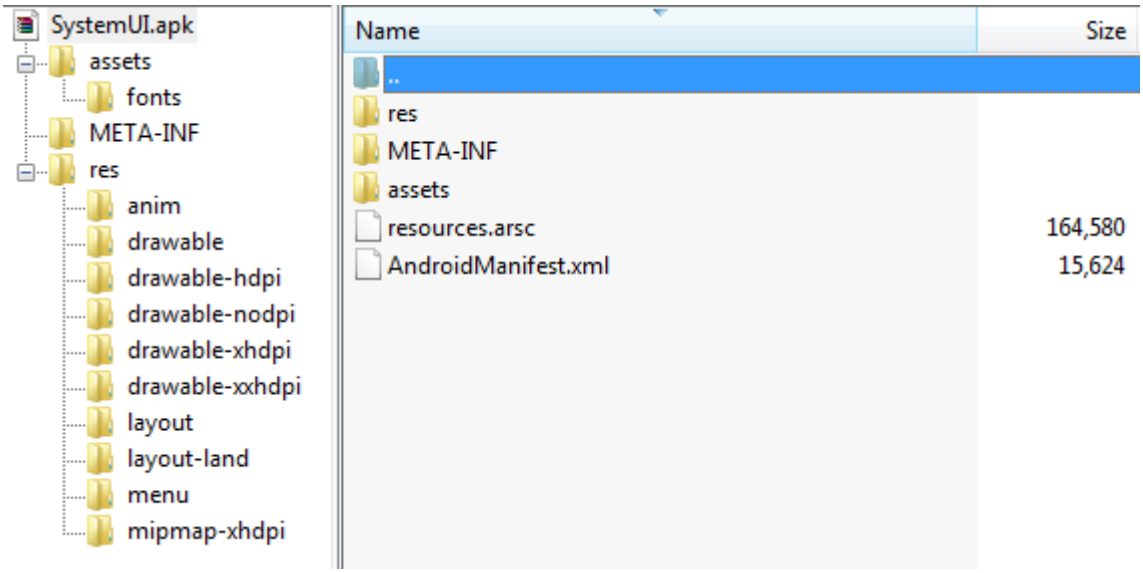


Figure 14.Internals of SystemUI.apk

Android developers have decided to create odex file from apk .odex is generally pre optimized version of apk. They moved classes.dex into odex file.thats why we need to decompile odex file for retrieving classes.dex. To retrieve classes.dex file from apk we need tool known as backsmail.jar and dex2jar which have framework to decompile systemUI.odex.as shown in fig, 14 we have view after decompiling system.odex.

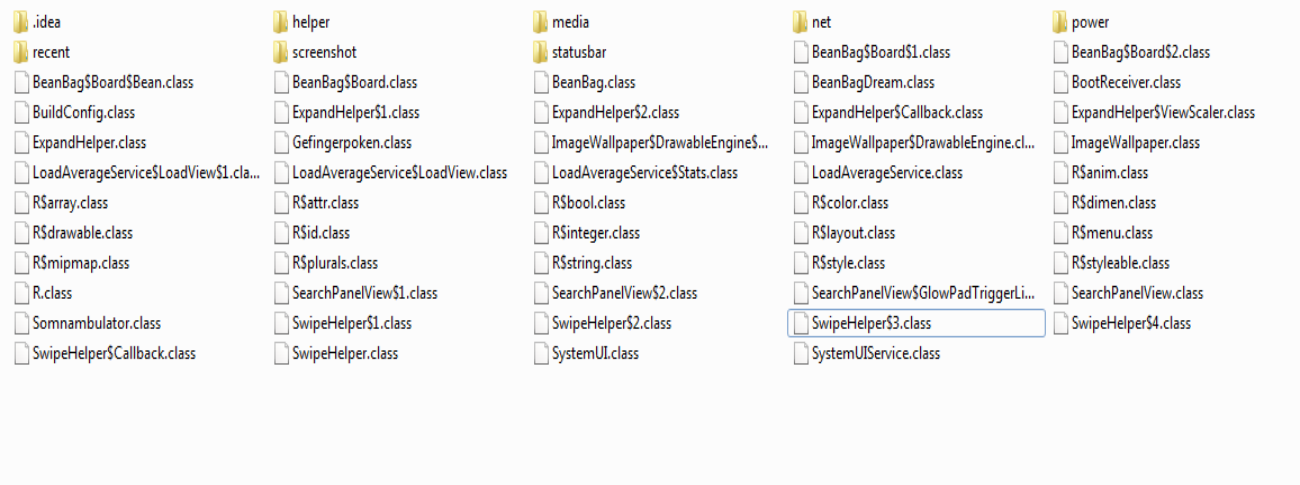


Figure 15: ***view after decompiling SystemUI.apk***

Now, we need another tool named Jdcompiler for decompiling .class file to. Java file. Jdcompiler provides read only class decompilation. if we want to change in this package we has to do changes we have to manually compile java file and add that .class to this package.

E. Exploit SystemUI.apk and Implementing Rootkit

Now our main goal is to add our service (c_t_s) which starts after boot completion within this package. For this, we have inject code into BootReceiver.class which can be seen in fig 15.BootReceiver is broadcast receiver which wait for boot complete intent. BootReciver accepts boot complete intents and starts c_t_s service which we have injected in SystemUI package. c_t_s uses telephony service and location service to trace gps coordinate and send it through SMS.

We have designed mechanism which listens for broadcast of SIM_STATE_CHANGE which signals our SMSManager to when to send GPS coordinates to predefined number.

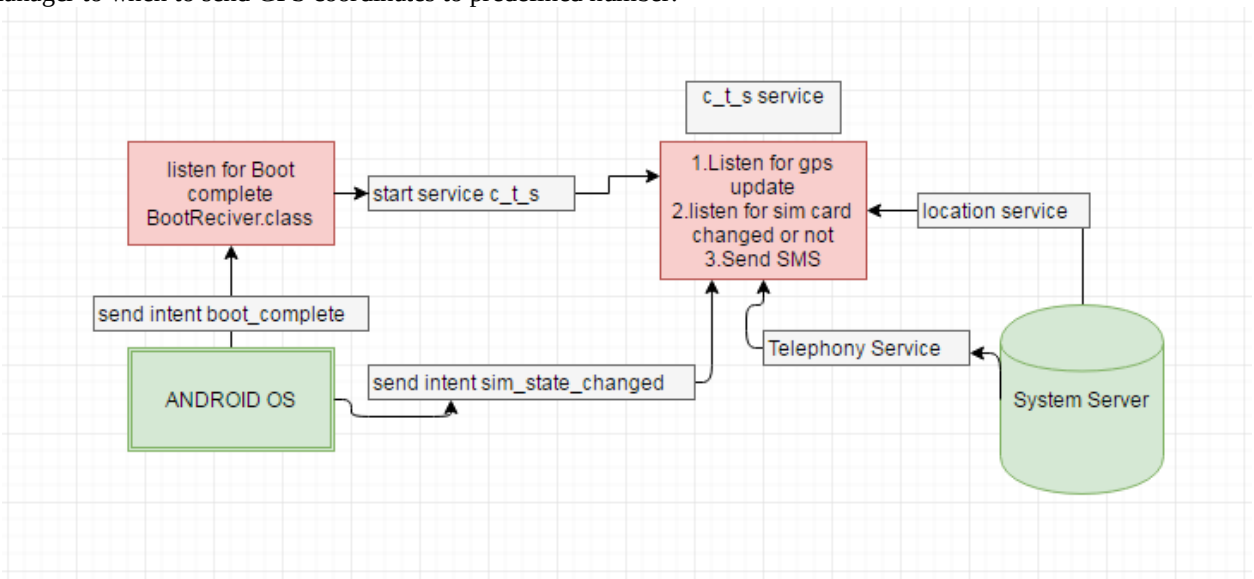


Figure 16.Our tracking rootkit flow

F. Repacking SystemUI.apk

After designing Rootkit we need to convert it into previous state there is two possibilities. First one is download AOSP SystemUI package manually change it according to our necessity, compile it and replace it exist SystemUI.apk. we can skip compilation part and retrieving part from our research paper if we want to follow that step. Second approach is to as we have done in previous steps. We need to put that c_t_sService class file into SystemUI ,put service, intent filter and permission entry in manifest and convert it back into odex file and put it with apk file in /system/app.

IV. RESULT

A. Feasibility of implemented Service

Implemented service cannot be detected by user because it runs within parent process. if we type command adb shell ps |grep SystemUI will show following result.

Figure 17.SystemUI runs under AID PID 941 and size is about 536928 bits

As we can see there is not child thread for service which start by braodcast reciever in SystemUI.Disadvantage of this service is it tooks little bit more computaion power and ram resources.

B. Final testing and exploit research

As rule of researcher, we are continuously trying to make it better. But at this time we didn't find any problem with this package. Figure shown below is screenshot of received location and unauthorized person's number from my rootkit. Here Long:73.22477 means longitude and lat:22.22945 for latitude coordinate and pn:8200382872 is phone number of thief .we can now check location by this coordinates via Google map.

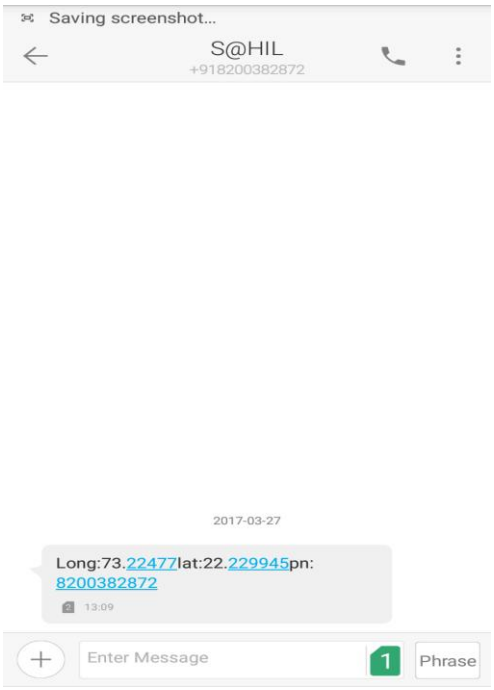


Figure 18: Auto SMS from rootkit with long, lat and phone number of subscriber.

I. FUTUER WORK

- Less use of resources and asynchronous call in replace of continuity.
- Protocol in replace of SMS.
- More Method for Covert Channel implementation.

II. CONCLUSION

Android is highly flexible and because of its open source availability every expert can find way to inject in it.By the we have used this rootkit for good purpose. It can be implemented in any android device and cannot detect by android user easily.

REFERENCES

[1] A New Covert Channel over Cellular Voice Channel in Smartphone,Author:- Publication & Year:-IEEE 2013
[2] Differential Power Analysis of MILENAGE Implementations in 3G/4G USIM Cards, Author:- Junrong Liu, Yu Yu, Fran¸cois-Xavier Standaert, Zheng Guo, Dawu Gu, Wei Sun, Yijie Ge, and Xinjun Xie, Publication & Year:- IEEE 2013
[3] Implementation of Location based Services in Android using GPS and Web Services, Author:- Manav Singhal, Anupam Shukla, Publication & Year:- IJCSI 2012
[4] Integrated Phone Locator (IPL): Lost Mobile phone tracking and recovery designs ,Author:- Newton Lwanga, Publication & Year:- IEEE 2014
[5] Location Based Services using Android Author:- Sandeep Kumar, Mohammed Abdul Qadeer, Archana Gupta, Publication & Year:- IEEE 2009
[6] Mobility Tracking using GPS,WI-FI and Cell ID Author:- Xiaoli Wang, Albert Kai-sun Wong, Publication & Year:- IEEE 2012
[7] mTracker: A Mobile Tracking Application for Pervasive Environment ,Author:- Luis Carlos Moreno Varandas , Bindol Vaidya ,joel jose puga Coelho rodrigues, Publication & Year:- IEEE 2010
[8] Real Time andOffline GPS Tracker Using Arduino, Author:- MangeshKolaskar, Aniket Chalke, MadhuraBorkar KedarNaik, Dr. B.K Lande, Prof VarshaSuralkar, Publication & Year:- IJIR 2016
[9] android architecture from android website: https://developer.android.com/guide/platform/index.html
[10]android analysis from virusbulletin website: https://www.virusbulletin.com/files/6014/5571/2236/SEAndroid-fig2.jpg
[11] Dissecting Android Malware: Characterization and Evolution,author: Yajin Zhou, Xuxian Jiang,*2010*
[12]Andorid security internals,Nikolay Elekov Publication & Year:- IJIR 2015
[13] Apk to ODEX convert website: https://forum.xda-developers.com/showthread.php?t=2092154
[14]Android Studio for development website: https://developer.android.com/studio/index.html

