ERNW
providing security.

ERNW Newsletter 38 / February 2012

Sell Your Own Device -
A Field Study on Decommissioning of Mobile Devices

Version: 1.0

Date: 24.02.2012

Author: Sergej Schmidt (sschmidt@ernw.de)

ERNW Enno Rey Netzwerke GmbH

Breslauer Str. 28

69124 Heidelberg

Tel. +49 6221 480390

Fax +49 6221 419008

www.ernw.de

ERNW
providing security.

Table of Contents

1 About the Project

The amount of mobile devices such as tablets and smartphones has risen rapidly in the past few years in the end-users' area. Consequently it was just a matter of time they would find their way into corporate environments. Many security managers and administrators have already been facing the problems of integration of such devices in their company; others will have to face it in near future.

Mobile devices share important properties. They are easy to steal, don't implement security mechanisms as known from PCs or notebooks and almost all include a flash storage device. For the focus of this project, the last property is most important.

Despite safe and secure integration, the decommissioning process of such devices turns out to be totally underestimated. Expensive and valuable smartphones won't simply be thrown away but resold or just given away; to the children or to friends. At this point control over the data on it is completely lost. Corporate environments often lack processes, which appropriately address the replacement of such devices.

The Bring Your Own Device (BYOD) policy allows companies to save money while it satisfies the employee allowing use of his desired device also at work. Considered the legit as well as the legal side; this is a situation where employers and administrators cannot control what will happen with these devices and in particular with potential business data stored on it, after the smartphone has been replaced by the owner.

This project shall demonstrate which data can potentially be extracted. We bought 19 cutting-edge mobile devices on eBay and tried to find undeleted or recoverable sensitive data. As sensitive data we understand personal media like photos, music and videos, credentials of access points, email accounts, VPN, personal messages and emails, especially with business context and miscellaneous information which should remain secret. We concentrated on mobile operation systems with the highest market share; iOS, Android and BlackBerry.

With this project we want to gain and raise insight on which data usually remain on mobile devices and how to deal with.

1.1 How to Read this Document

To be able to better understand the document structure it is to mention, that every mobile operation system stores data in a particular way. So different evaluation categories, depending on findings and device properties were defined.

The second purpose is to raise awareness for the process of obtaining data. Some sections give an overview on how data was made accessible and obtained. However processes will not necessarily be described in detail. E.g. this document does not include a jailbreaking step by step guide. However, technical details, as well as short code snippets will be provided, if they support the reader in rating the significance of presented results.

1.2 Data Privacy

It is to stress out that acquired data was handled with care. Accessed and evaluated by one person during the whole project, no third party was provided with personalized information revealed by examined devices. Data was stored on encrypted hard disk drive and securely deleted after the project was finished.

2 iOS

To obtain personal data from iOS it's necessary to bypass Apples security mechanisms. The general approach is to jailbreak the device and to install an SSH daemon. After connecting the device via USB cable to a notebook it is possible to log in with administrator privileges and pull the raw data from the partitions.

2.1 Overview - What we were looking for

iPhones and all other iOS devices have two partitions. */dev/disk0s1* is mounted as the root directory and */dev/disk0s2s1* (*disk0s2* on older devices) is mounted on */private/var*. The root partition is supposed to be readable only during the operation of the device or at least until the next firmware upgrade. As its mount point implies the second partition is much more interesting, since it stores all variable data including keychains, credentials, address book, text messages, photos, app data and much more stuff you could tag as sensitive or confidential.

2.2 Instruction - How we did it

As a first step it's necessary to jailbreak the iOS device. Therefore we use the "redsn0w"[1] tool which also allows to install an SSH bundle. After a successful jailbreak process SSH is listening on port 22, allowing access as user "root" with the default password "alpine". Now it's possible to access all data on the device which can be pulled via SSH/SFTP. But that's far not enough. Most people won't use common procedures known on hard disk drives to securely overwrite the storage on smartphones in order to prevent recovery, regardless of the fact such a feature is not always provided. The *dd*-command allows to retreive all data in raw format using

ssh root@ipaddress -2 -p 22 "dd if=/dev/rdisk0s2s1 bs=4096KB" | dd of=/path/private_var.img

Depending on the OS is used on a PC it's either possible to connect through a (wireless) network or through an USB cable connection. Regarding the fact the size of the flash memory varies between 8 and 64 GB the second variant appears as more suitable. In this case a tunneling tool is needed; "iphonetunnel-usbmuxconnectbyport"[2]. On Windows you additionally need to install "Cygwin"[3] for using the Unix *dd* command or just establish an SSH tunnel in order to work with a desired machine.

Usually some problems have to be faced due to the media partition being mounted, but using BSD-style /dev/rdiskX instead of /dev/diskX allows to access the raw data of the partition which can be obtained via *dd* even while the partition is mounted. After obtaining the image it was mounted on a usual GNU/Linux machine with the following command:

sudo mount -t hfsplus -o ro,loop /path/private_var.img /media/tmp

Now the mounted image can be processed by hand. The following folders, shown path relative to the mounted partition, are particularly interesting:

- *Keychain* OS X style Keychains
- *mobile/Applications* currently installed application, all their data, including data bases with all possible user data
- *mobile/Library* : databases for the SMS, email, calendar, notes, etc.
- *mobile/Media* containing all the multimedia files, including photos, videos and music
- *MobileDevice/ProvisioningProfiles* mostly found on corporate devices
- *root/Library/Caches/locationd/consolidated.db* since iOS 4, contains many GPS and Wifi hotspots location data. For further information see apple's FAQ

Besides the full overview of present data also already deleted files can be recover, since it's possible to pull the raw copy of the partition. *Photorec* of the "Testdisk"[4] suite is an easy to use recovery tool. The tool can recover files recognizing them by the file header. This is at the same time the reason why files

revealed with generated names. By explicitly advising *photorec* to search in the free space it recovers only the once deleted data. After the tool dealt with the image you get a bunch of unsorted files in many different folders given automatically generated names. Just file extensions are correct. The file types, searched for are png, jpg, mp3, ogg as well as XML plist (sometimes partly binary) and sqlite files, since most of user data is saved in SQLite databases.

In order to filter the resulting data by type and additionally copy into a single folder a small script attached in Appendix A of this document.

2.3 Evaluation – What we found

2.3.1 Tested Devices

- 2x iPhone 3G - iOS 4.1
- iPhone 4 - iOS 4.3.3
- iPhone 3G - iOS 3.1.3
- iPad – iOS 4.3.5
- iPhone 3GS – iOS 4.1

iPhone 3GS as well as the iPad will not be mentioned in the results. The 3GS couldn't be jailbroken due firmware problems and the iPad was either properly wiped or hasn't been used at all, so data could not be found or restored. On other devices we found noticeable clean- up attempts, which were either just partly successful or didn't work out at all.

2.3.2 Apps

Apps were found on three of four devices. On one device the owner uninstalled some apps but left behind his games and instant messenger app. The two other devices were cleaned, however again the instant messenger app was left untouched. The database files of these apps contained the whole contact list as well as all messages. One of the apps even revealed the phone numbers of persons in the contact list.

2.3.3 Mails / SMS

One device came with a functionally configured email account. Revealed emails contained personal conversations, information about attending a new work place and notifications about money transactions. Another person deleted only the inbox and trash folders of the email account but didn't delete credentials and saved outgoing emails, including the attachments.

In the remaining two cases accounts were properly cleaned. No SMS were found on all iOS devices.

2.3.4 Contacts

One person didn't clean up his/her contacts list. All contacts were hold with corresponding phone number and email address.

2.3.5 Keychain

Two devices contained email credentials and wireless LAN APs in their Keychains, one of them credentials of online gaming accounts.

Although the passwords are stored encrypted, it is possible to decrypt them with the appropriate effort. Jens Heider and Matthias Boll from Fraunhofer SIT made research in this area and described it in their paper "Lost iPhone? Lost Passwords!".

2.3.6 Consolidated.db

On the iPhone 4 and one of the 3G with iOS 4 the consolidated.db contained a significant amount of data of connected Wi-Fi hotspots and GPS transmitters with coordinates and a timestamp. On iOS 3 devices the file is not existent by default and on the wiped iPad the file also wasn't present.

2.3.7 Not removed media

All users deleted the multimedia files, especially the personal ones. The findings here were one mp3 file on a smartphone. The interesting stuff was found by automated examination of the files. One SQLite database file with all ever bought apps and cookies with a bunch of visited websites was found.

2.3.8 Recovered Media

The other devices provided quite disillusioning results. *photorec* provided thousands of recovered files. Per phone it was possible to recover/obtain up to 60.000 files. *Plist* as well as *sqlite* revealed a lot of information, even though it is necessary to deal with a large amount of unsorted files. Consequently sample checks were made when analyzing the obtained data.

The result were visited websites, WLAN-APs (with IP, SSID and MAC), a database file from latitude app containing hundreds of visited towns with the corresponding streets, and a lot of data of alarm sets. The founded music may not be sensitive, but is considered personal property. Some devices revealed only a couple of music files, while others contained several gigabytes of music.

The found pictures can be sorted into 3 categories. The first one is private photos. Only a couple of them were found on each device. Besides a lot of app noise, some instant messengers revealed avatars or sent photos. Even though those can be marked highly private, the found screenshots, showed even more significance.

2.3.9 Recovered Screenshots

When pushing the home button the currently open app is minimized with refreshing cool effect. However this is implemented through a scaled down screenshot. This feature is well-known for the arising security implications. In particular the resulting frequency in which sensitive data is stored through this mechanism is highly problematic. Taking into account data cannot reliable being deleted from a flash memory device the amount of data being stored over months or years is significant.

On each device it was possible to recover hundreds of screenshots, showing login screens of social networks, social network conversations, corresponding contacts as well as avatars and calls, partly combined with phone numbers.

2.3.10 Profiling

Analyzing emails, pictures, music, apps, combining this knowledge with information gathered through revealed credentials and login names, it is rather easy to create a personal profile including cultural background, friends and partner(s), determine gender, interests and appearance of the smartphone owner. Profiling the owners of the remaining devices yielded same result.

3 ANDROID

3.1 Overview - What we were looking for

Android Apps store their data in */data/data*, following naming conventions as known in java packages eg. com.google.mail. The app folders typically contain directories such as *lib (…* if required for corresponding app*)*, *files* (…saved on internal storage), *cache* (…from the app, often browser) and *databases* (SQLite files). Sometimes *shared_prefs* folder can be found containing key value pairs in a lightweight XML format. Depending on the app the data can be also stored on external storage in the folder */sdcard/data/*.

The media and email attachments are usually found on the external storage, in most cases the SD card, mounted on */sdcard*. On some devices, there was an emulated SD card mounted on the usual mount point. The external SD card on Samsung devices was mounted at */sdcard/external_sd/*.

Further interesting partitions for acquisition are */data* and */cache*. Additionally user data (in particular sqlite files) can sometimes be found at */dbdata*. */system* wasn't interesting due to the fact of being mounted read only by default. Consequently no interesting data is to expect at this point.

3.2 Instruction - How we did it

In order to pull the devices' non-volatile memory high privilege permissions were needed. Due to security reasons these are restricted on most mobile platforms including Android. So the acquiring of root permissions can only be performed through exploiting security flaws.

To fulfill this was time consuming, because the procedures differ for each device and often vary depending on the firmware version. *Z4Root*[6] is an easy to use app, which unfortunately only worked older versions of Android. *SoftRoot*[7] helped for the Wildfire, while remaining HTC devices were unlocked using *Revolutionary*[8] and booting into recovery mode in order to be able to pull the data as root. In rare cases *VISIONary*[9] helped out when other solutions didn't work.

The *Android SDK*[10] is an essential tool. It offers the opportunity to use the *adb shell* command which is used pretty similar to *SSH*. Before *adb* can be used the debugging mode on the Android device has to be activated. With the *adb shell* command a connection (as root, when Android is rooted) is established to the device. Afterwards it's possible to connect to the devices via USB cable and retrieve the raw images with

adb shell "dd if=/dev/block/devName" | dd of=/path/name.img

Or for the devices where *su* has to be entered before being able to login as root:

echo "echo 'dd if=/dev/block/devName && exit' | su" | ./adb shell | dd of=/path/name.img

As there is no generic naming convention across manufacturers, it is necessary to figure out about partition and mountpoint names using the *mount* command.

3.3 Evaluation – What we found

3.3.1 Tested Devices

- Samsung Galaxy S2 - Android 2.3.3
- Samsung Galaxy S - Android 2.3.3
- Samsung Galaxy S - Android 2.2
- HTC Desire HD - Android 2.3.3
- HTC Wildfire - Android 2.2
- 2x HTC Desire - Android 2.2
- LG Optimus GT 540 - Android 2.1

3.3.2 Mails / SMS

The result out of all Android phones were 2 undeleted SMS on one device. All remaining phones had been properly cleaned.

3.3.3 Contacts

The result here was also moderate. A couple of email contacts in a recovered xml file were found.

3.3.4 Not removed media on SD card

Only one owner didn't delete his media on the SD card. Even though he deleted his photos 17 thumbnails were left on the drive. Also pictures of a game app were found on the SD card.

3.3.5 Recovered Media from SD card

On 3 of 7 devices private and partly sensitive data was recovered. About 200 pictures containing private photos, graphics from apps and covers of music albums. On one device many screenshots of surfed sites, including home banking sites (captured in and out coming payments along with bank account numbers) as well as an online dating service along with a captured login name. 84 audio files were recovered, which consisted of mostly music files, two voice recordings and a voice message of navigation software. In addition we found 28 mp4 videos, including a personal video, music clips and pornographic content. Two readable pdf documents, a flight ticket, a restaurant menu and a lot of txt files were recovered as well. Most of the text files were cleartext snippets, recognized by the recovery tool. These snippets turned out to contain all sorts of stuff. On one device two links including a user ID for an online music store were found. The remaining files contained insensitive information.

3.3.6 Recovered Media

Android uses the partition mounted as */data* (depending on the device additionally */dbdata*) to store application data, configs and logs. In an exemplary matter all users of the phones reset their device to factory state so the data on the mentioned partition was deleted properly. Since Android 2.x version does not offer storage encryption a lot of data could be recovered. It was possible to recover numerous sqlite database files as well as thousands configuration and pictures files. These were analyzed by hand, so it was not possible to examine all of them. Therefore, again we relied on spot checks.

With more than three thousand picture files, most of them jpg, it was possible to determine once installed apps. Among the recovered pictures were a lot of thumbnails of private photos and cached pictures of a social networking app.

Tons of xml and sqlite files revealed visited websites and services (Google maps searches, internet radio, song names of purchased music), saved destinations of navigation software, names of once installed apps, two email accounts, one with a hashed password, ringtones and two Google accounts the particular devices had been linked to. Most interesting findings were two xml files with credentials to syncing, backend and remote wipe services of a particular device provider. In both files the email address was stored as ID as along as the IMEI, phone number and the password, which was hashed in one of the files but, saved in cleartext within the second one. Furthermore credentials for a Googlemail and Skype account were found, again including hashed passwords.

The recovered txt files revealed a lot of visited URLs including warez sites, Facebook with IDs, online car sellers, tourist agencies, eBay and other shopping sites, a URL of Google calendar, logs of messenger apps with unique contact IDs (used by the app) and corresponding phone numbers.

3.3.7 Profiling

Besides determining surf behavior, hobbies and age of owner in one case, it was possible to figure out the owner's appearance and age through personal photos as well as the music taste and where he bought his music. On another device we found pictures of friends, apartments and an office space. For the remaining device owners it was possible to figure out interests and place of residence. However the most interesting device disclosed email address, the name as well as two different social networks IDs and interests in particular products.

On two devices the data didn't allow an educated statement.

4 BLACKBERRY

4.1 Overview - What we looking for

The desire here was retrieve the typical data processed on BlackBerrys; contacts, email, SMS, calendar, etc.

Also the personal data and media stored on SD cards were of particular interest.

4.2 Instruction - How we did it

To retrieve the personal data *BlackBerry Desktop Manager*[11] was used, which creates a backup of all personal data stored on a BlackBerry device. Afterwards we examined the backup files with the *Oxygen Blackberry IPD backup viewer and reader*[12]. Direct access to the file system was not possible whereas no jailbreak for the BlackBerry phones (tablets excluded) currently exists.

The SD card delivered with the devices was also analyzed with the ulterior motive to recover corresponding files.

4.3 Evaluation – What we found

4.3.1 Tested Devices

- BB 9300 Curve OS v5.0.0.845
- BB 9300 Curve OS v5.0.0.832
- BB 9700 Bold OS v6.0.0.546
- BB 9800 Torch OS v6.0.0.600

4.3.2 Examination of Backup files

All examined devices were restored to factory state properly. No PIM (Personal Information Manager) or other personal information could be found.

4.3.3 Recovered Media from SD card

On the first examined SD card 34 personal photos and two binary files were found.

On the other SD card there were six personal photos, 65 RIM license agreements as pdf files. Furthermore thousands of txt files were recovered, whereas most of them were useless because they didn't contain any sensitive or personal data. However two xml files containing configuration data were found.

4.3.4 Profiling

It was possible to find owners family pictures. Furthermore the binary files could be identified as Mac OSX files, indicating that the owner used a Mac for synchronizing with his device.

5 CONCLUSION

Contacts, sensible photos, credentials and sometimes enough data to determine a user's virtual and real identity were found. That showed that the security mechanisms used by some mobile platforms won't let admins have peace of mind. Without even looking at the device platform itself it is disturbing to know that some app providers store the passwords in cleartext on your device. It's also disturbing to see how some people handle their data. It was noticeable that all used devices bought from commercial resellers were reset but the most from private sellers not. It was distinct the most iPhone owners didn't even delete their apps; one even told us not to upgrade the OS because the jailbreak and the SIM unlock will be disabled. I assume that he, as well as many other users, seen an added value not deleting the apps without a security look at having a 2 years old OS on his phone. Still, I think the most users just weren't aware of the "set to factory state" feature. This is critical especially if you think about BYOD in companies, mentioned in the beginning of this document. Even if users pay attention, it's a fact that we can restore most of once stored data if the storage is not encrypted.

In the end we showed that the users don't pay attention and don't have a sense for security or protection of private data. So you as a company's responsible person can't rely on users to delete their data properly (even if they are able to). You should be worried about the data stored on mobile devices in your company and implement controls for proper decommissioning. Allowing use of private devices in your corporate environment you won't be able to get your hands on the devices.

We hope you remember this project when your company is making the step to allow BYOD.

6 MITIGATION

6.1 Decommissioning Processes

In general case there should be an implemented process to reset the devices to factory state after a mobile device was replaced or given back. Also the given back devices should be reset by an administrative department (not necessary exclusively) responsible for decommissioning.

Remote wipe functionality is useful not only for lost devices. After an employee leaves the company and doesn't return the device it can be used to reset the device or at least to delete the corporate data on it when using a container solution (mentioned later).

6.2 Storage Encryption

To make decommissioning process effective only mobile devices with storage encryption should be used.

Many articles and security researchers abuse Apple's storage encryption feature being useless for protecting user data in case of theft. But it was primary introduced to allow a fast remote wipe. Deleting the encryption key (or at least a part of it) makes the data once stored also unrecoverable. It also makes a factory reset effective and makes a recovery impossible.

While BlackBerry already has features to encrypt internal and external storage, Android doesn't. This made the result described above possible. Especially the */data* partition is problematic. Using Android version >=3.0 mitigates this issue since reliable storage encryption is used. Android >=3.0 only the */data* partition is encrypted and is only decrypted during boot process after the password is typed in. If you have stored unencrypted data on a (micro) SD card, which is recoverable as described above. The methods of secure erasing used on regular hard disk drives are not efficient on flash drives. So there is no mitigation possible for external media as on older smartphones and tablets. The storage cards and the devices without encryption, which ever carried sensitive information, shall not leave the company and/or, depending on how sensitive the processed data is, may be destroyed.

6.3 Container Encryption

In BYOD scenarios it's not possible to rely on users to delete their private data. More important, there is no influence on which devices are used. Having a device with no storage encryption a company is compelled to use a container solution. A container offers dedicated apps e.g. for email, calendar, contacts and stores the corporate data separated from the private data on the mobile device. These solutions have three advantages; they effectively separate private and corporate data, don't rely on the security features of the mobile platform itself and they are encrypted. So it's possible to delete the corporate data without touching the user's data.

6.4 Backend Storage

An advice is not to store sensitive or confidential data on the device at all. This can be done by using web application with secure authentication. It's possible to create own application which are comfortable for use with mobile devices. There are already web applications which use look and feel suitable for smartphones and tablets e.g. the Google mail web app. There is a problem using the native browser of mobile devices: you have no influence which data being cached. To address this problem it's possible to develop and/or use apps which operate as frontend-ends. After authentication backend services (e.g. corporate network or third party) are accessed without storing confidential data on the device itself. The big disadvantage of this scenario you need to have an internet connection to access the data.

7 APPENDIX

7.1 A - Tools

1 - redsn0w jailbreak for OS X / Windows

http://blog.iphone-dev.org/ official developers' Homepage

https://sites.google.com/a/iphone-dev.com/files/home/ here you'll find the newest version of the tool

2 - iphonetunnel-usbmuxconnectbyport_for OS X / Windows

http://code.google.com/p/iphonetunnel-usbmuxconnectbyport/

3 - Cygwin

http://www.cygwin.com/

4 - Testdisk (includes photorec)

http://www.cgsecurity.org/

5 - SQLite Browser

http://sqlitebrowser.sourceforge.net/

6 - z4root App is a one click tool for rooting Android.

No official homepage, only links in forum posts

http://forum.xda-developers.com/showthread.php?t=833953

7 - SoftRoot App is an easy to use rooting tool.

http://forum.xda-developers.com/showthread.php?t=757191

8 - Revolutionary is S-OFF unlocker for many HTC mobile devices. Also tools to flash the recovery mode are provided on this site. This custom recovery ROM offers an activated android debugging bridge and root access.

http://revolutionary.io/

9 - VISIONary is an easy to use automated tool for rooting android devices

http://forum.xda-developers.com/showthread.php?t=808514

10 - Android SDK

http://developer.android.com/sdk/index.html

11 - BlackBerry Desktop Manager

http://us.blackberry.com/apps-software/desktop/

12 – Oxygen Blackberry IPD backup viewer and reader

http://www.oxygen-forensic.com/en/features/ipdbackup/

7.2 B, useful scripts

Scripts used to extract/isolate files of the recovered files to designated folders

```
#!/bin/bash
# findscript.sh
find $1 -name *.mp3 > mp3s
find $1 -name *.ogg > oggs
find $1 -name *.plist > plists
find $1 -name *.sqlite > sqlites
find $1 -name *.jpg > jpgs
find $1 -name *.png > pngs
ruby masscopy.rb mp3s mp3
ruby masscopy.rb oggs ogg
ruby masscopy.rb plists plist
ruby masscopy.rb sqlites sqlite
ruby masscopy.rb jpgs jpg
ruby masscopy.rb pngs png
```

```
#!/bin/ruby
#masscopy.rb
input = File.open ARGV[0], "r"     #specify file where the files are listened
out_folder = ARGV[1].nil? ? "recovery_filter" : ARGV[1] #name of the folder to copy files
begin
        Dir.mkdir(out_folder, 0700)
rescue
        nil
end
input.lines.each do |x|
```

```
        system("cp " + x.chomp + " " + out_folder) # copy to folder
end

input.close
```