The Quarterly Magazine for Digital Forensics Practitioners

Issue 27 • May 2016

DIGITAL FORENSICS / MAGAZINE

WIN! an iPod Nano

Digital Forensic Data Analysis *USING MATHEMATICA*

Latest News, 360
Book Reviews, IRQ
& much more inside!

PLUS!

Metadata Analysis in Email
DDoS Attacks & Mobile Devices
Beyond Keywords
JTag & Chip Off

9 772042 061004 26
Issue 27 / £14.99 TR Media

WINDOWS PHONE 8 FORENSICS

Mattia Epifani and *Francesco Picasso* *provide an overview of the current state of the art in Windows Phone 8 acquisition and analysis techniques.*

/ INTERMEDIATE

According to the latest statistics, Windows Phone, the operating system developed by Microsoft for mobile devices, is the third after Android and iOS for worldwide distribution at a rate of around 4%. For this reason, a lot of studies and research in the forensic acquisition and analysis of this kind of device have been developed in the last couple of years.

/ DEVICE ACQUISITION

At least three factors influence the possibility of the acquisition of devices with Windows Phone 8:

- The presence of different manufacturers
- The presence of a PIN code
- The native inability to make a local backup of the contents of the device, which only uses Microsoft cloud services

In general, therefore, it is possible to distinguish five types of acquisition:

1. Physical
2. Logical
3. Installing specific Apps
4. Manual
5. Cloud

Physical acquisition is not possible for all devices, but is of course the most preferable one from a forensic point of view because it permits the obtaining of a full dump of the internal memory. Moreover, this type of approach allows access to the memory of the device, regardless of the presence of a PIN code. At the time of writing, there are mainly three techniques to obtain a physical acquisition:

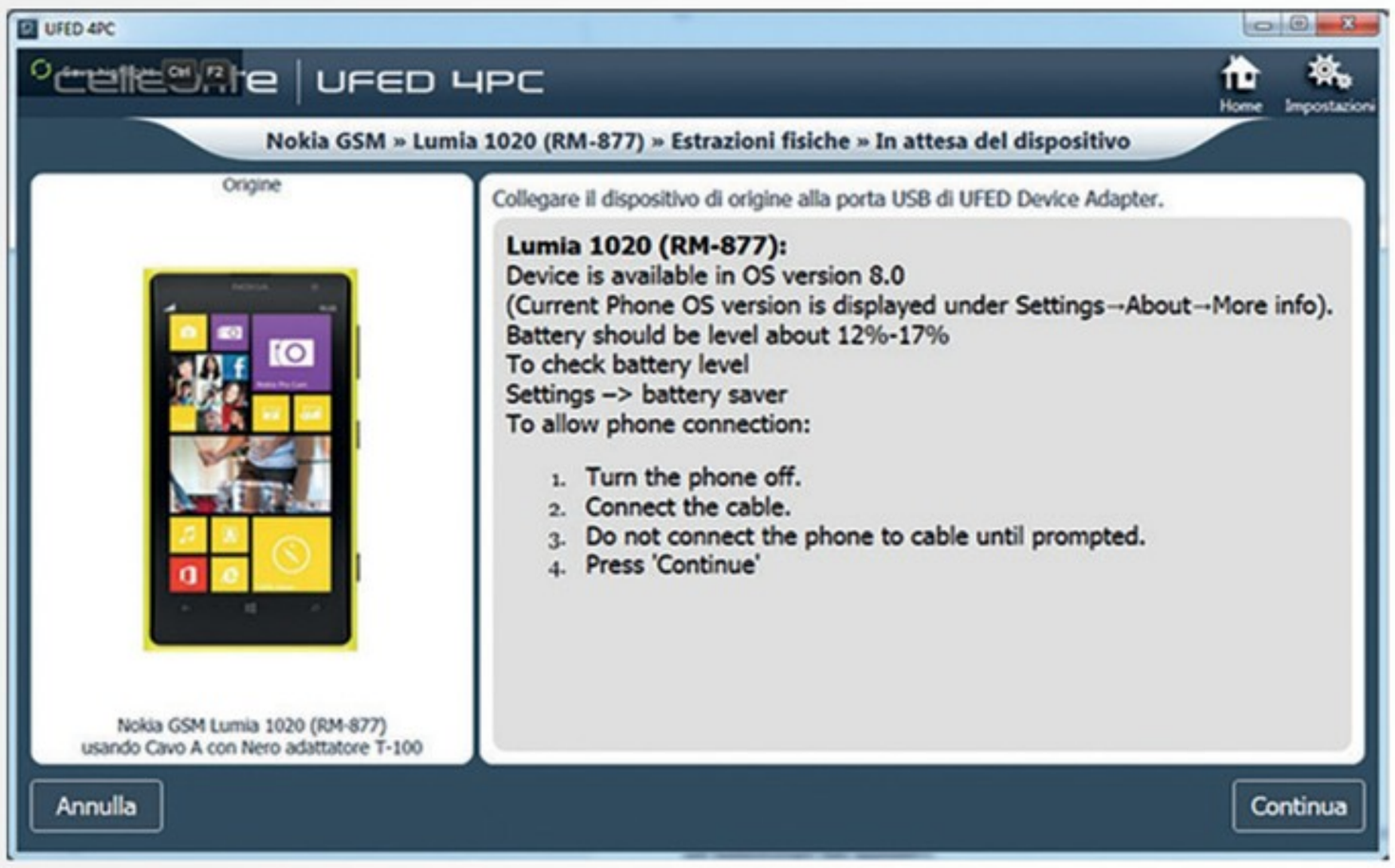


Figure 1. Acquisition of a Windows Phone 1020 Device

1. Extraction by impairment of the security measures of the boot process
2. Extraction through JTAG (Joint Test Action Group)
3. Extraction through Chip Off

Figure 2. Contacts + Message Backup

The first technique is only available for certain devices, which have vulnerabilities in hardware/software that allow booting the phone with an alternative operating system different from the one installed in the internal memory. At present, this technology is only supported by the hardware/software solutions implemented by the company Cellebrite.

This type of acquisition is done by connecting the device directly to the acquisition tool (e.g. Personal computer with Windows operating system and UFED4PC software). As of the writing of this article, Cellebrite supports the following 21 models: Lumia 1020 (RM-877), Lumia 1320 (RM-994), Lumia 1320.1 (RM-995), Lumia 520 (RM-914), Lumia 520.2 (RM – 915), Lumia 521 (RM-917), Lumia 525 (RM-998), Lumia 620 (RM-846), Lumia 625 (RM-941), Lumia 720 (RM-855) , Lumia 810 (RM-878), Lumia 820 (RM-825), Lumia 820.1 (RM-825), Lumia 820.2 (RM-

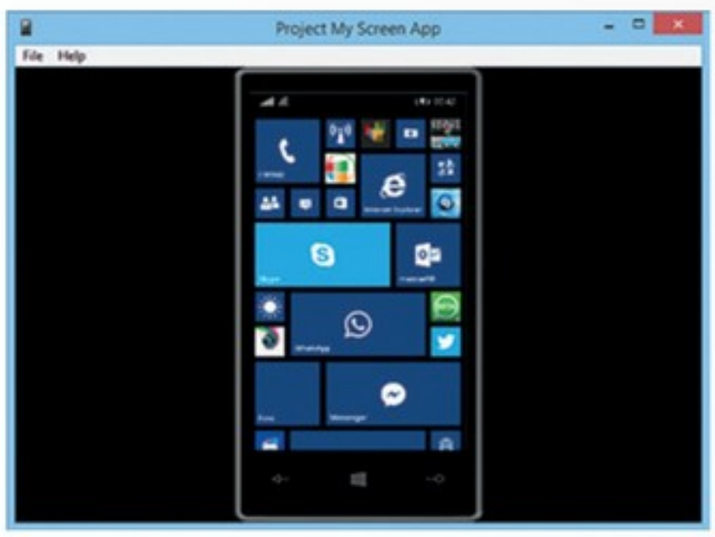


Figure 3. Project My Screen Appre

824), Lumia 822 (RM-845), Lumia 909.1 (RM-875), Lumia 920 (RM-820), Lumia 920.2 (RM-821), Lumia 925 (RM-892), Lumia 925 (RM-893), Lumia 928 (RM-860).

To perform this type of acquisition it is necessary that the phone satisfies a specific condition; the battery level should be between 12% and 17%. In Figure 1, by way of example, you can see the acquisition of a physical Windows Phone 1020 device.

The second technique is typically achieved by using hardware devices specially designed and developed, which allows the user to interface with the specific device to acquire. The market offers several boxes and adapters for the main models for which this solution is viable. Among those available, it is useful to highlight RIFF Box and Box ATF, for which there are many demonstration videos on different models of phones. The third technique involves the physical removal of mobile device internal memory and requires specialized training and tools. This technique is also invasive on the device and in case of failure can definitely destroy it and the stored data. The Canadian company Teel Technologies, among others, offers equipment and training for the use of the last two techniques

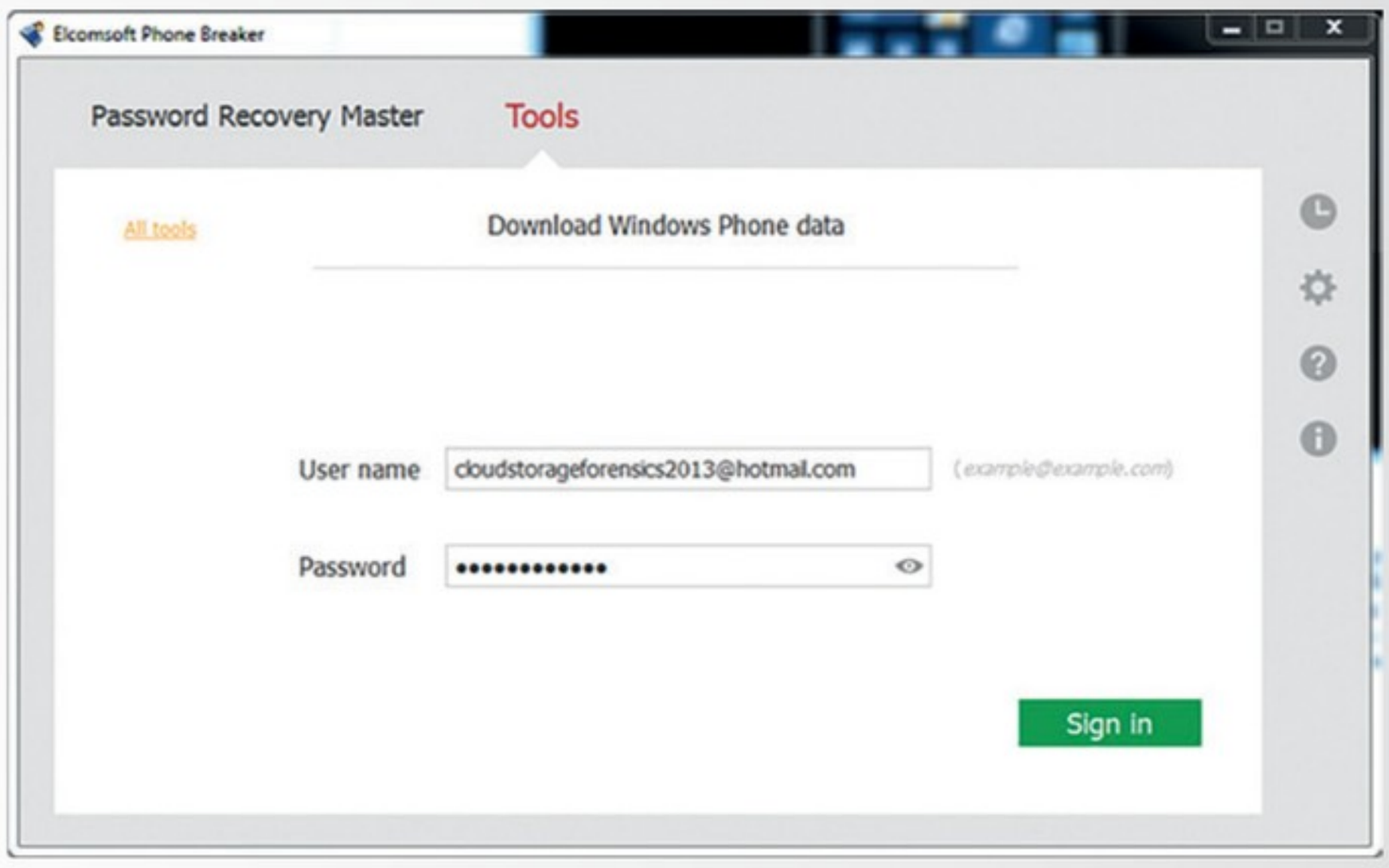


Figure 4. Windows Cloud Acquisition

Logical acquisition can be performed only if the PIN code is known. It is realized by connecting the device to a personal computer through USB cable or Bluetooth connection. The connection through USB cable is based on MTP (Media Transfer Protocol) and permits the extraction of Media Files (e.g. pictures, videos, audios) while connection through Bluetooth permits the extraction of the Address Book and Call History. Other information from native and third party applications is not available with this method. Various forensic tools, like UFED Cellebrite and Oxygen Forensics, support this kind of acquisition.

Acquisition through App is based on the installation of specific mobile Apps by which it is possible to save data on an external SD card. An example of this kind of applications is “contacts+message backup” from Microsoft (https://www.microsoft.com/it-it/store/apps/contacts-message-backup/9nblgggz57gm) that permits the saving of Contacts (in VCF format) and SMS data (in XML format). Also in this case it is necessary to know the PIN code.

Manual acquisition is made by browsing through the device menu and installed apps. A good way to capture the information is to connect the device to a PC with Windows operating system and “Project My Screen” application (https://www.microsoft.com/en-us/download/details.aspx?id=42536). Again it is necessary to know the device PIN code.

Cloud acquisition consists of extracting backup data stored on Microsoft cloud servers. In this case, user credentials must be known (username and passwords) and various solutions are available: restoring the backup to another device or downloading the information with specific forensic tools (e.g. Elcomsoft Phone Breaker, Oxygen Forensic Analyst).

/ PARTITIONS AND FILE SYSTEM STRUCTURE

In the case of physical acquisition, it is important to understand where the most valuable information is stored inside the device. The first aspect to consider is how the internal memory is organized. Memory is typically divided into 28 partitions. From an analysis carried out also by other researchers ([6] and [7]) the most interesting partitions are number 27 ("Main OS") and number 28 ("Data"). The remaining 26 partitions are related to the specific mobile SoC (bootloader, UEFI, backup configurations, etc.). Information related to individual partitions is stored in the Main OS partition and specifically within the file \Windows\ImageUpdate\DeviceLayout.xml.

The partition structure is described in a Microsoft document [2], which shows the three most interesting partitions, and specifically:

- Main OS (mapped on C:\)
- User Data (mapped on C:\Data)
- Removable User Data (mapped on D:\)

The Main OS partition is conceptually equivalent to the Windows operating system partition in a personal computer. A document published by Microsoft [3] describes the partition structure, pointing out that the most important folders are Programs and Windows.

The Programs folder contains binaries of built-in applications (e.g. Office and Outlook), where each app is contained within a sub-folder. The Windows folder contains registry files, executables, services and system drivers; no files can be modified by the user activities except the registry files.

From a forensic point of view, it is useful to highlight that the partition uses NTFS as the file system, with the relative presence of the metadata files ($MFT, $LogFile, $UsnJrnl). There are also the traditional Windows Registry file (Software, System, Security, SAM) and the paging file (pagefile.sys). Event log files, the hibernation file, the trash folder, the Prefetch folder and the shellbags are not present.

The Data partition contains information related to user activities with the phone, both with native and third party applications and uses NTFS as the file system. A document published by Microsoft [4] describes the partition structure. Upon first boot the subsystem that manages storage creates all top-level folders. The content of these folders is the following:

- Programs: code and executables of the installed applications from the Windows Phone App Store
- Users: default account for applications, built-in services (typically up to 23 users) and the Public User
- SystemData: various system files such as log and device updates
- SharedData: data shared between different applications

The Users folder contains information related to user profiles. There are three types of accounts:

- Service Accounts: system users who deal with managing services and system applications
- DefApps: user who handles the data of the applications installed in the system
- Public: user who retains media files (audio, video , images)

Among the service accounts, the most interesting is WPCOMMSSERVICES which contains various databases about native applications.

The Public account contains files that do not belong exclusively to an application and is used to store media files. By default, it contains the following sub-folders:

- Documents
- Downloads
- Music
- Pictures
- Ringtones
- Videos

Each of these folders may contain sub-folders managed directly by a specific application. For example, the Pictures folder may contain the following sub-folders:

- Camera Rolls: photos and videos taken with the device internal camera
- Saved Pictures: User-saved images from other applications (e.g. Facebook)
- Screenshots: user-generated screenshot of the phone screen
- WhatsApp: images and videos sent and received through the WhatsApp application

The DefApps account has the role of managing the component of application data installed in the device. This user folder contains only one sub folder called AppData which contains a sub-folder for each application. The applications are identified by a name or by an ID that matches the Windows Phone Store application ID (unique value assigned when the developer upload the app on the App Store).

/ ESE DATABASES

As reported on Microsoft website "The Extensible Storage Engine (ESE) is an advanced indexed and sequential access method (ISAM) storage technology. ESE enables applications to store and retrieve data from tables using indexed or sequential cursor navigation. It supports denormalized schemas including wide tables with numerous sparse columns, multi-valued columns, and sparse and rich indexes."

Various tools are available to parse and analyze an ESE database:

- libesedb packages by Joachim Metz (https://github.com/libyal/libesedb)
- ESEDatabaseView software by NirSoft
- (http://www.nirsoft.net/utils/ese_database_view.html)
- ESDbViewer software by Mark Woan
- (http://www.woanware.co.uk/forensics/esedbviewer.html)
- OS Forensics integrated ESEDbViewer (http://www.osforensics.com/esedb-viewer.html)

/ CLOUD ACQUISITION

When no other solution is available to extract physical or logical dump from the phone, it is useful to consider Cloud Acquisition as a possible way, but in this case user credentials are needed. One option is to download the backup on a test phone and then extract data from this phone, the other one is to use commercial solutions that permit the download of data directly from the Cloud (e.g. Elcomsoft Phone Breaker, Oxygen Forensics).

The structure of each application folder is uniform and specifically:

- INetCache: contains the application cache files, such as images and videos downloaded by the App during its usage
- INetCookies: contains the cookies necessary for the functioning of the App
- INetHistory: contains the history of Internet content accessed by the application
- Local: contains the data that is useful only within the specific device and must not be shared with other platforms. This is the folder that typically contains the App-specific data (e.g. WhatsApp and Skype chat, contacts and call databases)
- Local\Shared: contains Shared content from the application in the system, such as images for Tile application on Desktop
- Local\Shared\Transfer: used to upload/download files to the application
- Roaming: contains the data that, while being stored locally, is replicated on Cloud or shared among different installations of the same application in devices connected to the same user

The Removable User partition corresponds with the SD card that may be present inside the phone. On Windows Phone 8.0 the SD card can be used to store media files or to upload a file from outside (e.g. XAP App installer file downloaded from the Microsoft Store). Starting from Windows Phone 8.1 the user can also use the memory card to install applications and related data. The folder structure is similar to the user's Public (Partition Data) with the addition of WPSystem folder. Specifically, the WpSystem folder is used by the operating system to store two types of data: those relating to the applications installed on the SD card and those relating to files that can be shared among multiple applications. It contains three sub-folders:

Figure 5. WPSystem Folder for Viber

- Apps: contains binaries and libraries for applications installed on the SD card. For each application there is a sub-folder whose name corresponds with the unique identifier of the application within the Microsoft Store
- AppData: contains data for applications installed on the SD card. For each application there is a sub-folder whose name corresponds with the unique identifier of the application within the Microsoft Store
- SharedData: contains data that is shared among multiple applications (e.g. MediaMetaFile folder preserves previews (thumbnails) of the images and videos stored in the card)

All the files in Apps and AppData folders are encrypted and their contents are not readable by removing the memory card of the phone and analyzing it separately. On the contrary the contents of the SharedData folder are in clear text and interpretable therefore also outside the phone. This behavior leads to an obvious conclusion: it is not possible to carry out the analysis of an application that is installed on the SD card unless you go directly to the encryption/decryption of the phone, which depends on the specific SoC and the PIN code. Figure 5 shows the structure of the WPSystem folder for a specific installed application (Viber).

Tests have shown a further aspect of interest; a file received by an application (e.g. an image received through an instant messaging application) is saved in the clear in the memory card and then encrypted. This behavior makes it technically possible to carve the unencrypted version of the file from the unallocated space.

/ WINDOWS REGISTRY

A Windows Phone registry file is similar to the one that you can find on a Windows desktop installation. Some of the most interesting registry keys are:

Name	Type	Value
(Default)	REG_SZ	(value not set)
PhoneFriendlyName	REG_SZ	Windows Phone
PhoneFirmwareRevision	REG_SZ	3058.50000.1430
PhoneRadioSoftwareRevision	REG_SZ	2.0.242037.8
PhoneSupportLink	REG_SZ	http://link.nokia
PhoneROMLanguage	REG_SZ	0410
PhoneHardwareRevision	REG_EXPAN...	1.5.0.1
PhoneSOCVersion	REG_SZ	8227
PhoneMobileOperatorName	REG_SZ	000-IT
PhoneMobileOperatorDisplayName	REG_SZ	Italy - Country V
PhoneManufacturerModelName	REG_SZ	RM-914_eu_italy
PhoneModelName	REG_SZ	Lumia 520
PhoneOEMSupportLink	REG_SZ	http://link.nokia
PhoneManufacturer	REG_SZ	NOKIA

- **SYSTEM\Platform\DeviceTagetingInfo:** information about the firmware version, the phone model, the operator etc.

Name	Type	Value
(Default)	REG_SZ	(value not set)
Label	REG_SZ	WPB_CXE_R1
ParentBranchBuild	REG_SZ	14219
BuildNumber	REG_SZ	341
TimeStamp	REG_SZ	20141125-1417
Builder	REG_SZ	wpbldlab
MajorVersion	REG_SZ	8
MinorVersion	REG_SZ	10
QFELevel	REG_SZ	14219

- **SYSTEM\Versions:** the specific operating system installed in the device (MajorVersion, MinorVersion) and a time stamp related to the last update of the system.

Name	Type	Value
(Default)	REG_SZ	(value not set)
ActiveTimeBias	REG_DWORD	0xFFFFFFC4 (429
Bias	REG_DWORD	0xFFFFFFC4 (429
DaylightBias	REG_DWORD	0xFFFFFFC4 (429
DaylightName	REG_SZ	@tzres.dll,-321
DaylightStart	REG_BINARY	00 00 03 00 05 00
StandardBias	REG_DWORD	0x00000000 (0)
StandardName	REG_SZ	@tzres.dll,-322
StandardStart	REG_BINARY	00 00 0A 00 05 00
TimeZoneKeyName	REG_SZ	W. Europe Stand
DynamicDaylightTimeDisabled	REG_DWORD	0x00000000 (0)

- **SYSTEM\ControlSet001\Control\TimeZoneInformation:** information about the Timezone set in the system

Name	Type	Value
(Default)	REG_SZ	(value not set)
FullProcessInformationSID	REG_BINARY	01 06 00 00 00 00
ComponentizedBuild	REG_DWORD	0x00000001 (1)
CSDBuildNumber	REG_DWORD	0x00004035 (164
CSDReleaseType	REG_DWORD	0x00000000 (0)
CSDVersion	REG_DWORD	0x00000000 (0)
Directory	REG_EXPAN...	%SystemRoot%
ErrorMode	REG_DWORD	0x00000000 (0)
NoInteractiveServices	REG_DWORD	0x00000001 (1)
ShellErrorMode	REG_DWORD	0x00000001 (1)
SystemDirectory	REG_EXPAN...	%SystemRoot%

- **SYSTEM\ControlSet001\Control\Windows:** information about the operating system, together with the date of final shutdown expressed in the MS FileTime format (100-nanoseconds since January 1, 1601)

/ NATIVE APPLICATIONS

Several research activities have been developed in relation to the storage of native applications data by the Windows Phone operating system.

The most interesting files are stored inside the WPCOMMSSERVICES user folder and in particular:

- AppData\Local\UserData\Phone: an ESE database that contains the call history
- AppData\Local\Unistore\Store.vol: an ESE database that contains information regarding contacts, SMS/MMS messages and appointments

Phone database contains the call history in the CallHistory table. The main fields of this table are:

- Type:
  - 0 = Outgoing Call
  - 1 = Incoming Call
  - 2 = Lost Call
- Raw Number: calling/called number
- Raw CallerID: Caller ID
- Resolved Name: contact name in the address book
- Start Time: the start of the call stored in MS FileTime format
- End Time: the end of the call stored in MS FileTime format

The address book is stored in the Store.vol database and specifically within the Contact table. The most relevant information that can be found associated with a contact is:

- Name
- Telephone numbers (e.g. Mobile, Home, Work, etc.)
- Email(s)
- Avatar
- Work
- Address

SMS/MMS messages are stored in the Store.vol database and specifically within the Message table. The most important stored information is:

- • Type:
  - 1 = Received
  - 33 = Sent
- Label:
  - IPM.SMSText
  - IPM.MMSText
- Text
- Date and time, store MS FileTime format

/ JTAG

JTAG is becoming more and more important to extract Physical Dump of Windows Phone devices. Various hardware tools and techniques have been developed to extract data from this kind of device. On YouTube you can also find interesting references on how to perform this extraction on some specific devices:

- Nokia Lumia 520 – https://www.youtube.com/watch?v=HjkrSxbMoTs
- Nokia Lumia 535 – https://www.youtube.com/watch?v=w-sizgMUrcs
- Nokia Lumia 920 – https://www.youtube.com/watch?v=TmMAOLdZolQ

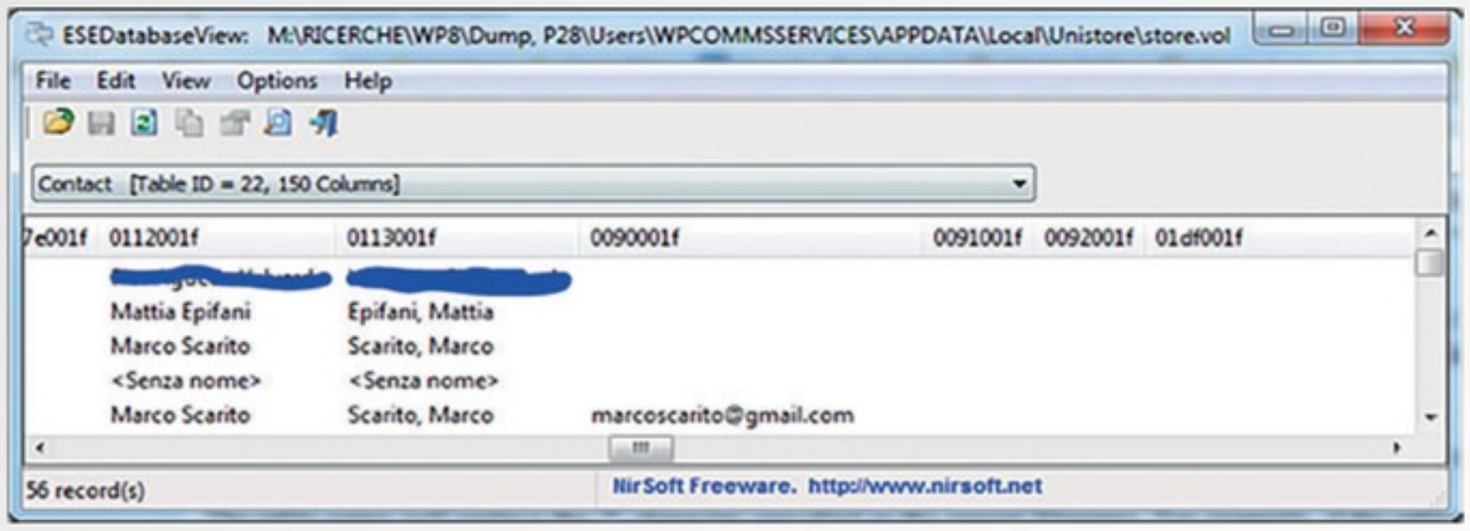


Figure 6. Contact Table

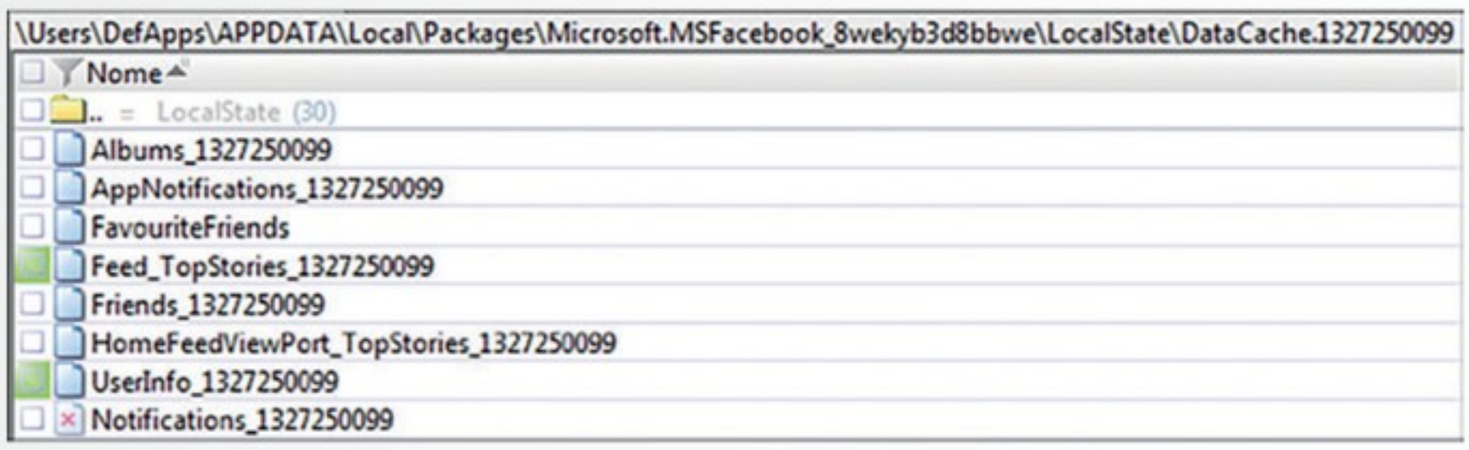


Figure 7. Facebook App Data

The MMS attachments are stored in the SharedData\Comms\Unistore\data folder and are linked to the messages by the data stored in the Attachment table within Store.vol database.

Appointments are stored in the Store.vol database and specifically within the Appointment table. The most important stored information is:

- Start date, stored in MS FileTime format
- Duration (in minutes)
- Type (All Day, Appointment)
- Place
- Text

Internet Explorer is the native browser installed on Windows Phone. It stores data in the same way as the Desktop version inside the DefApps user folder. The most interesting files are:

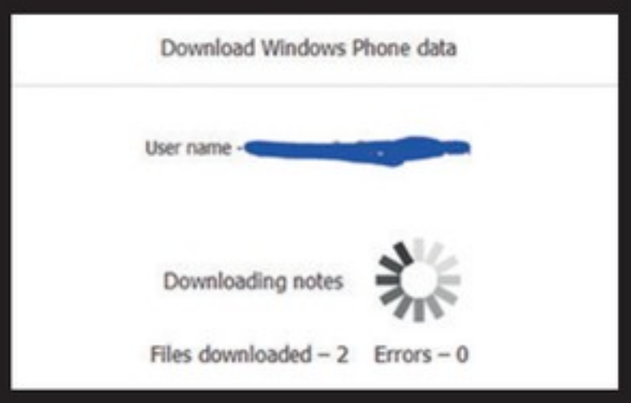
- AppData\Local\Microsoft\Windows\WebCache\WebCacheV01.dat, an ESE database that contains the URL of cached content with the last visited date
- AppData\INTERNETEXPLORER\INetCache folder that contains cached files
- AppData\INTERNETEXPLORER\INetCookies folder that contains cookies
- \SharedData\InternetExplorer\Favorites folder that contains Internet Explorer favorites

/ THIRD PARTY APPLICATIONS

Third-party applications downloaded from the Windows Store are installed within the Data partition. The application itself (executable and libraries) is installed in the path \Programs\WindowApps\[StoreID] or [Application Name]. Within each folder there is a file named WMAppManifest.xml containing key information on the specific application (ID, name, icons, capabilities, and so on).

/ GEOLOCATION DATA

Geolocation Data is typically really useful in the forensic analysis of a mobile device. Some interesting research on this topic is available on the blog post Finding Geo (http://cheeky4n6monkey.blogspot.it/2015/10/finding-geo.html). The article particularly highlights the pagefile.sys as the best place to look for textually encoded latitude/longitude data. Other important elements to consider are Internet Explorer cache, app logs, device camera picture/video files and the Registry.



The application data is stored in the user profile DefApps. In Windows 8 data is stored in the path AppData\[Store ID] or [Application Name] while in Windows Phone 8.1 data is stored in the path AppData\Local\Packages\[Store ID] or [Application Name].

The internal organization of the data is handled by the specific application, which has in any case the limit of being able to write only in the assigned folder or in folders shared with other applications (e.g. the SharedData folder or the Public user folder). Some of the most used Windows Phone third party apps are WhatsApp, Facebook and Skype.

WhatsApp application stores most of its information in the Local sub-folder inside SQLite databases:

1. Settings.db: contains UserID and phone number
2. Contacts.db: contains contacts' name, number, status, profile picture ID and blocked users list
3. Calls.db: contains call history (for calls made through WhatsApp)
4. Stats.db: contains statistical information about app usage
5. Message.db: contains information about messages

- Message Type (Received/Sent)
- Contact
- Date and Time
- Text
- Remote Media URL
- Local Media URL
- Media content size

A backup copy of the message.db is made every day by the application in SharedData\OEM\Public\WhatsApp\Backup folder. Contacts' profile pictures are stored in Profile Pictures sub-folder.

Facebook application stores its data JSON files typically inside folder \Local\DataCache.[FacebookID]. Some of the information that you can find is:

1. User information
2. Friends with ID, Full Name, Profile URL, Email, Birthday, Phone
3. Posts
4. Pages
5. Groups
6. Events
7. Photos and Albums

Local\Images folder contains contact's profile photo and INetCache folder contains App cache file.

Skype application stores its data in a profile folder that is equivalent to the Skype desktop application and so it can be analyzed by using traditional Skype forensic tools (e.g. SkypeLogView, Skypextractor). The user profile is stored in \Local\[SkypeID] folder.

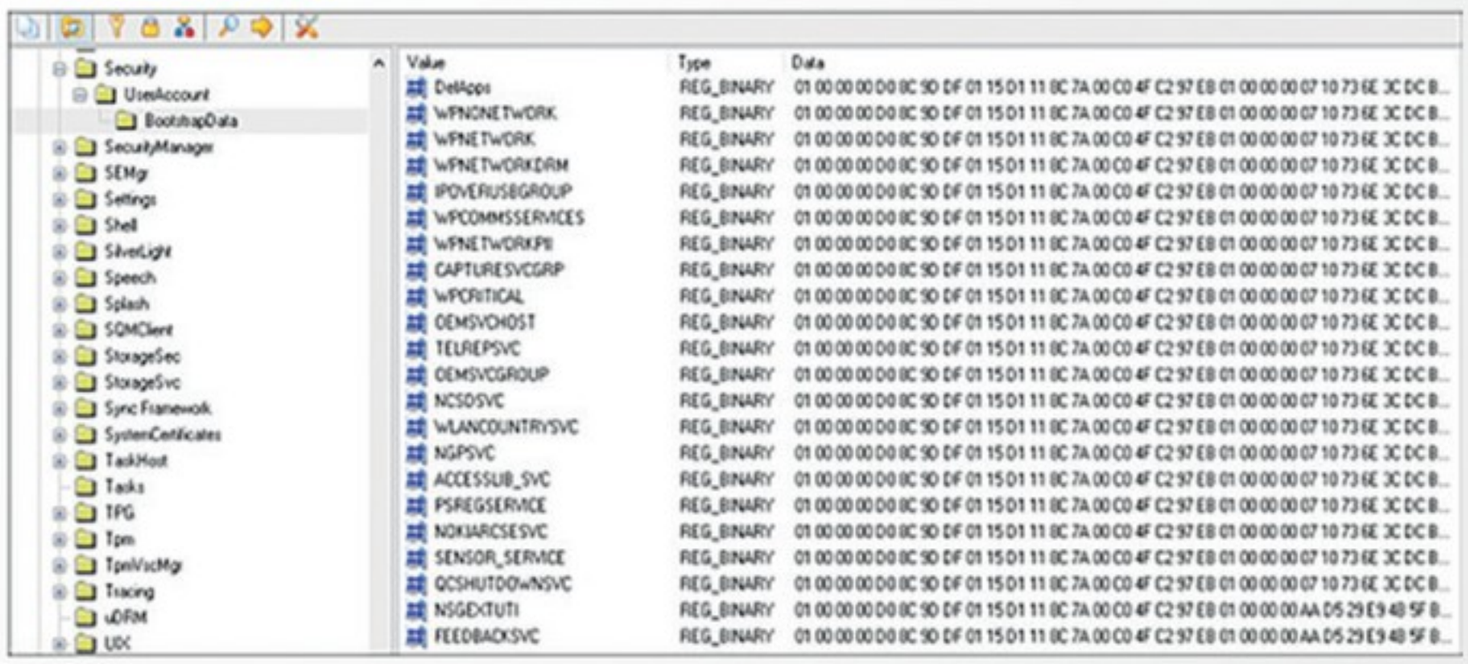


Figure 8. BootstrapData

```
dpapilab> blobdec.py
                    --security=SECURITY --system=SYSTEM --masterkey=sysmk
                    --entropy_hex=626BEDCBCA025E41847E3393369C2E5E
                    bootstrap_DefApps.blob
dpapilab>
E6056E45B3425B7BAB7E328971D8570DC0215D6821657AE0C3F6DD6F8059E3C283838A5CFAA30A2F7547EE12F766ADDFE3F03D22FA
E3DCDA36BA37ECED8807B7C5015E02FB4EF6160754C5ADEB1D1B4E292FED8419D986C3EE8A08901C85A34ABB35F40A770CB3149338
3602C898B9352884195021BBDFD026F452CBA22B2E6F
```

Figure 9. Entropy Hex

WINDOWS PHONE SECRETS

We have already mentioned that there are 25 default system user accounts. Every single user has its own password and from the SYSTEM and SAM registries we can get the corresponding NTLM hashes. Is it possible to crack these NTLM hashes? Not really, since passwords are 256 bytes long and pseudo-random generated when the phone is initialized. The phone needs to know users' passwords to manage accounts, as every Windows operating system does: the "secret" lies in the SOFTWARE hive and specifically in the Microsoft\Security\UserAccount\BootstrapData key. Under that key there are as many values as user accounts, each value representing a DPAPI (Data Protection API) blob.

These blobs are protected with a System Master Key, so they can be decrypted by extracting the DPAPI_SYSTEM password from the LSA Secrets stored in the Security registry; for more details on this please refer to previous research on this topic by Francesco Picasso [13]. To decrypt we need another component, the so called "entropy". In Windows Phone 8/8.1 the entropy is hardcoded in a system DLL called AccountProvSvc.dll and corresponds to 626BEDCBCA025E41847E3393369C2E5E. With this information we are able to decrypt users' passwords.

Once we know the specific system user password we can decrypt every user DPAPI Blob. For example the Windows Mail application uses Vaults to protect account tokens and passwords. And the Mail app is managed by the WPCOMMSERVICES account so, when we know its password, we can get mail passwords or the token. Again, detailed references are available from Francesco Picasso's research [15]. With the same approach we are able to extract the Author token from the Facebook or Twitter app.

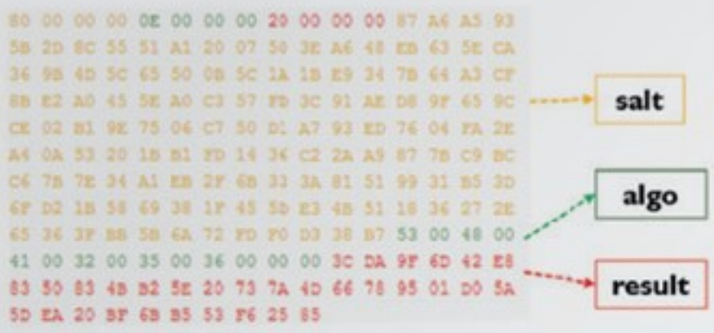


Figure 10. Cracking the PIN Code

An interesting point is that by applying the same rules we can crack the PIN code offline in a really easy way by taking some simple information from the software hive. In particular the SOFTWARE hive contains the Microsoft\Comms\Security\DeviceLock\Object21 hive and specifically the CredentialHash value. These bytes contain three elements: salt, hash algorithm and the hashed pin + salt result. The pin check is done by the following operation:

```
HashAlgo(UTF-16-LE-
NoTrailing0(PIN) + salt)
```

Other researchers [8] have found the CredentialHash value stored in Object31 keys, while previous CredentialHash (e.g. previous PIN codes) have been found in Object736 and Object44 keys.

You might ask why do we have to crack the PIN if we already have a physical dump? At least two examples are useful to provide an answer:

1. Not all the apps are easy to parse from the DD image
2. Starting from Windows 8.1 the user can decide to install apps on the external SD card, in this case information on the SD card is encrypted in a way that depends on both the hardware and the PIN.
   So also with the physical dump of the internal memory and the physical dump of the SD card we do not have access to the data and the only way is to turn on the phone, insert the PIN and browse through app data

The authors thank Marco Scarito (@marcoscarito) and Pasquale Stirparo (@pstirparo) for their contribution in writing and reviewing this article and Cindy Murphy (@CindyMurph) and Adrian Leong (@Cheeky4n6Monkey) for the impressive work and research on Windows Phone 8. /

REFERENCES

1. Windows Phone Architecture Overview
https://sysdev.microsoft.com/en-us/Hardware/oem/docs/Getting_Started/Windows_Phone_architecture_overview
2. Windows Phone Folder Layout
https://sysdev.microsoft.com/en-us/Hardware/oem/docs/Phone_Bring-Up/Folder_layout
3. Windows Phone Main OS folder layout
https://sysdev.microsoft.com/en-us/Hardware/oem/docs/Phone_Bring-Up/Main_OS_folder_layout
4. Windows Phone User Data folder layout
https://sysdev.microsoft.com/en-us/Hardware/oem/docs/Phone_Bring-Up/User_data_folder_layout
5. Windows Phone Removable User Data folder layout
https://sysdev.microsoft.com/en-US/Hardware/OEM/docs/Phone_Bring-Up/Removable_user_data_folder_layout
6. Monkeying around with Windows Phone 8.0
http://cheeky4n6monkey.blogspot.it/2014/06/monkeying-around-with-windows-phone-80.html
7.Windows Phone 8.0 SMS, Call History and Contacts Scripts
http://cheeky4n6monkey.blogspot.it/2014/10/windows-phone-80-sms-call-history-and.html
8. "Awesome" Windows Phone 8 Stuff
http://cheeky4n6monkey.blogspot.it/2014/10/awesome-windows-phone-8-stuff.html
9. Chunky4n6Monkey!
http://cheeky4n6monkey.blogspot.it/2015/07/chunky4n6monkey.html
10. Finding Geo
http://cheeky4n6monkey.blogspot.it/2015/10/finding-geo.html
11. Windows Phone 8.10 MMS (for Lumia 530)
http://cheeky4n6monkey.blogspot.it/2015/12/windows-phone-810-mms-for-lumia-530.html
12. Cheeky4n6monkey Git Hub repository
https://github.com/cheeky4n6monkey/4n6-scripts
13. Happy DPAPI!
http://blog.digital-forensics.it/2015/01/happy-dpapi.html
14. Windows DPAPI Laboratory
https://github.com/dfirfpi/dpapilab
15. Windows Revaulting
http://blog.digital-forensics.it/2016/01/windows-revaulting.html
16. Windows Phone PIN cracking
http://blog.digital-forensics.it/2015/07/windows-phone-pin-cracking.html
17. Window Phone PIN cracking tool
https://github.com/RealityNet/hotoloti/blob/master/sas/winphonepincrk.py

/ CHEEKY4N6MONKEY

The Cheeky4n6Monkey Blog maintained by Adrian Leong is the best available resource for Window Phone 8 forensic analysis and contains the results of several researches made on partitions' layout, file system structure, native applications databases (call history, SMS, MMS, contacts) and geolocation information.
http://cheeky4n6monkey.blogspot.com/

/ AUTHOR BIOGRAPHY

Mattia Epifani is partner and founder at REALITY NET – System Solutions, where he works as a senior consultant in Digital Forensics, Forensic Readiness, Mobile Security and Incident Response. He obtained a University Degree in computer science in Genoa (Italy) and a post-graduate course in Computer Forensics and Digital Investigations in Milan. He works as a digital forensics analyst for judges, prosecutors, lawyers and private companies, both as Court Witness Expert and Digital Forensics Expert. He has obtained several certifications in Digital Forensics and Security (GCFA, GREM, GNFA, GMOB, CIFI, CEH, CHFI, CCE, ACE, AME, MPSC, ECCE). He is a regular speaker on Digital Forensics matters in different Italian and European universities and events. He is a member of DFA, IISFA, ONIF and T&L Center. Co-author of the book "Learning iOS Forensics" edited by PacktPub in March 2015.

/ AUTHOR BIOGRAPHY

Francesco Picasso is a partner at REALITY NET – System Solutions, where he works as a DFIR and Information Security specialist. He obtained the University Degree in Computer Engineering and a Ph.D. in "Intelligent Electronics for Security" both from Genoa University. After a post-graduate course in Computer Forensics and Digital Investigations in Milan (2008), he started the work in Digital Forensics and Incident Response field. He is GIAC Certified Forensic Analyst (GCFA) since 2009 and GIAC Certified Incident Handler (GCIH). He also obtained the Certified Information Forensics Investigator (CIFI) and European Certificate on Cybercrime and Digital Evidence (ECCE) certifications. He is a regular speaker on Digital Forensics and Information Security matters in Italian universities and national/international events. Not without difficulty, he tries to keep up to date with the matter and, aware that the sharing of knowledge and experience is valuable if not essential, he focuses on sharing part of the daily work on the Reality Net blog (http://blog.digital-forensics.it) and on GitHub Hotoloti repository (https://github.com/RealityNet/hotoloti).