DISCOVERING WINDOWS PHONE 8 ARTIFACTS AND SECRETS

MATTIA EPIFANI – FRANCESCO PICASSO – MARCO SCARITO

DFRWS 2016 EU – LAUSANNE

30/03/2016

WINDOWS PHONE 8 DATA ACQUISITION

Physical Acquisition

- **Best acquisition** (bit-by-bit image) but **not available for all models**
- Generally made with JTAG or Chip-Off
- **Cellebrite supports 21 models** with known boot vulnerabilities

Logical and File System acquisition

- Phone **needs to be unlocked or PIN must be known**
- Various forensics tools (Cellebrite, XRY, Oxygen, MobilEdit, etc.) permit to extract **Media files** (Pictures, Videos, Music, Documents) through **USB connection**
- Cellebrite and Oxygen Forensic permit to recover **Contacts** and **Call History** through **Bluetooth connection**

App and manual acquisition

- Phone **needs to be unlocked or PIN must be known**
- Microsoft "**contacts+message backup**" App saves **Contacts** and **Messages** on SD Card
- Microsoft "**Project my Phone**" PC application permits to browse through the phone from a PC

Cloud acquisition

- **Associated Microsoft user credentials must be known**
- Elcomsoft Phone Breaker and Oxygen Forensic support **Contacts**, **Messages** and **Notes** acquisition

WINDOWS PHONE 8 DATA ACQUISITION – JTAG EXTRACTION

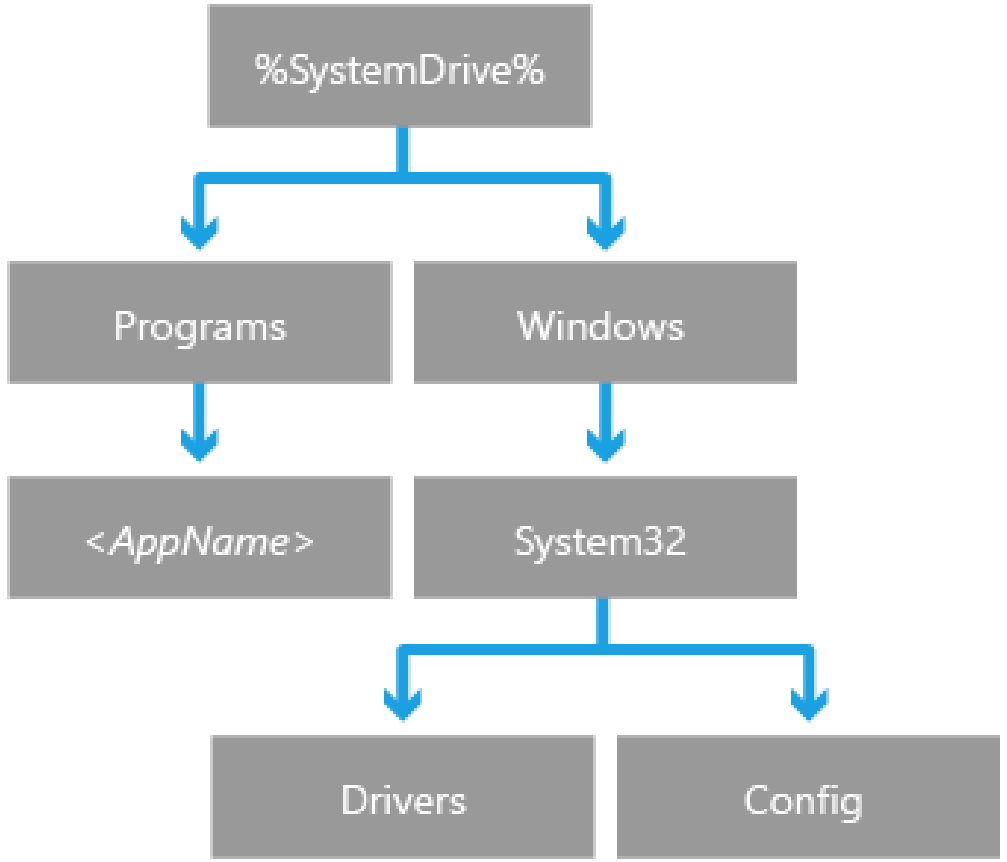
- RIFF Box
- ATF Box
- Lumia 535 eMMC Dump
  - https://www.youtube.com/watch?v=w-sizgMUrcs
- Many boxes / jigs / adapters
- Teel Tech, among others, provides kits and training

PARTITIONS

- **28 partitions**
- Most useful:
  - Partition27 **Main OS**
  - Partition28 **Data**
- Other partitions depend on the SoC (bootloader, configurations backup etc.)
- **External SD Card** is mapped as drive **D:**

MAIN OS PARTITION

- Equivalent to the Windows partition in a PC, without Users folder
- **Programs** folder contains executables from built-in applications
- **Windows** folder contains **windows registry files**, executables, dlls, drivers, and so on



MAIN OS PARTITION

- It uses **NTFS** (so you can find $MFT, $LogFile, $UsnJrnl)
- It contains
  - System registry hives (Software, System, Security, SAM)
  - Pagefile.sys
- It doesn't contain
  - Windows Events Logs
  - Hiberfil.sys
  - Recycle Bin
  - Prefetch
  - Shellbags

REGISTRY
SYSTEM\PLATFORM\DEVICETAGETINGINFO

- Firmware version, phone model, nation, …

Name	Type	Value
(Default)	REG_SZ	(value not set)
PhoneFriendlyName	REG_SZ	Windows Phone
PhoneFirmwareRevision	REG_SZ	3058.50000.1430.0005
PhoneRadioSoftwareRevision	REG_SZ	2.0.242037.8
PhoneSupportLink	REG_SZ	http://link.nokia.com/support
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 Variant
PhoneManufacturerModelName	REG_SZ	RM-914_eu_italy_215
PhoneModelName	REG_SZ	Lumia 520
PhoneOEMSupportLink	REG_SZ	http://link.nokia.com/support
PhoneManufacturer	REG_SZ	NOKIA
PhoneHardwareVariant	REG_SZ	RM-914
USSVersion	REG_SZ	8.10.15150.265
TimeMarker	REG_DWORD	0x00000000 (0)

REGISTRY
SYSTEM\VERSIONS

- Operating system version, last OS update, …

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

REGISTRY
SYSTEM\CONTROLSET001\CONTROL\TIMEZONEINFORMATION

- Timezone

Name	Type	Value
(Default)	REG_SZ	(value not set)
ActiveTimeBias	REG_DWORD	0xFFFFFFC4 (4294967236)
Bias	REG_DWORD	0xFFFFFFC4 (4294967236)
DaylightBias	REG_DWORD	0xFFFFFFC4 (4294967236)
DaylightName	REG_SZ	@tzres.dll,-321
DaylightStart	REG_BINARY	00 00 03 00 05 00 02 00 00 00 00 00 00 00 00 00
StandardBias	REG_DWORD	0x00000000 (0)
StandardName	REG_SZ	@tzres.dll,-322
StandardStart	REG_BINARY	00 00 0A 00 05 00 03 00 00 00 00 00 00 00 00 00
TimeZoneKeyName	REG_SZ	W. Europe Standard Time
DynamicDaylightTimeDisabled	REG_DWORD	0x00000000 (0)
ID	REG_DWORD	0x0000053C (1340)

REGISTRY
SYSTEM\CONTROLSET001\CONTROL\WINDOWS

- Last shutdown in MS FILETIME format (100-nanoseconds since January 1, 1601)

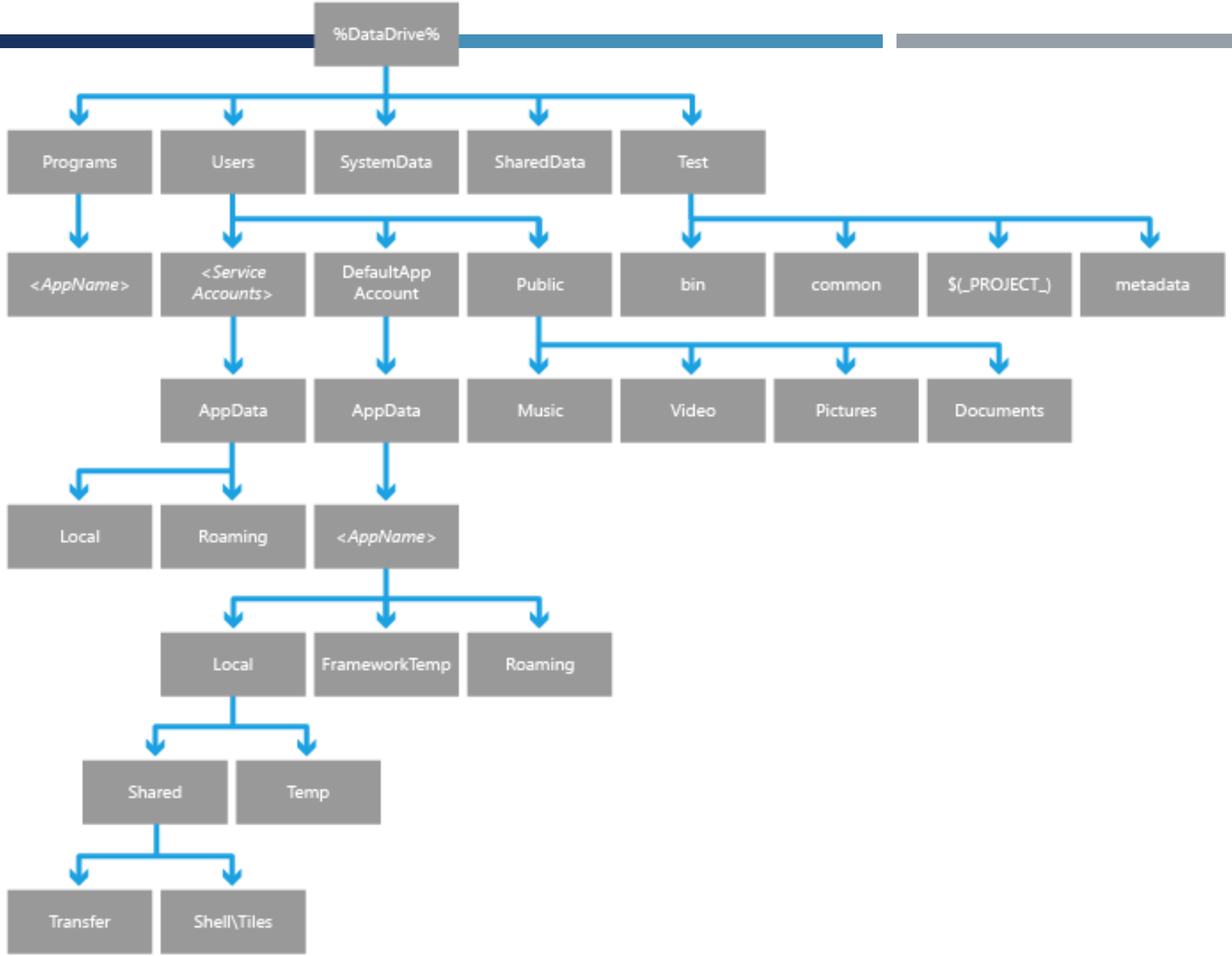
Name	Type	Value
(Default)	REG_SZ	(value not set)
FullProcessInformationSID	REG_BINARY	01 06 00 00 00 00 00 05 50 00 00 00 58 DB E7 73 52 1C 19 21 1C 4E 21 BE 8D 29 A9 C0 C4 E6 E7 92
ComponentizedBuild	REG_DWORD	0x00000001 (1)
CSDBuildNumber	REG_DWORD	0x00004035 (16437)
CSDReleaseType	REG_DWORD	0x00000000 (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%\system32
ShutdownTime	REG_BINARY	2E 1E 3C E1 71 08 D0 01

DATA PARTITION

- It uses **NTFS** (so you can find $MFT, $LogFile, $UsnJrnl)
- It contains information about **user activities**, both for native and third party Apps
- **Programs** Executables of the installed Third-party Apps
- **Users** User accounts
- **SystemData** Log files and updates information
- **SharedData** Data shared between Apps

USERS FOLDER

- There are actually **25 default user accounts**
- Most user accounts are used by the OS for services
- Three types of user accounts:
  - **Service Accounts** — System users for managing services and system apps
  - **DefApps** — User handling the data of third party apps and (some) native apps
  - **Public** — User storing media files


%DataDrive%
Programs
Users
SystemData
SharedData
Test
<AppName>
<Service Accounts>
DefaultApp Account
Public
bin
common
$(_PROJECT_)
metadata
AppData
AppData
Music
Video
Pictures
Documents
Local
Roaming
<AppName>
Local
FrameworkTemp
Roaming
Shared
Temp
Transfer
Shell\Tiles

WPCOMMSSERVICES USER

- Most interesting user for a digital forensics analyst
- In particular it contains **2 ESE (Extensible Storage Engine) Databases**
  - **\Users\WPCOMMSSERVICES\APPDATA\Local\Unistore\store.vol**
    - Address Book, Calendar, SMS/MMS Messages, Emails, …
  - **\Users\WPCOMMSSERVICES\APPDATA\Local\UserData\Phone**
    - Call History
- They can be analyzed with ESE DB Viewer/Parser (e.g. Libesedb, ESEDatabaseView)

PHONE DATABASE

- **Call History**
  - **CallHistory** DB Table
  - **Type**
    - Outgoing call 1
    - Incoming call 2
    - Lost call 3
  - **Raw Number** Calling/Called number
  - **Resolved Name** Contact name in the AddressBook (if available)
  - **Start Time** FileTime format (100-nanoseconds from 1st January 1601)
  - **EndTime** FileTime format (100-nanoseconds from 1st January 1601)

STORE.VOL DATABASE

- **Address Book**
  - **Contact** DB Table
  - Name, Surname, Address(es), Phone(s), Email(s), Avatar, etc.
- **Calendar**
  - **Appointment** DB Table
  - Start date, duration (in minutes), type (all day, appointment), place, text, etc.
- **SMS/MMS Messages**
  - **Message** DB Table
  - Type
    - 1 Received
    - 33 Sent
  - Label
    - IPM.SMSText
    - IPM.MMSText
  - Date and time
    - FileTime format (100-nanoseconds from 1st January 1601)
  - Text

PYTHON SCRIPTS FROM **CHEEKY4N6MONKEY**

- Opensource scripts to parse and analyze Contacts, Call History, SMS/MMS Messages
- **https://github.com/cheeky4n6monkey/4n6-scripts**

Windows Phone 8.0 SMS, Call History and Contacts Scripts

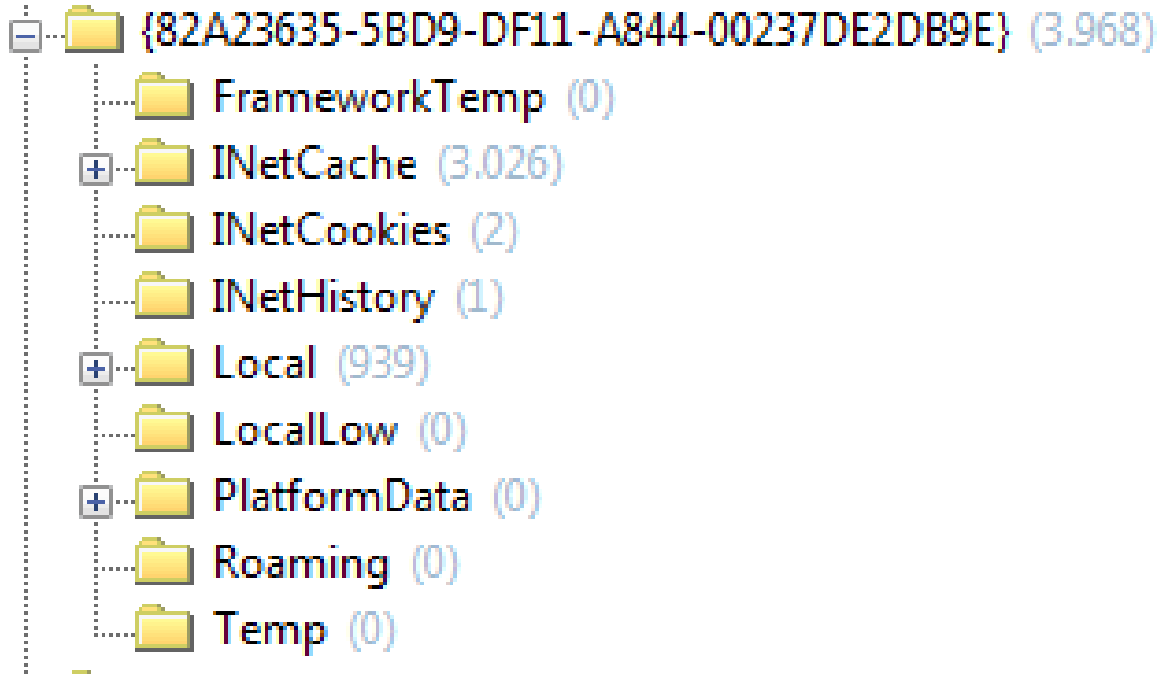
- References
  - http://cheeky4n6monkey.blogspot.it/2015/12/windows-phone-810-mms-for-lumia-530.html
  - http://cheeky4n6monkey.blogspot.it/2015/10/finding-geo.html
  - http://cheeky4n6monkey.blogspot.it/2015/07/chunky4n6monkey.html
  - http://cheeky4n6monkey.blogspot.it/2014/10/windows-phone-80-sms-call-history-and.html
  - http://cheeky4n6monkey.blogspot.it/2014/06/monkeying-around-with-windows-phone-80.html

PUBLIC USER

- It stores **multimedia files**
  - Documents
  - Downloads
  - Music
  - Pictures
    - Camera Roll — Photos and videos taken with the device internal camera
    - Saved Pictures — User-saved images from from applications (e.g. Facebook)
    - Screenshots — Phone monitor screenshots
    - WhatsApp — WhatsApp pictures (if installed, of course)
  - Ringtones
  - Videos
- This part of the file system is accessible when the phone is connected to a Windows PC

DEFAPPS USER

- It stores data for (some) native and third-party applications
- It contains one sub-folder for each application (inside the AppData sub-folder)
  - Preinstalled Apps have their own folders (e.g. INTERNETEXPLORER)
  - Every third party apps is identified by an **App ID** (Windows Phone Store ID) or a **friendly name associated to the AppID**



APPLICATION FOLDER STRUCTURE

- The internal structure of every application folder is consistent

- **INetCache** App cache files (images, videos, HTML content, etc.)
- **INetCookies** App cookies
- **INetHistory** App URL history
- **Local** Application related data (database and config) not shared with other devices
- **Local\Shared** Application data shared with the system (eg. Desktop Tile image)
- **Roaming** Application related data (database and config) shared with Cloud or other devices

INTERNET EXPLORER

- **Cache History**
  - \Users\DefApps\APPDATA\Local\Microsoft\Windows\WebCache\WebCacheV01.dat
  - ESE Database
- **Cached files**
  - \Users\DefApps\APPDATA\INTERNETEXPLORER\INetCache
- **Cookies**
  - \Users\DefApps\APPDATA\INTERNETEXPLORER\INetCookies
- **Favorites**
  - \SharedData\InternetExplorer\Favorites

WHATSAPP

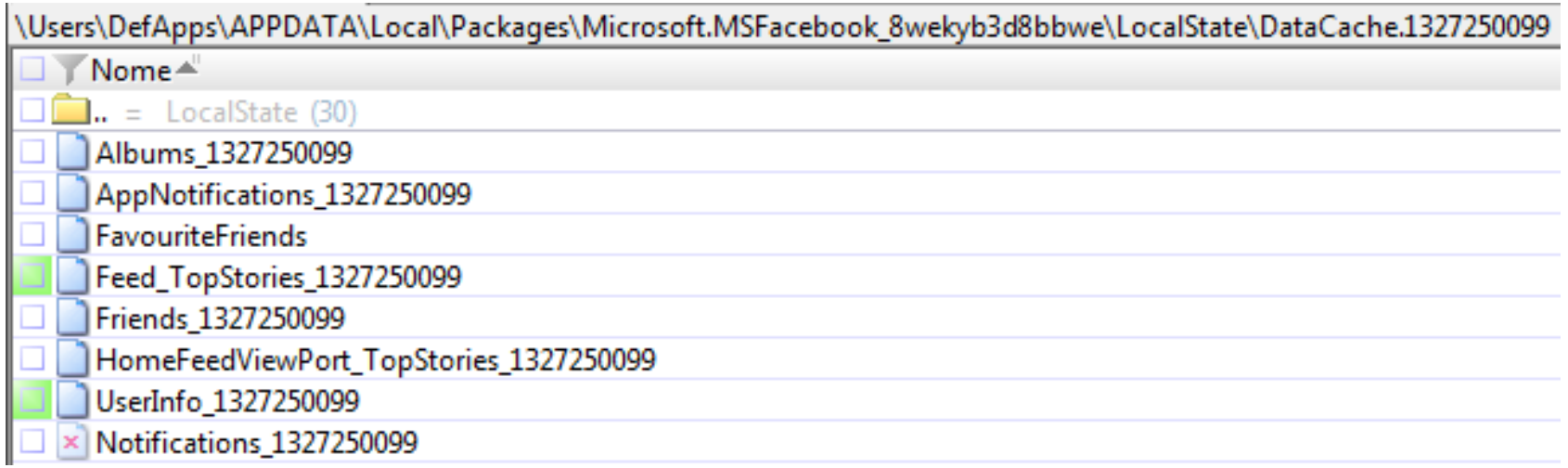
- Most information stored in **Local** subfolder
- It uses SQLite DBs
- **Settings.db** User ID and phone number
- **Contacts.db**
  - **UserStatuses** ID Whatsapp, Contact Name, Status, Photo ID, etc.
    - Contacts profile photo are stored in **ProfilePictures** folder
  - **BlockList** Blocked users
- **Calls.db** Call history (for calls made through WhatsApp)
- **Stats.db** Stats information about app usage

WHATSAPP

- **Messages.db**
  - Message type (Received/Sent)
  - Contacts
  - Date and time
  - Text
  - Remote media URL
  - Local media URL — Typically Public user folder. In some cases **\Local\Shared\Transfer**
  - Media content size
- A backup copy of the messages database is made every day by the application inside the **\SharedData\OEM\Public\WhatsApp\Backup** folder

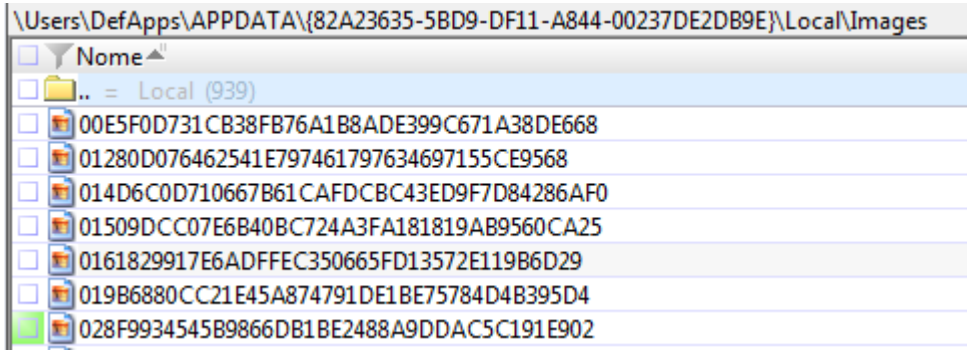
FACEBOOK

- **\Local\DataCache.<FacebookID>** contains JSON files related to:
  - User information
  - Friends
    - ID, Full Name, Profile URL, Email, Birthday, Phone
  - Posts
  - Pages
  - Groups
  - Events
  - Photos and Albums

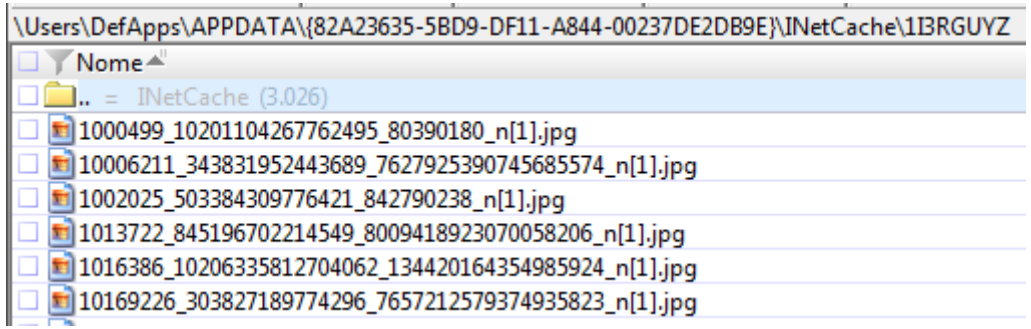


FACEBOOK

- **\Local\Images** contains contacts' profile photo



- **\INetCache** contains App cache files



AND NOW LET'S MOVE TO THE SECRETS!
USER PASSWORDS

shell> pycreddump\pwdump.py **SYSTEM SAM**

Administrator:500:aad3b435b51404eeaad3b435b51404ee:f194e99a9050a026b9445425b4e72c44:::

Guest:501:aad3b435b51404eeaad3b435b51404ee:30f60b1fe0fd69e1eb179d480d54f0d9:::

DefApps:2781:aad3b435b51404eeaad3b435b51404ee:a33a6392a476adfe34294b2029f14708:::

WPNONETWORK:2782:aad3b435b51404eeaad3b435b51404ee:67c041a3ddcc8eeaca176b32a068d97e:::

WPNETWORK:2783:aad3b435b51404eeaad3b435b51404ee:83106e7f2e2921e3597328d8e11eea8:::

WPNETWORKDRM:2784:aad3b435b51404eeaad3b435b51404ee:9fae0b63cd1ba7a85ab0365df786f49a:::

IPOVERUSBGROUP:2786:aad3b435b51404eeaad3b435b51404ee:e0b0fac51f921e5127e12776bee0197e:::

WPCOMMSSERVICES:2788:aad3b435b51404eeaad3b435b51404ee:daa06a338359eed6fc28ddb773369e9f:::

WPNETWORKPII:2790:aad3b435b51404eeaad3b435b51404ee:68981b3c606c20b092366c2c34a8685d:::

CAPTURESVCGRP:2791:aad3b435b51404eeaad3b435b51404ee:ee0cab2b8d7316cc1d109bcc6511533c:::

WPCRITICAL:2792:aad3b435b51404eeaad3b435b51404ee:5fce1c3e7e5b1803a424b51452101ccb:::

OEMSVCHOST:2793:aad3b435b51404eeaad3b435b51404ee:e6d325755d21dcaab124a9bd4957676e:::

TELREPSVC:2794:aad3b435b51404eeaad3b435b51404ee:dcbcde33eda70369566bf2b02149763l:::

OEMSVCGROUP:2795:aad3b435b51404eeaad3b435b51404ee:7550da005cceb21c5ba5c8ecad2ed377:::

NCSDSVC:2796:aad3b435b51404eeaad3b435b51404ee:58686d487dd2ca7503cfe6fd17233d88:::

WLANCOUNTRYSVC:2797:aad3b435b51404eeaad3b435b51404ee:7c72dd4bf4f75fb9570c2058e97f3b4b:::

NGPSVC:2798:aad3b435b51404eeaad3b435b51404ee:1b739d6d3f099293b434c52e193d86b5:::

ACCESSLIB_SVC:2799:aad3b435b51404eeaad3b435b51404ee:2314589 9db9c5a665fc00f3c3cdf636f:::

PSREGSERVICE:2800:aad3b435b51404eeaad3b435b51404ee:8f88aebc4bfd2b6a59ed7406dd41094a:::

NOKIARCSESVC:2801:aad3b435b51404eeaad3b435b51404ee:076d30c9b00c86ab93e761b0d96070ff:::

SENSOR_SERVICE:2802:aad3b435b51404eeaad3b435b51404ee:c182e91ca1361bd390630676ff25e02a:::

QCSHUTDOWNSVC:2804:aad3b435b51404eeaad3b435b51404ee:35007aa28a7b5a86ff38aa485491a272:::

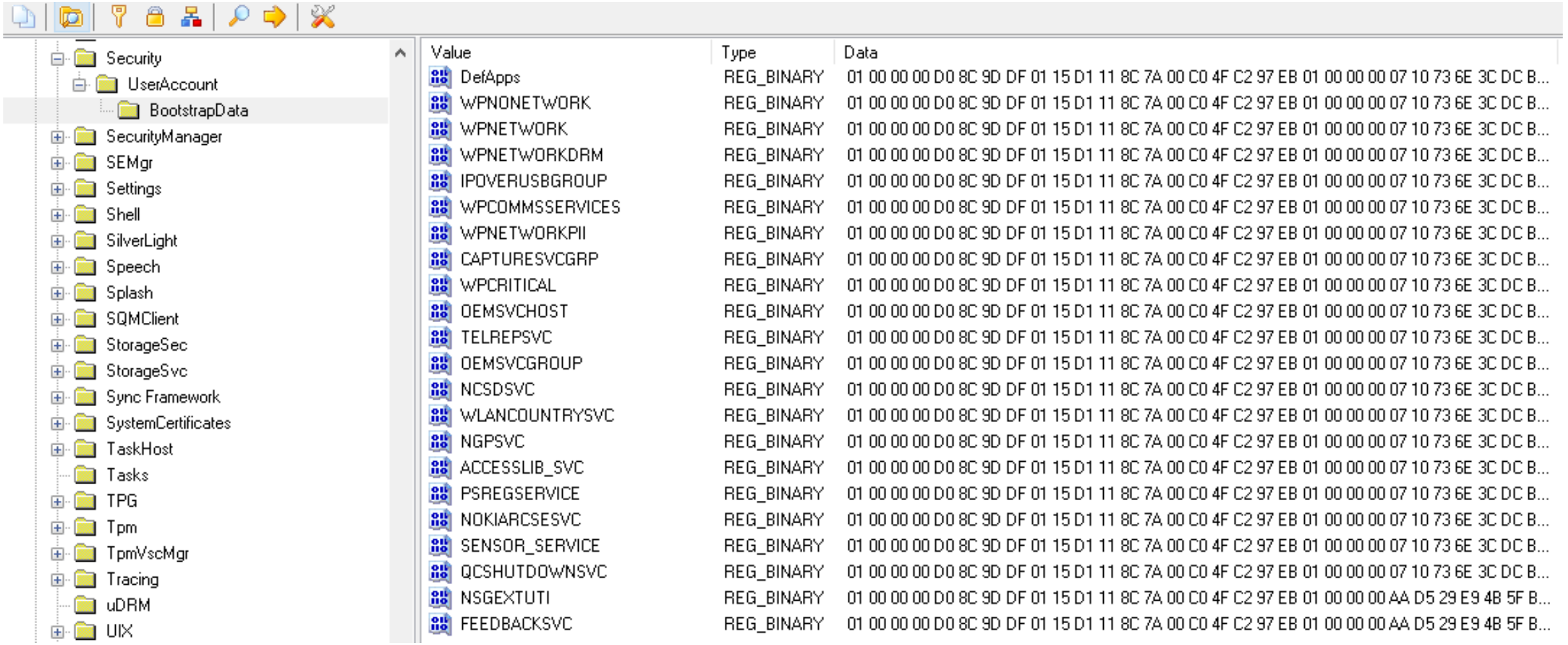
NSGEXTUTI:2805:aad3b435b51404eeaad3b435b51404ee:deb3282673fbb24ec84fbee636a52450:::

FEEDBACKSVC:2806:aad3b435b51404eeaad3b435b51404ee:27da8da6e58d6fde3de25881 82f96f2d8:::

USERS PASSWORDS CRACKING

- Those password ***cannot be cracked*** in practice
- They are **really complex and generated in a pseudo random way when the phone is initialized**
- Anyway the phone must know them…
- Searching for users names in the registry SOFTWARE hive reveals the «mistery»

- SOFTWARE\Microsoft\Security\UserAccount\BootstrapData



- SOFTWARE\Microsoft\Security\UserAccount\BootstrapData

Value	Type	Data
DefApps	REG_BINARY	01 00 00 00 D0 8C 9D DF 01 15 D1 11 8C 7A 00 C0 4F C2 97 EB 01 00 00 00 07 10 73 6E 3C DC B...
WPNONETWORK	REG_BINARY	01 00 00 00 D0 8C 9D DF 01 15 D1 11 8C 7A 00 C0 4F C2 97 EB 01 00 00 00 07 10 73 6E 3C DC B...
WPNETWORK	REG_BINARY	01 00 00 00 D0 8C 9D DF 01 15 D1 11 8C 7A 00 C0 4F C2 97 EB 01 00 00 00 07 10 73 6E 3C DC B...
WPNETWORKDRM	REG_BINARY	01 00 00 00 D0 8C 9D DF 01 15 D1 11 8C 7A 00 C0 4F C2 97 EB 01 00 00 00 07 10 73 6E 3C DC B...

```
00000000: 01 00 00 00 D0 8C 9D DF - 01 15 D1 11 8C 7A 00 C0 |            z  |
00000010: 4F C2 97 EB 01 00 00 00 - 07 10 73 6E 3C DC BE 45 |O         sn<  E|
00000020: B3 41 88 ED EF A5 EE 16 - 00 00 00 00 02 00 00 00 | A             |
00000030: 00 00 10 66 00 00 00 01 - 00 00 20 00 00 00 64 AD |   f          d |
00000040: 7E FA F8 21 12 90 20 F7 - A2 BA A1 65 D7 12 F3 22 |~  !       e   "|
00000050: 0F EA 15 54 A7 12 00 9E - D2 4D 0C CB D4 2A 00 00 |   T     M   *  |
00000060: 00 00 0E 80 00 00 00 02 - 00 00 20 00 00 00 B2 A7 |                |
00000070: BE 9A 00 88 47 68 92 98 - 9C D2 45 B2 B3 90 A4 EA |    Gh    E     |
00000080: 6F 71 63 1E 2E FD 18 18 - 9F 82 53 3C 0B B7 10 02 |oqc .     S<    |
00000090: 00 00 F6 89 1B DB 4C 9B - EF 96 87 97 57 8F 76 4A |      L     W vJ|
000000a0: FD E5 A0 85 E2 D7 1F 57 - 25 A3 CF 0A 40 3C 3C 03 |       W%   @<< |
000000b0: 4A 04 DC C7 2D F4 7B E8 - 9B A8 76 BE 76 28 E8 A2 |J   - {   v v(  |
000000c0: 86 45 7A 99 B2 6B DC 5C - 8A 77 B8 C4 05 5C 5D 43 | Ez  k \ w   \]C|
000000d0: 94 65 AD 45 5D 8B D5 78 - CE 20 E9 62 0B 7E BF 83 | e E]  x   b ~  |
000000e0: BB E2 FD FD 05 97 94 5B - 0B 32 59 30 93 DF 4B 8A |       [ 2Y0  K |
000000f0: ED 04 6E EC 5F 3A 61 E6 - 21 05 B6 8D EB E7 8A 88 |  n _:a !       |
00000100: F4 9A BC 3A CF 41 1A A1 - 7A A1 E3 F6 4D 4A 60 0F |   : A  z   MJ` |
00000110: 03 EC 99 87 F8 43 50 F5 - 1B DA DD 09 B0 61 6C 84 |     CP      al |
00000120: CF 85 FF E8 B0 1B BB 30 - 23 A5 2F 09 3F D7 FE F4 |       0# / ?   |
00000130: 47 3A 98 04 15 6E 1D ED - 0C 41 35 49 F2 62 33 9D |G:   n   A5I b3 |
00000140: 98 D1 54 84 A8 DF BA 70 - 4C E0 8B 68 E9 75 7D 42 |  T    pL  h u}B|
00000150: 63 D4 32 C9 E3 E2 F5 DA - BE DC 34 DB 07 6B D1 69 |c 2       4  k i|
00000160: 33 D6 00 70 0B 05 C7 4C - C1 57 67 77 BD B9 19 ED |3  p   L Wgw    |
00000170: 60 32 BB 2B 77 6A 8A 84 - BC E7 06 66 D7 6D 98 3B |`2 +wj     f m ;|
00000180: 5D CC 0A FA F2 32 26 17 - 87 5A 37 8C 8C 73 EB 77 |]    2&  Z7  s w|
00000190: 7D 84 E2 53 A7 BD 6F FD - FC 4D 24 A5 E0 3F 79 3A |}  S  o  M$  ?y:|
000001a0: 20 B6 F9 70 90 B0 BF 8C - 02 6C CF 49 5D 41 7F D0 |   p     l I]A  |
```

- Values are **DPAPI** blobs
- They can be decrypted with a System Master Key
- For references see
  - http://blog.digital-forensics.it/2015/01/happy-dpapi.html
  - https://github.com/dfirfpi/dpapilab

USERS PASSWORDS DECRYPTING

- The DPAPI System Master Key can be extracted from the **LSA Secrets** (SECURITY Registry)
- **Entropy** must be taken into account
- Hardcoded in **AccountProvSvc.dll**
- 626BEDCBCA025E41847E3393369C2E5E

```
.data:1001C054 unk_1001C054    DCB 0x62 ; b
.data:1001C054
.data:1001C055                 DCB 0x6B ; k
.data:1001C056                 DCB 0xED ; ý
.data:1001C057                 DCB 0xCB ; -
.data:1001C058                 DCB 0xCA ; -
.data:1001C059                 DCB    2
.data:1001C05A                 DCB 0x5E ; ^
.data:1001C05B                 DCB 0x41 ; A
.data:1001C05C                 DCB 0x84 ; ä
.data:1001C05D                 DCB 0x7E ; ~
.data:1001C05E                 DCB 0x33 ; 3
.data:1001C05F                 DCB 0x93 ; ô
.data:1001C060                 DCB 0x36 ; 6
.data:1001C061                 DCB 0x9C ; £
.data:1001C062                 DCB 0x2E ; .
.data:1001C063                 DCB 0x5E ; ^
```

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

dpapilab>
E6056E45B3425B7BAB7E328971D8570DC0215D6821657AE0C3F6DD6F8059E3C283838A5CFAA30A2F7547EE12F766ADDFE3F03D22FA
E3DCDA36BA37ECED8807B7C5015E02FB4EF6160754C5ADEB1D1B4E292FED8419D986C3EE8A08901C85A34ABB35F40A770CB3149338
3602C898B9352884195021BBDFD026F452CBA22B2E6F
```

WHY USERS PASSWORDS?

- Once we know the users passwords **we can decrypt DPAPI Blobs**!
- And Windows Phone is filled by DPAPI blobs
- For example, the **Windows Mail** application uses **Vaults** to protect **account tokens and passwords** (and it's not the only application using them)
- Given the proper legal authority…
- … online **email messages** can be acquired by exploiting such decryption
- Reference: http://blog.digital-forensics.it/2016/01/windows-revaulting.html

WPCOMMSSERVICES AND EMAIL

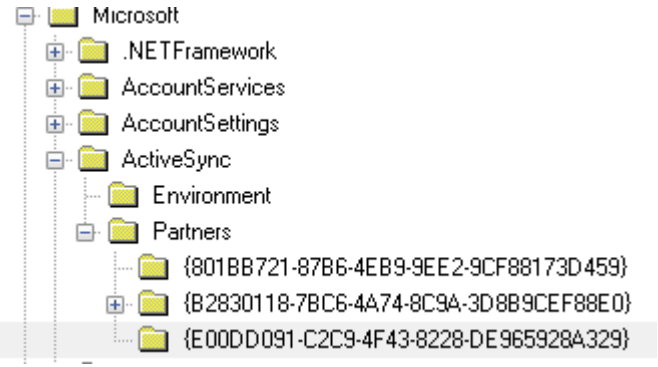
- Email accounts tokens/passwords are kept inside **WPCOMMSSERVICES** user **Vaults**

```
dpapilab> vaultdec.py
    --sid=S-1-5-21-2702878673-795188819-444038987-2788
    --masterkey=\Users\WPCOMMSSERVICES\AppData\Roaming\Microsoft\Protect\S-1-5-21-2421538757-
                1605280464-2344517820-1000

    --password=E91889F98E8A68703ABFC68464B16A1373BB6501192F9D68A5C87C41CF4356185250265073 5EF84ED
               27AF308AAD9E08C031B5FC21C3DDC9C978222D83FC3308498C2AE0528A38EB086841E2743DE1BCEC18
               FCEDB0775DEA869BF98DEFCE6B5B58A58B58275F42BDA15049DCD9AA3953782E4CD4109CB22795311
               ADBF8E2ABAD1

      \Users\WPCOMMSSERVICES\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704DDB28
```

WPCOMMSSERVICES AND EMAIL

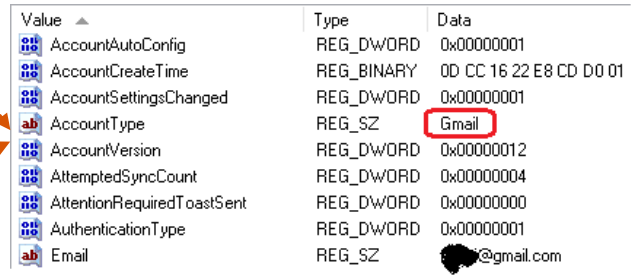
- **Resource** value contain a GUID that must be correlated with Registry keys
  - **SOFTWARE\Microsoft\ActiveSync\Partners**

```
Working on: 2DAF2FE224AF306A198BA169B1534ADCC618B47B.vcrd
…
Attribute: 3 (vault_schema_activesync)
identity: ActiveSyncCredentialDefaultUser
resource: SyncPassword{E00DD091-C2C9-4F43-8228-DE965928A329}MailIncoming
authenticator: WhatCouldPossiblyGoWrong?
```



WPCOMMSSERVICES AND GMAIL

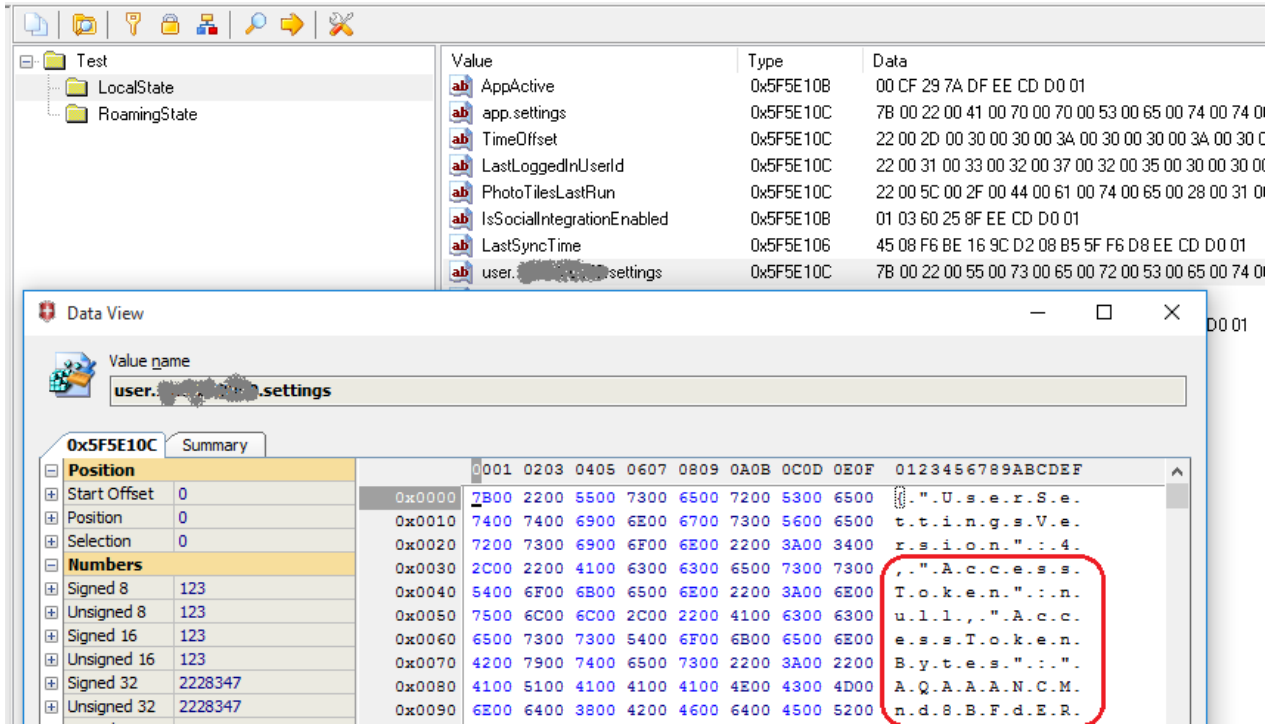
```
Working on: DB580D5ADA46907C4D7218A754491C55CF9C3342.vcrd

Attribute: 3 (vault_schema_activesync)
identity: ActiveSyncCredentialDefaultUser
resource: OAuthRefreshToken{801BB721-87B6-4EB9-9EE2-9CF88173D459}OAuth
authenticator: 1/UyGV1eG2Q7FfabcdEF6nb3dFgr354htJ6K-mgaj2gw

--------------

Working on: FE912C0BD34AD8BC6C00C8E879B2490495AF937E.vcrd

Attribute: 3 (vault_schema_activesync)
identity: ActiveSyncCredentialDefaultUser
resource: SyncPassword{801BB721-87B6-4EB9-9EE2-9CF88173D459}MailIncoming
authenticator: ya29.xBFGoPazVskbtY_HqmExc3PXmVbYhXrYLd3ldzfryQcP65p4CerTGTEVLe_BjqnGHe_
```

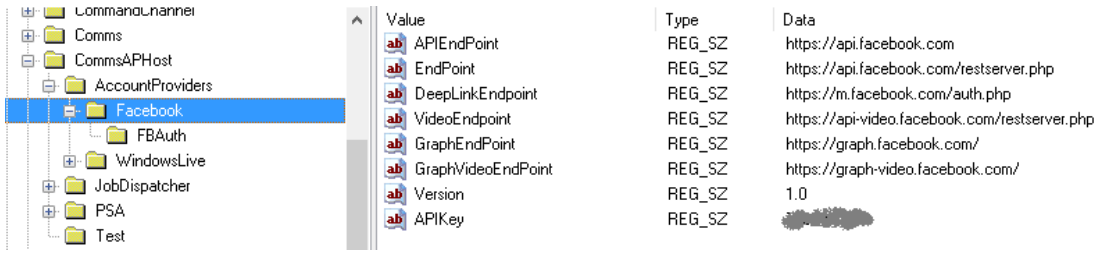


Value	Type	Data
AccountAutoConfig	REG_DWORD	0x00000001
AccountCreateTime	REG_BINARY	0D CC 16 22 E8 CD D0 01
AccountSettingsChanged	REG_DWORD	0x00000001
AccountType	REG_SZ	Gmail
AccountVersion	REG_DWORD	0x00000012
AttemptedSyncCount	REG_DWORD	0x00000004
AttentionRequiredToastSent	REG_DWORD	0x00000000
AuthenticationType	REG_DWORD	0x00000001
Email	REG_SZ	[redacted]@gmail.com

FACEBOOK TOKENS

- **Auth Token** in «[data]\Users\DefApps\APPDATA\Local\Packages\Microsoft.MSFacebook_8wekyb3d8bbwe\Settings»
- **ApiKey** in «[os] \Windows\System32\config\SOFTWARE»





- By using these information with (for example) https://github.com/pythonforfacebook/facebook-sdk you can access online data

TWITTER TOKENS

- **CurrentUserToken** and **CurrentUserTokenSecret** are kept inside the «__ApplicationSettings» file (XML) «\Users\DefApps\APPDATA\Local\Packages\9E2F88E3.Twitter_wgeqdkkx372wm\LocalState__ApplicationSettings»
- **ConsumerKey** and **ConsumerSecret** are hardcoded inside **Twitter.Core.dll** (C#) file, slightly obfuscated.

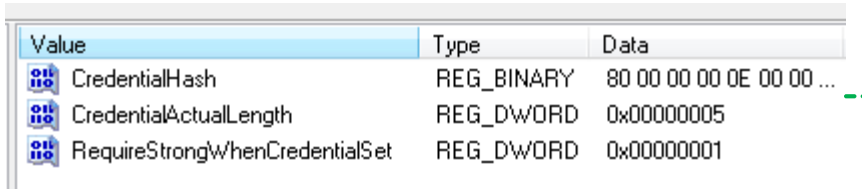
```
static Constants()
{
    Constants.ConsumerKeyBytes = new byte[]
    {
        124,
        [redacted]
        80,
        104,
        [redacted]
    };
    Constants.ConsumerSecretBytes = new byte[]
    {
        [redacted]
        57,
        59,
        [redacted]
```

```
from __future__ import absolute_import, print_function
import tweepy

consumerKeyBytes = [1[redacted]
consumerSecretBytes = [[redacted]
access_token="[redacted]"
access_token_secret="[redacted]"

consumer_key = str(bytearray( x-3 for x in consumerKeyBytes))
consumer_secret = str(bytearray( x-3 for x in consumerSecretBytes))

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.secure = True
auth.set_access_token(access_token, access_token_secret)
user_id = '[redacted]'
api = tweepy.API(auth)
print(api.me().name)
pmsg = api.sent_direct_messages(full_text=True)
for p in pmsg:
    print(p)
    print('-'*79)
```

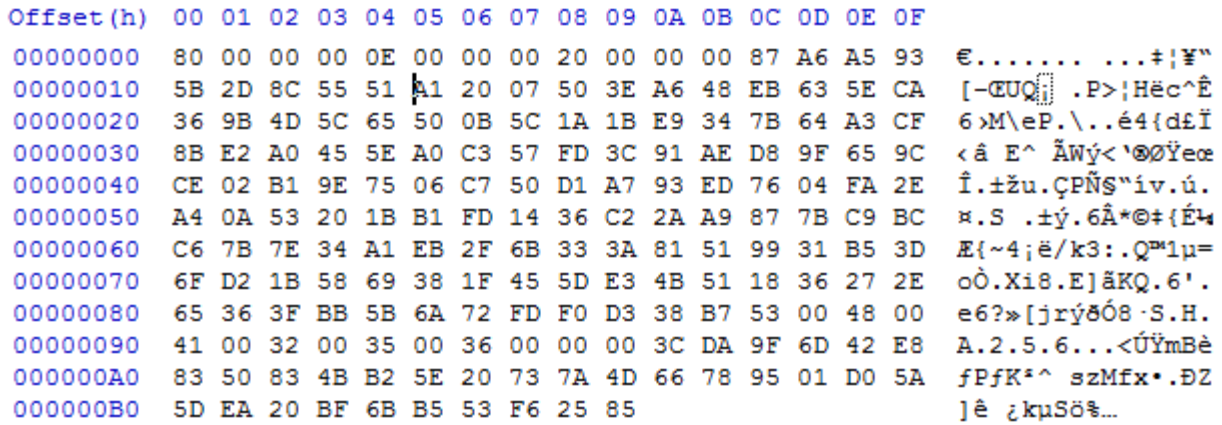
```
Constants.ConsumerKey = Constants.Transform(Constants.ConsumerKeyBytes, 3);
Constants.ConsumerSecret = Constants.Transform(Constants.ConsumerSecretBytes, 3);
Constants.SignupConsumerKey = Constants.Transform(Constants.SignupConsumerKeyBytes, 3);
Constants.SignupConsumerSecret = Constants.Transform(Constants.SignupConsumerSecretBytes, 3);
Constants.SignupAccountToken = Constants.Transform(Constants.SignupAccountTokenBytes, 3);
Constants.SignupAccountTokenSecret = Constants.Transform(Constants.SignupAccountTokenSecretBytes, 3);
Constants.FindFriendsEncryptionKey = Constants.Transform(Constants.FindFriendsEncryptionKeyBytes, 3);
```

PIN CRACKING

- Useful if you need to power on the device
- By having a physical dump (or at least the SOFTWARE hive) you can crack the PIN code in a really easy way
- The PIN is, in reality, like a *screensaver*
- Object21 (SOFTWARE\Microsoft\Comms\Security\DeviceLock\Object21)



Value	Type	Data
CredentialHash	REG_BINARY	80 00 00 00 0E 00 00 ...
CredentialActualLength	REG_DWORD	0x00000005
RequireStrongWhenCredentialSet	REG_DWORD	0x00000001

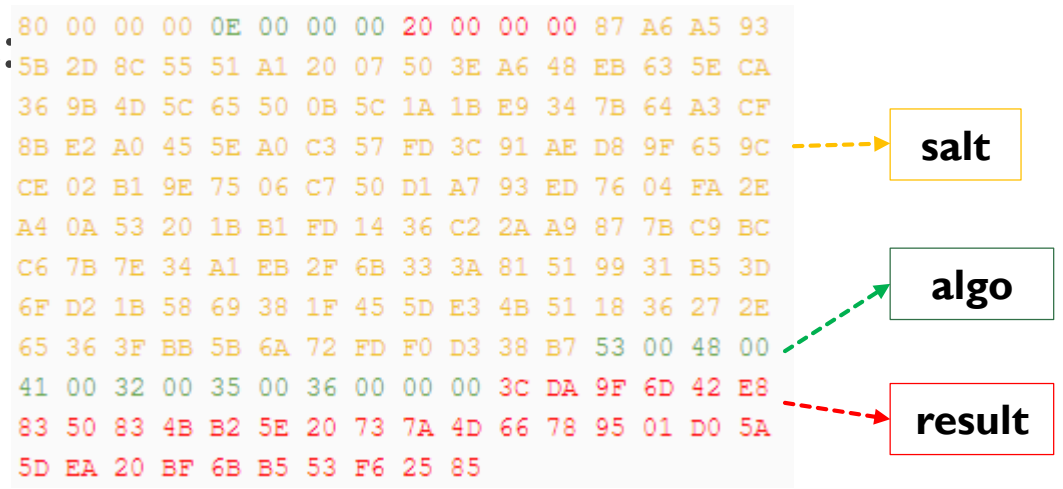


Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	80	00	00	00	0E	00	00	00	20	00	00	00	87	A6	A5	93	€....... ...‡¦¥"
00000010	5B	2D	8C	55	51	A1	20	07	50	3E	A6	48	EB	63	5E	CA	[-ŒUQ¡ .P>¦Hëc^Ê
00000020	36	9B	4D	5C	65	50	0B	5C	1A	1B	E9	34	7B	64	A3	CF	6›M\eP.\..é4{d£Ï
00000030	8B	E2	A0	45	5E	A0	C3	57	FD	3C	91	AE	D8	9F	65	9C	‹â E^ ÃWý<'®ØŸeœ
00000040	CE	02	B1	9E	75	06	C7	50	D1	A7	93	ED	76	04	FA	2E	Î.±žu.ÇPÑ§"ív.ú.
00000050	A4	0A	53	20	1B	B1	FD	14	36	C2	2A	A9	87	7B	C9	BC	¤.S .±ý.6Â*©‡{É¼
00000060	C6	7B	7E	34	A1	EB	2F	6B	33	3A	81	51	99	31	B5	3D	Æ{~4¡ë/k3:.Q™1µ=
00000070	6F	D2	1B	58	69	38	1F	45	5D	E3	4B	51	18	36	27	2E	oÒ.Xi8.E]ãKQ.6'.
00000080	65	36	3F	BB	5B	6A	72	FD	F0	D3	38	B7	53	00	48	00	e6?»[jrýðÓ8·S.H.
00000090	41	00	32	00	35	00	36	00	00	00	3C	DA	9F	6D	42	E8	A.2.5.6...<ÚŸmBè
000000A0	83	50	83	4B	B2	5E	20	73	7A	4D	66	78	95	01	D0	5A	ƒPƒK²^ szMfx•.ÐZ
000000B0	5D	EA	20	BF	6B	B5	53	F6	25	85							]ê ¿kµSö%…

PIN CRACKING

- The **CredentialHash** contains three elements: **salt**, hash **algorithm** and the **hashed pin + salt result**
- The pin check is done by the following

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



- Note that other keys could have the pin information (actually it's unclear why), as **Object31**
- References

code: https://github.com/RealityNet/hotoloti/blob/master/sas/winphonepincrk.py

blog: http://blog.digital-forensics.it/2015/07/windows-phone-pin-cracking.html

WHY PIN CRACKING

- **Why do we have to crack the PIN if we already have a physical dump?**
  - Not all the apps are easy to parse from the DD image
  - 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**

WINDOWS PHONE 8 ARTIFATCS LOCATION

File or Location	Description
Users/WPCOMMSERVICES/APPDATA/Local/UserData/Phone	Call History
Users/WPCOMMSERVICES/APPDATA/Local/Unistore/Store.vol	SMS Contacts Appointments Emails
/SharedData/Comms/Unistore/Data (in various subfolders)	MMS Attachments MMS Formatting Information MMS Message Content
SharedData/Comms/Messaging/Temp/MMS	File attachments from incoming and outgoing mms messages
Users/WPCOMMSERVICES/APPDATA/Temp/RequestManager/Cache	Cached images from message attachments
Users/Public/Pictures/CameraRoll/ WP_YYYYMMDD_###.jpg	Pictures taken with the device Videos taken with the device
Users/Public/Pictures/SavedPictures/	Pictures saved to the device from other sources (Facebook, etc...)
Users/Public/Pictures/Screenshots/	Phone monitor screenshots
Users/DefApps/APPDATA/Local/Microsoft/Windows/WebCache/WebCacheV01.dat	Internet Explorer Cache History
Users/DefApps/APPDATA/INERNETEXPLORER/INetCache/	Internet Explorer Cache files
Users/DefApps/APPDATA/INERNETEXPLORER/INetCookies/	Internet Explorer Cookies files
SharedData/Input/neutral/	ihds.dat - user input word list

LINKS

- **Windows Phone 8 Forensic Artifacts**
  http://www.sans.org/reading-room/whitepapers/forensics/windows-phone-8-forensic-artifacts_35787
- **Windows Phone 8 Forensic Artifacts and Case Study**
  https://files.sans.org/summit/Digital_Forensics_and_Incident_Response_Summit_2015/PDFs/WindowsPhone8ForensicArtifactsandCaseStudyCindyMurphy.pdf
- **Monkeying around with Windows Phone 8.0**
  http://cheeky4n6monkey.blogspot.it/2014/06/monkeying-around-with-windows-phone-80.html
- **Windows Phone 8.0 SMS, Call History and Contacts Scripts**
  http://cheeky4n6monkey.blogspot.it/2014/10/windows-phone-80-sms-call-history-and.html
- **"Awesome" Windows Phone 8 Stuff**
  http://cheeky4n6monkey.blogspot.it/2014/10/awesome-windows-phone-8-stuff.html
- **Chunky4n6Monkey!**
  http://cheeky4n6monkey.blogspot.it/2015/07/chunky4n6monkey.html
- **Automating Window Phone 8 Analysis**
  http://encase-forensic-blog.guidancesoftware.com/2014/10/encase-and-python-automating-windows.html
- **Has the smartphone finally outsmarted us?**
  http://digital-forensics.sans.org/blog/2015/02/25/has-the-smartphone-finally-outsmarted-us
- **Cheeky4n6monkey Git Hub repository**
  https://github.com/cheeky4n6monkey/4n6-scripts
- **Belkasoft – Analyzing Windows Phone 8.1 JTAG and UFED Dumps**
  https://belkasoft.com/en/jtag-analysis
- **Magnet – Analyzing Windows Phone Artifactw with IEF**
  http://www.magnetforensics.com/mobile-forensics/analyzing-windows-phone-artifacts-with-ief
- **Windows Phone 8 and RegRipper**
  http://windowsir.blogspot.it/2014/09/windows-phone-8-and-regripper.html

ACKNOWLEDGMENTS

- The authors thanks:
- **Pasquale Stirparo (@pstirparo)** for his contribution and suggestions during the research
- **Cindy Murphy (@CindyMurph)** and **Adrian Leong (@Cheeky4n6Monkey)** for the impressive work and research on Windows Phone 8/8.1

Q&A?

Mattia Epifani

- Digital Forensics Analyst and Mobile Device Security Analyst
- CEO @ REALITY NET – System Solutions
- Member of CLUSIT, DFA, IISFA, ONIF, Tech and Law Center
- GREM, GCFA, GNFA, GMOB
- CEH, CHFI, CCE, CIFI, ECCE, AME, ACE, MPSC

Mail	**mattia.epifani@realitynet.it**
Twitter	**@mattiaep**
Linkedin	**http://www.linkedin.com/in/mattiaepifani**
Company	**http://www.realitynet.it**
Company Blog	**http://blog.digital-forensics.it**