Advanced smartphone forensics

Apple iCloud: backups, FindMyPhone, document storage, keychain

Windows Live: SMS, mail, notes

BlackBerry 10 backup encryption

Vladimir Katalov, ElcomSoft Co. Ltd
## Smartphone forensics methods

<table>
<thead>
<tr>
<th></th>
<th>iOS</th>
<th>WP8</th>
<th>BB 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical acquisition</td>
<td>Yes(*)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Advanced logical</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Physical acquisition</td>
<td>Yes/No(**)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Chip-off</td>
<td>No</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Local backup</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Cloud backup</td>
<td>Yes</td>
<td>Yes/No(***)</td>
<td>No</td>
</tr>
<tr>
<td>iCloud keychain</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Documents in cloud</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Location service</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

(*) In fact, same as local backup  
(**) For A5+ devices, subject of jailbreak availability  
(***) No complete backup, only selected categories
iOS forensics

- **Logical acquisition**
  - “Ask” device to produce backup
  - Device must be unlocked (*)
  - Device may produce encrypted backup
  - Limited amount of information (but more than you think)

- **Physical acquisition**
  - Boot-time exploit to run unsigned code or jailbreak
  - Device lock state isn’t relevant, can bruteforce passcode
  - Can get all information from the device (incl. deleted data)

- **Advanced logical acquisition**
  - By direct access to some services running on the iPhone
  - Device must be unlocked*
  - Limited amount of information (some as in local backup, but plus something extra)
  - Backup password isn’t relevant
  - Can be performed over Wi-Fi

- **iCloud**
  - Need Apple ID and password
  - Can be performed without the device itself
  - Almost the same information as in local backup
  - Can get the documents and location data, too

*But there is a workaround ;)*
Backups – what and when

• Contacts and Contact Favorites
• Messages (including iMessages)
• Call history
• Application data
• Device settings
• Camera roll (photos and videos)
• Purchases (music, movies, TV, apps, books)
• Mail accounts
• Network settings (saved Wi-Fi hotspots, VPN settings etc)
• Paired Bluetooth devices
• Safari bookmarks, cookies, history
• ... and much more

★ Local backups
• iTunes create backups when:
  • Sync with iTunes
  • [File] | [Devices backup]

★ iCloud backups
• Backup runs daily when device is:
  • Connected to the Internet over Wi-Fi
  • Connected to a power source
  • Locked
• Can force backup
  • [Settings] | [iCloud] | [Storage & Backup] | [Back Up Now]
Wait, there is more…

• **Google Apps data: Search, Maps, YouTube etc**

  AppDomain-com.google.*

• **Social networking & communications**

  AppDomain-net.whatsapp.WhatsApp\*
  AppDomain-com.burbn.instagram\*
  AppDomain-com.facebook.Facebook\*
  AppDomain-com.facebook.Messenger\*
  AppDomain-com.skype.skype\*
  AppDomain-com.atebits.Tweetie2\*
  AppDomain-com.linkedin.LinkedIn\*
  AppDomain-com.naveenium.foursquare\*
  AppDomain-com.viber\*

• **Other**

  HomeDomain\Library\Keyboard\*
  HomeDomain\Library\Passes\*
  HomeDomain\Library\Voicemail\*
  HomeDomain\Library\Maps\*

• Message attachments (even from deleted conversations!)
• Pictures from twitter posts
• Last backup date & time
• Info on Wi-Fi access points you ever connected to (SSID, security, signal etc)
• … a lot of other interesting stuff :)
Frequent locations

Home
46 visits recorded since 22 декабря 2013 г.

Звездный бульвар 21
22 visits recorded since 24 декабря 2013 г.

улица Генерала Антонова 10
5 visits recorded since 24 декабря 2013 г.

улица Сущевский Вал 46с1
4 visits recorded since 29 января 2014 г.

Каширское шоссе 5
4 visits recorded since 29 декабря 2013 г.

If you choose to enable Improve Maps, Apple will collect the GPS coordinates obtained through the Frequent Locations feature on your device and correlate them with the street address associated with your Apple ID. This will enable Apple to better approximate the geographic location of that and other addresses. Apple will only retain the resulting coordinates in an anonymous form to improve Maps and other Apple location-based products and services. You can turn off Improve Maps or Frequent Locations at any time under Settings > Privacy > Location Services > System Services > Frequent Locations.
### Location services

A screenshot of a software interface showing a database table with columns such as `MCC`, `MNC`, `LAC`, `CI`, `U.P.`, `Timestamp`, `Latitude`, `Longitude`, and `HorizontalAccuracy`. The table contains several rows with data for each column.

### Database Table Snippet

<table>
<thead>
<tr>
<th>RecNo</th>
<th>MCC</th>
<th>MNC</th>
<th>LAC</th>
<th>CI</th>
<th>U.P.</th>
<th>Timestamp</th>
<th>Latitude</th>
<th>Longitude</th>
<th>HorizontalAccuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>250</td>
<td>2</td>
<td>7753</td>
<td>1</td>
<td>1</td>
<td>411552390.649334</td>
<td>55.8394231</td>
<td>37.5346787</td>
<td>144</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>2</td>
<td>7701</td>
<td>1</td>
<td>1</td>
<td>4115569213.881531</td>
<td>55.78641917</td>
<td>37.40668065</td>
<td>145</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>2</td>
<td>7701</td>
<td>1</td>
<td>1</td>
<td>4115569213.881531</td>
<td>55.78641917</td>
<td>37.40668065</td>
<td>145</td>
</tr>
<tr>
<td>4</td>
<td>250</td>
<td>2</td>
<td>7701</td>
<td>1</td>
<td>1</td>
<td>4115569213.881531</td>
<td>55.78641917</td>
<td>37.40668065</td>
<td>145</td>
</tr>
<tr>
<td>5</td>
<td>250</td>
<td>2</td>
<td>7701</td>
<td>1</td>
<td>1</td>
<td>4115569213.881531</td>
<td>55.78641917</td>
<td>37.40668065</td>
<td>145</td>
</tr>
<tr>
<td>6</td>
<td>250</td>
<td>2</td>
<td>7701</td>
<td>1</td>
<td>1</td>
<td>4115569213.881531</td>
<td>55.78641917</td>
<td>37.40668065</td>
<td>145</td>
</tr>
<tr>
<td>7</td>
<td>250</td>
<td>2</td>
<td>7701</td>
<td>1</td>
<td>1</td>
<td>4115569213.881531</td>
<td>55.78641917</td>
<td>37.40668065</td>
<td>145</td>
</tr>
<tr>
<td>8</td>
<td>250</td>
<td>2</td>
<td>7701</td>
<td>1</td>
<td>1</td>
<td>4115569213.881531</td>
<td>55.78641917</td>
<td>37.40668065</td>
<td>145</td>
</tr>
<tr>
<td>9</td>
<td>250</td>
<td>2</td>
<td>7701</td>
<td>1</td>
<td>1</td>
<td>4115569213.881531</td>
<td>55.78641917</td>
<td>37.40668065</td>
<td>145</td>
</tr>
</tbody>
</table>

Record 18 of at least 512
Do you really trust your charger?
Pair-locking

- iOS device: `/var/root/Library/Lockdown`
- Mac: `/var/db/lockdown`

**lockdownd service**

- backup service
- software installation service
- get device name & UDID
- sync data
- retrieve a screenshot
- request iOS diagnostic information
- put device into recovery mode
- manage provisioning profiles
Advanced logical acquisition

- for jailbroken devices - the entire file system
- device information: name, model, IMEI, UUID, serial number etc
- all the media (photos, videos, iTunes library, iBooks)
- application data (including temporary files and caches folder)
- various device settings
- log files and diagnostic information
- cached web data (e.g. pictures from social networks)
- keyboard typing caches
- SMS and iMessages (including attachments, even to deleted messages)
- address book
- calendar
- voice mail
- WAL (Write-Ahead Logging) files for most SQLite databases
- ...and more

Works even if device is passcode-locked and backup encryption is set

Can be done over Wi-Fi

Only need the pairing record
iCloud Control Panel
iCloud backups reverse engineering

- no backup to iCloud from iTunes :(

so...

- jailbreak iPhone
- Install Open SSH, get keychain (keychain-2.db)
- [Settings] | [iCloud] | [Delete Account] | [Delete from My iPhone]
- [Settings] | [General] | [Reset] | [Reset All Settings]
- reboot
- set up Wi-Fi connection (proxy)
- replace keychain with our own trusted root certificate
- ... read all the traffic :)
Files in iCloud

- Container 1
  - Chunk 1
  - Chunk 2
  - Chunk 3

- Container 2
  - Chunk 1
  - Chunk 2
  - Chunk 3

- Container 3
  - Chunk 1
  - Chunk 2
  - Chunk 3

- File
  - Chunk 1
  - Chunk 2
  - Chunk 3
  - Chunk 4
  - Chunk 5
  - Chunk 6
  - Chunk 7
  - Chunk 8
  - Chunk 9
iCloud backups – authentication, get token, get keys

Authentication:
https://setup.icloud.com/setup/authenticate/$APPLE_ID$
Authorization:Basic <authentication data>
authentication data = mime64 (AppleID:password)
returns: mmeAuthToken, dsPrsID

https://setup.icloud.com/setup/get_account_settings
Authorization:Basic <authentication data>
authentication data = mime64 (dsPrsID:mmeAuthToken)
returns: mmeAuthToken (new/other one!!)

https://p11-mobilebackup.icloud.com/mbs/(dsPrsID)
Authorization: <authentication data>
authentication data = mime64 (dsPrsID:mmeAuthToken)
returns: list of backup IDs (backupudid)

https://p11-mobilebackup.icloud.com/mbs/2005111682/(backupudid)/getKeys
Get snapshots, file auth. tokens, chunks URLs

Enumerate snapshots

HTTPS GET
https://p11-mobilebackup.icloud.com/mbs/(dsPrsID)/(backupudid)/(snapshotid)/listFiles?[…]

Get file authentication tokens

HTTPS POST
https://p11-mobilebackup.icloud.com/mbs/(dsPrsID)/(backupudid)/(snapshotid)/getFiles

Get URLs for file chunks

HTTPS POST
https://p11-content.icloud.com/(dsPrsID)/authorizeGet

Download chunks

Windows Azure:
http://msbnx000004.blob.core.windows.net:80/cnt/g6YMJKQBPxQruxQAr30C?[…]

Amazon AWS:
http://us-std-00001.s3-external-1.amazonaws.com/I9rh20QBPX4jizMAr3vY?[…]
iCloud backups – data encryption

- get keyData from iCloud
- wrappedOffset = keyDataSize - (ECP_LEN + WRAPPED_KEY_LEN)
- get wrappedKey (at wrappedOffset)
- get CLASS_KEY
- iOS 5/6: ((UINT32*) (keyData + wrappedOffset))[-1]
- iOS 7: ((UINT32*) (keyData + wrappedOffset))[-3]
- decrypt wrappedKey using CLASS_KEY
- get AES_KEY from wrappedKey
- file decryption: by 0x1000 blocks (unique IV for every block)

#define HFS_IV_GENERATOR 0x80000061
#define IV_GEN(x) (((x) >> 1) ^ (((x) & 1) ?
HFS_IV_GENERATOR : 0))

static UInt8 *genIV (UInt32 seed, void *pIV) {
  UInt32 *pdw = (UInt32*)pIV;
  pdw[0] = seed = IV_GEN(seed);
  pdw[1] = seed = IV_GEN(seed);
  pdw[2] = seed = IV_GEN(seed);
  pdw[3] = seed = IV_GEN(seed);
  return (UInt8*)pIV;
}

AES_KEY aesIV;
makeIVkey (&aesIV, abKey, SYSTEM_KEY_LEN);
(abKey is AES_KEY we have got from wrappedKey)

static AES_KEY *makeIVkey (AES_KEY *pAES, UInt8 *pb,
size_t cb) {
  SHA_CTX sha;
  UInt8 abHash[SHA_DIGEST_LENGTH];
  SHA1_Init (&sha);
  SHA1_Update (&sha, pb, cb);
  SHA1_Final (abHash, &sha);
  AES_set_encrypt_key (abHash, 128, pAES);
  return pAES;
}
iCloud backups – encryption, summary

**iCloud encryption**

- Data stored at 3rd-party storage providers is encrypted
- Apple has encryption keys to that data
- Few files are further encrypted using keys from OTA backup keybag
- Keychain items are encrypted using keys from OTA backup keybag
- Need key 0x835 (securityd) to decrypt most keys from OTA backup keybag

**iCloud backups – summary**

- There is no user-configurable encryption for iCloud backups
- iCloud backups are stored in Microsoft and Amazon clouds in encrypted form
- Apple holds encryption keys and thus have access to data in iCloud backups
- If Apple stores 0x835 keys then it can also have access to Keychain data
- Apple may have legal obligations to do this (e.g. LE)

- **No notification after backup downloading (as with device restore)!**
Find My Phone
Find My Phone protocol

Authentication

validate
https://setup.icloud.com/setup/ws/1/validate (ClientBuildNumber, ClientId)
ClientBuildNumber=1M.63768 (constant), ClientId is random GUID
<- instance

login
https://setup.icloud.com/setup/ws/1/login (id, password)
id=sha1(apple_id+instance)
<- dsid

Get devices with location

initClient
https://p11-fmipweb.icloud.com/fmipservice/client/web/initClient

refreshClient
https://p11-fmipweb.icloud.com/fmipservice/client/web/refreshClient (id, dsid)
<- content (location)
iCloud documents
Get files from iCloud

To get list of files

- Authentication request (with given AppleID & password). Client gets mmeAuthToken in return; which, in order, is used to create authentication token (together with dsid). dsid (Destination Signaling IDentifier) is an unique ID assigned to the user when registering at iCloud.com.

- Request to get AccountSettings. Client gets an URL (ubiquityUrl) with an address to get UUID (unique user identifier), file list, info on file tokens and for authorization.

- Request to get file list (POST). Output (for every file):
  - file name
  - file id
  - parent folder id
  - last change time
  - checksum
  - access rights

To download given file

- Request to get file token (using file id, checksum and aliasMap).

- Authorization request. Returns information on file chunks and containers. Output: container list (with URLs) and chunk information.
iCloud files: packages

- KeyNote: PDF, Microsoft PowerPoint, KeyNote
- Pages: PDF, Microsoft Word, Pages
- Numbers: PDF, Microsoft Excel, Numbers
- Some other programs: 1Password, WhatsApp etc

Storage: plist + content (text, media files)

Validate, login
https://setup.icloud.com/setup/ws/1/validate
https://setup.icloud.com/setup/ws/1/login

Export

Check export status

Download converted file

iWork'2013 files w/password: only available as packages
.iwph: password hint, .iwph2: password hash (pbkdf2(sha1, 100000))
iCloud keychain
To set up iCloud Keychain on this Mac, request approval from one of your other devices using iCloud Keychain. If no other devices are available, use your iCloud Security Code.
iCloud keychain (cont’d)

<table>
<thead>
<tr>
<th>Name</th>
<th>Kind</th>
<th>Date Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>accounts.google.com (Passwords not saved)</td>
<td>Web form password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>AirPort</td>
<td>application password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>aknet</td>
<td>AirPort network pass...</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>appleid.apple.com (<a href="mailto:apple@elcomsoft.com">apple@elcomsoft.com</a>)</td>
<td>Web form password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>appleid.apple.com (<a href="mailto:qtt.test@gmail.com">qtt.test@gmail.com</a>)</td>
<td>Web form password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>appleid.apple.com (<a href="mailto:qtt.test@icloud.com">qtt.test@icloud.com</a>)</td>
<td>Web form password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>appleid.apple.com (<a href="mailto:vkatalov@mail.ru">vkatalov@mail.ru</a>)</td>
<td>Web form password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>calendar.google.com</td>
<td>Internet password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>calendar.google.com</td>
<td>Internet password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>daw.apple.com (elcomsoft)</td>
<td>Web form password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>daw.apple.com (<a href="mailto:info@elcomsoft.com">info@elcomsoft.com</a>)</td>
<td>Web form password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>daw.apple.com (<a href="mailto:vkatalov@gmail.com">vkatalov@gmail.com</a>)</td>
<td>Web form password</td>
<td>11 Jun 2013</td>
</tr>
<tr>
<td>daw.apple.com (<a href="mailto:vkatalov@mail.ru">vkatalov@mail.ru</a>)</td>
<td>Web form password</td>
<td>11 Jun 2013</td>
</tr>
</tbody>
</table>

nullcon International Security Conference Goa 2014
iCloud keychain (cont’d)
iCloud keychain (cont’d)

To use iCloud Keychain on this iPhone, approve it from another device that uses iCloud Keychain or use your iCloud Security Code.

**Approve with Security Code**

Keep your passwords and contact information up to date across all the devices that you approve. Your information is protected and cannot be read by Apple.

---

⚠️ Allow “Элкомсофт iPhone 5 (JB)” to use iCloud Keychain?

Enter the Apple ID password for “apple@elcomsoft.com” to allow Элкомсофт iPhone 5 (JB) to use iCloud Keychain.
iCloud keychain components

- **AppleKeyStore.kext**
  - Get keychain records
  - Wrap/unwrap KeychainRecord keys

- **secd**
  - Restore keychain

- **com.apple.sbd**
  - Connect to iCloud keychain
  - Take KeyBagKeys

- **com.apple.lakitu**
  - Sync keychain & KeyBag
  - Launch

- **com.apple.preferences.icloud.remoteservice**
  - Save keychain & KeyBag to cache

- **iCloud panel**
  - Sync defaultsd

- **KeychainAccess**

- **com.apple.sbd.plist**

- **Keychain-2.db**
iCloud keychain components

**com.apple.preferences.icloud.remoteservice**
process that interacts with iCloud control

**com.apple.sbd**
(daemon) caching and restoring keychain, get notifications

**com.apple.lakitu**
(daemon) talks to iCloud (get requests from com.apple.sbd, make queries to iCloud, get and decrypt responses, passes them back to com.apple.sbd)

**secd**
(daemon) caching and restoring keychain, get notifications

**AppleKeyStore.kext**
driver to interact with KVS (key-value storage)

**Security.framework**
functions to restore keychain, save it to SQLite database, send notifications (e.g. to Keychain access)

**libcorecrypto.dylib**
encryption/decryption
iCloud keychain retrieval

GetAccountSettings

POST https://setup.icloud.com/setup/get_account_settings HTTP/1.1
Host: setup.icloud.com
Authorization: Basic MTc3MzgyNT … INLy9TbkU9

Authorization = Base64("apple_id" + ":" + "password")
Get mmeAuthToken in return
X-MobileMe-AuthToken = Base64("dsid" + ":" + "mmeAuthToken")
Sync (get Keychain and KeyBag) - query

POST https://p18-keyvalueservice.icloud.com/sync HTTP/1.1
Host: p18-keyvalueservice.icloud.com
Authorization: X-MobileMe-AuthToken MTc3Mzg … meWRINDg9

[...] 
<dict>
  <key>apns-token</key>
  <data>
  D7wxUEz2av7JaSgJD6j2IyQKENzH0e4DGJzfOeLBbYA=
  </data>
  <key>apps</key>
  <array>
    <dict>
      <key>bundle-id</key>
      <string>com.apple.security.cloudkeychainproxy3</string>
      <key>kvstore-id</key>
      <string>com.apple.security.cloudkeychainproxy3</string>
      <key>registry-version</key>
      <string>FT=-@RU=40c72786-6f77-4190-85d8-3ae1f4df91ca@S=1286</string>
    </dict>
    <dict>
      <key>bundle-id</key>
      <string>com.apple.sbd</string>
      <key>kvstore-id</key>
      <string>com.apple.sbd3</string>
      <key>registry-version</key>
      <string>FT=-@RU=40c72786-6f77-4190-85d8-3ae1f4df91ca@S=1259</string>
    </dict>
  </array>
  <key>service-id</key>
  <string>iOS</string>
</dict>
Sync (get Keychain and KeyBag) - response

<dict>
  <key>apps</key>
  <array>
    <dict>
      <key>kvstore-id</key>
      <string>com.apple.security.cloudkeychainproxy3</string>
      <key>keys</key>
      <array>
        <dict>
          <key>data</key>
          <data>AYYkF93rOBg ... AABVag==</data>
          <key>name</key>
          <string>com.apple.securebackup.record</string>
        </dict>
      </array>
      <key>bundle-id</key>
      <string>com.apple.sbd</string>
    </dict>
  </array>
  <key>timestamp</key>
  <integer>1384690786479</integer>
</dict>
SMS challenge?

POST https://p18-escrowproxy.icloud.com:443/escrowproxy/api/get_sms_targets HTTP/1.1
Authorization: X-MobileMe-AuthToken MTc3Mzgy...WkdXM9

...<dict><key>command</key><string>GET_SMS_TARGETS</string><key>version</key><integer>1</integer></dict><plist>

<dict><key>success</key><true/><key>enrollRequiresPhoneNumber</key><true/><key>isHSAEnabled</key><true/></dict>

...<dict>
SMS challenge (cont’d)

POST https://p15-escrowproxy.icloud.com:443/escrowproxy/api/generate_sms_challenge HTTP/1.1
Authorization: X-MobileMe-AuthToken MTY5NT…T0INS3FnQnM9

…
<dict>
  <key>command</key>
  <string>GENERATE_SMS_CHALLENGE</string>
  <key>label</key>
  <string>com.apple.securebackup.record</string>
  <key>version</key>
  <integer>1</integer>
</dict>

<dict>
  <key>version</key>
  <integer>1</integer>
  <key>success</key>
  <true/>
  <key>smsChallengeLength</key>
  <integer>6</integer>
  <key>message</key>
  <string>Success</string>
</dict>
Get KeyBagKey – srp_init

POST https://p18-escrowproxy.icloud.com:443/escrowproxy/api/srp_init HTTP/1.1
Host: p18-escrowproxy.icloud.com:443

<dict>
    <key>blob</key>
    <string>dSyhi0M/…CQ==</string>
    <key>command</key>
    <string>SRP_INIT</string>
    <key>label</key>
    <string>com.apple.securebackup.record</string>
    <key>phoneNumberToken</key>
    <string>AQAAAABSidUhUkYydkSNDx8dc4r/QMudn0Q1ctg=</string>
    <key>version</key>
    <integer>1</integer>
    <key>smsChallengeCode</key>
    <string>код</string>
</dict>

HTTP/1.1 200 OK
[...]
<plist version="1.0">
    <dict>
        <key>respBlob</key>
        <string>AAABiAAA…638rrzw8=</string>
        <key>dsid</key>
        <string>1773825601</string>
    </dict>
</plist>
Keychain recovery (get KeyBagKey)

POST https://p18-escrowproxy.icloud.com:443/escrowproxy/api/recover HTTP/1.1
Host: p18-escrowproxy.icloud.com:443

[...]
<dict>
    <key>blob</key>
    <string>AAAAYAAA ... +m8</string>
    <key>command</key>
    <string>RECOVER</string>
    <key>version</key>
    <integer>1</integer>
</dict>

HTTP/1.1 200 OK
[...]
<dict>
    <key>respBlob</key>
    <string>AAADKAA...1FHuEwbQ==</string>
</dict>

No prompt whether to allow the new device to use the keychain!!
Keychain decryption

- `genp` class: Wi-Fi passwords, credit card data
- `inet` class: saved Safari passwords and autocomplete fields

- Version (4 bytes)
- KeyClass (4 bytes): index in CLASS_KEY array
- KeyDataSize (4 bytes)
- KeyData (0x48 bytes)
  - RecordPublicKey (0x20 bytes)
  - WrappedKey (0x28 bytes)
- CryptedData

- SharedKey = curve25519(CLASS_KEY.privateKey, RecordPublicKey)
  
  \[
  md = \text{sha256}("x00\x00\x00\x01" + \text{SharedKey} + \text{RecordPublicKey} + \text{CLASS_KEY.publicKey})
  \]

  AES_unwrap_key(md)

- `gcm_init_key`
- `gcm_decrypt_message`

### Table

| Offset   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00000000 | 03 | 00 | 00 | 00 | 06 | 00 | 00 | 00 | 48 | 00 | 00 | 00 | 8C | B0 | 01 | CE |
| 00000010 | F5 | 80 | CD | 22 | 77 | 5F | 83 | CD | C7 | 9B | 4A | 91 | 0E | 47 | B2 | 3E |
| 00000020 | DE | A4 | 2E | 1D | 26 | A2 | 17 | 25 | B2 | CF | A1 | 20 | 1A | CA | A9 | 3A |
| 00000030 | BB | 4B | 00 | F4 | 7E | 0E | 7D | A4 | 7E | 5F | 07 | 4D | C2 | 4A | D9 | 31 |
| 00000040 | 3D | D1 | 2E | D6 | DC | C0 | 31 | 62 | 4C | 56 | 99 | 5B | 43 | SC | F6 | B7 |
| 00000050 | 61 | 8E | 6E | A6 | 79 | B2 | BF | D9 | 68 | 74 | FA | 18 | 31 | 89 | 26 | 49 |
iCloud keychain – no brute-force available
Apple iCloud: conclusion

- Balance between security, privacy and convenience
- iCloud security risks
- Use additional encryption
- Better 2FA implementation
- Need further work
  - My Photo Stream
  - Photo Sharing
  - 3rd party apps data
  - Back To My Mac
  - Touch ID (iPhone 5S)
Windows Phone backups

What is saved:

• Internet Explorer Favourites
• List of installed apps
• Theme and accent configuration
• Call history
• App settings (where applicable - email and accounts, lock screen etc)
• Test messages (SMS conversations)
• Photos (good quality - uses data allowance)

Can get with LiveSDK:

• Basic user information
• Contacts
• Calendars
• Files, photos, videos, documents

Download full backup?
Windows Phone: Live! SDK

Identity API
• Get basic information on user

Hotmail API
• Manage contacts
• Manage calendars & events

SkyDrive API
• Files & documents
• Photos
• Videos
Windows Phone: Live! SDK

- Authentication
- Needs client_id of registered application
- Several requests to https://login.live.com to get redirects and some parameters
- Send “SetDeviceInfo” request (same for Contacts/SMS/email/notes)

POST https://blu-m.hotmail.com/[...]
Authorization: Basic dGVzdGluZy53aW5kb3dzMUBtYWlsLnJ1OnF3ZXJ0eTEyMzQ=
<Settings xmlns="Settings:">
  <DeviceInformation>
    <Set>
      <Model>RM-820_nam_canada_246</Model>
      <IMEI>353653050052892</IMEI>
      <FriendlyName>Lumia 920</FriendlyName>
      <OS>Windows Phone 8.0.9903</OS>
      <OSLanguage>English</OSLanguage>
      <PhoneNumber>+***********</PhoneNumber>
      <UserAgent>MSFT-WP/8.0.9903</UserAgent>
      <EnableOutboundSMS>0</EnableOutboundSMS>
      <MobileOperator>Developer Device</MobileOperator>
    </Set>
  </DeviceInformation>
  <UserInformation>
    <Get/>
  </UserInformation>
</Settings>
Authorization: Base64( “login” + “:” + “password” )

- Request folder hierarchy
- Request sync key for folders
- GetItemEstimate to get item count
Windows Phone: folder hierarchy

<table>
<thead>
<tr>
<th>Value</th>
<th>Folder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User-created (generic)</td>
</tr>
<tr>
<td>2</td>
<td>Default Inbox</td>
</tr>
<tr>
<td>3</td>
<td>Default Drafts</td>
</tr>
<tr>
<td>4</td>
<td>Default Deleted Items</td>
</tr>
<tr>
<td>5</td>
<td>Default Sent Items</td>
</tr>
<tr>
<td>6</td>
<td>Default Outbox</td>
</tr>
<tr>
<td>7</td>
<td>Default Tasks</td>
</tr>
<tr>
<td>8</td>
<td>Default Calendar</td>
</tr>
<tr>
<td>9</td>
<td>Default Contacts</td>
</tr>
<tr>
<td>10</td>
<td>Default Notes</td>
</tr>
<tr>
<td>12..17</td>
<td>User-defined Mail, Calendar, Contacts etc.</td>
</tr>
</tbody>
</table>
WP8: get SMS

Server: https://???-m.hotmail.com
(to get correct name: send request to blum-hotmail.com)
Protocol: ASHTTP
Data format: wbxml
Can be compressed ("Accept-Encoding: gzip, deflate")

Get (login, password in base64
✓ success

FolderHierarchy
✓ success

SyncKeys
✓ success

CategoriesItems (CollectionId)
✓ return: SyncKey

GetItemEstimate (SyncKey, CollectionId)
✓ number of SMS

GetSMS (SyncKey, CollectionId)
✓ new SyncKey and SMS

GetClosedSMS
✓ new SyncKey
WP8: get mail

GET http://mail.live.com/ HTTP/1.1
Host: mail.live.com
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.116 Safari/537.36
Accept-Encoding: gzip, deflate, sdch

< redirect to authentication (https://login.live.com/login.srf?...)

> POST https://login.live.com/ppsecure/post.srf?... (login, password)

< redirect to mail.live.com?id=XXX

> GET https://mail.live.com/?id=XXX

< redirect to mailbox:

HTTP/1.1 302 Found
Location: https://col131.mail.live.com/default.aspx?id=XXX&rru=inbox

Go to mailbox:

GET https://col131.mail.live.com/default.aspx?id=XXX&rru=inbox HTTP/1.1
Host: col131.mail.live.com
Connection: keep-alive
Windows Phone: find my phone

- **Map phone's location**
  
  GET https://www.windowsphone.com/ru-ru/my/find HTTP/1.1
  or

- **Make the phone ring**

  https://www.windowsphone.com/ru-ru/my/phones/c34a5c89b6aabc87cdc457b49e5f3abbf81c72e0b19d48bdbd3918e36785f646/ring

- **Lock the phone and show a message**

- **Erase the phone**

  *Authentication is required, of course!*

  1. **If device is not found, service sends an email (with the link to windowsphone.com) ones it becomes available**
  2. **To get device location, we have to select it (so it becomes ‘active’), and the state is saved on the site**
  3. **The number of device location requests is limited (per day)**
BlackBerry backups

Old format:
- IPD files (all databases in a single container)
- BBB files (in fact, ZIP archives with several IPDs, one database per IPD)

New format:
- Unencrypted BBB-QNX (three .tar files inside); for PlayBook with firmware <2.0
- Encrypted BBB-QNX (all .tar files are encrypted); for BB OS 10 (created w/ BlackBerry Link)

For old formats - simple password protection:
- Encryption: AES-256
- Password verification:
  BlackBerry Desktop Software 5: pbkdf2 (1) - yes, just one iteration
  BlackBerry Desktop Software 6: pbkdf2 (20,000)
BB 10 backups

- mounting QNX6 partitions
- backup encryption: AES-256
- authentication/verification: HMAC-SHA1
- backup.cgi:backuparch
- backup.cgi:scramble

`bbid (BlackBerry ID)`
(libbbid.so:bbid_profile_get_user_properties(urn:bbid:username)

`qbek`
(libbbid.so:bbid_profile_get_user_properties(urn:bbid:backupandrestorekey)

`cache storage: /accounts/<id>/sys/bbid/keyCache`

if not found: request to BB Olympia Service (blackberryid.blackberry.com)
BlackBerry token service

- request: bbid, password, pin, salt (client’s entropy)
- response:

Hct=1379081439336&st=1379168703336&se=PF3V5ikbH8fx2wSb2mbHITGy0q1xlCgZZ66Oma3o66k&et=1381673439336&fn=John&ln=Doe&nn=johndoe-59094&un=john.doe%40gmail.com&ec=AcDGzWbVM12nd0BigqfJYw%3D&em=john.doe

- se - server entropy
- at - authentication token
- ec - user ID for BB cloud services (saved to /dev/rpmb/BBID_BDEK)
- at (creation time), st (server time), et (expiry time)
- further requests: RST (Request Secure Token) with token type and service name

To get qbek:

- get authentication token
- get BBIDAuthToken_1 token for urn:bbid:v1:olympia)
- send request for authzo:qbek token
- register device on BB server (using authzo:qbek token)
- get request on backupAndRestoreKey info (two IDs)
- get janusUrl by request to kronos.bbprotect.blackberry.com
- get qbek from %janusUrl%/FlashGetFile

- Valid device PIN should be supplied
- qbek depends on BlackBerry ID only
Authentication(deviceSerialNumber, usr, pwd)

QbekRequest(AES(request, tokenSecretKey), bbid)

WritePreamble(tarType, iterCount, salt)

WriteBlocks(SHA(block, KDF(qbek, salt, iterCount))
Special thanks to our team:

<table>
<thead>
<tr>
<th>Researches</th>
<th>Testers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anton K.</td>
<td>Artem G.</td>
</tr>
<tr>
<td>Sergey P.</td>
<td>Oxana B.</td>
</tr>
<tr>
<td>Denis B.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Developers</th>
<th>Managers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alina P.</td>
<td>Anna S.</td>
</tr>
<tr>
<td>Oleg K.</td>
<td>Vladimir S.</td>
</tr>
</tbody>
</table>

I could not have done it without you! ;}
Thank you!

Vladimir Katalov, ElcomSoft Co. Ltd.
(twitter: @vkatalov)

http://www.elcomsoft.com
http://blog.crackpassword.com
Facebook: ElcomSoft
Twitter: @elcomsoft