iOS Forensics: Overcoming iPhone Data Protection

Andrey Belenko
Chief Security Researcher
Elcomsoft Co. Ltd.
Agenda

• iOS Forensics 101
• iOS Data Protection
• iOS Forensics
  • Passcode
  • Keychain
• Storage
Forensics 101

Acquisition ➔ Analysis ➔ Reporting

GOALS:

1. Assuming physical access to the device extract as much information as practical

2. Leave as little traces/artifacts as practical
iOS Forensics 101

- **Passcode**
  - Prevents unauthorized access to the device
  - Bypassing passcode is usually enough

- **Keychain**
  - System-wide storage for sensitive data
  - Encrypted

- **Storage encryption**
  - iPhone 3GS and later can encrypt disk data
iOS Forensics 101

- iOS is modified version of Mac OS X
  - Familiar environment

- iOS enforces additional security
  - Code signing: can’t run unsigned executables
  - Sandboxing: access to system is limited

- Acquisition options:
  - Via exposed interfaces (i.e. Sync, Backup)
  - Via circumventing security and running own code
iOS Forensics 101

- Logical: iOS Backup
  - Ask device to produce a backup
  - Device must be unlocked
  - Device may produce encrypted backup
  - Limited amount of information

- Physical: filesystem acquisition
  - Boot-time exploit to run unsigned code
  - Device lock state isn’t relevant
  - Can get all information from the device
  - Since iOS 4 filesystem is encrypted
Pre-iOS 4 Forensics

- Device passcode can be bypassed
- Storage is effectively not encrypted
  - Device transparently decrypts data
- Keychain data is encrypted
  - One can either decrypt all or nothing. Usually all.

Once you have code execution, the rest is easy
New in iOS 4

• Passcode protection is much more robust
• Storage is encrypted
  • Metadata is not encrypted
  • Contents of (almost) every file is encrypted
• New (and better) Keychain encryption
• New (and better) iTunes backup format

All these are part of iOS Data Protection
AES Keys

• All iOS devices have built-in AES processor with 2 hardcoded keys:
  • GID Key is shared by all devices of the same kind
  • UID Key is unique to each and every device (hardware key)

• More keys are computed during startup:
  • Key 0x835 = AES_encrypt (UID, 0101..01) (device key)
  • Derived keys depend solely on GID or UID and thus are fixed for the particular device
Protection Classes

- Content is grouped into protection classes:
  - Available only when device is unlocked
  - Available after first device unlock (and until off)
  - Always available

- Each protection class assigned a master encryption key
- Master keys are protected by device key and passcode
- Protected master keys form system keybag
  - New keys created during device restore
System Keybag

- Stores protected (encrypted) master keys
- Keybag payload is encrypted before writing to disk
- Stored in /private/var/keybags/systembag.kb
- File has NSProtectionNone protection class
  - Meaning it is encrypted
- 11 protection classes in total
  - All but NSProtectionNone are stored in systembag.kb
  - NSProtectionNone is stored in Effaceable Storage
Effaceable Storage

- Region of flash memory
- Facilitates storage of small amounts of data with ability to quickly erase them
- Items within effaceable storage are called lockers
- As of iOS 4: 960 bytes capacity, 3 lockers:
  - ‘BAG1’ – systembag.kb payload key and IV
  - ‘Dkey’ – NSProtectionNone class master key
  - ‘EMF!’ – Filesystem encryption key
- In iOS 5 ‘EMF!’ locker is replaced with ‘LwVM’, conceptually the same.
Unlocking Keybag

Keybag (locked)

| Protected Key | WRAP = 1 |
| Protected Key | WRAP = 2 |
| Protected Key | WRAP = 3 |
| Protected Key | WRAP = 1 |
| Protected Key | WRAP = 3 |
| ... |

Passcode Key

if (WRAP & 0x2)

UNWRAP

Device Key

if (WRAP & 0x1)

UNWRAP

DECRYPT

Keybag (unlocked)

Key

Key

Key

Key

Key

...
Escrow Keybag

- “Usability feature” to allow iTunes to unlock the device
- Contains same master keys as system keybag
- Stored on the computer side
- Protected by 256 bit random “passcode” stored on the device
- With iOS 4, escrow keybag gives same powers as knowing the passcode
- iOS 5 fixed this issue: device can read escrow keybag only if it has been unlocked
iOS 4/5 Key Hierarchy

- **Must be done on the device**
- **Required to decrypt files/keychain**
- **Sufficient for key reconstruction**

- **Passcode**
  - **Passcode Key**
  - **KDF**
    - **UID Key**
      - **Key 89B**
      - **Key 835**

- **FS Key**
  - **Decrypt**

- **Effaceable Storage**
  - **‘EMF!’ / ‘LwVM’**
  - **‘Dkey’**
  - **‘BAG1’**

- **systembag.kb**
  - **Decrypt**

- **Unlock**
  - **System Keybag (locked)**
    - Class A Key (#1)
    - Class B Key (#2)
    - Class C Key (#3)
    - Class D Key (#4)
    - Class Key #5
    - ...
    - Class Key #11

- **System Keybag (unlocked)**
Pre-iOS 4 Passcode

• Lockscreen (i.e. UI) is the only protection

• Passcode is stored in the keychain
  • Passcode itself, not its hash

• Can be recovered or removed instantly
  • Remove record from the keychain
  • And/or remove setting telling UI to ask for the passcode
iOS 4/5 Passcode

- Passcode is used to compute passcode key
  - Computation tied to hardware key
  - Same passcode will yield different passcode keys on different devices!

- Passcode key is required to unlock all but 3 master keys in system keybag
  - Most files are NSProtectionNone thus don’t need passcode
  - Most keychain items are accessible WhenUnlocked or AfterFirstUnlock thus DO require passcode
iOS 4/5 Passcode

• Passcode-to-Key transformation is slow

• Offline bruteforce currently is not possible
  • Requires extracting hardware key

• On-device bruteforce is slow
  • 2 p/s on iPhone 3G, 7 p/s on iPad

• System keybag contains hint on password complexity
iOS 4/5 Passcode

- 0 – digits only, length = 4 (simple passcode)
iOS 4/5 Passcode

- 0 – digits only, length = 4 (simple passcode)
- 1 – digits only, length \(!= 4\)
iOS 4/5 Passcode

- 0 – digits only, length = 4 (simple passcode)
- 1 – digits only, length != 4
- 2 – contains non-digits, any length
iOS 4/5 Passcode

• 0 – digits only, length = 4 (simple passcode)
• 1 – digits only, length != 4
• 2 – contains non-digits, any length

Can at least identify weak passcodes
Pre-iOS 4 Keychain

- SQLite3 Database, only passwords are encrypted
- All items are encrypted with the device key (0x835) and random IV
- Key is unique for each device and is fixed for lifetime of the device
- Key can be extracted (computed) for offline use
- All past and future keychain items from the device can be decrypted using that key

<table>
<thead>
<tr>
<th>IV</th>
<th>Data</th>
<th>SHA-1 (Data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Encrypted with Key 0x835
iOS 4 Keychain

- SQLite3 Database, only passwords are encrypted
- Available protection classes:
  - kSecAttrAccessibleWhenUnlocked (+ ...ThisDeviceOnly)
  - kSecAttrAccessibleAfterFirstUnlock (+ ...ThisDeviceOnly)
  - kSecAttrAccessibleAlways (+ ...ThisDeviceOnly)
- Random key for each item, AES-CBC
- Item key is protected with corresponding protection class master key

<table>
<thead>
<tr>
<th>Class</th>
<th>Wrapped Item Key</th>
<th>Encrypted Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>
iOS 5 Keychain

Almost the same as iOS 4, but...
• All attributes are encrypted (not only password)
• AES-GCM is used instead of AES-CBC
  • Allows to verify integrity

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Class</td>
<td>Wrapped Key Length</td>
<td>Wrapped Key</td>
</tr>
</tbody>
</table>
Pre-iOS 4 Storage

• No encryption before iPhone 3GS

• Starting with iPhone 3GS:
  • Encryption uses EMF key for everything
  • Provides fast wipe, not confidentiality
  • Transparent to applications
  • Filesystem acquisition is not affected
iOS 4 Storage

- Available protection classes:
  - NSProtectionNone
  - NSProtectionComplete

- If no protection class is specified, EMF key is used
  - Filesystem metadata and unprotected files
  - Transparent encryption and decryption (same as pre-iOS 4)

- If protection class is specified, per-file random key is used
  - Key protected with master key is stored
  - com.apple.system.cprotect extended attribute
iOS 5 Storage

Almost the same as iOS 4, but...

• New protection classes:
  • NSFFileProtectionCompleteUntilFirstUserAuthentication
  • NSFFileProtectionCompleteUnlessOpen

• IV for file encryption is computed differently
iOS 4/5 Forensics

- Acquiring disk image is not enough for iOS 4+
  - Content protection keys must also be extracted from the device during acquisition
  - Effaceable Storage contents are also needed to decrypt dd images.
- Passcode or escrow keybag is needed for a complete set of master keys
- In real world it might be a good idea to extract source data and compute protection keys offline
iOS 4/5 Forensics

Must be done on the device
Required to decrypt files/keychain
Sufficient for offline key reconstruction

Passcode
UID Key
KDF
Passcode Key
Unlock
System Keybag (unlocked)

Effaceable Storage
‘EMF!’ / ‘LwVM’
‘Dkey’
‘BAG I’

systembag.kb
Decrypt

Decrypt

Class A Key (#1)
Class B Key (#2)
Class C Key (#3)
Class D Key (#4)
Class Key #5
…
Class Key #11

System Keybag (locked)
Conclusion

- iPhone physical analysis is possible
- Physical acquisition requires boot-time exploit
- Passcode is *usually* not a problem
- Both proprietary and open-source tools for iOS 4/5 forensics are available
Questions?
iOS Forensics: Overcoming iPhone Data Protection

a.belenko@elcomsoft.com
@andreybelenko
www.elcomsoft.com