Security Analysis and Decryption of FileVault 2
IFIP WG 11.9

Omar Choudary   Felix Gröbert   Joachim Metz

29 January 2013
FileVault 2

FileVault secures the data on your disk by encrypting its contents automatically.

FileVault is turned on for the disk "Macintosh HD".

A recovery key has been set.

The recovery key is a “safety net” which can be used to unlock the disk if you forget your password.

Make a copy and store it in a safe place. If you forget your password and lose the recovery key, all the data on your disk will be lost.

KKCA-WTBB-73OR-WJRZ-Z4XQ-MB4C

Click the lock to make changes.
Goals and Motivations

- **Goals**
  - Reverse engineer and analyse FileVault 2
  - Develop our own tool to read encrypted volumes

- **Motivations**
  - Need to know if secure
  - Use in forensic investigations
  - Cannot trust OS if compromised
  - Interoperability, need to access files remotely
Full Disk Encryption

- Need to encrypt all data
- Encryption requires a key that must be stored or derived somehow
- Practical requirement to encrypt disk sectors independently for fast access
AES-XTS: tweakable encryption

- Uses AES-ECB
- 2 keys
  - volume key (key 1)
  - tweak key (key 2)
- tweak value per sector (modified per AES block)
Tools

- GDB
- IDA Pro
- 3 MacBooks
- The Sleuth Kit
Structure analysis

- Entire disc
- GPT header
- Machintosh HD (CoreStorage)
- Recovery HD (HFS Plus)
- ... others
- CS Header
- Encrypted Volume
- Encrypted Metadata
- Disk Label Metadata
- Encrypted plist
The quest

- What are the key derivation mechanisms?
- How is the data encrypted?
General volume encryption architecture

- **Encrypted volume**
  - encrypted volume metadata
  - encrypted key blob
- **Intermediary key**
- **Key derivation**
- **Full volume master key**
- **Decrypt**
- **Decrypted volume**
EncryptedRoot.plist file

- File encrypted with AES-XTS using key from volume header
- Data for different users (including recovery key)
- Key encryption key (KEK) and volume key encrypted structures
- Unknown algorithms (found by reverse engineering)
PBKDF2

- Output keys of arbitrary lengths from any text
- Slow brute force attacks on passwords by iterating hash
- 3 parameters: iterations, salt, password
- Option of PRF (e.g. HMAC-SHA256)
- Used in FileVault 2 to derive a KEK from user password or recovery key
- salt and iterations* given in EncryptedRoot.plist
AESWrap

- Used to encrypt a key with another key
- Based on AES
- Needs IV, useful to verify correct decryption
- Used in FileVault 2 to decrypt volume KEK and volume key
FileVault 2 key derivation overview

- Header of CoreStorage Volume
- AES-XTS-128
- EncryptedRoot.plist
- Key-Encryption-Keys
- PassedphraseWrappedKEK
- KeyWrappedKEK
- RFC3394 Keyunwrapping
- KEK
- KEKWrappedVolumeKey
- RFC3394 Keyunwrapping
- VolumeKey
- RSA PKCS1
- FileVaultMaster Private Key
- EncryptdRoot.plist
- AES-XTS-128
- EncryptedRoot.plist
- 16 byte salt
- Password entry
Tweak key?

\[ \text{AES-enc} \]

\[ i \]

\[ a^j \]

\[ T \]

\[ P \]

\[ \text{AES-enc} \]

\[ \text{Key}_1 \]

\[ \text{CC} \]

\[ C \]
Searching the tweak key

- Looking at HFS+ metadata (misleading, found/fixed bug)
- Searching metadata
- Chasing encryption via GDB (no luck, found many unknown keys)
- Comparing memory with disk data (found encryption parameters, not our key)
- Finally found using IDA Pro (difficult due to C++)
Computing the tweak key

- Get volume key
- Find metadata blocks and decrypt some blocks which are encrypted
- Obtain logical volume family (lvf) UUID
- Compute the tweak key as follows:

\[
\text{tweak\_key} = \text{trunc}_{128}(\text{SHA256}(\text{volume\_key} \mid \text{lvf\_UUID}))
\]
Volume layout
Random number generator

- Used for derivation of recovery key (and possibly other keys)
- Randomness taken from /dev/random
- Performed detailed analysis and seems OK (see paper)
**libfvde**

- Open source tool to decrypt and mount CoreStorage volumes
- Available at Google code: http://code.google.com/p/libfvde/
- Ongoing investigation for the more general CoreStorage format and how to handle partially encrypted disks
Questions?

Omar Choudary: omar.choudary@cl.cam.ac.uk
Felix Gröbert: felix@groebert.org
Joachim Metz: joachim.metz@gmail.com
How to use the AES block cipher?

- Straight AES-CBC is not suitable
- Random IV in metadata does not allow independent sector encryption
- Constant IV poses problems of watermarking data
- Sector-based IV is better
- Tweakable encryption is the best option now
Other issues with FileVault 2

- Keys can be extracted from memory, so cold boot attacks possible
- Even the good password derivation mechanism does not protect against very bad passwords
How to use libfvde

- Get EncryptedRoot.plist.wipekey file (e.g. via mmls/fls/icat)
- Then run the tool to mount the volume or image:
  
  \[ \text{fvdemount -e EncryptedRoot.plist.wipekey -r 35AJ-AC98-TI1H-N4M3-HDUQ-UQFG /dev/sda2 /mnt/fvdevolume/} \]

- Finally mount the underlying HFS+ file system:
  
  \[ \text{mount -o loop,ro /mnt/fvdevolume/fvde1 /mnt/hfs_file_system} \]