Structural Artifacts of Secure Deletion with Full Disk Encryption

Andrew Brusso

Michigan Technological University

April 18, 2019

Overview

What is Secure Deletion?

- When deleting a file on a typical file system, deletion only changes metadata to make it look like the file is gone.
- Secure deletion uses either encryption or overwriting to more completely delete the data itself, instead of just changing the metadata.

Example of Non-secure Deletion

1. Freshly formatted FAT filesystem
2. 3 files added to filesystem
3. File 2 deleted from drive

```
START_SNAPSHOT
Page 0| ë<MSDOS5.0
Page 1| øÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6|
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14|
Page 15|
Page 16|
Page 17|
Page 18|
Page 19|
```

```
START_SNAPSHOT
Page 0| ë<MSDOS5.0
Page 1| øÿÿÿÿÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6| FILE1       ▯aëNN
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14| file1 contentsfile1
Page 15|
Page 16| file2 contentsfile2
Page 17|
Page 18| file3 contentsfile3
Page 19|
```

```
START_SNAPSHOT
Page 0| ë<MSDOS5.0
Page 1| øÿÿÿÿÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6| FILE1       ▯aëNN
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14| file1 contentsfile1
Page 15|
Page 16| file2 contentsfile2
Page 17|
Page 18| file3 contentsfile3
Page 19|
```

Example of Secure Deletion

1. Freshly formatted FAT filesystem
2. 3 files added to filesystem
3. File 2 *securely* deleted with windows sdelete utility (overwrite with zeroes)

```
START_SNAPSHOT
Page 0| ë<MSDOS5.0
Page 1| øÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6|
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14|
Page 15|
Page 16|
Page 17|
Page 18|
Page 19|
```

```
START_SNAPSHOT
Page 0| ë<MSDOS5.0
Page 1| øÿÿÿÿÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6| FILE1        aëNN
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14| file1 contentsfile1
Page 15|
Page 16| file2 contentsfile2
Page 17|
Page 18| file3 contentsfile3
Page 19|
```

```
START_SNAPSHOT
Page 0| ë<MSDOS5.0
Page 1| øÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6| FILE1        aëNN
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14| file1 contentsfile1
Page 15|
Page 16|
Page 17|
Page 18| file3 contentsfile3
Page 19|
```

Secure Deletion isn't *Truly* Secure

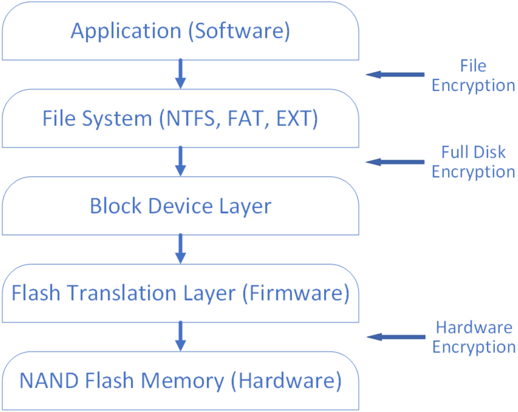
For a deletion scheme to be **Truly Secure**, it needs two properties [1]

1. Data is sanitized so that attacker cannot access it
2. Structural artifacts are removed so that adversary cannot *infer any* sensitive information about the deleted data

Flash memory's log structured writing creates structural artifacts with conventional secure deletion.

What is Full Disk Encryption?

- Full Disk Encryption uses a block cypher (often AES) at the block device layer. Everything is encrypted, including the File System.
- A pre-boot sequence prompts for a password. This "unlocks" the device, and then all data is encrypted at the **sector** level. [2][3][6]



The Problem

Does conventional secure deletion in Full Disk Encryption meet the requirements to be **Truly Secure**?

- A secure deletion will sanitize the data, just like it sanitizes in the case of plaintext.
- Is there any leakage or structural artifacts that appear when sanitizing inside a Full Disk Encryption scheme?

Why does this matter?

- Current secure deletion schemes that address structural artifacts for Flash Memory either require specialized file systems (YAFFS) [4][5], or implementations at the Flash Translation Layer[1], which only works with supported hardware.
- If FDE meets criteria for truly secure deletion, it could offer an alternative approach for users with existing systems to adopt a truly secure deletion scheme at minimal cost.

Adversarial Model

- The attacker doesn't have the encryption key, otherwise they can just decrypt and it is the same as the plaintext case.
- Attacker is able to take **multiple** snapshots of the raw flash. If you have only one snapshot, it is just random data.
- Attacker needs to be able to take a snapshot prior to data being added, after data is added, and after it is deleted, otherwise can't discern data changed after being added.

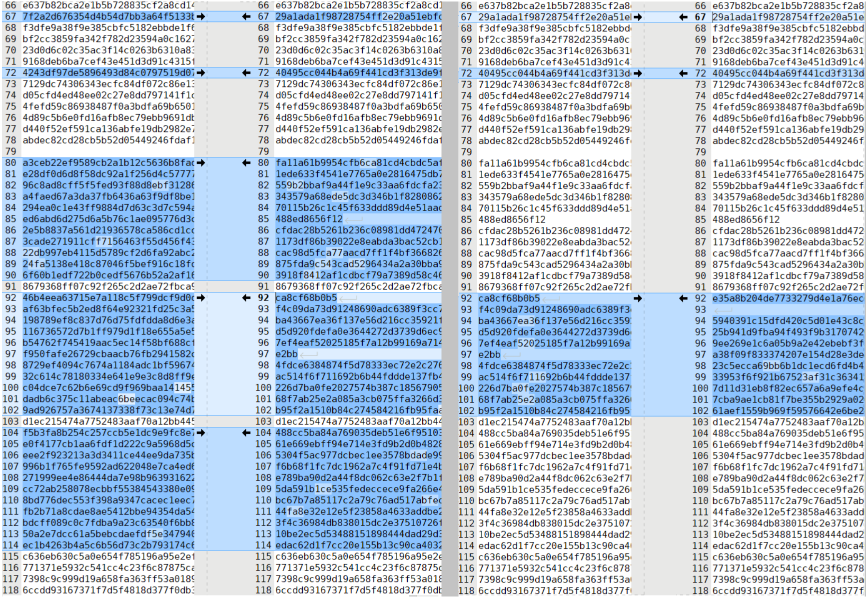
My Approach

1. Created a way to take snapshots of the raw flash on demand
2. Identified three candidate Full Disk Encryption programs: Veracrypt, CipherShield, and Symantec PGP Whole Disk Encryption
3. Encrypted flash device with each FDE
4. Performed a sequence of actions and took snapshots between actions
5. Observed changes between snapshots, and looked for signs of structural artifacts or leakages.

Add 3 files

Secure delete file 2

Before | After | Before | After



Results

- File system state with deletion performed vs. if it had never been performed is different, so scheme is not truly secure.
- Edits can be discerned from deletes if file size > size of a sector, and edits don't span every page.
- Not secure deleting at all might be **better**? Secure deletion causes changes to the data itself between snapshots, so artifacts are immediately apparent
- There are structural artifacts, but are they actually useful? Without the encryption keys you may be able to say a deletion occurred, but you don't have the file context like you do in the plaintext case. Can say some file was deleted, but anything else?

Potential Future Work

- Scripting interactions with dev board for reliability/repeatability. E.g. script generates test files, performs experiment actions, and then automatically takes snapshots.
- Scripted analysis with the snapshots. E.g. script that runs diffs between snapshots and tries to classify changes as add, delete and edit. Analysis of adds/deletes/edits in aggregate might help identify leakages?
- Experiment with different file systems. FAT is very simple, NTFS or EXT potentially introduces new difficulties.
- Windows 10 compatibility, test with Bitlocker, which utilizes TPM and AES CBC.

References

[1] Bo Chen et al. "Sanitizing Data is Not Enough!: Towards Sanitizing Structural Artifacts in Flash Media". In: *Proceedings of the 32Nd Annual Conference on Computer Security Applications*. ACSAC '16. Los Angeles, California, USA: ACM, 2016, pp. 496–507.

[2] Niels Ferguson. *AES-CBC + Elephant diffuser: A disk encryption algorithm for Windows Vista*. 2006.

[3] *How Whole Disk Encryption Works*. Symantec Corporation, Nov. 2010.

[4] Jaeheung Lee et al. "An Efficient Secure Deletion Scheme for Flash File Systems". In: *J. Inf. Sci. Eng.* 26 (2010), pp. 27–38.

[5] Joel Reardon, Srdjan Capkun, and David A. Basin. "Data Node Encrypted File System: Efficient Secure Deletion for Flash Memory". In: *USENIX Security Symposium*. 2012.

[6] *Veracrypt Technical Details*. Veracrypt. URL: https://www.veracrypt.fr/en/Technical%20Details.html.