Structural Artifacts of Secure Deletion with Full Disk Encryption

Andrew Brusso

Michigan Technological University

April 18, 2019

Overview

Background

- Secure Deletion
- Full Disk Encryption

2 The Problem

- Motivation
- Approach



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

- Freshly formatted FAT filesystem
- 2 3 files added to filesystem

File 2 deleted from drive

START_SNAPSHOT	START SNAPSHOT	START_SNAPSHOT	
Page 0 ë <msdos5.0< td=""><td>Page 0 ë<msdos5.0< td=""><td>Page 0 ë<msdos5.0< td=""></msdos5.0<></td></msdos5.0<></td></msdos5.0<>	Page 0 ë <msdos5.0< td=""><td>Page 0 ë<msdos5.0< td=""></msdos5.0<></td></msdos5.0<>	Page 0 ë <msdos5.0< td=""></msdos5.0<>	
Page 1 øÿÿ	Page 1 øÿÿÿÿÿÿ	Page 1 øÿÿÿÿÿÿ	
Page 2	Page 2	Page 2	
Page 3	Page 3	Page 3	
Page 4	Page 4	Page 4	
Page 5	Page 5	Page 5	
Page 6	Page 6 FILE1 DaëNN	Page 6 FILE1 🛛 🛛 🖓 aëNN	
Page 7	Page 7	Page 7	
Page 8	Page 8	Page 8	
Page 9	Page 9	Page 9	
Page 10	Page 10	Page 10	
Page 11	Page 11	Page 11	
Page 12	Page 12	Page 12	
Page 13	Page 13	Page 13	
Page 14	Page 14 file1 contentsfile1	Page 14 file1 contentsfile1	
Page 15	Page 15	Page 15	
Page 16	Page 16 file2 contentsfile2	Page 16 file2 contentsfile2	
Page 17	Page 17	Page 17	
Page 18	Page 18 file3 contentsfile3	Page 18 file3 contentsfile3	
Page 19	Page 19	Page 19	

Example of Secure Deletion

- Freshly formatted FAT filesystem
- 3 files added to filesystem

File 2 securely deleted with windows sdelete utility (overwrite with zeroes)

START_SNAPSHOT	START SNAPSHOT	START SNAPSHOT	
Page 0 ë <msdos5.0< td=""><td>Page 0 ë<msd0s5.0< td=""><td>Page 0 ë<msdos5.0< td=""></msdos5.0<></td></msd0s5.0<></td></msdos5.0<>	Page 0 ë <msd0s5.0< td=""><td>Page 0 ë<msdos5.0< td=""></msdos5.0<></td></msd0s5.0<>	Page 0 ë <msdos5.0< td=""></msdos5.0<>	
Page 1 øÿÿ	Page 1 øÿÿÿÿÿÿ	Page 1 øÿÿÿ	
Page 2	Page 2	Page 2	
Page 3	Page 3	Page 3	
Page 4	Page 4	Page 4	
Page 5	Page 5	Page 5	
Page 6	Page 6 FILE1 DaëNN	Page 6 FILE1 DaëNN	
Page 7	Page 7	Page 7	
Page 8	Page 8	Page 8	
Page 9	Page 9	Page 9	
Page 10	Page 10	Page 10	
Page 11	Page 11	Page 11	
Page 12	Page 12	Page 12	
Page 13	Page 13	Page 13	
Page 14	Page 14 file1 contentsfile1	Page 14 file1 contentsfile1	
Page 15	Page 15	Page 15	
Page 16	Page 16 file2 contentsfile2	Page 16	
Page 17	Page 17	Page 17	
Page 18	Page 18 file3 contentsfile3	Page 18 file3 contentsfile3	
Page 19	Page 10	Page 10	
Andrew Brusso (MTU)	Research Project	April 18, 2019	

5 / 15

Secure Deletion isn't Truly Secure

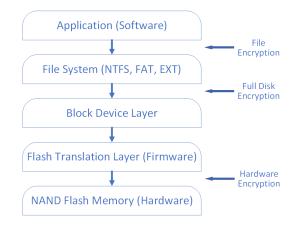
For a deletion scheme to be **Truly Secure**, it needs two properties [1]

- Data is sanitized so that attacker cannot access it
- 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]



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.

- 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

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

Add 3 Before	files After	Secure del Before	ete file 2 After
Delore			
66 e637b82bca2e1b5b728835cf2a8cd14	66 e637b82bca2e1b5b728835cf2a8cd1	66 e637b82bca2e1b5b728835cf2a8	66 e637b82bca2e1b5b728835cf2a8
67 7f2a2d676354d4b54d7bb3a64f5133t → 68 f3dfe9a38f9e385cbfc5182ebbde1ff	 67 29a1ada1f98728754ff2e20a51ebfc 68 f3dfe9a38f9e385cbfc5182ebbde11 	67 29a1ada1f98728754ff2e20a51el→ 68 f3dfe9a38f9e385cbfc5182ebbd	 67 29a1ada1f98728754ff2e20a51e
69 bf2cc3859fa342f782d23594a8c1627	69 bf2cc3859fa342f782d23594a8c162	69 bf2cc3859fa342f782d23594a0c	68 f3dfe9a38f9e385cbfc5182ebbd 69 bf2cc3859fa342f782d23594a0c
70 23d0d6c02c35ac3f14c0263b6310a8	70 23d0d6c02c35ac3f14c0263b6310a8	70 23d0d6c02c35ac3f14c0263b631	70 23d0d6c02c35ac3f14c0263b631
71 9168deb6ba7cef43e451d3d91c43151	71 9168deb6ba7cef43e451d3d91c431	71 9168deb6ba7cef43e451d3d91c4	71 9168deb6ba7cef43e451d3d91c4
72 4243df97de5896493d84c0797519d07+	72 40495cc044b4a69f441cd3f313de91	72 40495cc044b4a69f441cd3f313d +	 72 40495cc044b4a69f441cd3f313d
73 7129dc74306343ecfc84df072c86e1	73 7129dc74306343ecfc84df072c86e1	73 7129dc74306343ecfc84df072c8	73 7129dc74306343ecfc84df072c8
74 d05cfd4ed48ee02c27e8dd797141f1c	74 d05cfd4ed48ee02c27e8dd797141f1	74 d05cfd4ed48ee02c27e8dd79714	74 d05cfd4ed48ee02c27e8dd79714
75 4fefd59c86938487f0a3bdfa69b6501	75 4fefd59c86938487f0a3bdfa69b650	75 4fefd59c86938487f0a3bdfa69b	75 4fefd59c86938487f0a3bdfa69b
76 4d89c5b6e8fd16afb8ec79ebb9691dt	76 4d89c5b6e0fd16afb8ec79ebb9691c	76 4d89c5b6e0fd16afb8ec79ebb96	76 4d89c5b6e0fd16afb8ec79ebb96
77 d440f52ef591ca136abfe19db2982e7	77 d440f52ef591ca136abfe19db2982e	77 d440f52ef591ca136abfe19db29	77 d440f52ef591ca136abfe19db29
78 abdec82cd28cb5b52d05449246fdaf1	78 abdec82cd28cb5b52d05449246fda1	78 abdec82cd28cb5b52d05449246f	78 abdec82cd28cb5b52d05449246f
79	79	79	79
80 a3ceb22ef9589cb2a1b12c5636b8fac →	 # 80 falla61b9954cfb6ca81cd4cbdc5a1 	80 falla61b9954cfb6ca81cd4cbdc	80 falla61b9954cfb6ca81cd4cbdc
81 e28df0d6d8f58dc92a1f256d4c57777	81 lede633f4541e7765a0e2816475db7	81 1ede633f4541e7765a0e2816475	81 1ede633f4541e7765a0e2816475
82 96c8ad8cff5f5fed93f88d8ebf31286	82 559b2bbaf9a44f1e9c33aa6fdcfa2	82 559b2bbaf9a44f1e9c33aa6fdcf	82 559b2bbaf9a44f1e9c33aa6fdcf
83 a4faed67a3da37fb6436a63f9df8be1	83 343579a68ede5dc3d346b1f8280862	83 343579a68ede5dc3d346b1f8280	83 343579a68ede5dc3d346b1f8280
84 294ea0c1e43ff9884d7d63c3d7c594a	84 70115b26c1c45f633ddd89d4e51aac	84 70115b26c1c45f633ddd89d4e51	84 70115b26c1c45f633ddd89d4e51
85 ed6abd6d275d6a5b76c1ae095776d3c 86 2e5b8837a561d21936578ca586cd1cc	85 488ed8656f12	85 488ed8656f12	85 488ed8656f12
86 2e508837a561021936578ca586c01cc 87 3cade271911cff7156463f55d456f43		86 cfdac28b5261b236c08981dd472	86 cfdac28b5261b236c08981dd472
87 3cade2/1911ctt/156463t55d456t4: 88 22db997eb4115d5789cf2d6fa92abc2	87 1173df86b39022e8eabda3bac52cb1 88 cac98d5fca77aacd7ff1f4bf36682f	87 1173df86b39022e8eabda3bac52 88 cac98d5fca77aacd7ff1f4bf366	87 1173df86b39022e8eabda3bac52 88 cac98d5fca77aacd7ff1f4bf366
89 24fa5138e418c87046f5bef916c18fc	89 875fda9c543cad5296434a2a36bba5	89 875fda9c543cad5296434a2a30bl	88 cac98d5fca77aacd7ff1f4bf366 89 875fda9c543cad5296434a2a30b
90 6f60b1edf722b0cedf5676b52a2af16	90 3918f8412af1cdbcf79a7389d58c46	90 3918f8412af1cdbcf79a7389d58	90 3918f8412af1cdbcf79a7389d58
91 8679368ff87c92f265c2d2ae72fbcas	91 8679368ff07c92f265c2d2ae72fbca	91 8679368ff07c92f265c2d2ae72fl	91 8679368ff07c92f265c2d2ae72f
92 46b4eea63715e7a118c5f799dcf9d0c→	← 92 ca8cf68b0b5	92 ca8cf68b0b5	
93 af63bfec5b2ed8f64e92321fd25c3af	93 f4c09da73d91248698adc6389f3cc3	93 f4c09da73d91248690adc6389f3	93
94 198789ef8c837d76d75fdfdda8d6e3a	94 ba43667ea36f137e56d216cc359211	94 ba43667ea36f137e56d216cc359	94 5940391c15dfd420c5d01e43c8c
95 116736572d7b1ff979d1f18e655a5e5	95 d5d920fdefa0e3644272d3739d6ec	95 d5d920fdefa0e3644272d3739d6	95 25b941d9fba94f493f9b3170742
96 b54762f745419aac5ec14f58bf688c1	96 7ef4eaf52025185f7a12b99169a714	96 7ef4eaf52025185f7a12b99169a	96 9ee269e1c6a05b9a2e42ebebf3f
97 f950fafe26729cbaacb76fb2941582c	97 e2bb	97 e2bb	97 a38f09f833374207e154d28e3de
98 8729ef4094c7674a1184adc1bf59674	98 4fdce6384874f5d78333ec72e2c27€	98 4fdce6384874f5d78333ec72e2c	98 23c5ecca69bb6b1dc1ecd6fd4b4
99 32c614c78180334e641e9e3c8d8ff9e	99 ac514f6f711692b6b44fddde137fb4	99 ac514f6f711692b6b44fddde137	99 33953f6f921b67523af31c36341
108 c04dce7c62b6e69cd9f969baa14145	100 226d7ba0fe2027574b387c18567905	100 226d7ba0fe2027574b387c18567	100 7d11d31eb8f82ec657a6a9efe4c
101 dadb6c375c11abeac6beecac094c74t	101 68f7ab25e2a085a3cb075ffa3266d	101 68f7ab25e2a085a3cb075ffa326	101 7cba9ae1cb81f7be355b2929a02
102 9ad926757a3674137338f73c13e74d7	102 b95f2a1510b84c274584216fb95faa	102 b95f2a1510b84c274584216fb95	102 61aef1559b969f59576642e6be2
103 d1ec215474a7752483aaf70a12bb445	103 d1ec215474a7752483aaf70a12bb44	103 dlec215474a7752483aaf70a12bl	103 dlec215474a7752483aaf70a12b
104 f5b3fa8b254c257ccb5e1dc9e9fc8e7→ 105 e0f4177cb1aa6fdf1d222c9a5968d5c	104 488cc5ba84a769035deb51e6f95103 105 61e669ebff94e714e3fd9b2d0b4828	104 488cc5ba84a769035deb51e6f95 105 61e669ebff94e714e3fd9b2d0b4	104 488cc5ba84a769035deb51e6f95 105 61e669ebff94e714e3fd9b2d0b4
105 eee2f923213a3d3411ce44ee9da735t	105 516669601194671463109020004828 106 5304f5ac977dcbec1ee3578bdade99	105 51655960TT94671463Td902d004	105 5165596577946714637d952d054 106 5304f5ac977dcbec1ee3578bdad
107 996b1f765fe9592ad622048e7ca4ed6	107 f6b68f1fc7dc1962a7c4f91fd71e4t	106 530415ac977dcbec1ee3578bdad	100 530415aC977dcDec1ee3578bdad 107 f6b68f1fc7dc1962a7c4f91fd71
108 271999ee4e86444da7e98b963931622	108 e789ba90d2a44f8dc062c63e2f7b11	108 e789ba90d2a44f8dc062c63e2f7l	108 e789ba90d2a44f8dc062c63e2f7
109 cc72ab258078ecbb155384543380e09	109 5da591b1ce535fedeccece9fa266e4	109 5da591b1ce535fedeccece9fa26	100 5da591b1ce535fedeccece9fa26
110 8bd776dec553f398a9347cacecleec7	110 bc67b7a85117c2a79c76ad517abfec	110 bc67b7a85117c2a79c76ad517ab	110 bc67b7a85117c2a79c76ad517ab
111 fb2b71a8cdae8ae5412bbe94354da54	111 44fa8e32e12e5f23858a4633addbe2	111 44fa8e32e12e5f23858a4633add	111 44fa8e32e12e5f23858a4633add
112 bdcff089c0c7fdba9a23c63540f6bb8	112 3f4c36984db838015dc2e375107261	112 3f4c36984db838015dc2e375107	112 3f4c36984db838015dc2e375107
113 50a2e7dcc61a5bebcdaefdf5e347946	113 10be2ec5d53488151898444dad29d	113 10be2ec5d53488151898444dad2	113 10be2ec5d53488151898444dad2
114 ec1b4263b4a5c6b56d73c2b793174cf	114 edac62d1f7cc20e155b13c90ca4032	114 edac62d1f7cc20e155b13c90ca4	114 edac62d1f7cc20e155b13c90ca4
115 c636eb630c5a0e654f785196a95e2e1	115 c636eb630c5a0e654f785196a95e2e	115 c636eb630c5a0e654f785196a95	115 c636eb630c5a0e654f785196a95
116 771371e5932c541cc4c23f6c87875ca	116 771371e5932c541cc4c23f6c87875c	116 771371e5932c541cc4c23f6c878	116 771371e5932c541cc4c23f6c878
117 7398c9c999d19a658fa363ff53a0185	117 7398c9c999d19a658fa363ff53a018	117 7398c9c999d19a658fa363ff53a	117 7398c9c999d19a658fa363ff53a
118 6ccdd93167371f7d5f4818d377f0db	118 6ccdd93167371f7d5f4818d377f0dt	118 6ccdd93167371f7d5f4818d377f	118 6ccdd93167371f7d5f4818d377f

Andrew Brusso (MTU)

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

- 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.