Forensic Issues in IoT Devices Using NAND Storage

Project report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

Submitted by

Suchit Reddi (2010110507)

Under supervision

of

Dr. Sonal Singhal Department of Electrical Engineering



SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING SCHOOL OF ENGINEERING SHIV NADAR INSTITUTION OF EMINENCE

(December 2023)

Candidate Declaration

I/We hereby declare that the thesis entitled "Forensic Issues in IoT Devices Using NAND Storage" submitted for the B. Tech. degree program. This thesis has been written in my/our own words. I/We have adequately cited and referenced the original sources.

(Signature) Suchit Reddi (2010110507) Date: 27/11/2023

CERTIFICATE

It is certified that the work contained in the project report titled "Forensic Issues in IoT Devices using NAND Storage" by "Suchit Reddi" has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

(Signature) Dr. Sonal Singhal Dept. of Electrical Engineering School of Engineering Shiv Nadar Institution of Eminence Date: 27/11/2023

Abstract

This project focuses on rendering personal data unrecoverable on IoT devices utilizing NAND flash memory. Cloud providers use NAND flash storage to reduce latency. Due to high write and erase speeds, it is highly suitable for sensitive applications like DBMS and banking functions. Point-of-sale devices are an example of the wide usage of IoT devices to store critical and sensitive data.

It is logical to assume that many users are unaware of proper storage sanitization techniques. Personal information like medical or financial records must be sanitized before discarding storage devices. The recoverability of any erased file depends on the sanitization technique used and the determination of the entity attempting file recovery.

The working of modern storage devices using NAND flash memory differs from legacy magnetic drives. The old sanitization/recovery methods might not be effective. The effectiveness of a data sanitization technique on a specific device depends on various factors like type, compatibility, and level of access. Some will reduce the device's lifespan more than others. We will look into multiple data sanitization techniques at the hardware and software level.

Various tools are available to sanitize storage devices on laptops and computers with a display. However, IoT devices have limited access and resources. A tool that can be used on low-end IoT devices with just shell access is much needed. We developed a command-line utility to fulfill all these requirements. This will include the most effective and compatible methods for IoT devices.

4

Table of Contents

Lis	t of figures
Lis	t of commands
1.	Introduction
2.	Literature Review
	a. Erasing in HDD vs. SSD (pg no.9)
	b. Hardware Sanitization Techniques (pg no.9)
3.	Work done
	a. Sanitization Techniques researched (pg no.10)
	i. Firmware-based (pg no.10)
	ii. Cryptographic wipe (pg no.11)
	b. Verification (pg no.12)
	i. Tools used (pg no.12)
	ii. Process (pg no.13)
	c. Result
	i. Cryptographic wipe (pg no.13)
	ii. Firmware-based (pg no.16)
	d. Software sanitization tool (pg no.17)
4.	Conclusion
5.	Future Prospects
6.	References

List of Figures

3.1 Figure 1 of chapter 3 After OS-Level deletion	(pg no.11)
3.2 Figure 2 of chapter 3 After Encryption	(pg no.11)
3.3 Figure 3 of chapter 3 Setup	(pg no.12)
3.4 Figure 4 of chapter 3 Images extracted for testing	(pg no.13)
3.5 Figure 5 of chapter 3 Files in the initial SSD image	(pg no.14)
3.6 Figure 6 of chapter 3 Recovered folders	(pg no.14)
3.7 Figure 7 of chapter 3 After encryption	(pg no.15)
3.8 Figure 8 of chapter 3 After one pass, each of random data and zeros	(pg no.15)
3.9 Figure 9 of chapter 3 Final image of SSD	(pg no.16)
3.10 Figure 10 of chapter 3 Firmware sanitization compatibility	. (pg no.16)
3.11 Figure 11 of chapter 3 Memorywipe - sanitization methods	(pg no.17)
3.12 Figure 12 of chapter 3 Memorywipe - existing installation checking	(pg no.17)
3.13 Figure 13 of chapter 3 Memorywipe - sanitization	(pg no.17)
3.14 Figure 14 of chapter 3 Memorywipe - sanitization successful	(pg no.18)
3.15 Figure 15 of chapter 3 Memorywipe - a glimpse of source code	(pg no.18)
3.16 Figure 16 of chapter 3 Hardware-level sanitization	. (pg no.20)
a) Connections	(pg no.20)
b) Hack board for PoS	. (pg no.20)

List of Command

WARNING: Don't execute commands before learning what they do. Most of these commands result in data sanitization.

3.1 Command 1 Compatibility check for hdparm	(pg no.10)
3.2 Command 2 ATA enhanced secure erase	(pg no.10)
3.3 Command 3 Encryption using VeraCrypt	(pg no.11)
3.4 Command 4 Wiping process	(pg no.11)
3.5 Command 5 Imaging a drive using dd	(pg no.13)
3.6 Command 6 Unmanaged block image using mtd device	(pg no.20)

Chapter 1 Introduction

Advantages such as higher storage density, faster write and erase speeds, no mechanical latency from moving parts, and prices dropping every year in line with Moore's Law resulted in higher usage of NAND flash storage devices. They are used as cost-effective storage for IoT devices [1]. Solid State Drive (SSD) is a perfect testing device for this project that uses 100% NAND flash memory as its storage element [2][3].

IoT devices like Point-of-Sale devices store critical information like credit card data. The Target Breach, 2013, resulted from hackers scraping card data from the RAM and disk storage of PoS devices. About 40 million credit cards were stolen in a month [4]. Encrypting the storage from the beginning is recommended to avoid incidents like this and ensure data security and privacy, but it is not always possible. So, sanitization of data inside the disk storage of IoT devices is crucial.

In NAND flash devices, erase must be performed block-wise. If we flip and update a single bit in a block, the whole data block will be read and rewritten into a new block along with the altered bit. This creates a copy that is marked inaccessible but only deleted once wear leveling occurs. All these copies must be deleted for proper sanitization.

Data recovery and sanitization contradict each other. But progress in one drives the other. Sanitization tries to eliminate the possibility of recovery, while recovery tries to get back data even after sanitization is performed properly or improperly. Manufacturer-specific tools for sanitization exist but are not always compatible and are not implemented as expected [5].

We will discuss sanitization procedures for IoT devices with shell access, without any display or graphic user interface. We will look into possible hardware and software sanitization methods. We will test these methods using available equipment and observe the outcomes. All these methods are combined into a useful command-line utility for everyday users to sanitize and protect their sensitive information from prying eyes.

Chapter 2 Literature Survey

Erasing in HDD vs SSD:

Magnetic drives work differently from modern flash storage [6]. Overwriting mechanical disks with random bits once or multiple times is often enough. Degaussing randomizes the magnetization of grains on the magnetic medium of each disk, rendering it unusable. Physical destruction is a final step, and when done right, it ensures no possibility of data recovery. If any techniques are performed incorrectly, it is possible to recover data using advanced techniques such as Magnetic Force Microscopy. This process is discussed in detail by Vasu Kanekal in [7].

In SSDs, data is programmed electrically. So, most of the old methods will be ineffective. Simply overwriting the disk does not work because of wear leveling and TRIM performed by the NAND flash controller [8][9]. It fragments and stores a single file in various blocks. Flashspecific methods will be discussed in this paper.

Hardware Sanitization Techniques:

Sanitization can be done through software [10] or hardware. Performing hardware sanitization is complex and requires an advanced understanding of the intricate details of NAND chips and their interfacing. Experience handling specialized equipment like flash readers (PC-3000), programmers (Xeltek SuperPro), socket adapters (TSOP48 DIP48), and development boards (FTDI FT2232H) is needed.

This domain of research has very few resources online. Only a few researchers and enthusiasts performed operations directly on individual NAND flash chips and documented them [11][12][13][14]. This approach should provide a higher level of control over the sanitization process but is not very user-friendly. This hardware-oriented research might be useful for a specific device category called "mtd devices". Low-level codes for specific chips are written for mtd devices to write and erase data in block and bit levels. Our project does not cover this in-depth, but it is a good future prospect.

Chapter 3 Work Done

Sanitization on an SSD must be applied to the whole drive and not to smaller partitions or individual files because of file fragmentation by the NAND flash controller. After looking into many sanitization methods, we added the most effective and compatible ones to this section. For the source code, refer to the tool¹ we developed. This tool was made to be user-friendly to help the general public sanitize their storage drives.

I. Sanitization techniques researched:

Firmware-based:

This process uses firmware sanitization commands provided by the manufacturer. These are executed via different tools based on the device interface. ATA devices can use hdparm [18], SATA can use sg3-utils, and NVMe uses nvme-cli for sanitization. These tools only define functions like "secure erase" and "enhanced secure erase". The underlying operations are programmed by proprietary firmware complying with these standards. Thus, verifying if the command does what it claims is hard without source code.

It is a NIST-recommended method [15][16]. These commands run with higher access to the storage device at the firmware level. The "enhanced secure erase" claims to remove data even from the spare blocks with the help of the flash controller.

This method will be incompatible with most IoT devices, where cost-effective storage manufacturers do not prioritize security compliance. It depends on the device interface and manufacturer [17]. SCSI and UAS-interfaced devices are not supported.

Variables: (\$partition - device partition dev/sda*, \$strongp - password, \$name - device name)

sudo hdparm --sanitize-status \$partition `
 Cmd. 1: Compatibility check for hdparm

end. 1. compatibility check for hepathi

`sudo hdparm --user-master u --security-erase-enhanced \$strongp \$partition `

Cmd. 2: ATA enhanced secure erase

¹[Online]. Available: https://github.com/SuchitReddi/memorywipe

Cryptographic Wipe:

We came up with this procedure to compensate for the incompatibility of firmware-based sanitization. This method is compatible with all devices, irrespective of the manufacturer. The storage device is encrypted with a strong password. We used VeraCrypt [19] for encryption because it supports all major operating systems, including Raspberry Pi. It can be useful for IoT devices as it might support other devices running on ARM architecture like Pi.

`sudo veracrypt -t -c --volume-type=normal \$partition --encryption=aes --hash=sha-512 -filesystem=ntfs -p \$strongp --pim=0 -k "" --random-source=/dev/random ` Cmd. 3: Encryption using VeraCrypt

The encrypted device is then overwritten with one pass each of pseudorandom values and zeroes, using the `dd` command (Data Definition). Multiple passes will decrease the probability of data recovery but will have a negative impact on the drive's lifespan. The device is finally formatted to a usable filesystem.

`sudo dd if=/dev/random of=\$partition bs=1M status=progress && sudo dd if=/dev/zero of=\$partition bs=1M status=progress && sudo mkfs.ntfs -L \$name \$partition` Cmd. 4: Wiping process

When the whole drive is encrypted, it will appear as a single large file of garbage data. While overwriting the entire drive with random values, the NAND flash controller might mark some blocks as unmanaged. This might leave some original/encrypted data in the spare area, so the whole disk is overwritten again with a pass of zeroes. This drastically reduces the chance of recovering a complete unencrypted file in a readable format.

Hex			Т	ext			A		icati	on				F	ile M	etac	lata		OS Account
Page:	1	0	f (5260	0		Pa	ge	€	≯	Go	to P	age	: 1				J	ump to Offset
0x00000	0000	:	49	4E	44	58	28	00	09	00	F2	В4	20	00	00	00	00	00	INDX (
0x00000	0010	5	00	00	00	00	00	00	00	00	28	00	00	00	B0	00	00	00	
0x00000	0020	:	E8	0F	00	00	00	00	00	00	13	00	63	00	DA	01	63	00	c
0x00000	030	:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0x00000	040	:	3E	00	00	00	00	00	01	00	78	00	62	00	00	00	00	00	>x.b
0x00000	050	2	2A	00	00	00	00	00	04	00	4A	46	4F	02	FD	04	DA	01	*JF0
0x00000	0060	2	60	зc	3B	A4	8D	FE	D9	01	79	7B	A6	D2	8D	FE	D9	01	`<;y{
0x00000	0070	5	AA	6B	4F	02	FD	04	DA	01	00	A0	00	00	00	00	00	00	.k0
0x00000	080	:	AB	9F	00	00	00	00	00	00	20	08	00	00	00	00	00	00	
0x00000	090	:	10	00	73	00	68	00	65	00	72	00	6C	00	6F	00	63	00	s.h.e.r.l.o.c.
0x00000	00a0	2	єв	00	65	00	64	00	31	00	2E	00	6A	00	70	00	65	00	k.e.d.1j.p.e.
0x00000	00Ъ0	:	67	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	g
0x00000	00c0	:	10	00	00	00	02	00	00	00	78	00	62	00	00	00	00	00	x.b
0x00000	00d0	:	2A	00	00	00	00	00	04	00	4A	46	4F	02	FD	04	DA	01	*JFO
0x00000	0e0	:	60	3C	3B	A4	8D	FE	D9	01	79	7B	A6	D2	8D	FE	D9	01	`<;y{
0x00000	0±0	:	AA	6B	4F	02	FD	04	DA	01	00	A0	00	00	00	00	00	00	.k0
0x00000	100	:	AB	9F	00	00	00	00	00	00	20	08	00	00	00	00	00	00	
0x00000	0110	:	10	00	73	00	68	00	65	00	72	00	6C	00	6F	00	63	00	s.h.e.r.l.o.c.
									-										

Fig. 1: After OS-level deletion

Hex		Text			A	ppli	catio	on				F	ile M	etac	lata		OS Account
Page: 1	of	6393	36		Pag	ge	¢	>	Go	to P	age	1				J	ump to Offset
0x0000000	0: e	E 89	DA	96	8D	72	77	5A	5C	2C	94	AF	53	31	Fl	F2	nrwZSl
0x0000001	0: 9	F 1C	A1	AO	59	8E	46	A5	15	D5	25	7C	CC	6F	DD	50	Y.F% .o.P
0x0000002	0: 7	3 52	78	6F	EA	50	8C	CD	CB	E3	A6	FD	D4	90	C5	6C	sRxo.Pl
0x000003	0: E	3 72	2F	1B	15	23	42	90	A2	18	11	78	A8	07	60	AE	.r/#Bx`.
0x0000004	0: C	1 57	B 0	9A	9D	6C	F7	C7	8E	5F	42	7A	C1	3F	97	8A	.WlBz.?
0x0000005	0:8	8 26	20	86	E7	25	BE	DA	AA	AE	6B	Fl	BD	1A	99	23	.&*k#
0x000006	0: 9	9 29	C7	B4	BC	FE	33	60	CE	69	96	Ε9	BD	F3	32	5A	.)3`.i2Z
0x0000007	0: 7	3 A3	B6	8F	BO	04	11	E0	B8	75	4B	EB	AD	57	73	7C	suKWs
0x000008	0: 2	2 C5	6E	F5	13	B9	69	05	D5	31	DF	64	5E	ED	A 1	62	".nil.d^b
0x000009	0: E	6 FD	4F	89	44	76	2E	8B	B4	C0	A3	9D	F3	B 2	3C	FD	
0x000000a	0: 7	D A2	29	7F	B5	EC	33	9B	CO	07	8E	12	46	46	DF	F7	}.)3FF
0x00000b	0: E	FFE	2B	03	89	AD	69	AD	C4	BA	28	54	40	E3	F7	18	+i(T@
0x00000c	0: 9	1 D8	B 3	69	46	5D	DB	D1	F8	96	4D	98	17	FA	70	78	iF]Mpx
0x00000d	0: 3	0 53	1E	BF	37	E8	21	3B	F9	4D	B 0	СВ	Α7	56	48	EA	0S7.!;.MVH.
0x000000e	0: 9	9 DB	68	E9	7A	57	7C	F3	6A	FF	36	97	70	A6	27	34	h.zW .j.6.p.'4
0x00000f	0: 5	D Al	62	F6	33	2D	ЗD	0B	10	0C	C8	3F	8A	30	1E	DG].b.3-=?.0
0x0000010	0: 5	7 E9	08	52	0F	1A	E5	5E	22	32	Α9	21	45	27	D2	51	WR^"2.!E'.Q
0x0000011	0:8	B CO	7E	79	86	1E	59	2D	F7	00	92	2D	D2	D7	бD	6C	~yYml
<u>-</u>														7.2			

Fig. 2: After Encryption (Unreadable)

II. Verification

Tools used:

We used an 8GB SD card, a 32GB USB thumb drive, a 1TB HDD, and a 1TB SSD for testing various sanitization processes. Limited equipment and testable storage drives slowed the testing process and closed some paths altogether. To extensively test these sanitization functions, the required equipment is internal and external SSDs with different interfaces compatible with the SANITIZE feature set necessary for performing firmware-based sanitization.





But verification for successful sanitization on a NAND flash storage cannot be 100% certain when overprovisioning area comes into the picture. For absolute certainty that no data can be extracted, a physical chip read may be required to scan the OP blocks. Special equipment and some kind of documentation/tools from the manufacturer are needed.

The next best option is to use some forensic tool used by law enforcement that is capable of extracting incriminating evidence from storage devices. Some of the best out there are Magnet Axiom and Cellebrite UFED, but their subscriptions cost a fortune. So, we used Autopsy [20], a free, open-source tool that can run on Linux, Windows, and MacOS. Using `dd`, we extracted images from the target storage device, which was connected to an IoT device.

A command-line forensic recovery tool can verify successful sanitization on the same device where sanitization takes place. Scalpel or PhotoRec are such tools that can be incorporated into the tool in the future. This tool will be discussed in detail in further chapters.

Process:

1. Take a binary image of the storage device before starting the sanitization process. This command copies the entire drive, including the unallocated empty space.

`sudo dd if=\$partition of=/<output location>/image.bin status=progress `

Cmd. 5: Imaging a drive using dd

- 2. Send this image to the device running Autopsy and add the disk image as a data source.
- 3. Autopsy will show all the files in the binary image, including those recently deleted, using os-level deletion, which just removes the file pointers.
- 4. At different steps of the process, take images of the device using the dd utility.
- 5. When the image taken after sanitization is loaded into Autopsy, no file should be recoverable if the sanitization process is successful.

III. Results:

Cryptographic Wipe:

Cryptographic wipe is compatible with most devices irrespective of the manufacturer, unlike firmware-based sanitization. This method was successfully tested on all the devices, i.e., SD cards, USB thumb drives, HDDs, and SSDs. All these devices were successfully sanitized, and the personal data from before the process was unrecoverable.

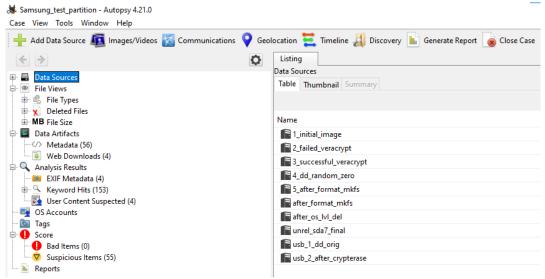


Fig. 4: Images extracted for testing

However, we will focus on the results of the SSD, which uses NAND flash storage. The verification process in detail is given below:

 The SSD was connected to the laptop with Autopsy on it, and its image was taken through Autopsy first, which was stored as `1_initial_image`. It consisted of various folders, as shown in the figure below.

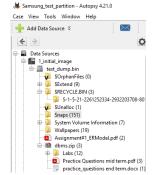


Fig. 5: Files in the initial SSD image

- 2. The same SSD was connected to the Raspberry Pi, and an image was taken via the Pi.
- 3. These images were successfully compared to ensure that Autopsy was error-free.
- 4. To test if a normal operating system level delete operation actually removes files, an image `after_os_lvl_del` was taken after deleting all folders using the `rm` command. All deleted files were recovered successfully, which emphasized the need for sanitization.

Name	S
SOrphanFiles	
CarvedFiles	
SExtend	
SRECYCLE.BIN	
🖌 SUnalloc	
[current folder]	
🖌 Snaps	
System Volume Information	
🖌 Wallpapers	
SAttrDef	
SBadClus	
SBadClus:SBad	
SBitmap	
SBoot	
SLogFile SBoot	
SIMFT	
SMFTMirr SMFTMirr	
SSecure: SSDS	
SUpCase	
SUpCase:SInfo	
SVolume	
Assignment#1_ERModel.pdf	
Assignment#1_ERModel.pdf:Zone.Identifier	
i dbms.zip	
dbms.zip:Zone.ldentifier	

Fig. 6: Recovered folders (red cross at the bottom)

- 5. VeraCrypt was installed on the Pi, and encryption was attempted while the drive was still mounted, which gave a failure message. This was stored as `2_failed_veracrypt`.
- 6. The drive was unmounted, and encryption was attempted again. This time, it was successful, and the extracted image was stored as `3_successful_veracrypt`. This image contained a single unallocated file with unreadable values from the first page to the last.

Samsung_test_partition - Autopsy 4	.21.0										-		×
ase View Tools Window Help + Add Data Source 📠 Images/V	ideos 🔣 Co	ommunicatio	ins ¥			\ge		••	Keyword	Lists	Q , Keyv	vord Sea	irch
< → Ö	Listing	1										\rightarrow	T
	/img_success	vera.bin										1	Result
Data Sources	Table Thu	mbnail Sun	mary										
1_initial_image			,										-
2_failed_veracrypt											Save	Table as	; CSV
3_successful_veracrypt	Name			s	с	0	Modif	ind To	ma	Chan	ge Time	Acces	ss Time
🗄 🧾 success_vera.bin					_	•						_	
4_dd_random_zero	🗙 Unalloc	_1767_0_104	7527424				0000-0	0-00	00:00:00	0000-0	00-00 00:00:00	0000-0	00-00 0
5_after_format_mkfs	<)
after_format_mkfs													
after_os_lvl_del			Anal	vsis Resi	dte				Annotati	0.00			
unrel_sda7_final	Hex	Text	Alla	Applica					Metadata	ons		ccount	
usb_1_dd_orig									victauata			ccount	
usb_2_after_crypterase	Page: 1	of 63936		Page	6 3	6	o to Pag	e: 1		1	ump to Offset		
File Views													
🖶 🏂 File Types	0x0000000 0x0000001			8D 72 59 8E			C 2C 94 5 D5 23	AF	53 31 CC 6F		nrwi		
B X Deleted Files	0x00000002			EA 50			5 D5 2: B E3 A		D4 90		sRxo.P.		
B MB File Size	0x0000003			15 23			2 18 1		A8 07		.r/#B		
Data Artifacts	0x0000004			9D 6C			E 5F 43		C1 3F		.Wl.		
	0x0000005	0: 88 26 2	0 86	E7 25	BE D	A A	A AE 61	8 F1	BD 1A	99 23			
Web Downloads (4)	0x0000006	0: 99 29 0	7 B4	BC FE	33 6	0 C	E 69 94	6 E 9	BD F3	32 5A	.)3	.4	22
🔍 Analysis Results	0x0000007	0: 73 A3 H	16 8F	B0 04	11 E	0 B	8 75 41	B EB	AD 57	73 7C	s	.uKW	(s)
- 🔤 EXIF Metadata (4)	0x000008			13 B9			5 31 DI		5E ED		".ni.		
🗄 🕒 Keyword Hits (153)	0x0000009			44 76			4 C0 A:		F3 B2		O.Dv.		
- 🛃 User Content Suspected (4)	0x000000a			B5 EC			0 07 81		46 46		}.)3		
- 📑 OS Accounts	0x00000b		B 03	89 AD	69 A				40 E3		+1		
🛯 🔯 Tags	0x000000c			46 5D			8 96 41 9 4D B		17 FA A7 56		iF]		
Score	0x0000000			37 £8 78 57			9 40 BI A FF 31		A7 56 70 86		h.zWI		
Reports		0. 99 DB N		15 57	10 2	J 5.	n 22 31	5 37	10 AD	61 3M	n.zwj.	1.0.P.	

Fig. 7: After encryption

- 7. The encrypted drive was overwritten with one pass of random values from the source /dev/random. This should result in all blocks being marked invalid, erased, and overwritten. There is a chance for OP spare blocks to be swapped with encrypted blocks.
- 8. So, we write a pass of zeros over the random data again, forcing the controller to either swap OP spare blocks with random data blocks or erase the existing blocks.
- 9. A drive will only have a small percentage of spare blocks. So, the flash controller must clear most of the drive even after swapping spare blocks with random or encrypted blocks. The extracted image at this step was stored as `4_dd_random_zero`.

€ ⇒	C Listing	random zero		•	eoloc	ation	-	imeline	a Di	scove	ry _	Genera	ite Kepo	rt 👍 Close	Case
🗄 📕 Data Sources		humbnail													
🖶 💼 1_initial_image		numonali .	Summa	ary											
⊕ ■ 2_failed_veracry															
B 3_successful_ve					s	с	0		ied Time						
4_dd_random_z					S	C	0				_	hange Time		Access Tim	-
dd_random		lloc_1770_0_	104752	7424				0000-0	0-00 00:	0:00	00	00-00-00 0	0:00:00	0000-00-00	00:00:0
🖶 🔚 5_after_format_															
🖶 🌆 after_format_m	fs <														
after_os_lvl_del															
🐵 🔚 unrel_sda7_fina	Hex Te:	xt Applicat	ion Fi	ile Meta	adata	OS Ac	count	Data A	Artifacts	Anal	ysis	Results Co	ontext	Annotations	Othe
🕀 🌆 usb_1_dd_orig	Page:	1 of 639	936	Р	age	6 4	Go	to Pag	r: 1			Jump to	Offset		1
Image: Book of the second s	rase				-9- L	· ·									
File Views	0x00000	000.000	0 00	00 00	0.00	00 00	0.0	00.00	00 0	0 00	00	00			
	0×00000	010: 00 0	0 00	00 00	00	00 00	00	00 00	00 0	0 00	00	00			
🖶 🗟 File Types				00 00	00	00 00	00	00 00	00 0	0 00	00	00			
		020: 00 0													
🖶 🗟 File Types	0x00000	020: 00 0			00	00 00	00	00 00	00 0	0 0 0	00				
🖶 🗟 File Types 🖶 😠 Deleted Files	0x00000 0x00000		0 00	00 00		00 00 00 00	00			0 00					
 B File Types B V Deleted Files B B File Size 	0x00000 0x00000 0x00000	030: 00 0	0 00	00 00	00		00		00 0		00	00			
Gamma Sile Sile Types Solution Sile Sile Sile Sile Sile Sile Sile Sile	(4) 0x00000 0x00000 0x00000 0x00000	030: 00 0 040: 00 0 050: 00 0 060: 00 0	0 00 0 00 0 00 0 00	00 00	00 00 00 00	00 00 00 00 00 00	00	00 00	00 0	0 00 0 00 0 00	00 00 00	00			
File Types File Types You Deleted Files MB File Size Data Artifacts Metadata (56) Web Download	(4) (x00000 (x00000 (x00000 (x00000 (x00000 (x00000	030: 00 0 040: 00 0 050: 00 0 060: 00 0 070: 00 0	0 00 0 00 0 00 0 00 0 00	00 00	000000000000000000000000000000000000000	00 00 00 00 00 00	00	00 00	00 0	0 00 0 00 0 00 0 00	00 00 00	00			
Generation of the second	(4) 0x00000 0x00000 0x00000 0x00000 0x00000 0x00000 0x00000	030: 00 0 040: 00 0 050: 00 0 060: 00 0 070: 00 0 080: 00 0	0 00 0 00 0 00 0 00 0 00 0 00	00 00 00	0 00 00 00 00 00 00 00 00 00 00 00 00 0	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00	00 00	00 0	0 00 0 00 0 00 0 00 0 00	00 00 00 00	00 00 00 00			
Sile Types Sile Types Sile Types Deleted Files MB File Size Data Artifacts Web Download Analysis Results Silf Metadata (6)	(4) (4) (4) (4) (4) (4) (4) (4)	030: 00 0 040: 00 0 050: 00 0 060: 00 0 070: 00 0 080: 00 0 090: 00 0	0 00 0 00 0 00 0 00 0 00 0 00 0 00	00 00 00 00 00 00 00 00 00 00	000000000000000000000000000000000000000	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00	00 00	000 0 000 0 000 0 000 0 000 0	0 00 0 00 0 00 0 00 0 00 0 00	00 00 00 00 00 00	00 00 00 00 00			
Second State	(4) (4) (4) (4) (4) (4) (4) (4)	030: 00 0 040: 00 0 050: 00 0 060: 00 0 070: 00 0 080: 00 0 090: 00 0 090: 00 0	0 00 0 00 0 00 0 00 0 00 0 00 0 00 0 0		0 00 00 00 00 00 00 00 00 00 00 00 00 0	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00	00 00	00000 00000 00000 00000 00000	0 00 0 00 0 00 0 00 0 00	00 00 00 00 00 00	00 00 00 00 00 00			

Fig. 8: After one pass, each of random data and zeros

- 10. After the passing of zeros, the drive will be unreadable without a file system. So, it is formatted. Recovering personal data from unmanaged blocks among all the encrypted, random, and zeroed blocks in the overprovisioning area will be practically impossible.
- 11. With this step, the cryptographic wipe process is complete. So, the final image was saved as `5_after_format_mkfs`. This final image only contained the filesystem but no recovered files before the cryptographic wipe.

Name	S	С	0	Modif	ied Time		Chang	ge Time		Acc	ess Time		Created Tir	ne	Size
😺 SOrphanFiles				0000-0	0-00 00:00	:00	0000-0	0-00 00:0	00:00	000	0-00-00:00	:00	0000-00-00	00:00:00	0
SExtend				2023-1	1-08 21:19	:08 IST	2023-1	1-08 21:1	19:08 IST	2023	3-11-08 21:19	:08 IST	2023-11-08	21:19:08 IST	344
🗸 SUnalloc				0000-0	0-00 00:00	:00	0000-0	0:00 00:0	00:00	0000	0-00-00 00:00):00	0000-00-00	00:00:00	0
[current folder]				2023-1	1-08 21:19	:08 IST	2023-1	1-08 21:1	19:08 IST	2023	3-11-08 21:19	:08 IST	2023-11-08	21:19:08 IST	56
SAttrDef			3	2023-1	1-08 21:19	:08 IST	2023-1	1-08 21:1	19:08 IST	2023	3-11-08 21:19	:08 IST	2023-11-08	21:19:08 IST	2560
SBadClus				2023-1	1-08 21:19	:08 IST	2023-1	1-08 21:1	19:08 IST	2023	3-11-08 21:19	:08 IST	2023-11-08	21:19:08 IST	0
SBadClus:SBad				2023-1	1-08 21:19	08 IST	2023-1	1-08 21:1	19:08 IST	202	3-11-08 21:19	:08 IST	2023-11-08	21:19:08 IST	104752332
SBitmap			1	2023-1	1-08 21:19	-08 IST	2023-1	1-08 21:1	19-08 IST	202	3-11-08 21:19	-08 IST	2023-11-08	21:19:08 IST	31968
SBoot			0		1-08 21:19			1-08 21:1			3-11-08 21:19			21:19:08 IST	8192
SLogFile			1		1-08 21:19			11-08 21:1			3-11-08 21:19			21:19:08 IST	5234688
SMFT			0		1-08 21:19			11-08 21:1			3-11-08 21:19			21:19:08 IST	27648
SMFTMirr			0								3-11-08 21:19				
			U		1-08 21:19			11-08 21:1						21:19:08 IST	4096
Secure:\$SDS					1-08 21:19			11-08 21:1			3-11-08 21:19			21:19:08 IST	262396
SUpCase			3	2023-1	1-08 21:19	:08 IST	2023-1	11-08 21:1	19:08 IST	2023	3-11-08 21:19	08 IST	2023-11-08	21:19:08 IST	131072
\$UpCase:\$Info				2023-1	1-08 21:19	:08 IST	2023-1	1-08 21:1	19:08 IST	2023	3-11-08 21:19	:08 IST	2023-11-08	21:19:08 IST	32
SVolume				2023-1	1-08 21:19	:08 IST	2023-1	1-08 21:1	19:08 IST	2023	8-11-08 21:19	:08 IST	2023-11-08	21:19:08 IST	0
Hex Text Applicati Page: 1 of 639			tadata Page 🧉		ount Data		ts Ana	-	ults Con ump to O		Annotations	_	Occurrences	>	
0x00000000: EB 52	2 90 4	E S	4 46 5	3 20	20 20	20 00	02 08	00 00	.R.1	NTFS					
0x00000010: 00 00	0 00 0	0 0	0 F8 0	00 00	3F 00	FF 00	00 28	51 74			?(Qt				
0x0000020: 00 00			0 00 8		FF 37			00 00			.7				
0x00000030: 04 00			0 00 0		7F F3			00 00							
0x00000040: F6 00 0x00000050: 00 00			1 00 0 E 1F E		32 F0 7C AC			E9 5C 56 B4			2.@A\				
0x00000060: 00 00			D 10 5		F0 32			19 EB			2				
0x000000070: FE 54			3 20 6		20 6E			20 62			not a b				
0x00000080: 6F 6B			2 6C 6		64 69			50 6C			disk. Pl				
0x00000090: 65 61			0 69 6		65 72			62 6F			ert a bo				
0x000000a0: 6F 74			C 65 2		6C 6F			61 6E			loppy an				
0x00000000: 64 01			2 65 7	3 73	20 61	6E 79	20 6B	65 79			any key				

Fig. 9: Final image of SSD

12. This step verifies the successful deletion of personal information from the SSD.

Firmware-based:

This process works only if the SANITIZE feature set is enabled for a storage device. We tested these commands on two notable storage devices: a Samsung T7 1TB SSD interfaced via SCSI (external) and a Toshiba MQ04ABF100 1TB HDD via SATA (internal). The SATA-interfaced device was compatible, while the SCSI-interfaced device was not, which was expected. As the compatible device is a personal drive still in use and was tested by mistake in the first place, further tests were not conducted on it. However, the ATA_hdparm() function we developed for our tool has been tested by others and was reported to work without errors.

*	SANITIZE_ANTIFREEZE_LOCK_EX	T command	
• • • • • • • • • • • • • • • • • • •	SANITIZE feature set		
*	OVERWRITE_EXT command		
*	reserved 69[1]		
*	Extended number of user add	moscable sectors	
	DOWNLOAD MICROCODE DMA comm		
×	DOWNLOAD MICROCODE DMA COMM	anu	
Security:			
Maste	r password revision code = 655	34 1	
	supported		
not	enabled		
not	locked		
	frozen		
not	expired: security count		
	supported: enhanced erase		
188mi	n for SECURITY ERASE UNIT. 188	min for ENHANCED	SECURITY ERASE UNIT
	WN Device Identifier: 5000039		SECONITY ERASE ONIT:
NAA	www.Device Idencifier. 5000039	902303480	
	ANT DATE TO P		
	DUI : 000039		
	e ID : 9c25034a8		
Checksum: cor			
(sherl0ck@	sherl0ck 2:52)-[~]		
<u>_\$</u>			

Fig. 10: Firmware sanitization compatibility (`supported: enhanced erase` should appear)

IV. Software sanitization tool:

As discussed in this paper, we developed a command-line utility to perform sanitization, extraction, and verification. It is named `memorywipe` and can be found here [21]. It has options for manual and automatic execution, with minimal user interaction. The sanitization techniques discussed in this paper were executed in this tool.

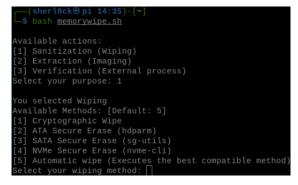


Fig 11. Memorywipe - sanitization methods

Once the user selects a method, the tool checks for the required programs and installs them if they are absent. If a known failure occurs at any point, it displays the reason and necessary tips to solve it. It also lets the user know the best way to apply a sanitization technique.

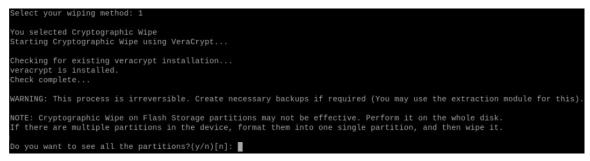


Fig. 12: Memorywipe - existing installation checking

The tool can even take users with little technical knowledge along the process by

abstracting the commands executed and automating wherever required.

Do you want	to see	all th	ne parti	tions?(y/n)[n]: y		
	FSTYPE	FSVER	LABEL	UUID	FSAVAIL FS	USE% MOUNTPOINTS
sda						
	ntfs		testusb	55C9623B7C5C3D61	28.6G	0% /media/sherl0ck/testusb
mmcblk0						
-mmcblk0p1					397.3M	22% /boot/firmware
└─mmcblk0p2	ext4	1.0	rootfs	3c215345-c59c-4a79-941a-ab121e090a42	8.1G	37% /
Unmounting t /dev/sda1 ha				unmounted.		
/dev/sda1 ha	is been	succes	sfully (unmounted.		
				or automatically? (m/a)[a]: a		
Uses AES, wi	th SHA	-512 ar	nd makes	an NTFS filesystem		
Enter strong	passwo	ord for	encrypt	ting. (You don't have to remember it)		
				ecial characters: 108dklancdi*sdt^*(5	#ej*JgF69(7	46

Fig. 13: Memorywipe - sanitization (Cryptographic Wipe)

Enter strong password for encrypting. (You don't have to remember it) So set a random password with special characters: 1923jhaHdYT6*(f4=%&hf\$8kr54^0ld Finished encrypting /dev/sda1 Wiping /dev/sda1. Unmounting the partition... /dev/sda1 already unmounted /dev/sda1 overwritten with one write each of random data and zeroes Set device name: testdevice Wiped /dev/sda1 Mounting /dev/sda1 at /media/testdevice mount: (hint) your fstab has been modified, but systemd still uses the old version; use 'systemctl daemon-reload' to reload. Finished mounting! Cryptographic Wipe procedure completed successfully!

Fig. 14: Memorywipe - sanitization successful

The whole program is a shell script where operations are performed using a combination of different functions.

men	norywipe.sh 🕱
184	⊒crypt_wipe() {
185	echo "Starting Cryptographic Wipe using VeraCrypt"
186	echo
187	ins_veracrypt #Script should not proceed if this step fails.
188	echo
189	
190	echo "WARNING: This process is irreversible. Create necessary ba
191	echo
192	echo "NOTE: Cryptographic Wipe on Flash Storage partitions may r
193	echo "If there are multiple partitions in the device, format the
194	echo
195	list_partitions
196	
197	# Setting a value for partition (/dev/sdb1) to wipe
198	echo
199	echo "Select /dev/sda, if you want to wipe the whole drive, part
200	read -p "Enter your device's partition (/dev/sda1): " partition
201	echo
202	
203	# Making sure disk is unmounted
204	if chk_unmount; then
205	echo
206	else
207	echo "Unmounting failed!! Please unmount manually before proc∈
208	echo
209	return 1
210	fi fi
211	
212	# Start the process
213	veracrypt_encrypt
214	echo
215	wipe_disk
216	echo
217	mount_disk
218	echo
219	echo "Cryptographic Wipe procedure completed successfully!"
220	L}
221	### Cryptographic Wipe ends here
222	
	###ATA Secure Erase starts here
	-# Installing hdparm
225	Fins_hdparm() {

Fig. 15: Memorywipe - a glimpse of source code

Chapter 4 Conclusion

Our exploration into secure sanitization provided an understanding of its significance and methodologies. We emphasized the requirement of disk storage sanitization for IoT devices. Differences in the working of older mechanical drives and modern drives utilizing NAND flash were highlighted. The need for a sanitization technique that can work on devices like SSDs was acknowledged and addressed.

Different techniques and their working were explored, and the most compatible ones with IoT devices were discussed in this paper. A method for verifying the success or failure of sanitization techniques was discussed and implemented. The detailed results of the sanitization process and verification were shown.

Moreover, our efforts continued beyond theoretical analysis. We took a proactive step by implementing the discussed techniques as a command-line utility developed using shell scripting. This helped in converting our research into a real-world solution that the general public can use to sanitize their storage devices, giving them the power to protect themselves.

Research into a more certain sanitization and verification by utilizing hardware techniques is very much needed. With the disturbing increase in cybercrime in recent years, there is a strong need to stay ahead in the relentless pursuit by maintaining data security and thorough sanitization. Through our project, we have not only highlighted the challenges faced but also presented a viable solution to improve personal data security.

Chapter 5

Future Prospects

- 1. The memory wipe tool in development can be further improved by adding better sanitization, extraction, and verification methods, which induces low wear to chips.
- 2. Incorporating terminal-based verification methods (PhotoRec, SleuthKit's Scalpel).
- 3. Preparing tools specific to widely used OS, device types, interfaces, etc.
- 4. Automating the tools to remove user interaction for IoT devices with limited access.
- 5. Exploring sanitization possibility for IoT devices without shell access.
- 6. Possibility of sanitizing storage on mobiles using ADB command interface.
- 7. Performing hardware-based bit and block-wise operations on NAND chips to improve the effectiveness of the sanitization process.
- 8. Factory Access Mode is similar to rooting a phone and allows low-level command execution but is limited to the manufacturers and service centers [22].
- 9. The Flash transition layer might allow low-level access without special hardware equipment. If it can, researching this can improve the certainty of sanitization.
- 10. `mtd` devices seem to support these low-level operations, where an image including unmanaged blocks can be extracted using the below command.

`nanddump -s \$partition --bb='dumpbad' -p -n `

Cmd. 6: Unmanaged block image using mtd device



(a)

(b)

Fig. 16: Hardware-level sanitization (a) Connections (b) Hack board for PoS

References

- [1] S. Bennett and J. Sullivan, "NAND Flash Memory and Its Place in IoT," 2021 32nd Irish Signals and Systems Conference (ISSC), Athlone, Ireland, 2021, pp. 1-6, doi: 10.1109/ISSC52156.2021.9467859.
- [2] Sanvido, Marco & Chu, Frank & Kulkarni, Anand & Selinger, Robert. (2008). NAND Flash Memory and Its Role in Storage Architectures. Proceedings of the IEEE. 96. 1864 -1874. 10.1109/JPROC.2008.2004319. M. N. DeMers, Fundamentals of Geographic Information Systems, 3rd ed. *New York: John Wiley*, 2005.
- [3] Blog post. Single package SSDs reduce size of IoT devices. [Online] Available from: https://www.electronicspecifier.com/products/memory/single-package-ssds-reduce-sizeof-iot-devices [Accessed 26th November 2023]
- [4] Blog post. Target Breach. [Online] Available from: https://www.securityweek.com/target-confirms-point-sale-malware-was-used-attack/
 [Accessed 26th November 2023]
- [5] Wei, Michael & Grupp, Laura & Spada, Frederick & Swanson, Steven. (2011). Reliably Erasing Data from Flash-Based Solid-State Drives. 105-117.
- [6] Xiaolu Zhang and Kim-Kwang Raymond Choo. 2019. Digital Forensic Education: An Experiential Learning Approach (1st. ed.). Springer Publishing Company, Incorporated.
- [7] Kanekal, Vasu. 2013. "Data Reconstruction from a Hard Disk Drive using Magnetic Force Microscopy."
- [8] Ahn, Na Young & University, Dong. (2022). Security of IoT Device: Perspective Forensic Anti-Forensic Issues on Invalid Area of NAND Flash Memory. IEEE Access. 10. 10.1109/ACCESS.2022.3190957.
- [9] Blog Post. Solid-State Drive (SSD) File Recovery Challenge. [Online] Available from: https://perez-aids.medium.com/solid-state-drive-ssd-file-recovery-challengecbde1935e33a [Accessed 26th November 2023]
- [10] Blog Post. (2023). Data Sanitization Methods. [Online] Available from: https://www.lifewire.com/data-sanitization-methods-2626133 [Accessed 26th Nov 2023]
- [11] Jeong Wook, Matt Oh. (2020). Reverse Engineering Flash Memory for Fun and Benefit.

Available from: https://www.blackhat.com/docs/us-14/materials/us-14-Oh-Reverse-Engineering-Flash-Memory-For-Fun-And-Benefit.pdf [Accessed 26th Nov 2023]

- [12] FT2232H NAND flash reader. [Online] Available from: https://spritesmods.com/?art=ftdinand&page=1 [Accessed 26th November 2023]
- [13] Blog Post. Dumping a SLC NAND Flash with Atmel PMECC. [Online] Available from: https://www.mickaelwalter.fr/dumping-a-slc-nand-flash-with-atmel-pmecc/ [Accessed 26th November 2023]
- [14] Murza, Yoshi. (2018). Reading Nand Flash with Raspberry Pi 3 b. [Online] Available from: https://www.youtube.com/watch?v=hHkBseIblBM [Accessed 26th November 2023]
- [15] Regenscheid, A., Feldman, L., & Witte, G. (2015). Pg-44, NIST Special Publication 800-88 Revision 1, Guidelines for Media Sanitization (No. ITL Bulletin February 2015).
 National Institute of Standards and Technology.
- [16] Computer Security Resource Center (CSRC). SANITIZE Command. Available from: https://csrc.nist.gov/glossary/term/sanitize_command [Accessed 26th November 2023]
- [17] Jon Tanguy. Data Sanitation: Securely Erasing Micron SATA SSDs. [Online] Available from: https://www.micron.com/-/media/client/global/documents/products/technicalmarketing-brief/brief_ssd_secure_erase.pdf?la=en [Accessed 26th November 2023]
- [18] Blog Post. Securely wipe an SSD with its built-in commands. [Online] Available from: https://code.mendhak.com/securely-wipe-ssd/ [Accessed 26th November 2023]
- [19] Veracrypt. [Online] Available here: https://veracrypt.fr [Accessed 26th November 2023]
- [20] Autopsy. [Online] Available here: https://www.autopsy.com [Accessed 26th Nov 2023]
- [21] Memorywipe. [Online] Available here: https://github.com/SuchitReddi/memorywipe [Accessed 26th November 2023]
- [22] Blog Post. Life after Trim: Using Factory Access Mode for Imaging SSD Drives.
 [Online] Available from: https://blog.elcomsoft.com/2019/01/life-after-trim-using-factoryaccess-mode-for-imaging-ssd-drives/ [Accessed 26th Nov 2023]

22