Forensic Trails Obfuscation and Preservation via Hard Drive Firmware

Paul Underhill[1], Toyosi Oyinloye[1], Lee Speakman[2], and Thaddeus Eze[1]
[1]The Department of Computer Science, University of Chester, UK
[2]School of Science, Engineering and Environment, University of Salford, UK
p.underhill@chester.ac.uk
t.oyinloye@chester.ac.uk
l.speakman@salford.ac.uk
t.eze@chester.ac.uk

Abstract: The hard disk drive stores data the user is creating, modifying, and deleting while a firmware facilitates communication between the drive and the operating system. The firmware tells the device and machine how to communicate with each other and will share useful information such as, disk size and information on any bad sectors. Current research shows that exploits exist that can manipulate these outputs. As an attacker, you can change the size of the disk displayed to the operating system to hide data in, likewise by marking an area of the disk as bad. Users may not be aware of these changes as the operating system will accept the readings from the firmware. However, although the data is not reachable via the operating system this paper looks at the traceability of manipulated data using data recovery software FTK Imager, Recuva, EaseUS and FEX Imager. This report examines the use of malicious techniques to thwart digital forensic procedures by manipulating the firmware. It is shown how this is possible and current forensic techniques or software does not easily detect a change within the firmware. However, with the use of various forensic tools, obfuscated trails are detectable. This report follows a black box testing methodology to show the validation of forensic tools or software against anti-forensic techniques. The analysis of the results showed that most tools can find the firmware changes, however, it requires an analyst to spot the subtle differences between standard and manipulated devices. The use of multiple software tools can help an analyst spot the inconsistencies.

Keywords: hard drive firmware, digital forensics, data recovery, data manipulation, security analysis

1. Introduction

The use of forensic analysis to examine devices has become a standardised way to collect evidence or help provide insight into cyber attacks. The use of these methods are relied upon in examinations and are expected to show a true, un-modified record of the data on the device. However, what if the firmware of the device could be changed? This could hide, change, or manipulate what an analyst sees when first mounting the drive to their machine. This could prevent an accurate and comprehensive analysis of the device, especially if the analysts do not probe further into the reliability of the devices recorded details such as its true size and features.

The process of compromising or changing the integrity of evidence in this way is known as anti-forensics (Harris, 2006). The Parliamentary Office of Science and Technology (2015) explain that having such a wide range of devices and firmware will increase difficultly for an analyst to spot maliciously changed storge media. The National Police Chiefs Council (Hewitt, 2021) suggest that the biggest issue with the collection of evidence will be ensuring analysts have the knowledge and capability to respond to these more sophisticated attacks. The last point the NPCC make is the issue of completing investigations promptly while sticking to strict budgets. Here a challenge arises, to perform a thorough review of the device in a forensically sound manner. To overcome this the Attorney General's Guidelines (2013) allows tools to be employed for imaging, searching and collection of evidence to increase efficiency. This paper will show how the use of malicious firmware could thwart detection when analysed with these tools, and how a manual inspection is a more reliable way to find the data. The main test will be to examine a drive where the data has been saved to a hidden area on the disk that has been changed at a firmware level. The success of each tool will be recorded, any further work that can build upon this and any solutions that can be offered to identity compromised firmware will also be discussed.

1.1 Research question / aim

Etow (2020) found little research has been published assessing the reliability of forensic tools against anti-forensics. This lack of findings means methodologies for anti-forensic attacks have not been securitised for their effectiveness. Therefore, this research will complete an anti-forensic strategy to test forensic and recovery software to find its effectiveness.

The aim of the research is to manipulate displayed data on a disk by editing the firmware. This leads to the research questions, 'Can the change be implemented without current forensic software detecting it and can the data be extracted or recovered in a forensically sound manner'? This will allow the findings to conclude any forensic failings during the testing stages.

2. Literature review

2.1 Firmware overview

All devices that communicate between an Operating System (OS)(software) and a storage medium (hardware) will need some sort of translation to be able to interpret the transferred data. This is the principle of firmware, it acts as a bridge between the hardware and software, by translating data from one another. The firmware has the highest privilege on a system (Hassan, Markantonakis and Akram, 2016). This allows the OS to tell the user details of the drive, such as the size or prevent the OS from saving data to any areas on the disk that may be marked as "bad".

When the device is turned on the firmware will begin a self-check, this ensures correct device operation and will mark unreadable areas of the disk or hide any areas that are not used by the OS. Each device has specific firmware and unique checks (Sutherland, Davies and Blyth, 2011). However, each firmware has a similar layout. For example, part A of the firmware will be on the printed circuit board and the drive will contain a service zone to store defect lists and self-monitoring analysis and reporting technology attributes which monitor the health of the drive. For example, read error rate, performance, temperatures, and the total number of hours the device has been used. This is a small selection of recordable attributes and encompasses numerous different tests (NTFS.com, 2021). Firmware part B will hold the Logical bock Addressing (LBA) translator table. If all the checks are successful it will begin to load the OS.

2.2 Forensic operations

Digital forensics is the science of dealing with digital information created, stored, and transmitted by electronic devices (Shanmugam, 2011). The data uncovered can be used to inform investigations and legal proceedings. The data contained on devices must be extracted, analysed, and presented using methods and software that is validated by the UK Forensic Science Regulator. This means the use of a scientifically proven method aimed at preservation, validation, identification, analysis, documentation, and presentation of such evidence (UK Forensic Science Regulator, 2015).

The Forensic Science Regulator (2015) has a strict framework so that investigative processes can be repeatable and therefore relied upon at court. This is also true for the tools used, as both the regulator and investigating analyst will need to validate software before it can be approved for use (UK Forensic Science Regulator, 2015). However, Becket and Slay (2007) explain these tests may not find all errors or limitations of the software, as this would require extensive resources which may not be readily available. The validation criteria are set out in standards such as ISO 17025:2005, which is balanced between cost, risk, and technical possibilities. Therefore, the limitations of software, and loopholes in standards, allows malicious users to exploit issues that may not have been explored. Most forensic analysis is focused on the operating system level. They will take direct copies of items such as storage information or the registry (Jeong and Lee, 2019; Wundram et al., 2013). This relies on the firmware being correct as the operating system will display what it receives.

2.3 Anti-forensic techniques

It is difficult to pinpoint the first firmware level attack. However, "Vault 7", a collection of documents found on Wikki Leaks shows that the CIA had capabilities in 2013 of snooping via firmware vulnerabilities, such as rootkits and injecting malicious code (Vording, 2018). It is suggested that four categories exist for firmware level attacks against forensic processes. Artefact wiping, data hiding, trail obfuscation and attacks against the tools or the processes (Rogers and Lockheed, 2005; Chandran, 2013; Majed, Noura and Chehab, 2020).

2.3.1 Artifact wiping

Stewin and Bystrov (2012) discovered it is possible to extract or delete cryptographic keys from a SSD device. This can be implemented by placing malware within the firmware, as the high privilege level could allow access to the Direct Memory Access (DMA) area. The DMA is used by the OS and legitimate software to securely

communicate with hardware devices. The research by O' 'Hara and Malisow (2016) showed that by removing these keys it would be impossible for an analyst to access any data. Additionally, if the use of crypto shredding were employed it would render the whole disk useless and completely unrecoverable (O' 'Hara and Malisow, 2016).

Harris (2006) found that using hard drive ATA secure erase features can make it almost impossible for data to be recovered. The ATA erase feature is a disk scrubber built into the device at a firmware level and will destroy all data on all areas of the disk, regardless of whether it is protected by the OS or other such features. Marupudi (2017) found that using the secure erase setting to permanently write and erase can cause wear levelling in SSD drives. This is when areas of the device have been written to, deleted and re-written until the area is damaged by overuse, becoming unreliable to store data. By allowing this to run over the whole disk it would render the storage device useless to any forensic analysis.

2.3.2 Data hiding

As the OS relies on firmware, data can be manipulated so rogue information about the hard drive is sent to the OS. These changes may not be apparent to a user, as performance may not seem to be affected during typical use (Haswell, 2006). Haswell (2006) shows that the firmware can tell the OS that the size of the internal storage is different to its physical. He found that a malicious user can add data to these hidden areas, and the OS will not display them. Brezinski and Killalea (2002) showed similar findings and found that malicious software can also be obfuscated in these hidden areas. This can then run during the imaging process making changes to the disk and invalidating the results. This poses an issue for law enforcement, as some forensic tools have been identified as not discovering the hidden areas on the internal storage device (Surtherland et al., 2009). Their research found that if the malicious user marks sectors as bad on the drives error lists within the firmware, the tools would not display the bad areas as they rely on the firmware's translator operating correctly.

Gruhn (2017) conducted similar research, showing that the size of the drive can be changed by changing the firmware overlays, partition tables and marking sectors as bad. However, went on to show that the EEPROM can be used as standalone hidden storage. Gruhn (2017) and Jeong and Lee (2019) both suggest that this area is not scrutinised by forensic tools or processes.

2.3.3 Trail obfuscation

Trail obfuscation is to mislead or confuse a forensic technique or analyst and ultimately the investigation. This can be achieved in several ways, deleting logs, IP/MAC spoofing, proxy servers, the use of zombie accounts and misinformation (Shanmugam (2011). Shanmugam (2011) shows this is possible with the use of the Metasploit framework "transmogrify". He showed that you can change the header file information, allowing the malicious user to change an image to a document at a header level. When the forensic tools scan the device for images this malicious file may not show as it is declared as a document.

Cho (2016) found several tools that can be used to change the creation, last modified and recent timestamps for files on the OS. Once this change has been made the forensic tools would display the modified date and time to the analyst. Schicht (2014) backs this statement saying the changes will not be detected unless the analysts look in shadow copies and logfiles. The recommendation is to also change the data for the shadow copy. Schicht (2014) suggests the logs are only short so the evidence will not be retained for long. He suggests that finding this information is "not trivial work". Gul and Kugu (2017) found that similar software "Timestomp" successfully changed the dates and times. But the forensics tools could easily spot the changes. The research found that copying the file to a new location after using the tool then prevented the original data from being found.

2.3.4 Attacks against forensics tools and processes

Firmware anti-forensics can be integrated at the time of imaging, this could be to destroy, hide or manipulate data. All these examples would damage the integrity of the evidence. Laurenson (2016) suggests current forensic software tools do not interrogate these damaged areas, leading to a gap for malicious attackers to manipulate evidence. Du, Ledwith and Scanlon (2018) further back up Laurenson (2016) and compared several forensic tools, finding that they did not scan the firmware of disks. Sutherland et al. (2009) used well-known forensic software Encase, FTK and AccessData. They found that changing the error lists in the firmware was not detected by any of the software tested. Further analysis also found that changing the Logical Block Addressing (LBA) table

to the actual Cylinder Head-Sector (CHS) location prevented the software from making a full image. These issues are caused by the forensic software relying on the LBA provided by the firmware.

Harris (2006) argues that the above research focuses too much on software ability and suggests that research should focus on the human element of forensic analysis. He explains that as humans we can be easily deceived by invalid dates or deleted logs. Therefore Harris (2006) and Rogers and Lockheed (2005) both suggest not to automate the whole procedure. Harris (2006) further explains that the software could work on a "what should be" rather than a "what is" functionality. This would allow the software to scan the master file table and notify the analysis of any non-standard code, such as NTFS flags. This would allow the analysts to review the change and allow the possibility of spotting anti-forensic techniques before they cause further issues. Lieberman (2000), found that less automation allowed analysts to follow impulses or intuitions leading to the examination being impaired. Adderley (2019) also noted that manual examinations would become very timely and costly, impacting digital forensics within law enforcement. He suggests that a mix of both automated and human manual exploration would be the best of both approaches.

3. Research methodology

This research is based on black box testing by Wilsdon and Slays (2006). They suggest following the six steps shown below:

- 1. Acquisition of software
- a) The software used to forensically analyse the procedure should be listed with version numbers, installed patches or any modifications that have been made to the tool.
- 2. Identification of software functionalities
- a) The functions of the tool are documented, via user manuals or use of the software etc. Only identified functionality of the software should be evaluated.
- 3. Development of test cases and reference sets

All tests must be based on black box testing methods, examining previously listed functions.

- a).Tests are prepared for each function of each tool. Some tests can be used on multiple tools, but all tests should be based on real-world scenarios.
- 4. Development of result acceptance spectrum
- a). The use of black-box testing allows the expected result to be expressed in advance. This allows any deviation from this result to be easily identified.
- 5. Execution of tests and evaluation of results
- a). All results will be compared against the acceptance spectrum compiled in section four.
- b). Any function that is not an expected result will be evaluated as "Failed."
- c). Any functions that are expected will be evaluated as "Passed."
- 6. Release of evaluation results
- a) Upon conclusion of all previous phases, the results should be made available to the community.

(Bhat, AlZahrani and Wani, 2021; Flandrin et al., (2014).

The results within this report will be presented on a scale from 0-3.

0 – No trace of data at all.
1 - Location or name of file found but no data within.
2 - Location and name of file found, some data recovered from within.
3 - Fully recovered document with all contents intact.

The use of this scale will allow a range of results to be shown, as data recovery can be partially fulfilled the scale will help show the true reliability of the tests and results.

3.1 Tools and software

The below tools have been selected for the research implementation tasks. This includes software to image, check, analyse the drive, and data. In line with the methodology, the tools functions have been listed, version numbers provided and a brief description. To further add to this the hardware used has been listed with additional relevant information.

Table 1: Information on tools, software, and hardware

Tool Name	Version	Functions
Md5Checker	3.3	Calculate and display MD5 Checksums. Verify if files have been changed and integrity of files.
FTK Imager	4.5	Create forensic images of various media. Preview and recover files, folders, and contents of files.
Recuva	1.53.1087	Advanced file recovery.
EaseUS	14.2	Recover from lost or deleted partitions.
FEX Imager	2.2.0(263)	Acquire disk images, files and hash information.
Hex Editor HxD	2.5.0.0	Hex Editor.
HDDHackr	1.40	Flashes the firmware on hard drives.
Bootable USB Creator	4.0	Creates bootable disk on USB to install Dos.
Western Digital 320GB	WD3200BEVT	The hard drive that will be manipulated.

The below files were added to the disk. The table below shows the data in the original state, once the tests have been concluded the original hash values will be compared against the extracted data to identify any differences.

Table 2: Original data on the device

File Type	File Name	Md5	Content
.docx	doc1	8A6AF8B3B74C2D9A44868F48B82EBB9A	Text
.txt	text doc	0C46BDBA0B995AEAF4A42BBE527D8B25	Text
.png	image 1	C37EE8102ADE5A30A390FE81F3D75014	Image
.jpg	image 2	066F76801FBE38BB54015AFA60ED51CB	Image

4. Implementation

When first connecting the device to the machine (in its original state) we can see Windows recognises it as a 298.09GB disk. This is close to 320GB and would be expected as other space is used for configuration, shadow copy services etc. Before editing the firmware, a copy of the original is taken from the disk. The Hex editor is then used to examine sector 16, the security sector for the drive containing information on the firmware and disk. Figure 1 shows the first bytes of data in that sector. The hex editor displays a series of "20's" this is hex for white space or empty areas. After this, a list of "decoded text" is shown. Each of these are explained in table 3 and shown in the corresponding figures.

Table 3: The layout of sector 16

Offset	Data Type	Decoded Text		Hex Value
00 – 0B	Empty	Space		
0C - 13	Serial Number	6VDC44HD	36 56 44 43 34 48 44	
14 – 1B	Firmware Revision	0002CE02	30 30 30 32 43 30 32	
1C - 26	Manufacturer/Model Number	ST9320325AS	53 54 39 33 32 3033 32 35 41 53	
58 – 5B	LBA Size		B0 EA 42 25	

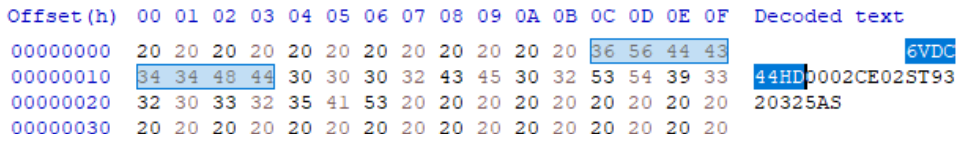

Figure 1: The serial number location

Understanding the layout allows the correct area to be changed. For instance, if a user intended to update the serial or model name, the numbers or letters can be converted to hex and entered here. This will produce a .bin file that can be uploaded to the device.

4.1 Changing the device size

Only four bits need to be changed to show the disk drive size at offset 58 to 5B. The disk size is displayed in little-endian, so B0 EA 42 25 becomes 25 42 EA B0. By multiplying this by 512 (the standard sector size) it will give the total disk size. However, this is in Hex, converting the hex to decimal can be seen in table 4.

Table 4: Converting hex to decimal original hard drive size

Hex	25 42 EA B0	X	200	=	4A 85D5 6000
Decimal	625,142,448	X	512	=	320,072,933,376

Table 4 shows that the original size of this drive is 320GB. To update the firmware to show the disk size as 120GB the following hex would need to be used: B0 4B F9 0D converted to little-endian 0D F9 4B B0.

Table 5: Converting hex to decimal for new firmware

Hex	0D F9 4B B0	x	200	=	1B F297 6000
Decimal	234,441,648	x	512	=	120,034,123,776

Therefore, giving us a total space of 120GB. At this point, the .bin file is ready to be uploaded to the device. This is done using HDDHACKR. The software works by sending vendor-specific commands to modify the firmware. It allows the user to enter specific commands.

5. Results

When connecting the device to a Windows operating system the drive is showing as 111.79GB. At this point the recovery is taken pre and post initialisation.

5.1 Unallocated volume tests

The first set of results are taken before the disc is not initialised. When scanning the drive with FTK imager it shows the device as 120GB. The forensic software showed that it has unpartitioned space [basic disk]. Within this unpartitioned space, it shows some data exists, when searching more in-depth you can find the data fully recovered. EaseUS data recovery shows the drive as 111.79GB, as expected with the changed firmware. However, lists it as a 'device' rather than a hard drive, as previously shown. All data was successfully recovered.

Before scanning the drive FEX shows the user an overview screen. Here the software displayed the serial number as 000000000. However, the sectors (234441648) and sector size (512) displayed shows that the drive is 120GB as programmed within the firmware. All data was successfully recovered.

Recuva was unable to recover any of the data on the disk.

Table 6: Results from the unallocated disk drive

File type	Windows Explorer	Recuva	EaseUS	FTK imager	FEX Imager
Docx	0	0	3	3	3
Text	0	0	3	3	3
.png	0	0	3	3	3
.jpg	0	0	3	3	3
Md5	0	0	3	3	3

5.2 Allocated volume tests

For these series of tests, the drive was assigned a drive letter to simulate a normal drive on Windows operating system. The same recovery and forensic analysis methods are employed.

FTK and FEX did find traces of the data but was not able to display it fully. Although Recuva and EaseUS used advanced or "deep" scans fully recovering the images and the .docx file. However, they did not find any trace of the text document. All other data was found. Recuva found the two images and the document however the .jpg md5 had changed from the original file. All other MD5's stayed the same.

The forensic tools were able to detect these files but could not display them. This may be due to the software not being able to recover these files in a forensically sound way. This can be proposed as the recovery software changing the MD5 hash value.

Table 7: Results from the allocated volume

File type	Windows Explorer	Recuva	EaseUS	FTK imager	FEX Imager
Docx	0	3	3	1	1
Text	0	0	0	1	1
.png	0	3	3	1	1
.jpg	0	3	3	1	1
Md5	0	0	0	0	0

6. Analysis

6.1 Drive unallocated

When scanning the drive FEX imager showed the serial number as 0000000. If the serial was explicitly entered into the firmware, it may not have displayed this. However, the software still displayed the size as 120GB. This mismatch in the firmware may allow analysists to pick up on this.

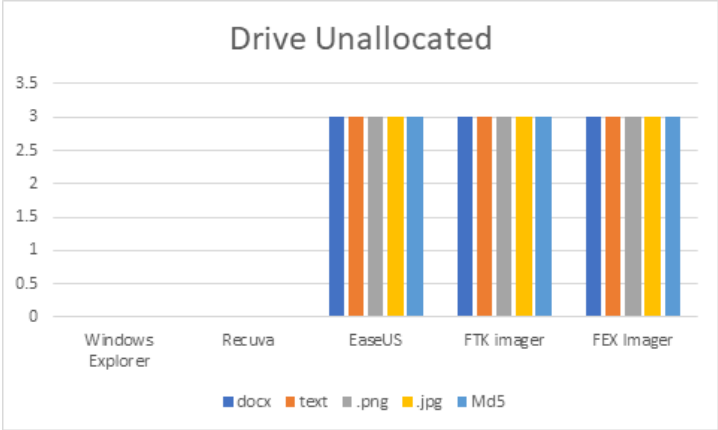

Figure 2: Results from the unallocated drive

Once the firmware was uploaded the drive was shown to the host machine as one drive with a total of 120GB. This new total is the unallocated size, and the first examinations were taken when the drive was in the uninitialized state. As the disk cannot be mounted by the operating system Windows and Recuva cannot begin to extract any data. This may suggest that Recuva is relying on the OS to display and scan the drive. EaseUS, FTK and FEX imager all managed to extract the data in a forensically sound manner (figure 2). All the data was clearly shown in “unpartitioned space”. This could suggest that these software tools do not rely on the firmware to give an accurate reading from the disk.

6.2 Drive allocated

The drive was then allocated a drive letter and the tools used to scan the disk again. Windows relied on the firmware to give all details to display. Therefore, no data was found as the new allocated drive “D” was an empty drive.

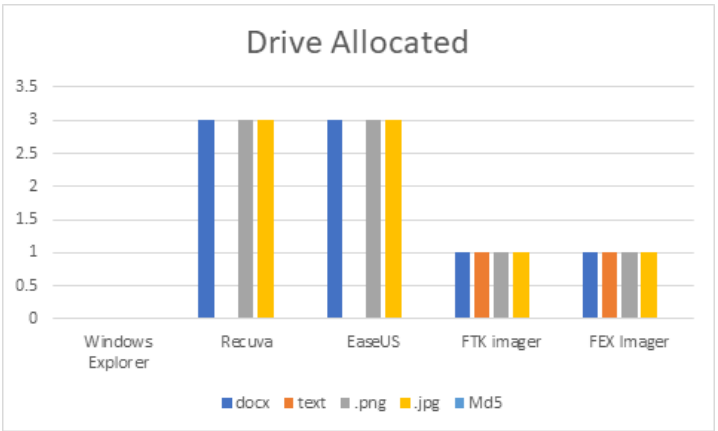

Figure 3: Results from the allocated drive

Both recovery software tools managed to recover a readable version of docx, png and jpg. However, although the data was fully recovered and looked identical to the originals, the MD5 hashes did not match. The recovered files also did not have the same title as originally given; this may suggest the header information has been lost or removed for these files, or the forensic tools have renamed the found files. It is unsure why this has happened; it may be that this change occurred when the drive was allocated a drive letter. As the previous test provides all MD5 hashes are maintained (figure 3).

FTK and FEX imager did find traces of the data but not fully recovered within the software itself. The tools may have prevented the automatic recovery of these files as it would cause the MD5 to be changed. This would then prevent the data from being used in law enforcement or as evidence in cases. As both forensic tools did not provide a full recovery of the files the hashes could not be compared.

The lack of recovery of the '.txt' file from both data recovery programmes may be from the tools not displaying unrecoverable data to be more user friendly. Another approach to finding the text document could be data carving, where the user can search the HEX for specific details within the document. File carving can be completed using tools that search the files header information, where contains the start of data points and 'the end of file' in the unallocated data. The data between the two points can be extracted and examined further. Processes such as data carving could lead to further data being recovered; however, this was out of scope for this experiment.

6.3 Answers to research question

To determine if the anti-forensic technique had been completed in an undetectable way, the disk drive was examined with forensic software. Most of the software showed maliciously changed firmware values, and found the unpartitioned space from scans. The main test for the tools was to see if they showed any traces that a change had taken place. Other than displaying the unpartitioned space, most software packages did not show any tampering of the drive. Only one forensic tool showed that the disk drive serial number was incorrect. In most investigations the recovery of data would need to be reliable. Therefore, the firmware change has obstructed some of the forensic tools from acquiring the data in a forensically sound manner. This was shown by the altered files, backed up by the MD5 hash not being the same as the original file.

A1 Based on the research findings within this study, the data shows that the firmware can be manipulated to change, hide, or delete data. Although each of these had various outcomes overall the idea to prevent investigations from proceeding was seen as achieved.

A2 The findings have shown that the technique can be hidden from most software used here. All but one has shown that the drive was not tampered with. Although the results did find the unpartitioned space, the software did not suggest any tampering or changes to the disk had been made or found. The only software that did was FTK imager, with the unusual serial number.

A3 The results from this experiment have shown that most data could be found but not necessarily read, displayed, or recovered in a forensically sound manner after the firmware change. This shows that it would hinder an investigation. Although this goal was achieved, it is still important to point out that recovery of the data is still possible.

7. Conclusion

To conclude, it has been shown that forensic and recovery tools can recover data in most situations. However, some of the techniques within this research can hinder forensic analysis. This is mainly recovering deleted or hidden files, impacting the reliability of data. It is important that an analyst can spot signs of malicious changes before data is extracted for use in an investigation.

Further research on decompiling the forensic tools and examine the inner workings could be conducted. This may give a clearer answer to the discovery of empty unpartitioned space and why it exists on a disk that shows it as different size. This might show if the software uses a mixture of hardware and software-based firmware to scan the drive, or if the software searches the drive without talking to the firmware. Research on the use of hardware modified firmware would be a good next step. Hutchins (2015), Cipriani (2016) and Yliluoma (2018) all report similar attacks outlined in this paper but with tougher to recover artefacts.

Overall, all the signs of tampering will most likely be picked up by some tools. Therefore, the analyst should not rely on any one tool but a library. The examination should closely note the serial numbers and the size displayed by the software compared to the drive itself, although the label could easily be changed. If this is the case, comparing the model number and name is important. Most manufactures will have white space before or after the manufacturing name. This again can be an oversite by an attacker who may not have observed this when modifying the firmware.

References

Adderley, A., 2019. Graph-based Temporal Analysis in Digital Forensics. Masters. Harvard University.

Attorney General's Office, 2022. Attorney General's Guidelines on Disclosure For investigators, prosecutors and defence practitioners. [online] Assets Publishing Service. Available at: < shorturl.at/jyHJ7 > [Accessed 26 August 2021].

Bhat, W., AlZahrani, A. and Wani, M., 2021. Can computer forensic tools be trusted in digital investigations? Science & Justice, 61(2), pp.198-203.

Brezinski and Killalea, 2002. Guidelines for Evidence Collection and Archiving. [online] Rfc-editor. Available at: <https://www.rfc-editor.org/pdfrfc/rfc3227.txt.pdf> [Accessed 26 July 2021].

Chandran, R., 2022. Overview of Digital Forensics and Anti-Forensics Techniques. Masters. Auckland University of Technology.

Cho, G., 2016. Data Hiding in NTFS Timestamps for Anti-Forensics. International Journal of Internet, Broadcasting and Communication, 8(3), pp.31-40.

Cipriani, T., 2016. Coreboot on the ThinkPad X220 with a Raspberry Pi. [online] Tylercipriani.com. Available at: <https://tylercipriani.com/blog/2016/11/13/coreboot-on-the-thinkpad-x220-with-a-raspberry-pi/> [Accessed 12 June 2021].

Du, X., Ledwith, P. and Scanlon, M., 2018. Deduplicated Disk Image Evidence Acquisition and Forensically-Sound Reconstruction. 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp.1674-1679.

Etow, T., n.d. Impact Of Anti-Forensics Techniques On Digital Forensics Investigation. Ph.D. Linnaeus University.

Gruhn, M., 2017. Forensic limbo: Towards subverting hard disk firmware bootkits. Digital Investigation, 23, pp.138-150.

Gul, M. and Kugu, E., 2017. A survey on anti-forensics techniques. 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), pp.1-6.

Harris, R., 2006. Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem. Digital Investigation, 3, pp.44-49.

Hassan, R., Markantonakis, K. and Akram, R., 2016. Can You Call the Software in Your Device be Firmware?. 2016 IEEE 13th International Conference on e-Business Engineering (ICEBE), pp.188-195.

Haswell, J., 2016. SSD Architectures to Ensure Security and Performance. In: Flash Memory Summit.

Hewitt, M., 2021. Npcc. [online] Npcc Police UK. Available at: <https://www.npcc.police.uk/ThePoliceChiefsBlog/Default.aspx> [Accessed 10 March 2021].

Hutchins, M., 2016. Hard Disk Firmware Hacking (Part 1). [online] MalwareTech. Available at: < shorturl.at/fvHR0 > [Accessed 29 June 2021].

Jeong, D. and Lee, S., 2019. Forensic signature for tracking storage devices: Analysis of UEFI firmware image, disk signature and windows artifacts. Digital Investigation, 29, pp.21-27.

Laurenson, T., 2016. Automated Digital Forensic Triage: Rapid Detection of Anti-Forensic Tools. Ph.D. University of Otago.

Lieberman, M., 2000. Intuition: A social cognitive neuroscience approach. Psychological Bulletin, 126(1), pp.109-137.

Majed, H., Noura, H. and Chehab, A., 2020. Overview of Digital Forensics and Anti-Forensics Techniques. 2020 8th International Symposium on Digital Forensics and Security (ISDFS),.

Marupudi, S., 2017. Solid State Drive: New Challenge for Forensic Investigation. [online] St. Cloud State University. Available at: <https://repository.stcloudstate.edu/msia_etds?> [Accessed 16 July 2021].

NTFS.com, 2021. S.M.A.R.T. Attributes. [online] NTFS. Available at: <https://ntfs.com/disk-monitor-smart-attributes.htm> [Accessed 17 June 2021].

O'Hara, B. and Malisow, B., 2017. Ccsp (ISC)2 certified cloud security professional official study guide. CA: John Wiley & Sons.

Rogers, M. and Lockheed, M., 2005. Anti-forensics - Lockheed Martin. CA.

Schicht, J., 2014. GitHub - jschicht/SetMace: Manipulate timestamps on NTFS. [online] GitHub. Available at: <https://github.com/jschicht/SetMace> [Accessed 17 June 2021].

The Parliamentary Office of Science and Technology, 2016. Digital Forensics and Crime. [online] Parliament UK. Available at: <https://post.parliament.uk/research-briefings/post-pn-0520/> [Accessed 19 June 2021].

Shanmugam, K., 2011. Validating digital forensic evidence. Ph.D. Brunel University Uxbridge.

Stewin, P. and Bystrov, I., 2013. Understanding DMA Malware. Detection of Intrusions and Malware, and Vulnerability Assessment, pp.21-41.

Sutherland, I., Davies, G. and Blyth, A., 2011. Malware and steganography in hard disk firmware. Journal in Computer Virology, 7(3), pp.215-219.

Sutherland, I., Davies, G., Pringle, N. and Blyth, A., 2009. The Impact of Hard Disk Firmware Steganography on Computer Forensics. Journal of Digital Forensics, Security and Law, 4(2), pp.73-84.

UK Forensic Science Regulator, 2015. Overseeing Quality. [online] publishing Service UK. Available at: <https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/470526/FSR_Newsletter_26__October_2015.pdf> [Accessed 20 July 2021].

Vording, J., 2018. Vault 7 and the Paradox of Democratic Society. Masters. Leiden University.

Wilsdon, T. and Slay, J., 2006. Validation of forensic computing software utilizing black box testing techniques. Edith Cowan University.

Wundram, M., Freiling, F. and Moch, C., 2013. Anti-forensics: The Next Step in Digital Forensics Tool Testing. 2013 Seventh International Conference on IT Security Incident Management and IT Forensics, pp.83-97.

Yliluoma, J., 2018. Replacing BIOS in a ThinkPad with GPL CoreBoot. [online] Youtube. Available at: < shorturl.at/cuvyS > [Accessed 12 August 2020].