

# Bachelor Thesis

Bachelor's Programme in IT-forensics and Information Security, 180 credits



## An exploratory forensic analysis of the Xbox One S All Digital

Digital Forensics, 15 credits

Halmstad 2020-06-01

Victor Elfving, Robbin Lidström



## Abstract

Gaming consoles' relevance to the field of digital forensics has steadily been growing since their presence in society has increased. Given how gaming platforms, such as the Xbox One, are produced for commercial interest, they are likely to be secured by use of proprietary knowledge to safeguard personal data. The means by which information is secured is unknown, thus displaying the need for investigations to determine what information can be extracted from hard drive disk images and whether any of it is personally identifiable data. Furthermore, predecessors to the Xbox One were successfully modified by users, allowing unsigned code to be run; however, this is currently not possible on the Xbox One. In addition, due to the generational aspect of game consoles, proper digital forensic methodology needs to be developed specifically adapted to the Xbox One. An exploratory approach was pursued to allow for the scope to remain dynamic, letting information found to point to additional avenues of investigation and research. No personally identifiable information was found, yet the analysis of selected files allowed for hypotheses concerning their intended purpose. Through file analysis, encryption was found to be in use on the console. Moreover, the Master File Table was demonstrated as a significant extension to the foundation of console forensics methodology. Lastly, it was established that the Xbox One successfully prevents the running of unsigned code, showing a significant improvement compared to its predecessors.

## Table of Contents

1. Introduction.....	1
1.1 Research questions .....	2
1.2 Problematisation of research questions .....	2
2. Background.....	4
2.1 File systems .....	4
2.1.1 NTFS .....	4
2.1.2 FAT .....	5
2.1.3 Comparison of NTFS and XFAT .....	6
2.2 Running modified software on the Xbox One .....	10
2.3 Related work .....	11
3. Method .....	13
3.1 Tools.....	15
3.2 Physical tools .....	16
4. Experiment arrangement.....	18
5. Results.....	19
5.1 Autopsy analysis .....	19
5.1.1 First Image.....	19
5.1.2 Second Image .....	21
5.1.3 Third image .....	26
5.2 Scalpel & bulk_extractor.....	31
5.2 Parsing the MFT – a logical view .....	31
5.2.1 Partition 1 - Temp Content.....	31
5.2.2 Partition 2 – User Content.....	34
5.2.3 Partition 3 – System Support.....	37
5.2.4 Partition 4 - System Update.....	38
5.2.5 Partition 5 – System Update 2.....	41
6. Discussion .....	42
7. Conclusion .....	44
References.....	I
Glossary .....	IV

# 1. Introduction

Although gaming consoles have existed for a substantial length of time, their inclusion and relevance in/to the field of digital forensics has been limited. As advancements within console development have been made, such as cost, popularity and technology, their overall use across society has increased dramatically. This increase in popularity alongside the transition from gaming platforms to greater multimedia systems has led to a correlated increased relevance to the field of computer forensics. As mentioned above, the change in console devices' purpose to more closely resemble the intended use of personal computers has prompted the realisation that the data contained on such consoles can contain personally identifiable or other critical information whose relevance to a digital forensic investigation is significant.

In line with the shift in purpose and the ever-decreasing gap of technology used, 8<sup>th</sup> generation consoles have more in common with personal computers than ever before. The Xbox One (XB1<sup>1</sup> hereinafter) uses the NTFS filesystem as opposed to XTAF, the more proprietary filesystem used in its predecessor, the Xbox 360, which was based on the FAT-16 and FAT-32 filesystems. Analysis of the filesystem used will therefore be necessary and critical to forming a comprehensive understanding of its limitations and what the impact upon forensic imaging and data extraction will be.

Digital forensic investigations and the evidence produced depend, in large part, on the tools employed by the practitioners. This leads to the requirement that the tools used are relevant and applicable to their target system. This dissertation will cover which tools can be employed, their strengths and weaknesses in a general sense, whilst also specifically highlighting and justifying the choice of tools used in the experiment. The results gained from said tools will be compared to any extracted information from other research papers / projects to conclude whether the XB1 can be considered more or less secure than its predecessor. In this way, the volume, type and veracity of the data extracted can be compared to that of other experiments performed within the same field of study (console forensics).

---

<sup>1</sup> This report will use XB1 to refer to the Xbox One S All Digital version

As mentioned above, the field of console forensics could still be considered to be in its infancy, thus forensic procedures will most likely not have been uniquely adapted and instead attempts to mirror industrial best practices to the maximum extent possible. As is the case with any other project of similar scope, the methodology followed will need to be detailed and proofed in order to secure the necessary data, prevent unwanted data manipulation or data loss and reflect the order of volatility. Inspiration will be drawn from other research papers in order to ultimately be able to judge the evolution of the console. Most likely, the methodology used in other console forensic investigations will need to be evaluated by this report's authors and uniquely adapted to the topic matter.

Storage forensics will be explored in order to determine the validity of processes followed. Central to this issue is preventing false positives by tools that perhaps were not developed for use on consoles. Given that storage forensics can be considered an umbrella term, incorporating the above-mentioned areas (tools and filesystem analysis), its focus will be on determining the value of data gained by judging the results' veracity.

## 1.1 Research questions

The four questions that will be attempted to be answered are:

- What information can be extracted from an XB1 through use of a digital forensically acquired image?
- Develop a foundation for a valid forensic methodology specifically adapted to the 8<sup>th</sup> generation console, the XB1, that accounts for its unique characteristics.
- How, and to what extent, do the implemented security features of the 8<sup>th</sup> generation console, the XB1, affect a forensic examiner's ability to extract uniquely identifiable information from a forensic disk image?
- Does the XB1 represent significant advancements in OS modification prevention when compared to its predecessor?

## 1.2 Problematisation of research questions

This report's research questions will at first glance seem broad in scope; however, the very nature of this dissertation is its focus on performing a valid and complete digital forensic analysis of the XB1 gaming console. Therefore, given this dissertation's exploratory nature, the results obtained will provide further avenues for more detailed specificity and contribute

to a more thorough understanding of the subject area. Succinctly summarised, the scope will be left dynamic to account for new information/approaches discovered as a result of this dissertation's exploratory nature.

The lack of attention the field of console forensics has garnered since consoles' inception alongside their increased frequency in society displays the need for a functional methodology applicable to 8<sup>th</sup> generation gaming consoles. In order to contextualise this dissertation's usefulness to the academic world, the XB1's predecessor will provide a legitimate basis for comparison; primarily, the focus will be on the methodology used

## 2. Background

### 2.1 File systems

This section will aim to provide the reader with the necessary information needed to comprehend the importance of the file system within the context of digital forensics. Moreover, at the end of this section, a comparison will be drawn between the file systems covered. In the interest of presenting succinct information, the file systems will not be described in minute detail; however, if such points were to remain unexplained post-comparison, they will be explained during, or after, the comparison.

#### 2.1.1 NTFS

The New Technology File System (NTFS hereinafter) is a file system developed by Microsoft for use in personal computers. Its first version, 1.0, was used in Windows NT 3.1 with new versions being released for each new OS in the future, up to 3.1 in Windows XP. This version is still being used in Windows 10, albeit one with minor updates added throughout its development process. (Custer, 1994) It is also the file system in use on the XB1.

It was developed as the successor to the FAT file systems, many of which are still in use today, as well as the High Performance File System (HPFS hereinafter). During the 1980s, Microsoft and IBM collaborated to create the next generation of graphical operating systems with OS/2 being the resulting product. However, due to disagreements on various issues, this partnership was terminated with IBM continuing working on and employing OS/2 with HPFS, whereas Microsoft started development of Windows NT ultimately resulting in the coupled creation of NTFS. (Kozierok, 2001)

In the interest of brevity and of limiting scope, although including several new features over its predecessors, these will not be covered in full detail. In summation, these were:

(Microsoft, 2007)

- Sparse file support
- Disk use quotas
- Reparse points
- Distributed link tracking
- File-level encryption through use of the Encrypting File System (EFS)

There were several goals associated with the development of NTFS. The objectives are summarised below (in no specific order) (Kozierok, 2001):

**Reliability:** The new file system needed to be able to recover from problems without loss of data.

**Security and Access Control:** The file system needed to have built-in controls to provide access control to all folders and files on a hard disk. This was something the FAT file systems lacked, and its inclusion in the new file system would be a significant improvement.

**Partition Size:** The file system needed to allow large partition sizes for future scalability. This would have been a direct improvement upon the limitations of FAT-16 systems, which allowed a maximum size of 4GB for all partitions. In addition to the above, the new file system needed to provide support for RAID disk setups.

**Storage:** NTFS uses a significantly different method of allocating space to files than FAT does. This was mostly due to the inefficient storage management of FAT systems resulting in large amounts of disk space being used as slack.

**Length of File Names:** A limiting factor of FAT-based systems was the inability to name files using more than 11 characters. NTFS allows up to 255 characters to be used in the naming of files.

**Networking capability:** Although networking capability is the standard for all devices employed in the modern age, it was not commonplace when Windows NT was developed. As networking was still in its infancy, alongside bearing in mind that Microsoft wished to build a file system for future compatibility, this was a marked improvement over FAT-based systems.

### 2.1.2 FAT

File Allocation Table (FAT hereinafter) is a file system developed by Microsoft with help from: NCR, SCP, IBM, Compaq, Digital Research, Novell, and Caldera. Its first version, FAT-8, was introduced in 1977 for use on floppy disks. (Minitool, N/A) As the predecessor to NTFS, the FAT file system saw a long history of use from 1977 up to the modern age where it still sees use in lightweight scenarios. FAT is still commonplace on flash and other



solid-state memory cards, as well as being the primary file system in use across USB devices. Furthermore, in addition to being frequently used in portable and embedded devices, it remains the standard file system for digital cameras. (Standard of the Camera & Imaging Products Association, 2010)

Given its widespread use since its inception, in large part due to successive improvements to the file system (increasing maximum number of clusters), it remains an essential box to be ticked regarding compatibility of future developed solutions. The accompanying number to FAT signifies the size of the file allocation table; FAT16 has a 16-bit file allocation table, FAT32 has a 32-bit file allocation table etc.

### 2.1.3 Comparison of NTFS and XFAT

This section will aim to explore the above two file systems and provide an accurate comparison to any readers of this report. Such a comparison is significant due to the XB1 employing NTFS and its predecessor employing XFAT. Given how critical knowledge of the file system is in discovering artefacts potentially pertinent to a forensic investigation, this section will endeavour to cover any relevant information. The comparison will take place across six areas: key structures, storage, file names, directories, file date and time, and lastly file deletion. For reference, the architectures of both the NTFS and the FAT file system can be seen below.

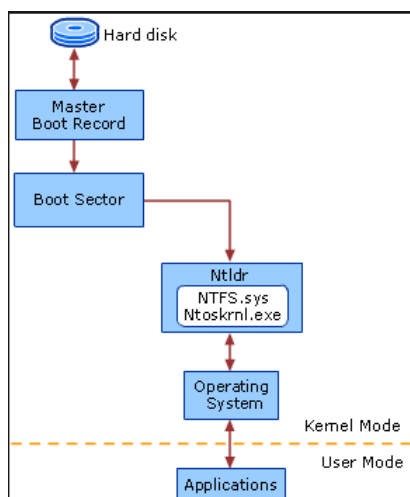


Figure 1: NTFS architecture

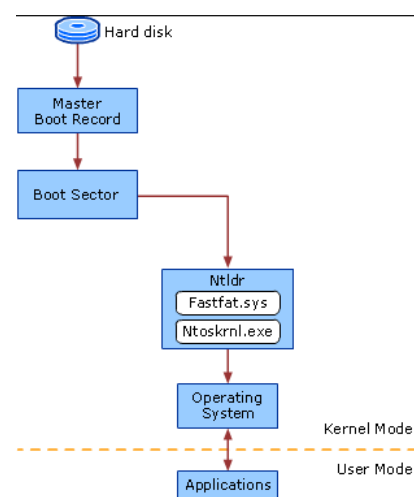


Figure 2: FAT architecture

### 2.1.3.1 Data structures

Responsible for the sorting of files and their respective data into a coherent system are data structures. The structure in NTFS starts with the 512 byte large boot record which contains boot codes, disk signatures (responsible for maintaining and identifying partitions) as well as a table of primary partitions. (Rusbarsky, 2012) Comparing NTFS to FAT, the former uses a journaling file system whereas the latter does not. Such a system guarantees the consistency of internal data structures in case of a system crash, data moved in defragmentation purposes as well as allowing a simple method of rolling back uncommitted changes to said structures upon eventual remounting of a volume. These events are stored in the NTFS log file *\$LogFile*. Put succinctly, it keeps track of system changes by use of a logfile. (LSoft Technologies Inc, 2019)

The most essential element of NTFS is the master file table (MFT hereinafter). The first 16 records are reserved for MFT data files. In regard to naming convention, all files that start with “\$” are metadata files stored in the MFT. The remainder of the records are used for files and folders. Files stored in said remaining records such as \$MftMirr which contains a duplicate of the first four records of \$Mft. (LSoft Technologies Inc, 2019)

Worthy of import is the distinction that NTFS treats every entry as a file, thus the MFT contains an entry from every file and directory. All MFT entries share the same structure, namely that the first 42 bytes are designated for data (acting as a header) with the remainder housing attributes. Attributes contain the actual file data and are of particular forensic relevance as unused space at the end of the entry, also known as *slack*, can be used to conceal data. All attributes are organised into a B-tree structure, with records pointing to clusters potentially containing further files. An image of a standard MFT entry can be seen below.

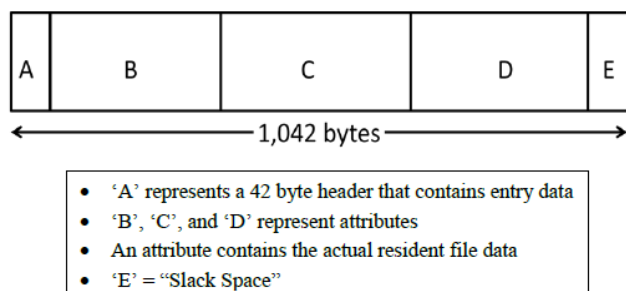


Figure 3: MFT entry (Rusbarsky, 2012)

In relation to the above, the FAT file system can be considered simplistic. Two main data structures are present in FAT, a file allocation table as well as directory entries produce a binary tree. (Microsoft, 2009) In such an organisational structure, a file listing can only be generated after the file system scans a large folder. Every file and folder is assigned a directory entry which are stored in a cluster. Were data to reach beyond one cluster, the file access table would need to be accessed to locate the remainder of the data.

### 2.1.3.2 Storage mechanisms

It is assumed any readers of this report are aware of the structure of hard drives and thusly, an intricate explanation is beyond the scope of this essay. However, an image of the hard disk structure can be seen to the right.

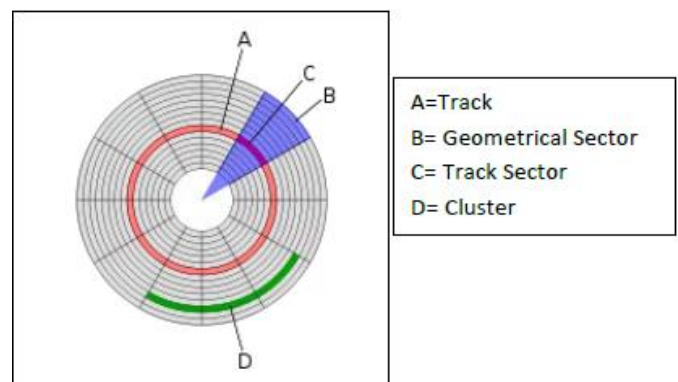


Figure 4: Hard disk model (Wikipedia, 2015)

Upon assignation of data to a cluster, NTFS will use the \$Bitmap MFT metadata file to find the first available cluster. (LSoft Technologies Inc, 2019) In the cluster that is found by the afore-mentioned file, NTFS always assigns data to sector zero. Furthermore, another MFT metadata file, \$BadClus, is responsible for keeping a list that identifies any bad clusters fully updated. These bad clusters will be skipped when searching for empty space.

In contrast, FAT only allows data to be recorded after a set number of reserved sectors and FAT areas. (Microsoft, 2009) The reserved area starts at sector zero. Each FAT area contains FAT structures, wherein which two copies are stored in the file system. Each cluster is allocated an entry in the FAT structure. The file system assigns varying values to table entries dictating the health of the cluster, or simply whether it should be used for allocation. Moreover, FAT cluster addresses never begin at the start of the file system and require sector addresses to be identified. (Microsoft, 2009) Due to this less efficient method of accessing clusters, data size does not always match the cluster size; this allows for the possibility of extra sectors existing at the end of a data area that are neither part of the cluster or of slack space. Such areas are of significant forensic interest as data can be purposely obfuscated or files hidden.

#### *2.1.3.3 File naming conventions and directories*

NTFS stores file names in the MFT file record at which point it becomes known as a resident attribute. Of particular relevance to digital forensic investigations is the \$Boot metadata file structure which has information pertaining to volume size, clusters as well as the MFT itself. Said metadata file structure has the following attributes: (Rusbarsky, 2012)

- \$STANDARD\_INFORMATION
- \$FILE\_NAME
- \$DATA

The first attribute holds all core metadata for a file or directory, the second contains the parent directory file reference, and the last attribute is employed to store data. These attributes relate closely to MFT entry records, as is signified by their '\$' prefix, by encapsulating metadata that an MFT entry holds. (Alazab, et al., 2009)

In contrast, FAT saves data under an 8.3 file name, as well as a long file name. The former is a compressed version of the latter which MS-DOS uses in order to locate and access a file.

Summarised, NTFS allows a forensic examiner more options for data recovery through MFT metadata files and attributes whereas FAT puts the examiner at a disadvantage. Even were the FAT table fully accessible, difficulties could be encountered in reconstructing the file.

#### *2.1.3.4 File date/time*

The primary comparison that can be drawn between NTFS and FAT regarding time/date stamps is that NTFS stores times in UTC, whereas FAT stores it in the computer's local time. This is highly relevant to a forensic investigation as if the target uses NTFS as a file system, the examiner would not be required to check the computer's local time to confirm timestamps as accurate. Given that the XB1 uses NTFS, this removes a step otherwise necessary.

Regarding time/date stamp, NTFS has four such attributes: creation time, modified time, MFT entry modified time, and accessed time. (Rusbarsky, 2012) The first shows the time at which the file was created. The second shows the last time at which attribute content of either \$DATA or \$INDEX was last modified. The third shows the time at which the metadata of the file was last modified. The final attribute shows the time at which the file was last accessed. (Where is my data, n.d.)

The three highlighted MFT attributes are shown to be of further significance as they contain pertinent information regarding time and/or date. The \$STANDARD\_INFORMATION attribute houses the primary set date/time stamps with \$FILE\_NAME also recording a set of timestamps, albeit timestamps that correlate to file creation, renaming or moving. Lastly, \$DATA has no defined values. (Microsoft, 2018)

In contrast, FAT has three date/time stamps that are used by a directory entry: last accessed, last written, and created. Worthy of note is the fact that time values in FAT are not essential and could sometimes be inaccurate. Upon creation of a directory entry for a new file, a new date/time stamp is created which will remain unchanged even were a copy to be moved to a different directory. In addition to the above, if a file were to be moved or renamed, the date/time stamp will also remain. However, if the file is moved from a W2000/XP system to a different volume, a new creation time will be generated upon the addition of new content to the file.

#### *2.1.3.5 File deletion*

Regarding file deletion, both systems operate similarly. Upon deletion of a file in NTFS, a special indicator file is unmarked, indicating to the OS that the location the file used to reside can have another file reallocated to it. The data remains there until overwritten.

In contrast, FAT replaces the first character of the directory entry with a HEX E5h special character entry, thus instructing the OS that the file can be ignored.

## *2.2 Running modified software on the Xbox One*

The above area is challenging to explore given that no available official accounts exist discussing the subject matter. This can be deemed logical, as were such information to be publicly available, proprietary information would have been revealed. Hence, any information contained within this section was retrieved from unofficial sources such as various forums created and operated as a centralised location for enthusiasts to aggregate their knowledge.

One primary reason as to why the XB1 is unable to be modified is due to the simple fact that unsigned code is prohibited from being run. Therefore, even were applications able to be copied to the XB1 (through use of a PC – or any other medium/tool), they would be unable to

be executed. This is due to the way in which applications are divided into sandboxes, thus preventing applications from reading or modifying another's files. (Anon., 2018) Such a setup prevents the hiding of one file in another, then using the hidden application to modify a third application. As an example, this would enable a user to employ software to gain advantages, e.g. aim assist (cheating).

The XB1 uses a virtualised environment wherein which application virtual machines (VMs) and game VMs are run concurrently. (Anon., 2018) This means that even were a malicious user to gain unauthorised access to one VM, the other would remain inaccessible; primarily this is accomplished by use of a Hyper-V hypervisor. (Anon., 2018) No way to exploit memory in the form of buffer overflow attacks have been identified by members of the modding community that would have allowed one application to rewrite another's memory. Given that Windows employs Data Execution Prevention (DEP), the modding community surmises it also to be in place on the XB1 and reason as to why buffer overflow attacks are prevented. (Anon., 2018) DEP functions by randomising the location of data within RAM, therefore even were users to overcome the afore-mentioned countermeasure, it would result in an application or OS crash.

## 2.3 Related work

Xbox 360: A digital forensic investigation of the hard disk drive, Xynos et al.

Xynos et al. focused their investigation on the hard disk drive of an Xbox 360, in much the same way this report used hard disk drives as the basis for forensic evidence. Through the performed examination, it was found that the inspiration for the partition-based layout of the XB1 can be seen in the XB360 that used four partitions to the XB1's five. Even though the XB360 was of the 7<sup>th</sup> generation of gaming consoles, the field of digital forensic analysis had yet to identify gaming consoles as a relevant application and thus, a lack of applicable research within said area was easily noticeable.

The file system the XB360 used was based on older implementations of FAT and was known as XFAT. Given that NTFS was added in Windows NT in 1993 and used thereafter in all releases of Windows, the choice of a specifically adapted version of its preceding file system was curious as it limited the possibly interaction between the XB360 and the user's PC. It is the opinion of this report's writers that this was due to the change in how gaming consoles

evolved from simply being gaming consoles to complete multimedia platforms. Said evolution and its resulting improvement regarding inter-operability to PC machines, heightened the necessity of guaranteeing security and verification of executed code's integrity.

The above-described shift in perspective between Microsoft's 7<sup>th</sup> generation XB360 and the 8<sup>th</sup> generation XB1 required addressing the issue of users modifying their console to run unsigned code. Xynos et al. stated that "despite Microsoft's protection mechanisms, hacking communities have already found code exploits/loopholes, which allow the Xbox 360 to execute unsigned code and software." (Xynos, et al., 2010) Through careful selection of firmware version, users were able to install Linux, thus allowing the console to run as any PC running Linux would. Given that the XB1 has yet to be modified, it can be conclusively stated that a significant improvement in security features dealing with the prevention of OS modification and/or the execution of unsigned code was present in the 8<sup>th</sup> generation console.

The authors' choice of methodology was limited due to the variance in file systems. Xynos et al. Said limitation took the form of the authors only being able to analyse the gaming console through examination of HDD images alone, rather than any possibility – or necessity - of using a logical view. However, this limitation's impact was substantially reduced given that security implementations were weaker on the XB360, allowing for files to be able to be analysed by a forensic imaging software which remains in stark contrast to this report's inconclusive results and analysis produced resulting from the use of Autopsy for analysis of the created HDD images.

Lastly, similarities in methodology were also present. Imaging points were established after actions were taken to help determine causality. Xynos et al. also used an XB360 never-before run to function as their control and to allow for comparison.

### 3. Method

This report will be based on theoretical qualitative sources alongside quantitative data with the intention of answering the research questions. The primary method that will be used is a literature analysis coupled with a secondary method of performing an experiment to explore what information can be extracted from an XB1 pre- and post-modification of the console. All sources will be employed in relation to their importance regarding said source's aid in answering the research questions.

Literature analysis is composed of the following sections:

- Literature review of:
  - o Scholarly articles
  - o Journal publications
  - o Important documents
- Document analysis
- Research methodology

Literature reviews form a central part of any thesis and thus will not be covered in further detail than what already described above.

Document analysis is the process of examining documents, as well as the data contained therein, through a different lens to provide a different perspective on subject matter indirectly related to the research question of the original document. Its goal within this report will be to provide useful, and relevant, comparisons as well as offer insight into what subject matter was deemed out-of-scope for other scholarly publications. Such limitations in scope provide avenues for new research and new questions to be answered.

The research methodology will be dictated by the above sections, namely finding relevant, useful and scholarly sources to provide readers of this report with the context and understanding needed to comprehend the information aimed at answering the research questions.



The second method this report will use is an experiment. The primary section of the experiment revolves around performing actions on the XB1 followed by acquiring a forensically sound image of its HDD. The secondary aspect of the experiment focuses on determining the extent to which data can be hidden on the console. The experiment's process will be explored in further detail in the section entitled *Experiment arrangement*; however, a summary of the process will be seen at the end of this section. It will be performed using a before-and-after evaluation for which a measurement will be taken before, and after, changes have been introduced to the scenario. Such an evaluation process produces better evidence regarding causality. This means the ability to determine whether a modification of a variable had an impact on the comparative metric used will be possible.

In this experiment, the comparative metric used will be a hash of the hard disk image. This hash will be taken after each stage of the experiment and will be used to determine whether data integrity has been compromised. However, worthy of note is that said hashes will simply provide the impetus to perform forensic analysis on the image and a change in hash values is expected.

For an increased ability to determine causality, the experiment will be split into phases. These phases will each produce an image (and related hash) that will be analysed during a later stage. Each phase will be designed in such a way that any new artefacts discovered will be analysed to determine whether said artefacts occurred as a direct result of the new actions taken. A difference in hash values means the images are no longer identical allowing for further analysis.

The first phase will be the most basic with the aim of removing the hard drive from the case and creating a forensic image. As per digital forensics procedures, a hash will be created of the image file. This image will function as the control.

The second phase concerns itself with generating artefacts related to the sign-in procedure as well as the installation and launching of games, alongside playing in both single and multiplayer game modes.

The third phase will focus on generating artefacts related to various apps available such as: Skype, YouTube, Twitch and Web Browser. After installation and prior to completing the imaging process (including a hash of the drive), each app will be opened and utilised for a

pre-determined length of time. This stage will also deal with generating network-related artefacts through taking the below actions:

- User signs out and signs in to his/her profile
- YouTube: opened, searches performed, videos watched
- Skype: opened, call begun/ended
- Internet Explorer (IE hereinafter), websites visited

### 3.1 Tools

In order to successfully perform an exploratory analysis of the XB1, various software will need to be utilised. This report will employ the below listed tools and provide justification for each one.

In order to image the target drive, AccessData's *FTK Imager* will be used.

For examination of the image, *Autopsy* will be used. This program is open-source, highly modular, and heavily used across the digital forensics field. (Autopsy Digital Forensics, 2020) Although possessing a plethora of features, its keyword search will be used in conjunction with a list of likely matches. Said list will be displayed in a later section.

Thirdly, SleuthKit's *Scalpel* will be used in order to carve files. File carving is the process of retrieving files from a raw disk image after said files' deletion regardless of the target image's filesystem. (Richard & Roussev, 2005) The chosen file carver is frugal – allowing for its operation on low-end machines – and high performance – allowing for high speed file carving.

Although several tools are available, this investigation will employ *bulk\_extractor* – a highly reputable forensic tool capable of not missing data in unallocated regions as well as processing any form of media. (Garfinkel, 2013) Succinctly summarised, *bulk\_extractor* extracts data without contact with the file system ensuring the usefulness of the data.

Lastly, in order to calculate file entropy to estimate the extent of a file's encryption and/or compression the system internal (sysinternal) tool known as *sigchecker* will be used. All system internal tools are lightweight in nature. Example output can be seen to the right.

```
Sigcheck v2.73 - File version and signature viewer
Copyright (C) 2004-2019 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\users\vikk\desktop\R1_running-config.txt:
Verified:      Unsigned
File date:     23:15 14/03/2019
Publisher:     n/a
Company:       n/a
Description:   n/a
Product:       n/a
Prod version:  n/a
File version:  n/a
MachineType:   n/a
Binary Version: n/a
Original Name: n/a
Internal Name: n/a
Copyright:     n/a
Comments:      n/a
Entropy:       4.978
```

Figure 5: Example output of sigchecker

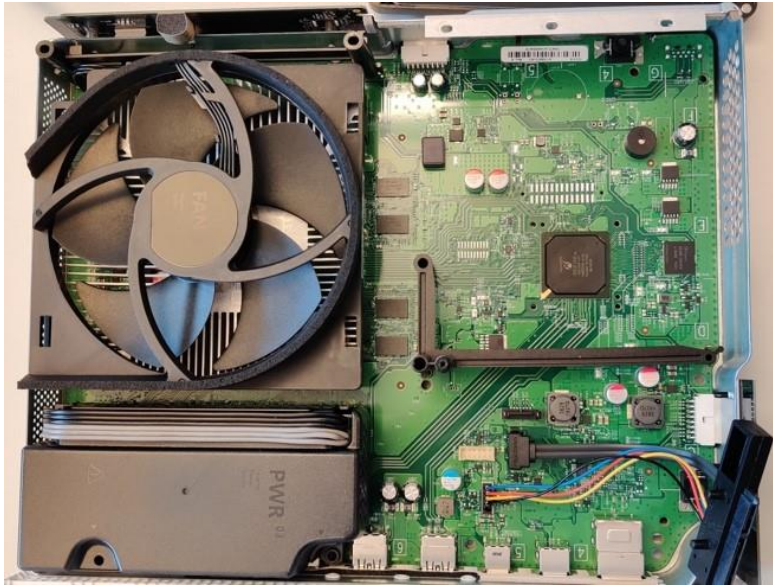
### 3.2 Physical tools

To open the Xbox One S and get the hard drive out several physical tools were needed. We used a plastic small prying tool to get the plastic bottom cover of the XB1 off. On the underside of the console there were six green 50 mm T10 torx screws, these were removed using a T10 torx screwdriver. We then needed a T8 torx screwdriver to remove 3 screws securing a metal cover on the top of the console. With this cover removed we had access to the inside of the console. To remove the 2.5" 1TB hard drive first two T10 torx screws and then four T10 torx screws had to be removed.

With the hard drive free, we could connect it to a computer for analysis. This was done using a Deltaco SATA to USB adapter.



Figure 6: Deltaco SATA to USB adapter connected to the XB1S hard drive



*Figure 7: The innards of the Xbox One S*

## 4. Experiment arrangement

1. Disconnect HDD from XB1.
2. Ensure SAFE Block is running on forensic workstation with the settings shown in the corresponding image.
3. Connect XB1 HDD to SATA USB adapter and connect HDD to forensic workstation.
4. Open FTK Imager. This experiment employs v. 4.3.0.18.
5. Add XB1 HDD as evidence item.
6. Right-click the added evidence item, select "Export Disk Image".
7. Ensure "Verify images after they are created" is checked. For the sake of time management, "Precalculate progress statistics" was also checked.
8. Select the target storage device.
9. Upon successful completion of the imaging and verification processes, a screenshot is taken for documentation purposes.
10. Re-connect HDD to XB1 and power on the console.
11. Perform actions described in phase 2 of the method.
12. Repeat steps 3 through 11.
13. Perform actions described in phase 3 of the method.
14. Repeat steps 3 through 11.
15. Termination of experiment.

## 5. Results

Autopsy was used in order to provide a first glance view at the taken images. Given that this experiment employed a 5400 rpm 2TB external HDD, alongside the size of the images being of significant size, ~903GB, the choice of Autopsy ingest modules to employ was purposely limited to allow for operations to be completed within a reasonable time limit.

Worthy of note is that prior to this decision being made, certain modules' inclusion (e.g. encryption detection) led to each case halting progress at 0% and staying at said point for up to 15 hours. At said point, other avenues of detecting encryption were explored and the ingest module unselected.

### 5.1 Autopsy analysis

#### 5.1.1 First Image

In the interest of avoiding confusion to all readers of this report, it needs to be noted that the below section is a comparison and thus, was written after the succeeding two sections.

Figure 7 shows the chosen ingest modules for the Autopsy case. Given that other methods will be used to ascertain whether the XB1 is encrypted, as well as the reason stated above under this section's introduction, encryption detection was not selected. Moreover, hash lookup was not employed as the primary use of hashes in this report is to determine that disk images are separate and in hopes of determining causality.

The first image will function as the control and as relevant point of comparison. The XB1 was not booted prior to having its HDD removed and imaged, allowing for a fair standard for comparison. In this report, the emphasis will be placed on the volume of files that differ between the keyword search matches of the second, and third images, with the control.

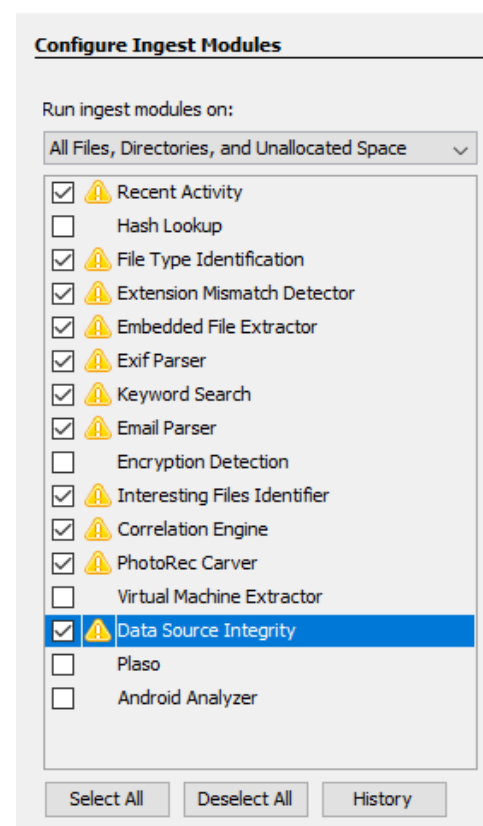


Figure 8: Selected ingest modules

Structurally, all images share the same structure. Similarly to the second and third images, all carved files shared the extension mft. As can be seen below, more files were reported as carved by Autopsy on the first than on either the second, or the third, image.

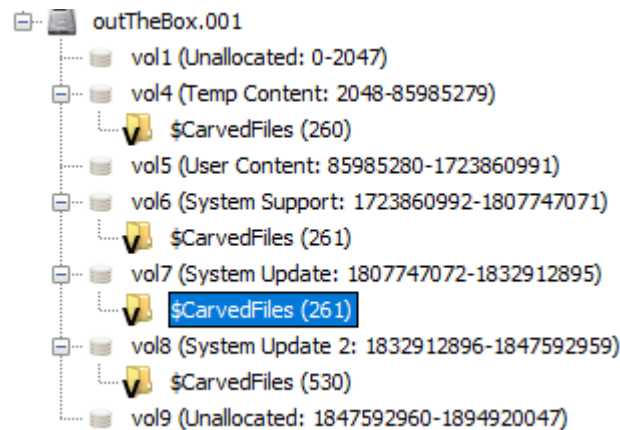


Figure 9: Structure & carved files

The two volumes with differing amounts of carved files were volume 7 and volume 8. The tables showing the carved files from volume 7 were exported and compared. The file that differed was not identified; however, it can be noted that the two tables were identical up to the fifth row. It remains difficult to conclude whether these results are of value as through manual observation, it was noted that the two tables contained an overwhelming large amount of identically named files yet were on different rows. This could be the cause for csv comparison not being successful.

Regarding volume 8, it was determined upon manual observation that this was due to each file having a duplicate. The remaining additional 10 files were unable to be identified and were assumed to be additional copies.

To further a useful comparison, the same search terms were used as those used for both second and third images. None of the searches returned results, demonstrating how artefacts are created in relation to the installation and running of apps and games. Critically, given that no personally identifiable information was found through the search terms on any images, these results strongly indicate that artefacts are indeed created but that no personally identifiable information is stored unencrypted or at the same location.

## 5.1.2 Second Image

The second image's general structure can be seen below.

Name	ID	Starting Sector	Length in Sectors	Description	Flags
vol1 (Unallocated: 0-2047)	1	0	2048	Unallocated	Unallocated
vol4 (Temp Content: 2048-85985279)	4	2048	85983232	Temp Content	Allocated
vol5 (User Content: 85985280-1723860991)	5	85985280	1637875712	User Content	Allocated
vol6 (System Support: 1723860992-1807747071)	6	1723860992	83886080	System Support	Allocated
vol7 (System Update: 1807747072-1832912895)	7	1807747072	25165824	System Update	Allocated
vol8 (System Update 2: 1832912896-1847592959)	8	1832912896	14680064	System Update 2	Allocated
vol9 (Unallocated: 1847592960-1894920047)	9	1847592960	47327088	Unallocated	Unallocated

Figure 10: Disk structure

The below image shows the number of files Autopsy was able to carve from each of the volumes on the disk image.

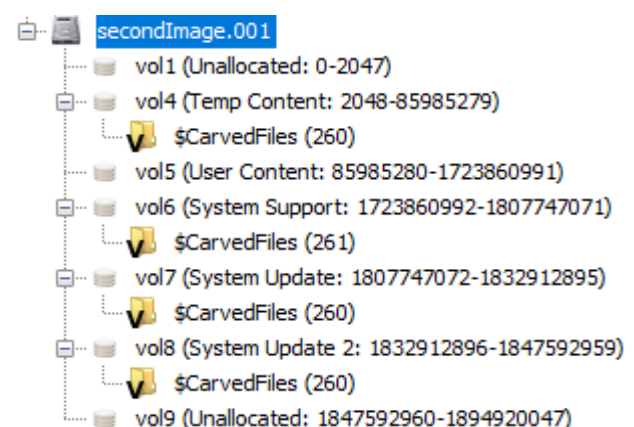


Figure 11: Number of carved files

Although Autopsy reported carved files, said files did not provide any strictly relevant information. All files carved had the extension .mft, lacked timestamps, the same size (1024 bytes) and eight characters. Each file name starts with “f”. An example can be seen below:

Name	S	C	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(Meta)	Known	Location
f0000016.mft			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1024	Unallocated	Unallocated	unknown	/img_secondImage.001/vol_vol4/\$CarvedFiles/f0000016.mft
f0000018.mft			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1024	Unallocated	Unallocated	unknown	/img_secondImage.001/vol_vol4/\$CarvedFiles/f0000018.mft
f0000020.mft			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1024	Unallocated	Unallocated	unknown	/img_secondImage.001/vol_vol4/\$CarvedFiles/f0000020.mft
f0000022.mft			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1024	Unallocated	Unallocated	unknown	/img_secondImage.001/vol_vol4/\$CarvedFiles/f0000022.mft
f0133992.mft			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1024	Unallocated	Unallocated	unknown	/img_secondImage.001/vol_vol4/\$CarvedFiles/f0133992.mft
f0133994.mft			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1024	Unallocated	Unallocated	unknown	/img_secondImage.001/vol_vol4/\$CarvedFiles/f0133994.mft
f0133996.mft			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1024	Unallocated	Unallocated	unknown	/img_secondImage.001/vol_vol4/\$CarvedFiles/f0133996.mft
f0133998.mft			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1024	Unallocated	Unallocated	unknown	/img_secondImage.001/vol_vol4/\$CarvedFiles/f0133998.mft
f0134000.mft			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1024	Unallocated	Unallocated	unknown	/img_secondImage.001/vol_vol4/\$CarvedFiles/f0134000.mft

Figure 12: Files carved by Autopsy

The results gained from Autopsy are limited; this is logical given that encryption assumed to be employed on the XB1. As with the prior two images, no file sizes were successfully determined by Autopsy, timestamps were all zeroed and no valid email addresses were identified. Moreover, no data was considered “Extracted Content” by Autopsy.



As with the first image, Autopsy was used in order to perform keyword searches related to the examiners' names, games played and searching for both the username and email employed for the XB1 sign-in procedure. The below terms were searched for as substrings; the bolded list items returned results.

- **Roblox**
- **Victor**
- Forza
- Robbin
- hotmail

#### 5.1.2.1 "Roblox" keyword search

The search for "Roblox" returned matches in a file Autopsy names "Unalloc\_5\_44024463360\_882616827904". The full path of the file is: "/img\_secondImage.001/vol\_vol5/Unalloc\_5\_44024463360\_882616827904".

Below is a screenshot taken of the metadata tab on the above file.

Name	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_882616827904
Type	Unallocated Blocks
MIME Type	application/octet-stream
Size	838592364544
File Name Allocation	Unallocated
Metadata Allocation	Unallocated
Modified	0000-00-00 00:00:00
Accessed	0000-00-00 00:00:00
Created	0000-00-00 00:00:00
Changed	0000-00-00 00:00:00
MD5	Not calculated
Hash Lookup Results	UNKNOWN
Internal ID	12

Figure 13: Metadata tab

The context around which the keyword was successfully found can be seen below:

```
[XBL:]http://assets1.xboxlive.com/1/4fecf8f0-4ba8-41dd-a25e-
a8a07955eadc/f321baf7-f79c-455f-a87b-cf3cb4bc48f2/1.426.39.7176.1f60521e-
28eb-4cda-81c1-67a6ffdc0306/
roblox_1.426.39.7176_x64_6nk717fhhb5z8t
[XBL:]http://assets2.xboxlive.com/1/4fecf8f0-4ba8-41dd-a25e-
a8a07955eadc/f321baf7-f79c-455f-a87b-cf3cb4bc48f2/1.426.39.7176.1f60521e-
28eb-4cda-81c1-67a6ffdc0306/
roblox_1.426.39.7176_x64_6nk717fhhb5z8t
```

```
b[XBL:]http://d1.xboxlive.com/1/4fecf8f0-4ba8-41dd-a25e-
a8a07955eadc/f321baf7-f79c-455f-a87b-cf3cb4bc48f2/1.426.39.7176.1f60521e-
28eb-4cda-81c1-67a6ffdc0306/
roblox_1.426.39.7176_x64_6nk717fhhb5z8t
[XBL:]http://d2.xboxlive.com/1/4fecf8f0-4ba8-41dd-a25e-
a8a07955eadc/f321baf7-f79c-455f-a87b-cf3cb4bc48f2/1.426.39.7176.1f60521e-
28eb-4cda-81c1-67a6ffdc0306/
roblox_1.426.39.7176_x64_6nk717fhhb5z8t
```

Although an overwhelming amount of seemingly encrypted information being available in the file, there was also some human-readable text in the above-mentioned file unrelated to the search term. This can be seen below:

```
This event starts tomorrow
This event starts in {0}
This event starts in
This event starts
This event is now live.This event is now
Third Party NoticesTerms of Use
Microsoft Software License Terms
Show more
Show all {0} in Store
Show all
Share eventEvents Settings Page
Settings
Microsoft Services Agreement
{0} seconds{0} second
Press A to view image in fullscreen mode.
Press B to exit fullscreen mode.
Average star rating of {0} star.
Average star rating of {0} stars.
{0} review
{0} reviews{0} Event
Microsoft Privacy StatementOriginal price {0}
on sale for {0}Price: {0}
{0} event {1} starts {2}
{0} event {1} started on {2}
{0} people interested
{0} PEOPLE
You don't have any upcoming events yet.Go back to the Events list and
choose some you might be interested in.
My upcoming events
```

My past events  
 My Events  
 My Events Page  
 My active events  
 More events{0} minutes{0} minute  
 {0}M  
 Manage your notifications  
 Manage notifications  
 LIVE  
 LIVE NOW  
 Learn more  
 Launch game{0} of {1}  
 I'm interested!{0} hours  
 {0} hour  
 Home Page  
 Help  
 Go to official clubGamertag {0}  
 Gamerscore {0}  
 {0} friends interested  
 {0} friend interested  
 {0} to {1}  
 Your event has started!Starts {0}  
 Started on {0}  
 Events SettingsEvents  
 Events Home Page  
 Events (Beta)  
 The Events (Beta) App provides you with information on the in-game events going on in the games you love, all in one place. The Events app is currently in beta and will add new features and experiences in the coming months.Events Ending Soon  
 EVENT ENDEDEvent endedYou are interested in this event.  
 Retry  
 You can retry below, or try back later.We ran into a problem getting your content.{0} event {1} started on {2} and ended on {3}  
 Stay tuned! New events are added all the time.No events to show right now.  
 {0}:{1}:{2}{0} hours, {1} minutes, {2} second  
 {0} hours, {1} minutes, {2} seconds{0} hours, {1} minute, {2} second  
 {0} hours, {1} minute, {2} seconds  
 {0} hour, {1} minutes, {2} second  
 {0} hour, {1} minutes, {2} seconds  
 {0} hour, {1} minute, {2} second

{0} hour, {1} minute, {2} seconds

Close

The above results can be considered unimportant when juxtaposed to the aim of this report; however, it does demonstrate that not all information contained on an XB1 is encrypted or non-human readable.

### 5.1.1.3 "Victor" keyword search

Several matches were found in the same described file as can be seen below:

Source File	S	... Keyword	Keyword Regular Expression	... Modified Time	Access Time	Change Time	File Path
Unalloc_5_44024463360_882616827904		motociclistvictorian	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		victoriach	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		victoriaanszwmvinnen	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		lehengavictorian	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		victori	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		bikervictoriennepalmes	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		victoria	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		victoriarrahegalakmandarina	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		victorian	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		victorianskv	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...
Unalloc_5_44024463360_882616827904		victoriano	victor	... 0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	/img_secondImage.001/vol_vol5/Unalloc_5_44024463360_...

Figure 14: "Victor" keyword search

Essential to note is the keyword column. Due to the search being performed on a substring, none of the matches related to one of this group's authors. Thus, the relevance of these results can be estimated to be low. Curiously, many of the files were in different languages and seemed to describe various aspects that could be applicable to a digital avatar. Examples are shown below:

Short sleeve untucked  
Short sleeve tuckedShirt and tie  
Collarless  
Long sleeve untucked  
Bikini top  
Long sleeve crop top

While browsing the file with Dutch content, the following text was found, providing strong evidence for the contents of this file pertaining to a digital avatar.

De Avatar Store kan niet worden geopend

Translated to English, the above phrase reads: “The Avatar Store cannot be opened”. Such language-specific content indicates that space is taken up in order to allow for a smoother installation and activation of the XB1.

### 5.1.3 Third image

The general structure of the third image is identical to that of the second image and can be seen below:

△ Name	ID	Starting Sector	Length in Sectors	Description	Flags
vol1 (Unallocated: 0-2047)	1	0	2048	Unallocated	Unallocated
vol4 (Temp Content: 2048-85985279)	4	2048	85983232	Temp Content	Allocated
vol5 (User Content: 85985280-1723860991)	5	85985280	1637875712	User Content	Allocated
vol6 (System Support: 1723860992-1807747071)	6	1723860992	83886080	System Support	Allocated
vol7 (System Update: 1807747072-1832912895)	7	1807747072	25165824	System Update	Allocated
vol8 (System Update 2: 1832912896-1847592959)	8	1832912896	14680064	System Update 2	Allocated
vol9 (Unallocated: 1847592960-1894920047)	9	1847592960	47327088	Unallocated	Unallocated

Figure 15: Disk structure

The below image shows the number of files Autopsy was able to carve from each of the volumes on the disk image.

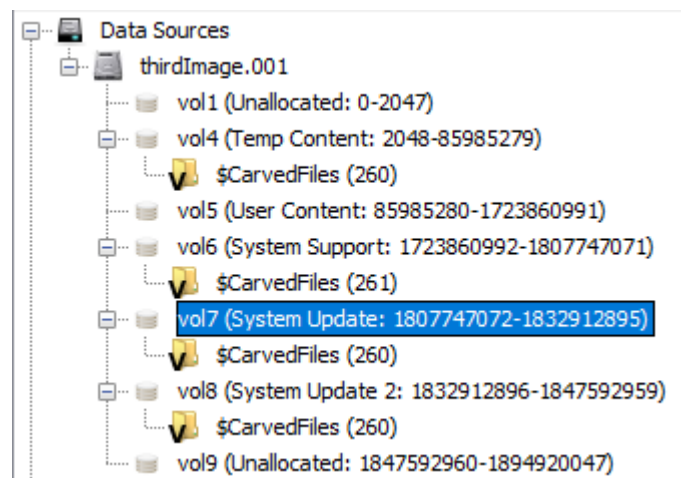


Figure 16: Files carved by Autopsy

The results gained from Autopsy are limited; this is logical given that encryption assumed to be employed on the XB1. As with the prior two images, no file sizes were successfully determined by Autopsy, timestamps were all zeroed, and no valid email addresses were identified. Moreover, no data was considered “Extracted Content” by Autopsy.

### 5.1.3.1 “Skype” keyword search

The “Skype” substring keyword search returned 587 matches. Many files contained text that was not human-readable, possibly due to encryption. An example of such text can be seen to the right.

A#(w  
GMLC4  
XX/  
EAIv  
9x^I  
\_wPZ  
HMsI  
2a4E  
e#`U  
EBX=

Figure 17: Non human-readable text found in substring match

Given the large volume of files, a detailed examination of each match was not possible. However, through manual browsing of various substring matches, a pattern was identified. Namely, that many files contained *xml* code detailing app extensibility through use of the UAP protocol. Examples of this will be shown further below.

Upon further investigation, UAP was characterised as a UDP-based application layer protocol. (Lv, et al., 2011) Authors Lv, Xu, Su and Chen state UAP’s primary goal is to “meet the needs of the applications to have all the following merits: reliability, real time, fairness, file-transport, P2P-friendliness and TCP-friendliness.” (Lv, et al., 2011) Many of the above strengths are well suited to Skype: reliability would ensure constant call quality; calls function in real time; lastly, Skype is a peer-to-peer (P2P) VoIP client. (Baset & Schulzrinne, 2005)

The start of the XML portion of the file whose keyword was “skypesetup” can be seen below, with its file metadata below that.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Package IgnorableNamespaces="uap mp rescap uap4 build" xmlns="http://schemas.microsoft.com/appx/manifest
lns:rescap="http://schemas.microsoft.com/appx/manifest/foundation/windows10/restrictedcapabilities" xmlns
<Identity Name="Microsoft.SkypeApp" Publisher="CN=Skype Software Sarl, O=Microsoft Corporation, L=Luxem
<mp:PhoneIdentity PhoneProductId="c3f8e570-68b3-4d6a-bd8b-c0a3f4360a51" PhonePublisherId="89188381-c4b
<Properties>
  <DisplayName>ms-resource://Microsoft.SkypeApp/Resources/SkypeVideo_ProductName</DisplayName>
  <PublisherDisplayName>Skype</PublisherDisplayName>
  <Logo>SkypeApp\Assets\SkypeLogo.png</Logo>
</Properties>
<Dependencies>
  <TargetDeviceFamily Name="Windows.Xbox" MinVersion="10.0.14393.0" MaxVersionTested="10.0.16299.0" />
  <PackageDependency Name="Microsoft.NET.Native.Framework.1.6" MinVersion="1.6.24903.0" Publisher="CN=
  <PackageDependency Name="Microsoft.NET.Native.Runtime.1.6" MinVersion="1.6.24903.0" Publisher="CN=Mi
  <PackageDependency Name="Microsoft.VCLibs.140.00" MinVersion="14.0.25426.0" Publisher="CN=Microsoft
</Dependencies>
```

Name	/img_thirdImage.001/vol_vol5/Unalloc_5_44024463360_882616827904
Type	Unallocated Blocks
MIME Type	application/octet-stream
Size	838592364544
File Name Allocation	Unallocated
Metadata Allocation	Unallocated
Modified	0000-00-00 00:00:00
Accessed	0000-00-00 00:00:00
Created	0000-00-00 00:00:00
Changed	0000-00-00 00:00:00
MD5	Not calculated
Hash Lookup Results	UNKNOWN

Figure 18: Start of XML and file metadata

The context “skypesetup” was found in is shown below. Note the syntax dictating the use of the UAP protocol.

```
<uap:Extension Category="windows.protocol">
  <uap:Protocol Name="skypeuwp" />
</uap:Extension>
<uap:Extension Category="windows.protocol">
  <uap:Protocol Name="skypesetup" />
</uap:Extension>
<uap:Extension Category="windows.protocol">
  <uap:Protocol Name="skypesettings" />
</uap:Extension>
<uap:Extension Category="windows.protocol">
  <uap:Protocol Name="skypepage" />
```

Figure 19: “skypesetup” substring match

### 5.1.3.2 “Youtube” keyword search

The “youtube” keyword search returned only 13 results. This was surprising given that “skype” returned so many. The returned results can be seen below:

Source File	S	C	Keyword	Keyword Regular Expression	Keyword Preview	Modified Time	Access Time	Change Time
Unalloc_5_44024463360_882616827904	▼		googleinc.youtube	youtube	> <identity name="googleinc.youtube" publisher="cn=...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		googleinc.youtube_1.2.97.70_x64_yfg5n0ztvsklop	youtube	n0ztvsklop9fh( h \<googleinc.youtube_1.2.97.70_x64_y...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		googleinc.youtube_1.2.97.70_x64_yfg5n0ztvsklop9f	youtube	\$sdh( h rcrd( \<googleinc.youtube_1.2.97.70_x64_...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		googleinc.youtube_1.2.97.70_x64_yfg5n0ztvsklop9fh	youtube	h rcrd( 5gh( h \<googleinc.youtube_1.2.97.70_x64_...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		googleinc.youtube_yfg5n0ztvsklop.uwa	youtube	( (8' 8@#i30nfs#googleinc.youtube_yfg5n0ztvsklop...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		www.youtube.com	youtube	entrypoint="https://www.youtube.com/rtv"> <uap:vis	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		youtube	youtube	s> <displayname>youtube</displayname> <	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		youtube_logo.s	youtube	logo.scale-200.pngyoutube_logo.scale-200.pngfile0 h	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		youtube_logo.scale	youtube	e-200.png( h(8' 8youtube_logo.scale-200.pngfile0 ...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		youtube_splash_screen.html	youtube	appx_appx_( h 4<youtube_splash_screen.html( h(8' 8...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		youtube_splash_screen.htmlh	youtube	cache(i(i@h@ ( h 4<youtube_splash_screen.htmlhercd( ...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		youtube_splash_screen.htmlp	youtube	en<plash_screen4<youtube_splash_screen.htmlp<4<y...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00
Unalloc_5_44024463360_882616827904	▼		youtube_splash_screen.htmls	youtube	rcrd( g(8' 8( h 4<youtube_splash_screen.htmls( h (...	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00

Figure 20: "youtube" keyword search results

More *xml* code was found in many of the files with human-readable text. An example can be seen below:

```
<Properties>
  <DisplayName>YouTube</DisplayName>
  <PublisherDisplayName>Google Inc</PublisherDisplayName>
  <Logo>Assets\Square44x44Logo.scale-100.png</Logo>
  <uap:SupportedUsers>single</uap:SupportedUsers>
</Properties>
<Dependencies>
  <TargetDeviceFamily Name="Windows.Xbox" MinVersion="10.0.10586.0" MaxVersionTested="10.0.15063.0" />
  <PackageDependency Name="Microsoft.VCLibs.140.00" MinVersion="14.0.24123.0" Publisher="CN=Microsoft Corpora
```

Figure 21: "youtube" keyword match

Interesting to note is that one of the listed dependencies within the *xml* code is that of “TargetDeviceFamily Name=Windows.Xbox”. This suggests the YouTube app was specifically designed for the Xbox platform and shows that checks can be created based off its target platform. This could potentially mitigate threats of software piracy or of OS



modification (modding). As with the “skype” keyword search, the UAP protocol is also present within this *xml* code.

Moreover, like the prior keyword search, many files contained non human-readable text.

#### 5.1.3.3 “Web browser” keyword search

This search returned only one result and was part of a Terms of Service prompt. Surrounding said prompt were a multitude of responses dealing with rote responses by the system.

Examples include:

- Can't play because your system needs an update. Check to see if updates are available.
- Sorry, can't play or download. Please try again in a few minutes.
- Sorry, can't access your account right now. Please try again later.
- Your Xbox account has been suspended.

It can be stated that the above keyword search’s results are disappointing in their scarcity, limited in relevance and do not aid answering any of the research questions.

#### 5.1.3.4 “Twitch” keyword search

The “twitch” keyword substring search returned 208 matches. It should however be noted that were an exact case search performed, simply one case would be found. The contextual surrounding of the keyword search referenced *.layout* and *.timeline* files. As this seemed to deal with design-based elements, the likelihood of personally identifiable information being contained therein was concluded to be severely unlikely. More so than with prior keyword searches, more non-human readable text was found.

## 5.2 Scalpel & bulk\_extractor

Sadly, the above two tools did not provide valuable evidence. This report's authors see this being a result of employed encryption on the console. When Scalpel was employed, files were extracted but were neither able to be opened or viewed suggesting that the carving process was not successful in its entirety. Furthermore, bulk\_extractor provided no data that was human readable.

## 5.2 Parsing the MFT – a logical view

Due to no personally identifiable information being found by Autopsy, as well as by Scalpel and bulk\_extractor, this report's authors deemed a logical view as the next step in the exploratory process. To accomplish this, the XB1 HDD was removed and attached to the forensic workstation with a software write-blocker active.

Due to this report's exploratory nature, a logical view analysis of the target system was not predicted. Combined with limited storage capabilities, a duplicate of the first image was never able to be moved onto a separate HDD for comparative purposes. This led to investigators being unable to logically view the console's HDD at any point prior to the completion of the experiment. However, the open-source program *mft2csv* was used to create csv files containing the parsed MFT data of each image which was then imported into Excel allowing for accurate comparison. Each partition will be examined, with notable files selected for analysis.

### 5.2.1 Partition 1 - Temp Content

The logical view of the "Temp Content" partition can be seen below. It shares its version with the third image taken, i.e. post experiment.

Name	Date modified	Type	Size
System Volume Information	03/04/2020 18:27	File folder	
\$RECYCLE.BIN	03/05/2020 04:08	File folder	
\$sosrst.xvd	01/09/2014 18:00	XVD File	3,164,364 KB
cms.xvd	01/09/2014 18:00	XVD File	1,061,020 KB
appswapfile.xvd	01/09/2014 18:00	XVD File	2,109,584 KB
GDVRIndex.xvd	01/09/2014 18:00	XVD File	103,628 KB
ScreenShots.xvd	01/09/2014 18:00	XVD File	1,061,020 KB
DeploymentSoftwareDistribution.xvd	03/04/2020 18:57	XVD File	3,145,740 KB
temp00	03/04/2020 20:30	File	2,097,164 KB
ConnectedStorage-retail	05/04/2020 19:43	File	9,548,892 KB

Figure 22: Temp content partition

Firstly, it ought to be noted that the last modified timestamp is shared for all but three files and the date of 01/09/2014 coincides with the early stages of the XB1's release.

Secondly, as mentioned earlier in this report, Autopsy was not used to detect encryption. The reasons outlined were to save time and that another approach would be taken to determine whether encryption or compression are in use. This will be accomplished through calculating files' Shannon entropy values. As Matthew Shannon, the techniques creator and source of its moniker, stated in his paper "Forensic Relative Strength Scoring: ASCII and Entropy Scoring" from 2004: "With all things being equal, the probability of discerning valuable textual information from a given data unit rises based on the portion of ASCII to non-ASCII values it contains." (Shannon, 2004) Summarised, entropy is a measure of information density or the extent to which compression is possible. Shannon states thusly that "the more a given unit can be compressed, the lower the Entropy value; the less a given unit can be compressed, the higher the Entropy value." In practice, entropy is an indicator of how varied the number of bits per a given byte is with three to five bits per byte being the range for human readable text formats and seven to eight being the range for highly encrypted or compressed files. (Shannon, 2004)

Selected as noteworthy files were the following: *DeploymentSoftwareDistribution.xvd*, *temp00* and *ConnectedStorage-retail* due to their timestamps providing evidence for them having been modified. Firstly, *temp00* is a file that is not present on the first image and was created shortly after, or in conjunction with, the initial set-up process of the XB1. Due to its name, it is hypothesised to be a temporary file and its direct purpose unknown. Possibly, it

holds data that has yet to be synchronised to the cloud, thus providing a form of redundancy in the case of network disconnection. Due to its naming convention, it is likely more than one temporary file could be created by the system as needed; the convention would logically follow with *temp01*, *temp02* etc. Its entropy value was calculated to be 0.4784, showing that said file is likely neither encrypted nor compressed. Given that encryption is assumed to be in use on the XB1, such a low value indicates a lower likelihood the files contents contain any personally identifiable information. This reinforces the hypothesis that the temporary file's function is to allow for cloud synchronisation of non-personal information as such data would likely not require encryption.

*DeploymentSoftwareDistribution.xvd* was another selected file. This file was originally created on 01/09/2014 18:00 but was overwritten at the start of the second phase of this report's experiment when the XB1 was initially launched and set-up at 03/04/2020 14:57 and showed no changes after the conclusion of the third and final stage of the experiment. This reinforces its change being tied to the original launch of the XB1 prior to the second phase of the experiment's start. Furthermore, based on its name, it could perhaps be surmised that its function is as a reference or check on the availability of required files. Its entropy score was calculated as 0.2543, suggesting files contained within the ".xvd" wrapper could be human-readable if successfully unwrapped. Ultimately, it can be concluded this file has no correlation to the performed actions of the experiment and due to any apparent lack of encryption and/or compression, is unlikely to contain personally identifiable data.

*ConnectedStorage-retail* was recorded as having been created 03/04/2020 16:25. This time matches the start time of the second phase of the experiment. It was modified at 05/04/2020 15:43, which lines up with the shutdown of the console following the experiment's third phase, yet was not directly accessed as shown by its RTime value in the image below:

FN_FileName	FileSizeBytes	SI_CTime	SI_ATime	SI_MTime	SI_RTime
ConnectedStorage-retail	9778065408	03/04/2020 16:25	05/04/2020 15:43	05/04/2020 15:43	03/04/2020 16:25

Figure 23: *ConnectedStorage-retail* RTime

The above file's entropy was calculated and found to be 0.262, indicating a noticeable lack of encryption and/or compression. Its timestamps did not allow for an accurate conclusion to be drawn; however, its name suggests the file contains information related to, or the handling of, any storage devices connected – such as small USB devices. The entropy score matches the file's hypothesised function as external storage related operations do not contain personally

identifiable information, thus not requiring encryption, or is simply smaller in size, thus not requiring heavy compression.

## 5.2.2 Partition 2 – User Content

The logical view of the “User Content” partition can be seen below. It shares its version with the third image taken, i.e. post experiment.

Name	Date modified	Type	Size
PLS	01/09/2014 18:00	File folder	
SharedStorage	01/09/2014 18:00	File folder	
System Volume Information	03/04/2020 18:27	File folder	
\$RECYCLE.BIN	11/04/2020 14:10	File folder	
LastConsole	01/09/2014 18:23	File	1 KB
Microsoft.XboxEvents_8wekyb3d8bbwe.UWA	03/04/2020 19:01	UWA File	100,520 KB
Microsoft.XboxIdentity_8wekyb3d8bbwe.UWA	03/04/2020 19:01	UWA File	281,772 KB
Microsoft.MicrosoftRewardsonXbox_8wekyb3d8bbwe.UWA	03/04/2020 19:02	UWA File	125,800 KB
Microsoft.Avatars_8wekyb3d8bbwe.UWA	03/04/2020 19:02	UWA File	110,396 KB
F321BAF7-F79C-455F-A87B-CF3CB4BC48F2	03/04/2020 19:50	File	314,372 KB
f321baf7-f79c-455f-a87b-cf3cb4bc48f2.22c21d21-1f1b-46b5-916a-a0d690ae...	03/04/2020 19:50	22C21D21-1F1B-4...	2 KB
f321baf7-f79c-455f-a87b-cf3cb4bc48f2.de37e6d6-559b-40b0-b1ad-ba2779fb...	03/04/2020 19:50	DE37E6D6-559B-4...	1 KB
BethesdaSoftworks.FalloutShelter_3275kfvn8vcwc.UWA	03/04/2020 19:53	UWA File	534,228 KB
Microsoft.SkypeApp_kzf8qxf38zg5c.UWA	05/04/2020 18:58	UWA File	187,328 KB
GoogleInc.YouTube_yfg5n0ztvskxp.UWA	05/04/2020 19:07	UWA File	167,680 KB
TwitchInteractive.TwitchApp_7kd9w9e3c5jra.UWA	05/04/2020 19:12	UWA File	166,064 KB
Microsoft.ZuneVideo_8wekyb3d8bbwe.UWA	05/04/2020 19:21	UWA File	140,044 KB
BooStudioLLC.ModernChrome_b6e429xa66pga.UWA	05/04/2020 19:30	UWA File	119,236 KB

Figure 24: User content partition

Prior to file selection and examination, the names and timestamps of various files provide a starting point. The folders share a modified timestamp with those from the prior partition, i.e. 01/09/2014 18:00. Interestingly, the file *LastConsole* was last modified three minutes after the folders were on 01/09/2014 18:23 yet is only 1 KB in size.

The first file to be analysed is *F321BAF7-F79C-455F-A87B-CF3CB4BC48F2*. This is one of three files in the partition with hexadecimal names, leading to calculating the entropy of each such file in hopes of determining said files’ importance. Given that the other two files with hexadecimal names had entropies of 3.492 and 2.925 respectively, they were disregarded as unlikely to contain personally identifiable information. The chosen file has an entropy of 7.991 which strongly suggests a high degree of compression or encryption. It could be argued that such a small file would not require heavy amounts of compression given the vast amount of storage capability present on this partition (780 GB), thus reinforcing the belief that this file is encrypted. There is potential for this file to contain information directly related to the

XB1's user based on both the name of the partition alongside its encrypted nature. However, given the difficult aspect of determining its exact nature, it is equally probable for compression to be in use to minimise the size of necessary base files required for the console's operation. The probability of it being an installed game is estimated to be lower due to its size (314,372 KB) coupled with the existence of the file

*BethesdaSoftworks.FalloutShelter\_3275kfyn8vcwc.UWA* which has a file size of 534,228 KB and an entropy of 7.246. It can be surmised that any files directly related to games would share a similar naming convention and compression schemes. All files except *BethesdaSoftworks.FalloutShelter\_3275kfyn8vcwc.UWA* mentioned above share a timestamp of 03/04/2020, indicating their involvement in the second phase of the experiment, with the mentioned file having a timestamp of 19:53, three minutes later than the afore-mentioned files who share a timestamp of 19:50. The file was not present prior in the first image and was created on 03/04/2020 15:50, suggesting that the file is related to the second phase of the experiment.

The second file to be selected is the largest file in the partition, namely *F321BAF7-F79C-455F-A87B-CF3CB4BC48F2-PLS* found within the PLS folder. Its size is 4,193,292 KB or 4.19 GB and has an entropy of 6.416. Aside from its final three characters, it shares its name with the first file selected for analysis in this partition. The entropy score indicates the file is compressed, suggesting a larger original size. The fact that this file shares its timestamp, 03/04/2020 15:50 with the afore-mentioned file of similar name, suggests the two are connected. Said timestamp coincides with the end of Fallout Shelter's installation process; together with the *BethesdaSoftworks.FalloutShelter\_3275kfyn8vcwc.UWA* file being created three minutes later upon launch of the game at 03/04/2020 15:53 strongly suggests the .UWA file containing information related to the user's playing of Fallout Shelter that was updated shortly after the game's launch. This can be seen in the below image:

FN_FileName	FileSizeBytes	SI_CTime	SI_ATime	SI_MTime	SI_RTime
F321BAF7-F79C-455F-A87B-CF3CB4BC48F2-PLS	4293931008	03/04/2020 15:50	03/04/2020 15:50	03/04/2020 15:50	03/04/2020 15:50
BethesdaSoftworks.FalloutShelter_3275kfyn8vcwc.UWA	547049472	03/04/2020 15:53	03/04/2020 15:53	03/04/2020 15:53	03/04/2020 15:53
F321BAF7-F79C-455F-A87B-CF3CB4BC48F2	321916928	03/04/2020 15:50	03/04/2020 15:50	03/04/2020 15:50	03/04/2020 15:50

Figure 25: *FalloutShelter*-related files & timestamps

Due to these reasons, the authors of this report conclude that *F321BAF7-F79C-455F-A87B-CF3CB4BC48F2* is directly tied to Fallout Shelter.



### 5.2.2.1 .UVA & .xvd file extensions

The above file extension was heavily featured in the second partition. After performing preliminary research on this file type, its exact nature was left unclear due to its apparent lack of referencing online. To aid in determining its nature, it was examined with a hex editor.

The file *BethesdaSoftworks.FalloutShelter\_3275kfvn8vcwc.UVA* was examined using HxD, a freely available hex editor intended for use on Windows operating systems. Its use can be seen below:

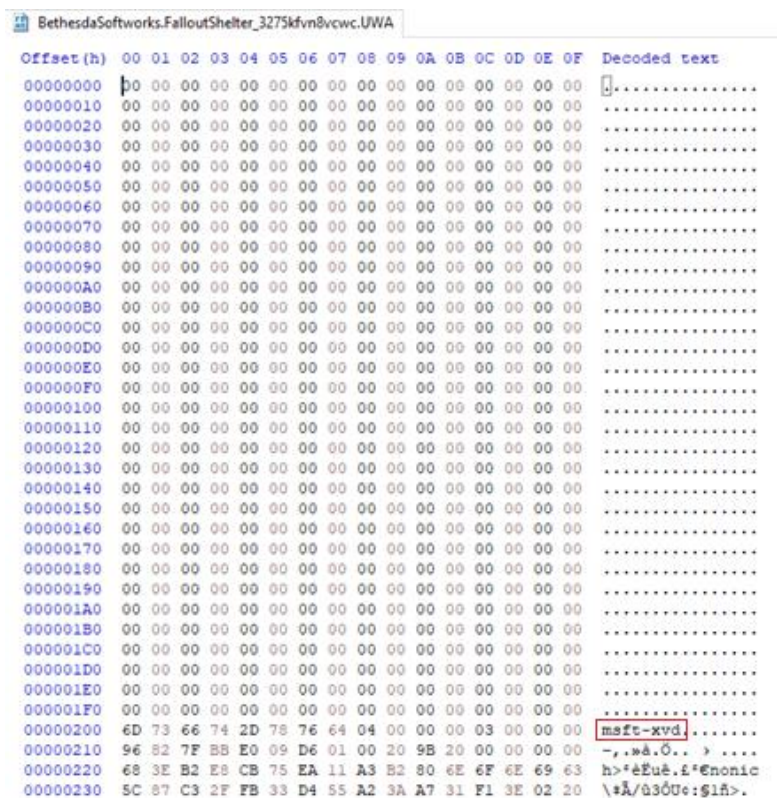


Figure 26: .UVA file in HxD

As can be seen from the highlighted red area in the above image, the .UVA file type seems to simply be a wrapper for .xvd files. As .xvd files are themselves a wrapper file format developed by Microsoft, it logically follows that “msft” stands for Microsoft. Across all files of type .xvd and .UVA, “msft-xvd” was present at address 0x200.

Members of the modding enthusiast forum named “se7ensins”, a community dedicated to modifying consoles through reverse-engineering, hypothesise .xvd files being treated as virtual disks by the XB1 for the purpose of the console being able to unmount a .xvd file to then mount a different .xvd file without necessitating the killing of a process. According to the user “schitzotm”, this would allow for higher levels of function, whilst employing less hardware. (schitzotm, 2018) Due to how running unsanctioned code or modifying the XB1 is against Microsoft’s T&C, this information cannot be substantiated by official sources.

### 5.2.3 Partition 3 – System Support

The logical view of the “System Support” partition can be seen below. It shares its version with the third image taken, i.e. post experiment.

Name	Date modified	Type	Size
oddfwupd	01/09/2014 18:00	File folder	
System Volume Information	03/04/2020 18:27	File folder	
\$RECYCLE.BIN	11/04/2020 14:10	File folder	
HostMemory.dmp	01/09/2014 18:00	Memory Dump File	475,144 KB
LastConsole	01/09/2014 18:00	File	1 KB
user.xvd	01/09/2014 18:00	XVD File	16,777,228 ...
WER.xvd	01/09/2014 18:00	XVD File	3,145,740 KB
F321BAF7-F79C-455F-A87B-CF3CB4BC48F2.xct	01/09/2014 18:01	XCT File	4 KB
F321BAF7-F79C-455F-A87B-CF3CB4BC48F2.xvi	01/09/2014 18:01	XVI File	4 KB
working	03/04/2020 20:30	Type 1 Font file	0 KB
esram.bin	03/04/2020 20:41	BIN File	32,768 KB

Figure 27: Partition 3 - System support

The following files were selected for analysis:

- *cms.xvd*
- *F321BAF7-F79C-455F-A87B-CF3CB4BC48F2.xvi*
- *esram.bin*

The first file selected for analysis is *cms.xvd*, mainly due to its name and its size of 16,777,228 KB (16.7 GB). This file makes up 42% of the entire partition yet its timestamp did not change through any of the experiment phases, remaining at 01/09/2014 14:00 through all stages. Interestingly the file seems to be larger than its reported size of 16.7 GB as in each image version of the third partition it has a size of 17,179,881.472 KB (17.1 GB). The reason behind this mismatch in reported size is unclear. Its entropy of 0.483 indicated a remarkable lack of compression and/or encryption which raises questions as to what its true purpose is.

The above file was also chosen due to it being featured in a preliminary forensic analysis of the XB1 by authors Moore et al. where they found *cms.xvd* was shown to have been updated on each shutdown of the console, yet were unable to ascertain its true purpose. (Moore, et al., 2014) This was not reflected in the results gained through this report’s experiment, suggesting changes in how the file is handled over time or this being an element specific to the Xbox One S All Digital.



The second file selected for analysis is *F321BAF7-F79C-455F-A87B-CF3CB4BC48F2.xvi*, chosen due to its hexadecimal name. The hexadecimal string is identical to that of the files selected for analysis in the previous partition showing it is directly related to Fallout Shelter. Furthermore, it is the first observed file with the extension “.xvi”. After performing research on said file type, it was found to be the extension matching “Xbox One Virtual Disk Info”. (DataTypes.net, 2020) This reinforces the claims put forward on “se7ensins.com” described above that .xvd files are treated as virtual disks by the XB1.

The third file to be analysed is *esram.bin*. This file has a size of 33554 KB (33.5 MB) and was not present on the first image and was recorded to have been created on 03/04/2020 at 16:41. At said point in time, no particular action was taken leading to an inability to determine its cause. However, based on its name and its size this report’s authors feel confident in stating that this file corresponds to the 32 MB of embedded static RAM (esRAM) the console was advertised as possessing.

#### 5.2.4 Partition 4 - System Update

Given the names of partition four and five they will be analysed together as it is unlikely for them to contain personally identifiable information but as in prior sections, files will be selected and examined.

The below images show a logical view of partition four’s structure.

Name	Date modified	Type	Size
\$RECYCLE.BIN	11/04/2020 14:10	File folder	
A	01/09/2014 18:00	File folder	
B	01/09/2014 18:00	File folder	

Name	Date modified	Type	Size
systemmisc.xvd	01/09/2014 18:05	XVD File	1,255,472 KB
system.xvd	01/09/2014 18:03	XVD File	1,228,508 KB
systemaux.xvd	01/09/2014 18:04	XVD File	725,752 KB
systemauxf.xvd	01/09/2014 18:04	XVD File	256,084 KB
SettingsTemplate.xvd	01/09/2014 18:01	XVD File	120,784 KB
SystemTools.xvd	01/09/2014 18:04	XVD File	62,284 KB
bootanim.dat	01/09/2014 18:04	DAT File	32,771 KB

Figure 29: "B" directory of partition 4

The first file to be examined is *updater.xvd* due to it being the sole file in the root directory.

At first glance it was hypothesised that this file is used or created in an update process.

Comparing the timestamps of the first image with the second and third image, its time of creation can be made clear. This can be seen in the below figures:

RecordOffset	FN_FileName	FileSizeBytes	SI_CTime	SI_ATime	SI_MTime	SI_RTime	FN_CTime	FN_ATime	FN_MTime	FN_RTime
0x0000000D784102400	systemmisc.xvd	1285603328	01/09/2014 14:04	01/09/2014 14:05	01/09/2014 14:05	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04
0x0000000D784101000	system.xvd	1257992192	01/09/2014 14:01	01/09/2014 14:03	01/09/2014 14:03	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01
0x0000000D784101400	systemaux.xvd	743170048	01/09/2014 14:03	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:03	01/09/2014 14:03	01/09/2014 14:03	01/09/2014 14:03	01/09/2014 14:03
0x0000000D784101800	systemauxf.xvd	262230016	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04
0x0000000D784100C00	SettingsTemplate.xvd	123682816	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01
0x0000000D7840F9800	\$LogFile	66535424	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D784102000	SystemTools.xvd	63778816	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04
0x0000000D784101C00	bootanim.dat	33557504	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04
0x0000000D784102800	LzmsYI	6291456	01/09/2014 14:24	01/09/2014 14:24	01/09/2014 14:24	01/09/2014 14:24	01/09/2014 14:24	01/09/2014 14:24	01/09/2014 14:24	01/09/2014 14:24
0x0000000D7840FAB00	\$Bitmap	393216	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FB400	\$Secure	262592	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840F9000	\$MFT	262144	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FB800	\$UpCase	131072	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FAC00	\$Boot	8192	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840F9400	\$MFTMirr	4096	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FA000	\$AttrDef	2560	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840F9C00	\$Volume	0	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FB000	\$BadClus	0	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FC000		0	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00				
0x0000000D7840FC400		0	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00				
0x0000000D7840FC800		0	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00				
0x0000000D7840FCC00		0	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00				
0x0000000D784100000	\$Repair	0	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FA400	.		01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FB000	\$Extend		01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FF000	\$Quota		01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FF400	\$Objid		01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FF800	\$Reparse		01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D7840FFC00	\$RmMetadata		01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D784100400	A		01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D784100800	B		01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00

Figure 30: First image parsed MFT - note the absence of "updater.xvd"

RecordOffset	FN_FileName	FilePath	FileSizeBytes	SI_CTime	SI_ATime	SI_MTime	SI_RTime	FN_CTime	FN_ATime	FN_MTime	FN_RTime
0x0000000D784106400	system.xvd	:\A\system.xvd	1360355328	03/04/2020 14:48	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:48	03/04/2020 14:48	03/04/2020 14:48	03/04/2020 14:48	03/04/2020 14:48
0x0000000D784102400	systemmisc.xvd	:\B\systemmisc.xvd	1285603328	01/09/2014 14:04	01/09/2014 14:05	01/09/2014 14:05	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04
0x0000000D784101000	system.xvd	:\B\system.xvd	1257992192	01/09/2014 14:01	01/09/2014 14:03	01/09/2014 14:03	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01
0x0000000D784101400	systemaux.xvd	:\B\systemaux.xvd	1171406848	03/04/2020 14:51	03/04/2020 14:52	03/04/2020 14:52	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:51
0x0000000D784101800	systemauxf.xvd	:\B\systemauxf.xvd	743170048	01/09/2014 14:03	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:03	01/09/2014 14:03	01/09/2014 14:03	01/09/2014 14:03	01/09/2014 14:03
0x0000000D784106C00	systemaux.xvd	:\A\systemaux.xvd	714620928	03/04/2020 14:50	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:50
0x0000000D784105800	delatas.xvd	:\A\delatas.xvd	696152064	03/04/2020 14:47	03/04/2020 14:48	03/04/2020 14:48	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47
0x0000000D784107800	systemauxf.xvd	:\B\systemauxf.xvd	393138176	03/04/2020 14:52	03/04/2020 14:52	03/04/2020 14:52	03/04/2020 14:52	03/04/2020 14:52	03/04/2020 14:52	03/04/2020 14:52	03/04/2020 14:52
0x0000000D784101800	systemauxf.xvd	:\B\systemauxf.xvd	262230016	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04
0x0000000D784100C00	SettingsTemplate.xvd	:\B\SettingsTemplate.xvd	123682816	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01	01/09/2014 14:01
0x0000000D784106800	SettingsTemplate.xvd	:\A\SettingsTemplate.xvd	123682816	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:50	03/04/2020 14:50
0x0000000D784105400	updater.xvd	:\updater.xvd	83275776	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47
0x0000000D7840F9800	\$LogFile	:\\$LogFile	66535424	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00	01/09/2014 14:00
0x0000000D784102000	SystemTools.xvd	:\B\SystemTools.xvd	63778816	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04
0x0000000D784107000	SystemTools.xvd	:\A\SystemTools.xvd	57511936	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:51	03/04/2020 14:51
0x0000000D784105C00	bootanim.dat	:\A\bootanim.dat	33673216	03/04/2020 14:47	03/04/2020 14:48	03/04/2020 14:48	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47	03/04/2020 14:47
0x0000000D784101C00	bootanim.dat	:\B\bootanim.dat	33557504	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04	01/09/2014 14:04

Figure 31: Second image parsed MFT - updater.xvd is highlighted

The above image shows that *updater.xvd* was created on 03/04/2020 14:47. This matches the time at which an ethernet connection was first provided to the console, i.e. when the console was powered on for the first time. Its entropy score was calculated to 7.928, strongly indicating encryption or compression. Based on the name of the partition and the name of the file itself, it is believed this file is a heavily compressed update manager of some kind; however, this is simply a hypothesis and cannot conclusively be proven.

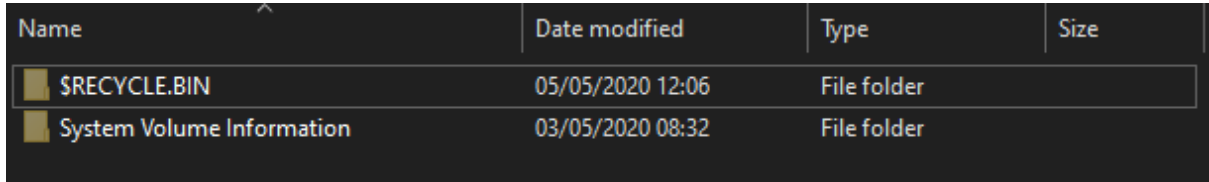
The second file to be analysed is *system.xvd*. There were two files with the afore-mentioned name, one within directory “A” and the other in directory “B”. The former has an entropy score of 8, the timestamp of 03/04/2020 14:48 and a size of 1360355 KB (1.36 GB), whereas the latter has an entropy score of 7.999, the timestamp of 01/09/2014 and a size of 1257992 KB (1.25 GB).

Entropy scores reported are the largest of any file examined within this report and strongly indicate encryption/compression in use. Although conjecture, this report’s authors believe such scores suggest this file could contain potentially proprietary and/or sensitive information concerning system operations. Moreover, there is a difference in file size between that of the first file and that of the second – specifically 102363 KB (1.02 GB) – generated resulting from the initial starting of the XB1. Thus, the data making up the difference in file size was a direct result of the console’s initial launch. Furthermore, the second file’s inclusion on the original image, coupled with said file being 92.5% the size of the first file, lends credence to the conjecture proposed above. Ultimately, even were this conjecture to be proven true, it is still highly unlikely for it to contain personally identifiable data.

The remainder of files contained within directory “A” had timestamps ranging from 03/04/2020 14:48 to 03/04/2020 14:52 showing said files were generated after the initial launch of the XB1, whereas the remainder of files within directory “B” had timestamps ranging from 01/09/2014 14:00 to 01/09/14:04. The latter set of files never changed since the assumed date of the console’s creation. In conclusion, no personally identifiable information was identified in partition four.

### 5.2.5 Partition 5 – System Update 2

The final partition does not contain any files viewable through a logical view. Neither of the two folders were accessible. The regrettable lack of any files to analyse led to this report's authors inability to select files for examination. The logical view can be seen below:



Name	Date modified	Type	Size
\$RECYCLE.BIN	05/05/2020 12:06	File folder	
System Volume Information	03/05/2020 08:32	File folder	

*Figure 32: Logical view of partition 5*

## 6. Discussion

Gaming consoles' relevancy to the field of digital forensics has steadily grown as their production cost has decreased and their prevalence across society has increased. Not only have consoles' popularity risen over time but so too have they evolved from independent gaming platforms to more complete multimedia systems. Such a change in intended purpose prompted the realisation that information contained on such consoles can be used as evidence and led to the birth of the field of console forensics.

As the above-described shift occurred recently (relative to the birth of gaming consoles), the future will see the difference between a PC and a gaming console shrink. However, this does not preclude a lessening of consoles' numbers across society, more likely the opposite. If the trend set by systems such as the XB1 holds its course, families might choose to purchase a console rather than a PC furthering the prevalence of consoles across homes as well as their relevancy in digital forensic investigations. Already consoles are being used in criminal investigations as evidence; however, said evidence's weight relies largely on a close working relationship between law enforcement and makers of the consoles. In large part, this is due to the proprietary technology involved requiring equally proprietary knowledge to unravel and to specifically identify relevant data.

Such a relationship is troubled by questions surrounding privacy and oversight. This is especially concerning when users have no ability to tell how data is stored, how it is secured and in what way data is to be transferred to authorities outside of the company's guarantee or the promise of legal oversight. Investigations such as the one performed in this report and those performed in the future will allow for a more concrete picture to be painted for current and future users and set the stage for further research.

Modification of consoles have garnered substantial interest in the digital forensic community due to both the temptation of unravelling proprietary techniques used in their production and the knowledge that, once modifiable, data can be obfuscated without checks and balances.

The XB1's two predecessors, the Xbox and the XB360, were both modified by gaming enthusiasts prior to the launch of the next generation's consoles allowing for unsigned code to be run. This lets the proverbial cat out of the bag as Microsoft's ability to assist authorities or guarantee its product's protective mechanisms is weakened significantly. Although the XB1 is of the 8<sup>th</sup> generation, at the time of this report's writing it is the first to successfully prevent

modification of the console and the running of unsigned code, there remains a possibility for it to be eventually modified. Moreover, if future Microsoft consoles were to employ similar methods to guard against modification, information gleaned from the XB1 could assist in their future modification.

## 7. Conclusion

This dissertation has revolved around an exploratory analysis of the XB1. To answer the research questions posed, a twinned approach of literature study alongside performing an experiment was chosen. The literature study aimed to explain the necessary knowledge regarding the XB1, whereas the experiment would generate the results needed to answer the asked research questions. Worthy of further reinforcement was the exploratory aspect of this report, which dictated a fluid & dynamic scope prone to constant evaluation. Over the course of this investigation, several decisions were made aimed at limiting the potential of scope creep.

*What information can be extracted from an XB1 through use of a digital forensically acquired image?*

The above question was the first listed research question aimed at general discovery with no caveats. The XB1 was imaged prior to being turned on for the first time; this image functioned as the control and as a point of comparison. The second stage of the experiment used Autopsy to perform keyword searches related to the games installed, played (single player and multiplayer), names of the researchers and the email used at initial registration. Substring searches were used, thus generating a large volume of data that was then examined manually. Through said examination, references to assets hosted online were found for “Roblox”, whereas the search term “Victor” produced largely irrelevant results. However, all terms searched for showed that responses and background for contained applications such as the built-in calendar, or for a digital avatar, existed as well as showing that some code in the form of “.xml” sheets were present.

The final stage of the experiment aimed at generating artefacts related to various installable applications. No sensitive data was uncovered from this stage; however, information was still gleaned concerning protocols used by certain applications. This could be seen through how the UAP protocol was referenced in “.xml” code that was found in substring keyword search matches. Furthermore, it was discovered that it is likely that applications that can be installed on the XB1 are specific to the platform through analysis of “.xml” code. This presented the possibility that checks could be performed on whether the system is truly an XB1, thus prompting the question of whether the applications available for installation form a defence against unsigned code.

Lastly, due to encryption employed on the XB1, Autopsy was not able to report the names of files and instead, reported all matches from the same file of an incredibly large size. This was curious and was looked at in further detail but was unable to be resolved, leading to the realisation that another approach would be required.

*Develop a valid forensic methodology specifically adapted to the 8<sup>th</sup> generation console, the XB1, that accounts for its unique characteristics.*

The XB1 is Microsoft's 8<sup>th</sup> generation console and was the target device of this report. The methodology was designed to allow for causality to be determined through three sets of actions taken after which each HDD was imaged. Proper digital forensic procedures were identified and moulded into the methodology such that results gained could be deemed valid. After using Autopsy to analyse and perform keyword searches on each of the images, no identifiable information was identified.

This led to this report's researchers' decision to explore other avenues for analysis and ultimately, the decision to use a logical view was made. Such a decision was not considered a deviation from the methodology, rather an evolution. However, since a logical view only presents up-to-date information, a complementary form of evidence was required to verify and distinguish patterns in the changes of files. This was accomplished through parsing the MFT of each partition on each image, exporting said results to excel in ".csv" format wherein timestamps could be compared. Such an approach proved to produce relevant results which corroborated the presence and use of encryption as well as compression on files logically viewable on the XB1 through use of Shannon's entropy score. Each partition was examined through the selection of various files, providing a basis for hypotheses concerning the roles of the partitions and their likelihood of containing personal information. Logically viewing available files also presented an insight into the starting position for any would-be malicious user intent on hiding information.

Given how limited the results gained from the use of Autopsy were, this report determines the MFT as highly critical to any digital forensic investigation of an XB1 and steps should be taken by investigators to fully secure this data.



*How, and to what extent, do the implemented security features of the 8<sup>th</sup> generation console, the XBI, affect a forensic examiner's ability to extract uniquely identifiable information from a forensic disk image?*

The above question was most pertinent to a digital forensic investigation yet was predicted to be challenging to answer conclusively. It was deemed unlikely to find any personally identifiable information on the gaming console, given that it was developed for a large market and had to be compliant with laws governing personal data. In summary, no such information was found through use of any of the techniques and procedures followed through the course of this investigation. However, it was found that timestamps did change because of the console's initial launch and in other files, upon shutdown. Such information could be used by investigators to corroborate alibis or determine a timeline, but its potential applicable scenarios are limited given the working relationship Microsoft has with authorities. Due to proprietary information that only the console's makers possess as well as the personal data of their users which is protected, data concerning the times at which actions were taken would be easier to request from Microsoft directly rather than solely being reliant on a HDD image.

Ultimately, the security features presented insurmountable challenges for the examiners and severely limited the data gained.

*Does the XB1 represent significant advancements in OS modification prevention when compared to its predecessor?*

It can be conclusively stated that a significant improvement in OS modification prevention is seen in the XB1 given that its two predecessors were both successfully modified by enthusiastic members of the gaming community. This presents a double-edged sword situation wherein no information concerning the inner workings of the XB1 has been unlocked by anonymous users whilst no technical aspects of the console's technology are revealed by Microsoft, given that it is classified as proprietary knowledge. The aforementioned situation displays the necessity for a dynamic scope in an exploratory investigation as information is limited on all sides.

The way in which the running of unsigned code is prevented is based on the XB1's use of sandboxes (virtualised environments) where one application is unable to read and/or modify another application's files. This builds upon the concept of identifying and isolating potential points of failure as it allows for a modular, yet secure, approach through which one application's weaknesses will not impact another's. DEP is also employed in order to prevent the rewriting of memory thus preventing buffer overflow attacks adding to the console's protection.

## References

Alazab, M., Watters, P. & Venkatraman, S., 2009. Effective digital forensic analysis of the NTFS disk image. *Ubiquitous Computing and Communication Journal*, 4(3), p. 9.

Anon., 2018. *Reddit*. [Online]

Available at:

[https://www.reddit.com/r/xboxone/comments/7wfauv/tech\\_why\\_is\\_hacking\\_on\\_xbox\\_one\\_not\\_possible/dtzzwb5/](https://www.reddit.com/r/xboxone/comments/7wfauv/tech_why_is_hacking_on_xbox_one_not_possible/dtzzwb5/)

[Accessed 7 May 2020].

Autopsy Digital Forensics, 2020. *Autopsy / Use Cases*. [Online]

Available at: <https://www.autopsy.com/about/use-cases/>

[Accessed 29 February 2020].

Baset, S. A. & Schulzrinne, H. G., 2005. *An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol*, s.l.: IEEE.

Custer, H., 1994. *Inside the Windows NT file system*. Redmond: Microsoft Press.

DataTypes.net, 2020. *XVI file - The easiest way to open .xvi files in 2020 / DataTypes.net*.

[Online]

Available at: <https://datatypes.net/open-xvi-files>

[Accessed 29 April 2020].

Garfinkel, S. L., 2013. Digital media triage with bulk data analysis. *Computer & Security*, 2013(32), pp. 56-72.

Hope, C., 2019. *What is MFT (Master File Table)?*. [Online]

Available at: <https://www.computerhope.com/jargon/m/mft.htm>

[Accessed 16 May 2020].

Hope, C., 2019. *What is Slack Space*. [Online]

Available at: <https://www.computerhope.com/jargon/s/slack-space.htm>

[Accessed 15 May 2020].

Kozierok, C. M., 2001. *History of NTFS - PCGuide*. [Online]  
Available at: <https://www.karlstechnology.com/blog/history-of-ntfs/>  
[Accessed 11 February 2020].

LSoft Technologies Inc, 2019. *NTFS.com*. [Online]  
[Accessed 18 February 2020].

Lv, G., Xu, X., Su, K. & Chen, Q., 2011. *UAP: A New UDP-Based Application Level Transport Protocol*. Beijing, s.n.

Microsoft, 2007. *New Capabilities and Features of the NTFS 3.1 File System*. [Online]  
Available at: <http://support.microsoft.com/kb/310749>  
[Accessed 11 February 2020].

Microsoft, 2009. *FAT Technical Reference*. [Online]  
[Accessed 24 February 2020].

Microsoft, 2018. *File Times - Win32 apps / Microsoft Docs*. [Online]  
Available at: <https://docs.microsoft.com/en-us/windows/win32/sysinfo/file-times?redirectedfrom=MSDN>  
[Accessed 27 February 2020].

Minitool, N/A. *File Allocation Table (FAT): What Is It? (Its Types & More)*. [Online]  
Available at: <https://www.minitool.com/lib/file-allocation-table.html>  
[Accessed 12 February 2020].

Moore, J., Baggili, I., Marrington, A. & Rodrigues, A., 2014. Preliminary forensic analysis of the Xbox One. *Digital Investigation*, Volume 11, p. 9.

Richard, G. & Roussev, V., 2005. *Scalpel: A Frugal, High Performance File Carver*. New Orleans, s.n.

Rusbarsky, K. L., 2012. *A Forensic Comparison of NTFS and FAT32 File Systems*, Kansas City: Marshall University Forensic Science Center & FBI Computer Forensics Laboratory.

schitzotm, 2018. *Has anyone figured out how to decrypt or know what .xvd files are*. [Online]  
Available at: <https://www.se7ensins.com/forums/threads/has-anyone-figured-out-how-to->

[decrypt-or-know-what-xvd-files-are.1733343/](#)

[Accessed April 2020].

Shannon, M., 2004. Forensic Relative Strength Scoring: ASCII and Entropy Scoring. *International Journal of Digital Evidence*, 2(4), p. 19.

Standard of the Camera & Imaging Products Association, 2010. *CIPA DC- 009-Translation-2010*, s.l.: Camera & Imaging Products Association.

Where is my data, n.d. *Dates: NTFS Created, Modified, Accessed, Written / Where is your data*. [Online]

Available at: <https://whereismydata.wordpress.com/2009/02/14/dates-ntfs-created-modified-accessed-written/>

[Accessed 27 February 2020].

Wikipedia, 2015. *Track (disk drive)*. [Online]

Available at: [https://en.wikipedia.org/wiki/Track\\_\(disk\\_drive\)](https://en.wikipedia.org/wiki/Track_(disk_drive))

[Accessed 24 February 2020].

Xynos, K. et al., 2010. Xbox 360: A digital forensic investigation of the hard disk drive. *Digital Investigation*, 104(111), p. 8.

## Glossary

The below terms were defined using “Techopedia”, a website specialising in providing definitions for all IT-related terms, unless otherwise referenced.

- **New Technology File System (NTFS):** “The New Technology File System (NTFS) is the standard file structure for the Windows NT operating system. It is used for retrieving and storing files on the hard disk.”
- **File Allocation Table (FAT):** “A file allocation table (FAT) is a file system developed for hard drives that originally used 12 or 16 bits for each cluster entry into the file allocation table. It is used by the operating system (OS) to manage files on hard drives and other computer systems. It is often also found on in flash memory, digital cameras and portable devices.”
- **High Performance File System (HPFS):** “The high-performance file system (HPFS) is a file system designed especially for the IBM OS/2. It is known for handling large files of up to 2 GB across multiple hard disks, as well as for handling long file names of up to 256 bytes. HPFS was designed to improve on the weaknesses of the file allocation table file system.”
- **Redundant Array of Independent Disks (RAID):** “Redundant array of independent disks (RAID) is a method of storing duplicate data on two or more hard drives. It is used for data backup, fault tolerance, to improve throughput, increase storage functions and to enhance performance.”
- **Slack space:** “Slack space refers to the storage area of a hard drive ranging from the end of a stored file to the end of that file cluster.” (Hope, 2019)
- **Master File Table (MFT):** “The MFT is an index of all files on an NTFS volume. It contains the file name, a list of the file attributes, and pointers to the file fragments.” (Hope, 2019)
- **Cluster:** “A computer cluster is a single logical unit consisting of multiple computers that are linked through a LAN. The networked computers essentially act as a single, much more powerful machine.”
- **Sandbox:** “A sandbox, in computer security, is a security mechanism in which a separate, restricted environment is created and in which certain functions are prohibited.”

- **Timestamp:** “A timestamp is temporal information regarding an event that is recorded by the computer and then stored as a log or metadata.”
- **Entropy:** Entropy is a measure of information density. (Shannon, 2004)
- **Forensic Relative Strength Scoring:** “The more a given unit can be compressed, the lower the entropy value; the less a given unit can be compressed, the higher the FRSS value.” (Shannon, 2004)
- **Data Execution Prevention:** “Data execution prevention (DEP) is a security feature within operating system that prevents applications from executing code from a non-executable memory location.”
- **Buffer overflow:** “A buffer overflow occurs when more data are written to a buffer than it can hold. The excess data is written to the adjacent memory, overwriting the contents of that location, and causing unpredictable results in a program.”

Robbin is born and raised in Avesta, Sweden. He has always had an affinity for technology and when he decided to switch careers, after six years in the Swedish Armed Forces, cybersecurity seemed like a perfect fit.

Victor has always known his future lay somewhere within the broad domain of computing thanks to a longstanding fascination with technology and computers. Raised in Zurich, Switzerland he moved to Sweden for the first time in his life to attend Halmstad University.



PO Box 823, SE-301 18 Halmstad  
Phone: +35 46 16 71 00  
E-mail: [registrator@hh.se](mailto:registrator@hh.se)  
[www.hh.se](http://www.hh.se)