

ABSTRACT

The array of electronic storage devices is staggering in both number and type. The most common of these are IDE hard disk drives, laptop drives, thumb drives and other removable media like CD's and DVD's. Other media that are not as common but certainly as functional are high capacity magnetic card media, cell phones, PDA's, and game consoles. These devices provide a convenient means to store data of all kinds, but they also provide a way for criminals to possess and hide illegal material.

The Xbox game console is a reasonable place for a criminal to store and view illegal material such as child pornography. The Xbox is not designed to support this activity, and because of this, it takes some modification to the operating environment to get this to work. Once an Xbox has been modified, it can be used to store and view illegal material. The Xbox uses the FATX file system. FATX is largely un-documented and is not readable by the leading forensic software; consequently, without an automated tool, a forensic examiner will have a difficult time getting information from a FATX file system, most likely resorting to a low-level analysis of a hex dump of the hard drives contents. Tools exist to extract files of a specific type in this fashion, however forensic browsing, searching, and organizing files in the file system is not feasible with this manual, low level approach.

XFT is a command line utility that will mount an image of a FATX file system as a logical drive, allowing full traversal of the directory structure. The XFT user interface is similar to a Linux shell. Once the Xbox file system is mounted, the analyst can use shell commands to browse the directory tree, open files, view files in hex editor mode, list the contents of the current directory in short or long mode, and expand the current directory to list all associated subdirectories and files. In addition, XFT will record a session to a log file so that the entire browsing session can be played back in a court of law. The tool is currently under development and more functionality will follow including searching, sorting, recovery of deleted files, and smart file type recognition.

1. XBOX MODIFICATION

These modifications or "mods" can be of the hardware or software variety. A hardware mod is a physical alteration of hardwired circuitry on the Xbox motherboard, which allows a user to boot from a hard drive containing a file system that is not FATX. These added hard drives can be high capacity drives with 3rd party operating systems, effectively turning the game console into a personal computer. [4] Software mods are, by contrast, achieved by exploiting some buffer overrun vulnerability in a specific game, and spawning a third party operating environment over existing hardware. This environment could support anything that the programmer envisions, but usually includes file browsing, FTP and image viewing utilities.

A criminal wishing to store and view illegal material would find the soft-mod an attractive choice for several reasons: The soft-mod does not require the individual to physically open the box. This reduces the chances of detection. The soft-mod can and usually does leverage the existing FATX file system. This file system is largely un-documented and is not readable by the leading forensic software. Finally, with the soft-mod, the game console is exploited dynamically. By this we mean that without the vulnerable game and corresponding buffer overrun software, the Xbox is, forensically or otherwise, un-modified.

2. EXPLOITING AN XBOX

XFT is targeted at Xbox systems exploited by soft-mods which use the underlying FATX file system for storing and hiding information. One such exploit is Softmod Installer Deluxe, written by an anonymous hacker who goes by the initials DJB. Softmod Installer Deluxe is available to anyone as a free download from <http://www.xbox-hq.com>. This exploit takes advantage of buffer overrun vulnerability in an older version of the 007 Agent Under Fire Xbox game. The exploit comes in the form of a game save which is stored on the Xbox just as any legitimate game save would be stored. When 007 Agent Under Fire is loaded, the user would then load the game save. The game save, however, contains code that will overrun a buffer in memory causing a third party operating environment to be executed. This operating environment has FTP, file browsing, file viewing, partition cloning, and network configuration functionality. Once the individual is done, she can then eject the game, restart the Xbox, and the system boots to the normal interface with no indication of any modification having taken place. Without examining the hard drive in a forensic manner (imaging it and traversing the file system), it would be difficult to tell that anything had happened.

Once the Xbox is exploited and running a rogue operating environment, a user can set up networking through a provided interface. The network configuration usually allows for DHCP or static IP addresses. Once the Xbox has an IP address, the user can set up FTP by setting the appropriate option and re-starting networking. At this point, the user may wish to create a directory on the Xbox by simply using FTP and the mkdir command. Once the directory is created, the user can then make an FTP connection to the machine, change directories, and upload images to the Xbox. The Softmod Installer Deluxe has an image viewer installed, so not only can the user store images on the Xbox, but he can view them as well. This is an attractive feature to a criminal, because it reduces the number of transfers, and locations of illegal material, thus making it more difficult for a forensic analyst to find.

3. AQUIRING THE EVIDENCE

An Xbox hard drive acquisition is performed in a similar fashion as any other hard drive acquisition. The Xbox hard drive is an ATA/IDE device with a 10GB storage capacity. The drive has a locking mechanism such that 3rd party software will not recognize it unless it already booted or unlocked. If an analyst connects a write-blocker to an Xbox hard drive while the Xbox system is not booted, the imaging software (or any software such as the acquisition machine operating system) will not see the drive. To get around this problem, the Xbox must first be booted, and then we can remove the IDE cable and replace it with a write blocker connected to the analysis machine. The analysis machine will then see the Xbox hard drive and the analyst can use imaging software to create an image of the drive. It is always important in forensics to be able to verify that data has not changed after being copied. Hashing the drive before and after all copies and transfers is recommended in any forensic examination, including Xbox forensics. Once hashes are made and the copy has been verified, we can then manually examine the file system. We will first discuss manual analysis of FATX, and then examine the XFT tool which automates much of the process.

4. THE FATX FILE SYSTEM

FATX is the file system used in the Xbox. FATX is similar to FAT32 but with some differences. This similarity makes the file system easy to examine for an analyst familiar with

FAT32. [3] The differences do, however, preclude the leading forensics software packages from reading FATX because FATX is simply not recognized by these packages.

FATX file systems are organized into 6 partitions:

1. The disk config area
2. File cache area 1
3. File cache area 2
4. File cache area 3
5. System files
6. Game save storage

5. THE DISK CONFIGURATION AREA

This area contains meta-data about the file system. This partition is 512 KB and is largely un-decoded. We can, however, ascertain some important information from this partition. The disk configuration area contains the magic flag “BRFR” ASCII identifier at offset 0x600. [2] This identifier tells us that the drive is formatted with the FATX file system. A 4 byte value at offset 0x604 tells us the number of boots. Everything from offset 0x608 through 0x101c has not yet been de-coded. [2] The following table shows the offset, size and description of the remaining meta-data in the disk configuration area:

Offset	Size	Description
0x104c	2	Xbox Live Settings
0x104e	6	Xbox Live MAC Address
0x1054	4	Xbox Live IP Address
0x1058	4	Xbox Live Subnet Mask
0x105c	4	Xbox Live Default Gateway
0x1060	4	Xbox Live Primary DNS
0x1064	4	Xbox Live Secondary DNS
0x1068	40	Xbox Live Hostname
0x1090	64	Xbox Live PPPOE Username
0x10d0	64	Xbox Live PPPOE Password
0x1100	40	Padding
0x1138	40	Xbox Live PPPOE Service Name

The remaining partitions are FATX partitions. [2] Each FATX partition is organized in the same fashion. A FATX file system contains a meta-data section comprising the first 18 bytes of the partition and they are as follows:

<u>Offset</u>	<u>Size</u>	<u>Description</u>
0x0000	4	“FATX”
0x0004	4	Volume ID (?)
0x0008	4	Cluster size in 512 byte sectors
0x000C	2	Number of active FATs (always 1)
0x000E	4	Unknown (always set to 0)

[2] The section that follows starts at offset 0x1000 and is sometimes referred to as the cluster chain map but it functions the same way that the FAT table functions in a FAT32 file system. In order to traverse a FATX file system, an analyst must discern the size of the cluster chain map. This value is not given in the meta-data section and must be calculated. The cluster chain map (CCM) is nothing more than a listing of pointers to clusters. The CCM can be of the 2 or 4 byte variety corresponding to FAT16 or FAT32. Each entry represents a cluster in the partition and all clusters, allocated and unallocated, are represented in the CCM. This means that if we divide the partition size by the cluster size, we get the number of clusters in the partition. If we then multiply the number of clusters in the partition by the byte mode (2 or 4) we arrive at the size of the CCM in bytes simply because the CCM has entries for each cluster. There is one more factor in determining the exact size of the CCM and that is the notion of blocks. The smallest unit that is usually read from a disk is a sector and they are usually 512 bytes. Files tend to be bigger than this, so we use clusters as a further abstraction. Each file uses at least one cluster, so from the file systems point of view, the smallest a file can be is the cluster size. Most files do not take up an exact multiple of a cluster, so we do have unused space at the end of most clusters called file slack. [1] Although the file system sees clusters, data, including the CCM, are usually written to disk in smaller units called blocks. Consequently, the size of the CCM must be a multiple of the block size. To ensure that we have the correct CCM size we must perform the previous calculation, and then round up to the nearest block. The formula is as follows:

- **CCM_SIZE=(PARTITION_SIZE/CLUSTER_SIZE) * BYTE_MODE**
- **Round the CCM_SIZE up to the nearest 4096 byte block.**
 - **CCM_SIZE=((CCM_SIZE/4096)+1) * 4096**
 - *** Integer division in C truncates the remainder**

We must calculate this value before proceeding further because otherwise we will not know where the data area begins. The data area is found at offset 0x1000 + CCM size. [2]

6. THE DATA AREA

The data area begins immediately after the CCM. The root directory is implied with the start of the CCM. The first entry in the data area corresponds to the beginning of cluster 1 and it is the first child of root. If the file system contains data, then the first entry in the data area will be a directory entry (it is called a directory entry regardless of file/directory status). Each directory entry is 64 bytes in length, and describes the directory or file associated with it. Meta-data about the associated directory or file including file name, first cluster, file attributes, created, last modified, last access times, and file size are included in the directory entry. The key with file system traversal is the starting cluster of the associated file. This is a 2 or 4 byte value stored in

little endian fashion, indicating the starting cluster of the file. It is important at this stage to realize that the file may be fragmented and take up multiple clusters. If the file is larger than 1 cluster, we will need to consult the CCM for a cluster chain. We can make this determination by checking the file size at offset 0x30 from the start of the directory entry. If the file size is greater than the cluster size, then the file takes up more than 1 cluster and may be fragmented. If this is the case, we must consult the CCM, traverse the cluster chain, and build the file. CCM entries contain either a pointer to the next cluster in the file, an end of file marker (0xFF or 0xFFFF depending on byte mode) or 0 which indicates an unallocated cluster. [2] When building a file, we get the starting cluster from the directory entry and add it to a list. We then multiply that value by the byte mode to find the byte offset in the CCM for the next cluster pointer. If the next cluster pointer is allocated and if it is not an end of file marker, then we add it to the list and repeat the process until an end of file (EOF) marker is reached. Once an EOF marker is reached, following the above procedure, we will have accumulated an ordered list of clusters belonging to the file in question. To build the file, we must first convert the cluster numbers to byte offsets from the beginning of the data section. Once we have a list of byte offsets, we can create an empty file, and then we can jump to each offset (each jump originating from the beginning of the data area) in order, concatenating the contents of each identified cluster to the file. Once this is complete, we have the file data and we can then detach it for further analysis separate from the file system. [2]

7. XFT – AUTOMATING THE PROCESS

XFT, written by the author of this paper, is a command line utility that will mount an image of a FATX file system as a logical drive, allowing full traversal of the directory structure. The XFT user interface is similar to a Linux shell. Once the Xbox file system is mounted, the analyst can use shell commands to browse the directory tree, open files, view files in hex editor mode, list the contents of the current directory in short or long mode, and expand the current directory to list all associated subdirectories and files. In addition, XFT will record a session to a log file so that the entire browsing session can be played back in a court of law. The tool is currently under development and more functionality will follow including searching, sorting, recovery of deleted files, and smart file type recognition.

To start the program, the user would follow the usage rules of the software listed below:

Usage: xft3.1 [INPUT IMAGE][-l][-p LOG FILE]

Options:

INPUT IMAGE: The image you wish to analyze
-l: Event recording for use as input to playback option.
-p LOG FILE: Play's a session from a specified Log

XFT begins by giving the user a '>'. The user can only use the 'mount' command at this point to mount a FATX file system. A typical command and output would be:

```
>mount /GAMESAV  
/GAMESAV>
```

The current command list follows:

cd [..][<i>Directory</i>]	Change directories
expand	Expand all children
exit	Quit the program
ls [-l]	Short or long file listing
mount [/Partition]	Mount a partition
open [-h]	Open a file in hex mode
[-v]	Launch a default viewer
pinfo	Display partition info
vinfo	Display volume info

REFERENCES

1. Carrier, B. "File System Forensic Analysis".
2. DeQuincey, A., and Murray-Pitts, L., "Xbox Partitioning and Filesystem Details". http://www.xbox-linux.org/wiki/Xbox_Partitioning_and_Filesystem_Details
3. Steil, M. "Differences between Xbox FATX and MS-DOS FAT". http://www.xbox-linux.org/wiki/Differences_between_Xbox_FATX_and_MS-DOS_FAT
4. Huang, Andrew, "Hacking the Xbox, An Introduction to Reverse Engineering".