



Forensic Analysis of the Nintendo Wii Game Console

Peter Stewart

Department of Computer and Information Sciences

September 2010

Declaration

A thesis submitted to the Department of Computer and Information Sciences, University of Strathclyde, in part fulfilment of the regulations for the degree of Master of Science in Forensic Informatics.

I declare that, in accordance with University Regulation 20.1.20, this dissertation embodies the results of my own work and that it has been composed by myself. Following normal academic conventions, I have made due acknowledgement to the work of others.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to provide copies of the dissertation, at cost, to those who may in the future request a copy of the dissertation for private study or research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to place a copy of the dissertation in a publicly available electronic archive.

Yes ☐ **No** ☐

Signed _____ **Date** _____

Student Number: 200957905

Abstract

Like other modern game consoles, the Nintendo Wii provides users with a powerful networked device capable of performing many of the tasks carried out by a conventional desktop personal computer. Unlike other modern game consoles however, the Nintendo Wii utilises an internal NAND flash storage device in lieu of a standard hard disk drive, and thus cannot be imaged in the same manner as the Microsoft Xbox or Sony Playstation 3. The difficulties in imaging the device are exacerbated by the tightly-controlled, proprietary nature of the platform, and have led to forensic examiners being faced with the choice of ignoring the Nintendo Wii completely, or performing a “live examination” and potentially destroying evidence.

Through a series of experiments, investigates the feasibility of a number of hardware and software procedures designed to capture the raw data held by the Nintendo Wii’s NAND flash storage device so that conventional digital forensic techniques may be applied to the console. In addition to the successful capture of data, this report also describes a process by which the console can be restored to a previously-captured state, reducing the risks associated with performing a “live examination”. Also described is the analysis of the captured NAND flash image, which has demonstrated the recovery of a partial history of internet usage and sent “Wii Message Board” communications – information which was previously thought to be inaccessible by any other means.

Acknowledgements

I would like to thank Ian Ferguson, my project supervisor, for all of his assistance and support throughout the various stages of this project.

I also wish to thank Bruce Ramsay of the Lothian and Borders Police Forensic Computer Unit for his valuable insight into the hardware-based experiments conducted during the project.

Table of Contents

Declaration	ii
Abstract.....	iii
Acknowledgements	iv
Table of Contents	v
List of Figures.....	ix
List of Tables.....	x
Chapter 1 – Introduction	1
1.1 Rationale.....	1
1.2 Aims and Objectives.....	2
Chapter 2 – Overview of the Nintendo Wii	4
2.1 Hardware	5
2.1.1 NAND Flash Storage	6
2.1.2 Secure Digital Flash Memory Card Reader	7
2.1.3 The “Broadway” Processor	8
2.1.4 The “Hollywood” and “Starlet” Processor Package	8
2.1.5 Motherboard Revisions	9
2.2 Software	12
2.2.1 IOS	12
2.2.2 Wii System Menu	13
2.2.3 Channels	14
2.3 General Operation.....	15
2.3.1 The Boot Process	16

2.3.2 Network Connectivity.....	18
2.3.3 Internet Browsing.....	20
2.3.4 Wii Message Board	21
Chapter 3 – Literature Review	23
3.1 Microsoft Xbox & Xbox 360 Forensics	23
3.2 Sony Playstation 3 Forensics.....	25
3.3 Nintendo Wii Forensics	27
Chapter 4 – Proposed NAND Flash Imaging Methods	31
4.1 Using Hardware.....	31
4.1.1 Joint Test Action Group (Boundary-Scan)	32
4.1.2 The <i>Infectus2</i> NAND Flash Controller.....	32
4.2 Using Software.....	34
4.2.1 Linux Approach	34
4.2.2 Homebrew Software Approach	35
4.3 Evaluation of Proposed Methods.....	36
4.3.1 Potential for NAND Data Modification	37
4.3.2 Requirement of Specialist Skills or Tools	38
4.3.3 Robustness of Approach	40
4.4 Proposed NAND Flash Imaging Guidelines	41
Chapter 5 – Equipment and Experimental Methodology	43
5.1 Equipment	43
5.2 Experimental Methodology	45
Chapter 6 – Hardware-based Imaging Experiment	48
6.1 Preparation of the <i>Infectus2</i> Chip	48

6.2 Imaging Procedure	51
6.3 Analysis of Experiment	56
6.3.1 Possible Causes of Failure	56
Chapter 7 – Software-based Imaging Experiments.....	59
7.1 Exploiting Vulnerabilities in Wii System Software	59
7.1.1 The Twilight Hack.....	60
7.1.2 Bannerbomb.....	60
7.1.3 Smash Stack.....	61
7.1.4 Indiana Pwns.....	61
7.2 Using Linux.....	62
7.2.1 Preparation of Secure Digital Flash Memory Card	62
7.2.2 Imaging Procedure.....	66
7.2.3 Analysis of Experiment	68
7.3 Using Homebrew Software.....	69
7.3.1 Preparation of Secure Digital Flash Memory Card	69
7.3.2 Imaging Procedure.....	71
7.3.3 Analysis of Experiment	78
Chapter 8 – Restoration of an Acquired Image.....	80
8.1 Preparation for Restoration	81
8.2 Restoration Procedure	81
8.3 Analysis of Experiment	86
Chapter 9 – Data Analysis.....	87
9.1 Creation of Test Data.....	87
9.1.1 General Use	87

9.1.2 Internet Browsing	88
9.1.3 Exchange of Messages	88
9.2 Accessing the NAND flash file system	89
9.3 Analysis of Extracted Data	91
9.3.1 Web Browsing	92
9.3.2 Internet Bookmarks / Favourites.....	94
9.3.3 Saved Contacts	95
9.3.4 Received Messages.....	97
9.3.5 Recently Sent Messages	98
Chapter 10 – Conclusions and Future Work	100
10.1 Project Achievements	100
10.2 Difficulties Encountered.....	100
10.3 Future Work.....	101
Bibliography	103
Appendix A – Console Disassembly	108
Appendix B – NAND Flash Device Datasheets.....	115
Appendix C – The WiiTools Python Script Collection	116

List of Figures

Figure 2.1: Photograph of the "Hollywood" & "Broadway" Processors	9
Figure 2.2: Photograph of a "Type A" Motherboard	11
Figure 2.3: Photograph of a "Type B" Motherboard.....	11
Figure 4.1: Photograph of the <i>Infectus2</i> PCB and USB Mini Connector	33
Figure 5.1: Diagram of a Tri-Wing Screw Head.....	44
Figure 5.2: Photograph of a 2GB Secure Digital flash memory card.....	45
Figure 6.1: The <i>Infectus2</i> PCB, USB interface & ribbon cable.....	49
Figure 6.2: Prepared <i>Infectus2</i> chip connected to the USB interface.....	50
Figure 6.3: Annotated NAND Flash Electrical Interface	52
Figure 6.4: Traced Electrical Pathways on a "Type A" Motherboard	53
Figure 6.5: Traced Electrical Pathways on a "Type B" Motherboard.....	54
Figure 6.6: Error message produced by <i>amoxiflash</i>	56
Figure 7.1: Pop-up dialog indicating the success of Bannerbomb exploit.....	66
Figure 7.2: White Linux booting and login prompt	67
Figure 7.3: Results of BootMii installer compatibility tests	73
Figure 7.4: Wii System Menu updated after Homebrew Channel installation	74
Figure 7.5: The Homebrew Channel menu	75
Figure 7.6: BootMii NAND flash backup menu screens	76
Figure 7.7: BackupMii during the imaging process.....	77
Figure 7.8: BackupMii after completion of imaging process	77
Figure 8.1: The first screen shown by the RestoreMii application	82
Figure 8.2: The warning screen displayed by the RestoreMii application.....	83
Figure 8.3: Illustration of the differences between NAND flash device images....	84
Figure 8.4: Confirmation of the success of the restoration procedure	84
Figure 8.5: Wii System Menu before (L) and after (R) the restoration process	85
Figure 9.1: Usage information for the <i>wiinandfuse</i> file system tool	91
Figure 9.2: Output of <i>grep</i> when searching for Internet Channel data	93

Figure 9.3: Cookie data stored by the Internet Channel.....	93
Figure 9.4: Favourites data stored by the Internet Channel	95
Figure 9.5: Headers of the most recently sent Wii Message Board message	99

List of Tables

Table 2.1: NAND flash device divisions	6
Table 2.2: NAND Physical Layout	6
Table 2.3: Truncated history of Wii System Menu versions	14
Table 6.1: Mapping of <i>Infectus2</i> pads to NAND Flash pins.....	52
Table 7.1: Compatibly of Wii software exploits.....	60
Table 7.2: Summary of BootMii controls.....	74
Table 9.1: Structure of Wii Address Book file.....	96
Table 9.2: Structure of Wii Friend List entries	96

Chapter 1 – Introduction

The primary aim of this project is to determine the feasibility of applying conventional digital forensic techniques to the analysis of the Nintendo Wii game console, which is typically treated as a complex embedded device or ignored altogether by forensic examiners.

This report includes a description and explanation of:

- The Nintendo Wii hardware and software platform
- The current state of game console forensic analysis, including the Nintendo Wii, Sony PlayStation 3 and Microsoft Xbox platforms
- Potential approaches to imaging the console's internal NAND flash storage device
- Investigation and extraction of data from the Nintendo Wii NAND flash file system

This chapter describes the rationale and motivation behind the project and includes an overview of its set aims and objectives.

1.1 Rationale

Recent years have seen video game consoles evolve from highly-specialised stand-alone devices into networked home entertainment hubs capable of handling streaming music, video and internet traffic in addition to their conventional game play use. This is especially true of the current generation of consoles – the Microsoft Xbox 360, Sony PlayStation 3 and Nintendo Wii – all of which provide users with permanent internal storage, message exchange mechanisms, and the ability to browse the World Wide Web.

Game consoles are designed as closed systems with a heavily-vetted set of approved software applications which greatly reduce the opportunities for misuse. However in addition to these standard capabilities, a growing number of hardware and software modifications can be exploited in order to allow the execution of arbitrary unsigned code, and in some cases, the installation and use of a non-native operating system. By combining the stability and flexibility of the Linux kernel with the powerful graphics hardware which comprises a modern game console, such non-native operating systems can turn game consoles into extremely powerful, relatively low cost, general purpose computing devices with an increasing relevance to digital forensic investigations.

Despite being the least powerful of the three current generation consoles, the Nintendo Wii is the cheapest and arguably the most popular, having sold in excess of 67 million units since its release in 2006 (1).

Taking into account the large volume of sales, there is a notable lack of published information relating to the forensic analysis of the Nintendo Wii. It is hoped that this project can provide a detailed explanation of the Nintendo Wii hardware and software platform, its operation, and the methods by which it can be exploited to reveal information to a forensic examiner; providing a springboard for future research into the forensic analysis of the device.

1.2 Aims and Objectives

The primary aim of this project is to determine the feasibility of applying conventional digital forensic techniques to the analysis of the Nintendo Wii game console.

It is hoped that this aim will be accomplished through extensive research into the current state of forensic analysis of the Nintendo Wii platform, aided by a

series of experiments designed to explore the internal NAND flash storage device which is used by the Nintendo Wii in lieu of a standard hard disk drive.

Given the relatively broad aim outlined above, the following objectives are used to judge the success of the project:

1. The development of a procedure to acquire a forensic image of the Nintendo Wii internal NAND flash storage device
2. The development of a procedure to restore a previously captured forensic image to the internal NAND flash storage device
3. The production of an intelligible representation of the Nintendo Wii NAND flash file system
4. The analysis and extraction of data from the NAND flash file system which would otherwise be inaccessible using the current “live examination” methodology

Chapter 2 – Overview of the Nintendo Wii

First released across Europe, the Americas and Japan in late 2006, the Wii is Nintendo's seventh-generation game console. The Wii is by far the least powerful seventh-generation console, lacking the advanced graphics hardware and disk-based storage of its rivals. However it is the cheapest and is marketed toward a wider, more casual gaming demographic with an emphasis placed on social interaction rather than cutting-edge graphics. As a result it is arguably the best-selling console of its generation – having sold 67.45 million units as of 31 December 2009 (1).

Nintendo have released very few details regarding the technical specification of the Wii, though unofficial reports suggest that the console is built around an updated version of Nintendo's sixth-generation GameCube platform, making the Wii fully backward-compatible while being roughly twice as powerful as its predecessor (2).

Due to Nintendo's reluctance to release technical data to the public, this project is heavily reliant on information produced by the thriving “homebrew” hardware and software hacking community which has grown around the Wii platform¹. Although the community was founded with the aims of exploring the capabilities of the hardware and making the platform more accessible to individual software developers, the obvious association with video game copyright infringement has led Nintendo to go to great efforts to hamper its use – periodically updating the Wii system software to fix vulnerabilities used by the homebrew community to enable arbitrary code execution.

As a result of the cat-and-mouse game played out between Nintendo and the homebrew community, the Nintendo Wii platform is still evolving, with system

¹ WiiBrew Wiki. Available online: http://wiibrew.org/wiki/Main_Page

software updates being released every 6-12 months with the apparent aim of blocking the execution of arbitrary unsigned code. This chapter aims to give an overview of the current state of the Wii platform, detailing hardware and software revisions which may help or hinder the forensic analysis of the device as well as detailing the boot process and general operation of the console.

2.1 Hardware

The hardware which comprises the Nintendo Wii has undergone a number of minor changes over its lifespan, the majority of which have been to the secondary circuit board and chipset which control the console's optical disc drive. It is likely that these hardware revisions were introduced as newer or cheaper fabrication methods became available, however one notable effect is that many third-party modification chips (commonly known as "drive chips") have become obsolete as the copy-protection measures built around the optical disc drive are improved.

As "drive chips" typically seek only to bypass copy-protection measures, they are primarily associated with video game copyright infringement rather than the homebrew software community, and do not come under the consideration of this project.

This section of the report aims to detail the main hardware elements which allow the Wii to operate, including the NAND flash storage device and the two main processor packages. This is followed by a brief overview of the revisions made to the console's motherboard, paying particular attention to the placement and accessibility of the NAND storage device.

2.1.1 NAND Flash Storage

The single component most likely to be of interest to a forensic examiner is the console's 512MiB NAND flash storage device which is used to system software and settings, saved games, and software such as the "Internet Channel" web browser or Virtual Console games, which can be downloaded through Nintendo's online shopping application, the Wii Shop Channel.

The NAND flash storage device is divided into 4096 Blocks, each of 8 Clusters. Each Cluster is comprised of 8 Pages, while each Page consists of 2048 bytes of data, followed by 64 bytes of error-correction and hashed message authentication codes. These divisions are illustrated in Table 2.1, while the physical layout of the NAND is detailed in Table 2.2.

Unit	Description	Length (excl. ECC)
Page	2048 bytes data + 64 bytes ECC / HMAC	2048 bytes (+ 64 bytes)
Cluster	8 * Page	16,384 bytes
Block	8 * Cluster	131,072 bytes
NAND	4096 * Block	536,870,912 bytes

Table 2.1: NAND flash device divisions

Location (Blocks)	Description
0	Contains "boot1"
1-7	Contains "boot2"
8	Beginning of per-console unique data
8-4063	Encrypted file system data
4064-4096	Unencrypted file system metadata

Table 2.2: NAND Physical Layout

It is important to note that the majority of the data stored by the NAND flash device is encrypted with a console-specific key. This means that the console-specific encryption keys must first be extracted before data from the NAND flash device can be made intelligible. This console-specific encryption of data also has the effect of preventing the direct transferral of NAND file system data between Wii consoles.

Although certain Toshiba NAND flash chips are reportedly supported by the Wii system software NAND flash driver, the most commonly utilised chips appear to be the Samsung K9F4G08U0B and Hynix HY27UF084G2M / HY27UG084G2M (3). The typical electrical interface of a NAND flash chip can be seen later in the report in Figure 6.3, while datasheets for the Samsung and Hynix NAND flash chips are included in Appendix B.

2.1.2 Secure Digital Flash Memory Card Reader

In addition to the internal NAND flash storage chip, the Nintendo Wii is equipped with a front-mounted Secure Digital flash memory card reader. Officially the maximum storage capacity of a Secure Digital flash memory card is 2GB, although Secure Digital High-Capacity cards (which are supported through a Wii System Menu update) have a maximum storage capacity of 32GB.

The Secure Digital flash memory card reader can be used to load and save console data, such as saved game files, Channels, pictures and video.

2.1.3 The “Broadway” Processor

The Nintendo Wii's central processing unit was designed and manufactured by IBM, and based around a specially adapted implementation of IBM's Power Architecture core.

Neither Nintendo nor IBM have been forthcoming in releasing technical details relating to the Broadway processor, although in keeping with the theme of backward compatibility, the Broadway processor essentially appears to be an upgraded version of the “Gekko” PowerPC processor used in the sixth-generation Nintendo GameCube, albeit with a 20% reduction in energy consumption (4).

The IBM-branded Broadway processor is located below the “Hollywood” package on the console motherboard, and is pictured in Figure 2.1.

2.1.4 The “Hollywood” and “Starlet” Processor Package

The “Hollywood” processor package was designed by ATi and functions primarily as a graphics processing unit. However in addition to its graphics capabilities, the Hollywood package also contains a full ARM9 core which has been unofficially named “Starlet” by the Wii homebrew community.

The Starlet processor manages many of the console's input/output functions, as well as controlling the wireless networking, USB, and optical disc drive hardware. This core also acts as a security controller, and is thought to contain hardware implementations of the Advanced Encryption Standard and SHA-1 cryptographic hash function.



Figure 2.1: Photograph of the "Hollywood" & "Broadway" Processors

The Starlet core also contains an area of One-Time Programmable memory which holds a number of console-specific encryption keys, including the NAND AES key which is required before the data held on the NAND flash storage device can be rendered intelligible (5).

2.1.5 Motherboard Revisions

As with the Wii's secondary circuit board and chipset, the console's motherboard has also undergone a series of changes over its lifetime. The majority of these changes involve minor shifting of components or re-routing of electrical pathways and are inconsequential as far as the forensic analysis of the device is concerned. With that in mind, this section sets out to detail only the

changes which affect the positioning and accessibility of the console's NAND flash storage device.

The motherboard variants can be split into two broad groups based solely on the layout of the NAND flash device. For the sake of simplicity these groups will be referred to as "Type A" and "Type B" from this point onwards.

The "Type A" motherboard variants appear in early model Wii consoles. While there may be minor differences between certain "Type A" revisions, it is important to note that the area surrounding the NAND flash storage device is unchanged.

Newer (including all black "Limited Edition") model Wii consoles contain a revised "Type B" motherboard.

The main difference between "Type A" and "Type B" motherboards is the orientation of the NAND flash storage device. This can be seen clearly by comparing Figure 2.2 to Figure 2.3. In addition to the change in its orientation, the electrical pathways surrounding the NAND flash device have been significantly re-routed.

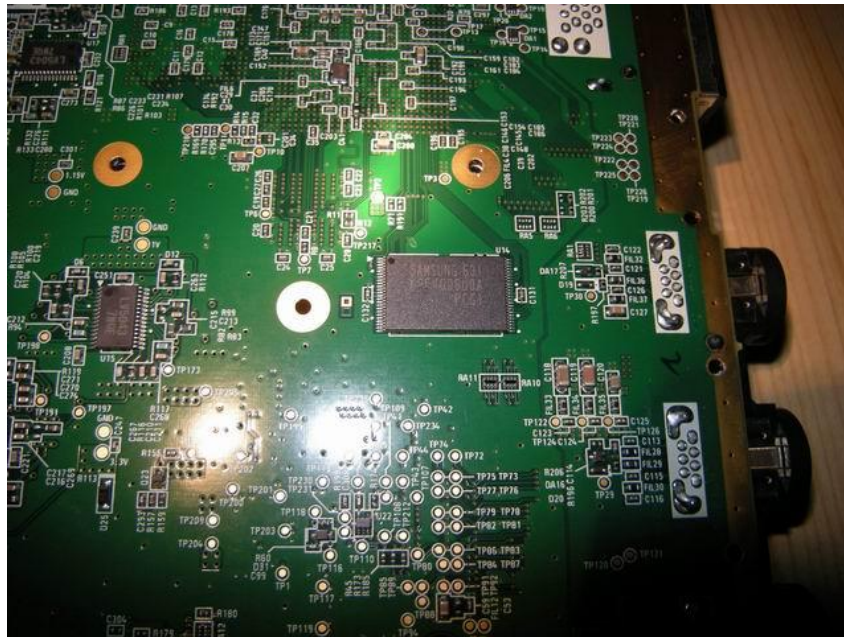


Figure 2.2: Photograph of a "Type A" Motherboard

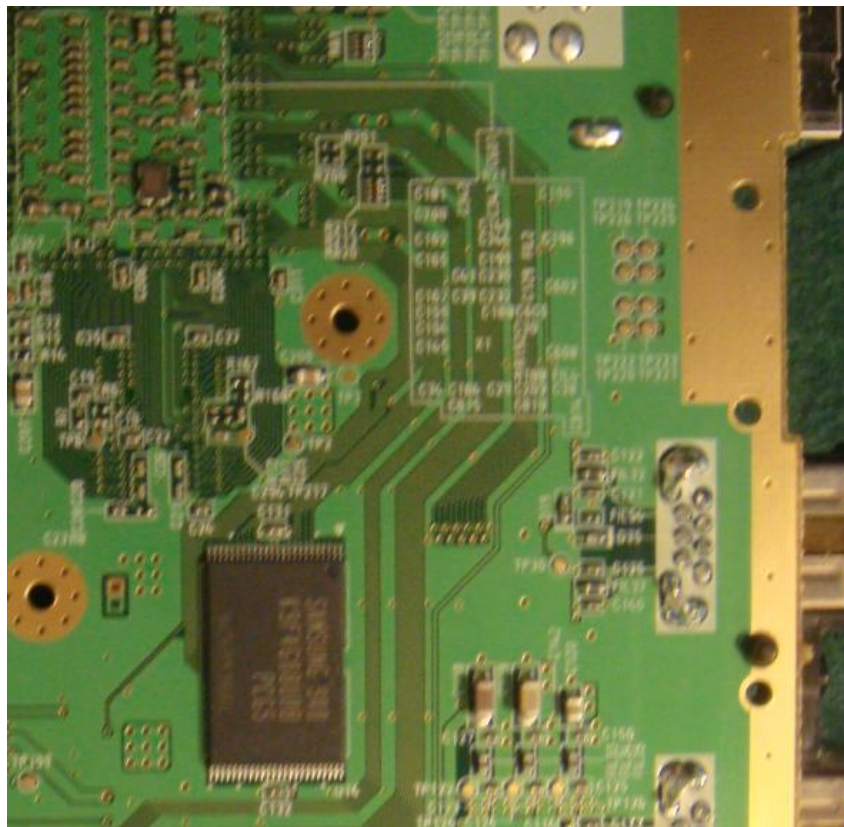


Figure 2.3: Photograph of a "Type B" Motherboard

2.2 Software

The software executed by the Nintendo Wii can be split into three broad categories:

- IOS
- The Wii System Menu
- Channels

Each of these categories of software run at different levels and performs varying functions. Briefly, IOS runs as a low-level operating system (similar to firmware), the Wii System Menu functions as a user-facing operating system, and Channels run at the application level.

This section contains a detailed examination of the categories of software likely to be found on the Nintendo Wii.

2.2.1 IOS

The term “IOS” refers to Nintendo's proprietary embedded operating system which runs on the Hollywood package's Starlet core and provides the input/output services that allow the Broadway processor and Wii software applications to access low-level functions and system devices (6).

The design of IOS is somewhat unconventional in that multiple versions are stored within the NAND file system, and there is no single definitive IOS instance. Many games and applications supply and use their own versions of IOS, which are stored in what may be termed “slots” in the NAND file system.

Although multiple versions of IOS are stored on a console only one IOS can be running at any given time. The dormant IOS versions are not aware of each other, and can be modified or removed with few repercussions so long as they are not called upon by the Wii System Menu.

2.2.2 Wii System Menu

The Wii System Menu is pre-installed on every new Wii console and provides a graphical user interface which allows the user to launch games and applications, modify system settings, and perform basic functionality such as re-booting the console and ejecting discs from the optical drive. When the console is booted the Wii System Menu is responsible for displaying a warning message and prompting the user for input before continuing to boot.

The Wii System Menu has gone through a number of iterations in the years since the console was first launched, with updates being delivered over the internet or occasionally accompanying a game on an optical disc. The majority of these updates introduce new features and bug fixes, although recent Wii System Menu updates have had the apparent aim of removing homebrew software and many of the IOS exploits used by the homebrew community (7) (8).

While it is not possible for Nintendo to force pre-existing console owners to install updated versions of the Wii System Menu, users who choose not to install updates may find that new games fail to run, or that certain online features such as the Wii Shop Channel become unavailable.

Table 2.3 shows a truncated history of Wii System Menu updates which have had adverse effects on the use of homebrew software, thus potentially reducing the options available to a forensic examiner when presented with a Wii console.

Version	Release Date	Changelog
1.0	19 November, 2006	Shipped with early Wii consoles
2.0	19 November, 2006	Enabled downloadable System Menu updates, Increased support for SD-cards
3.3	17 June, 2008	First attempt to block the "Twilight Hack"
3.4	17 November, 2008	Second attempt to block the "Twilight Hack"
4.0	25 March, 2009	Permanently blocked the "Twilight Hack", Introduced the "SD Card Menu"
4.2	28 September, 2009	Blocked the "Bannerbomb v.1" hack, Removed customised versions of "boot2"
4.3	21 June, 2010	Blocked that "Bannerbomb v.2" hack, Removed homebrew software installed as IOS

Table 2.3: Truncated history of Wii System Menu versions

2.2.3 Channels

Aside from games, the Nintendo Wii has the ability to run a wide range of applications, known as Channels.

A number of Channels are shipped with each new console alongside the Wii System Menu. These include:

- the Mii Channel, which allows the creation and limited sharing of user-generated avatars
- the Photo Channel, which enables JPEG and MJPEG images and videos stored on an SD card to be displayed on-screen
- the Wii Shop Channel is an online shop which allows users to purchase and download new content such as games and additional Channels, for example, the Internet Channel and BBC iPlayer Channel, which are downloadable at no cost

- the Forecast Channel downloads information about weather reports and forecasts and displays it to the user
- the News Channel functions in the same manner as the Forecast Channel, downloading and displaying news headlines from agencies such as the *Associated Press* or *Agence France-Presse* depending on the geographical location of the console

2.3 General Operation

Before it can be of use the Nintendo Wii must be physically connected to a mains power supply and audio/video output – typically a television. The device can be controlled with a Nintendo GameCube controller which can be physically connected to one of the circular ports on the top of the console, but is most commonly controlled with the wireless Wii Remote, also known as a “Wiimote”. The position of the Wiimote is calculated relative to a sensor bar which is generally placed above or below the display, allowing the console to accurately trace the movement of the Wiimote.

As the Wii was designed to encourage more social game playing activities, it allows the creation of avatars, known as “Miis”, which can be used to represent different users within the system.

In addition to the features commonly associated with network connectivity, the console also offers a limited picture viewing application, an RSS-type news reader, and an online shop which allows the purchase and installation of further applications, known as “Channels”.

This section details the general operation of the console, starting with the boot process before continuing to explore the operation and features believed most likely to be of interest to a forensic examiner.

2.3.1 The Boot Process

The process of booting the Nintendo Wii requires a chain of three separate bootloaders before the file system held by the NAND flash storage device can be read. This section provides an overview of each stage of the bootloader chain and describes the hardware and software components utilised throughout the process.

Although understanding the boot process may seem unrelated to the acquisition of data from the NAND flash storage device, it is very important in understanding how software is executed on the console.

boot0

The first stage of the bootloader chain is known as `boot0` and is executed upon the console being powered-up. When the power button on the front of the console is pressed the Starlet processor is powered-on.

Starlet contains instructions to read the first 48 pages of the NAND flash storage device, decrypt them with a fixed AES key, hash the decrypted pages with SHA-1, and compare the hash to a value stored within a One-Time Programmable area within the Starlet core. If the hash values do not match, the console will halt and refuse to boot. Otherwise, the bootloader chain continues to the next stage.

boot1

The second-stage bootloader is known as `boot1` and is stored in the first 48 pages of the NAND flash device which were previously extracted and verified by `boot0`.

The `boot1` code is run entirely from Starlet's on-die memory and initialises the external DDR3 memory before proceeding to read from blocks 1-7 of the NAND flash storage device. Once read, this data is decrypted and RSA verification is performed against a stored certificate. If the RSA verification procedure is a success, the code is loaded into external memory as the third-stage bootloader. If verification fails, the process halts and the console refuses to boot.

It is at the RSA verification stage that an exploitable bug first appeared in the boot process. Due to a bug involving the C programming language's `strncmp` function and its poor handling of `NULL` bytes in the RSA certificate hash, it became computationally feasible to force a SHA-1 hash collision which effectively bypassed the RSA verification procedure and allowed the execution of arbitrary code in the third-stage bootloader (9). This bug was fixed sometime in 2008, meaning that Wii consoles manufactured for sale after that time are unlikely to allow the modification of the third-stage bootloader.

boot2

The third-stage of the bootloader chain is termed `boot2` and is primarily responsible for retrieving the Wii System Menu code from the NAND file system and loading it into memory.

After retrieving the Wii System Menu code, `boot2` reads its headers to determine which particular IOS the Wii System Menu is expecting and re-loads itself. At this point the Wii System Menu is loaded into external memory and the Broadway processor is powered-on.

The Wii System Menu

Although this is the final stage of the boot process, it is the first that is actually visible to the user as the entire bootloader chain up to the completion of `boot2` takes place before the console outputs anything to the display.

The first thing to be displayed is a warning screen which prompts the user for input before continuing. At this point, the Wii System Menu performs a check of the full NAND file system in order to determine which Channels are installed, and also checks whether or not a disc is present in the optical disc drive.

Upon the selection of a Channel or content from the optical disc drive the Wii System Menu passes control to the IOS running on Starlet which decrypts, verifies, and loads the appropriate Channel code into memory before finally re-booting the Broadway processor, which executes the selection application.

2.3.2 Network Connectivity

The Nintendo Wii ships with a built-in 802.11b/g wireless networking module although a USB-to-Ethernet adaptor is available as an optional accessory. Once the console has established a connection to the internet a number of services become available to the user.

Nintendo Wi-Fi Connection

One such service is an online gaming network known as Nintendo Wi-Fi Connection. The Nintendo Wi-Fi Connection is designed to allow free online play for compatible Wii and Nintendo DS games. Although the Nintendo Wi-Fi Connection service is accessible to anyone with an internet-connected Wii or

DS console, Nintendo have introduced an authentication system which requires users to exchange and mutually register “friend codes” before certain communication and game play features become available.

These “friend codes” take the form of a twelve digit code generated from identifiers unique to the user's copy of the game and the Nintendo Wi-Fi Connection ID of their Wii console. Although some games can be played online without one, a “friend code” is required in order to play with a specific person.

The Nintendo Wi-Fi Connection has not been included within the scope of this project however a similar authentication system forms part of Nintendo's proprietary messaging system, the Wii Message Board, which is explored later in this chapter.

WiiConnect24

The WiiConnect24 system is used to allow the console to receive content via an internet connection. It is enabled by default, but can be turned off through the Wii system software. Examples of content delivered through the WiiConnect24 system include Wii Message Board messages and emails, and notification of updates to game, channel and system software.

One interesting aspect of WiiConnect24 is that it has the ability to utilise the Starlet processor to allow a network connection to be maintained while the console is in stand-by mode. This stand-by connection mode requires WiiConnect24 to be enabled in order to function, but can be disabled separately.

When examining a Wii console which appears to be turned off, it is important to note the colour of the LED contained within the power button on the front of the console. A red LED indicates that the stand-by connection is off, while a yellow

LED indicates that the stand-by connection is functioning. Another possible indication is a neon-blue glow emitted from the edges of the optical disc drive. Depending on system software settings, this light may pulsate to indicate that new data has been received while the console is in stand-by mode. However as this notification may be turned off completely, it is important to note the colour of the power LED before unplugging the console.

2.3.3 Internet Browsing

In April 2007 the full edition of a specially developed version of the Opera 9 web browser was released for download through the Wii Shop Channel (10). Known as the Internet Channel it was sold for 500 “Wii points” until September 2009, when it was updated and made available as a free download in the Wii Shop Channel.

Although the Internet Channel is now available at no cost, it is not presently distributed with new Wii consoles and is not guaranteed to be installed on any given device.

It is unclear as to whether or not the Internet Channel records a history of visited web pages. No option exists to allow the user to clear records of browsing history, although there is an option to remove internet cookies suggesting that session cookies are kept which may provide an indication of user behaviour given access to the NAND flash storage device data.

Another quirk of the Internet Channel is that although it allows the user to save visited web pages as “favourites”, the URL is not displayed when browsing the “favourites” list. This is possibly due to the Internet Channel interface being designed for viewing on a television screen rather than a computer monitor, as

thumbnail images of the saved web page are displayed along with a title instead.

2.3.4 Wii Message Board

The Wii Message Board is a software application which allows users to leave text or picture messages for other local console users and, in conjunction with WiiConnect24, allows the user to exchange email-like messages with other pre-approved consoles and standard email addresses.

In keeping with Nintendo's "friend code" concept, content from the Wii Message Board can only be exchanged with an email address or another Wii console after it has been mutually authenticated by the Wii Message Board Address Book.

At a basic level the Address Book is a list of 100 records, with each record containing a nickname and single contact address (either email or a 16-digit console identifier). When a new entry is added to the Address Book, an automated message is sent to the contact address explaining that the user of a Wii console with a specified 16-digit identifier wishes to exchange messages, and asks that the recipient reply to the automated message to confirm that they are known to the console user. This is all that is required in order to register a contact with a standard email address, however in order to register another Wii console the process must be performed on both users' consoles.

This mutual authentication procedure could potentially be of use to a forensic examiner as it requires a degree of prior contact between the console user and the contacts stored in the Address Book.

Once a contact is registered, messages can be exchanged and will appear in the Wii Message Board application as a flashing envelope icon. Using the

calendar feature of the Wii Message Board, it is possible to view envelope icons representing messages which were received any given date. While this may be impractical if there are a wide range of dates to be checked, it should be adequate if an examiner has particular dates in mind when conducting an examination.

While this approach is viable for uncovering received messages, it does not appear that the Wii Message Board application maintains records of outgoing messages, other than a note that one or more messages have been sent to a contact on a particular date.

Although it is possible to exchange messages with picture attachments between Wii consoles, the Wii Message Board application does not appear to support the exchange of pictures when one participant is using a standard email address.

Another important function of the Wii Message Board is to record and display statistics relating to the usage history of the console. Examples of this automated logging include the recipients of sent messages, as noted above, time spent using each Channel or game (e.g. 23 minutes use of the Internet Channel), and the total playing time for that day. While these statistics do not note the time of day when the actions took place, these messages cannot be deleted by a user and may be instrumental in establishing a pattern of use for the device.

Chapter 3 – Literature Review

This chapter aims to provide an overview of the current state of the field of game console forensic analysis, detailing current best practice and newly proposed approaches to the forensic analysis of the Microsoft Xbox and Xbox 360, Sony PlayStation 3, and Nintendo Wii platforms.

3.1 Microsoft Xbox & Xbox 360 Forensics

The Microsoft Xbox 360 was released in late 2005 and heralded the arrival of the current generation of network-capable, multi-function game consoles. It features a built-in Ethernet interface, although an 802.11b/g wireless network adapter is available as an option. Depending on the model, the console is likely to make use of a removable hard disk drive contained within a proprietary enclosure.

An extensive search of digital forensics journals and conference proceedings uncovered only a single peer-reviewed paper directly related to the forensic analysis of the Xbox 360 (11). The paper by Xynos et al. revisits and updates much of the previously published work relating to the original Microsoft Xbox console and presents a detailed examination of the Xbox 360 file system and networking capabilities.

Although contained within a proprietary enclosure, it appears that the hard disk drive can be removed from the console and connected to a PC via a standard SATA interface (12).

The Xbox 360 hard disk drive appears to use a modified version of the standard FAT file-system, and can be explored using a free, but closed-source, software

application or a proposed toolkit developed with forensic analysis in mind (13) (14).

Although it appears that there is a relative lack of published research into the forensic analysis of the Xbox 360 this is not the case for its predecessor – the Microsoft Xbox.

The original Xbox was released in late 2001 and finally discontinued in 2006 after the release of the Xbox 360. Unusually for its time, the Xbox shipped with an internal hard drive and Ethernet adaptor. As a result the platform quickly attracted the attention of hobbyists who set to work bypassing its security restrictions; essentially turning the Xbox into a low-end personal computer (15) (16).

Although the Xbox uses a standard IDE hard disk drive it is “locked” by a little-used protection scheme included in the ATA specification, and utilises a cryptographically generated password created from elements of the console and hard disk drive model and serial numbers (17).

Fortunately the act of booting the Xbox “unlocks” the hard drive, allowing the IDE cable to be replaced with a hardware write-blocker in order to create a forensic image of the drive which can then be taken away for examination (18).

Memory analysis is commonly overlooked when dealing with the forensic examination of game consoles. One published approach involves exploiting a buffer overflow vulnerability in a saved game file in order to execute code which captures an image of virtual memory (19). Although the technique is in need of refinement, it is possible that it can be modified to capture a physical storage device and generalised to work on a wide variety of embedded systems.

3.2 Sony Playstation 3 Forensics

The Sony PlayStation 3 is the most technically advanced of the three current-generation consoles. Though it has undergone several changes and updates since its initial release in late 2006 each model contains an Ethernet, and in some cases, 802.11b/g network adaptor and a 2.5" SATA hard disk drive which can be upgraded by the user.

One unique feature of the PlayStation 3 was option which allowed users to partition the hard disk drive in order to install a secondary – typically Linux-based – operating system on the console without the need for unauthorised modifications. This “OtherOS” was subject to restrictions placed on partition size and lacked direct access to certain hardware components, however the officially-supported feature proved to be popular with hobbyists and those wishing to harness the power of the multi-core Cell processor on which the PlayStation 3 is built. It can be argued that the “OtherOS” increased the scope for misuse of the console by effectively turning it into a general-purpose Linux desktop personal computer, but alternatively it has been credited with helping to maintain the security of the platform, as providing an officially sanctioned method of installing a Linux-based operating system removes one of the primary motivators for hardware and software hackers.

This feature was notably absent from the re-designed “Slim” model PlayStation 3 which was released in late 2009, and was disabled on all PlayStation 3 consoles by a downloadable firmware update released in April 2010 (20).

As with the Microsoft Xbox 360, there appears to be a lack of published material relating to the forensic analysis of the PlayStation 3, although a mock investigation involving two suspects trading illicit data over the internet using PlayStation 3 consoles was the focus of the 2009 Digital Forensics Research

Workshop Challenge, leading to a number of challenge submission reports being made available on the DFRWS website².

Although the PlayStation 3 hard disk drive can be removed from the console and connected to a write-blocker for imaging, the security measures put in place by Sony result in the data contained on hard disk drive being encrypted with a console-specific encryption key. This security measure also means that a hard disk drive taken from one PlayStation 3 console will fail to function when an attempt is made to use it in a second console (21). It is worth noting that this encryption applies only to the main PlayStation 3 system software partition and, as a result, any "OtherOS" partition which may exist can be carved from the hard disk drive image and examined with standard forensic investigation software (22) (23).

Due to the encryption of the main PlayStation 3 system software partition, an investigation method has been proposed which combines conventional image-based forensic investigation with "live examination" by removing and imaging the console's hard disk drive before copying the image to a new hard disk drive which is then inserted into the console. This hybrid approach allows an investigator to obtain potential evidence from the encrypted partition by booting the suspect console and viewing data through the PlayStation 3 system software without having to risk the destruction or inadvertent modification of the data. (21).

While the security measures put in place by Sony appear to have held firm since the initial release of the console in late 2006, there are signs that cracks are beginning to appear, potentially heralding the development of a new homebrew software community around the PlayStation 3 platform. It is interesting to note that the "OtherOS" feature was disabled by downloadable

² DFRWS Challenge 2009. Available online: <http://www.dfrws.org/2009/challenge/index.shtml>

firmware update shortly after the publication of an exploit thought to provide access to the console hypervisor through some facet of the "OtherOS" feature (24).

Another potential PlayStation 3 exploit was reportedly made available in August 2010, utilising a USB "dongle" in order to allow the execution of unsigned code (25). While independent confirmation of this exploit is unavailable, the ability to execute arbitrary unsigned code on the console could potentially allow the development of a forensic application capable of capturing an intelligible copy of the data contained on the encrypted system partition.

3.3 Nintendo Wii Forensics

Despite selling in excess of 67 million units, it appears that only one peer-reviewed paper covering the forensic analysis of the Nintendo Wii has been published: Benjamin Turnbull's "Forensic Investigation of the Nintendo Wii: A First Glance" (26).

In his paper, Turnbull argues that the Wii's relative lack of power and storage capacity make it the least attractive target for misuse amongst the current generation of game consoles. Additionally, the lack of disk-based storage presents a problem in that traditional methods of evidence acquisition are not applicable. These factors, coupled with the misconception that game consoles are secondary sources of evidence may lead an examiner to conclude that the Nintendo Wii is not worth the trouble of developing new investigatory techniques.

The lack of a hard drive does not necessarily mean that there is no data to be recovered from the console. The Nintendo Wii includes a flash memory card reader/writer which can be used to transfer saved games and user data to a

Secure Digital flash memory card. Additionally the internal NAND flash device presumably contains data created by the automated usage logging feature, as well as interactions with the Internet Channel and Wii Message Board applications.

While it does not seem to be possible to remove the automated logging messages through the standard Wii system software, Turnbull suggests that homebrew software such as a non-native operating system may bypass the logging feature entirely.

Due to the difficulty of accessing the data stored on the Nintendo Wii's internal NAND flash device in a forensically-sound manner, Turnbull proposes a methodology for the "live examination" of a suspect console, with the examiner's actions recorded by a video camera or screen capture device.

While a "live examination" procedure is not ideal due to the inherent risk of modifying or destroying potential evidence, the good practice guidelines for handling computer-based electronic evidence set out by the Association of Chief Police Officers do allow for such procedures, but only if absolutely necessary and all changes made can be explained and accounted for by the examiner (27).

Turnbull's contains a full walkthrough of the proposed "live examination" process, including the examination of the Wii System Menu version, the date and time settings, network settings and the status of automated usage logging and Wii Message Board messages. The Mii Channel application is also highlighted by Turnbull as a potential area of interest to an examiner. The Mii Channel contains any user-created avatars, and may provide an indication of the number of people who had regular interactions with the console.

Concerning the lack of recorded web browser history data in the main system software Turnbull theorises that this may be due to the lack of spare capacity on

the internal NAND flash storage device. He also notes that there is no method of examining the true location of a web page saved as a "Favourite" without following it. As a workaround, he suggests the use of a packet sniffing device on the network, the records of which can then be examined to determine the true location of the saved resource.

Turnbull's paper also illustrates the point that game consoles can no longer be viewed as static platforms. The inclusion of a network connection means that system software can be forcibly updated by the vendor, either to add new functionality or to patch security flaws which could be exploited to execute unsigned code. This point is illustrated by the growing availability of the Internet Channel application, which required once an extra payment but is now available as a free download through the Wii Shop Channel.

While investigating the possibilities for hardware-based imaging of the internal NAND flash storage device, an approach was uncovered which uses the Joint Test Action Group (JTAG) test port to directly access and image the memory chips of an embedded device, writing the data to an analysis workstation via a parallel port (28). While the procedure as set out in the paper is not specifically tailored to the Nintendo Wii, it is implied by the authors that it could be adapted to function on any embedded system with an accessible JTAG port. Unfortunately however, preliminary research undertaken as part of the project indicated that no such test port is accessible on the Nintendo Wii, leading to the abandonment of this proposal.

An alternative hardware-based imaging approach does away with the need for a JTAG port by attaching an external NAND flash controller to the console and using it to read data directly from the internal NAND flash storage device (29). This technique was originally developed by a member of the Nintendo Wii hardware hacking community who was seeking to develop a method to repair corrupted Wii NAND flash file systems, but may be formalised to produce a

forensically-sound process for acquiring an image of the internal NAND flash storage device.

It was decided during the early stages of the project that if such a process of capturing an image of the internal NAND flash storage device proved successful, a method of restoring a previously captured NAND flash device image would be sought. If such a restoration procedure was developed, it may result in the a “hybrid” investigative methodology similar to that proposed by Conrad et al. for the Sony PlayStation 3, where Turnbull’s “live examination” procedure can be carried out on a suspect console, which can later be restored to its “seized” condition if necessary.

Chapter 4 – Proposed NAND Flash Imaging Methods

One of the main objectives of this project is to determine whether or not it is feasible to create a block-for-block replica of the data stored on the Nintendo Wii's internal NAND flash storage device. The Nintendo Wii differs from its seventh-generation competitors in its lack of a removable primary storage device. Unlike the Microsoft Xbox 360 and Sony PlayStation 3 which both utilise standard SATA hard disk drives, the Wii makes use of a NAND flash storage device which is soldered to its motherboard. This lack of removable primary storage leaves a forensic examiner with the choice to either treat the Nintendo Wii as a complicated embedded device, or to develop a novel method of acquiring a forensic image of the data stored within the console.

This chapter of the report introduces three proposed methods of creating a forensically-sound copy of the data held by the NAND flash storage device, one involving the use of hardware and disassembly of the console, and two which run entirely in software and require no hardware modifications whatsoever.

4.1 Using Hardware

As it was believed that software-based approaches to imaging the Nintendo Wii internal NAND flash storage device are typically reliant on flaws in the Wii System Menu and IOS, a decision was taken to first attempt to develop a hardware-based imaging method which was presumed to be more robust when faced with Wii System Menu security updates.

This section describes the pair of proposed hardware-based approaches to accessing the data stored by the internal NAND flash device. The first involving the low-level manipulation of data buses using a JTAG test port, while the second operated at a higher level, utilising an external NAND flash controller.

4.1.1 Joint Test Action Group (Boundary-Scan)

The Joint Test Action Group standard, also known as boundary-scan, was defined in 2001 and provides an extra port or set of electrical pads which can be used to perform various testing and debugging functions on electronic components and embedded software applications (30). While boundary-scan is typically of most use during development and test cycles, many consumer electronic items ship with JTAG-enabled Printed Circuit Boards left intact, though not usually directly accessible to the end user.

The possibility of using JTAG to manipulate the data buses associated with the Nintendo Wii's NAND flash storage device was first raised during early discussions between the author and project supervisor. It appeared that the approach set out by Breeuwsma (28) to image the SDRAM chips of embedded devices could potentially be adapted to function on the Nintendo Wii console however, as early research indicated that the Nintendo Wii motherboard was not JTAG-enabled, this proposal was discarded.

4.1.2 The *Infectus2* NAND Flash Controller

As it became apparent that JTAG access to the NAND flash storage device was not feasible, attention was shifted to finding a suitable NAND flash controller which would allow access to the NAND flash storage device in situ.

The *Infectus2* external NAND flash controller (Figure 4.1) was originally developed and marketed as a universal game console modification chip (31). It features a fully-programmable core and Universal Serial Bus interface for communication with a standard personal computer.



Figure 4.1: Photograph of the *Infectus2* PCB and USB Mini Connector

Due primarily to their flexibility the *Infectus* and its successor the *Infectus2* are commonly used and well supported by the Nintendo Wii homebrew and wider game console hacking communities. This support extends beyond the *Infectus* hardware itself to the software applications which can be used to reprogram and operate the chip (29).

It was proposed that an *Infectus2* chip operating as an external NAND flash controller be attached to the NAND flash storage device mounted on the Nintendo Wii motherboard and, through software running on a desktop personal computer, instructed to read data directly from the internal NAND flash device which could then be passed to the desktop computer via USB connection.

4.2 Using Software

While it was presumed that an imaging process utilising an external NAND flash controller was more likely to provide an exact copy of the data stored by the console's internal NAND flash device, the identification of a large number of potential difficulties which may have arisen during the process resulted in the exploration of software imaging methods which could be applied without the need for specialist skills and equipment.

The first software approach to be proposed involved the use of a specially adapted Linux distribution, while the second made use of software developed by the Wii homebrew community.

Although only one of these approaches was specifically labelled as homebrew software, it should be noted that in order to execute arbitrary code, either the Wii System Menu or IOS must first be exploited to allow these software applications to be loaded into memory. A more detailed examination of the publicly available exploits for Nintendo system software vulnerabilities can be found on page 59.

4.2.1 Linux Approach

Since its initial release in 1991, operating systems based around the Linux kernel have been written and adapted to operate on a multitude of different processor architectures and embedded devices. As the hardware used by game consoles evolved to the point that it was almost indistinguishable from a standard desktop personal computer, it was only a matter of time before an attempt was made to port the Linux kernel to these relatively cheap, powerful computing devices.

In addition to the homebrew software community, the Nintendo Wii also sports an active Linux development community which has its roots in the Wii's predecessor, the Nintendo GameCube³.

The main achievement of the Wii and GameCube Linux communities is the release of full, stable implementation of a Linux operating system. Based around a modified kernel image and the *Debian Lenny* file system, White Linux is capable of running entirely from an SD card and system memory in a similar manner to a standard live-cd environment (32).

While the exact level of access provided by White Linux to the low-level hardware contained in the Wii was unknown, it was presumed that read access to the NAND flash storage device would allow the use of standard Linux-based disk imaging applications such as `dd` and `netcat`, negating the need for the development of special-purpose imaging software.

4.2.2 Homebrew Software Approach

As part of an effort to simplify the installation and use of homebrew software on the Nintendo Wii, a group of developers known colloquially as “Team Twiizers” have created and released an application called “BootMii” which acts as a customised bootloader and is designed to replace `boot2`, allowing full control of the Wii hardware without interference from the Wii System Menu (33).

As the majority of the software developed by the homebrew community is unauthorised and not supported by Nintendo, the creators of BootMii advise homebrew software users to create a full back-up copy of the data stored by their console's NAND flash device to be used in case of system failure (caused

³ GameCube Linux Wiki. Available online: http://www.gc-linux.org/wiki/Main_Page

either by an official Nintendo Wii System Menu update or by poorly coded homebrew software). In order to ease this process of backing-up and restoring data, the BootMii software package contains two data imaging applications known as “BackupMii” and “RestoreMii” respectively.

Research conducted during the early stages of project led to the conclusion that the image created by the BackupMii application could be considered to be a full block-for-block copy (including Error-Correcting Code) of the data stored by the NAND flash device. As the NAND flash storage device is encrypted, so the resulting image would also be encrypted. Unlike the proposed White Linux approach however, the BackupMii application also has the ability to extract the console-specific encryption keys from the One-Time Programmable area of the Starlet processor. It was presumed that the knowledge of these encryption keys would allow the data contained in the NAND flash image to be made intelligible, potentially allowing access to the contents of the underlying file system.

The RestoreMii application may be dismissed as a secondary concern, but the demonstration of a method of restoring a captured NAND flash image to the console may give a forensic examiner the confidence to perform a full “live examination” without having to be overly concerned about the destruction of potential evidence.

4.3 Evaluation of Proposed Methods

This section examines the advantages and disadvantages of each of the proposed imaging methods described above. Issues which could potentially reduce the integrity or feasibility of the proposed approaches are explored along with possible mitigating factors and solutions.

4.3.1 Potential for NAND Data Modification

It was presumed that the use of an external NAND flash programmer chip such as the *Infectus2* would provide the most accurate image of the state of the internal NAND flash storage device as it was at the time it came into the possession of the examiner. Similarly, it was presumed that the proposed Whiite Linux approach also had the potential to create an accurate image of the NAND flash storage device although as with a standard “live imaging” technique, it was thought likely that the data stored by the internal NAND flash device would be altered by the mere act of booting the console in order to exploit the Wii System Menu or IOS and load the Linux environment into memory.

While the Whiite Linux approach would be run from system memory and a Secure Digital flash memory card, the BootMii application package must be installed on the target console before BackupMii can copy the data held by the NAND flash storage device. Unlike the Whiite Linux approach which was thought likely to make changes only to the daily use and playing time statistics tracked by the console, the fact that BootMii requires installation to the internal NAND flash storage device meant that there exists a potential to overwrite data which may have been of value to an investigation.

While relying on an imaging method which made changes to the data being collected was far from ideal, a number of ways of reducing and quantifying any changes were proposed.

The impact of changes made by the exploitation of Wii system software could be reduced by following Turnbull's suggestion of not beginning an analysis of the console until it had been turned off and in the possession of the examiner for at least 24 hours, as this would isolate the recording of the examiner's actions in a new “day” rather than potentially modifying valid records of user actions.

Although care must be taken to avoid corrupting the automated logging data associated with user actions, in cases involving older consoles it is possible to predict with near certainty the location of changes which would be made to the NAND flash data. As the BootMii package was designed as a replacement for the `boot2` stage of the bootloader chain, it attempts to copy itself to blocks 1-7 of the NAND flash storage device, where it would automatically be loaded by `boot1`. The `boot2` data is not accessible to the user during normal console use and is only ever consulted by `boot1` during the console start-up process. With this in mind, it is advised that blocks 1-7 of the acquired image be kept outside of the scope of any investigation, although as these blocks are reserved for use by the bootloader chain the effects of this restriction should be minimal.

It must be noted however that in most consoles manufactured since 2008, it is not currently possible to install the BootMii package as a `boot2` replacement. This is due to a hard-coded update which fixed the RSA certificate hash-collision bug which existed in early versions of `boot1` and allowed arbitrary modification of `boot2`. As a result of this update the BootMii package should detect that `boot2` cannot safely be modified, and instead install itself as an IOS in an unused slot. During installation the BootMii IOS will be placed in the first empty IOS slot, counting down from 254. The result of this is that BootMii IOS should typically be installed as IOS 254 however future Wii System Menu updates by Nintendo have the potential to fill arbitrary IOS slots and possibly alter the behaviour of the BootMii installation package.

4.3.2 Requirement of Specialist Skills or Tools

The main advantage of the proposed software-based imaging methods was thought to be that no specialist skills or tools were required in order to acquire an accurate copy of the data stored by the internal NAND flash device. It was

presumed that a Linux environment with access to the NAND flash device would also have access to the console's wireless network adaptor, allowing any captured image to be passed to a remote host via `netcat` or similar application. The BackupMii application requires only that an SD flash memory card with enough free space be inserted into the card reader on the front of the console. The back-up copy of the internal NAND flash memory should be written to the SD card and later be transferred to a standard desktop PC for analysis.

The same presumption cannot be made for the hardware-based approaches proposed above. External NAND flash controllers are highly specialised pieces of hardware, and may not be easily available. The *Infectus2* chip used in this project was purchased over the internet and imported from North America, as very few of these chips appeared to be available for sale in the United Kingdom. Although other methods of reading NAND flash storage chips exist, the *Infectus2* was chosen due to two main factors. Firstly because of its support for accessing NAND flash chips in situ, removing the need to de-solder and remount the chip in a TSOP-style adaptor, and secondly because the *Infectus* and *Infectus2* chipsets are widely supported in the Wii homebrew and wider game console hacking communities.

Before the *Infectus2* chip could be used a number of leads and a momentary push-button switch had to be soldered to the correct pads on the *Infectus2* Printed Circuit Board. These pads are large enough that they should present no difficulties to any person with a small amount of previous soldering experience, but the fact that the *Infectus2* requires any soldering at all may be enough to discourage its use altogether.

In addition to the potential difficulties in acquiring a suitable external NAND flash controller, the proposed hardware-based imaging method required direct physical access to the internal NAND flash storage device, which is soldered to

the back of the console's motherboard. Accessing this chip required the full disassembly of the Nintendo Wii, which is presumed to be a time-consuming task and is known to require the use of a relatively uncommon Tri-Wing screwdriver.

While it was considered likely that any person with basic soldering skills could prepare the *Infectus2* chip, the same could not easily be said for the process of securing these leads to the appropriate electrical pins of the internal NAND flash storage device. A solder-less approach which utilised the electrical vias was thought to be the safest and most accessible method, but would likely require a great deal of dexterity and wire of exactly the right size to ensure consistent electrical contact with the NAND flash storage device's electrical pathways and pins. Using solder to secure the leads to the appropriate vias would provide a better quality circuit however this would require more advanced soldering skills and a fine-tip soldering iron. A third proposal involved the attachment of the leads directly to the appropriate electrical pins of the NAND flash storage device. While this direct soldering approach was likely to ensure the best possible electrical contact, the high level of difficulty and risk of damage to the hardware meant that it was not considered feasible in the scope of this project.

4.3.3 Robustness of Approach

One of the main concerns which faced the proposed software-based imaging methods was their reliance on the ability to exploit flaws in the Wii System Menu or IOS. At present this is not an issue, as every version of the Wii System Menu is exploitable in one way or another however with each Wii System Menu update Nintendo fix bugs and reduce the number of publicly available exploits which allow the execution of unsigned code.

The Wii System Menu was updated to version 4.3 on 21 June 2010 (34). This update fixed a number of bugs including a buffer overflow which had previously

allowed a malformed Channel banner to be used to launch an arbitrary executable from an SD flash memory card. As a result of Nintendo's efforts to secure the Wii software platform, console owners who chose to install the Wii System Menu update (or purchased a console with the 4.3 update pre-installed) were unable to install the BootMii application package until it was updated by its developers. In the case of the Wii System Menu 4.3 update, it took developers 35 days to circumvent the new security measures and release an updated BootMii installation package.

While consoles which use current Wii System Menu versions will continue to be exploitable, any future Wii System Menu update has the potential to remove the bugs which allow unsigned code to be executed on the console, making software NAND flash device imaging techniques vulnerable to forced obsolescence unless officially sanctioned by Nintendo.

As hardware updates are far more difficult to force onto pre-existing consoles, and the interface with a component as fundamental as the internal NAND flash storage device is unlikely to be altered in new hardware iterations, a hardware-based approach to NAND flash imaging was considered to be the most robust approach barring any new method officially sanctioned by Nintendo for public use.

4.4 Proposed NAND Flash Imaging Guidelines

Given the issues raised in this chapter, the following guidelines were proposed to form the basis of a number of experiments to determine whether or not a forensic image of the Nintendo Wii's internal NAND flash storage device could be obtained.

1. The console would be fully disassembled and an image created using an *Infectus2* NAND flash controller before re-assembling the console.
2. In the event of the failure of the *Infectus2* procedure, an attempt would be made using Whiite Linux to acquire an image with `dd`.
3. As the *Infectus2* procedure will not extract the console-specific encryption keys needed to access the NAND file system, BootMii should be installed and BackupMii used to obtain a new forensic image and dump of the console-specific encryption keys.

It was proposed that a combination of imaging methods be used, as once a forensic image of the console NAND flash storage device has been obtained, it could be used in conjunction with the console-specific encryption keys in an attempt to access the encrypted NAND flash file system.

Chapter 5 – Equipment and Experimental Methodology

This chapter of the report details the equipment used throughout the project as well as providing an overview of the methodology used when designing and scheduling the various imaging experiments.

5.1 Equipment

The Nintendo Wii console is arguably the most important piece of equipment used during this project. The proposed experiments were carried out on a black, PAL, “Limited Edition” Nintendo Wii, which was purchased over the internet in late May 2010 for £129.99. The console was bundled with a single Wiimote, the Wii MotionPlus Wiimote accessory, and the Wii Sports and Wii Sports Resort games.

Despite the console’s “Limited Edition” moniker, the disassembly process confirmed the initial impression that the only difference between it and the standard new-model Wii console is that the console body and controller are black rather than white. Importantly, the console was built upon a standard Type B motherboard, and shipped with Wii System Menu version 4.2 installed by default. Due to the lack of homebrew software compatibility with Wii System Menu version 4.3 in the weeks following its release in late June 2010, the console’s system software was not updated from version 4.2.

To assist in the disassembly of the console a Tri-Wing screwdriver was purchased over the internet for £2.70. The Tri-Wing screw is rarely seen and thus used primarily as an anti-tampering measure. A diagram of the head of a Tri-Wing screw can be seen in Figure 5.1.

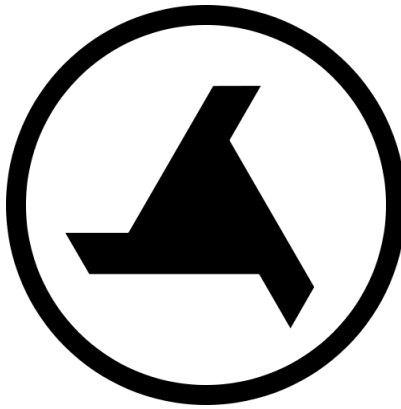


Figure 5.1: Diagram of a Tri-Wing Screw Head

An *Infectus2* external NAND flash controller was acquired for use as part of the hardware-based NAND flash imaging experiment. After an extensive search it appears that the *Infectus2* chip is no-longer being manufactured on a large scale. As a result, the chip used in this project was purchased over the internet from a North American retailer specialising in game console modifications.

A 2GB Secure Digital flash memory card was used to introduce software and transfer data to and from the console (Figure 5.2).

In addition to the Wiimote which shipped with the console, a standard USB keyboard was used to allow input when conducting the Linux-based software imaging experiment, and a Nintendo GameCube controller was used to enter the confirmation code in the homebrew software NAND image restoration experiment.

The extraction and analysis of data from the internal NAND flash device images was performed on a VirtualBox virtual machine installed with Ubuntu 10.04 Desktop Edition Linux and running on a MacBook Pro host machine.



Figure 5.2: Photograph of a 2GB Secure Digital flash memory card

5.2 Experimental Methodology

In order to accomplish the main aim of the project and determine the practicality of acquiring a forensic image of the console's internal NAND flash device, three experiments were proposed. These imaging experiments are grouped into hardware-based and software-based methods, and are fully described in Chapter 6 and Chapter 7 respectively.

As part of an attempt to ascertain the extent of changes made to the internal NAND flash storage device during the setup and continued use of the console, the hardware-based imaging experiment was conducted prior to powering-on the console for the first time. It was hoped that an image recording the internal NAND flash device in its "out of the box" state would be useful in assessing the changes which occur through normal use as well as providing the option to reset the console to its factory state for use in future projects.

After the completion of the hardware imaging experiment, the console was reassembled, booted in order to complete the initial setup procedure, and a set

of test data was created. The test data set formed the basis of data analysis efforts described in Chapter 9.

Due to the unsatisfactory results of the *Infectus2* and White Linux NAND flash device imaging experiments, the original methodology was altered slightly and the BootMii homebrew software package was installed prior to creating the test data set. The successful outcome of the BootMii-based NAND imaging experiment resulted in the creation of a NAND flash device image while the console was in a state which was as close to the “factory condition” as was possible.

The BootMii imaging experiment was conducted multiple times during the console setup process, as noted in the list of steps below:

1. Nintendo Wii console removed from box and disassembled.
2. Failure of *Infectus2* NAND imaging experiment. Console re-assembled.
3. Console booted and initial setup completed.
4. Failure of White Linux NAND imaging experiment.
5. Installation of BootMii. Successful BootMii NAND imaging.
6. Console configured to join 802.11b/g wireless LAN.
7. Successful BootMii NAND imaging.
8. Installation of the Internet Channel.
9. Successful BootMii NAND imaging.
10. Creation of test data and BootMii NAND imaging as required.

The test data set was created through daily use of the console, including game play, Internet Channel browsing (including use of web-based email services), and exchanging messages and images to remote Wii consoles and external email addresses through the Wii Message Board application.

A final experiment was planned in order to gauge the feasibility of restoring a forensic image to the console's internal NAND flash device. While this

experiment was dependent on the success of at least one of the three NAND flash imaging methods, it was hoped that the successful restoration of a forensic image could reduce the risks associated with the live examination of the console. Due to the potential risks involved with writing data to the internal NAND flash storage device, this experiment was only conducted upon the completion of all other imaging and data analysis tasks.

Chapter 6 – Hardware-based Imaging Experiment

This chapter describes the procedure followed in attempting to acquire a forensic image of the data held by the Nintendo Wii's internal NAND flash storage device. The description begins with an overview of the preparation of the *Infectus2* programmable NAND flash controller, before explaining the possible methods of connecting the *Infectus2* to the internal NAND flash device and reading its contents.

It should be noted that before conducting this experiment, the complete disassembly of the console was required. Full instructions as to the disassembly of the console are included in Appendix A.

6.1 Preparation of the *Infectus2* Chip

The *Infectus2* was designed to function as a fully-programmable universal game console modification chip. Its fully-programmable nature has led to it becoming extremely well-supported by both the Nintendo Wii and wider game console hardware hacking communities. The *Infectus2* chip is shipped on a printed circuit board measuring approximately 40x45mm, with a separate USB Mini interface which can be connected using the supplied flat ribbon connector cable (Figure 6.1).

Before the *Infectus2* chip could be attached to the internal NAND flash device, it was necessary that a number of wires be soldered to the appropriate solder pads on the *Infectus2* printed circuit board.

There are sixteen wires which must be soldered to the command and data pads on the right hand side of the *Infectus2* printed circuit board. These wires will form the bulk of the connection between the *Infectus2* chip and the console's NAND

Although the command, data, power and ground wires are all that are strictly necessary for the Infectus2 chip to read from the internal NAND flash device, it is useful to include a mechanism which prevents the console from modifying its

contents during the imaging process. This was accomplished by adding an extra connection from `Data0` to ground through a momentary push button switch. This switch is pressed and held closed in the seconds before and after the console power button being pressed. The resulting short circuit prevents `boot1` from being read from the NAND flash storage device, thus halting the boot process while still maintaining the power supply to the internal NAND flash storage device.

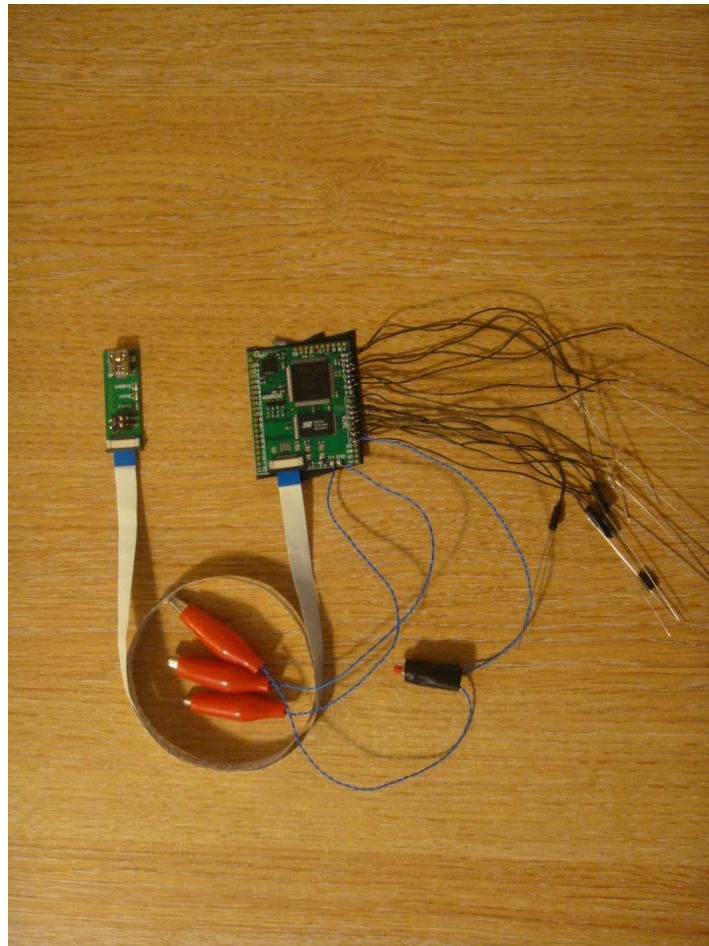


Figure 6.2: Prepared *Infectus2* chip connected to the USB interface

The final stage of preparation involved affixing a layer of insulating tape to the rear of the *Infectus2* printed circuit board in order to prevent inadvertent

contact between the *Infectus2* package and components on the console's motherboard.

A photograph of the fully prepared *Infectus2* device is included as Figure 6.2. It should be noted that the needles which are soldered to the ends of the command and data wires were attached in an attempt to ease the solder-less connection of the wires to the electrical vias.

6.2 Imaging Procedure

The Nintendo Wii's internal NAND flash storage device is soldered to the rear side of the motherboard. When the console is viewed from the front, the motherboard itself is set vertically along the left hand side. Unfortunately this placement means that the console must be completely disassembled before access can be gained to the NAND flash storage device, a time-consuming task which requires the use of both Philips and Tri-Wing screwdrivers. Full instructions as to the disassembly of the console are included in Appendix A.

Once the console was disassembled, the command and data wires which had previously been soldered to the *Infectus2* package were attached to the appropriate electrical vias on the console motherboard. As noted on page 40, a number of attachment techniques were initially considered, including soldering the wires to the electrical vias, and soldering directly to the pins of the NAND flash storage device. Due to the risks associated with the hand-soldering of small-scale electronic components, it was decided that the stability of the connection would be sacrificed in order to minimise the potential damage to the console components.

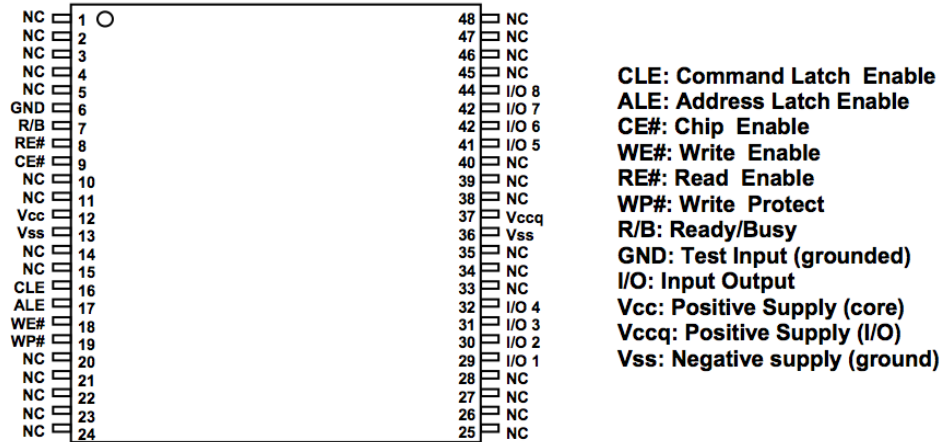


Figure 6.3: Annotated NAND Flash Electrical Interface

<i>Infectus2</i> Pad	NAND Flash Device Pin	<i>Infectus2</i> Pad	NAND Flash Device Pin
U	7	0	29
M	8	1	30
N	9	2	31
V	12	3	32
O	16	4	41
P	17	5	42
Q	18	6	43
T	19	7	44

Table 6.1: Mapping of *Infectus2* pads to NAND Flash pins

Following from this decision, the next course of action was to insert the loose end of each of the command and data wires into the electrical via connected to the appropriate pin of the NAND flash device. A diagram and table showing the electrical interface and mapping of connections between the *Infectus2* and NAND flash storage device are included as Figure 6.3 and Table 6.1 respectively.

The layout of the electrical vias differs greatly between Type A and Type B motherboards, but by starting at the correct NAND flash storage device pin and tracing the electrical pathways, it was possible to determine the electrical via

The insertion of wires into the appropriate vias is a task requiring a great deal of luck and manual dexterity. It was originally proposed that American Wire Gauge size 30 "wire wrap" be used to form the required connections however early testing indicated that smaller wire was needed, and a second attempt was made with wire obtained by splitting an IDE hard drive ribbon cable into its component strands.



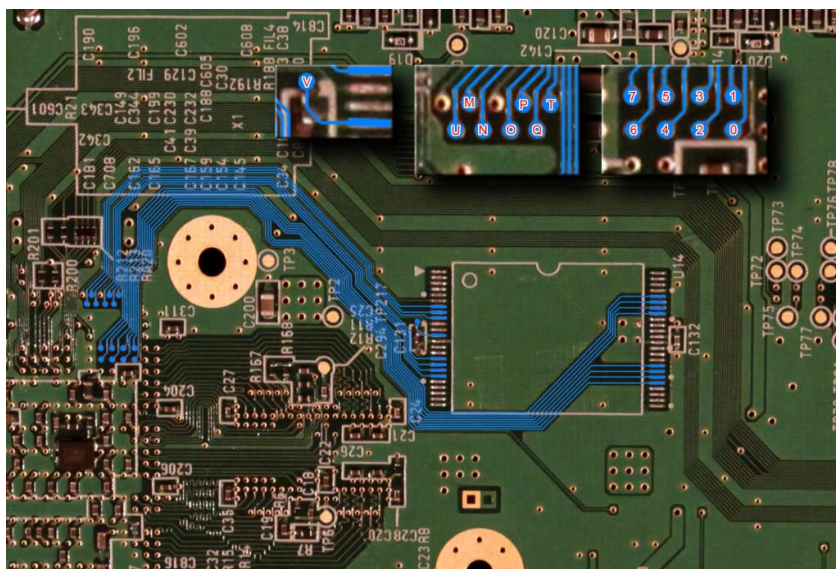


Figure 6.5: Traced Electrical Pathways on a "Type B" Motherboard

After a discussion involving the author, project supervisor and a member of the Lothian and Borders Police Forensic Computer Unit, it was proposed that an approach commonly used in the manufacture and testing phases of circuit board design may be applicable to the experiment. This approach would utilise precisely positioned testing pins which would form a stable connection between the *Infectus2* wire and the associated NAND flash storage device pin. Ideally these testing pins would be set in a rigid framework allowing the *Infectus2* device to be quickly and consistently attached to the Nintendo Wii's internal NAND flash storage device.

It was agreed that this "rigid framework" approach had the highest potential to provide a stable, solder-less connection between the components however, due to financial and time constraints this approach was scaled back, and fine-pointed, size 12 beading needles were soldered to the command and data wires in place of specialist testing pins. The addition of fine-pointed hand-sewing needles increased the ease with which connections could be made between

the components, although a great deal of time and manual dexterity was still required.

With command and data connections in place, the power and ground lines were connected to the motherboard. Power was taken by attaching the power line “crocodile” clip to a surface mounted device labelled `FIL34`. The ground lines were connected to the metal strips encircling the motherboard. Once all of the necessary connections were in place the *Infectus2* package was connected to the analysis computer using the supplied USB interface, and the console motherboard connected to the mains power supply using the standard Nintendo Wii mains power supply adaptor.

In order to prevent the console from fully booting and potentially corrupting any acquired image of the NAND flash storage device, the following steps were followed when powering-on the console:

1. Mains power was switched on. Red power button LED was observed.
2. Press and hold the momentary switch in order to short-circuit `Data0` to ground.
3. Press the console’s front-mounted power button and observe the LED colour change to green.
4. After observing the LED colour change, the momentary switch was released.

The final stage of the imaging process involved the use of a software application which instructed the *Infectus2* package to read data from the NAND storage device and write it to a file on the analysis workstation. The `amoxiflash` application was developed by a member of the Wii homebrew hardware and software hacking community who was dissatisfied with the official software distributed with the *Infectus2* chip, and was released to the public under the GPL. (29)

Unfortunately, as shown in Figure 6.6, the `amoxiflash` application was unable to properly detect the console's internal NAND flash storage device, thus resulting in the failure of the experiment.

```
root@analysis:~# amoxiflash-linux-0.5 dump nand.bin
amoxiflash version 0.5, (c) 2008,2009 bushing
infectus Device Found @ Address 002
infectus Vendor ID 0x010c4
infectus Product ID 0x082e3
Infectus version (?) = 81
Infectus Loader version = 0.32
PLD ID: NAND Programmer
ID = 200
ID = 200
ID = 200
Unknown flash ID 0200
If this is correct, please notify the author.
root@analysis:~#
```

Figure 6.6: Error message produced by `amoxiflash`

6.3 Analysis of Experiment

Due to the lack of success in acquiring a copy of the data held by the internal NAND flash storage device, this experiment must be considered a failure. The exact reason for its failure is unknown, although it is thought that the imposed condition that nothing be soldered to the motherboard is likely to have been a primary factor.

6.3.1 Possible Causes of Failure

This section of the report describes the issues thought most likely to have been responsible for the failure of this experiment.

Fault with the Infectus2 package

The initial primary concern was a fault with the *Infectus2* package however the `amoxiflash` terminal output (Figure 6.6) appears to show that the *Infectus2* chip was successfully detected. As a result, the probability of an error in this area was deemed to be low.

Bad Solder Connections

The completed *Infectus2* package required a total of 40 solder joints to function perfectly in order to correctly read data from the console's NAND flash storage device. Electrical continuity testing and a thorough visual inspection of the *Infectus2* package did not result in any obvious weak or bridged solder connections. Given the relatively large margin for error afforded by the use of solder pads rather than soldering directly to the chips, the probability of an unseen error occurring in this area was deemed to be low.

Poor Electrical Properties of Sewing Needles

Metal hand-sewing needles are not typically known for their good electrical properties. Although appearing to provide a well-fitting connection between command and data wire, and electrical via, it was thought that the hitherto unknown electrical properties of the sewing needles were likely to have an adverse effect on the performance of the electrical circuit.

Another concern in this area was that the length of the needles, when added to the length of the command and data wires, may have been too great to allow the high quality transmission of an electrical signal between the NAND flash storage device and the *Infectus2* package.

Due to these factors, the probability for error in this area was considered to be high.

Unknown Issue concerning the NAND flash storage device

The terminal output from the `amoxiflash` application (Figure 6.6) appears to indicate that the *Infectus2* package was recognised by the analysis workstation, but the console NAND flash storage device was not.

It was noted that `amoxiflash` incorrectly determined the NAND flash chip identifier to be `0x0200`, rather than `0xECDC` as expected for a Samsung NAND flash storage device such as that used by the Nintendo Wii. An examination of the `amoxiflash` source code revealed that, in addition to the identifiers of NAND flash storage devices commonly found in the Nintendo Wii, there is also a case which will display a terminal message if no NAND flash chip is detected (35). The fact that a NAND flash chip identifier was specified at all appears to support the conclusion that some data – albeit incorrect – is read from the NAND flash chip by the *Infectus2* package.

It was also noted that anecdotal information from the Nintendo Wii homebrew community suggests that the Samsung NAND flash device used by the console is sensitive to electrical disturbances, and is best removed completely from the motherboard and mounted in a TSOP-style adaptor before attempting hardware-based imaging. As explained above, this approach was ruled out due to its complexity and potential to damage the device.

With these factors in mind, the probability of an error occurring in this area was deemed to be high.

Chapter 7 – Software-based Imaging Experiments

This chapter details the process of exploiting the system software running on the Nintendo Wii, and the steps which must then be taken in order to acquire a forensic image of the console's NAND flash storage device.

7.1 Exploiting Vulnerabilities in Wii System Software

Before unsigned code can be loaded and executed by the Nintendo Wii, the security measures put in place by Nintendo system software must be circumvented. Typically these measures are circumvented by buffer overflow exploits, which abuse poor data input handling practices to allow the execution of arbitrary code.

There are currently four publicly available buffer overflow exploits which enable unsigned code execution on the Nintendo Wii. Three of these exploits rely on flaws in game software, while the Bannerbomb exploit relies on a flaw in the Wii System Menu.

The effectiveness of these exploits depends largely upon the version of the Wii System Menu which is installed on the console. Table 7.1 shows the compatibility of each of the four public exploits with each version of the Wii System Menu since version 3.0. Detailed information regarding compatibility with earlier versions of the Wii System Menu was not available, but it is presumed that the same vulnerabilities exist in all previous Wii System Menu versions.

In summary, only the Indiana Pwns exploit is known to work correctly on all currently available Wii System Menu and console combinations. The Smash Stack exploit is known to work only on NTSC Wii consoles however it is believed to be compatible with all current Wii System Menu Versions. The Twilight Hack

and Bannerbomb exploits are compatible with Wii System Menu versions below 4.0 and 4.3 respectively.

System Menu Version	Twilight Hack	Bannerbomb	Smash Stack	Indiana Pwns
3.0	Beta 1	Version 1	NTSC Only	PAL & NTSC
3.1	Beta 1	Version 1	NTSC Only	PAL & NTSC
3.2	Beta 1	Version 1	NTSC Only	PAL & NTSC
3.3	Beta 1	Version 1	NTSC Only	PAL & NTSC
3.4	Beta 2	Version 1	NTSC Only	PAL & NTSC
4.0	Fixed	Version 1	NTSC Only	PAL & NTSC
4.1	Fixed	Version 1	NTSC Only	PAL & NTSC
4.2	Fixed	Version 2	NTSC Only	PAL & NTSC
4.3	Fixed	Fixed	NTSC Only	PAL & NTSC

Table 7.1: Compatibility of Wii software exploits

7.1.1 The Twilight Hack

The Twilight Hack is the first publicly-available exploit which enabled the execution of unsigned code and utilises a specially crafted saved game file for the game “The Legend of Zelda: Twilight Princess”. As of Wii System Menu version 4.0 the Twilight Hack no longer functions, however its source code has been made available to the public with the hope that it will encourage the homebrew software community to find and develop additional saved game buffer overflow exploits (36).

7.1.2 Bannerbomb

The Bannerbomb exploit differs from other publicly-available exploits in that it targets a flaw in the Wii System Menu rather than in a specific game. This means that unsigned code can be executed on any console which uses a vulnerable version of the Wii System Menu without the need for additional software. This

factor is potentially of enormous benefit to the forensic examiner, as it allows a standard SD flash memory card image to be prepared and distributed without having to account for multiple copies of an exploitable game.

The main advantage of Bannerbomb however also proved to be its greatest weakness. While it is very difficult for Nintendo to redress flaws in games (which exist in static formats and are often developed by third-parties), flaws in the Wii System Menu can be patched and pushed to users with relative ease.

The vulnerability on which Bannerbomb relied was rectified by the Wii System Menu version 4.3 update (37).

7.1.3 Smash Stack

Smash Stack takes advantage of a buffer overflow vulnerability in the NTSC version of the game "Super Smash Bros. Brawl". Unlike the Twilight Hack and Indiana Pwns this exploit functions against an in-game custom stage loader rather than the more traditional malformed saved game file. This difference allows the exploit to load unsigned code directly from the SD card reader, the Wii System Menu and thus making the exploit very hard to block.

As the buffer overflow relied on by Smash Stack exists only in NTSC versions of the game, consoles manufactured for sale outside of North America are unlikely to be vulnerable to this exploit (38).

7.1.4 Indiana Pwns

Indiana Pwns is a saved game exploit based on the now defunct Twilight Hack source code. It operates in the same fashion as the Twilight Hack, but requires a copy of the game "LEGO Indiana Jones: The Original Adventures".

Unlike the Twilight Hack, Indiana Pwns is thought to be compatible with all current Wii System Menu versions (39).

7.2 Using Linux

This section describes the experiment which was carried out to determine the suitability of using a modified Linux-based operating system to acquire a forensic image of the Nintendo Wii's internal NAND flash storage device.

Ultimately this experiment failed, as although the console could be made to boot into a Linux operating system with no requirement for additional homebrew software to be installed, the internal NAND flash device did not appear to be accessible.

7.2.1 Preparation of Secure Digital Flash Memory Card

In order to function as a complete Linux operating system the SD flash memory card must be correctly partitioned and formatted in such a manner that it can be read by both the Wii System Menu and White Linux environments.

As outlined in Chapter 5, the Nintendo Wii console used in this experiment utilised Wii System Menu version 4.2. The Bannerbomb exploit was fixed with the release of Wii System Menu version 4.3, but the Smash Stack or Indiana Pwns exploits are likely to be fully-functional on updated consoles.

This section describes the process of partitioning, formatting and preparing a 2GB SD flash memory card for use as a self-contained Linux operating system for the Nintendo Wii. The steps below were carried out using a workstation running a standard installation of Ubuntu 10.04 Desktop Edition and an external SD flash

memory card reader. In these instructions it is assumed that the SD flash memory card is seen as a device named `/dev/sdb`.

1. With the SD flash memory card inserted into the reader, ensure that any existing partitions on the card are unmounted.
2. Start the `fdisk` partition management utility.

```
root@analysis:~# fdisk /dev/sdb
```

3. Remove any existing partitions from the SD flash memory card by entering the command `'o'`.
4. Create a 256MB primary FAT16 partition to store the loader and White Linux kernel.

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)

p

Partition number (1-4): 1
First cylinder (1-984, default 1): <RETURN>
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-984, default 984): +256M

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 6
Changed system type of partition 1 to 6 (FAT16)
```

5. Create a primary Linux partition to store the White Linux file system. All files created by the White Linux instance will be stored in this partition.

```

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)

p

Partition number (1-4): 2
First cylinder (126-984, default 126): <RETURN>
Using default value 126
Last cylinder or +size or +sizeM or +sizeK (126-984, default 984): 984

```

6. Verify the new partition table layout by entering the command 'p' and write the table to the SD flash memory card by pressing 'w'.

```

Command (m for help): p

Disk /dev/sdb: 2032 MB, 2032664576 bytes
64 heads, 63 sectors/track, 984 cylinders
Units = cylinders of 4032 * 512 = 2064384 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1          1         125     251968+    6   FAT16
/dev/sdb2        126         984     1731744    83   Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

Syncing disks.

```

7. Create a FAT16 file system labelled `boot` on the first newly-created partition.

```

root@analysis:~# mkfs.vfat -n boot /dev/sdb1

```

8. Create an ext3 file system labelled `white` on the second newly-created partition.

```

root@analysis:~# mkfs.ext3 -L white /dev/sdb2

```

9. Mount the `boot` partition and untar the White Linux kernel tarball into the root directory.

```
root@analysis:~# mount /dev/sdb1 /media/boot
root@analysis:~# tar -C /media/boot -xjvf gc-linux-mikep5-
v2.6.32+white-1.10.tar.bz2 -o --strip-components 1
```

10. Unzip the Bannerbomb exploit and move the `private` directory into the root directory of the `boot` partition.

```
root@analysis:~# gunzip abd6a_v200.zip
root@analysis:~# mv private /media/boot
```

11. Unzip the LoadMii archive and copy the file `loadMii.elf` to the root of the `boot` partition as `boot.elf`. Some applications are unable to execute directly from Bannerbomb. The LoadMii bootloader application can be used to reduce this risk.

```
root@analysis:~# gunzip loadMii.zip
root@analysis:~# cp loadMii/loadMii.elf /media/boot/boot.elf
```

12. Unmount the `boot` partition, and mount the `white` partition.

```
root@analysis:~# umount /media/boot
root@analysis:~# mount /dev/sdb2 /media/white
```

13. Untar the White Linux file system tarball into the root directory of the `white` partition.

```
root@analysis:~# tar -C /media/white -xjvf 'debian-lenny-5.0+white-
1.10.tar.bz2'
```

14. Unmount the `white` partition.

```
root@analysis:~# umount /media/white
```

The SD flash memory card is now fully prepared for use as a self-contained, persistent Linux operating system.

7.2.2 Imaging Procedure

Before booting the console the following steps were taken:

- A standard USB keyboard was attached
- The prepared SD flash memory card was inserted

The console was powered on as normal. Once booted to the Wii System Menu, the SD Card Menu in the bottom left hand corner of the screen was selected. The Bannerbomb exploit operated as expected and displayed a warning dialog and prompt to execute the `boot.dol/boot.elf` file in the root directory of the `boot` partition on the SD flash memory card (Figure 7.1).

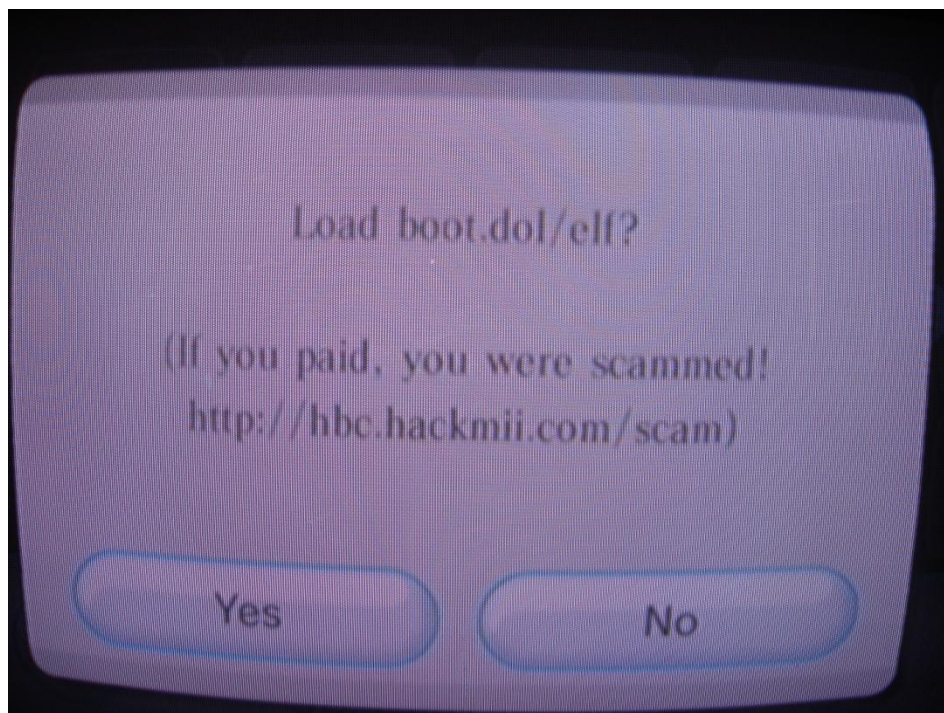


Figure 7.1: Pop-up dialog indicating the success of Bannerbomb exploit

After selecting the “Yes” option, the LoadMii bootloader application was executed and appeared on-screen after a slight delay. Using the buttons of the Wiimote to navigate the directory structure, the file `apps/mikep5.110/boot.elf` was selected and executed by the LoadMii application.

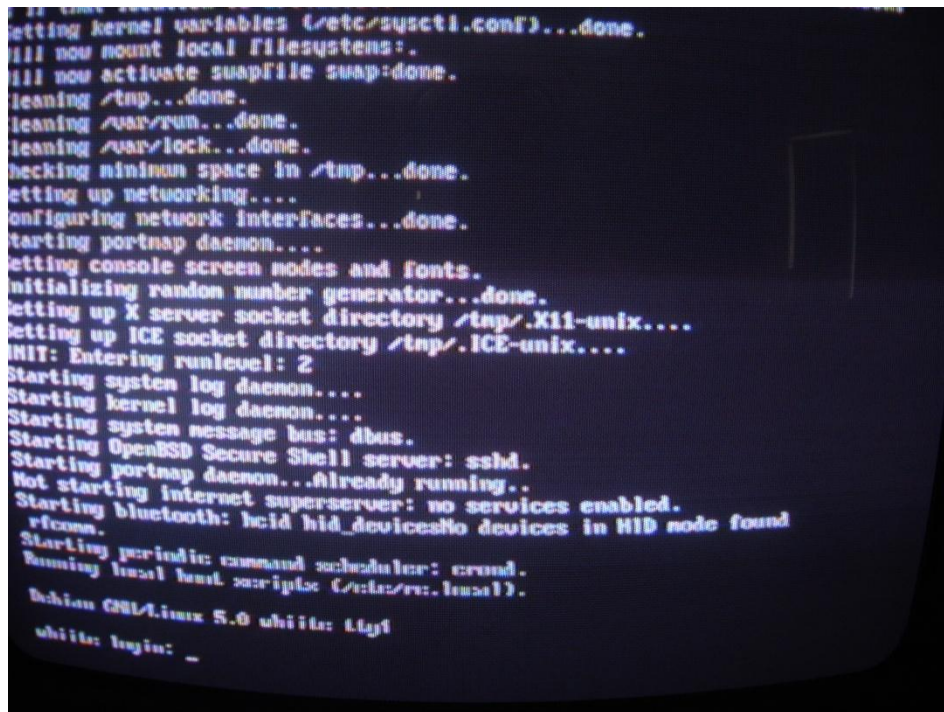


Figure 7.2: White Linux booting and login prompt

After another short delay the White Linux boot process began, printing status messages to the console throughout the process. At its completion a text login prompt was displayed at the bottom of the screen (Figure 7.2).

The White Linux file system tarball is initially configured to allow access with the username `root` and the password `white`. If a USB-to-Ethernet adapter is connected to the console, White Linux will use DHCP to attempt to configure it with the existing Local Area Network. A wireless networking configuration utility is

provided however it did not appear to function correctly when White Linux is booted directly from the SD flash memory card.

Cursory checks indicated that the `dd` low-level data manipulation tool is installed by default, although the `netcat` networking tool is not.

The experiment was halted at this point as it was determined after much searching and probing of devices that the White Linux operating system was unaware of the console's internal NAND flash storage device and, thus, unable to acquire a forensic image of the data which it contains.

7.2.3 Analysis of Experiment

This experiment has shown that the Nintendo Wii is capable of supporting a fully-functional Linux operating system. As White Linux is based on Debian Linux, the Debian `apt-get` package management system can be used to install and update software allowing White Linux to be tailored toward forensic use and distributed as an SD flash memory card image.

Although it has been shown possible to successfully boot a Linux operating system on the Nintendo Wii without the need to install any additional software, this experiment must be regarded as a failure due to the inaccessibility of the console's internal NAND flash storage device.

While this approach may be feasible in the future, the reverse-engineering skills and low-level programming knowledge likely to be required in order to develop a suitable operating system NAND flash storage device driver are far beyond the scope of this project.

7.3 Using Homebrew Software

This section describes the experiment which was conducted in order to determine the suitability of the BootMii homebrew software package for use in the acquisition of data from the Nintendo Wii's internal NAND flash storage device.

While this procedure resulted in the acquisition of a copy of the data held by the NAND flash storage device and the extraction of the console-specific encryption keys necessary to render the data intelligible, the experiment cannot be considered a complete success due to concerns over the integrity and repeatability of the imaging process. It is believed that this experiment it has produced an imaging method which is valid as a proof-of-concept, however further work is suggested which may result in the refinement of the method.

7.3.1 Preparation of Secure Digital Flash Memory Card

As outlined in Chapter 5, the Nintendo Wii console used in this experiment utilised Wii System Menu version 4.2. The Bannerbomb exploit was fixed with the release of Wii System Menu version 4.3, but the Smash Stack or Indiana Pwns exploits are likely to be fully-functional on updated consoles.

This section describes the process of partitioning, formatting and preparing a 2GB SD flash memory card in order to install and use the BootMii homebrew software package. The steps below were carried out using a workstation running a standard installation of Ubuntu 10.04 Desktop Edition and an external SD flash memory card reader. In these instructions it is assumed that the SD flash memory card is seen as a device named `/dev/sdb`.

1. With the SD flash memory card inserted into the reader, ensure that any existing partitions on the card are unmounted.

2. Start the `fdisk` partition management utility

```
root@analysis:~# fdisk /dev/sdb
```

3. Remove any existing partitions from the SD flash memory card by entering the command 'o'.

4. Create a primary FAT16 partition.

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)

p

Partition number (1-4): 1
First cylinder (1-984, default 1): <RETURN>
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-984, default 984): 984

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 6
Changed system type of partition 1 to 6 (FAT16)
```

5. Verify the new partition table layout by entering the command 'p' and write the table to the SD flash memory card by pressing 'w'.

```
Command (m for help): p

Disk /dev/sdb: 2032 MB, 2032664576 bytes
64 heads, 63 sectors/track, 984 cylinders
Units = cylinders of 4032 * 512 = 2064384 bytes

Device Boot      Start          End      Blocks   Id  System
/dev/sdb1            1          984     1983712    6   FAT16

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

Syncing disks.
```

6. Create a FAT16 file system labelled `BootMii` on the first newly-created partition.


```
root@analysis:~# mkfs.vfat -n BootMii /dev/sdb1
```

7. Mount the newly-created BootMii partition.

```
root@analysis:~# mount /dev/sdb1 /media/BootMii
```

8. Unzip the Bannerbomb exploit and move the `private` directory into the root directory of the BootMii partition.

```
root@analysis:~# gunzip abd6a_v200.zip  
root@analysis:~# mv private /media/BootMii
```

9. Unzip the HackMii installer archive and move the file `boot.elf` into the root directory of the BootMii partition.

```
root@analysis:~# gunzip hackmii_installer_v0.6.zip  
root@analysis:~# mv hackmii_installer_v0.6/boot.elf /media/BootMii
```

10. Unmount the BootMii partition.

```
root@analysis:~# umount /media/BootMii
```

The SD flash memory card is now fully prepared.

7.3.2 Imaging Procedure

Before the BootMii package can be used to create an image of the internal NAND flash storage device, it must be installed on the console. The installation process makes permanent changes to the contents of the NAND flash device, although the package installer does provide an option to uninstall the each of the BootMii package components.

Prior to beginning the installation process, the SD flash memory card was prepared as described above and inserted into the front-mounted SD flash memory card reader.

The console was booted as normal and the SD Card Menu was selected from the lower left-hand corner of the Wii System Menu, causing the Bannerbomb exploit to be run (Figure 7.1) which in turn executed the BootMii package installer.

The first visible action of the BootMii package installer is to display a screen informing the user that the software is to be distributed freely and that no money should be exchanged for it. Once the user has indicated that the message has been read, the package installer runs a series of tests to determine the compatibility of homebrew software with the specific hardware and software revisions used by the console.

As shown in Figure 7.3, the Homebrew Channel and DVDX components of the package can be installed as intended by the developers, but a notice is given stating that BootMii can only be installed as an IOS. This restriction is an effect of Nintendo's efforts to patch the `boot1` link in the bootloader chain. Ideally, BootMii will be installed as a `boot2` replacement, providing protections against file system corruption and installing itself predictably to blocks 1-7 of the internal NAND flash storage device. With the patching of the `boot1` RSA certificate hash collision bug on consoles manufactured sometime in 2008, `boot1` cannot be tricked into loading modified versions of `boot2` on new-model Wii consoles. While the installation of BootMii as a `boot2` replacement is preferable, the BootMii IOS provides sufficient functionality to complete this experiment.

The first component to be selected for installation is the Homebrew Channel. The Homebrew Channel acts as a permanently installed Wii System Menu exploit allowing unsigned code to be loaded directly from an SD flash memory card

without the need to reuse the Bannerbomb (or similar) exploit. Its installation is optional, but highly recommended when BootMii is to be installed as IOS.

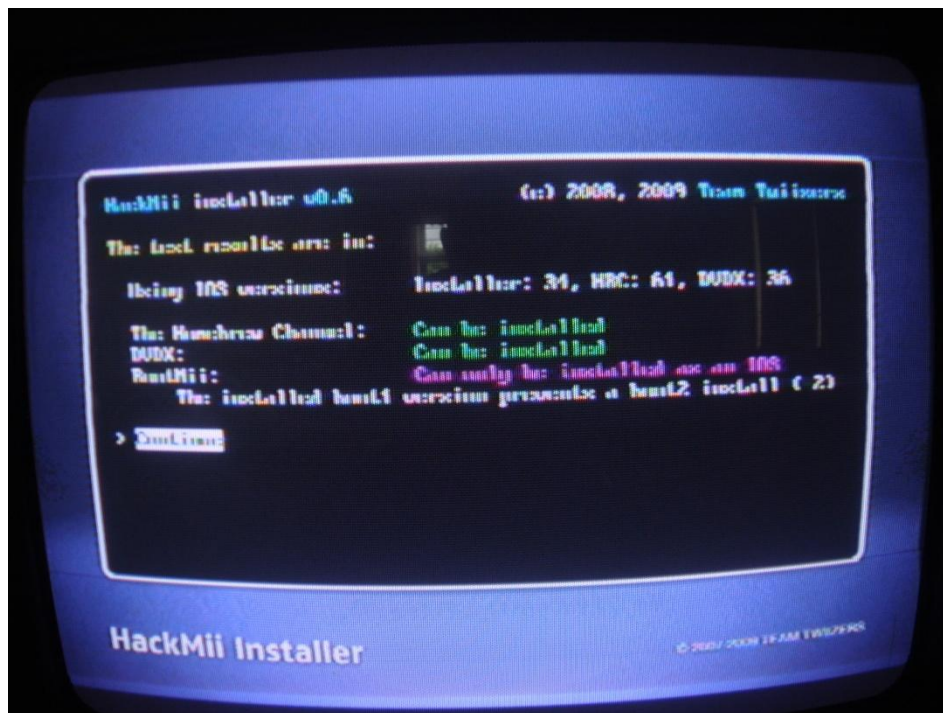


Figure 7.3: Results of BootMii installer compatibility tests

Following the successful installation of the Homebrew Channel, the BootMii application was successfully installed as IOS before exiting the installation package which returned the console to the Wii System Menu. Figure 7.4 shows the Wii System Menu which now contains an icon for the newly-installed Homebrew Channel. Selecting the Homebrew Channel and pressing the “Home” button on the Wiimote displays the Homebrew Channel menu from which the BootMii application can be executed (Figure 7.5). It should be noted that the BootMii package installer creates a number of configuration files on the SD flash memory card during the Homebrew Channel and BootMii installation process. This SD flash memory card should be inserted before starting the Homebrew Channel application.






Action	Wii Console Buttons	Nintendo GameCube
Previous Option	—	
Next Option		
Select Option		

Table 7.2: Summary of BootMii controls

Once BootMii is launched the WiiMote controller is no-longer recognised as a valid input device. Instead, a Nintendo GameCube controller or the Power and Reset buttons on the front of the console can be used to navigate the BootMii application menus. A summary of the controls is contained in Table 7.2.



Figure 7.4: Wii System Menu updated after Homebrew Channel installation

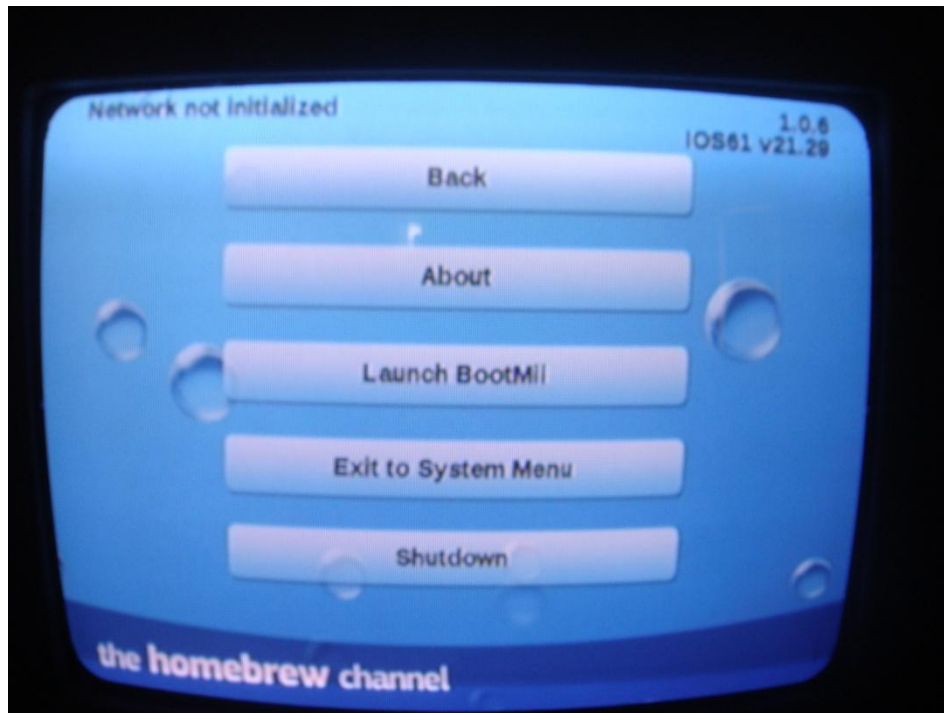


Figure 7.5: The Homebrew Channel menu

The BootMii application displays a main menu with four options, as shown in Figure 7.6 (Left). Upon selection of the forth option the NAND flash operations menu, shown in Figure 7.6 (Right), is displayed to the user. The first option starts the BackupMii application which reads the data from the internal NAND flash storage device and writes a copy of it to the SD flash memory card. The second option uses the RestoreMii application to copy an acquired image from the SD flash memory card to the console's internal NAND flash file system.

After selecting the first option, the BackupMii application began the process of copying the data from the internal NAND flash device on to the SD flash memory card.

During the copying process a number of warnings referring to "Factory Bad Blocks" should be expected. These blocks are a result of errors in the manufacturing process, which works to produce a NAND flash device operating

within a certain tolerance level. It has been reported that of the 4096 blocks contained on the NAND flash memory chip, at least 4016 must be valid. This tolerance allows for up to 80 “Factory Bad Block” warnings before a chip is to be considered faulty (40).

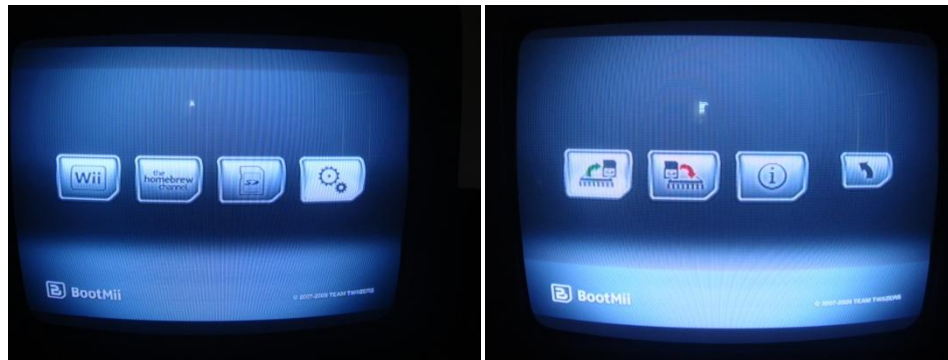


Figure 7.6: BootMii NAND flash backup menu screens

Figure 7.7 shows the BackupMii application as it progresses through the NAND flash imaging process. The application's progress is measured by the coloured grid in the top half of the screen. Each square in the grid represents one of the internal NAND flash device's data blocks. As each block is copied its representative grid square changes colour from grey to green, allowing the user to accurately assess the progress of the operation.

Once the data has been copied to the SD flash memory card a verification procedure is run against the original data on the internal NAND flash device (Figure 7.8). It is possible to skip this operation but not recommended as it is likely to be the most practical method of assessing the integrity of the BackupMii imaging operation.

Upon the completion of the imaging process the user is invited to press any button to exit the BackupMii application. Returned to the BootMii NAND operation menu, the forth option was selected to return to the BootMii main

menu, and then the first option which reboots the console to the Wii System Menu.

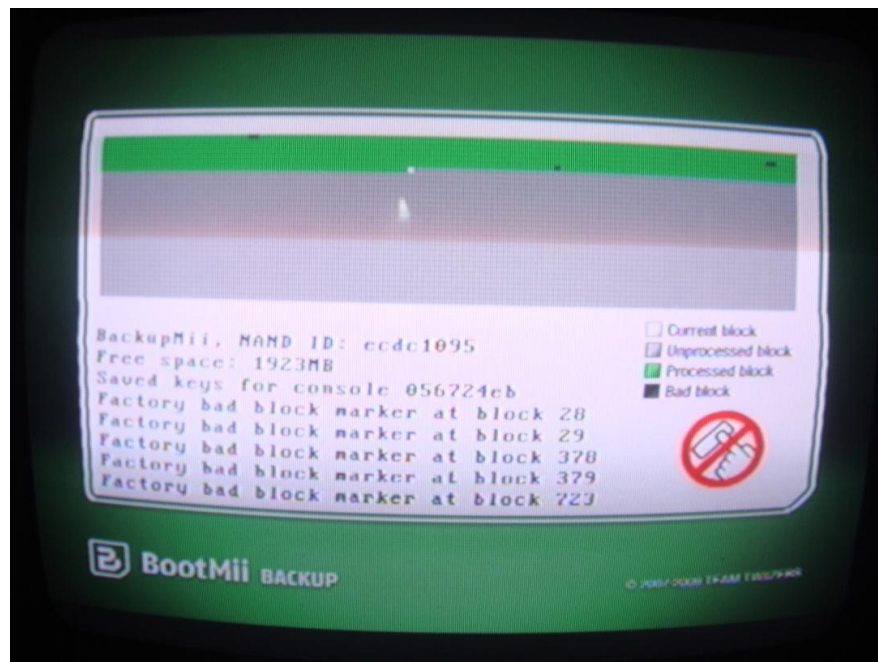


Figure 7.7: BackupMii during the imaging process

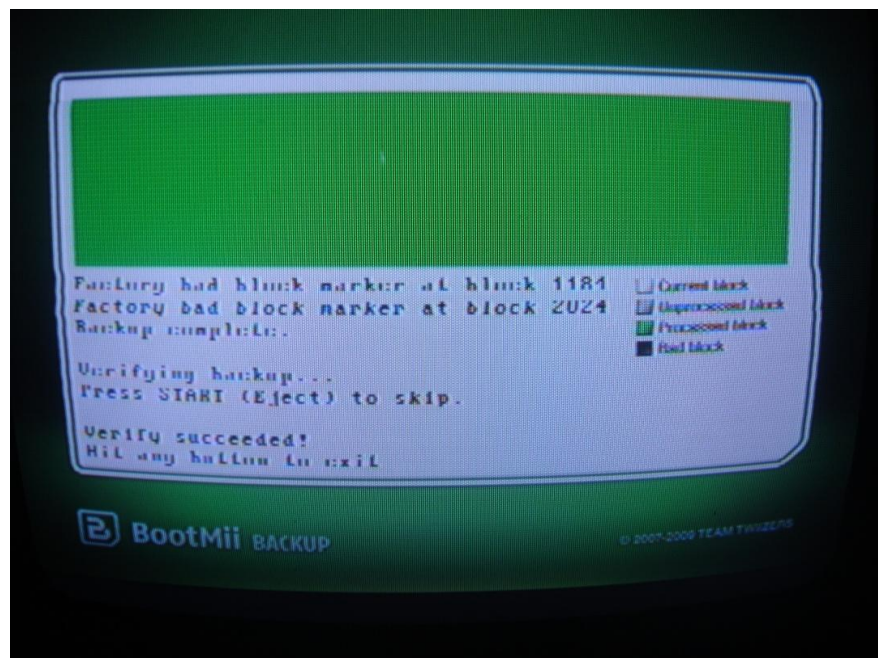


Figure 7.8: BackupMii after completion of imaging process

7.3.3 Analysis of Experiment

This experiment has shown that it is possible for homebrew software running on the Nintendo Wii as IOS both to access and record the contents of the console's internal NAND flash storage device. In addition to creating an image of the NAND flash storage device, the BackupMii application also extracted the console-specific encryption keys which are required before any attempt can be made to extract meaningful data from the image of the NAND flash storage device.

While this experiment was successful in that an image was captured, it has not been possible to show conclusively that the captured image represents a wholly accurate copy of the data stored by the internal NAND flash device. The BackupMii application runs a verification routine which takes place after copying the data from the internal NAND flash to the SD flash memory card, although it is believed that this verification routine is most likely to involve verifying that the data written to the SD flash memory card matches that on the internal NAND flash device, without access to the BackupMii source code it is not possible to be certain of the exact operation being performed.

In addition to the concern over the integrity of the data copying functionality, there is reason to be concerned over the repeatability of the entire BackupMii process. This concern came to light after repeating the BackupMii imaging process multiple times in quick succession. Despite not exiting the BootMii application and minimising the period of time between acquisition attempts, it does not appear to be possible to create two images of the same NAND flash storage device which produce matching MD5 or SHA-1 hashes. One likely explanation for this issue is the fact that the Starlet processor periodically accesses and possibly changes to the data held by the NAND flash storage

device. This conclusion is drawn from the need for a momentary push-button switch to be used in conjunction with the *Infectus2* NAND flash controller, the purpose of which was to short-circuit the `Data0` line to ground, halting the boot process while maintaining a power supply to the NAND flash device.

With these concerns in mind it is believed that this experiment can be considered a successful proof-of-concept. It is recommended that the software imaging process be treated in the same regard as the live-imaging of a standard personal computer. It is also believed that a legitimate case can be made for the development of an open-source alternative to the BootMii package, concentrating on emulating the functions of the BackupMii application while reducing the footprint of the package to the minimum required while still ensuring effective operation. Although it is believed that this goal may be accomplished through the use of the open-source "MINI" IOS replacement libraries, the time-scale of the project means that this potential solution must be regarded as future work (41).

Chapter 8 – Restoration of an Acquired Image

This experiment was originally proposed to determine the feasibility of restoring a previously acquired copy of the data held by the internal NAND flash device to the console in order to assist with its “live examination”, as described in Turnbull’s paper “Forensic Investigation of the Nintendo Wii: A First Glance” (26).

Due to the security measures built in to the Nintendo Wii, NAND flash device images captured from one console cannot be made to function on another console without making extensive modifications relating to the console-specific encryption keys (42). As a result it is necessary that the seized console be used when conducting any “live examination”, modifying data and potentially destroying evidence in the process.

A successful NAND flash device image restoration technique would allow the console to be reset to its seized state, providing multiple opportunities for examination, and potentially reducing the impact of inadvertent modification of the console’s data.

The experiment described in this chapter uses the RestoreMii application bundled with the BootMii homebrew software package to copy an image from the SD flash memory card to the internal NAND flash storage device. In this case an attempt will be made to restore the “baseline” NAND flash device image captured after the initial setup and installation of the BootMii homebrew software package; reverting the console to a state as close to “clean” as is possible with a software-based imaging procedure.

8.1 Preparation for Restoration

The Secure Digital flash memory card used in this experiment is the same card used in the BackupMii NAND flash device imaging experiment described in section 7.3. No modification of the partition table or file system should be necessary, although the configuration files created by the BootMii homebrew software package installer appear to be required in order to launch the BootMii application.

The file containing the NAND flash storage device image captured in the homebrew software imaging experiment was copied to the root directory of the SD flash memory card as `nand.bin`.

Finally, a Nintendo GameCube controller was connected to one of the GameCube controller ports hidden by a flap on the top of the console. This is required in order to enter a button sequence confirming that the user understands the risks associated with writing data directly to the internal NAND flash device and that they are willing to proceed with the operation.

8.2 Restoration Procedure

Prior to booting the console, the prepared SD flash memory card was inserted into the reader on the front of the console and the Nintendo GameCube controller was plugged into a port hidden by a flap on the top.

Once the console had booted to the Wii System Menu, the Homebrew Channel was selected and the BootMii application started as described on page 75.

The GameCube controller was used to navigate the BootMii menus as shown in Figure 7.6. First to select the NAND flash operations menu, and then to select the second option which launches the RestoreMii application.

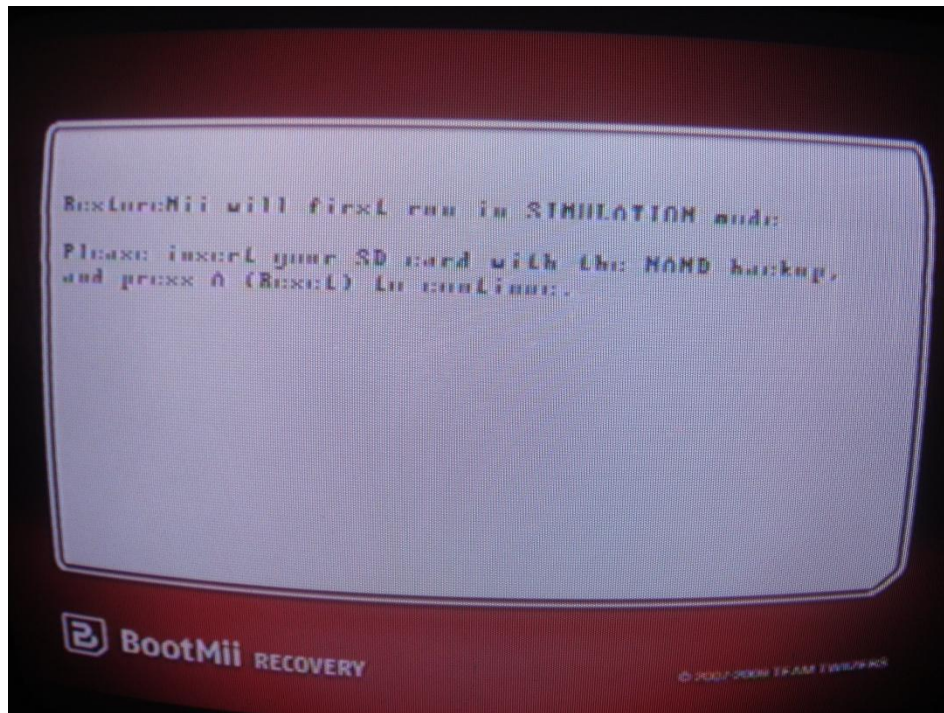


Figure 8.1: The first screen shown by the RestoreMii application

The launch screen, shown in Figure 8.1, informs the user that the restoration process will first be run as a simulation, and prompts for controller input to continue. Proceeding to the next screen (Figure 8.2), RestoreMii checks the internal NAND flash file system to determine the state of the `boot2` bootloader.

If the BootMii application has been installed as a modification to the `boot2` bootloader stage, the risks associated with writing directly to the internal NAND flash storage device are greatly reduced. In this case – as is likely with all Nintendo Wii consoles manufactured after mid-2008 – the BootMii application was only available as an IOS, which does not afford any of the protections possible through the modification of `boot2`. If a BootMii IOS installation is detected a warning is displayed advising that any failure during the restoration process may result in the console becoming unusable.

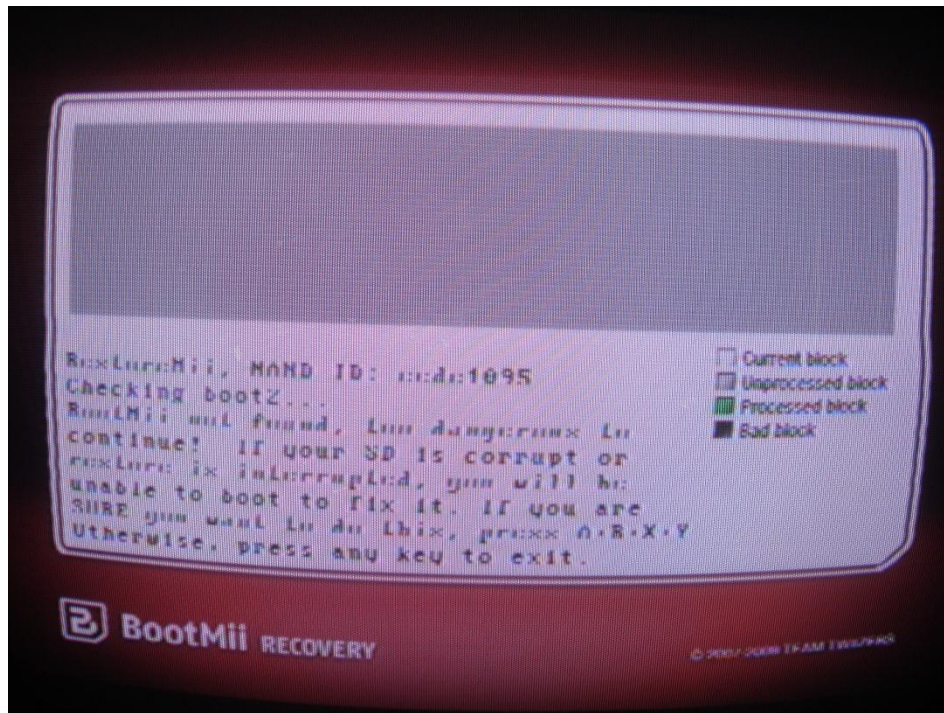


Figure 8.2: The warning screen displayed by the RestoreMii application

After pressing the deliberately awkward A + B + X + Y button combination on the Nintendo GameCube controller, the simulated restoration begins. On the first pass, each block of data on the internal NAND flash storage device is compared to the corresponding block in the NAND flash device image stored on the SD flash memory card. If the data blocks are found to be identical they will be skipped during the restoration process. Where data blocks are found to differ, the block held on the SD card will be written to the internal NAND flash device. An illustration of the differences found between the current internal state of the console and the previously captured NAND flash device image can clearly be seen in Figure 8.3.

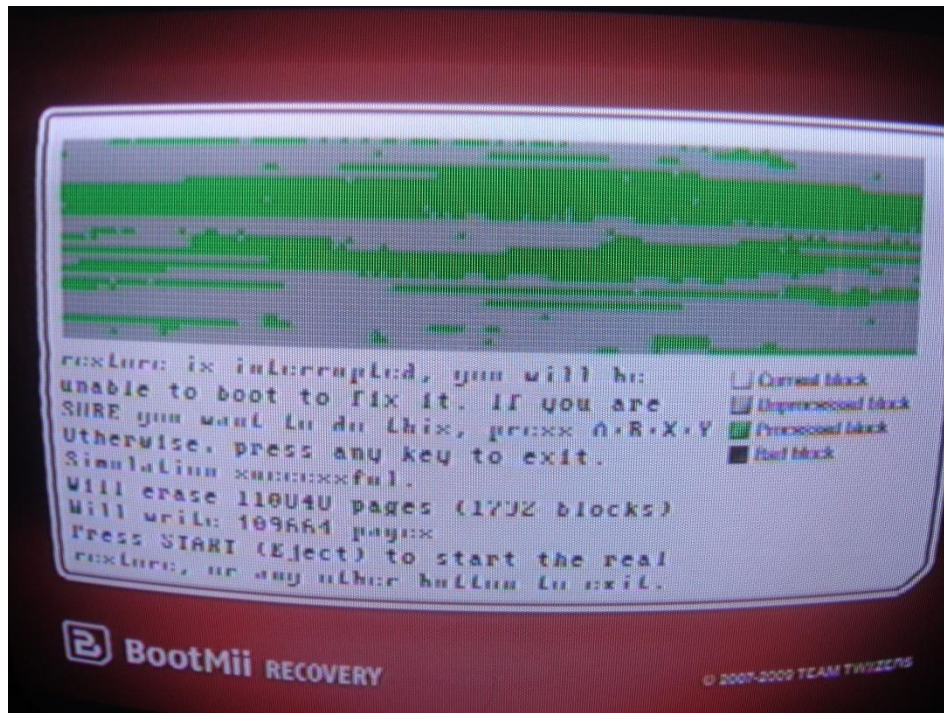


Figure 8.3: Illustration of the differences between NAND flash device images

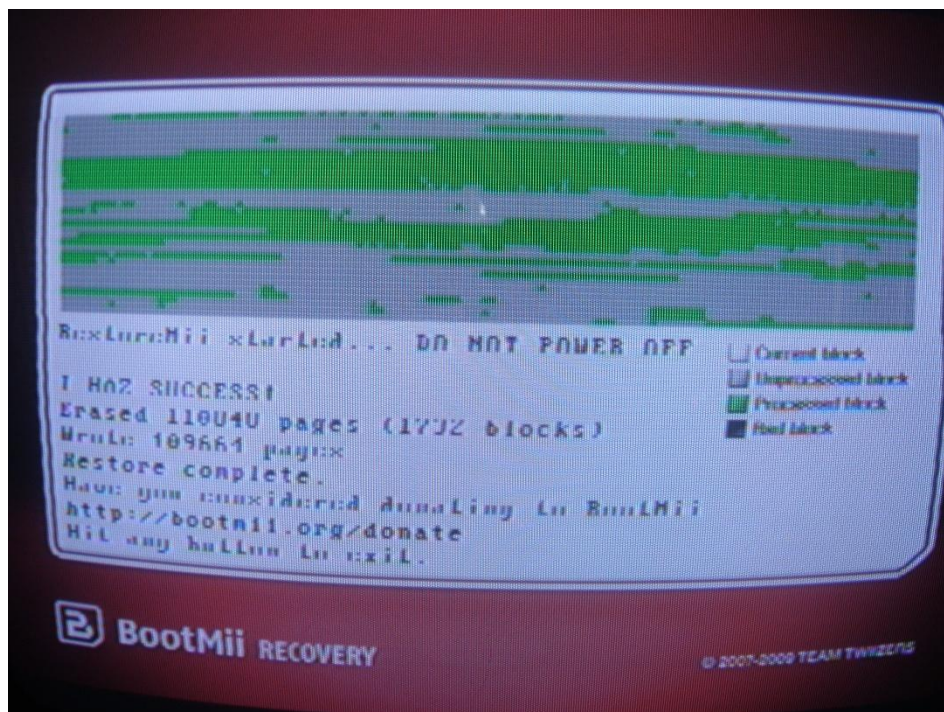


Figure 8.4: Confirmation of the success of the restoration procedure

Upon the successful completion of the simulation, the user will be prompted to begin the final restoration process, at which point the NAND flash device image contained on the SD flash memory card will be written to the internal NAND flash storage device. It is important to note that any interruption to the procedure from this point onward may lead to the console being rendered completely inoperable.

The data blocks highlighted during the simulated restoration process are overwritten with the corresponding data blocks from the NAND flash device image stored on the SD flash memory card. At the conclusion of this process, a confirmation screen similar to that shown in Figure 8.4 will be displayed.

At this point the RestoreMii application was exited and the console rebooted to the Wii System Menu. The changes made by the restoration procedure are clearly visible in Figure 8.5, the most obvious of which are the removal of the Internet Channel application, and the replacement of the photograph of a bird with the standard icon for the Photo Channel.

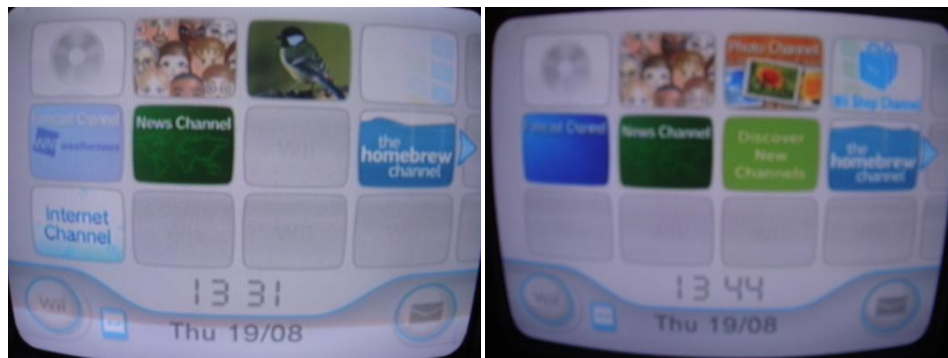


Figure 8.5: Wii System Menu before (L) and after (R) the restoration process

8.3 Analysis of Experiment

This experiment resulted in the complete restoration of a previously acquired NAND flash storage device image, and can thus be considered a success.

Despite the success of this experiment there are a number of serious risks associated with writing data directly to the internal NAND flash storage device, as a corrupt NAND flash device image or loss of power during the restoration process will almost certainly leave the NAND flash file system corrupt and render the device unable to boot. These risks are mitigated on older consoles with `boot1` bootloader stages which are vulnerable to the RSA certificate hash collision bug as described on page 17, as this vulnerability allows the console to be partially booted without the need for a valid NAND flash file system through the BootMii application. However, this `boot1` bug was rectified on consoles manufactured since mid-2008, meaning that NAND flash file system corruption will likely leave the console completely unusable unless a valid NAND flash storage device image can be restored using external hardware.

While this experiment has shown that it is possible to restore the internal state of the Nintendo Wii to an earlier snapshot, the risks outlined above lead the author to recommend that this procedure be carried out only when necessary, particularly on post-2008 consoles, as any error in the procedure may result in the console being rendered completely inoperable.

Chapter 9 – Data Analysis

This chapter details the creation, extraction and analysis of test data from the captured image of the internal NAND flash storage device.

9.1 Creation of Test Data

A great deal of time was spent during the planning phase of the project determining the most efficient method of creating a set of test data. The test data should ideally be representative of typical use of the console while still containing data relating to the use of uncommon or niche features which may have potential for abuse.

After some discussion between the author, project supervisor, and a member of Lothian and Borders Police Forensic Computer Unit, it was decided that the focus of any test data should be on the networking capabilities of the console – primarily the exchange of messages and images. In an extension to scenarios used in laboratory-based coursework exercises, test data was created which involved the downloading and exchange of ornithological images, where data relating to birds or bird watching topics would be considered suspect. With this in mind, the creation of test data can be broken down into the three areas described below.

9.1.1 General Use

This area aimed to simulate the typical use of the console by a home user. The 802.11b/g wireless network adapter was configured to join a local area network, and the Internet Channel was downloaded from the Wii Shop Channel. Two

“Mii” avatars were created, named “Peter” and “John Doe”. A short period of time was also spent playing the games listed below:

- Wii Sports
- Wii Sports Resort
- Madden NFL 08

9.1.2 Internet Browsing

The internet browsing sessions used to create the test data can be roughly split into two groups. The first group of browsing sessions were an extension to the laboratory-based coursework examples involving the viewing of images of birds and internet searches for bird watching topics. During these sessions, a specially-created web-based email account was accessed and used to exchange a number of emails with another specially-created email address.

A second group of sessions involved more general internet browsing, including the University of Strathclyde Computer and Information Sciences website, and a number of searches relating to Alan Turing and his paper “On Computable Numbers, with an Application to the Entscheidungsproblem”⁴.

9.1.3 Exchange of Messages

Extensive use of the Wii Message Board feature was made in exchanging text messages and images between the test console and a second, unrelated console, and between the test console and a number of specially-created web-based email accounts named “Alice” and “Bob”.

⁴ London Mathematical Society, 1937. Proceedings of the London Mathematical Society. Vol. 42, pp. 230-265. Available online: <http://www.turingarchive.org/viewer/?id=466&title=01a>

Particular attention was paid to the Wii Message Board Address Book, which acts as a white listing mechanism requiring the mutual authorisation of both sender and recipient before the exchange of messages is permitted.

9.2 Accessing the NAND flash file system

It is thought likely that the Nintendo Wii uses a specially-developed, proprietary file system. No details have been made public by Nintendo, but the efforts of the homebrew software community have resulted in the release of a number of tools which are able to read the Nintendo Wii file system, albeit in a rudimentary manner.

One such tool is `wiinandfuse`, which is released under the GPL-compatible MIT Licence and uses the Filesystem in Userspace (FUSE) Linux kernel module in order to mount an encrypted Wii NAND flash device image as a virtual Linux file system. The `wiinandfuse` tool is currently under development, but at the time of writing the most recent stable release was version 1.1 (43).

When executed without providing any command-line arguments, `wiinandfuse` displays a screen of usage information which is reproduced in Figure 9.1.

Before `wiinandfuse` can be used to mount a NAND flash device image it is necessary to create mount point to which the virtual file system will attach itself. In this case a new directory named `nand` was created in the system root directory.

It was found through experimentation that in addition to the specified file containing the NAND flash device image and mount point, two command-line flags were required before the virtual file system could be successfully created. First, the `-h` flag was required to disable Hashed Message Authentication Code verifications. Setting this flag means that writing to the virtual file system is not

permitted, resulting in read-only access. The second flag used is `-e`. This is currently required due to a bug in `wiinandfuse` version 1.1 which prevents the correct handling of the Error-Correcting Code included in the NAND flash device image created by the BootMii procedure. Rather than correctly processing bad ECC exceptions, attempting to access files contained within the virtual file system will often result in software crashes due to input/output errors. While not ideal, the use of this flag is currently the only method of accessing the entire virtual file system.

The procedure followed to manually mount a NAND flash device image as a virtual file system with `wiinandfuse` is shown below:

```
root@analysis:~# mkdir /nand
root@analysis:~# wiinandfuse ~/nand.bin /nand -h -e
wiinandfuse v1.1 by yellowstar6
root@analysis:~#
root@analysis:~# ls -l /nand
total 0
drwxr-xr-x 2 root root 0 1970-01-01 00:59 import
drwxr-xr-x 2 root root 0 1970-01-01 00:59 meta
drwxr-xr-x 2 root root 0 1970-01-01 00:59 shared1
drwxr-xr-x 2 root root 0 1970-01-01 00:59 shared2
drwxr-xr-x 2 root root 0 1970-01-01 00:59 sys
drwxr-xr-x 2 root root 0 1970-01-01 00:59 ticket
drwxr-xr-x 2 root root 0 1970-01-01 00:59 title
drwxr-xr-x 2 root root 0 1970-01-01 00:59 tmp
root@analysis:~#
```

This procedure was later automated in a small collection of Python applications named WiiTools which was developed during the data analysis phase in order to simplify the extraction and processing of user data from a Nintendo Wii NAND flash device image. Further details regarding the WiiTools Python script collection can be found in Appendix C.

```
root@analysis:~# wiinandfuse
wiinandfuse v1.1 by yellowstar6
Mount Wii NAND images with FUSE.

Usage:
wiinandfuse <nand.bin> <mount point> <options>

Options:
  -s: Dump contains only the 4MB SFFS. Reading/writing files will do
      nothing, the data reading buffers will be cleared.

  -k: Directory name of keys to use for raw NAND images. Default for
      keyname is "default". Path: $HOME/.wii/<keyname>

  -p: Use NAND permissions. UID and GUI of objects will be set to the
      NAND UID/GID, as well as the permissions. This option only enables
      setting the UID/GID and permissions in stat, the open and readdir
      functions don't check permissions.

  -g<supercluster>: Use the specified SFFS supercluster index. If no
      number is specified, the superclusters are listed.

  -h: Disable SFFS HMAC verification. Default is enabled.

  -v: Abort/EIO if HMAC verification of SFFS or file data fails. If
      SFFS verification fails, wiinandfuse aborts and NAND isn't mounted. If
      file data verification fails, read will return EIO.

  -r<supercluster>: Disable round-robin SFFS updating, default is on.
      When disabled, only the first metadata update has the version and
      supercluster increased. If supercluster is specified, the specified
      supercluster index has the version set to the version of the oldest
      supercluster minus one.

  -e: Ignore ECC errors, default is disabled. When disabled, when
      pages have invalid ECC reads return EIO.
```

Figure 9.1: Usage information for the wiinandfuse file system tool

9.3 Analysis of Extracted Data

This section of the report details the locations of files which contain data likely to be of interest to a forensic analyst. With the networking capabilities of the console in mind, the focus of this data tends to be on internet browsing, email-like messaging, and the list of authorised contacts which acts as a message white listing service.

While much of this data is accessible from the Wii System Menu, it may be preferable in some cases to have the ability to access the raw data without interference from the console system software.

Although for some files it was possible to gain a reasonable understanding of the internal structure and develop the automated parsing tools included in Appendix C, the majority of files were examined manually with a hex editor and standard UNIX tools such as `grep` and `strings`.

9.3.1 Web Browsing

As noted above, the Internet Channel does not store a detailed history of browsing activity however by entering the Internet Channel settings menu the user is presented with the option to remove internet cookie files, leading to the presumption that, at the very least, records of cookie files created during the browsing session would be stored somewhere within the NAND flash file system.

Figure 9.2 shows the process by which the location of the Internet Channel data was determined. While the keyword "Turing" was found in 6 files, a manual inspection of the five `*.app` files appeared to indicate that they were unrelated to the Internet Channel browsing records. A second search for the keyword "Strathclyde", also entered during the second test browsing session, returned only a single result. Due to this single file appearing in the results of both searches the file `<mount_point>/title/00010001/48414450/data/opera.vff` was determined to be the most likely location of stored Internet Channel data.

When examining this file with a hex editor (Figure 9.3) it soon became apparent that certain pieces of browsing history information is in fact written to the NAND flash storage device by the Internet Channel application.

```

root@analysis:~# find /nand -type f -print | xargs grep -li 'turing'
/nand/title/00010001/48414450/content/0000002c.app
/nand/title/00010001/48414450/content/00000029.app
/nand/title/00010001/48414450/content/0000002d.app
/nand/title/00010001/48414450/content/0000002f.app
/nand/title/00010001/48414450/content/0000002e.app
/nand/title/00010001/48414450/data/opera.vff
root@analysis:~#
root@analysis:~# find /nand -type f -print | xargs grep -li
'strathclyde'
/nand/title/00010001/48414450/data/opera.vff
root@analysis:~#

```

Figure 9.2: Output of `grep` when searching for Internet Channel data

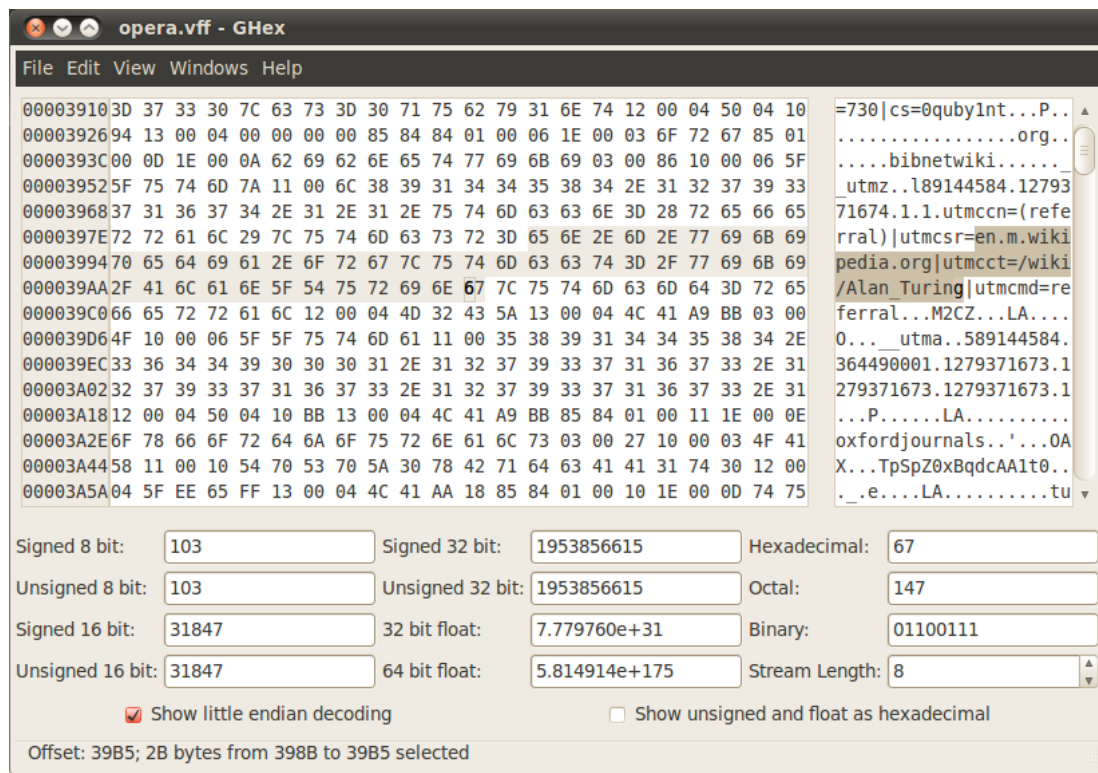


Figure 9.3: Cookie data stored by the Internet Channel

The `utmz` cookie segments relating to the “Alan Turing” Wikipedia page can be seen in Figure 9.3. It should also be noted that the `utma` segments which follow

appear to show UNIX-style timestamps corresponding with the time that the resource was last visited, as shown below.

```
peter@analysis:~$ date -d @1279371673  
Sat Jul 17 14:01:13 BST 2010  
peter@analysis:~$
```

Although manual inspection of the Internet Channel data shows that some browsing information is stored, time constraints and the non-contiguous nature of the file meant that it was not possible to develop a robust automated parser for this data.

9.3.2 Internet Bookmarks / Favourites

During the manual inspection of the `opera.vff` file it was discovered that details of saved “Favourite” web pages are also stored alongside the internet cookie data.

A number of previously visited web pages were saved as “Favourites” during the creation of test data, including the BBC News homepage, the record for which is visible in its raw form in Figure 9.4.

Although it was not possible to create a robust automated parser for this data, the fact that it can be understood reasonably well through manual inspection may be a solution to the concern set out in Turnbull’s paper that the saved “Favourite” titles and thumbnail icons displayed by the Internet Channel may not accurately represent the true details and content of the saved resources (26).

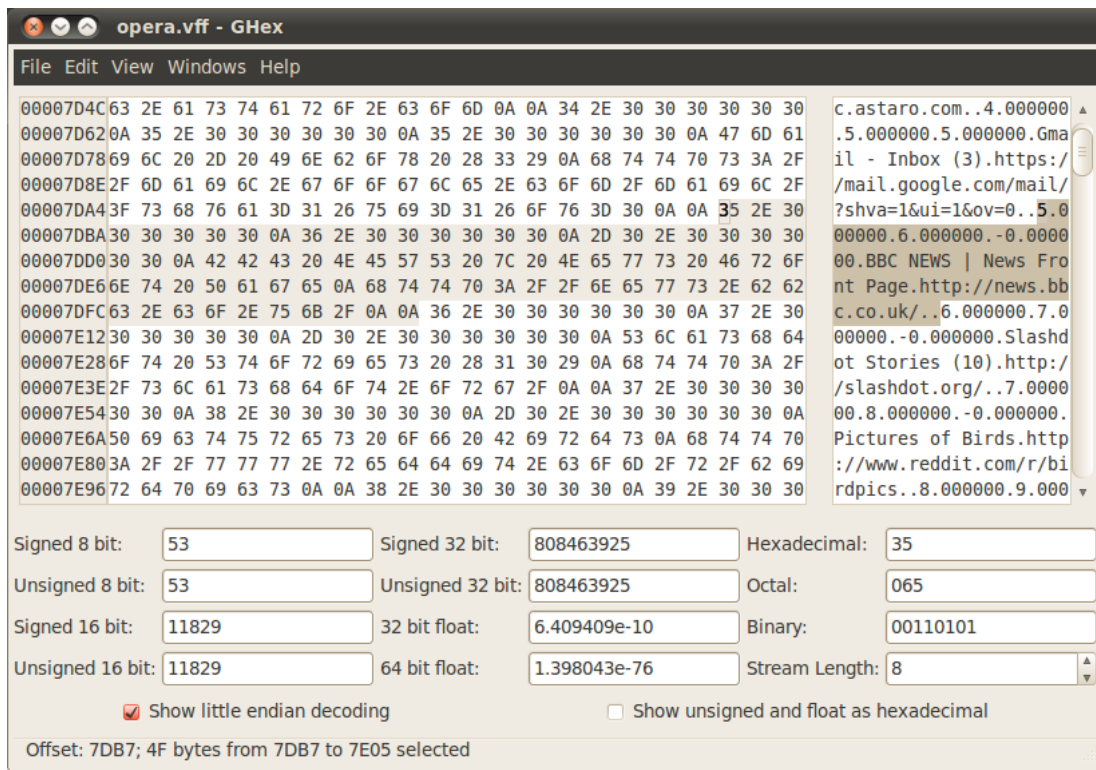


Figure 9.4: Favourites data stored by the Internet Channel

9.3.3 Saved Contacts

Before messages can be exchanged through the Wii Message Board system, the sender and recipient must undergo a mutual white listing procedure controlled by the Wii Message Board Address Book. The Wii Message Board Address Book maintains its records of authorised contact details in a well-structured file which can be found at the path `<mount_point>/shared2/wc24/nwc24fl.bin`.

Unlike many of the files contained within the Nintendo Wii file system, `nwc24fl.bin` is reasonably well documented and understood, although its use to the wider homebrew software community is limited (44).

The main structure of the file is outlined in Table 9.1 while the structure of each specific contact, or Friend List entry, is described in Table 9.2.

Start	Length (Bytes)	Description
0x000	4	Magic Number: "wcf1" Possibly used to identify file
0x004	4	Unknown
0x008	4	Maximum number of records stored. Seemingly always 100
0x00C	4	Actual number of records stored
0x010	48	Padding
0x040	800	Friend Codes
0x360	32000	Friend List entries. Described in Table 9.2

Table 9.1: Structure of Wii Address Book file

Start	Length (Bytes)	Description
0x000	4	Contact type. 0x0001 if unique console code. 0x0002 if email address
0x004	4	Confirmed contact. 0x0001 if unconfirmed. 0x0002 if confirmed. (Email contacts are seemingly always 0x0001)
0x008	24	Contact nickname
0x020	4	Identifier of associated "Mii" avatar
0x024	4	System identifier
0x028	24	Unknown
0x040	96	Contact email address or unique console code. Console codes appear to be base64 encoded
0x0A0	160	Unknown

Table 9.2: Structure of Wii Friend List entries

With the potential to store details of up to 100 contacts, the well-structured nature of the `nwc24f1.bin` file lends itself particularly well to automated parsing. With this in mind, a simple parser was written in Python during the data

analysis phase and is available as `wiifriends.py` in the WiiTools script collection.

9.3.4 Received Messages

Messages can be received by the console in a number of ways. Some messages are received from authorised email addresses or other Wii consoles. These messages appear to be cached in a mailbox file for a short period of time before being transferred to a permanent central store elsewhere in the file system. Other messages are created automatically by system software, for example the automated records detailing the time spent using various features of the device, which seem to bypass the mailbox file and are created directly in the central Wii Message Board repository.

The mailbox used to temporarily hold received messages actually comprises of two files:

- `<mount_point>/shared2/wc24/mbox/wc24recvctl`
- `<mount_point>/shared2/wc24/mbox/wc24recvmbx`

The exact purpose of the `wc24recvctl` file is unknown, however given the seemingly arbitrary structure of the `wc24recvmbx` file, it is considered likely that it contains a table of contents or similar for the temporary mailbox file.

The central storage repository for Wii Message Board messages is the file `<mount_point>/title/00000001/00000002/data/cdb.vff`. As with `wc24recvmbx`, this structure of this file appears to constantly be subject to change although individual messages can be extracted manually with the aid of a hex editor.

The ill-defined structure of these files means that work on the development of a robust automated parser had to be abandoned due to time constraints. It is however possible to manually extract messages of interest, either from the raw data provided by the mailbox files, or through “live examination” of the Wii Message Board application as described in Turnbull's paper (26).

9.3.5 Recently Sent Messages

Few details of sent messages are retained by the Wii Message Board application. The fact that a message has been sent to a particular recipient is recorded in that day's automated usage logging message, but no further details are available through “live examination” of the Wii Message Board.

Through manual investigation of the mailbox files it was discovered that the most recently sent Wii Message Board message is preserved in its entirety in the file `<mount_point>/shared2/wc24/mbx/wc24send.mbx`. The headers of this message are highlighted in Figure 9.5 below. It was determined that only the most recent sent message is preserved after an experiment involving the dispatch of multiple messages in quick succession followed by the capture of a NAND flash storage device image. Analysis of the mailbox files of this new image uncovered details of only the final message of the group.

The data relating to the sender, recipient, timing and content of the message can easily be understood by a manual inspection, and with the aid of a hex editor, it is believed possible to extract the content of any message attachments, which appear to be base64 encoded.

As the file `wc24send.mbx` appears to contain only a single message at any given time, the difficulties relating to automated parsing were greatly reduced

and allowed the development of a robust parsing application for this file. The parser is available as `wiisent.py` in the WiiTools script collection.

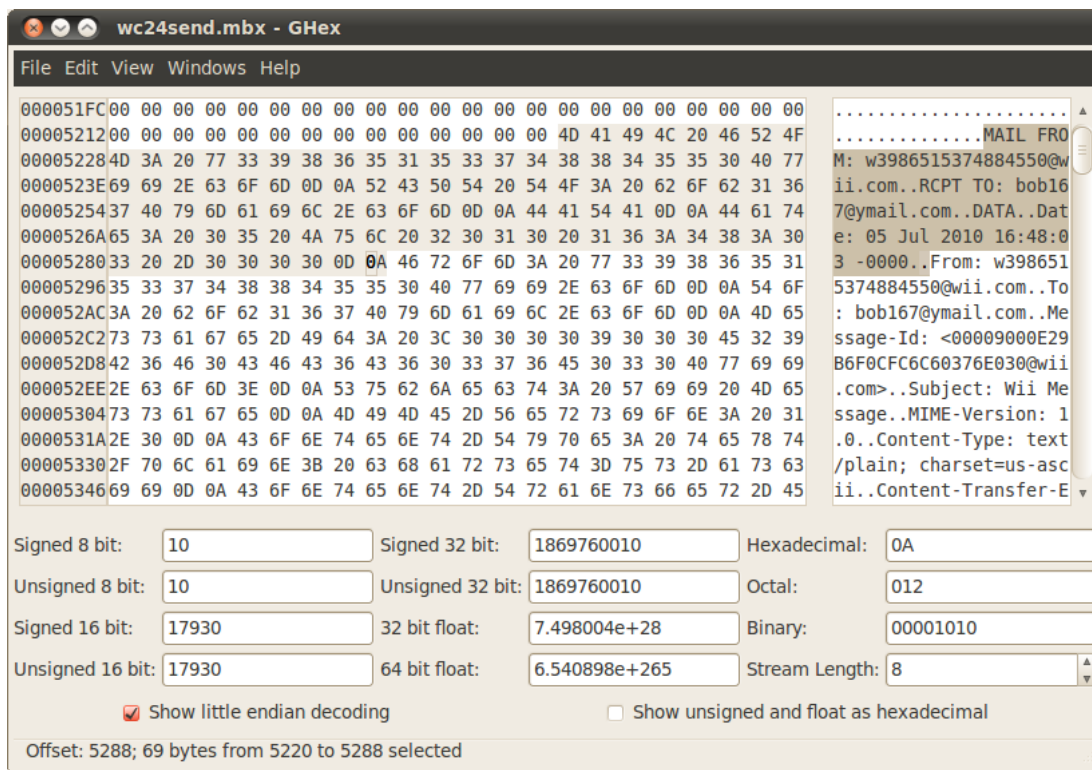


Figure 9.5: Headers of the most recently sent Wii Message Board message

Chapter 10 – Conclusions and Future Work

This chapter presents an overview of the achievements which have emerged from the project, as well as a brief discussion of the major difficulties which were encountered. The chapter ends with a number of suggestions for future work relating to the forensic analysis of the Nintendo Wii game console.

10.1 Project Achievements

The primary aim of the project, to acquire a copy of the data held by the Nintendo Wii's internal NAND flash storage device, was completed successfully.

Following from this, a number of secondary achievements were also attained and are outlined below:

- Extensive research was conducted to formalise knowledge of the Nintendo Wii hardware and software platform
- Demonstrating that a previously captured NAND flash file system image can be restored, and the console reverted to an earlier state to aid in “live examination” of the device
- Demonstrating that a previously captured NAND flash file system image can be decrypted and mounted as a pseudo-Linux file system
- The extraction and interpretation of internet browsing and messaging activity unobtainable through “live examination” of the device

10.2 Difficulties Encountered

The main area where difficulties were encountered in this project was with respect to the efforts to use an external NAND flash controller to acquire a

forensic image of the Nintendo Wii's internal NAND flash storage device. It is likely that the stipulation that nothing be soldered to the console's motherboard contributed to the failure of the experiment as this undoubtedly resulted in a poor quality of connection between the *Infectus2* package and the internal NAND flash storage device.

This approach is widely claimed to be successful in the Nintendo Wii homebrew and hardware hacking communities, and although these claims cannot be independently verified, it is believed that the *Infectus2* chip or similar can be used to successfully read data from the internal NAND flash storage device. However due to the time taken to disassemble the console and the specialist skills and equipment required to ensure a stable electrical connection between the components, it is thought that the "homebrew software" approach is a far more effective method of acquiring a copy of the data held by the internal NAND flash storage device.

10.3 Future Work

While this project has produced a number of achievements there is still significant scope for improvements and further work.

It is felt that the most obvious area is for further investigation of the feasibility of hardware-based NAND flash device imaging, as this would allow data to be acquired from the console, unadulterated by modifications made by the installation of additional software.

In a similar vein, the use of the BootMii application to acquire an image of the internal NAND flash device is akin to using a sledgehammer to crack a nut. One proposal for future work is the development of a specialised NAND flash device

imaging application with a smaller footprint, and ideally, the ability to be executed directly from an external memory card.

Aside from imaging, there is much potential for further investigation of the file system and contents of the internal NAND flash storage device, including the extraction of data which is usually hidden by system software, and thus inaccessible during the conventional “live examination” of the console.

Word Count = 20,706

(Excluding Title, Abstract, Acknowledgements, Table of Contents, Bibliography and Appendices)

Bibliography

1. **Nintendo Co., Ltd.** Consolidated Results for the Nine Months Ended December 2008 and 2009. [Online] 2010. [Cited: 4 August 2010.] <http://www.nintendo.co.jp/ir/pdf/2010/100128e.pdf#page=9>.
2. **IGN.** Wii: The Total Story. [Online] 2006. [Cited: 4 August 2010.] <http://uk.wii.ign.com/launchguide/hardware1.html>.
3. **WiiBrew.** Hardware/NAND. *WiiBrew Wiki*. [Online] 5 October 2009. [Cited: 4 August 2010.] http://www.wiibrew.org/wiki/Hardware/NAND#Supported_chips.
4. **IBM.** IBM Ships First Microchips for Nintendo's Wii Video Game System. [Online] 8 September 2006. [Cited: 4 August 2010.] <http://www-03.ibm.com/press/us/en/pressrelease/20213.wss>.
5. **HackMii.** Keys, keys, keys. *HackMii: Notes from inside your Wii*. [Online] 15 April 2008. [Cited: 4 August 2010.] <http://hackmii.com/2008/04/keys-keys-keys/>.
6. —. IOS: history, build process. *HackMii: Notes from inside your Wii*. [Online] 30 June 2009. [Cited: 6 August 2010.] <http://hackmii.com/2009/06/ios-history-build-process>.
7. —. Wii Menu 4.2: a lack of imagination. *HackMii: Notes from inside your Wii*. [Online] 29 September 2009. [Cited: 6 August 2010.] <http://hackmii.com/2009/09/wii-menu-4-2-a-lack-of-imagination/>.
8. —. System Menu 4.3 update. *HackMii: Notes from inside your Wii*. [Online] 24 June 2010. [Cited: 6 August 2010.] <http://hackmii.com/2010/06/system-menu-4-3-update/>.
9. **WiiBrew.** Signing bug. *WiiBrew Wiki*. [Online] 19 July 2010. [Cited: 6 August 2010.] http://wiibrew.org/wiki/Signing_bug.

10. **Opera Software.** The Internet Channel: Web browsing for your Wii. *Opera Devices*. [Online] 2010. [Cited: 5 August 2010.]
<http://www.opera.com/devices/wii/>.
11. *Xbox 360: A digital forensic investigation of the hard disk drive.* **Xynos, Konstantinos, et al.** 3-4, 2010 : s.n., May 2010, *Digital Investigation*, Vol. 6, pp. 104-111.
12. **Delahunty, James.** Access Xbox 360 HDD in Windows & Backup. *AfterDawn.com*. [Online] 18 December 2009. [Cited: 3 August 2010.]
http://www.afterdawn.com/guides/archive/access_and_backup_xbox_360_hdd_in_windows.cfm.
13. **Gizmodo.** XBox 360 Hacked - Full HD/Memory Card Read-Write. [Online] 13 February 2006. [Cited: 3 August 2010.] <http://gizmodo.com/gadgets/home-entertainment/xbox-360-hacked-full-hdmemory-card-readwrite-154405.php>.
14. *XFT: a forensic toolkit for the original Xbox game console.* **Collins, David.** 2, 2009, *International Journal of Electronic Security and Digital Forensics*, Vol. 2, pp. 199-205.
15. **Huang, Andrew.** *Hacking the Xbox, An Introduction to Reverse Engineering*. San Francisco, CA : No Starch Press, 2003.
16. *17 Mistakes Microsoft Made in the Xbox Security System.* **Steil, Michael.** Berlin : s.n., 2005. *Proceedings of the 22nd Chaos Communication Congress*. pp. 378-390.
17. *Xbox Forensics.* **Burke, Paul K and Craiger, Philip.** 4, 2006, *Journal of Digital Forensic Practice*, Vol. 1, pp. 275-282.
18. *Xbox security issues and forensic recovery methodology (utilising Linux).* **Vaughan, Chris.** 3, 2004, *Digital Investigation*, Vol. 1, pp. 165-172.

19. *Using a software exploit to image RAM on an embedded system.* **Rabaiotti, J R and Hargreaves, C J.** 3-4, May 2010, *Digital Investigation*, Vol. 6, pp. 95-103.
20. **Sony.** PS3 Firmware (v3.21) Update. *PlayStation Blog*. [Online] 28 March 2010. [Cited: 19 August 2010.] <http://blog.us.playstation.com/2010/03/28/ps3-firmware-v3-21-update/>.
21. *Forensic Analysis of a Sony Play Station 3 Gaming Console.* **Conrad, Scott, Dorn, Greg and Craiger, Philip.** 2010. Proceedings of the Sixth Annual IFIP WG 11.9 International Conference.
22. **van Dongen, Wouter S and van Hoof, Alain.** Digital Forensics Research Workshop Challenge 2009 Report. *Digital Forensics Research Workshop*. [Online] July 2009. [Cited: 2 August 2010.] http://www.dfrws.org/2009/challenge/vandongen_vanhoof.pdf.
23. **Lee, Byungkil, Yang, Hongsuk and Yu, Hyeon.** 2009 DFRWS Challenge Report. *Digital Forensics Research Workshop*. [Online] July 2009. [Cited: 2 August 2010.] http://www.dfrws.org/2009/challenge/lee_yang_yu.pdf.
24. **Delahunty, James.** Geohot releases PS3 exploit. *AfterDawn.com*. [Online] 27 January 2010. [Cited: 19 August 2010.] http://www.afterdawn.com/news/article.cfm/2010/01/28/geohot_releases_ps3_exploit.
25. **CmdrTaco.** PS3 Hacked via USB Dongle. *Slashdot.org*. [Online] 19 August 2010. [Cited: 19 August 2010.] <http://games.slashdot.org/story/10/08/19/139235/PS3-Hacked-via-USB-Dongle>.
26. *Forensic Investigation of the Nintendo Wii: A First Glance.* **Turnbull, Benjamin.** 1, 2008, *Small Scale Digital Device Forensics*, Vol. 2, pp. 1-7.
27. **Association of Chief Police Officers.** Good Practice Guide for Computer-Based Electronic Evidence. [Online] 2008. [Cited: 3 August 2010.]

http://www.7safe.com/electronic_evidence/ACPO_guidelines_computer_evidence_v4_web.pdf.

28. *Forensic imaging of embedded systems using JTAG (boundary-scan)*.

Breeuwsma, Ing. M.F. 1, 2006, *Digital Investigation*, Vol. 3, pp. 32-42.

29. **HackMii**. amoxiflash. *HackMii: Notes from inside your Wii*. [Online] 2008.

<http://hackmii.com/2008/05/amoxiflash/>.

30. **IEEE**. IEEE Std 1149.1-2001 IEEE Standard Test Access Port and Boundary-Scan Architecture -Description. *IEEE Standards Association*. [Online] 23 July 2001. [Cited: 8 August 2010.]

http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.1-2001_desc.html.

31. **InFeCtuS**. Infectus Home Page. [Online] 15 January 2009. [Cited: 8 August 2010.] <http://www.infectus.biz/index.php>.

32. **GameCube Linux**. White-linux. *GameCube Linux Wiki*. [Online] 20 June 2010. [Cited: 5 August 2010.] <http://www.gc-linux.org/wiki/WL:white-linux>.

33. **WiiBrew**. BootMii. *WiiBrew Wiki*. [Online] 29 July 2010. [Cited: 7 August 2010.] <http://wiibrew.org/wiki/BootMii>.

34. —. System Menu 4.3. *WiiBrew Wiki*. [Online] 29 July 2010. [Cited: 7 August 2010.] http://wiibrew.org/wiki/System_Menu_4.3.

35. **bushing**. amoxiflash.c. *Google Code*. [Online] 30 May 2009. [Cited: 16 August 2010.]

<http://code.google.com/p/amoxiflash/source/browse/trunk/amoxiflash.c>.

36. **WiiBrew**. Twilight Hack. *WiiBrew Wiki*. [Online] 28 July 2010. [Cited: 8 August 2010.] http://wiibrew.org/wiki/Twilight_Hack.

37. —. Bannerbomb. *WiiBrew Wiki*. [Online] 31 July 2010. [Cited: 8 August 2010.] <http://wiibrew.org/wiki/Bannerbomb>.
38. —. Smash Stack. *WiiBrew Wiki*. [Online] 31 July 2010. [Cited: 8 August 2010.] http://wiibrew.org/wiki/Smash_Stack.
39. —. Indiana Pwns. *WiiBrew Wiki*. [Online] 7 August 2010. [Cited: 8 August 2010.] http://wiibrew.org/wiki/Indiana_Pwns.
40. **Team Twiizers**. Frequently Asked Questions. *BootMii*. [Online] 2010. [Cited: 14 August 2010.] <http://bootmii.org/faq/>.
41. **WiiBrew**. Mini. *WiiBrew Wiki*. [Online] 28 December 2009. [Cited: 11 August 2010.] <http://wiibrew.org/wiki/Mini>.
42. —. Betwiin. *WiiBrew Wiki*. [Online] 16 August 2009. [Cited: 17 August 2010.] <http://wiibrew.org/wiki/Betwiin>.
43. —. wiinandfuse. *WiiBrew Wiki*. [Online] 12 May 2010. [Cited: 14 August 2010.] <http://wiibrew.org/wiki/Wiinandfuse>.
44. —. /shared2/wc24/nwc24fl.bin. *WiiBrew Wiki*. [Online] 28 March 2009. [Cited: 17 August 2010.] <http://wiibrew.org/wiki//shared2/wc24/nwc24fl.bin>.

Appendix A – Console Disassembly

Two tools are required to disassemble the Nintendo Wii console. The first is a common small Philips screwdriver. The second is a relatively uncommon Tri-Wing screwdriver, which was purchased over the internet at a cost of £2.70.

Other tools which may prove useful include:

- A small knife for the removal of rubber feet and stickers
- Tweezers to retrieve recessed screws

It is important to carefully label and separate components upon removal, particularly screws as sizes vary and may only fit certain components.

The disassembly process is as follows:

1. Flip the console upside down and remove the exposed Philips screw. Remove the battery tray from the console.
2. Remove the rubber foot closest to the battery tray slot. Also remove the three square stickers from the bottom of the console.
3. Remove the exposed screws (2 x Tri-Wing, 3 x Philips) shown in Figure 1.



Figure 1

4. Place the console on its right-hand side. Remove the pair of rubber feet at the rear of the console, and the pair of stickers at the front. Remove the exposed screws (4 x Tri-Wing).
5. Place the console upright. Gently remove the plastic flaps which conceal the GameCube controller ports.
6. Remove the exposed screws (3 x Philips) shown in Figure 2.
7. Carefully pull the faceplate from the front of the console. A red/black wire plug must be disconnected from the console body before the faceplate can be completely removed. The faceplate and red/black wire are shown in Figure 3.



Figure 2



Figure 3

8. Remove the plastic back plate from the GameCube controller port area. Remove the newly exposed screws (2 x Tri-Wing, 2 x Philips).
9. Place the console on its left-hand side. Slowly work the plastic cover off of the console. The result is shown in Figure 4.
10. Remove the screws (4 x Philips, each appearing to have a round washer attached) which secure the optical disc drive unit. Tweezers may be useful for retrieving the slightly recessed screws.
11. Carefully tilt the optical disc drive unit upward toward the top of the console. Two wires must be disconnected before the unit can be removed. One white plug which can be gently pulled from its socket and one ribbon strip which is released by lifting the brown catch. These plugs are shown in Figure 5.



Figure 4

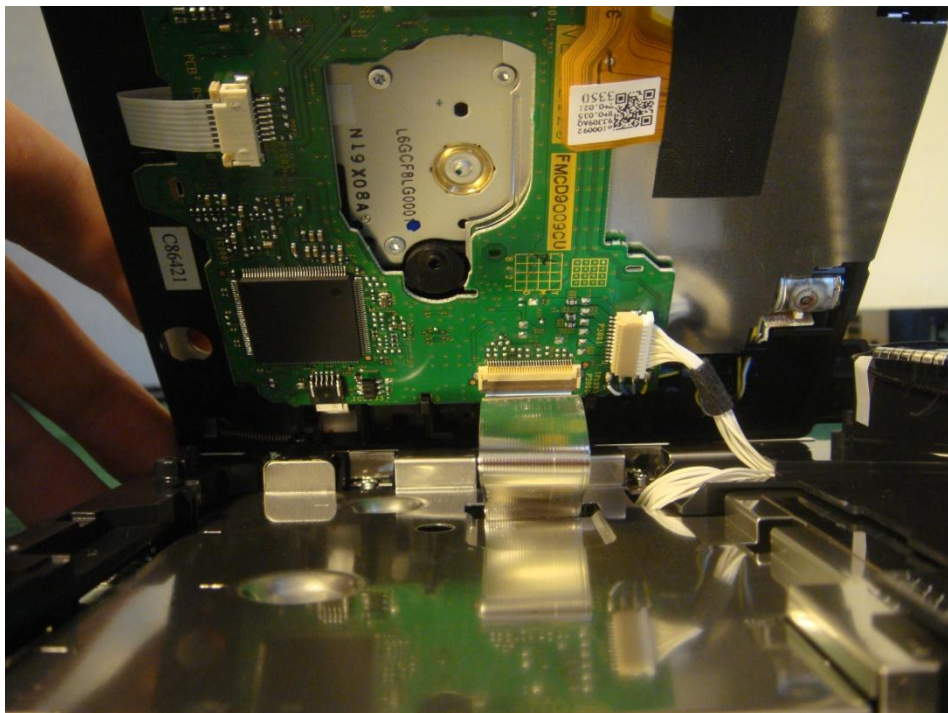


Figure 5

12. Locate and remove the pair of wireless antennae from the console base. One is located near the bottom right corner and can be removed by pressing against the tabs beneath and lifting the card. The second is near the top right corner and requires the removal of 1 x Philips screw. To remove the cards, unthread the connecting wire from the console base. This may require the cutting of two small pieces of electrical tape.
13. At the right-hand side of the console, remove the screws (2 x Philips) which secure the fan. Disconnect the white fan power wire and remove the fan unit, shown in Figure 6.

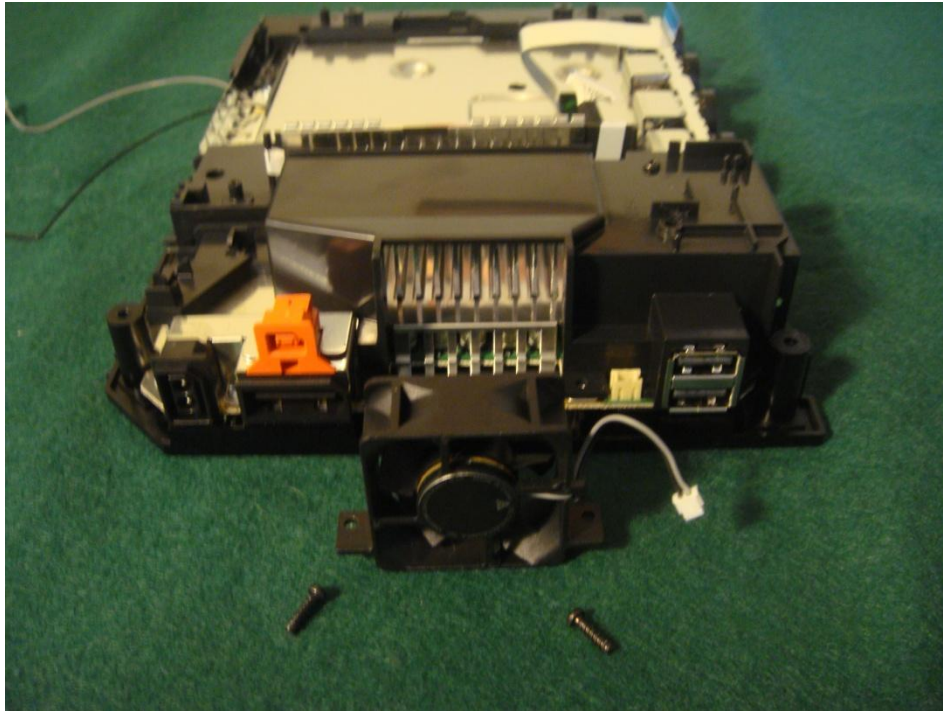


Figure 6

14. Remove the screws (3 x Philips) which secure the plastic fan-flow guide to the right-hand side of the console body.
15. Remove the screws (2 x Philips) which secure the plastic piece in the middle of the console body, and 1 x Philips slightly to the left of the middle plastic piece.
16. Carefully lift the plastic pieces away from the console body. The optical disc drive unit wires and thin metal heat shielding must be carefully manipulated in order to remove the middle plastic piece.

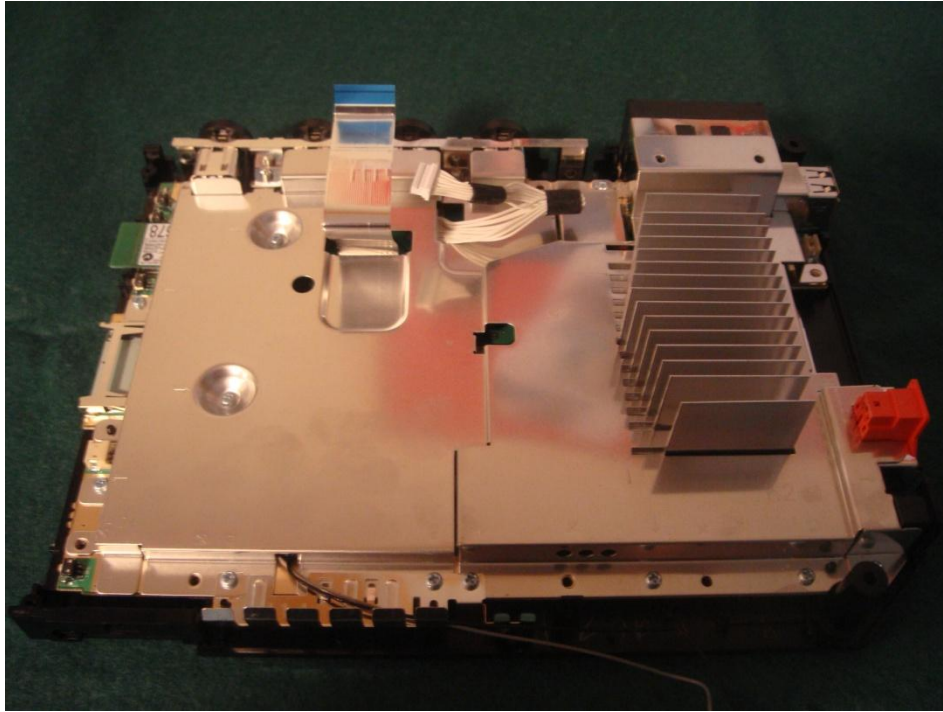


Figure 7

17. Remove the screws (3 x Philips) securing the smaller plastic piece to the left-hand side of the console body. Care must be taken as there is a small square nut which must also be retained. Gently manipulate the metal heat shielding to allow the removal of the plastic piece.
18. As shown in Figure 7, the metal heat shielding is now fully exposed. Remove the screws (11 x Philips) from around the edge of the heat shielding which are marked by ↑, ↓, ← and → symbols.
19. Remove the screws (2 x Philips) from around the edge of the metal heat shielding which are marked by ■ symbols.
20. Remove the screws (2 x Philips) from around the edge of the metal heat shielding which are marked by ▲ symbols.
21. Gently lift away the three pieces of metal heat shielding from the console body, shown in Figure 8.

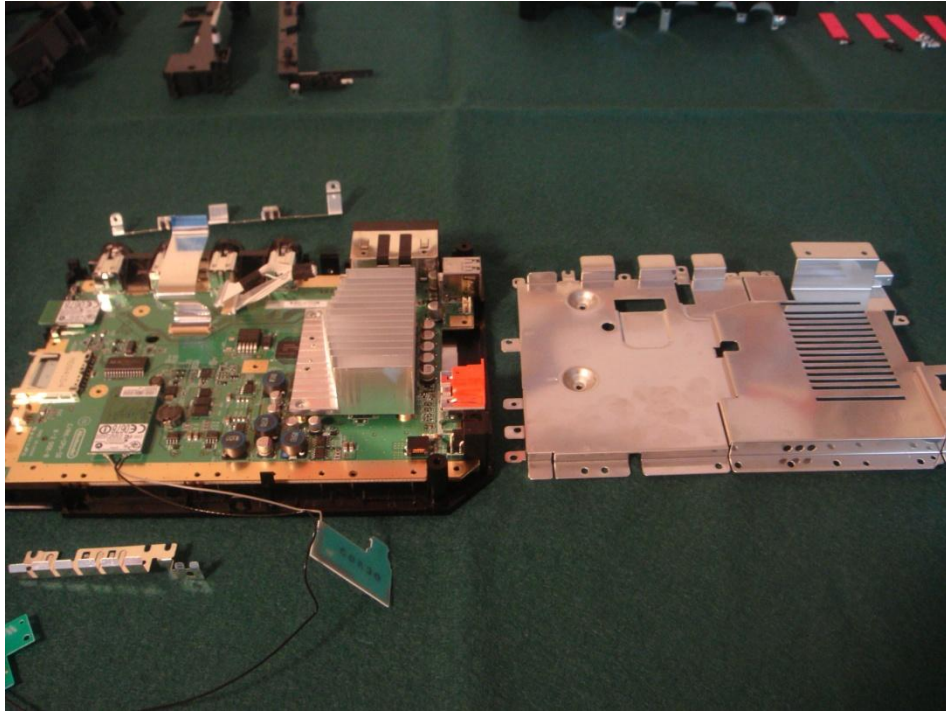


Figure 8

22. Remove the screws (4 x Philips) which secure the large metal heatsink.

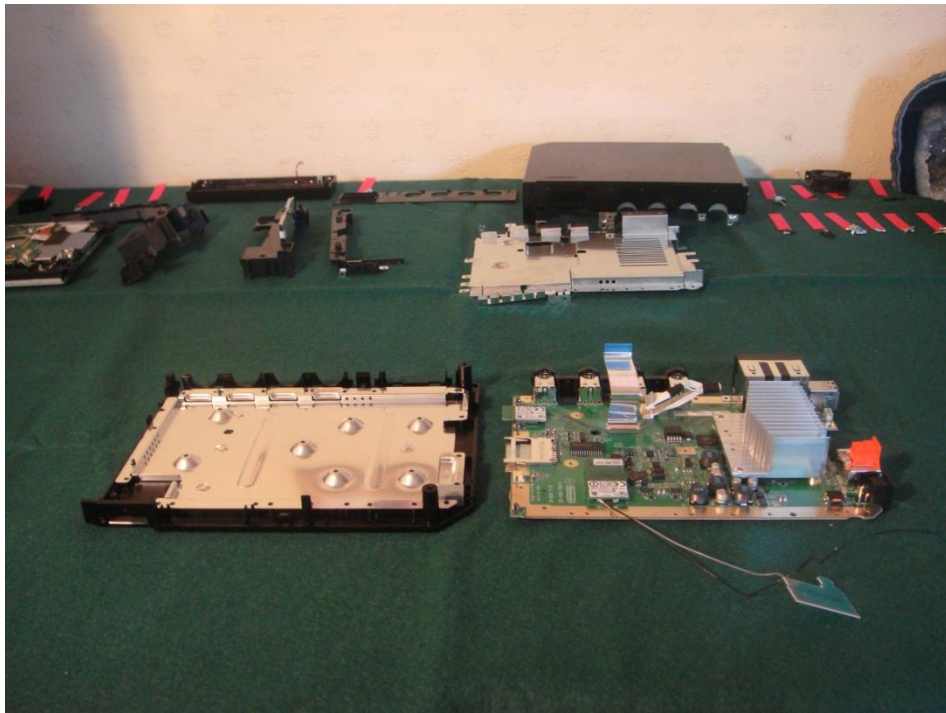


Figure 9

23. Lift the motherboard away from the console base, as shown in Figure 9.

Appendix B – NAND Flash Device Datasheets

The datasheets for the Samsung and Hynix NAND flash storage devices known to be installed in the Nintendo Wii console are only available as large PDF documents and so have been included on the CD-ROM which accompanies this report.

The datasheets can be viewed by navigating to the `/datasheets` directory of the accompanying CD-ROM.

Filename	MD5 Hash Value
<code>/datasheets/Hynix-HY27UF084G2M</code>	<code>600ac1669614c4d894fc6e1b4aca437c</code>
<code>/datasheets/Hynix-HY27UG084G2M</code>	<code>0541960ac99cba38c2301aeb9e20c897</code>
<code>/datasheets/Samsung-K9F4G08U0B</code>	<code>43d45990ef2fbe78601ffabdb10e59aa</code>

Appendix C – The WiiTools Python Script Collection

This appendix describes a small collection of Python scripts which were developed during the Data Analysis phase of the project. The README file is reproduced for reference below:

WiiTools Python Script Collection

Copyright © 2010 Peter Stewart

A collection of Python scripts developed to aid in the extraction of useful data from a Nintendo Wii NAND flash filesystem image.

DEPENDENCIES

The WiiTools scripts were developed and tested using Ubuntu Linux 10.04 and Python 2.6.5

wiitools.py requires wiinandfuse in order to mount NAND filesystem images
wiinandfuse - <http://code.google.com/p/wmb-asm/>

wiifavourites.py

Attempts to extract Favourite webpage title/URL. Cannot handle the frequent, unpredictable changes made to the 'opera.vff' file.

Wiifriends.py

Parses the file 'nwc24fl.bin' to extract and display the contact details stored by the Wii Message Board system.

116isent.py

Parses the file 'wc24send.mbx' to extract the most recent message sent through the Wii Message Board system.

Wiitools.py

A wrapper application which uses "wiinandfuse" to mounts an image, copies specified "interesting" files to a new directory & unmounts the image before running the other WiiTools scripts against the newly copied files.

KNOWN ISSUES

wiifavourites.py: Unlikely to function correctly without modification due to unpredictable changes to 'opera.vff'

wiitools.py: MD5 verification of file copy operations will occasionally fail due to a 'wiinandfuse' ECC-handling bug.

It is assumed that Any script which is to be called by 'wiitools.py' will first be placed on the \$PATH.

CHANGELOG

20100903: 1.0 - Initial release.

The WiiTools Python script collection can be accessed by navigating to the /wiitools directory of the accompanying CD-ROM.

Filename	MD5 Hash Value
/wiitools/COPYING	d32239bcb673463ab874e80d47fae504
/wiitools/README	d6445f3d5cfdd49a614bbf2ce4bd8ff3
/wiitools/wiifavourites.py	87e142549bf71344aeb48e14ca38ce9d
/wiitools/wiifriends.py	04dd9c3d7c2aeccdaf3736fd2305c7c5
/wiitools/wiisent.py	423d8f193e6568ef02903006917cfa62
/wiitools/wiitools.py	14cc0289bc6f5864e45ccf66f0972a56