TDDC03 Projects, Spring 2005

Alternative Xbox copyright protection designs

Albert Holm albho302@student.liu.se

Karl-Johan Karlsson karka728@student.liu.se

Supervisor: Tina Lindkvist

Alternative Xbox copyright protection designs

Albert Holm
albho302@student.liu.se

Karl-Johan Karlsson
karka728@student.liu.se

Abstract

The Microsoft Xbox is a successful gaming console that attempts to include strong copyright protection. We present an overview of the copyright protection schemes and their weaknesses and discuss possible alternative designs for them. We conclude that while the Xbox design is technically flawed, and may be impossible to correct, it is economically quite sound.

1. Introduction

Microsoft released the Xbox in November 2001 into a game console market totally dominated by the Playstation 2. The Xbox was designed as a slightly modified PC, which probably contributed to its low end-user price. These factors made it attractive to people who wanted to use it as a generic PC, who were backed up by hardware hackers who wanted to learn about Microsoft's modifications.

To protect the games revenue stream, Microsoft included copyright protection measures with the intent of only allowing the execution of approved code. We present an overview of these measures, and how they were broken, in section 2.

In section 3 we discuss alternative designs for the Xbox copyright protection systems.

2. Summary of Xbox copyright protection measures

2.1. Goals

Being a PC, much of the Xbox can use cheap, off-the-shelf components. Some of the special components, such as the southbridge and GPU, have also been modified to be used in PC:s. However, the Xbox is still sold at a loss of about $100 per unit [9, 16], to keep the price on par with that of the Playstation 2 [4, 7].

To make up for this loss, Microsoft needs to sell games as well as hardware, so they have a strong incentive to prevent illegal copying. At the same time, the copy protection system must allow the production of new hardware revisions that are still compatible with old software, and not put too great a burden on the developers.

2.2. Design

Viewed from the outside, the Xbox security system seems well thought-out. Binaries are signed, so copy protection measures within games cannot be removed, and they contain a list of which media they are allowed to reside on, so you cannot e.g. copy a game from a DVD to a CD-R and run it. But viewed from inside this system is revealed to rest on a very shaky foundation, rife with possibilities for attacks on the layer below.

Xbox Live, Microsoft's online gaming service, implements checks in addition to those made by the system itself to see if the connecting Xbox has been modified. Nothing conclusive has been proven about how these work, but one theory that has many followers is that Microsoft records a unique identifier from the hard drive in each Xbox the first time it is connected to Xbox Live, and if that ever changes, that Xbox is not allowed to connect to Xbox Live again [12]. Changing the hard drive is popular, since the original one is very small (8 GB) and with a larger one copies of games can be moved over the network and stored on the drive, instead of on burned DVD:s.

Another problem with changing the hard drive is that it is locked using the standard ATAPI security extensions [21], making it impossible to read any data from it until a password has been presented. Defeating this lock once, to be able to look at and modify the contents, is easy—let the Xbox unlock the drive and then, without disconnecting power, swap the IDE cable from the Xbox to a PC [2]. To make the hard drive work in an Xbox, e.g. from a modchip, the password is required. This password is calculated by an SHA-1 hash of several serial numbers and other unique identifiers from hardware inside the Xbox, encrypted with RC-4 [16]. Since the other security mechanisms must already be broken before you have to worry about access-

ing the hard drive, this obscurity provides no extra security, as the values used in the hash and the RC-4 key are easily found from a disassembly of the boot loaders.

Games have to be compatible with all system revisions, so the outside view has not changed, but there are currently eight revisions of the underlying hardware and security systems [8]. The greatest difference is between revisions 1.0 and 1.1, where most of the security system changed, while later revisions have only minor hardware differences.

2.2.1. Revision 1.0 This hardware revision, the one originally attacked by Andrew Huang [13], was protected mostly by security through obscurity.

Since the processor is almost a standard Intel Pentium-III Celeron, the details of its boot process are easily obtained from the architecture manuals [18]. The memory at the address of the reset vector, where the processor begins execution, was traced to a flash chip on the motherboard, which was desoldered and put into a ROM reader to extract the code.

The very first code block executed, 512 bytes at the very top of the address space, is a decoy, containing subtly incorrect code. Rewriting that area in the flash chip has no effect, so the code must be stored someplace else. Checking which buses contained the secret data pointed towards the real boot ROM being hidden inside the MCPX (southbridge) chip. [14] discusses several alternative methods for retrieving it, but the one that was finally used was tapping the HyperTransport bus between the northbridge and southbridge chips. This bus has few (ten) signals, but higher clock frequency than common logic analyzers can handle (200 MHz DDR), so Huang built a custom circuit board with an FPGA to be able to tap, decode and record the traffic.

Reverse engineering of the recorded code showed that it was decrypting the boot loader, found in the flash chip, with RC-4 using a 128-bit key, and verifying that a magic number in the decrypted code corresponded to a hardcoded value. Since RC-4 [20] is a symmetric cipher, knowing the key allows anyone to create a new boot loader, encrypt it, write it to flash, and have the Xbox execute it.

An even easier way of replacing the flash contents is through the LPC bus [17], an approach that has been used by almost all modchips to date. It uses a feature that seems to have been used for testing assembled Xboxes in the factory, but is still present in the shipped hardware.

The LPC bus is intended for connecting low-speed peripheral devices in a simple way, and so has very few connectors (the version used on the Xbox has 15). It was discovered that, by connecting one signal on the flash chip's data bus to ground, the address space formerly mapped to the flash chip was now mapped to the LPC bus instead, so the boot loader, kernel and jam tables are read through the LPC bus. This is probably used to run testing software on newly assembled Xboxes, but can also be used to run a modified boot loader and kernel image that does not perform any checks on the binaries it runs later, allowing play of illegally copied games.

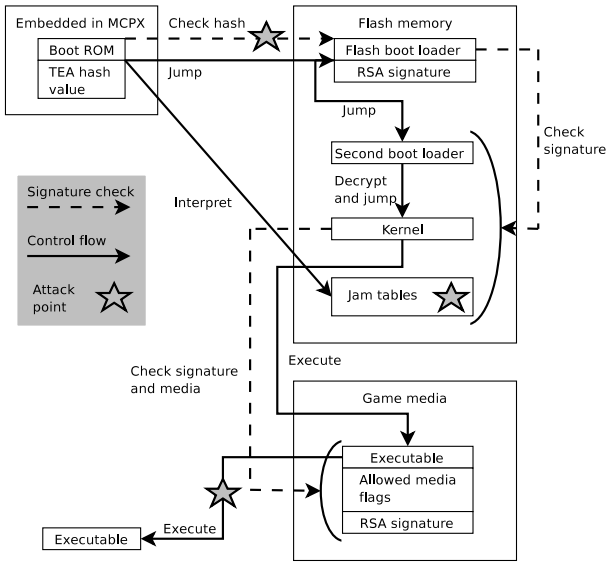

Figure 1. Chain of trust, flow of execution and attack points in system revision 1.1.

2.2.2. Revision 1.1 With the introduction of revision 1.1 in the autumn of 2002, Microsoft changed the security system in a few fundamentally different ways. As in revision 1.0, the real boot ROM was stored in the MCPX chip. Because of the restriction to 512 bytes, a full implementation of RSA or similar algorithm was not possible. Instead, a part of the boot loader in the flash ROM was hashed using the Tiny Encryption Algorithm [22] in Davies-Meyer mode [20] and compared to the reference value stored in the MCPX ROM. The intention was that once the flash boot loader had been verified to be correct, the more space-consuming public-key implementations stored there could be used for verifying the integrity of the rest of the flash ROM.

The flash boot loader would then continue to verify the signatures of various parts of the flash. Among these parts is the so called second boot loader that con-

tains the functionality to decrypt, decompress and execute the kernel and the kernel itself.

Part of the boot ROM is an interpreter for instructions on how to initialize the hardware. These instructions, so called jam tables, are stored in the flash memory, but are not checked for validity until in the flash boot loader, long after they have been used. As the interpreter allows writing to any memory address, it is possible to use them to write instructions to RAM. This was used by Andy Green [16] to write instructions at a specific memory address that would transfer the flow of control back to another part of the flash memory, where code had been loaded in advance.

The Tiny Encryption Algorithm has a weakness in its key schedule which makes it possible to get one controlled change in one bit at the price of one uncontrolled change in another bit in every 64-bit block [19]. This was used to modify the very beginning of the flash boot loader to contain a jump to the memory location written to by the jam table interpreter instead of the intended location.

Other parts of the flash could then be rewritten to get the desired functionality of the boot loaders and the kernel as the code to verify them will either be removed or never executed.

2.2.3. Revision 1.6 Introduced in March 2004 [8], revision 1.6 does not have writable flash memory. The standard modchips are not affected by this, since they use the LPC bus to take control of the motherboard flash address space and divert it to their own flash chips. There is also a class of pure software modifications that are not affected [6]. These use series of buffer overflow attacks to install and run a loader program on the hard disk, which will then be able to load unsigned programs. Both these modifications can be easily reversed, so the Xbox can still be used to play online games through Xbox Live.

The only category of modifications that is hindered by the removal of the writable flash is the so called hardware method [3], that uses buffer overflow attacks to load a program that overwrites the motherboard flash. But this method is very hard to reverse (since the original flash contents are hard to find), and is therefore only useable to permanently turn the Xbox into a Linux computer or to play games without using Xbox Live.

2.3. Summary of problems

Security through obscurity. Most of the security in revision 1.0 came from obscurity, protected by the perceived difficulty of tapping the fast buses on the motherboard. But security through obscurity is ultimately futile, especially when the adversary has complete control of the hardware in question (see e.g. chapter 14 of [11]). Some of this is replaced with security through cryptography in revisions 1.1 and above, but some remain (e.g. the hard drive lock described in section 2.2), providing no real benefit, only more problems in manufacturing.

Time-of-check-to-time-of-use. An interesting twist on the time-of-check-to-time-of-use problem can be seen in the jam tables in revision 1.1. They are interpreted by the boot ROM in the MCPX, but are not hash checked until the flash boot loader. If you do not intend to use the original flash boot loader, the jam tables will never be checked.

Cryptographically weak hash functions. The Tiny Encryption Algorithm, used in the MCPX boot ROM to verify the flash boot loader, is not collision resistant. This means that the flash boot loader can be modified but still be accepted. Later boot stages use SHA-1 hashes and RSA signatures, and are thus not vulnerable in themselves, but when the first step is broken later security doesn't matter.

Debug facilities left in shipped hardware. The LPC bus allows loading an alternate firmware with solderless touch-contacts. There are also probe points on some of the buses, making them easy to find and tap.

Signatures guarantee authenticity, not funtionality. This is a common problem in the mental model people have of cryptography. Retail software on DVD:s is signed, but the signature only says that it's the same software that Microsoft has seen before, not that it functions correctly. For example, the game *MechAssault* has a buffer overflow vulnerability that is used to run the installer for Xbox Linux [5].

3. Design alternatives

The decision to use a well known platform and add only minor changes has both advantages and disadvantages. Among the advantages is that most game developers already know the specifications and how to get the most out of it already from the games released early in the history of Xbox. Of course this also made hardware hackers know just as much about the platform, which made it easier to find out about the differences that make an Xbox unique. Using off-the-shelf compo-

nents makes for a cheap design, but in this case the chosen one has known problems, e.g. buffer overflows.

The specific problem of buffer overflows has both hardware [10] and software [1] solutions available, but the hardware solution requires modified processors, which would have made the console even more expensive, and the software solution extra execution time, requiring a faster and more expensive processor.

One reason for the heavy use of copyright protection in the Xbox is that the hardware is sold at a loss, and Microsoft thus needs every customer to buy several original games to break even. Selling the hardware at break-even prices would obviate the financial need for more extreme measures, and as long as there is still a hardware modification required to circumvent the copyright protection, roughly the same set of users will attempt to circumvent it. There is a great mental hurdle involved in actually opening the box, and that is probably what is significant for the mass market. Customers who are ready to pick up a screw driver and soldering iron to take control of their hardware probably won't care much if it takes an hour or a week to do it.

Selling at break-even prices would, however, have made the current design about $100 more expensive than the Playstation 2, and since the Xbox was Microsoft's first inroads into the console market they might not have been able to gain as much market share as they wanted at that price level.

From revision 1.1, Microsoft uses the Tiny Encryption Algorithm to verify the integrity of the flash boot loader. Later stages in the boot process use RSA signatures of SHA-1 hashes. It would have been much harder to change the contents of the flash chip if every piece of code that exists outside of the MCPX chip had its signature verified with RSA. On the other hand, that would probably result in the need of a larger, and therefore more expensive, MCPX.

Even this solution would not not lead to total security, since it would still be possible (albeit very expensive) to modify or exchange the MCPX chip.

Physically getting to chips and buses on the motherboard could be made more difficult by coating the board and components with epoxy after testing. This would lead to higher maintenance costs, since repairs would then be as hard as physical attacks, probably leading to units being replaced instead of repaired. Protection would still not be complete, since there are solvents and milling machines that can remove such coatings.

Removing the active LPC bus would not have made the design more resistant to professional, directed attacks, but would have necessitated a more complicated solution for taking over the flash address space. The Milksop device [15] is one such solution, but it carries a cost of over $300 in single-item quantities. That almost all modchips to date have used the LPC bus points to its importance in taking modifications to the mass market.

4. Conclusions

When developing the Xbox, Microsoft seems to have tried to make it "secure enough". They knew of several more secure ways to reach their goals of controlling which pieces of software are running and from which media they are executed, but did not see or care about the threats.

From a technical point of view, there are no perfectly secure solutions—it is only possible to influence the cost required to break the security system. From an economic point of view, a balance between the cost of security and lost revenue due to illegal copying is desirable.

We think Microsoft is close to achieve this balance since they are about to release their second generation game console, but a simple way to get even closer would be to remove the LPC bus. If the bus is currently used after the unit is delivered, e.g. when servicing, that use would disappear, but the increased difficulty and cost in producing modchips would probably more than make up for this.

References

[1] 2003. ⟨URL:http://pax.grsecurity.net/docs/pax.txt⟩. Design of the PaX system.

[2] Hard disk hotswap HOWTO. *Xbox Linux Wiki*, 2005. ⟨URL:http://www.xbox-linux.org/Hard_Disk_Hotswap_HOWTO⟩.

[3] Hardware method HOWTO. *Xbox Linux Wiki*, 2005. ⟨URL:http://www.xbox-linux.org/Hardware_Method_HOWTO⟩.

[4] Playstation 2. *Wikipedia*, 2005. ⟨URL:http://en.wikipedia.org/wiki/PlayStation_2⟩.

[5] Software method HOWTO. *Xbox Linux Wiki*, 2005. ⟨URL:http://www.xbox-linux.org/Software_Method_HOWTO⟩.

[6] Version 1.6 warning. *Xbox Linux Wiki*, 2005. ⟨URL:http://www.xbox-linux.org/Version_1.6_Warning⟩.

[7] Xbox. *Wikipedia*, 2005. ⟨URL:http://en.wikipedia.org/wiki/Xbox⟩.

[8] Xbox versions HOWTO. *Xbox Linux Wiki*, 2005. ⟨URL:http://www.xbox-linux.org/Xbox_Versions_HOWTO⟩.

[9] P. Abrahams. Profile Microsoft iBox [sic]–market requires hard work and high investment. *Financial Times*, Sept. 2001. ⟨URL:http://specials.ft.com/ftit/sept2001/FT39KPD96RC.html⟩.

[10] Advanced Micro Devices. *AMD64 Architecture Programmer's Manual Volume 2: System Programming*, 2005. ⟨URL:http://www.amd.com/us-en/Processors/TechnicalResources/0,,30_182_739_7044,00.html⟩.

[11] R. Anderson. *Security Engineering*. John Wiley & Sons, 2001.

[12] D. Becker. Is Microsoft using 'Halo 2' to thwart Xbox hackers? *CNET News.com*, Nov. 2004. ⟨URL:http://news.com.com/Is+Microsoft+using+Halo+2+to+thwart+Xbox+hackers/2100-1043_3-5449160.html⟩.

[13] A. "bunnie" Huang. Keeping secrets in hardware: The Microsoft Xbox case study. AI Memo 2002-008, MIT AI lab, 2002.

[14] A. "bunnie" Huang. *Hacking the Xbox*. No Starch Press, unlimited edition, 2003.

[15] A. Green. The Milksop project. 2004. ⟨URL:http://www.warmcat.com/milksop/milksop.html⟩.

[16] A. Green, M. Steil, and M. Meriac. Its [sic] my box: how the hardware and software traps in the Xbox were beaten and Linux installed. Presentation at the 19th Chaos Communication Congress, 2002. ⟨URL:http://www.ccc.de/congress/2002/fahrplan/event/399.en.html⟩.

[17] Intel Corporation. *Intel Low Pin Count (LPC) Interface Specification*, 2002. ⟨URL:http://www.intel.com/design/chipsets/industry/lpc.htm⟩.

[18] Intel Corporation. *IA-32 Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture*, 2004. ⟨URL:http://www.intel.com/design/pentium4/manuals/index_new.htm⟩.

[19] J. Kelsey, B. Schneier, and D. Wagner. Key-schedule cryptanalysis of 3-WAY, IDEA, G-DES, RC4, SAFER, and Triple-DES. In *Advances in Cryptology—CRYPTO '96 Proceedings*, pages 237–251. Springer-Verlag, 1996.

[20] B. Schneier. *Applied Cryptography*. John Wiley & Sons, second edition, 1996.

[21] T13 Technical Committee. *AT Attachment with Packet Interface—6*, 2001.

[22] D. J. Wheeler and R. M. Needham. TEA, a tiny encryption algorithm. *Lecture Notes in Computer Science*, 1008:363–??, 1995. ⟨URL:http://citeseer.ist.psu.edu/article/wheeler95tea.html⟩.