

# Modern Game Console Exploitation

Eric DeBusschere, Mike McCambridge

## Abstract

The goal of this paper is to provide a high-level, technical summary of the significant exploitations of the Xbox 360 and PlayStation 3. Few academic resources discussing console exploitation exist, especially resources considering the current generation systems, and thus a technical survey of successful exploitations represents a significant contribution to the academic community. The security of both the Xbox 360 and PS3 are discussed, along with the three main Xbox 360 exploits: the King Kong glitch/JTAG exploit, Timing Attack, and the Glitch Attack, as well as the three significant PS3 hacks: Hypervisor Exposure through glitching, PS Jailbreak, and the Root Key Discovery.

## 1 Introduction

Although the successful exploitation of the original Xbox drew considerable media attention and paved the way for sophisticated homebrew software developed by a large community of hackers, pirates, and Linux enthusiasts, modifying or hacking consoles of the current generation has never had the same appeal. Perhaps it is because modern gamers highly value the ability to play online, or the higher degree of technical expertise required to perform current modifications, or maybe the emulators, roms, and homebrew software just cannot compare with contemporary games and entertainment. Whatever the reason, the hacking of current generation consoles has been largely ignored, even in academic communities, despite the fact that modern exploitations are far more complex and interesting, and are mounted against far more sophisticated security systems, than ever before.

## 2 Modern Console Exploitation

### 2.1 Security

Gaming consoles of the current generation were built from the ground up with security as the foremost concern. A console that is easily exploited represents an extraordinary loss in profits for a system manufacturer due to piracy, especially for high-end systems such as the PlayStation 3 that are sold at a loss to the manufacturer with the expectation that profits will be redeemed with the sale of games. The Wii, for example, uses protection almost identical to Nintendo's previous generation console, the GameCube, and thus was hacked almost immediately. The consequence is that over one million Wiis run homebrew software, and a large percentage of these users play pirated games, representing 10s of millions of dollars in losses to Nintendo [25].

Because the Xbox 360 and PlayStation 3 are sold at a much higher price point than the Nintendo Wii, Microsoft and Sony have far more to lose if their systems are exploited, and because of this both companies implement the strictest security policies achievable at a reasonable cost. Software security is relatively inexpensive, and both the Xbox 360 and PlayStation 3 achieve complete software protection by only running signed code, encrypting memory, and utilizing firmware updates

to patch vulnerabilities. Additionally, the PlayStation 3 uses a layered Operating System and secure processors to achieve separation of privilege, forcing hackers to compromise multiple levels before gaining complete access. Hardware protection is more expensive, but both the 360 and PS3 use a secure boot process and a chain of trust seeded by a unique hardware console key. Furthermore, the Xbox 360 uses eFuses, which are physical bits that can be blown to create a hexadecimal value. The 360 blows an eFuse whenever an update is performed, physically preventing a console from downgrading [30].

### Modern Console Security Practices

	360	PS3	Wii	Xbox
Per Console Key	●	●	●	
Manufacturer Private Key	●	●	●	●
Encrypted Memory	●	●		
Firmware Updates	●	●	●	
Secure Boot	●	●	●	●
Internet Banning	●	●		
eFuses	●			
Hypervisor	●	●		
Signed Executables	●	●		●
Security Coprocessor	●			
Encrypted Hard Disk		●	●	

Figure 1: An overview of modern console security

Although the Xbox 360 and PlayStation 3 have rigorous software and hardware protection, the most effective security method is through the internet. Both Sony and Microsoft enforce firmware updates via Xbox Live and the PlayStation network, allowing them to immediately respond to a vulnerability. Also, internet updates force hackers to choose between running an exploited system and playing online, or risk having their console banned. Sony has gone so far as to require updates to play the latest games, once again forcing hackers to choose between homebrew and a complete gaming experience.

## 2.2 Exploitation Strategies

### 2.2.1 Software Attacks

Practically speaking, all modern consoles are impenetrable from a software perspective. This is not to say that software attacks do not occur, but for the most part, they are memory overflows/bugs that are patched quickly via a software update. Oftentimes these mistakes are not made by Sony or Microsoft, but by third party developers that program games or manufacture console accessories. For example, until August 2011, all Xbox 360 exploits used a direct memory attack exposed by unchecked shaders in Peter Jackson's King Kong game [34].

When software attacks are mounted successfully, they often build off of a hardware attack. For example, if a hardware attack can dump the hypervisor or decrypt executables, the code can then be thoroughly examined for bugs that would normally be masked by encryption.

After a software vulnerability is discovered and patched, downgrading to a vulnerable kernel

becomes a possibility, and this strategy is used on both the PlayStation 3 and Xbox 360. Although this does not allow a user to play online or buy the most recent games, for some running Linux is more important.

### 2.2.2 Hardware Attacks

Almost all successful attacks on modern gaming consoles are hardware initiated, and utilize either a Timing Attack-measuring the time execution takes on branches-or a Glitch Attack-slowng down the processor and sending it an appropriate signal. These attacks are effective because physically securing a console is expensive, and likely is not a cost-effective strategy for combating piracy. However, one of the main dangers of hardware attacks is that they often cannot be fixed through firmware updates, and can sometimes be turned off, allowing a user to play online without risk of being banned. Because of this, both Microsoft and Sony fix exposed hardware vulnerabilities on newly sold consoles, so a successful hardware attack will not work on a newly purchased system [28].

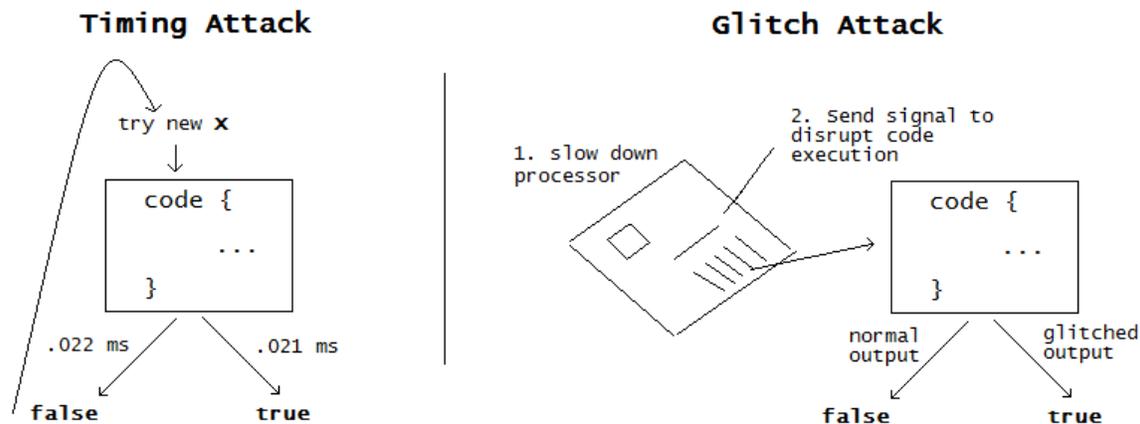


Figure 2: Both Timing and Glitch Attacks use physical exploits to manipulate code execution.

In most cases, hackers use a combination of software and hardware attacks to successfully penetrate a console's security. For example, GeoHot, a famous device hacker, used software commands sent via Sony's OtherOS to prepare the Hypervisor for exploitation. Then, he used hardware-based glitching to sidestep the Hypervisor's security checks. In August 2007, a hardware based timing attack was discovered on the Xbox 360 that allowed downgrading to an exploitable kernel. From here, the software based King Kong glitch could be performed [34].

### 2.2.3 Exploit Timeline

Although the Xbox 360 and PlayStation 3 present tremendous challenges to console hackers, both systems were eventually exploited. Interestingly, the PlayStation 3 was not successfully hacked for more than three years after its launch. Many speculators believe this is because Sony chose to support Linux on the PlayStation 3 until early 2010, only disabling it with the release of the new Slim design. Shortly after disabling Linux, the PlayStation 3 was hacked, and was found to have far greater vulnerabilities than the Xbox 360.

Although the Xbox 360 was exploited early in its lifecycle, the physical eFuses used by Microsoft to prevent downgrading made it exceedingly difficult for the exploit to propagate after it was patched via a firmware update. The real security mistake was allowing a particular manufacturer setting to bypass the eFuses. A timing attack discovered in mid 2007 allowed hackers to take advantage of this setting and downgrade to an exploitable kernel.

## 3 XBOX 360

### 3.1 Microsoft Security Practices

From a software standpoint, the Xbox 360 was designed to be totally secure. Microsoft implements a variety of security techniques designed to make exploiting its system as difficult as possible. First, the Operating System only runs signed code [30], so modifying even a single bit of a game or system software component renders it unusable. Second, no unencrypted executable code is ever written to memory. This is designed to prevent common exploitations via memory snooping. Third, all vulnerabilities are patched whenever a console connects to Xbox Live and downloads the latest update [15]. Additionally, new Xbox 360s are sold with the latest software patches and hardware modifications, and because of this, hacks usually only work on particular system/firmware combinations and require that those systems not update or connect to Xbox Live.

As one would expect due to the cost of secure hardware, the Xbox 360 is more vulnerable to hardware-based exploits, though it still presents some extraordinarily difficult obstacles to hackers. Specifically, the 360 is saddled with a Xenon CPU, which contains 768 bits of eFuse, a technology created by IBM. These individual hardware fuses are grouped into 12 Fusesets and blown in clusters of 8 to make a hexadecimal value that is used in combination with the CPU key to sign and verify firmware software and secure the boot process. The main purpose of these eFuses is to prevent unwarranted downgrading via irreversible hardware changes. When an Xbox 360 connects to Xbox Live and downloads a new update, an eFuse is blown, presenting both a software barrier by way of the new firmware as well as hardware barrier through the updated eFuses. This combination, along with a tightly-controlled boot process, make even low-level exploits challenging [4].

### 3.2 Successful Hacks

#### 3.2.1 Playing Backups: Custom firmware for DVD-ROM.

It did not take long for hackers to write a custom firmware for many of the Xbox DVD-ROM drives (there are a variety of make/models) that simply returns a media value equivalent to what the XBOX 360 expects to receive from game discs for all DVD+R-DL [28]. The actual content of the disc must be identical to the original game, otherwise the digital signature will not remain intact. Note that this is in no way a compromise of the Xbox 360 Operating System or Hypervisor.

#### 3.2.2 The King Kong Glitch (Exposed Nov. 16, 2006. Fixed Feb. 28, 2007)

*Renders all Xbox 360s shipped before Feb. 28, 2007 that were not updated vulnerable to exploit.*

A kernel update (4532) exposed a privilege escalation vulnerability in which an unsigned shader performs a series of unchecked memory exports in certain games, namely Peter Jackson's King Kong [15]. Normally, these memory exports are for writing the results of a pixel shader into physical memory, however the hack uses the shader to overwrite the idle-thread context and instructs the kernel to jump to a particular position in memory while retaining control of some of the registers. Control of all registers is done with a second jump to the interrupt restore handler. These jumps

are possible because, on a particular system call, a 64 bit input address is only verified with a 32 bit check value, leaving the upper 32 bits of the input vulnerable to exploit [30]. Once these two jumps are made, all registers can be filled with determined values, and unsigned code is loaded from a secondary location into memory and executed.

Although the King Kong Attack allows a hacker to take complete control of the Hypervisor and run unsigned code, it requires a user start their console using the King Kong disc everytime they want to enter the exploited state. Because the exploit is a DMA (direct memory access) attack, in theory it could be initiated by any hardware or software that has the authorization to directly manipulate memory, and, shortly after the vulnerability was discovered and exploited, modders found that it could also be triggered directly via hardware means [34].

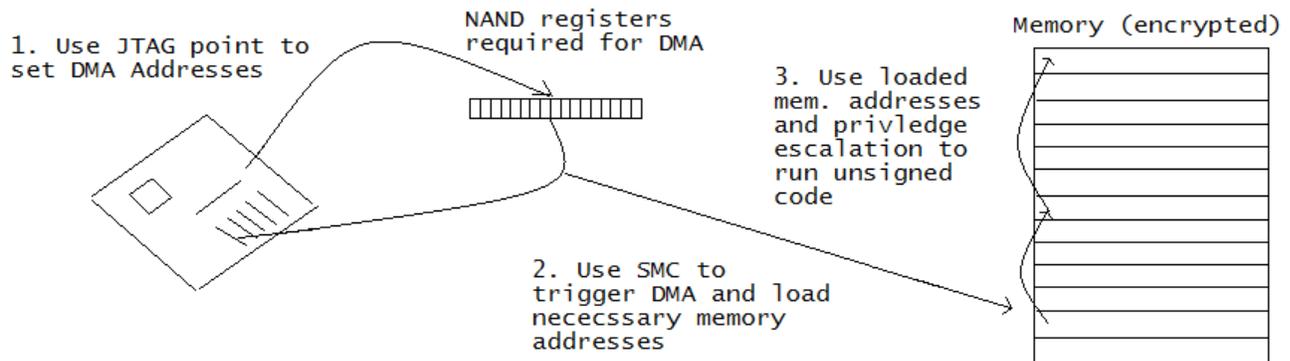


Figure 3: **A direct memory attack, possible with Kernel 4532.**

It had been known for quite sometime that the SMC (System Management Controller) could trigger a DMA. Unfortunately, the SMC cannot write the NAND registers necessary to enable the correct DMA, so it simply executes a DMA to the kernel’s last DMA destination, which is usually address 0. A few months after the SMC was initially explored as a hardware trigger for the DMA attack, a separate group of hackers reverse-engineered the GPU JTAG (joint Test Action Group) and were able to successfully send it commands. JTAG was developed to build points into chips and facilitate the testing process. JTAG alone could not be used to trigger the DMA because its interface is disabled early in the boot process, however, Xbox hackers programmed the GPU JTAG with the DMA target addresses, and then used the System Management controller to initiate the DMA [34], completing the exploit in a faster, more efficient manner than continually using the King Kong disc.

The King Kong glitch was a good first step, but was discovered too late. By the time the general hacking community knew about the vulnerability, Microsoft had patched it for anyone that had connected to the internet and all new Xboxes.

### 3.2.3 Zero-Pairing and The Base Kernel

In early summer of 2007, the MfgBootLauncher mode was discovered. Essentially, this runs the base kernel-the initial kernel shipped with all Xbox 360s-whenver all of 2BL pairing block bits are set to 0. A flash image of this state could be extracted, but no games could be run, nor could the normal dashboard be entered. Technically, in this mode the 4BL refuses to apply update patches and thus enters the base kernel [3].

### 3.2.4 Microsoft Upgrades Security on Zero-Pairing

In what was most likely an effort to increase the security of the 4BL code in response to the discovery of MfgBootLauncher, the 4BL encryption key was changed to use the CPU-key for creation via a CB (4BL) update (CB 1920). This meant the CPU-Key was now required to decrypt the 4BL code during stage 2BL of the boot process.

This CB update also made two very interesting changes to the zero-pairing configuration. Because the CPU-key was now required to decrypt 4BL, simply setting all 2BL pairing bits to 0 no longer worked to enter MfgBootLauncher mode. However, Microsoft also modified the 4BL so that if Zero-Pairing mode was successfully entered (which now required the CPU key), any update-slots that are also zero-paired are applied, regardless of how many eFuses are set. In theory, assuming a hacker knew their CPU-Key, this allowed a bypass of the hardware eFuses [34]. Unfortunately, there was no method to discover a newly purchased Xbox's CPU-Key (unique to each console).

### 3.2.5 The Timing Attack (Exposed Aug. 2007. Fixed early 2008)

*Renders all Xbox 360s shipped before December 2007 that were not updated vulnerable to exploit. It also rendered some newer 360s vulnerable depending on their CB version (has to be 1920). However, during this time Microsoft released consoles with new technical improvements (Zephyr, Falcon, Jasper, and Opus) which were patched on manufacture against the Timing Attack.*

Microsoft made a critical error with CB version 1920. Because the CPU-key was now required to apply the Zero-Pairing configuration, Microsoft assumed it would be impossible to hack, and thus allowed Zero-Pairing mode to bypass the physical eFuses, and apply any Zero-Paired update [3].

This presented a "chicken and egg" problem for the eventual hackers. If a person knew their CPU key, they could run any kernel update they wanted and bypass the eFuses, however, in order to find a console's CPU-key, an earlier kernel version needed to be run-a person could extract their CPU key if they could run the King Kong exploit which required an older kernel.

To solve this dilemma, the hackers took an unmodified base kernel (which was extracted and saved during the initial discovery of the MfgBootLauncher) and patched certain values in the kernel update (SMC, KeyVault, CB, CD, and CE) with current system values, which could be found by taking a flash dump of the current system [34]. This created an unbootable base kernel, because the 4BL lockdown counter hashed from the physical eFuses still showed the current update, not the base kernel.

Although the problem was not solved, it had been shifted from finding the console's CPU-Key to finding a hash value of a lower 4BL lockdown counter, which would enable the base kernel to be loaded and Zero-Pairing state to be entered. Fortunately, this hash value was only 16 bytes long, and was checked with a simple byte-by-byte memcmp function. This meant that, if a timing attack could be executed, there were only 16 bytes \* 256 different values per byte = 4096 necessary tries to decrypt the hash. Experimentation showed that a time difference of 2200 microseconds existed between a true and false byte value. A special flash emulator chip was built that would rapidly change the hash value and reboot the machine-re-flashing with a new hash-value on each try could easily have destroyed the system [2].

### 3.2.6 Successfully Downgrading and Extracting the CPU-Key

Once the hash value for the modified base kernel was found, it could be booted successfully. Because with CB version 1920 Microsoft now allowed any Zero-Paired update to be loaded from the base kernel, kernel 4532 could easily be loaded and the King Kong exploit could be performed. This

was the first time the Xbox 360 was hacked on a wide-scale, because, as discussed before, the King Kong exploit was discovered after it was patched, making it only available to those who had not performed a dashboard update in a long time [34].

### 3.2.7 Deriving CB 1921 from CB 1920. (Exposed Aug. 2009. Fixed Aug. 2009)

*Renders all Xbox 360s of all types shipped before August 2009 that have not been updated vulnerable to exploit.*

Microsoft realized the vulnerability with CB 1920 in early 2008, and most new consoles were manufactured to use a newer CB version, 1921, which used a memdiff function as opposed to memcmp to validate 4BL hash values.

Unfortunately for Microsoft, because the memdiff to memcmp was the only change between CB 1920 and CB 1921 on the original Xenon Xbox 360 machines, using the changes between the 1920 2Bl and 1921 2BL, along with a decrypted 1920 4BL, hackers were able to derive a zero-paired 1921 4BL which hashed validly. This was then ported to Xbox 360s with newer technical configurations (Zephyr, Falcon, Jasper, and Opus) by changing version numbers and slightly modifying other parts of the code [34].

At this point, all Xbox 360s shipped before August 2009 were vulnerable to exploit. However, around the same time Microsoft, likely in expectation of complete current model exploitation, released an invasive dashboard update that overwrote the entire first stage of the bootloader, thwarting all known exploits while statistically destroying one in a thousand consoles due to its intrusiveness [31].

Although slightly modifying the hacked bootloader allowed some newer consoles to utilize the timing attack, there were no major new Xbox 360 exploits for almost three years.

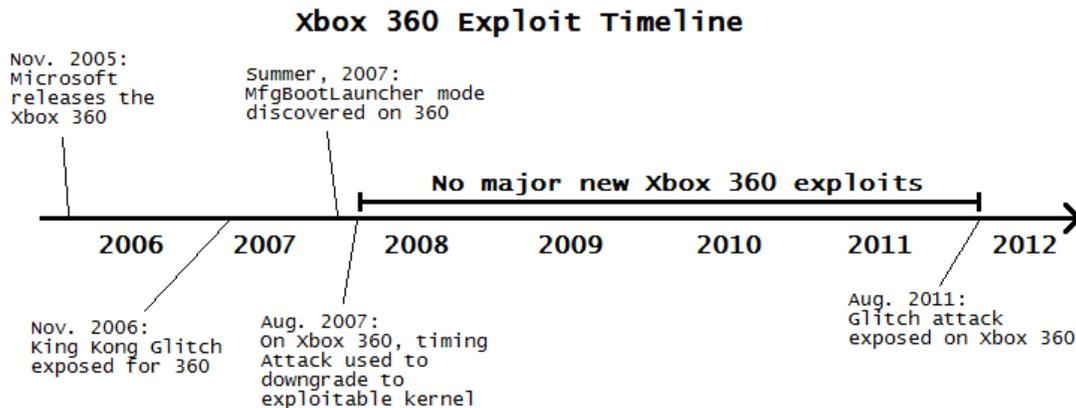


Figure 4: No major new Xbox 360 exploits from Aug. 2007 to Aug. 2011

### 3.2.8 The Glitch Attack (Exposed August 2011. Some newer consoles fixed)

*Renders almost all current Xbox 360s vulnerable to exploit. This exploit cannot be fixed via a software update by Microsoft. Newly released consoles will most likely contain hardware changes that make the Glitch attack more difficult.*

Because there had not been a successful exploitation of the Xbox 360 in almost two years, the Linux/open source community was in desperate need of a new strategy. A glitch attack had been

discussed since 2007, but most hackers thought it would be either require expensive hardware or be impossible to time correctly. One hacker, who had been working on the Xbox 360 for about eight months, became desperate enough to experiment with the glitch attack in the fall of 2010, and successfully ran unsigned code about a month after his first glitch attempt [12].

There are several variations of the glitch attack depending on the Xbox 360 hardware version. Generally speaking, the "Fat" consoles and the newer "Slim" models each require a different attack strategy, with some special processors requiring an even more specific approach.

On the "Fat" consoles, the bootloader glitched is the CB, or the second stage of the boot process. Asserting the CPU-PLL-BYPASS signal was found to slow the CPU execution down by a factor of 128. A counter is maintained, and at a certain count after a particular POST value, a CPU-RESET pulse is sent. If everything works correctly, this CPU-RESET pulse will correspond to the exact moment when the memcmp function is being performed to evaluate the hash of the flash image. Because the hash verification function uses a branch if equals 0 to decide whether or not continue booting, resetting the CPU registers at that exact moment means the branch instruction will evaluate to true, continuing the boot process even though the hash value in the flash image is incorrect [27].

The "Slim" consoles have an interesting change in that their CB, or 2BL, is split into two distinct phases, nicknamed CB-A and CB-B. Furthermore, a motherboard track for the CPU-PLL-BYPASS signal was not found, so another method to slow the CPU clock needed to be discovered. Experimentation showed that the HANA chip could be controlled in a way that would slow down the CPU, however only by a factor of 3, meaning that the CPU-RESET signal would have to be far more precise [35].

It was shown that, after a period of CPU-RESETS (sometimes taking as long as a few minutes), a flash image containing a normal CB-A and a patched CB-B could be validated in spite of its incorrect hash value using the glitch attack. The patched CB-B required, however, was difficult derive, because, compared with previous versions, "Slim" consoles were found to have stricter algorithmic protection of their CB code, utilizing RC4 encryption with the CPU-Key as the encryption-key [35].

In response, the hackers cleverly built upon their glitch of the "Fat" consoles by guessing that the first few bytes of CB code would be the same in the "Fat" and "Slim" consoles. Then, they simply encrypted a small amount of the known code and "dumped" the CPU Key. This exploit was successful because RC4 is a stream-cipher (it encrypts one byte at a time) and has known vulnerabilities when the content encrypted is the same as a previous instance [9].

Because the hackers guessed the first few bytes of plaintext for CB-B using their earlier exploit, they were able to obtain the pseudo-random key stream (the CPU-Key) and sign a patched CB-B. The patched CB-B is modified in a way that it always activates zero-paired mode, doesn't decrypt the CD (or 4BL), and doesn't halt the boot sequence on an incorrect CD hash, effectively allowing any unsigned code to run.

It is important to note that the glitch attack is the first known exploit that does not utilize the infamous King Kong glitch. In fact, the traditional exploits would be impossible at this point because newer CDs (or 4BLs) contain a hardcoded hash of kernel versions 4532 and 4548 (those susceptible to the King Kong exploit), and specifically check that those versions are not being loaded [5].

## 4 PlayStation 3

### 4.1 Sony's Security Practices

The Sony PS3 is similar to the Xbox 360 in that it is completely secure from a software point of view. All software running on the PS3 is encrypted, and, like the Xbox 360, the PS3 relies on hardware protection as the backbone of its security. Specifically, the PS3 uses the Cell Broadband Engine, which was one of the first mass-market processors designed with security as its primary focus [32]. Cell has eight SPEs (synergistic processing elements), which are booted in a way that all of the code controlling the SPE is entirely walled off from the remainder of the system, including the Operating System and Hypervisor. This enables code running on a particular SPE to be verified at any point in time in a way that is completely independent from the rest of the system. Cell stores a hardcoded master root key, which can only be accessed by a particular SPE if the code it is running has been verified and is confirmed to be unmodified [1].

### 4.2 Successful Hacks

#### 4.2.1 The Cell Wall Remains Intact

The PS3 was the last console in the current generation to be hacked, remaining largely unexploited for an unprecedented three years. Its success may have been attributed to the fact that, until firmware 3.21, it allowed consumers to run Linux, much to the appeal of legitimate homebrew software users [25]. Most hackers who exploit gaming consoles do so under the banner of open systems, insisting that they be allowed to run homebrew software on any piece of hardware they purchase. For the most part, until firmware 3.21 (released in April of 2010), Sony allowed them to do so, and a variant of Linux running on the PS3 could access six of the seven SPEs, only being restricted from the main GPU [7].

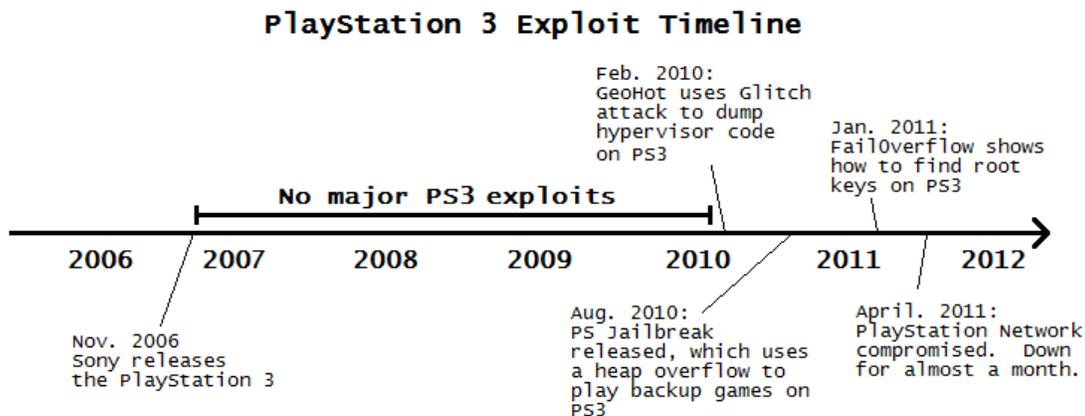


Figure 5: The PlayStation 3 remained unexploited until February 2010.

#### 4.2.2 "Hello Hypervisor, I'm GeoHot."

A 21 year old hacker named George Hotz, infamous for his jailbreaking of the Iphone, decided to enter the PS3 hacking scene late in 2009. In a blog post in early 2010, he revealed how he

utilized the glitching technique to read/write arbitrarily to RAM and dump the entire contents of the hypervisor.

George’s technique was a bit different from the glitch attack used to compromise the Xbox 360 a year and a half later. In fact, one could say it was superior in the sense that it did not require slowing down the processor and could be manually triggered. The pulse could be imprecise because the RAM was prepared in a way that extended the window of opportunity to the point where manual timing was sufficient.

The target of the attack was the hashed page table (HTAB), which, if compromised, would allow access to the main segment and eventually allow George complete control of memory mapping within the system. The entry point of the attack was OtherOS, Sony’s tool which allowed a variant of Linux to be virtualized under the control of the Hypervisor [23].

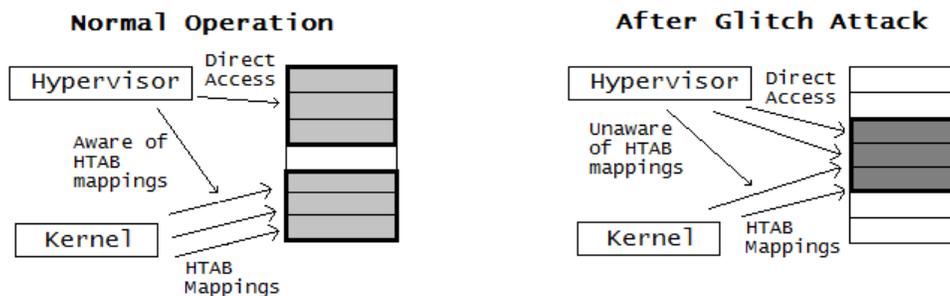


Figure 6: **Software based exploit preparation allowed the glitching to be triggered manually**

First, from OtherOS, George’s software requested the Hypervisor create a large number of HTAB mappings, all of which pointed to a singular buffer it had previously requested the Hypervisor allocate. From here, the software had OtherOS request the Hypervisor deallocate the buffer without properly closing the HTAB mappings. None of this alone is enough to trigger an exploit, and under normal circumstances the Hypervisor would simply deallocate all of the mappings. However, George found that sending a pulse down the FPGA (Field-Programmable Gate Array) memory bus directly after instructing the Hypervisor to deallocate the numerous HTAB mappings causes some of the deallocation commands to fail [23], and a mapping to a deallocated buffer remains intact and invisible to the supervising kernel.

Next, George’s software creates virtualized segments until it finds the segment in which the relevant HTAB is at the freed buffer’s address. From here the software can overwrite the HTAB without being stopped by the Hypervisor, and the exploit writes HTAB entries allowing it complete access to the main segment, and thus all of memory once the Hypervisor switches to the corresponding virtual segment. Although this attack allowed the running of unsigned code, it did not fully compromise the Hypervisor, which would be required to run pirated/back-up games due to their on-disc encryption via a ROM Mark [23].

Sony’s response to the exploit was to disable the OtherOS tool, enraging open hardware supporters enough that a lawsuit was later filed against Sony. Some speculators suggested that Sony dug its own grave by infuriating the open hardware community and giving hackers further motivation to crack the PS3 [25].

### 4.2.3 PS Jailbreak (Exposed August 19th, 2010. Fixed September 6th, 2010)

Renders all PS3s shipped before September, 2010 that have not been updated past firmware 3.42 vulnerable to an exploit that allows game backups to be stored and run from the consoles hard drive.

PS Jailbreak was the first exploit of the Playstation 3 that allowed a user to play backup games stored on the hard drive. It was available for purchase on a USB dongle, and the medium represented an emerging trend in PS3 exploit delivery. Although the exploit infiltrates the system via the USB port, it does not require the dongle be plugged in after the hack is successful, but the creators of the modchip designed the device so that if the dongle is removed the console will shut down. The device was made available in late August, 2010 at the steep price of \$100 to \$130. The high price tag gave hackers even more incentive to reverse engineer it and make it available for free, which was done shortly thereafter.

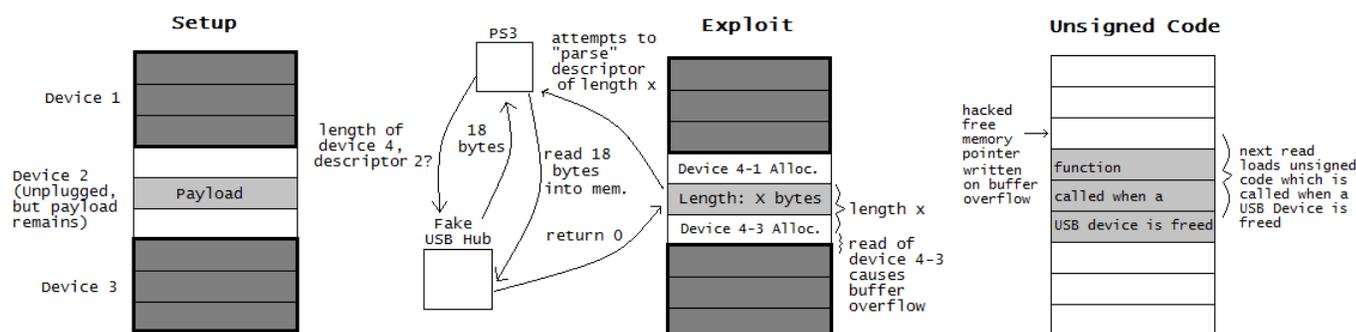


Figure 7: Masquerading as a USB hub allows the PS Jailbreak to exploit a USB device verification vulnerability

Initially, hackers thought the exploit was a stack overflow attack, but it was later confirmed to be a heap overflow [6]. The attack makes use of a short window of time after the PS3 is booted when it is doing nothing but initialize USB devices. The dongle imitates a 6 port USB hub, and plugs/unplugs devices until the heap is corrupted and prepared for exploit.

Without being able to see the code responsible for USB device regulation, it was difficult for hackers to decipher exactly why the exploit was possible, however reverse engineering explained precisely the sequence of events that triggered it. First, the exploit software plugs in fake USB devices with large descriptors into the first three ports on its imitation USB hub. It subsequently sends a signal that the device on port two has been removed, freeing memory between the descriptor for the device on port one and port three. Then, a device is plugged into port four containing three large descriptors, which are allocated in memory between the descriptors for the devices on port one and port three [6]. This sequence of events sets up the heap for exploit.

Next, the second port four descriptor is changed in size from 18 bytes to 0, which causes the third port four descriptor to overwrite the malloc boundary tag of port three. This overwrite is possible because the PS3 determines the length of the descriptor from its first 8 bytes, and shortening the length of the descriptor between the first, length-determining read and the actual read causes the PS3 to attempt to parse the full descriptor from memory even though it was never loaded [25].

At this point the PS3 is now parsing memory it thinks is the second port four device descriptor, but is actually the payload from the previously unplugged port two device descriptor. The port two device descriptor's payload contains specially-planted bytes which tell the PS3 the second port four descriptor buffer it is parsing is extraordinarily long, and thus the PS3 overwrites the malloc boundary tag at the beginning of port three's descriptor. The attack is initialized so that the

substitute malloc boundary tag, which normally points to the next section of free memory, points to a function called after a USB device is freed.

Finally, the dongle sends a signal that an official PS3 service jig has been plugged into port five. A challenge key is sent to the dongle, and it responds with static data, which is allocated in memory at the next free location, planting unsigned code in memory. The dongle then unplugs the device on port three, which calls the overwritten function and completes the exploit [6].

The interesting aspect of this exploit was that it only compromised about 20% of the security system, ignoring the SPEs and even the Hypervisor, yet was still able to play pirated games by simply copying them to the hard drive and patching LV2 (the lowest kernel level) to run them from there.

Sony responded almost immediately to the PS Jailbreak, releasing firmware 3.42 on September 6th, 2010, disabling its use to anyone who updated and fixing the issues related to the Hypervisor and level 2 kernel security [8].

#### 4.2.4 Root Key Exposure (Exposed January 2nd, 2011)

*Renders all consoles shipped before January 2nd, 2011 that have not updated to firmware 3.56 vulnerable to complete exploit. In theory all future firmware should be exploitable, but Sony has added other obfuscation techniques and the PS3s main hackers are legally barred from pursuing further exploits on the console.*

On January 7th, 2011 the hacking team Fail0verflow released an astonishing video in which they detailed how a hacker could take complete control of the PS3, even deriving the root keys.

Essentially, much of the code running on the PS3 is in the common ELF format (the classic Unix extensible linking format), but is signed with secure keys and only decrypted on an SPU in isolation from the rest of the system. While in theory this would make decrypting code extremely difficult, Fail0verflow found that the PS3 makes no attempt to verify that the lower level kernels (those which can ask for code to be decrypted) have not been compromised, so using the level 2 kernel exploit provided by PS Jailbreak, Fail0verflow could simply ask an SPU to decrypt any code it wanted access to [25].

As surprisingly simple as the SPU security hole was, the decrypted high-level code Fail0verflow now had access to revealed a far greater security flaw. As detailed before, the code running on the PS3 is in ELF format and signed using ECDSA (Elliptic Curve Digital Signature Algorithm). To maintain a high level of security, ECDSA requires a random number that is recreated each time code is encrypted. Unfortunately, Sony uses the same random number every time an ELF file is signed, making it trivial for a knowledgeable hacker to extract the root keys. George Hotz, the hacker who exposed the first real exploit on the PS3, released the root keys several days later, as well as a custom firmware (version 3.55) [29].

Although George Hotz's firmware specifically did not allow a user to play game backups, Sony's response was to sue both George Hotz and Fail0verflow, neither of whom returned to the scene. Additionally, Sony released a new firmware which patched many of the security flaws and attempted to disabled downgrading [8]. However, exploits have been revealed which allow downgrading any current firmware, and in theory all future software updates are exploitable.

## 5 Conclusion

Unfortunately for the console hacking community and Linux enthusiasts, game system modification has lost much of its appeal in recent years. This dampening trend is likely due to both the unprecedented security measures built into current generation systems as well as the increased cost

and risk associated with modern day console modifications. Whereas the original Xbox could be modified with a relatively inexpensive, solder-free chip, users nowadays are often forced to perform more complex installations. Furthermore, those who do successfully install a mod chip are not only cut off from the latest games but risk a permanent ban from online gameplay. Considering that both the Xbox 360 and PlayStation 3 had three year periods in which they were largely unexploited, it is not hard to imagine a future in which gaming console security is too sophisticated or too much work for the hobbyist hacker to compromise. These future gaming systems will undoubtedly utilize an entirely online platform, forcing users to remain online even while playing in single-player mode, shifting the backbone of console security to the gaming network, a domain which can be aggressively monitored and quickly patched by its parent company.

## References

- [1] Playstation 3 secrets. [http://www.edepot.com/playstation3.html#PS3\\_Security](http://www.edepot.com/playstation3.html#PS3_Security).
- [2] Timing attack. [http://free60.org/Timing\\_Attack](http://free60.org/Timing_Attack), 2007.
- [3] Xboxhackerbbs. [xboxhacker.org](http://xboxhacker.org), 2009.
- [4] Fusesets. <http://www.free60.org/Fusesets>, oct 2011.
- [5] How we run xell on up to date fat and slim 360s, aka the reset glitch hack. <http://www.xboxhacker.org/index.php?topic=16949.0>, 2011.
- [6] Ps3 wiki. [http://ps3wiki.lan.st/index.php/Main\\_Page](http://ps3wiki.lan.st/index.php/Main_Page), feb 2011.
- [7] Playstation 3. [http://en.wikipedia.org/wiki/PlayStation\\_3#OtherOS\\_support](http://en.wikipedia.org/wiki/PlayStation_3#OtherOS_support), mar 2012.
- [8] Playstation 3 system software. [http://en.wikipedia.org/wiki/PS3\\_System\\_Software#Version\\_3](http://en.wikipedia.org/wiki/PS3_System_Software#Version_3), mar 2012.
- [9] Rc4. <http://en.wikipedia.org/wiki/RC4>, 2012.
- [10] David Becker. Microsoft wants to hire xbox hacker. *CNET News*, Sep 2002.
- [11] David Becker. Lindows ceo funds xbox hacking contest. *CNET News*, Jan 2003.
- [12] GliGli. Glitched! <http://gligli360.blogspot.com/2011/08/glitched.html>, aug 2011.
- [13] Joe Grand. *Game Console Hacking: Xbox, PlayStation, Nintendo, Game Boy, Atari, & Sega*. Syngress Publishing, 2005.
- [14] Joe Grand, Ryan Russell, and Kevin Mitnick. *Hardware Hacking: Have Fun While Voiding Your Warranty*. Syngress Publishing, 2004.
- [15] Anonymous Hacker. Xbox 360 hypervisor privilege escalation vulnerability. <http://securityvulns.com/Qdocument211.html>, feb 2007.
- [16] Ben Heckendorn. *Hacking Video Game Consoles*. Wiley Publishing, 2005.
- [17] Ulrika Hedquist. Crackstation uses game console for hacking. *PCWorld*, Nov 2007.

- [18] Andrew Huang. *Hacking the Xbox: An Introduction to Reverse Engineering*. No Starch Press, 2003.
- [19] iQD. iwd: Ps3 is hacked - the urban legend continues. <http://www.ps3news.com/forums/ps3-hacks-jailbreak/iqd-ps3-hacked-urban-%legend-continues-109555.html>, jan 2010.
- [20] Kakaroto. Status update on the ps3 4.0 hen. <http://kakaroto.homelinux.net/2012/01/status-update-on-the-ps3-4-0-hen/>.
- [21] David Kravets. First criminal trial over game-console modding begins tuesday. *WIRED*, Nov 2010.
- [22] Ben Kuchera. Modder arrest a reminder that most console hacks are illegal. *ARS Technica*, 2010.
- [23] Nate Lawson. How the ps3 hypervisor was hacked. <http://rdist.root.org/2010/01/27/how-the-ps3-hypervisor-was-hacked/>, jan 2010.
- [24] Robert Lemos. Old game machine gets hack trick. *CNET News*, Aug 2002.
- [25] Yifan Lu. 27c3 - console hacking 2010. <http://vimeo.com/18278625>, dec 2010.
- [26] Jon Peck. Why hack your game console? *Free Software Magazine*, Nov 2006.
- [27] Powerslave. How the rgh actually works. [http://techmantis.net/forums/index.php?/topic/952-how-the-rgh-actually-%works/page\\_\\_pid\\_\\_11576#entry11576](http://techmantis.net/forums/index.php?/topic/952-how-the-rgh-actually-%works/page__pid__11576#entry11576), sep 2011.
- [28] Ranger72. What type of mod is right for me? <http://forums.xbox-scene.com/index.php?showtopic=718103>, dec 2011.
- [29] Gregory Rasputin. Ps3 history. <http://ps3history.consolehistories.net/>, sep 2011.
- [30] Michael Steil. Why silicon based security is still that hard: Deconstructing xbox 360 security. <http://www.youtube.com/watch?v=XtDTNnEv1f8>, dec 2007.
- [31] Michael Steil. Dangerous xbox 360 update killing homebrew. [http://www.free60.org/index.php?title=849x\\_System\\_Update&redirect=no](http://www.free60.org/index.php?title=849x_System_Update&redirect=no), aug 2009.
- [32] Jon Stokes. Cell's security architecture: Ibm's prize and sony's achilles heel. <http://arstechnica.com/old/content/2006/04/6694.ars>, 2006.
- [33] tmbinc. King kong exploit - how anon knew what to change? <http://www.xboxhacker.org/index.php?topic=9714.msg62740#msg62740>, may 2008.
- [34] tmbinc. Smc hack. <http://free60.cvs.sourceforge.net/viewvc/free60/imgbuild/hack.txt?revis%ion=1.1&view=markup>, aug 2008.
- [35] Xbox-Scene. The xbox 360 reset glitch hack - new homebrew hack! <http://forums.xbox-scene.com/index.php?showtopic=735015>, aug 2011.