# State of the Hack

*All your consoles are belong to us*

Pierre Bourdon

delroth@lse.epita.fr
http://www.lse.epita.fr

April 26, 2012

# Plan

# Why consoles?

- Gaming consoles are powerful computers
- Probably the most secure electronic device in your house
- Loads of interesting features
- Curious systems: strange architectures, etc.

# Why are they so secured?

- Fear of piracy
- Online cheating (achievements, multiplayer games)
- First production cycles are usually sold at a loss
- If people do not buy games, console manufacturers lose money

# What is this talk about?

- How are consoles secured?
- How do people manage to hack into their game console?
- How could console manufacturers avoid that?
- If you have questions, please interrupt me!

# Plan

2 Wii
- Security model
- Tweezer attack
- Attacks using savegames
- Breaking the chain of trust
- Current state

# Security model

- Every game binary and data is signed
- Game runs under an "hypervision" layer for most system accesses
- System storage encrypted, saves signed and encrypted
- Keys in an OTP in the CPU die
- Chain of trust: boot0 -> boot1 -> boot2 -> SM

# First hack: tweezer attack

- GC compatibility mode allowed to run GC exploits
- In that mode a game can normally only access only part of the RAM
- Used tweezers to connect two address lines and access the other part of the RAM
- Decryption keys were stored there without protection!
- Allows reading the system storage, bootloader, etc. and craft savegames
- Memory bus should be secured to avoid such hardware attacks

# Security 101

- *Do not trust user input!*
- Games parsed savegames assuming they were always valid
- Stack buffer overflows, heap overflows, etc.
- No protection against these basic attacks (/GS, DEP, ASLR, ...)
- Back to the 1990s!

# Epic fail

- boot1 is a bootloader written in ROM which checks boot2 (NAND) signature to see if it was altered
- Used `strncmp` to compare hashes
- Test stopped at the first NUL byte in the hash
- People could replace boot2 and completely break the chain of trust
- How did that pass code reviews?

# Current state of Wii homebrew

- Homebrew channel installed on more than 1M Wii
- More than half of those are unpatchable
- Lots of useful applications released by "amateur" developers
- But also a lot of piracy : USB loaders, burned DVD, ...

# Plan

3 PS3
- Security model
- Attacks on the hypervisor
- Private keys recovery

# Security model

- Game binaries are encrypted and signed, data is not
- Everything runs under an hypervisor
- eFuses to disable debugging features (JTAG, UART, . . . )
- System storage encrypted, saves not transferable
- Chain of trust: cellinit -> bootldr -> lv0 -> lv1 -> lv2 -> appldr
- Allows arbitrary code execution (OtherOS) with no access to RSX

# Memory glitch attack

- Same idea as the Wii tweezer attack: glitch the memory bus
- Use a syscall to map an RWX page
- Glitch the memory to remap the hypervisor
- Full write access on the hypervisor
- Lots of countermeasures could have been used here (hashes memory, `W^X`, etc.)
- Sony fixed that by removing OtherOS support. Bad idea.

# USB hub attack

- Uses a specially crafted USB device which acts as a 6 port hub
- Generates invalid USB descriptors which confuse the hypervisor
- Simulated USB attach/detach requests causes heap overflow
- Overwrite a vtable to replace an object virtual destructor
- The PS3 then executes the given shellcode
- Could have been avoided with ASLR / DEP... 1990s all over again

# ECDSA fail

- All PS3 signatures are done using the ECDSA algorithm
- ECDSA requires a secure random number generator to be safe
- Instead of that, Sony basically used a constant number...
- Knowing that, we can recover the private key using two signatures and the public key
- Homebrew developers can sign everything they want!

# Patch

- Sony fixed this security problem by using new keys in firmware version 3.60+
- Nobody has publicly reversed the firmware to get the new public key *yet* so 3.60+ binaries can't be decrypted
- People can downgrade by flashing their PS3 NOR with a modchip, but can't play more recent games
- People are making a business of decrypting 3.60+ binaries for piracy...

# Plan

4 Xbox 360
- Security model
- HDDVD drive attacks
- Hypervisor exploit
- Reset glitch

# Security model

- Probably the most secure of all those consoles
- Game binaries are encrypted and signed, data is not
- Everything runs under an hypervisor
- eFuses to avoid downgrading the system version
- System storage encrypted, saves not transfereable
- Optional hashed memory to avoid DMA/glitching attacks
- Chain of trust: ROM bootloader checks the NOR bootloader signature, . . .

# Using the HDDVD drive firmware

- Microsoft outsourced their HDDVD drive production
- Firmware stored in a non secure chip
- Could be overwritten to do DMA and modify read data
- Alone, doesn't allow for an hypervisor exploit because of RAM hashing

# Xenos attack

- Xbox 360 GPU
- From there you can modify read shaders
- Xenos shaders have a feature with which you can write back to RAM
- Another DMA vector. . . still no hypervisor exploit

# Syscall hack

- Wrong computation of the syscall address from the syscall number
- Could be used to make HV call anywhere in memory and execute user specified code
- Patched by Microsoft very fast, and downgrades are not possible

# More hardware attacks...

- When booting, the first thing the console does is running the ROM bootloader and checking the NOR bootloader hash
- Sends progress on UART port
- CPU clock is not in the chip
- We can manipulate the CPU clock to make it slower
- When using the CPU reset line during the hash check, it sometimes glitches and says the hash is valid
- Allows replacing the bootloader and breaking the chain of trust
- Still unpatched!

# Plan

# What is left to hack?



- 3DS - encrypted storage, encrypted games, not a lot of hardware attacks were tried yet
- PSVita - very recently released, not a lot of stuff were tested on it
- iPad2 - not really a game console, but shares a lot of security aspects

# Questions?

- @delroth_
- http://blog.delroth.net/