EXHIBIT L

DECLARATION OF RYAN BRICKER IN SUPPORT OF *EX PARTE* MOTION FOR TEMPORARY RESTRAINING ORDER AND ORDER TO SHOW CAUSE RE PRELIMINARY INUNCTION; ORDER OF IMPOUNDMENT

Console Hacking 2008: Wii Fail

Is implementation the enemy of design?

marcan and bushing
Team Twiizers

Introduction: The Wii

Design goals:

- Cheap
- Sold at a profit
- Small, sleek, reasonably portable
- Backwards compatible with the GameCube
- Support for common standards
    - WiFi, USB, Bluetooth, SD
- "Always on" networking: WiiConnect24

Primary hardware overview

Improve and extend the GameCube

- IBM PowerPC 750CL "Broadway" @ 729Mhz
- ATI "Hollywood" GPU+DSP @ 243Mhz
- 24MB 1T-SRAM (*MEM1*) + 64MB GDDR3 DRAM (*MEM2*)
- Standard GameCube I/O (pads, memcards)
- 480p video output
- USB 2.0, SD, WiFi, Bluetooth
- 512MB NAND Flash (SLC)
- Modified DVD reader (Dual Layer)
- Security subsystem

Security architecture

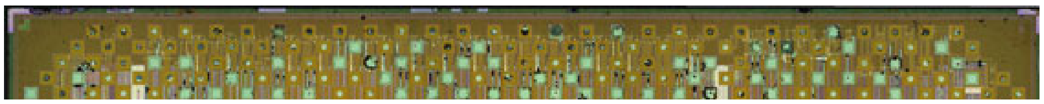
Two custom processors

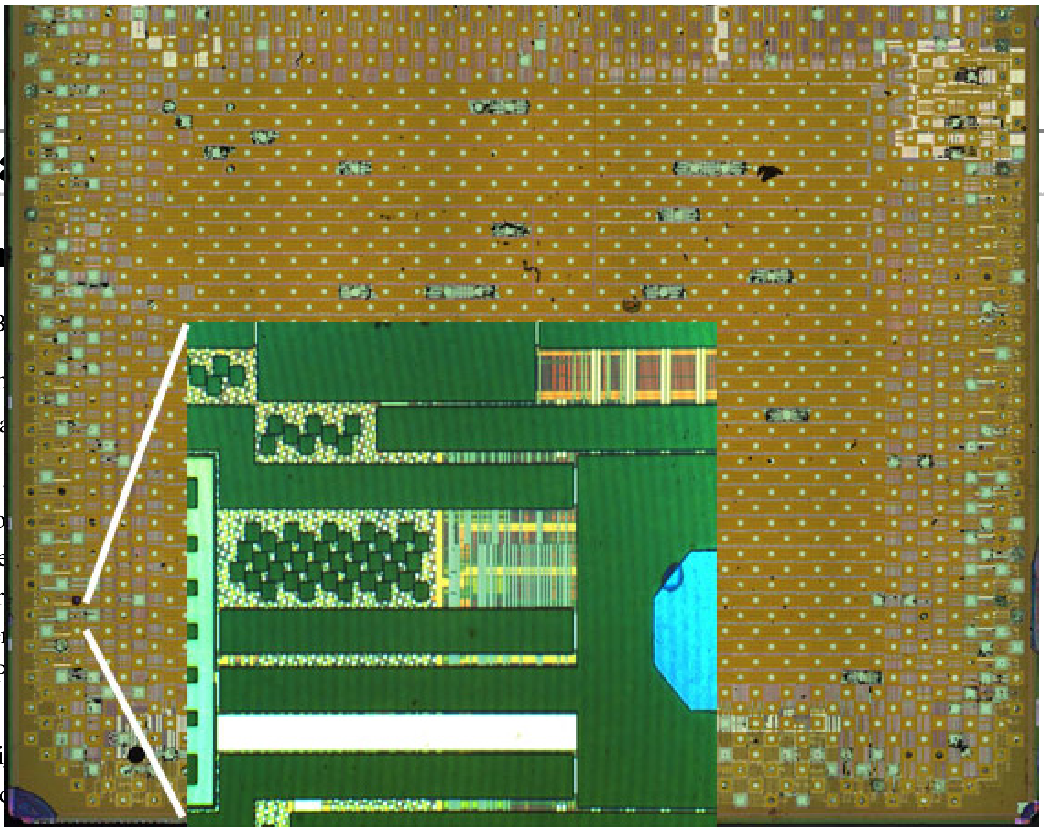
PowerPC 750CL "Broadway": Fast and insecure

- No OS! Games run on "bare metal". Fast and cheap.

Hollywood: ATI Graphics, peripherals, memory, "IO Bridge"
IO Bridge is a NEC ARM926 SoC: "Starlet"

"Starlet" (photo by Flylogic)





Security

Two custom

PowerPC 750CL "B

- No OS! Gam

Hollywood: ATI Gra

- IO Bridge is
- Runs a custo
- Many feature
  - Secur
  - Drive
  - HTTP
  - Runs
- All code is si
- All abstracte

Secure Boot process

Code is booted directly from an internal 512MB NAND Flash chip

- boot0: small (1.5k) bootloader mask ROM in Hollywood
- boot1: 2nd-stage loader (17k) in flash
  - Verified against a factory-burned hash
- boot2: main loader (160k) in flash (mini IOS)
- IOS: ARM code (2MB) read from flash filesystem, running on Starlet
- Menu: PPC code read from flash filesystem, and pushed to Broadway
  - boot2, IOS, Menu are signed using RSA

Multi-stage process reduces cost and increases flexibility

Software titles

- Channels, Games, WiiWare, System software are all **titles**
- A signed package of software, identified by a TitleID
- TMD: Title MetaData signs and describes the contents
  - Contains SHA-1 hashes of the content files
  - Permissions, group IDs, region locking
- eTicket: Your *license* to use the title (the key)
  - Contains the encrypted AES key used to decrypt the title on installation
    - The master key is stored in OTP ROM and hard to extract
  - May contain time limits
- TMD and eTicket are signed using RSA-2048
- eTickets may be specific to one console

Wii Optical Discs (WODs)

- Modified DVD format (with physical anti-duplication measures)
- Discs contain multiple partitions (update, game)
- Partition data is encrypted using AES (and the eTicket key)
- Each block is hashed using SHA-1
  - A hash tree traces each block to a master hash
- All data and game assets are signed and encrypted this way!
- The "root" signature is in the TMD
- The encryption key is in the eTicket

IOS

Custom micro-kernel OS designed by BroadOn (California)

- handles most I/O to Broadway
- talks to Broadway via an IPC interface
- provides high-level network API
- decryption / authentication of Broadway's code
- enforces POSIX-like FS permissions
  - Games (Title IDs) are users, vendors are groups
  - IOS tracks the current permissions of Broadway
  - Broadway can't see system files
- Starlet controls Broadway boot and memory limits
- Modular architecture - modules run as isolated userspace processes
- Kernel runs on internal SRAM, userspace uses the top 12MB of MEM2
  - Broadway can't use this area (it's protected)

All in all, this is a pretty secure system.

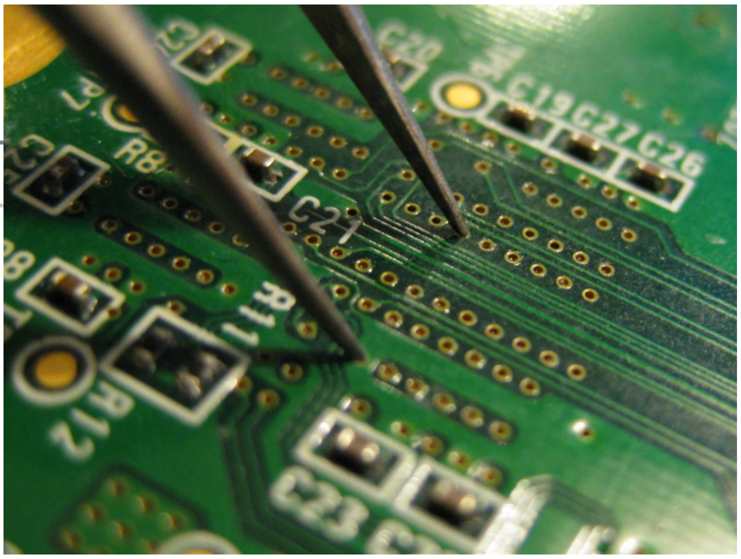
Breaking in: GameCube Mode

- GameCube software is totally unsigned, but runs in a sandbox
- The DVD drive is similar to the GameCube's
  - Outsourced to Matshita
- GameCube drivechips were easily "ported" to the Wii
  - Wii game piracy
- GameCube homebrew possible via GC mode discs
  - But sandboxed, no IOS running, no Wii features
- Wii always boots first into native mode, then reboots into GameCube mode
- GameCube mode uses the first 16MB of MEM2 (as ARAM)

Hack: Tweezer Attack!

- Upper 48MB is not cleared when entering GameCube mode
- Hardware register prevents Broadway from accessing memory
- Address lines of DRAM chip can be manipulated with hardware
- Possible to temporarily move 16MB "window" throughout DRAM

- Dump the entire 64MB to a computer for analysis (bit-banged joypad line)
- Hmm, there's IOS



Keys

Per-console keys

- ECC private key
- ECC public certificate
- NAND AES key
- NAND HMAC key

Global keys

- Common key 0
- SD key
- Root certificate
- New common key 1 (Korean)

Key locations

- Hardcoded in IOS:
  - SD key
  - Default common key 0
- One-time-programmable memory area (Hollywood):
  - Common key 0
  - ECC private key
  - NAND AES key
  - NAND HMAC key
- Serial EEPROM die (inside Hollywood):
  - ECC public certificate
  - Common key 1 (Korean only)

Inside IOS

- Isolated userspace processes
- Talk to kernel using system calls
  - Privileged hardware access
  - Process/thread management
  - Talking to other processes
- Inter-process communication using standard calls
  - open(), close(), read(), write(), seek(), ioctl(), ioctlv()
- Processes set up devices under /dev/
  - ES (eTicket Services, /dev/es): application security
  - DI (Drive Interface, /dev/di): DVD driver and crypto
  - Many more...
- Broadway can issue inter-process calls too
  - Appear to come from PPCBOOT process

Signatures

- All RSA signature comparison is done by one function
- ES_VerifySign uses hardware SHA-1 engine, and software RSA
- Before loading content, TMD must exist containing SHA1 of that content
- SHA-1 of TMD is signed by Nintendo
- When validating TMD, IOS decrypts RSA signature to produce expected TMD hash
- Real TMD hash is calculated, and the two are compared

RSA primer

- RSA signature verification is very simple
- $c = m^e \bmod n$
  - *m*: encrypted signature
  - *c*: decrypted signature
  - *e*: public key exponent
  - *n*: public key modulus
- *c* is created by taking the SHA-1 of what is being signed, and prepending constant padding
  - the padding is required to ensure the security of RSA
- Verification compares the resulting *c* with the expected *c* from the above calculation

RSA the Nintendo Way

Hack: Fakesigning

- RSA: $0^e \bmod n$ is 0 for any *e* and *n*
  - All zero input means all zero output!
- This means that the SHA-1 that IOS compares is all zeroes too
- This will compare equal to any SHA-1 that starts with 00
- Bruteforce it!
  - Change some bytes of the data until the SHA-1 starts with 00
- Fakesigning lets us:
  - Use unsigned games
  - Install an unsigned System Menu
  - Install unsigned IOSes
  - Install an unsigned boot2

Fakesigning Demo!

Data:

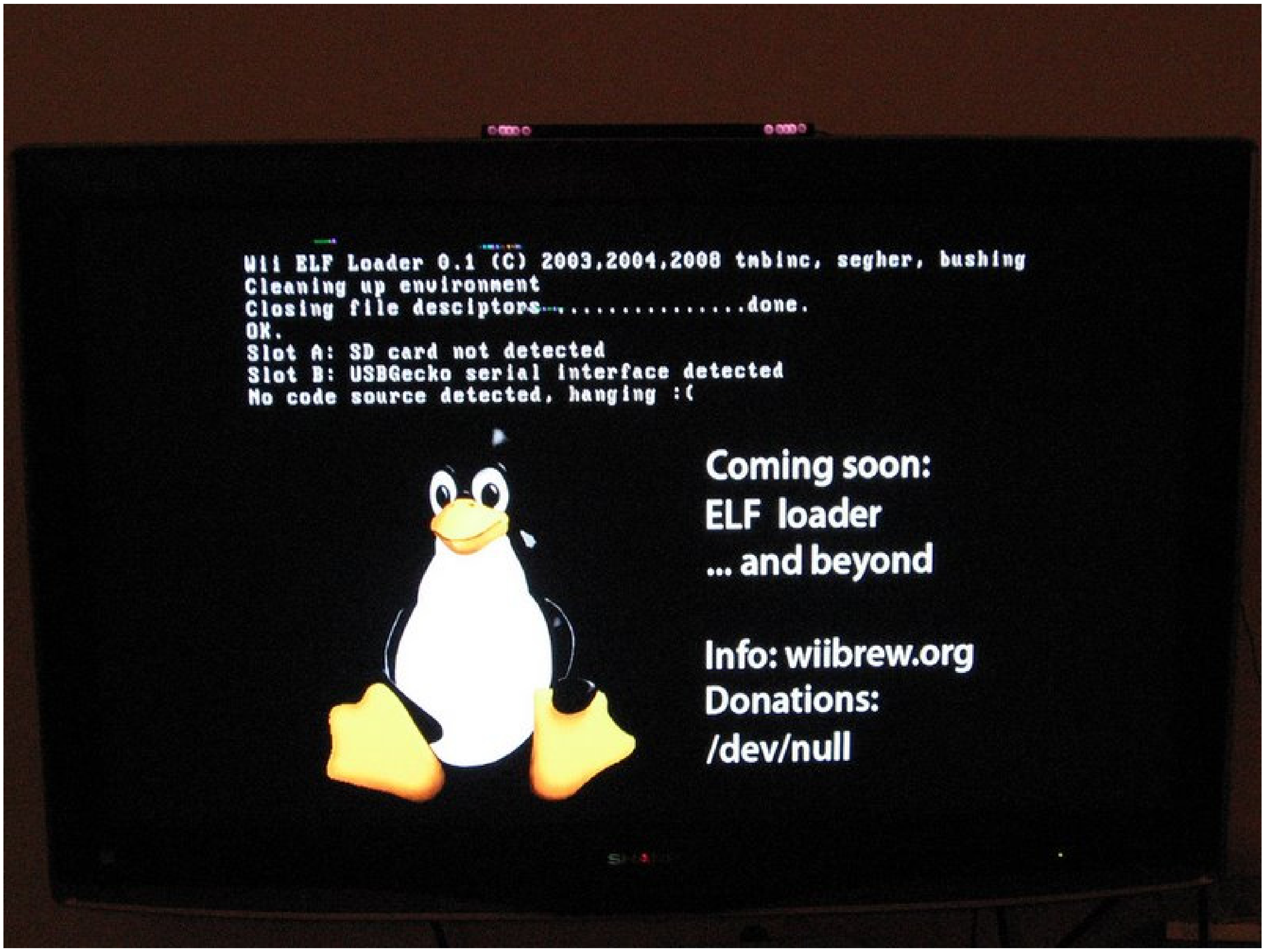
Fakesign!

Hack: Twilight Hack

- Savegames are exported to an SD card signed with the console's private key
- We can extract the keys, so we can sign any savegames too

- Exploit a stack buffer overflow in The Legend of Zelda: Twilight Princess
- Direct execution to a stub inside the savegame
- Load a loader from another file in the savegame
- Loader reads an executable from an SD card
- Easily run arbitrary Broadway code

The Birth of a Hack

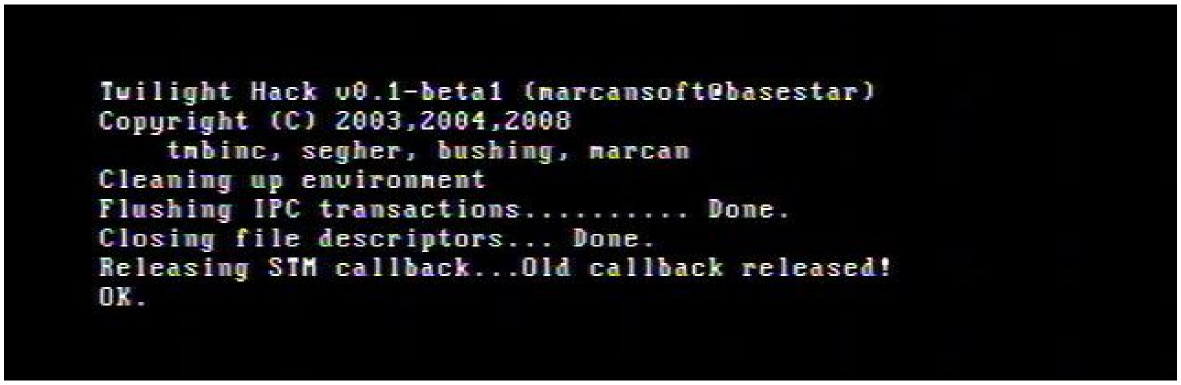


```
Wii ELF Loader 0.1 (C) 2003,2004,2008 tmbinc, segher, bushing
Cleaning up environment
Closing file desciptors...............done.
OK.
Slot A: SD card not detected
Slot B: USBGecko serial interface detected
No code source detected, hanging :(
```

Coming soon:
ELF loader
... and beyond

Info: wiibrew.org
Donations:
/dev/null

Twilight Hack in 10 steps



```
Twilight Hack v0.1-beta1 (marcansoft@basestar)
Copyright (C) 2003,2004,2008
    tmbinc, segher, bushing, marcan
Cleaning up environment
Flushing IPC transactions.......... Done.
Closing file descriptors... Done.
Releasing STM callback...Old callback released!
OK.
```

Life of a typical exploit

1. You find a bug
2. You use the bug for a while
3. Vendor fixes bug
4. GOTO 1

Life of the Twilight Hack

1. You find a bug
2. You use the bug for a while
3. Vendor tries to detect exploit and remove it
4. Vendor botches the detection
5. You keep using the same tweaked bug
6. Vendor **really** detects the exploit this time

DI_Verify

- Multiple versions of IOS are stored in flash for compatibility
- When booting a game, the System Menu loads its requested version
  - this is okay, as long all versions of IOS are secure
- When IOS reloads, it forgets the current state
- When DI opens the disc partition again, it sends the TMD and eTicket to ES
  - Permissions are established according to the currently inserted disc
- ES sets up the new permissions
- This is a private ioctlv in ES

Abusing DI_Verify

- ES doesn't check the requesting process!!
- We can run the same ioctlv from Broadway (as PPCBOOT) and pass in any TMD and eTicket
- Allows privilege escalation (*sudo*)
  - Modify saved data of any title
- GroupID 0x00 is reserved for "system stuff"
  - We can set this GroupID in the TMD and fakesign it

    - Modify executable code of any title
    - Extract secret keys or executables to downloadable applications (WiiWare/Virtual Console)
- We call this ES_Identify :-)

Abusing DVD Video

- Disc drive firmware (ROM) rejects non-Wii discs when loading games
- Can't write a warez loader, because you can't even read the disc
- DVD Video commands left in firmware, to support potential DVD Video channel
- IOS will not let you use those commands ... unless you set a magic bit in TMD
- Result: Homebrew ability to play DVD Videos without firmware patching
- Result: DVD-Rs look a lot like DVD Video discs, so someone wrote a warez loader
- Tried to inform Nintendo about this, they responded by harassing us
- Moral: Don't bother

Vendor Response

- First unsigned code demonstrated: Dec. 2007
- First optional fix for strncmp bug: 21 Mar. 2008
    - Near useless, limited to one new IOS
- First operational fix for strncmp bug + Twilight Hack "fix": 16 Jun. 2008
    - Limited to System Menu IOS, easily bypassed; hack fix is a failure
- First near-complete rollout of strncmp fix: 23 Oct. 2008
    - Fairly effective against VC piracy
- Second Twilight Hack fix attempt: 17 Nov. 2008
    - Still a failure

Crypto Problems

- Bug in signature verification (hash check)
- Keys stored in external GDDR3 RAM in cleartext
- Memory not cleared when entering GameCube mode
- Signatures verified at installation time only
    - Chain of trust easily breakable via raw NAND access

Broadway API Problems

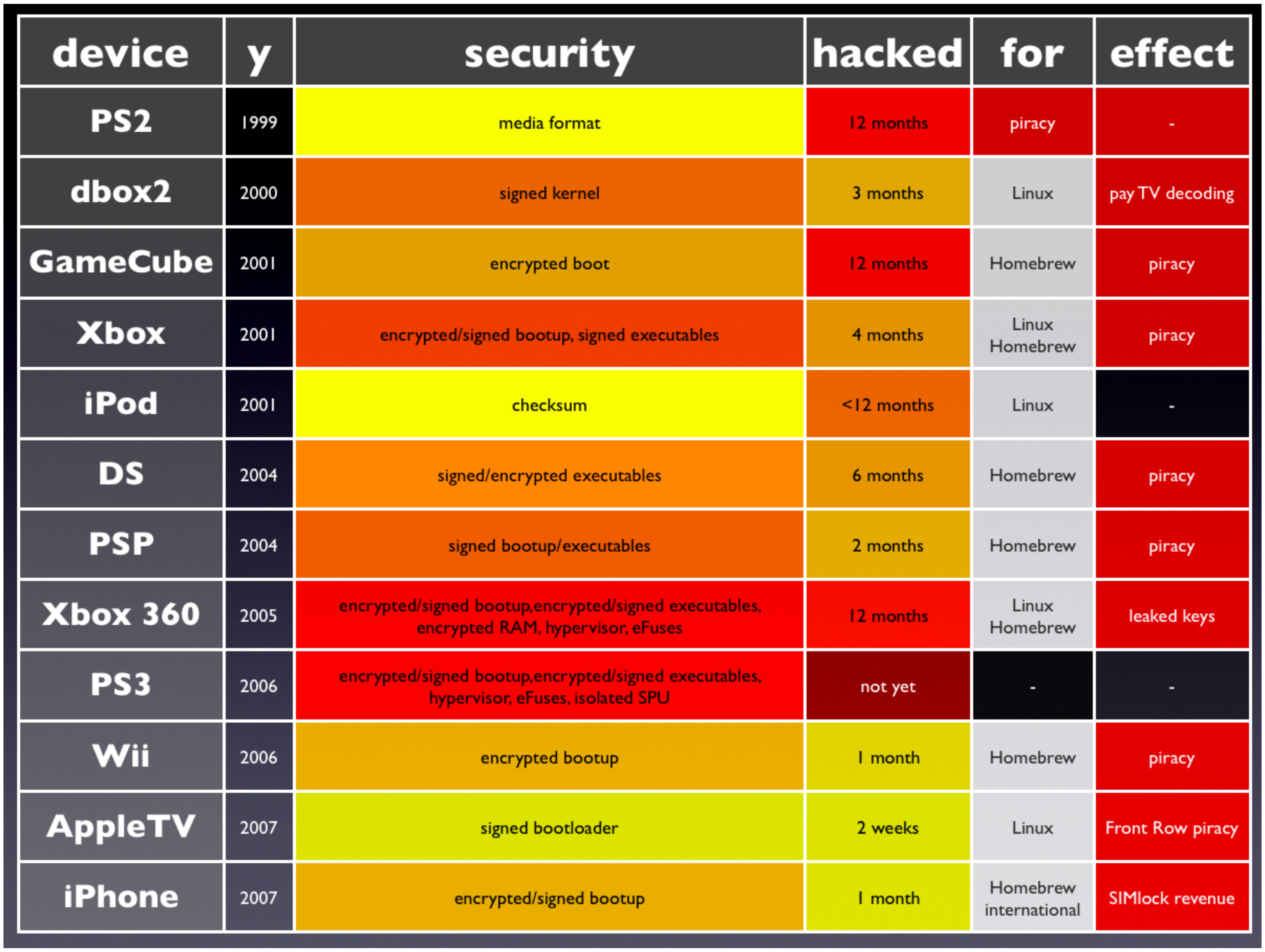
- Broadway code can reload IOS
- Broadway code can call private IOS functions
    - Read/write encrypted flash at low level
    - Identify using TMD/eTicket
- Poor parameter verification in syscalls
- Poor caller process checks in syscalls
- Latent DVD-mode code

Procedural problems

- Long testing cycles

- Unwillingness to talk to security researchers
- Left boot1 unpatched for a year
- "knee-jerk" bugfixes (fixed irrelevant holes without improving architecture)
- Two different teams working on software -- poor communication?

Embedded Device Scorecard



device	y	security	hacked	for	effect
PS2	1999	media format	12 months	piracy	-
dbox2	2000	signed kernel	3 months	Linux	pay TV decoding
GameCube	2001	encrypted boot	12 months	Homebrew	piracy
Xbox	2001	encrypted/signed bootup, signed executables	4 months	Linux Homebrew	piracy
iPod	2001	checksum	<12 months	Linux	-
DS	2004	signed/encrypted executables	6 months	Homebrew	piracy
PSP	2004	signed bootup/executables	2 months	Homebrew	piracy
Xbox 360	2005	encrypted/signed bootup,encrypted/signed executables, encrypted RAM, hypervisor, eFuses	12 months	Linux Homebrew	leaked keys
PS3	2006	encrypted/signed bootup,encrypted/signed executables, hypervisor, eFuses, isolated SPU	not yet	-	-
Wii	2006	encrypted bootup	1 month	Homebrew	piracy
AppleTV	2007	signed bootloader	2 weeks	Linux	Front Row piracy
iPhone	2007	encrypted/signed bootup	1 month	Homebrew international	SIMlock revenue

Homebrew demos

- Homebrew Channel
- BootMii