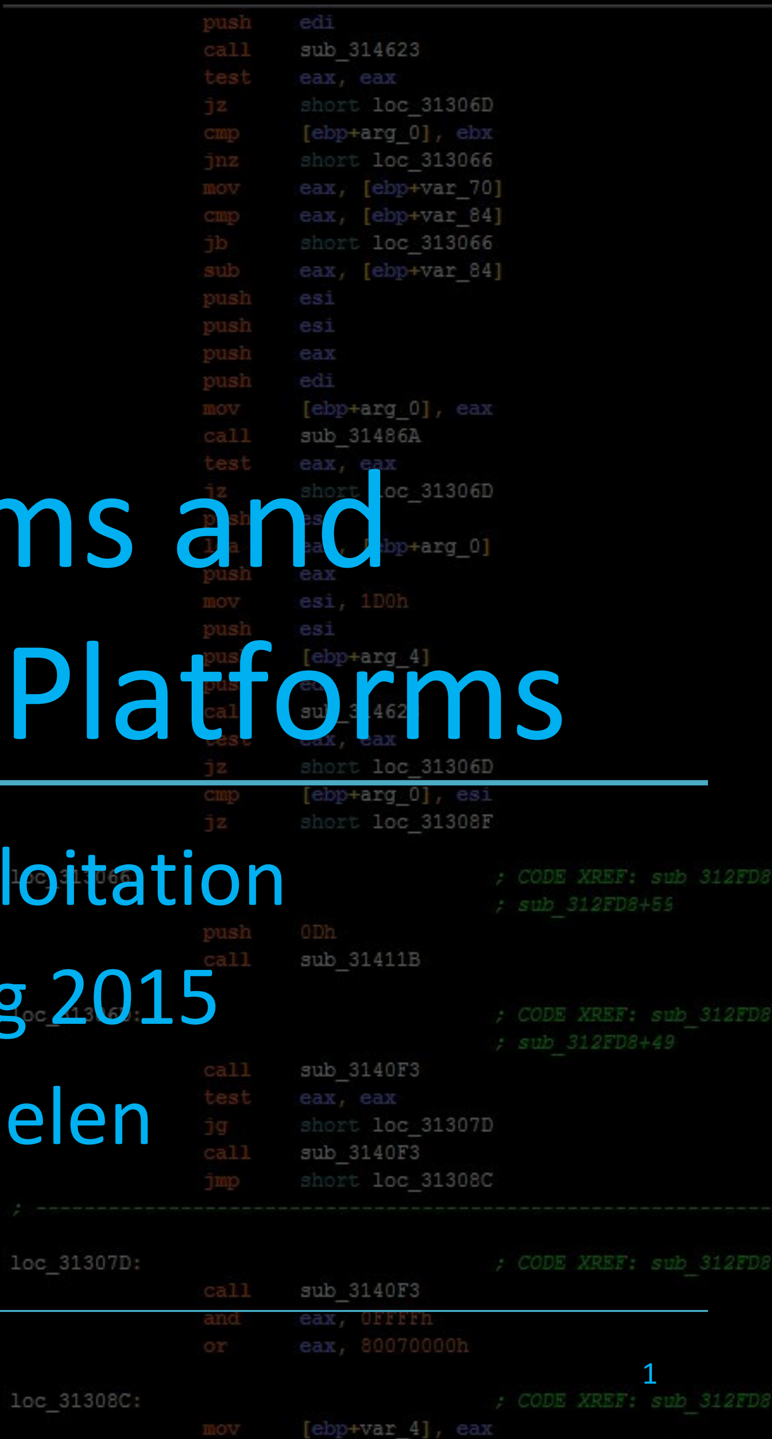
Secure Systems and Pwning Popular Platforms

Modern Binary Exploitation

CSCI 4968 - Spring 2015

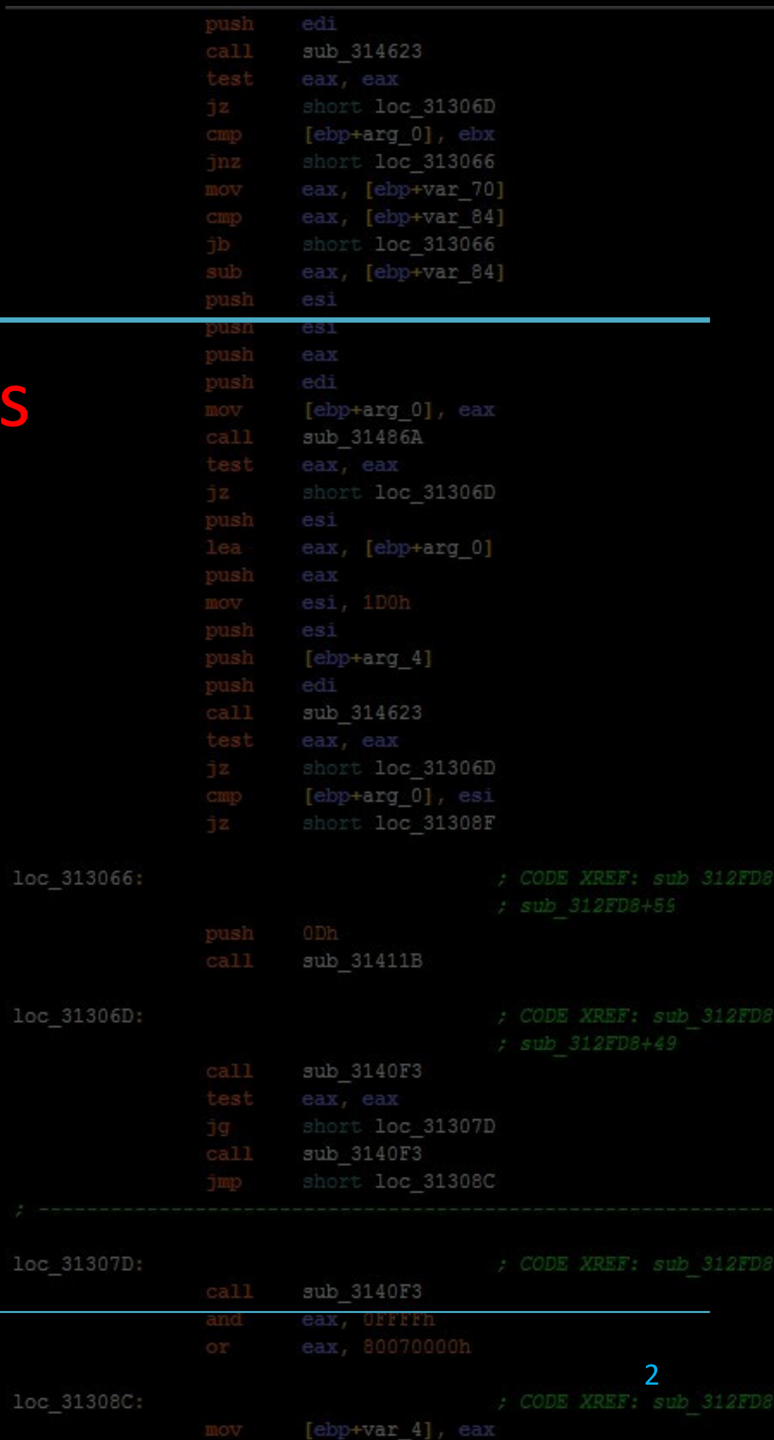
Markus Gaasedelen



```
          push    edi
          call    sub_314623
          test    eax, eax
          jz      short loc_31306D
          cmp     [ebp+arg_0], ebx
          jnz     short loc_313066
          mov     eax, [ebp+var_70]
          cmp     eax, [ebp+var_84]
          jb      short loc_313066
          sub     eax, [ebp+var_84]
          push    esi
          push    esi
          push    eax
          push    edi
          mov     [ebp+arg_0], eax
          call    sub_31486A
          test    eax, eax
          jz      short loc_31306D
          push    esi
          lea     eax, [ebp+arg_0]
          push    eax
          mov     esi, 1D0h
          push    esi
          push    [ebp+arg_4]
          push    edi
          call    sub_314623
          test    eax, eax
          jz      short loc_31306D
          cmp     [ebp+arg_0], esi
          jz      short loc_31308F

loc_313066:                          ; CODE XREF: sub_312FD8
                                     ; sub_312FD8+55
          push    0Dh
          call    sub_31411B

loc_31306D:                          ; CODE XREF: sub_312FD8
                                     ; sub_312FD8+49
          call    sub_3140F3
          test    eax, eax
          jg      short loc_31307D
          call    sub_3140F3
          jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                          ; CODE XREF: sub_312FD8
          call    sub_3140F3
          and     eax, 0FFFFh
          or      eax, 80070000h

loc_31308C:                          ; CODE XREF: sub_312FD8
          mov     [ebp+var_4], eax
```

Lecture Overview

- Secure Systems & Patch Sets
  - OpenBSD
  - SELinux
  - Grsecurity
- Owning Game Consoles
  - Xbox 360
  - Nintendo 3DS
  - PS3
- Current Generation



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

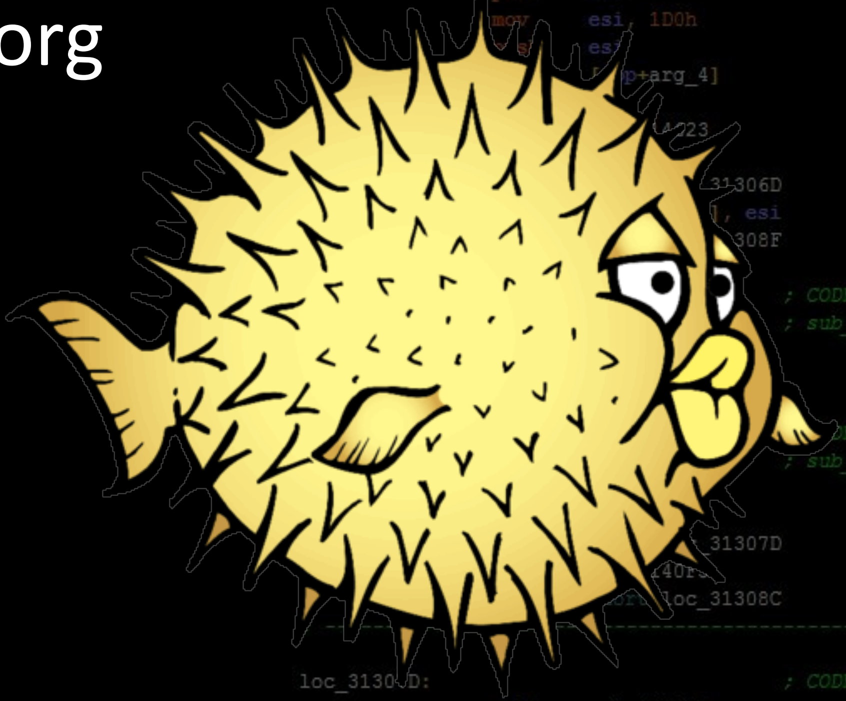
OpenBSD

“The OpenBSD project produces a **FREE**, multi-platform 4.4BSD-based UNIX-like operating system. Our efforts emphasize portability, standardization, correctness, proactive security and integrated cryptography.”

–openbsd.org

OpenBSD

- Started in 1995 forking from NetBSD 1.0
- “Try to be the #1 most secure operating system” –openbsd.org



OpenBSD – Added Technologies

- Adds hardening & security technologies
  - W^X
  - Privilege isolation
  - Jails
  - Randomized malloc/mmap
  - Ships Crypto
  - A few other things

OpenBSD – "secure by default"

- Good code is inherently secure
  - Fewer bugs
  - Utilizes secure coding practices
- Extensive code review and audits
- Reduces attack surface by disabling most remote services in default install

FEWER BUGS != MORE SECURE

One bug is still enough to blow things wide open

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Security-Enhanced Linux (SELinux)

"SELinux is an implementation of mandatory access controls (MAC) on Linux. Mandatory access controls allow an administrator of a system to define how applications and users can access different resources such as files, devices, networks and inter-process communication."

–selinuxproject.org

SELinux – Overview

- Open sourced by the NSA in 2000
- Extended filesystem permissions controls
  - Users and services should only have access to exactly what they need

Grsecurity (GRSEC)

"Grsecurity is an extensive security enhancement to the Linux kernel that defends against a wide range of security threats through intelligent access control, memory corruption-based exploit prevention, and a host of other system hardening that generally require no configuration ..."

–grsecurity.net

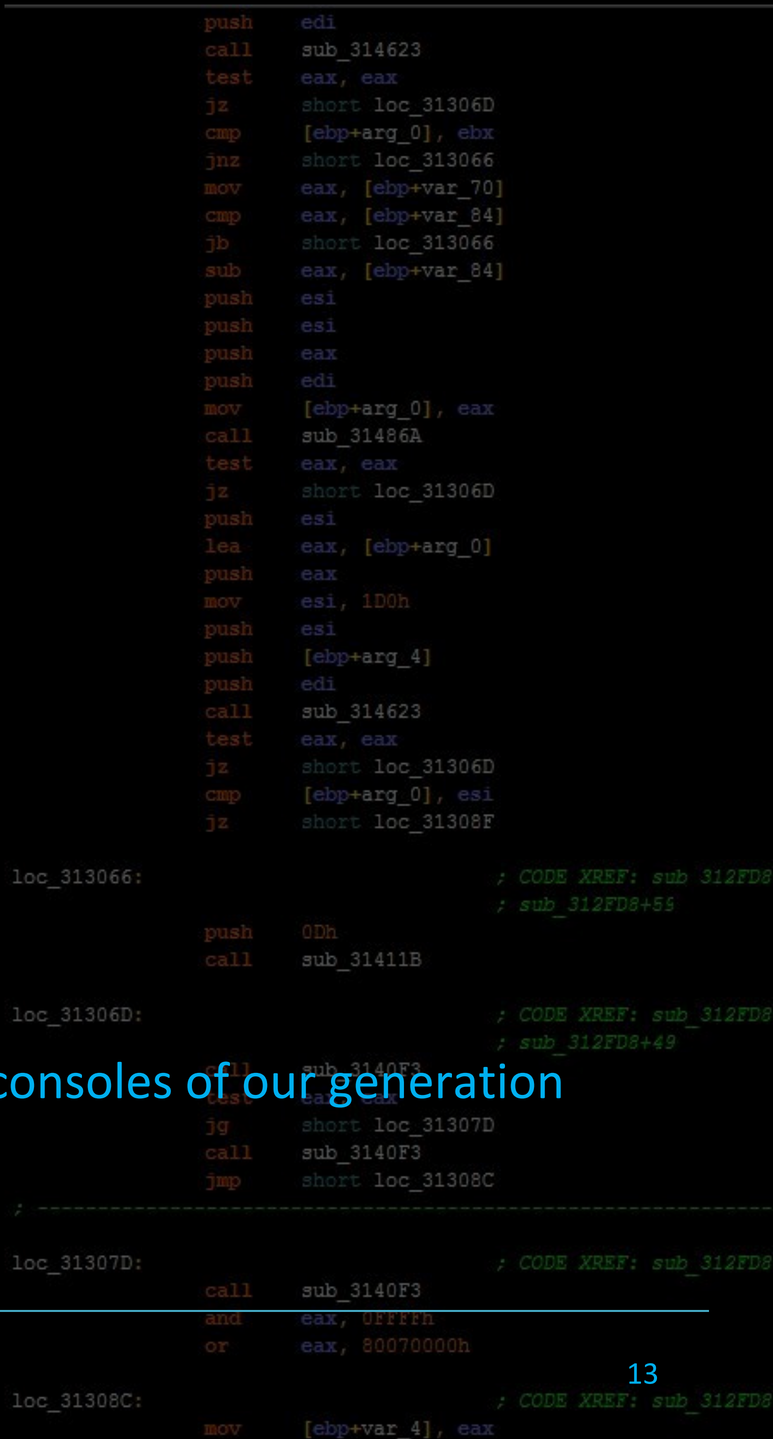
GRSEC – Overview

- Started in 2001 as a port of OpenWall
- Free, relatively easy to setup
- Besides robust access control like SELinux, GRSEC has a large focus on hardening against memory corruption based exploits
  - High quality PAX ASLR, Memory Sanitization, Heap Hardening, Active Response, to name a few

Lecture Overview

- Secure Systems & Patch Sets
  - OpenBSD
  - SELinux
  - Grsecurity
- Owning Game Consoles
  - Xbox 360
  - Nintendo 3DS
  - PS3
- Current Generation

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```



```
                  push    edi
                  call    sub_314623
                  test    eax, eax
                  jz      short loc_31306D
                  cmp     [ebp+arg_0], ebx
                  jnz     short loc_313066
                  mov     eax, [ebp+var_70]
                  cmp     eax, [ebp+var_84]
                  jb      short loc_313066
                  sub     eax, [ebp+var_84]
                  push    esi
                  push    esi
                  push    eax
                  push    edi
                  mov     [ebp+arg_0], eax
                  call    sub_31486A
                  test    eax, eax
                  jz      short loc_31306D
                  push    esi
                  lea     eax, [ebp+arg_0]
                  push    eax
                  mov     esi, 1D0h
                  push    esi
                  push    [ebp+arg_4]
                  push    edi
                  call    sub_314623
                  test    eax, eax
                  jz      short loc_31306D
                  cmp     [ebp+arg_0], esi
                  jz      short loc_31308F

loc_313066:                               ; CODE XREF: sub_312FD8
                                          ; sub_312FD8+55
                  push    0Dh
                  call    sub_31411B

loc_31306D:                               ; CODE XREF: sub_312FD8
                                          ; sub_312FD8+49
                  call    sub_3140F3
                  test    eax, eax
                  jg      short loc_31307D
                  call    sub_3140F3
                  jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                               ; CODE XREF: sub_312FD8
                  call    sub_3140F3
                  and     eax, 0FFFFh
                  or      eax, 80070000h

loc_31308C:                               ; CODE XREF: sub_312FD8
                  mov     [ebp+var_4], eax
```

GAME CONSOLES

A closer look at the bugs that brought down consoles of our generation

Game Consoles

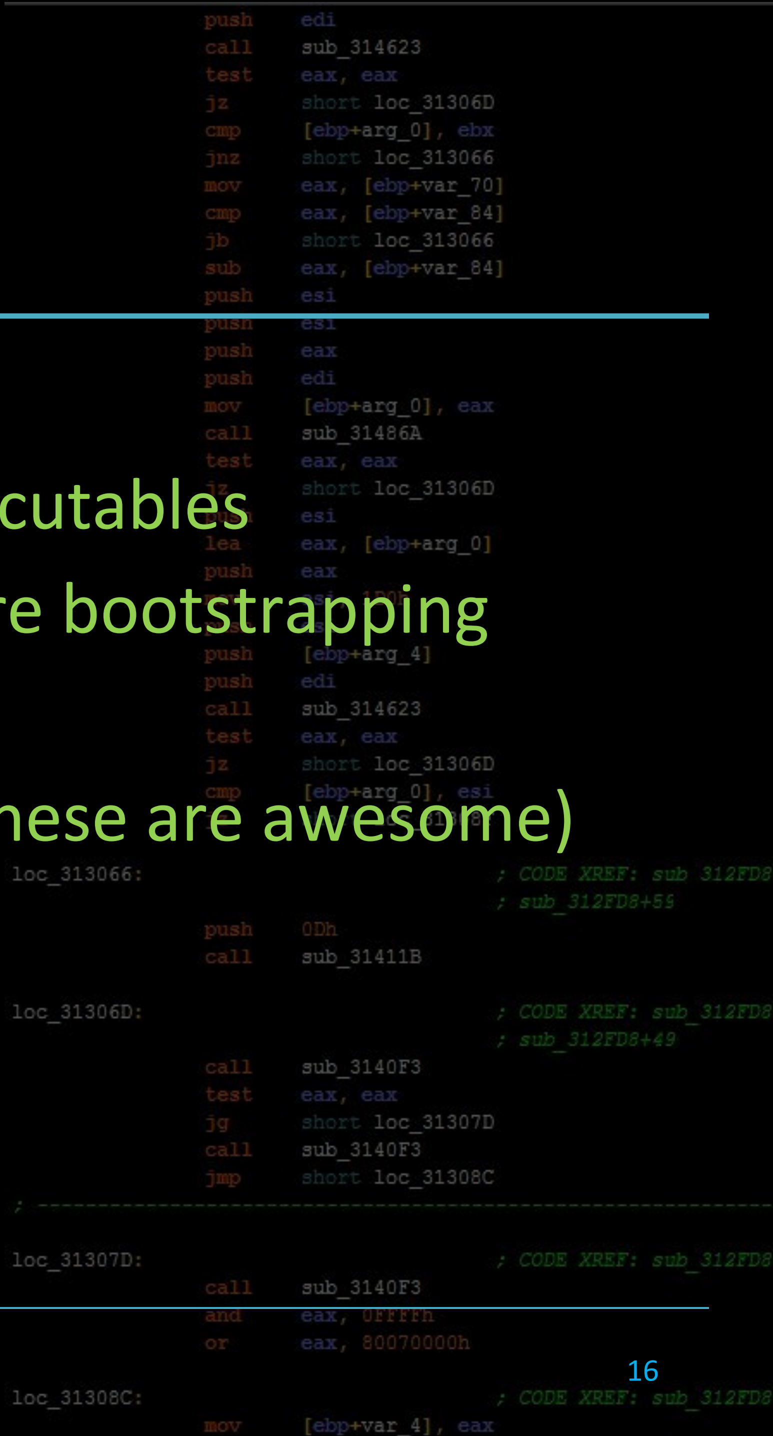
- Evolving entertainment platforms
  - Play games, stream media, browse the web
- 100% consistent machine for developers
  - Don’t have to account for different specs (eg. PC’s)
- Enforces DRM much better than PC’s can
  - It’s a controlled platform that only runs code as blessed by Sony, Microsoft, Nintendo

Xbox 360 – Nov. 2005

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Xbox 360 – Nov. 2005

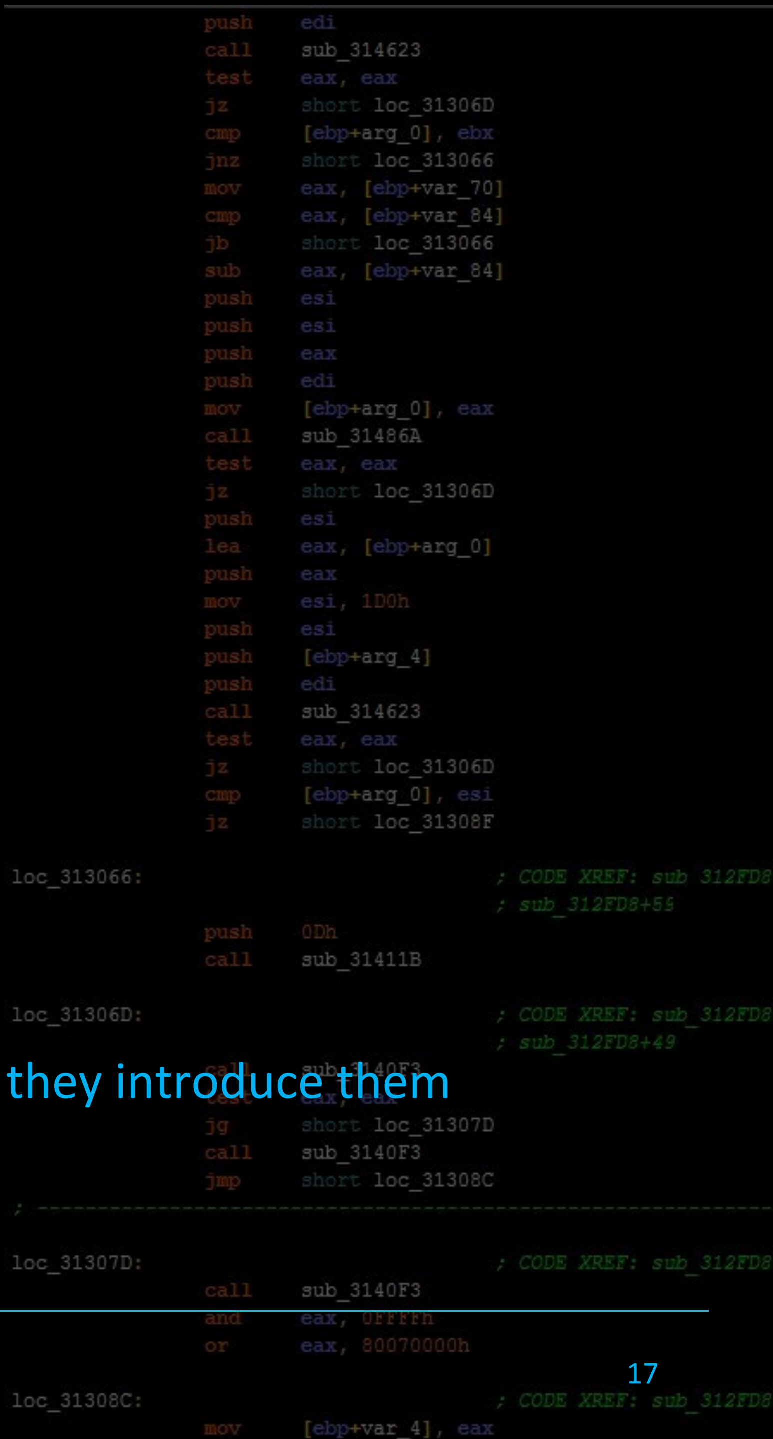
- Security Perspective
  - Only runs signed code or executables
  - Rigorous chain of trust, secure bootstrapping
  - Encrypted runtime memory
  - eFuses to enforce updates (these are awesome)
  - NX/DEP
  - No ASLR



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax

                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

KING KONG EXPLOIT

updates don’t always patch bugs, sometimes they introduce them



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

King Kong Exploit – Dec. 2006

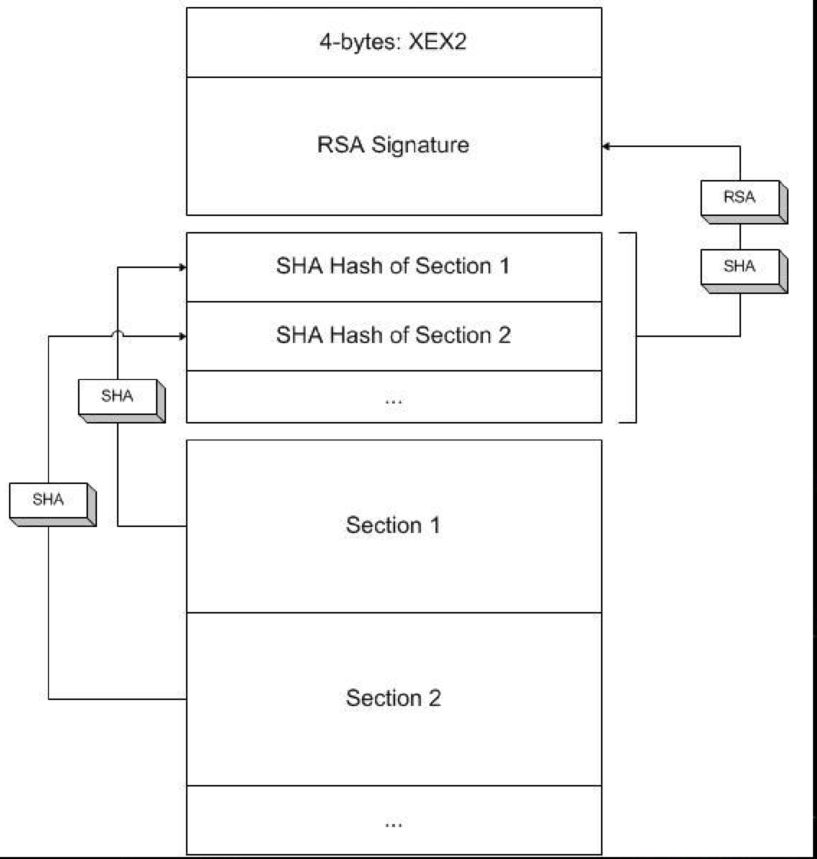
- Integer based bug, resulting in code execution at the Hypervisor context
  - Complete system control

- The bug leveraged by the King Kong Exploit was INTRODUCED in kernel version 4532, and patched two updates later in v4552
  - For reference, the Xbox 360 shipped on v1888

About the Xbox 360 & Games

- All executables (.XEX's) are signed by Microsoft which the system verifies to prevent tampering with code
- Data assets such as textures, models, shaders, and audio as used by games are NOT signed!
  - Find bugs in game asset parsers

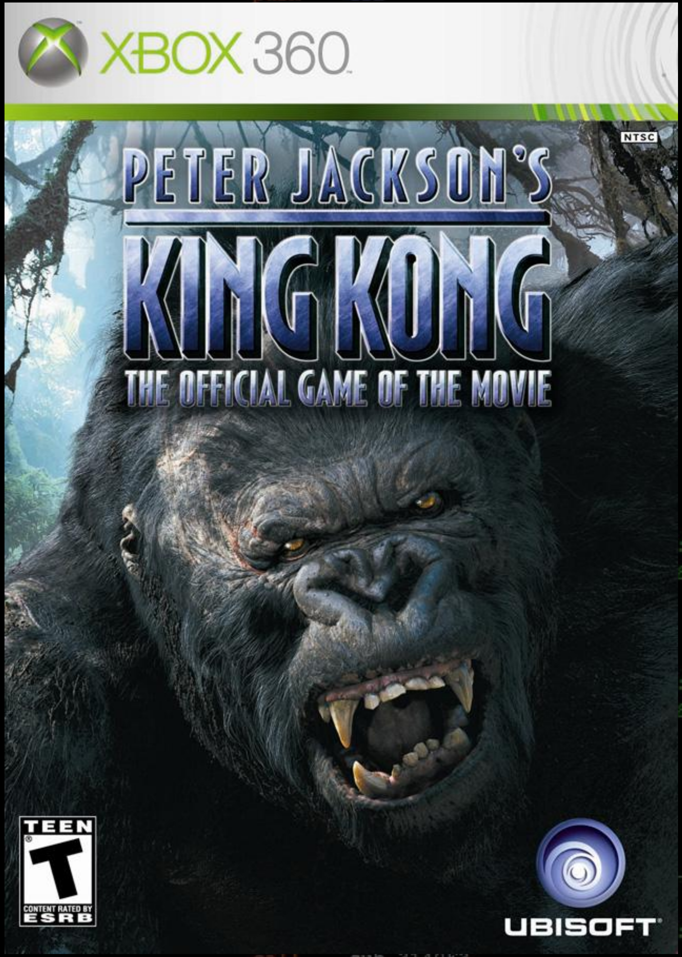
Figure 1. The Cryptographic Anatomy of a XEX File



Stage One: King Kong's Role

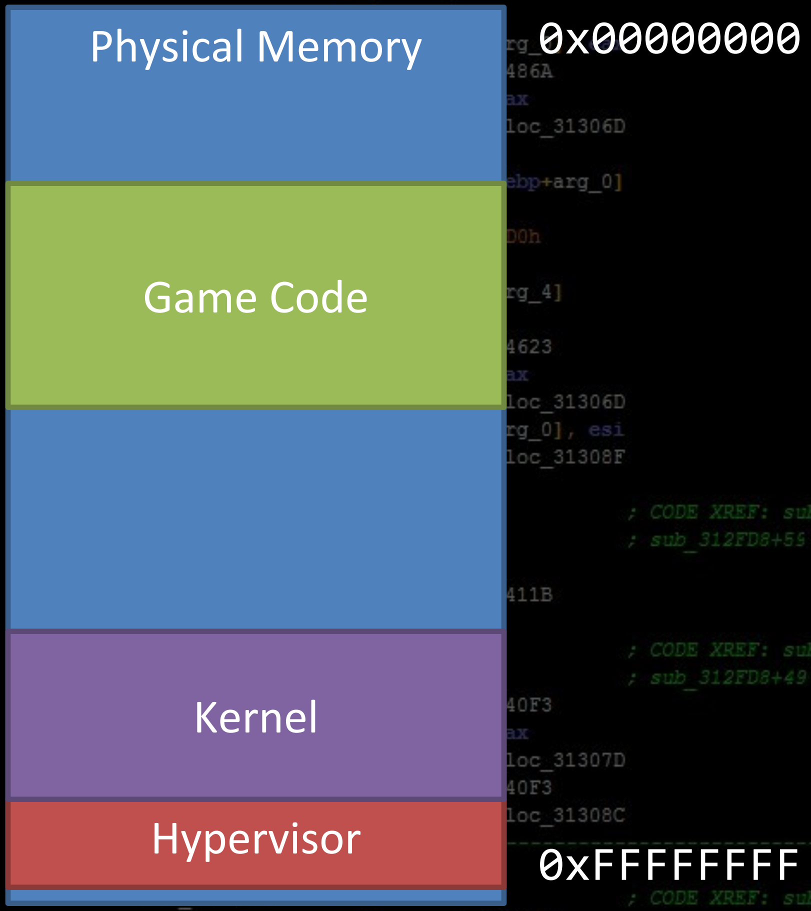
- A maliciously crafted unsigned shader file parsed by the signed King Kong game XEX, can lead to unprivileged code execution on the system

- King Kong was one of many possible memory corruption vectors that could have been used to get basic code exec



About the Xbox 360 Hypervisor

- A small Hypervisor (Hv) sits next to the kernel, near the top of memory
- The Hv handles some crypto keys, low level IO, memory encryption/decryption operations and more
- If you can take over the Hv, you have access to physmem and the highest privilege of execution

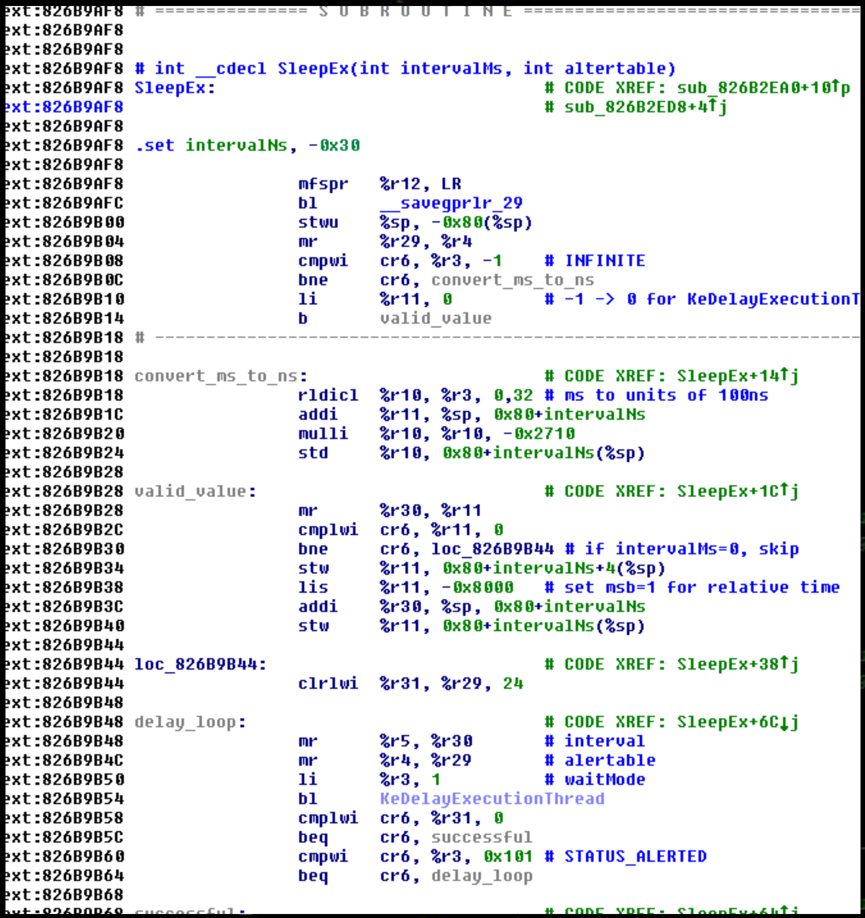


Stage Two: Hyper Escalation

- The PPC instruction ‘sc’ is used to make system calls on the Xbox 360, the Hv handles these calls as they are made

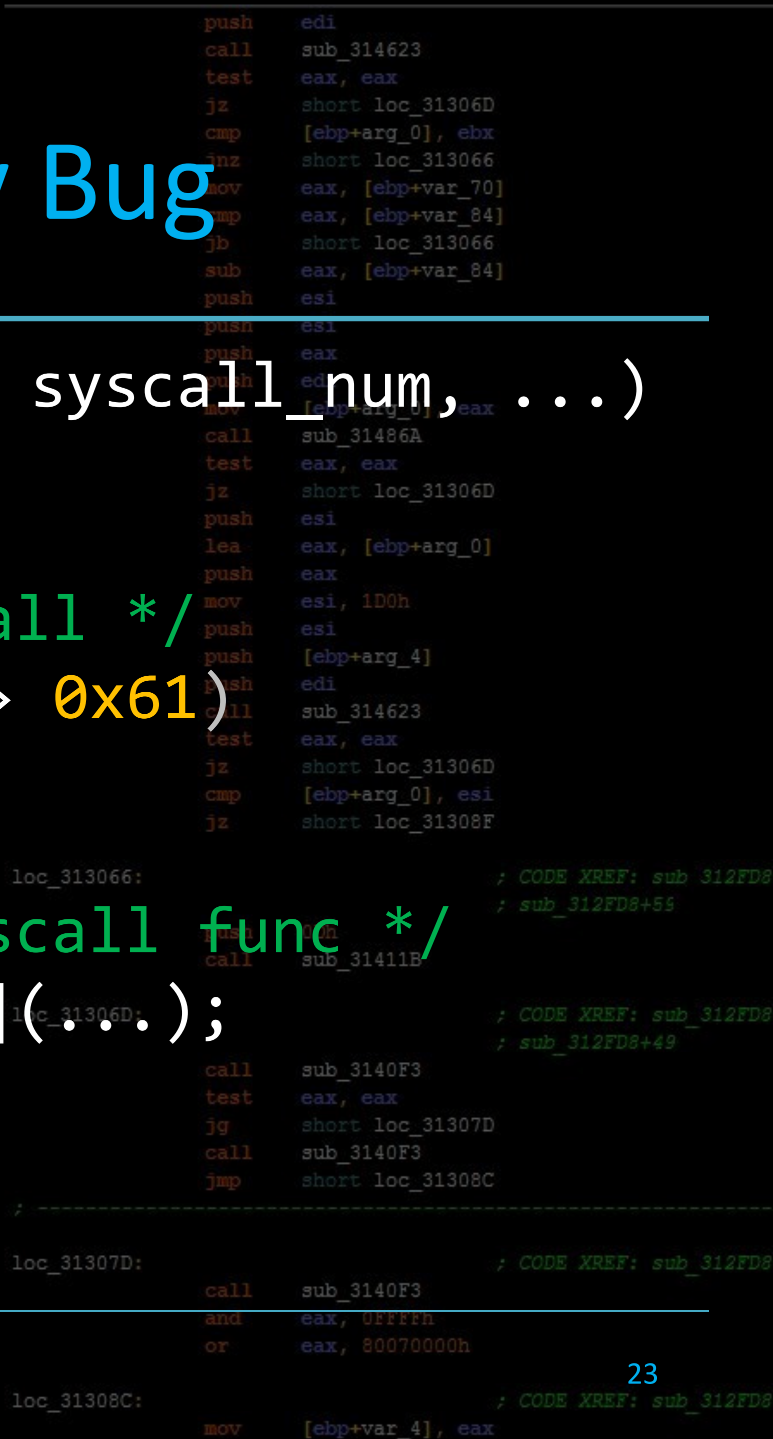
- Unfortunately, along came a bug in the syscall handler):

  random ppc ------------->



```
ext:826B9AF8 # =============== S U B R O U T I N E ===================================
ext:826B9AF8
ext:826B9AF8
ext:826B9AF8 # int __cdecl SleepEx(int intervalMs, int altertable)
ext:826B9AF8 SleepEx:                                 # CODE XREF: sub_826B2EA0+10↑p
ext:826B9AF8                                          # sub_826B2ED8+4↑j
ext:826B9AF8
ext:826B9AF8 .set intervalNs, -0x30
ext:826B9AF8
ext:826B9AF8                 mfspr   %r12, LR
ext:826B9AFC                 bl      __savegprlr_29
ext:826B9B00                 stwu    %sp, -0x80(%sp)
ext:826B9B04                 mr      %r29, %r4
ext:826B9B08                 cmpwi   cr6, %r3, -1    # INFINITE
ext:826B9B0C                 bne     cr6, convert_ms_to_ns
ext:826B9B10                 li      %r11, 0         # -1 -> 0 for KeDelayExecutionT
ext:826B9B14                 b       valid_value
ext:826B9B18 # ---------------------------------------------------------------------
ext:826B9B18
ext:826B9B18 convert_ms_to_ns:                        # CODE XREF: SleepEx+14↑j
ext:826B9B18                 rldicl  %r10, %r3, 0,32 # ms to units of 100ns
ext:826B9B1C                 addi    %r11, %sp, 0x80+intervalNs
ext:826B9B20                 mulli   %r10, %r10, -0x2710
ext:826B9B24                 std     %r10, 0x80+intervalNs(%sp)
ext:826B9B28
ext:826B9B28 valid_value:                             # CODE XREF: SleepEx+1C↑j
ext:826B9B28                 mr      %r30, %r11
ext:826B9B2C                 cmplwi  cr6, %r11, 0
ext:826B9B30                 bne     cr6, loc_826B9B44 # if intervalMs=0, skip
ext:826B9B34                 stw     %r11, 0x80+intervalNs+4(%sp)
ext:826B9B38                 lis     %r11, -0x8000   # set msb=1 for relative time
ext:826B9B3C                 addi    %r30, %sp, 0x80+intervalNs
ext:826B9B40                 stw     %r11, 0x80+intervalNs(%sp)
ext:826B9B44
ext:826B9B44 loc_826B9B44:                            # CODE XREF: SleepEx+38↑j
ext:826B9B44                 clrlwi  %r31, %r29, 24
ext:826B9B48
ext:826B9B48 delay_loop:                              # CODE XREF: SleepEx+6C↓j
ext:826B9B48                 mr      %r5, %r30       # interval
ext:826B9B4C                 mr      %r4, %r29       # alertable
ext:826B9B50                 li      %r3, 1          # waitMode
ext:826B9B54                 bl      KeDelayExecutionThread
ext:826B9B58                 cmplwi  cr6, %r31, 0
ext:826B9B5C                 beq     cr6, successful
ext:826B9B60                 cmpwi   cr6, %r3, 0x101 # STATUS_ALERTED
ext:826B9B64                 beq     cr6, delay_loop
ext:826B9B68
```

Pseudocode of the Hv Bug



```
int syscall_handler(uint64_t syscall_num, ...)
{

   /* check for invalid syscall */
   if((uint32_t)syscall_num > 0x61)
      return 0;

   /* call the respective syscall func */
   syscall_table[syscall_num](...);
   ...
```

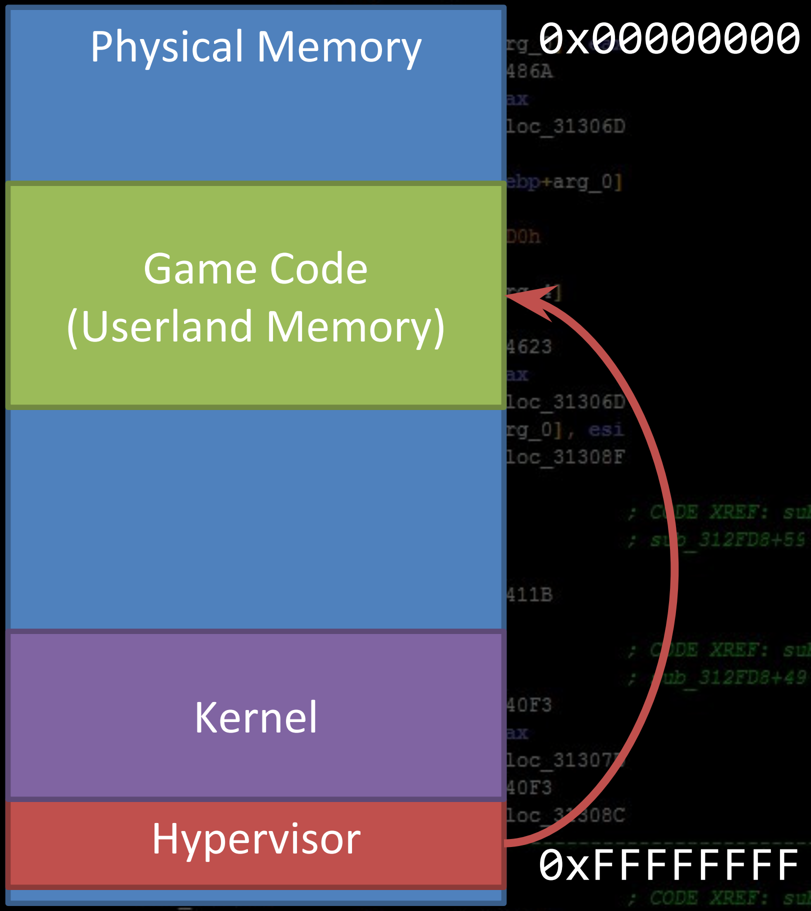
The Oops

- Only the lower 32 bits of the syscall number are sanity checked

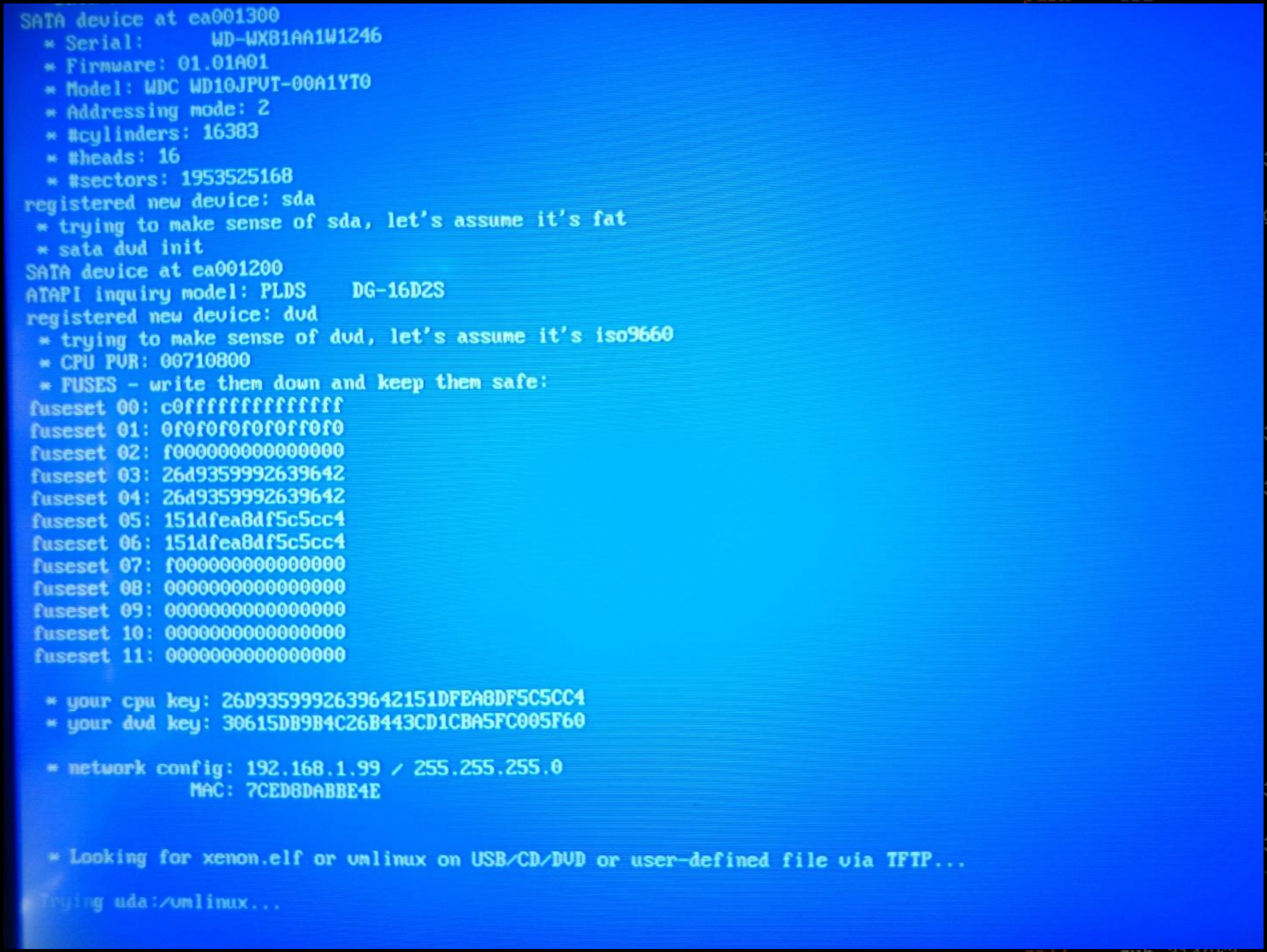
- The whole 64 bit number is used in address calculation

```
syscall_table[syscall_num](...);
```

Arbitrary jump into userland memory/code at the HV Context



Game Over



```
SATA device at ea001300
  * Serial:       WD-WXB1AA1W1246
  * Firmware: 01.01A01
  * Model: WDC WD10JPVT-00A1YT0
  * Addressing mode: 2
  * #cylinders: 16383
  * #heads: 16
  * #sectors: 1953525168
registered new device: sda
 * trying to make sense of sda, let's assume it's fat
 * sata dvd init
SATA device at ea001200
ATAPI inquiry model: PLDS    DG-16D2S
registered new device: dvd
 * trying to make sense of dvd, let's assume it's iso9660
 * CPU PVR: 00710800
 * FUSES - write them down and keep them safe:
fuseset 00: c0ffffffffffffff
fuseset 01: 0f0f0f0f0f0ff0f0
fuseset 02: f000000000000000
fuseset 03: 26d9359992639642
fuseset 04: 26d9359992639642
fuseset 05: 151dfea8df5c5cc4
fuseset 06: 151dfea8df5c5cc4
fuseset 07: f000000000000000
fuseset 08: 0000000000000000
fuseset 09: 0000000000000000
fuseset 10: 0000000000000000
fuseset 11: 0000000000000000

  * your cpu key: 26D9359992639642151DFEA8DF5C5CC4
  * your dvd key: 30615DB9B4C26B443CD1CBA5FC005F60

  * network config: 192.168.1.99 / 255.255.255.0
               MAC: 7CED8DABBE4E


  * Looking for xenon.elf or vmlinux on USB/CD/DVD or user-defined file via TFTP...

 Trying uda:/vmlinux...
```

XBOX 360 HARDWARE ATTACKS

Straying from binary exploitation, but still interesting

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_3141

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

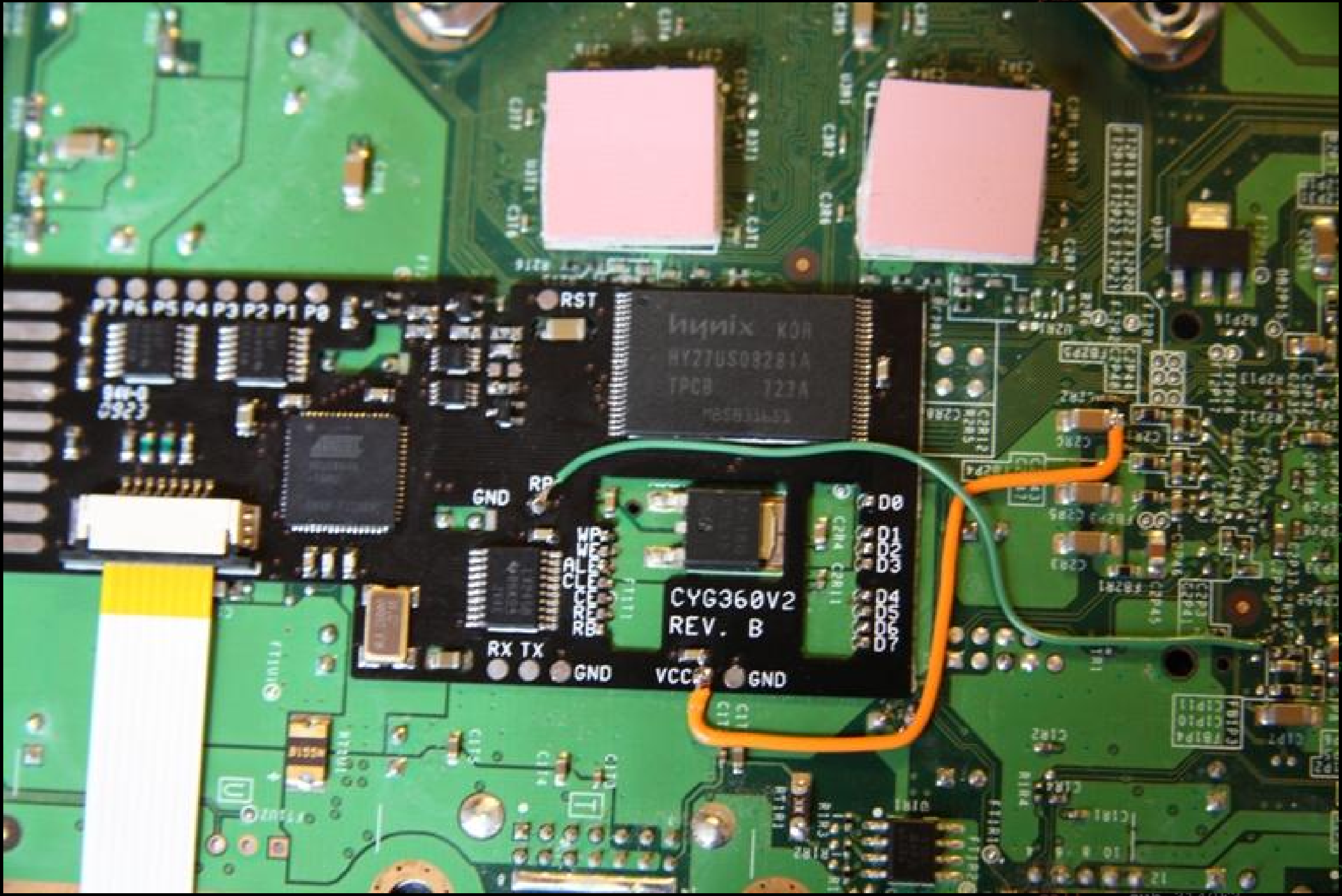
SMC / JTAG Hack – 2007-2009

- Uses the SMC and JTAG to trigger a DMA overwrite instantly at bootup rather than having to load a game such a King Kong

- Cat and mouse for a few years, allowing hackers to boot into downgraded, exploitable kernels (eg v4532)

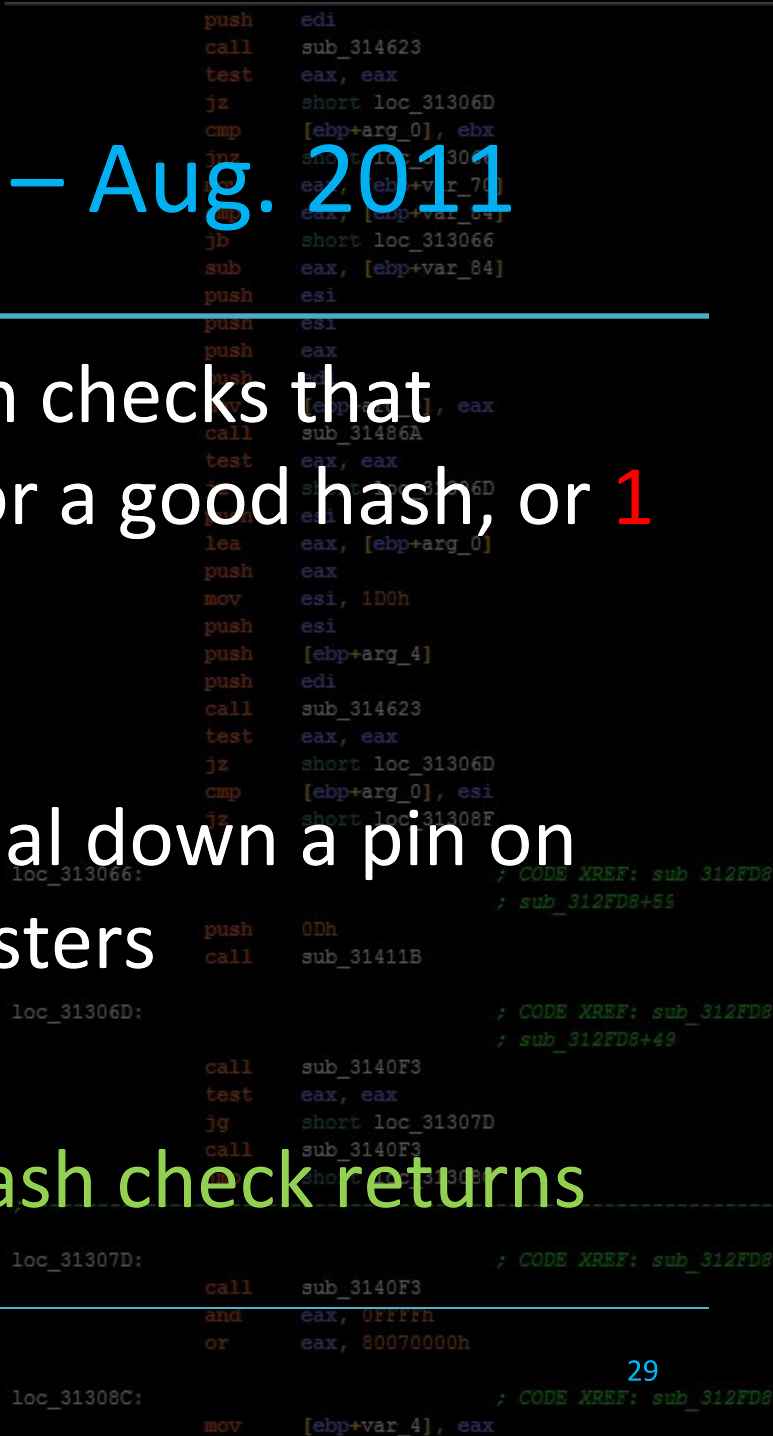
- Eventually Patched by MS when they decided to rework the boot process from the 2BL and up

SMC / JTAG Hack



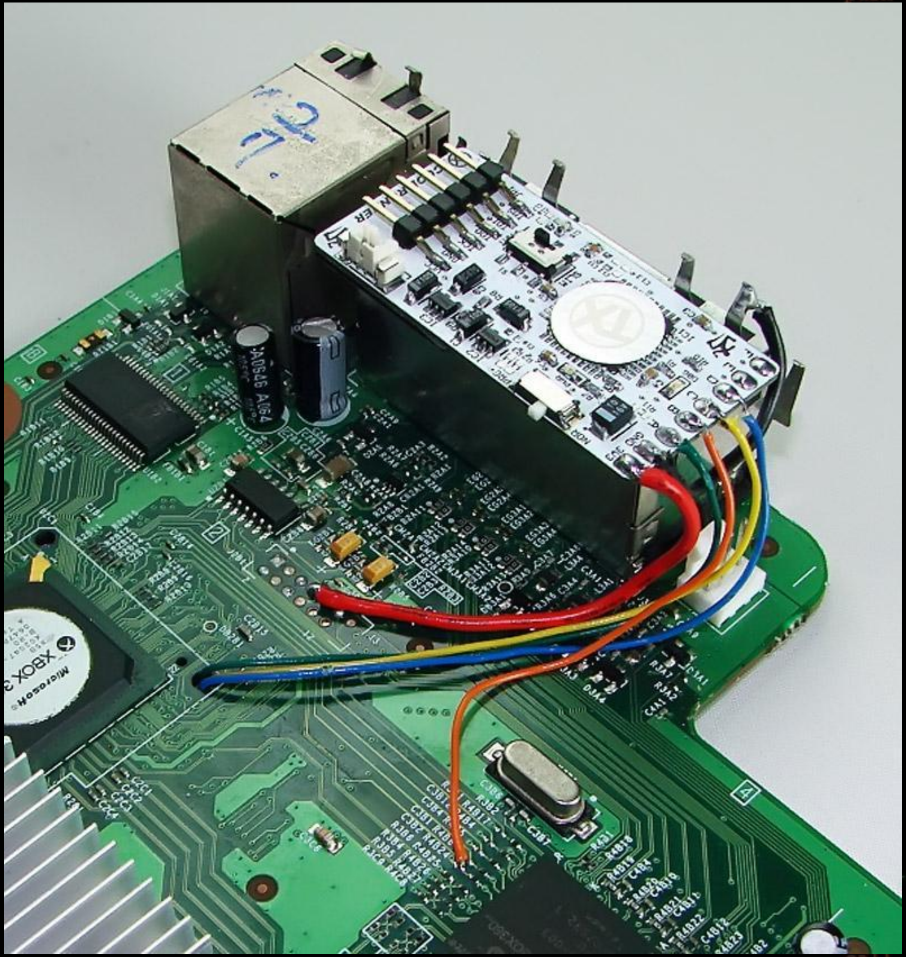
Reset Glitch Hack (RGH) – Aug. 2011

- In the 2bl there's some hash checks that expect a 0 to be returned for a good hash, or 1 for a hash mismatch (fail)
- Sending a specific reset signal down a pin on the CPU clears the CPU registers
- Reset the registers as the hash check returns

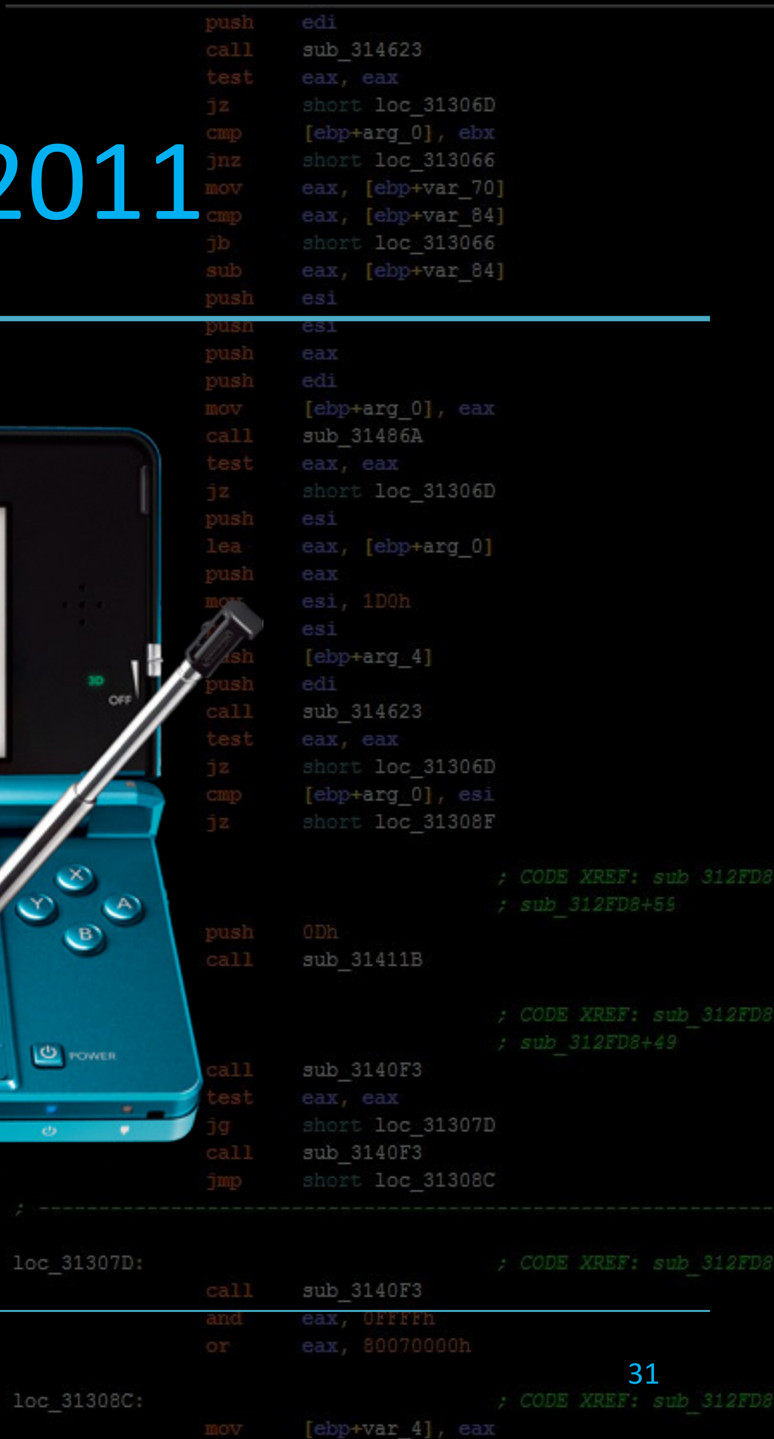


```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
[illegible]
[illegible]
[illegible]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
[illegible]
[illegible]
                call    sub_31486A
                test    eax, eax
[illegible]
[illegible]
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
[illegible]

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Xbox 360 Reset Glitch Hack (RGH)



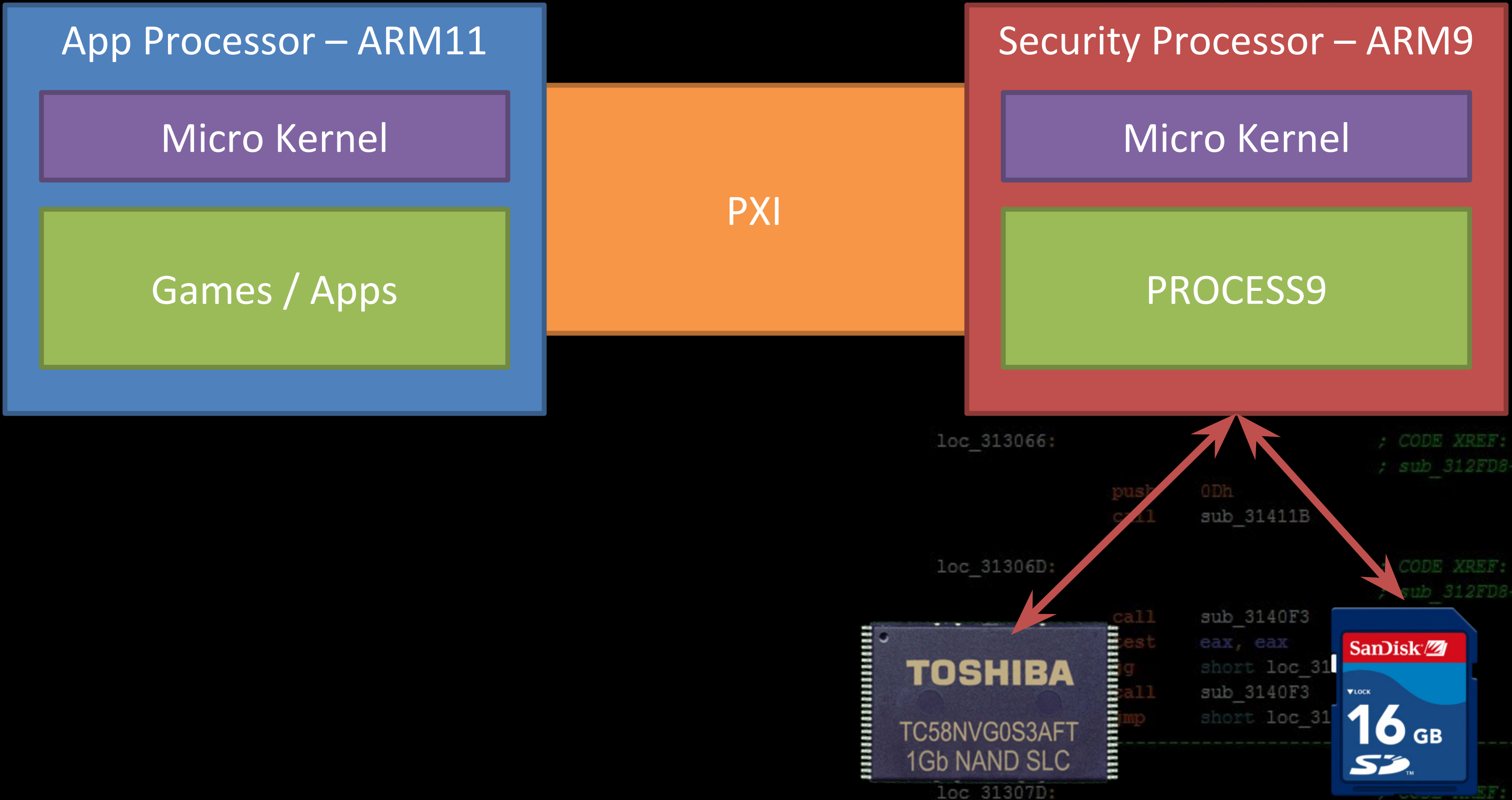
Nintendo 3DS – Feb. 2011


```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mo      esi, 1D0h
                        esi
                  sh    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

                                        ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

                                        ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Nintendo 3DS – Feb. 2011

- Security Perspective
  - Very tightly sealed bootrom, hardware disabled
  - Only runs signed code or executables
  - Hardware based keyscrambler for crypto keys
  - NX/DEP (Only used on the ARM11 Core)
  - Runtime memory is not encrypted
  - Has eFuses, not really used
  - No ASLR

Nintendo 3DS Architecture



Nintendo 3DS Architecture

- Application Processor (ARM11) – ‘high level’
  - Runs your games, apps, anything visual
- Security Processor (ARM9) – ‘low level’
  - Crypto, system IO, talks to hardware, like a Hv
- PXI
  - Pipeline for the cores to talk to each other

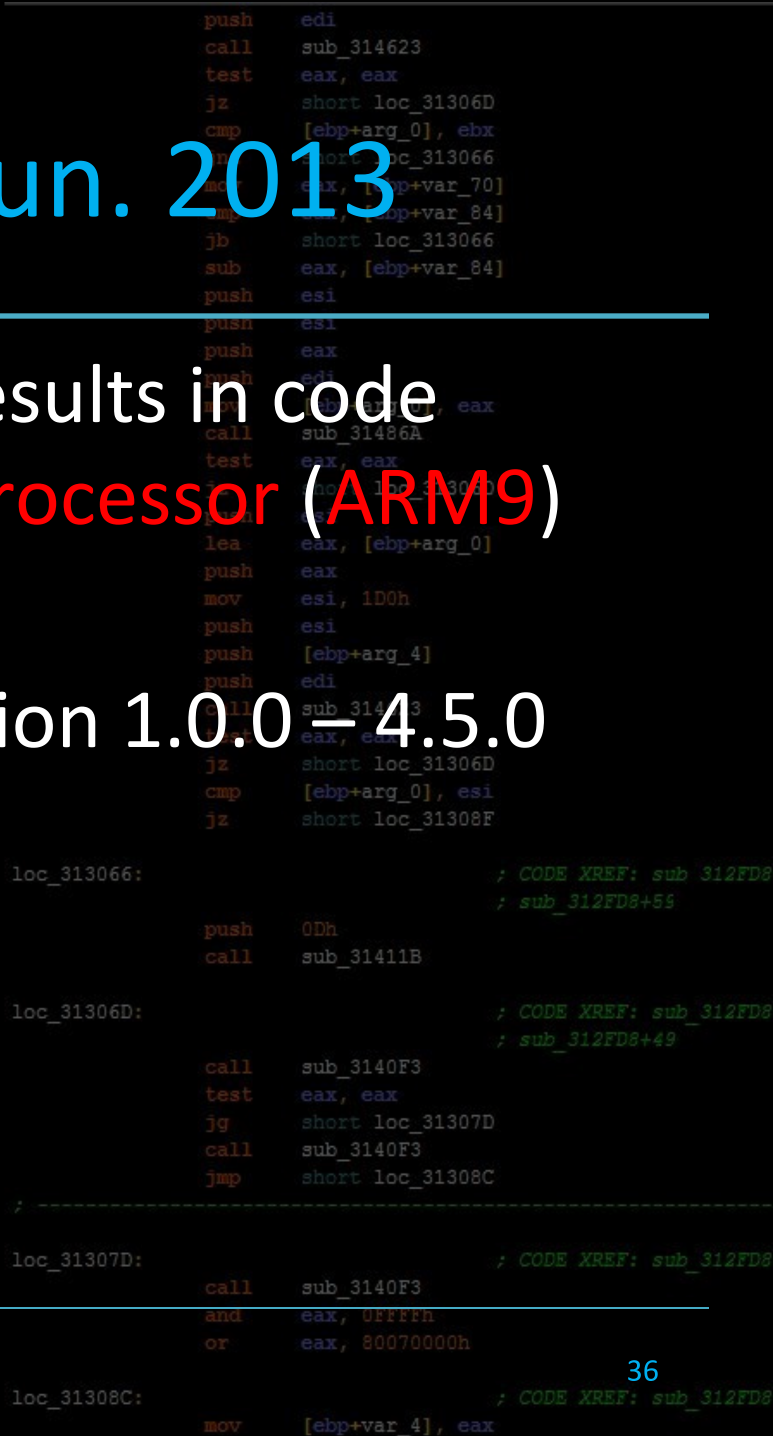
PWNING OVER THE PXI

Owning the SysCore through the PXI

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

VerifyRsaSha256() – Jun. 2013

- Straight stack smash bug, results in code execution on the Security Processor (ARM9)
  - Complete system control
- Present from firmware version 1.0.0 – 4.5.0
- Bug discovered in 2012

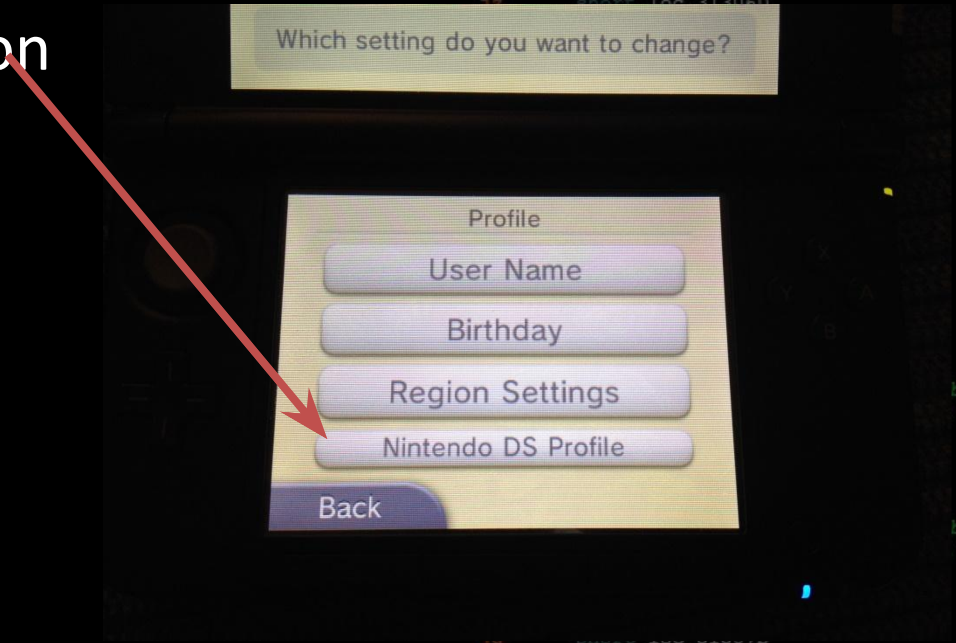


```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                [illegible] short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                [illegible]     [illegible], eax
                call    sub_31486A
                test    eax, eax
                [illegible]     [illegible] loc_31306D
                push    [illegible]
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314[illegible]3
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

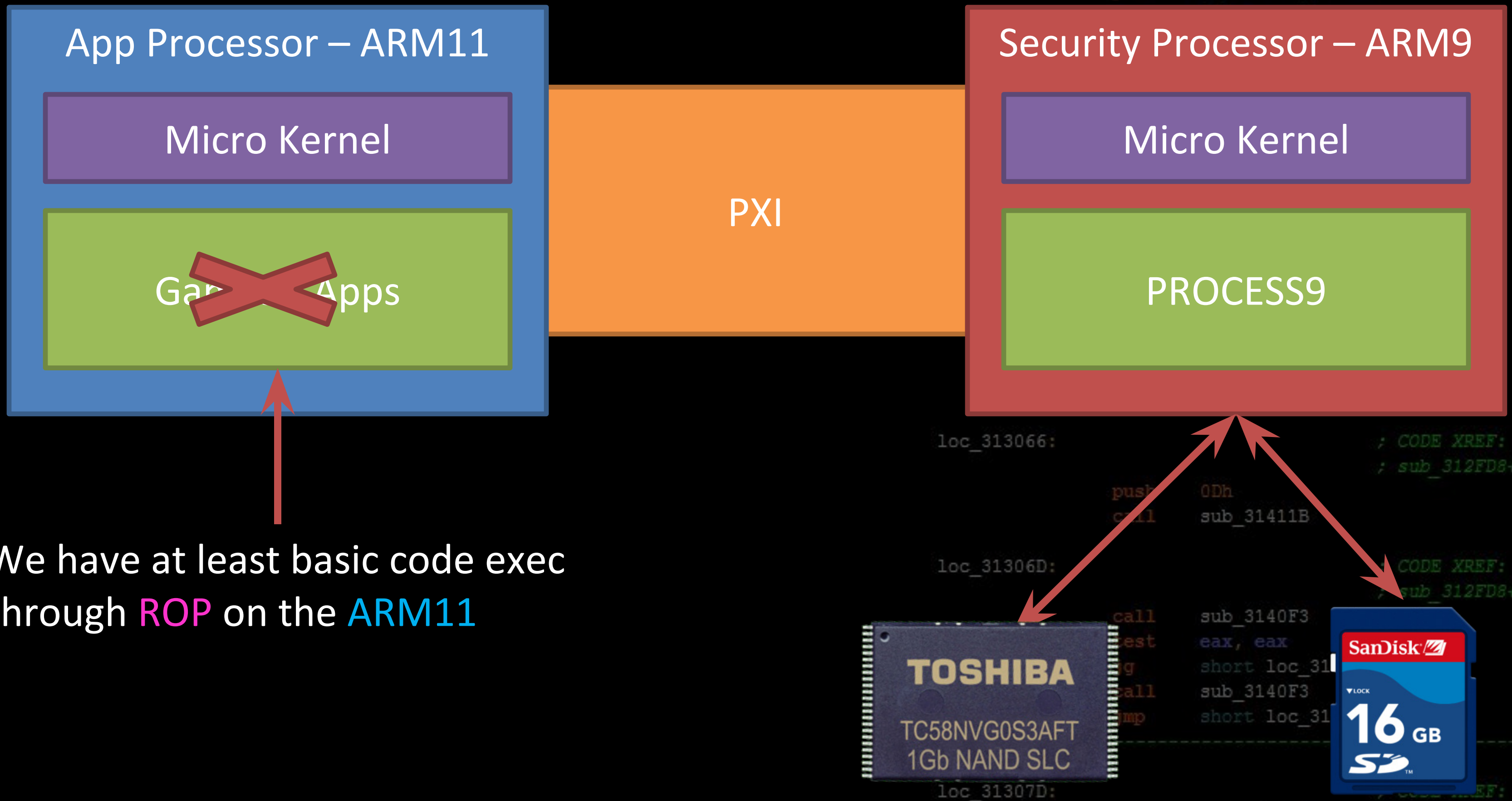
Stage One: ARM11 Code Exec

- A stack smash exists in the DS Profile fields in the native settings application on all 3DS's at the time. No need for any games!

- This is a straight stack smash that will get us control, but there is DEP on the ARM11 so you must ROP

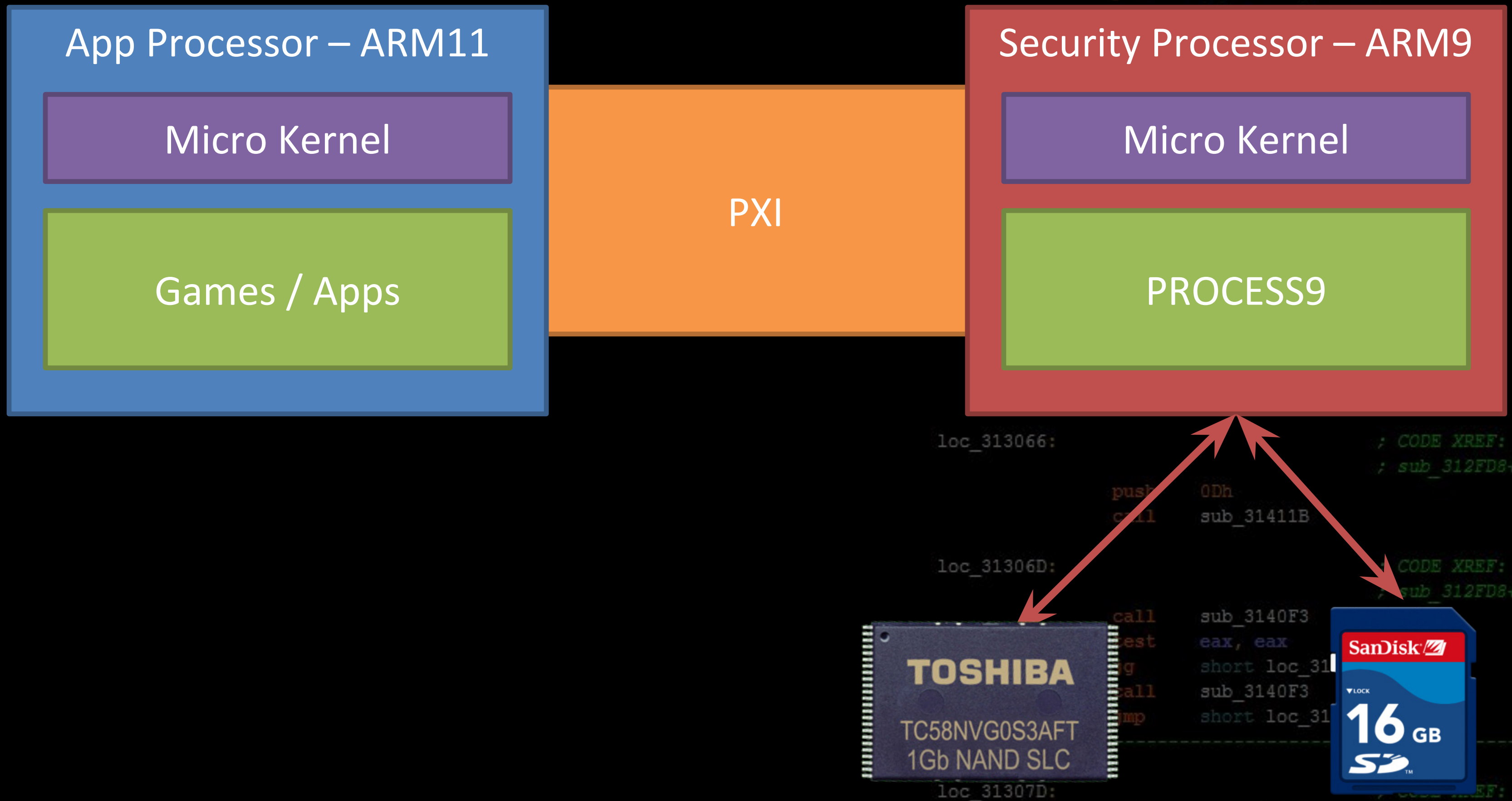


State of Control

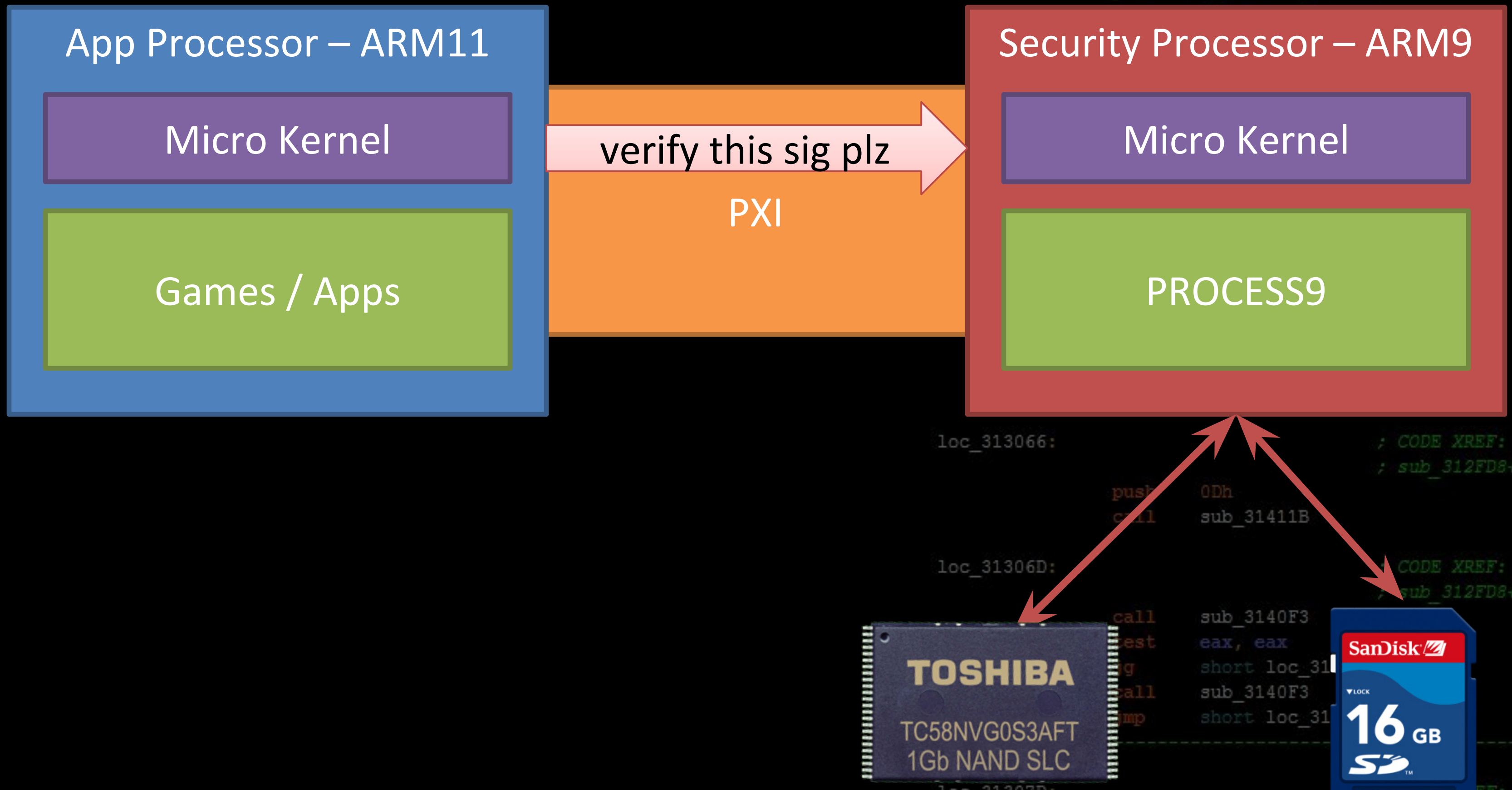


We have at least basic code exec through ROP on the ARM11

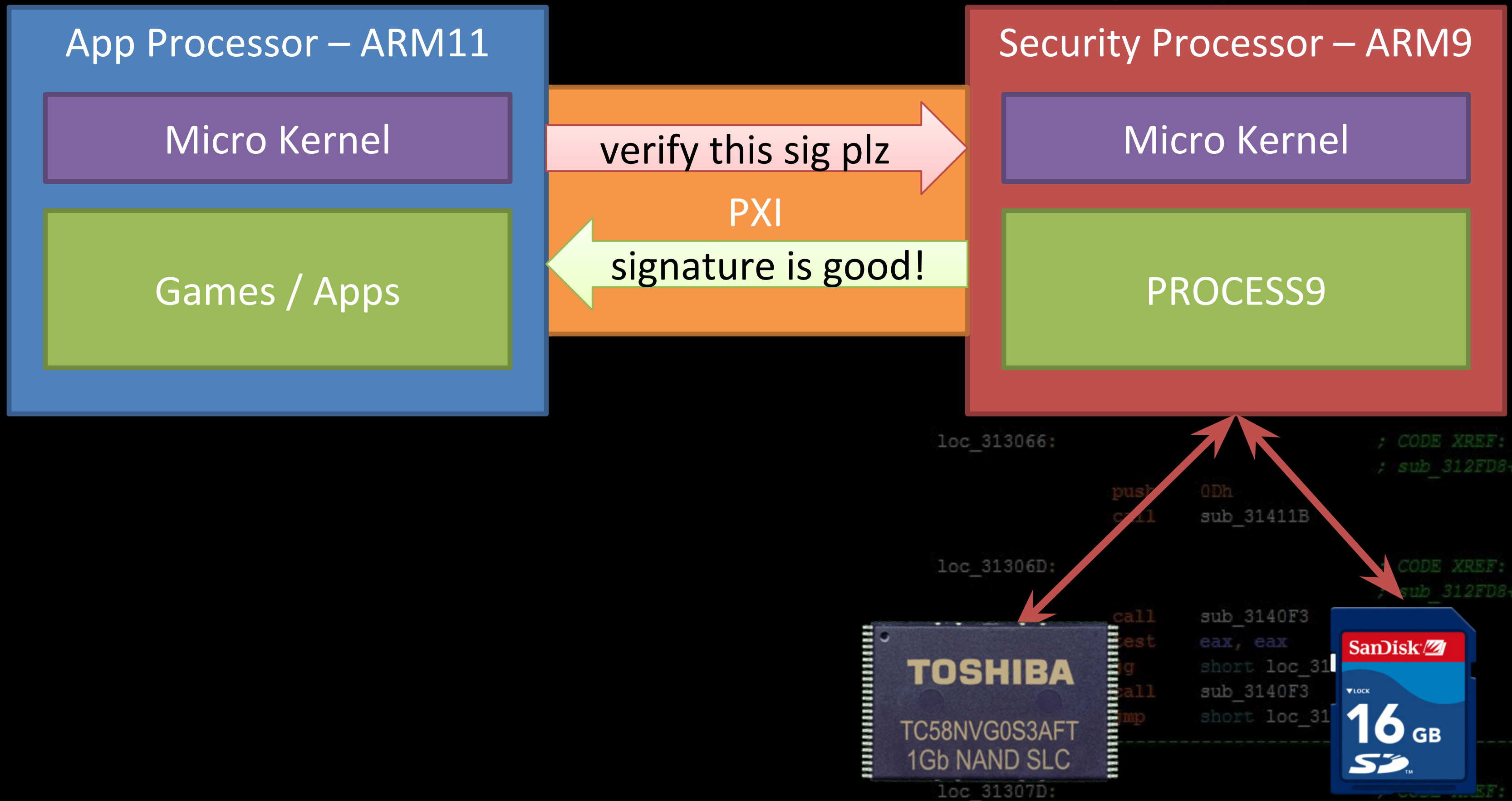
Normal PXI Requests



Normal PXI Requests



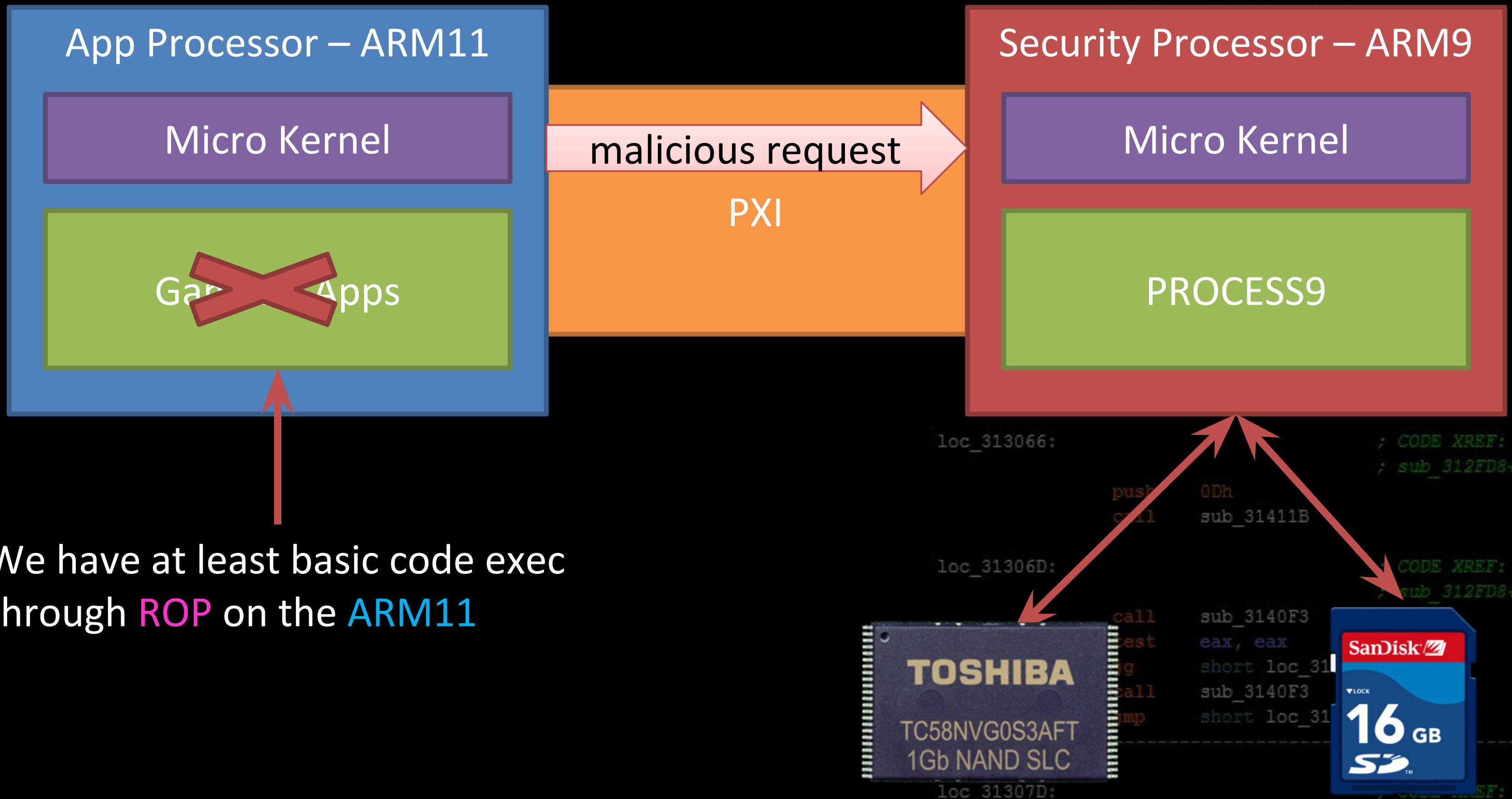
Normal PXI Requests



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

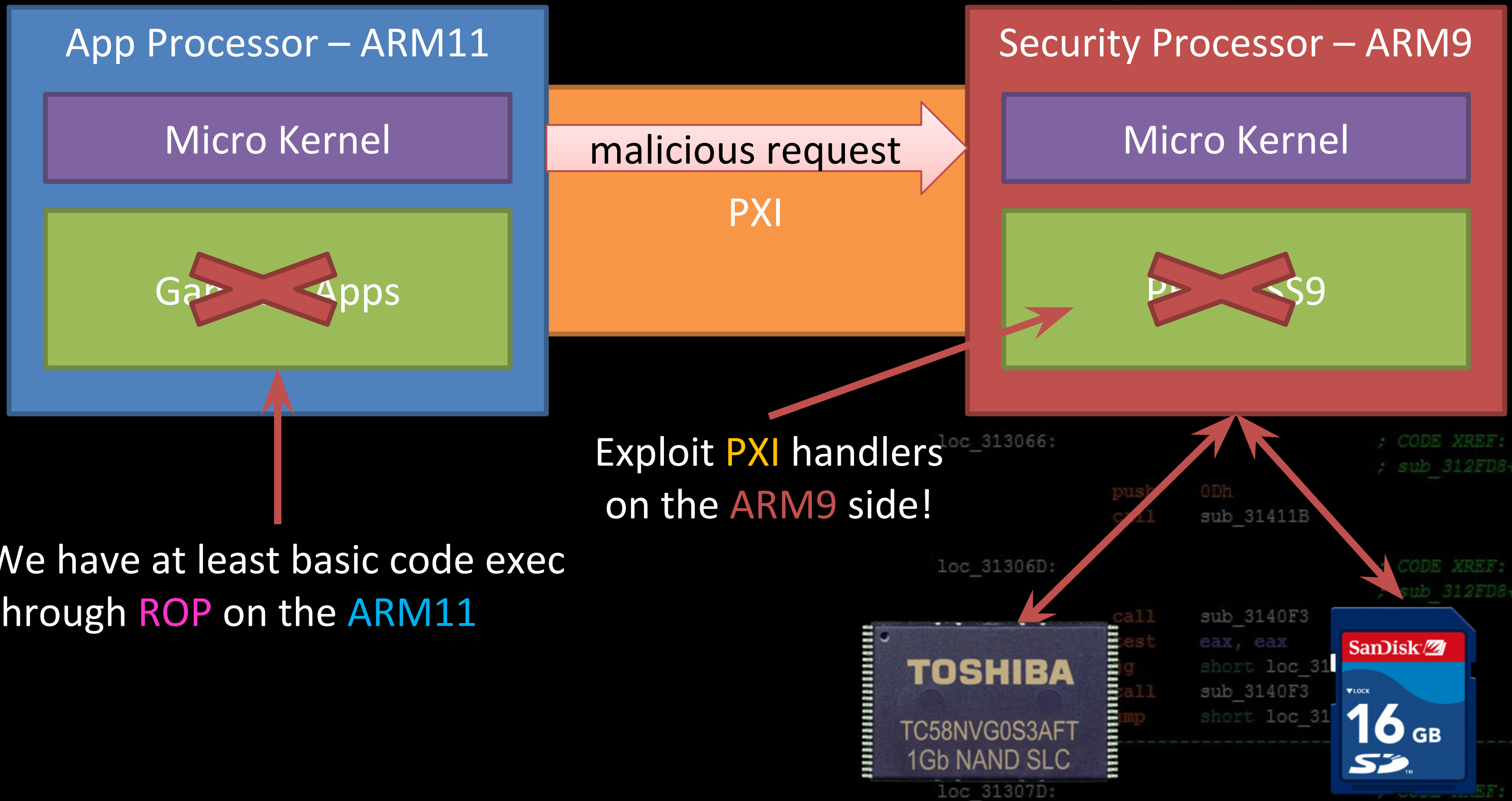
TAKING OVER THE ARM9

Malicious PXI Requests



We have at least basic code exec through ROP on the ARM11

Malicious PXI Requests

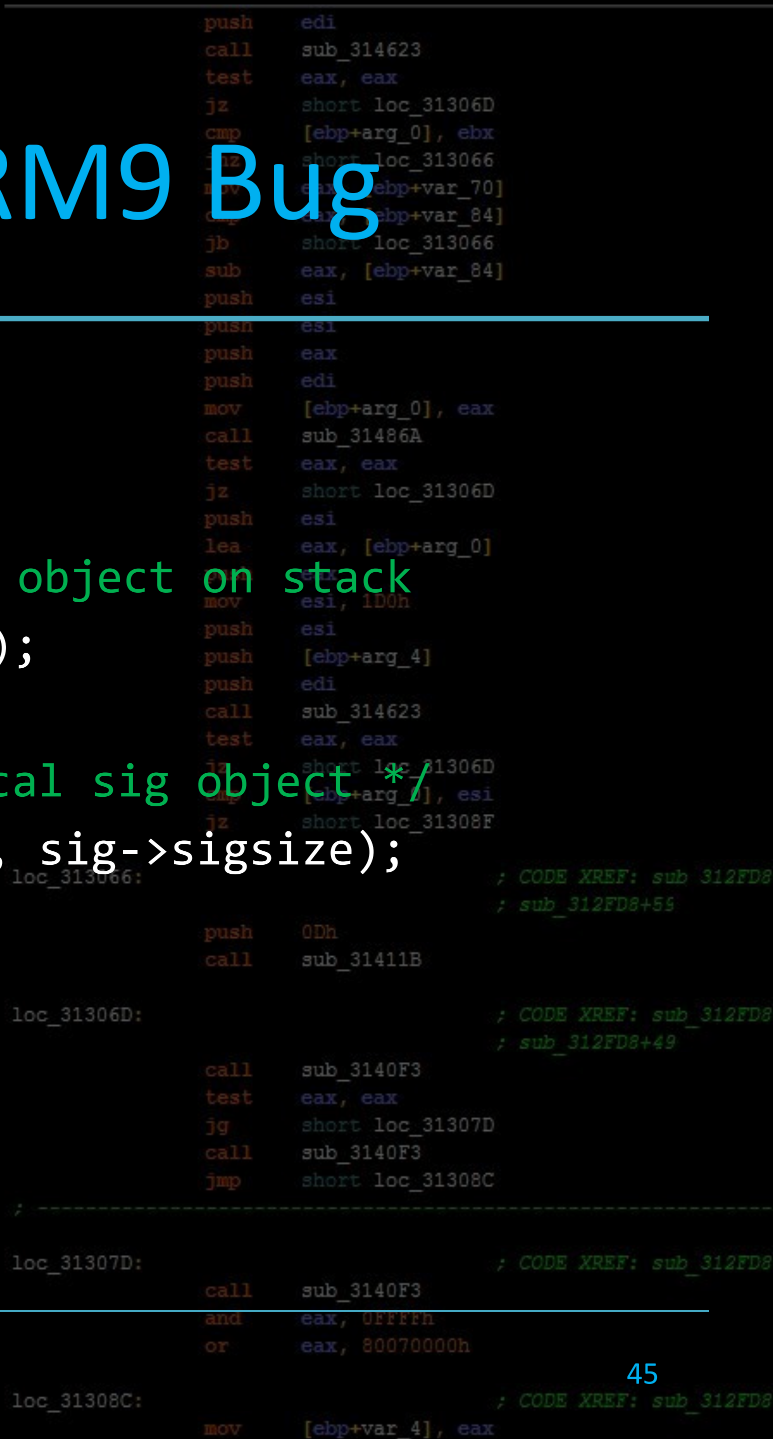


Exploit PXI handlers on the ARM9 side!

We have at least basic code exec through ROP on the ARM11

Pseudocode of the ARM9 Bug

```
int ps_VerifyRsaSha256(RSA_SIG * sig)
{
    RSA_SIG localsig; // 0x208 byte sig object on stack
    memset(localsig, 0, sizeof(RSA_SIG));

    /* copy the RSA signature into a local sig object */
    memcpy(localsig.sigbuf, sig->sigbuf, sig->sigsize);

    ...

    return result;
}
```



```
        push    edi
        call    sub_314623
        test    eax, eax
        jz      short loc_31306D
        cmp     [ebp+arg_0], ebx
        jz      short loc_313066
        mov     eax, [ebp+var_70]
        cmp     eax, [ebp+var_84]
        jb      short loc_313066
        sub     eax, [ebp+var_84]
        push    esi
        push    esi
        push    eax
        push    edi
        mov     [ebp+arg_0], eax
        call    sub_31486A
        test    eax, eax
        jz      short loc_31306D
        push    esi
        lea     eax, [ebp+arg_0]
        push    eax
        mov     esi, 1D0h
        push    esi
        push    [ebp+arg_4]
        push    edi
        call    sub_314623
        test    eax, eax
        jz      short loc_31306D
        cmp     [ebp+arg_4], esi
        jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
        push    0Dh
        call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
        call    sub_3140F3
        test    eax, eax
        jg      short loc_31307D
        call    sub_3140F3
        jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
        call    sub_3140F3
        and     eax, 0FFFFh
        or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
        mov     [ebp+var_4], eax
```

Pseudocode of the ARM9 Bug

```
int ps_VerifyRsaSha256(RSA_SIG * sig)
{
    RSA_SIG localsig; // 0x208 byte sig object on stack
    memset(localsig, 0, sizeof(RSA_SIG));

    /* copy the RSA signature into a local sig object */
    memcpy(localsig.sigbuf, sig->sigbuf, sig->sigsize);

    ...

    return result;
}
```

Attacker Controlled Data

VerifyRsaSha256() – Jun. 2013

- Bug is basically a memcpy with user controlled data, and a user specified size
- No DEP or ASLR on the ARM9, simply overwrite return address and jump onto your buffer! (:
- With control of the ARM9 you can do anything
  - Load a custom firmware & soft reboot the system

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]

                        _0], eax
                        6A

                        c_31306D

                        p+arg_0]

                        h

                        _4]

                        23

                        c_31306D
                        _0], esi
                        c_31308F

                                        ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55

                        1B

                                        ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                        F3

                        oc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

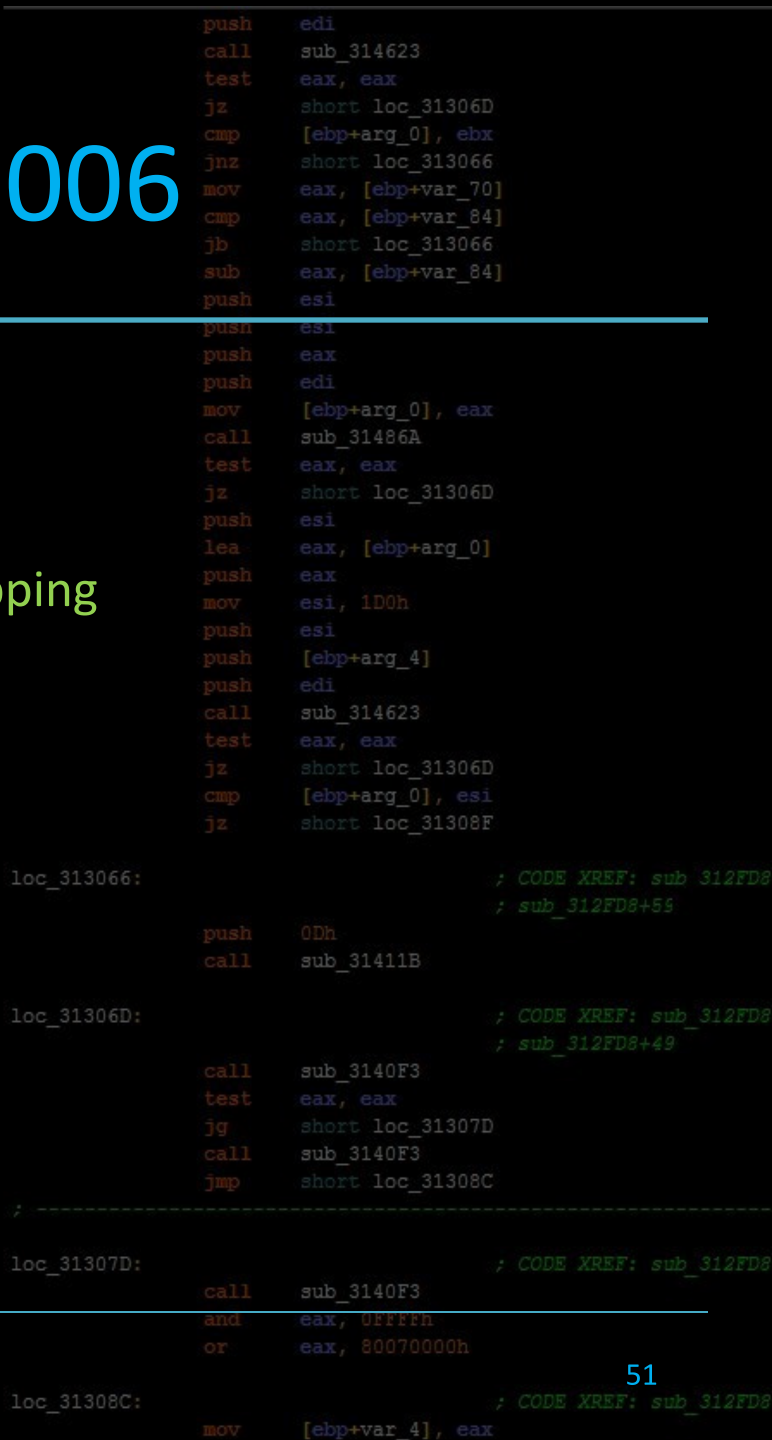
Owning the 3DS

- Code exec on the ARM11 is easy
  - Tons of crappy vulnerable games everywhere, less exciting exploits exist to do this
- Owning the ARM9 is much harder
  - Limited attack surface with little user input
  - Owning the ARM9 is what separates the boys from the men

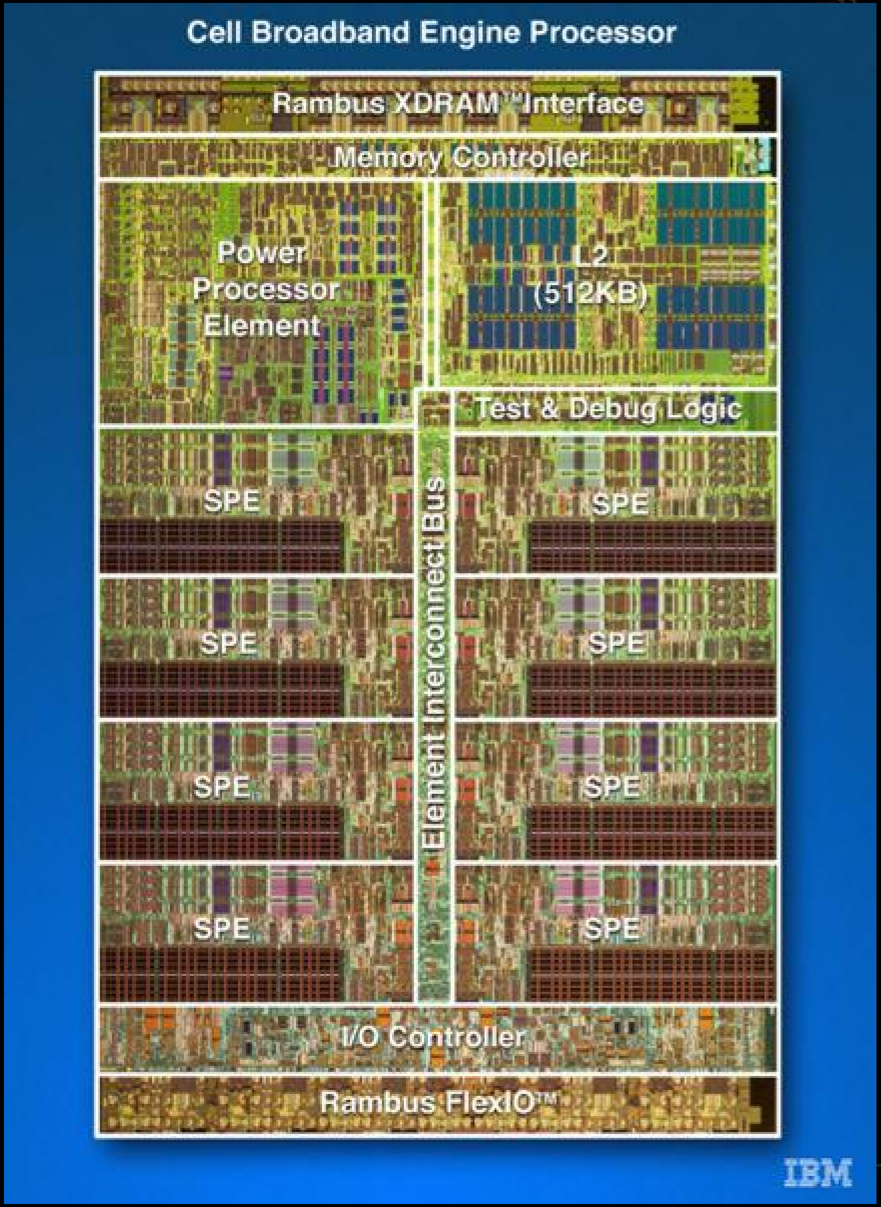
PlayStation 3 – Nov. 2006

PlayStation 3 – Nov. 2006

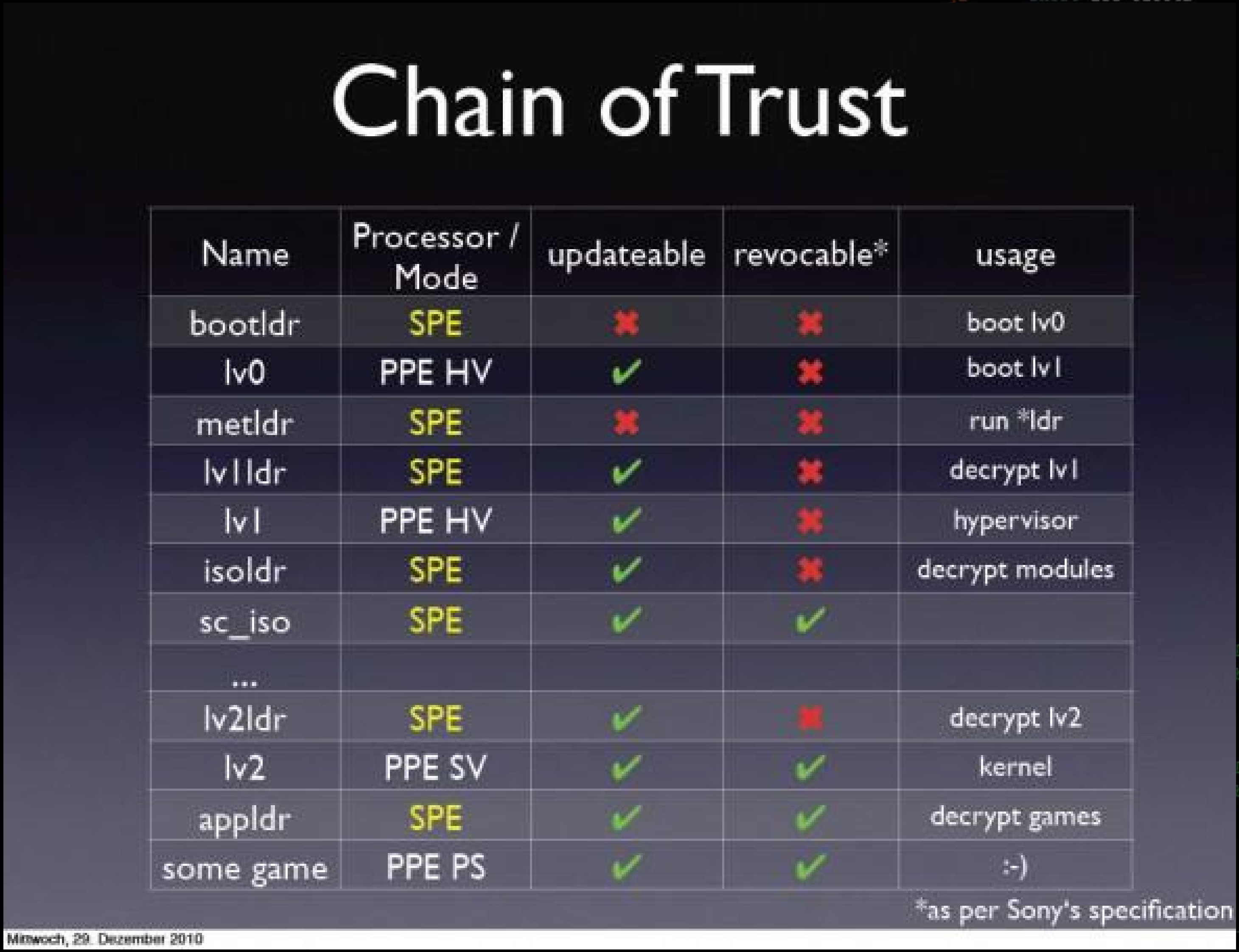
- Security Perspective
  - FreeBSD Based OS
  - Only runs signed code or executables
  - Rigorous chain of trust, secure bootstrapping
  - Cell Architecture
    - Isolates cores from each other, HV
    - Dedicated System / Security Cell
  - Encrypted runtime memory
  - Encrypted HDD
  - eFuses
  - NX/DEP
  - No ASLR



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```



```
push    edi
        sub_314623
        eax, eax
        short loc_31306D
        [ebp+arg_0], ebx
        short loc_313066
        eax, [ebp+var_70]
        eax, [ebp+var_84]
        short loc_313066
        eax, [ebp+var_84]
        esi
        esi
        eax
        edi
        [ebp+arg_0], eax
        sub_31486A
        eax, eax
        short loc_31306D
        esi
        eax, [ebp+arg_0]
        eax
        esi, 1D0h
        esi
        [ebp+arg_4]
        edi
        sub_314623
        eax, eax
        short loc_31306D
        [ebp+arg_0], esi
        short loc_31308F

                                ; CODE XREF: sub_312FD8
                                ; sub_312FD8+55
        0Dh
        sub_31411B

                                ; CODE XREF: sub_312FD8
                                ; sub_312FD8+49
        sub_3140F3
        eax, eax
        short loc_31307D
        sub_3140F3
        short loc_31308C
---------------------------------------------------------

loc_31307D:                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```



Chain of Trust

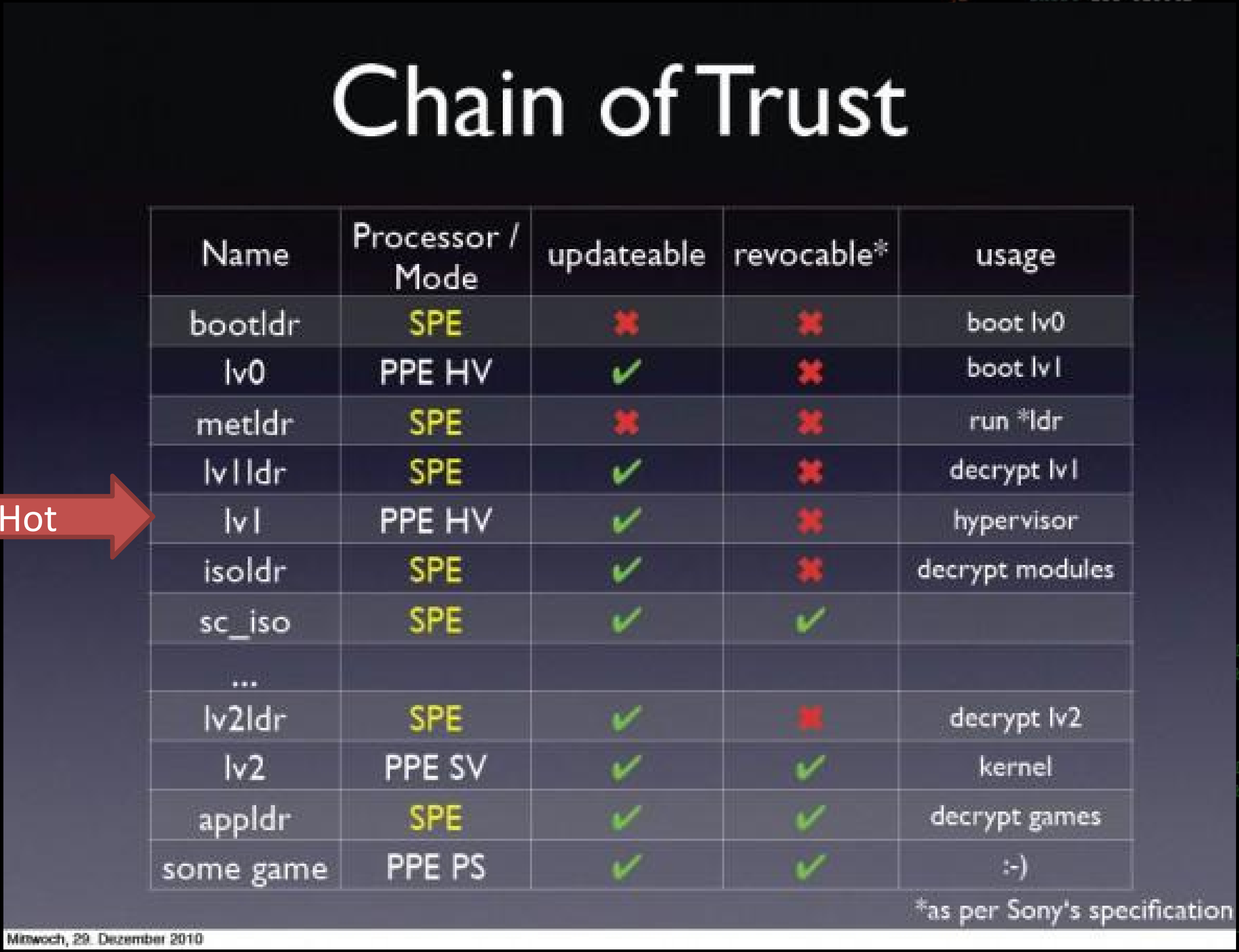
Name	Processor / Mode	updateable	revocable*	usage
bootldr	SPE	✖	✖	boot lv0
lv0	PPE HV	✓	✖	boot lv1
metldr	SPE	✖	✖	run *ldr
lv1ldr	SPE	✓	✖	decrypt lv1
lv1	PPE HV	✓	✖	hypervisor
isoldr	SPE	✓	✖	decrypt modules
sc_iso	SPE	✓	✓	
...				
lv2ldr	SPE	✓	✖	decrypt lv2
lv2	PPE SV	✓	✓	kernel
appldr	SPE	✓	✓	decrypt games
some game	PPE PS	✓	✓	:-)

*as per Sony's specification

Mittwoch, 29. Dezember 2010

GeoHot Owns PS3 Hv – Jan. 2010

- Through OtherOS (Linux on PS3) and chip glitching, GeoHot owns the PS3 Hypervisor
- Glitching 'creates' a use after free scenario in the Hypervisor that is then exploited to get code exec
- Dumps of PS3 HV & kernel make their way public



Chain of Trust

Name	Processor / Mode	updateable	revocable*	usage
bootldr	SPE	✗	✗	boot lv0
lv0	PPE HV	✓	✗	boot lv1
metldr	SPE	✗	✗	run *ldr
lv1ldr	SPE	✓	✗	decrypt lv1
lv1	PPE HV	✓	✗	hypervisor
isoldr	SPE	✓	✗	decrypt modules
sc_iso	SPE	✓	✓	
...				
lv2ldr	SPE	✓	✗	decrypt lv2
lv2	PPE SV	✓	✓	kernel
appldr	SPE	✓	✓	decrypt games
some game	PPE PS	✓	✓	:-)

*as per Sony's specification

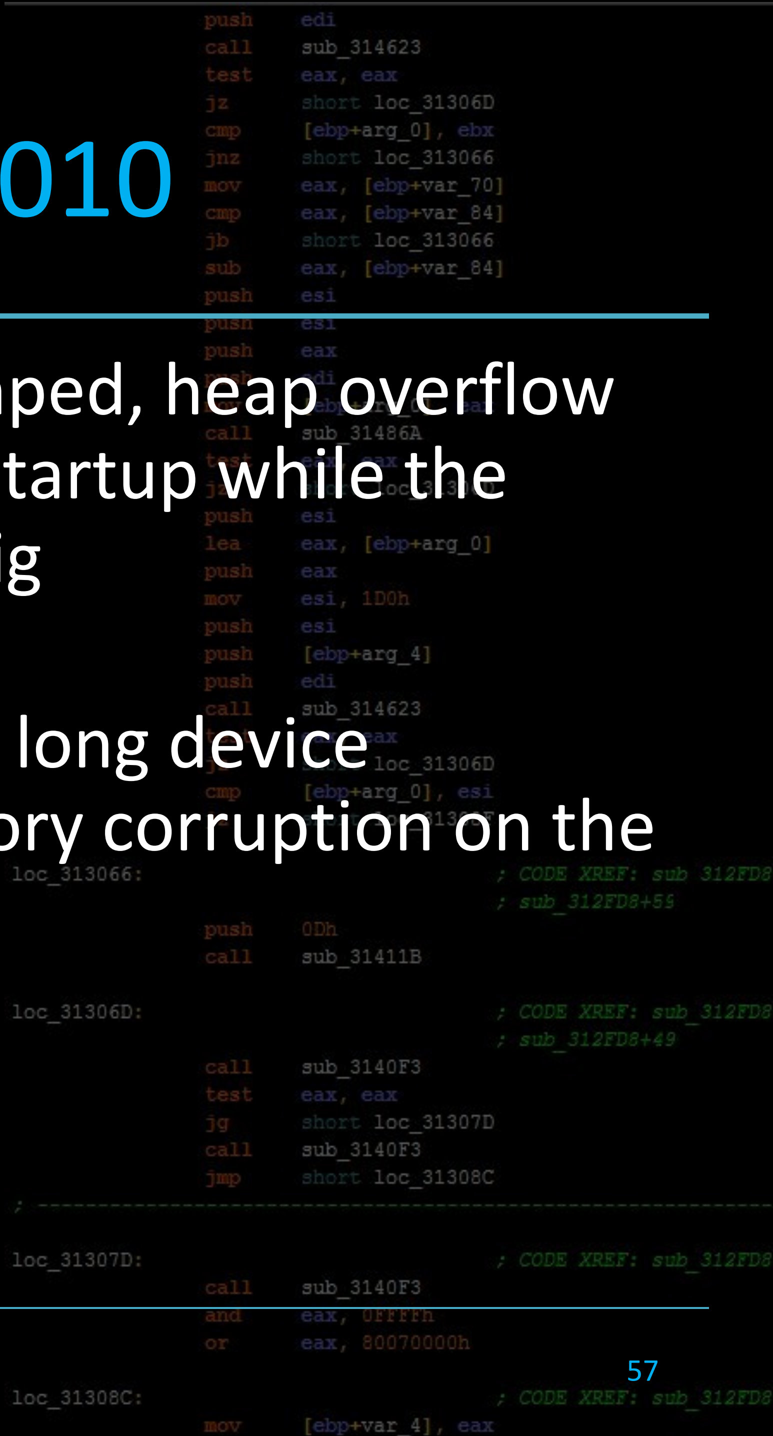
Mittwoch, 29. Dezember 2010

GeoHot

More Privileged

Sony Disables OtherOS – Mar. 2010

PS3 Jailbreak – Aug. 2010



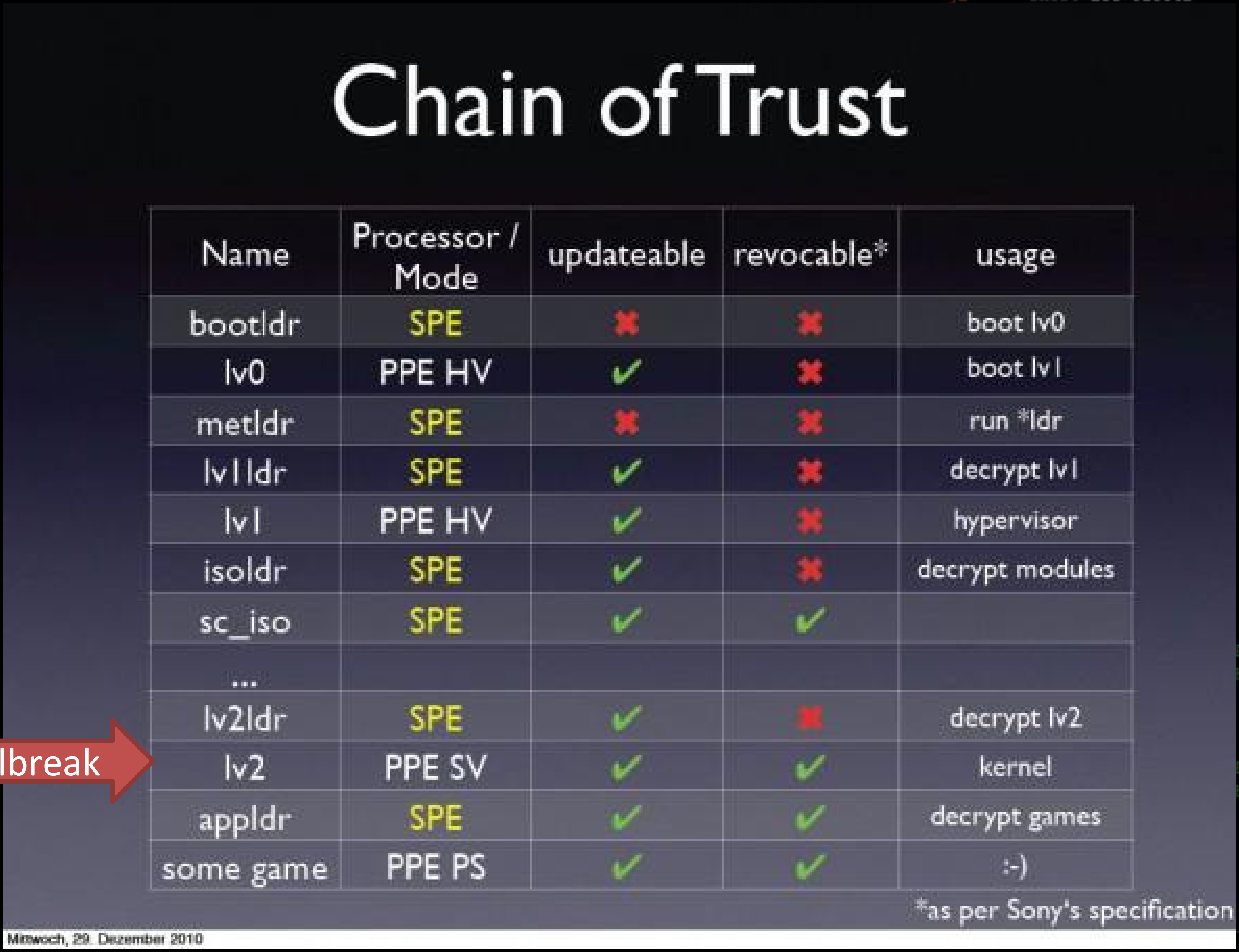
- With the PS3 Kernel (LV2) dumped, heap overflow found in USB handling during startup while the system searches for a service jig
- The main bug is an overflow in long device descriptors that leads to memory corruption on the heap
- Results in control of the LV2

PS3 Jailbreak – Aug. 2010

PS3 Jailbreak – Aug. 2010

- Heap overflow setup and triggered through a <u>USB hub</u> (oops) and six USB's

- It's a bit like musical chairs, plugging and unplugging a number of USB's to malloc/free stuff – everyone just emulates this process with a single USB



Chain of Trust

Name	Processor / Mode	updateable	revocable*	usage
bootldr	SPE	✗	✗	boot lv0
lv0	PPE HV	✓	✗	boot lv1
metldr	SPE	✗	✗	run *ldr
lv1ldr	SPE	✓	✗	decrypt lv1
lv1	PPE HV	✓	✗	hypervisor
isoldr	SPE	✓	✗	decrypt modules
sc_iso	SPE	✓	✓	
...				
lv2ldr	SPE	✓	✗	decrypt lv2
lv2	PPE SV	✓	✓	kernel
appldr	SPE	✓	✓	decrypt games
some game	PPE PS	✓	✓	:-)

*as per Sony's specification

Mittwoch, 29. Dezember 2010

PS3 Jailbreak

More Privileged

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
```

```
loc_31307D:                             ; CODE XREF: sub_312FD8
        call    sub_3140F3
        and     eax, 0FFFFh
        or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
        mov     [ebp+var_4], eax
```

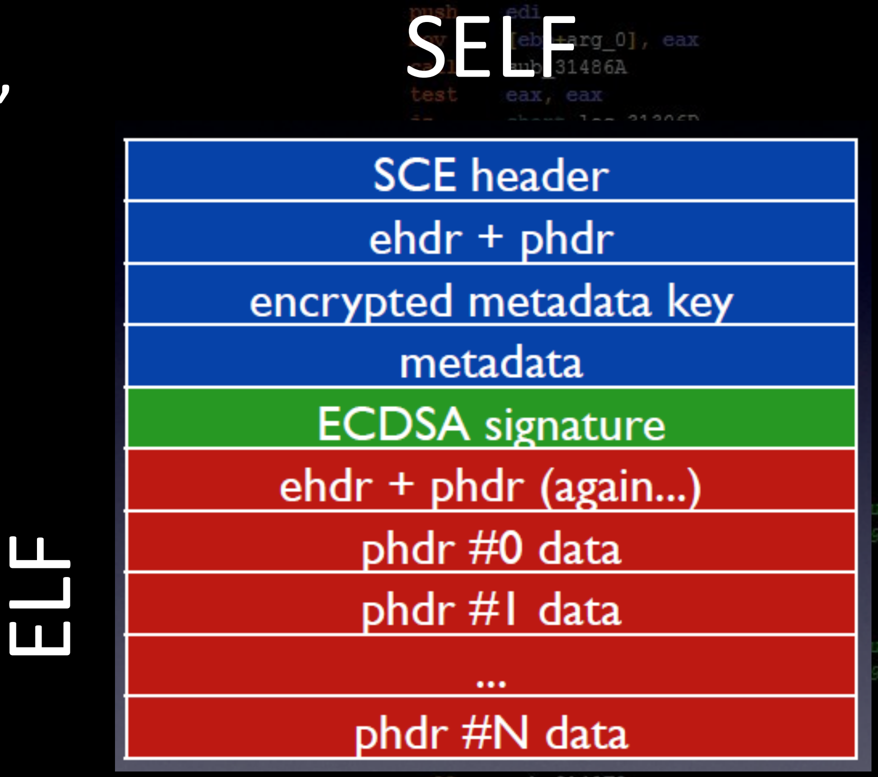
PS3 ECDSA KEY EXTRACTION

Largest console break of this generation stems from crypto flaw

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

PS3 ECDSA Key Extraction – Jan. 2011

- Executables running on the PS3 are modified ELF's known as SELF's
- Signed by Sony's ECDSA Key, encrypted by the associated Lv(0,1,2) keys
  - Elliptic Curve Digital Signature Algorithm

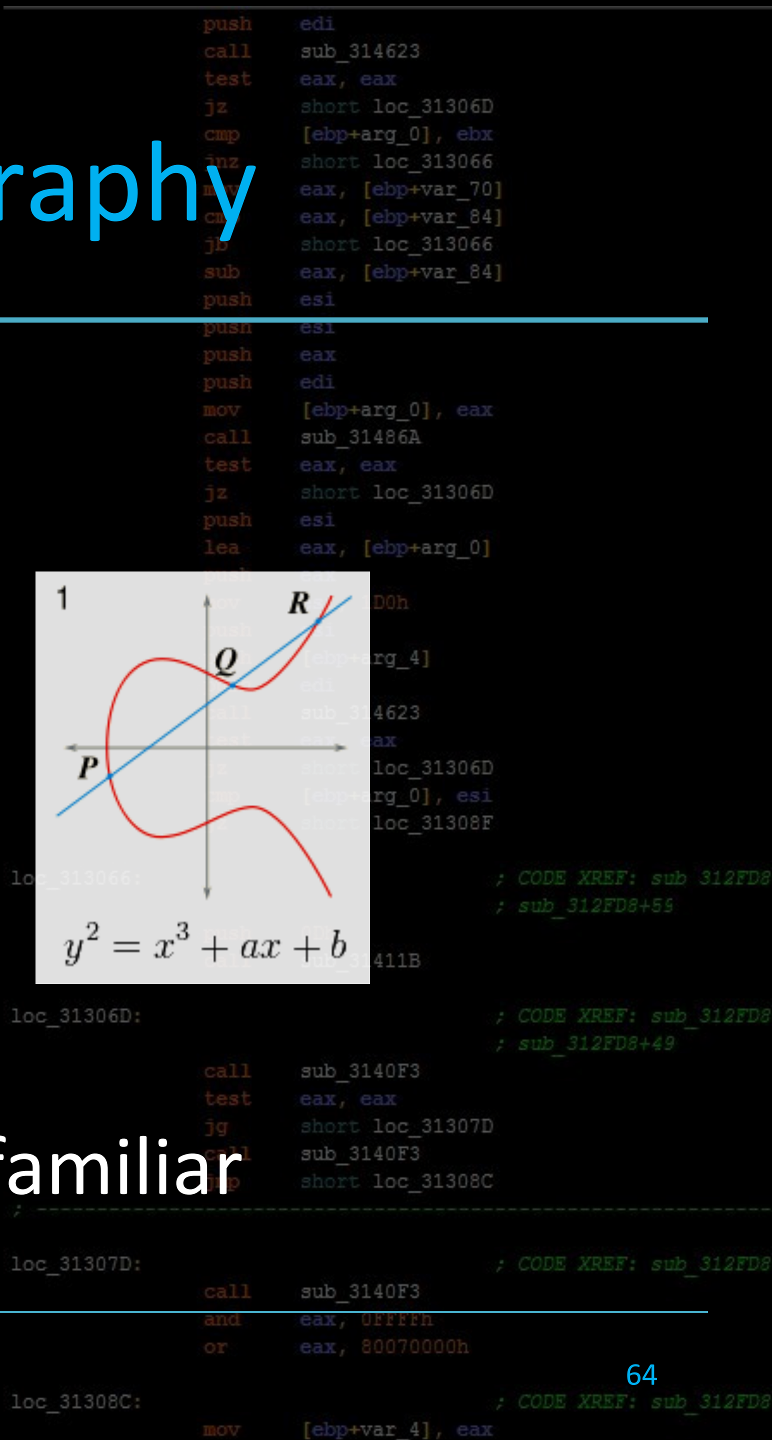


PS3 ECDSA Key Extraction – Jan. 2011

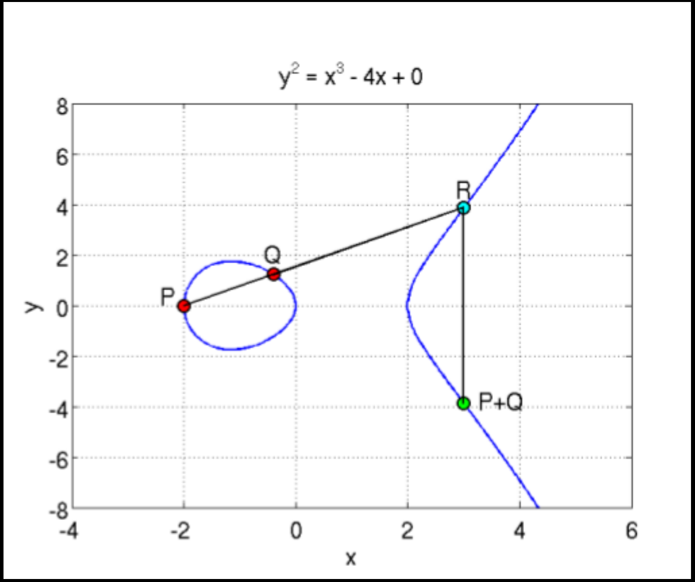
- With control of the LV2, you can make crypto requests to the security SPE and use it as a black box

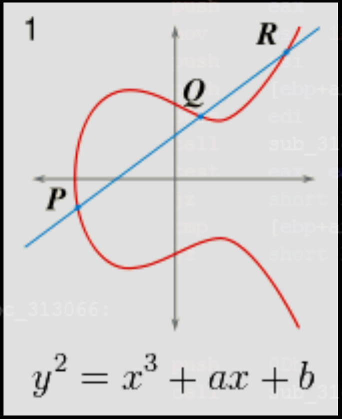
- An egregious crypto implementation flaw is uncovered by fail0verflow regarding Sony’s ECDSA signatures

Elliptic Curve Cryptography



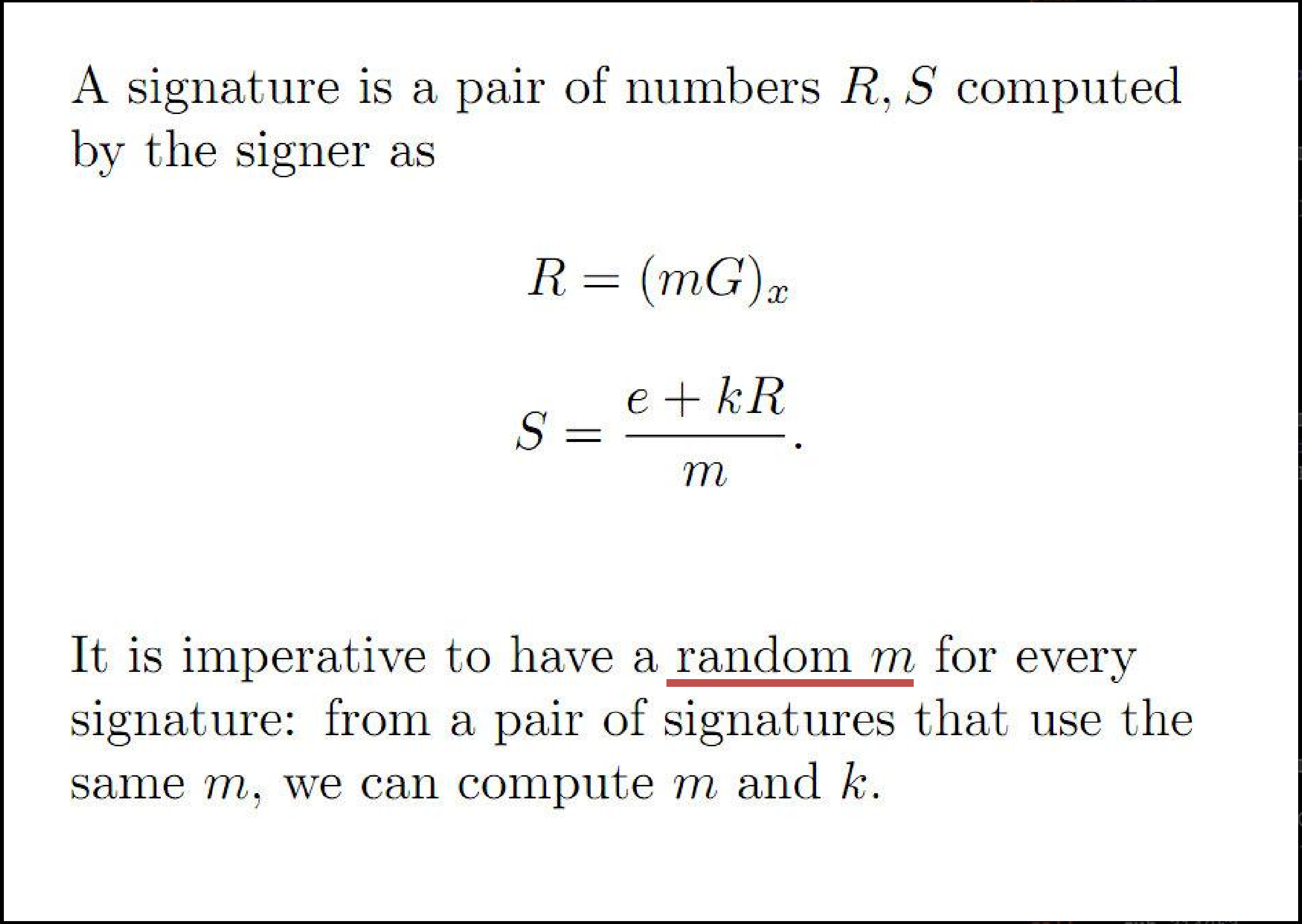
```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
        D0h
        [ebp+arg_4]
        edi
        sub_314623
        eax
        loc_31306D
        [ebp+arg_0], esi
        loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
        sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```





these might look familiar

Const Instead of Nonce

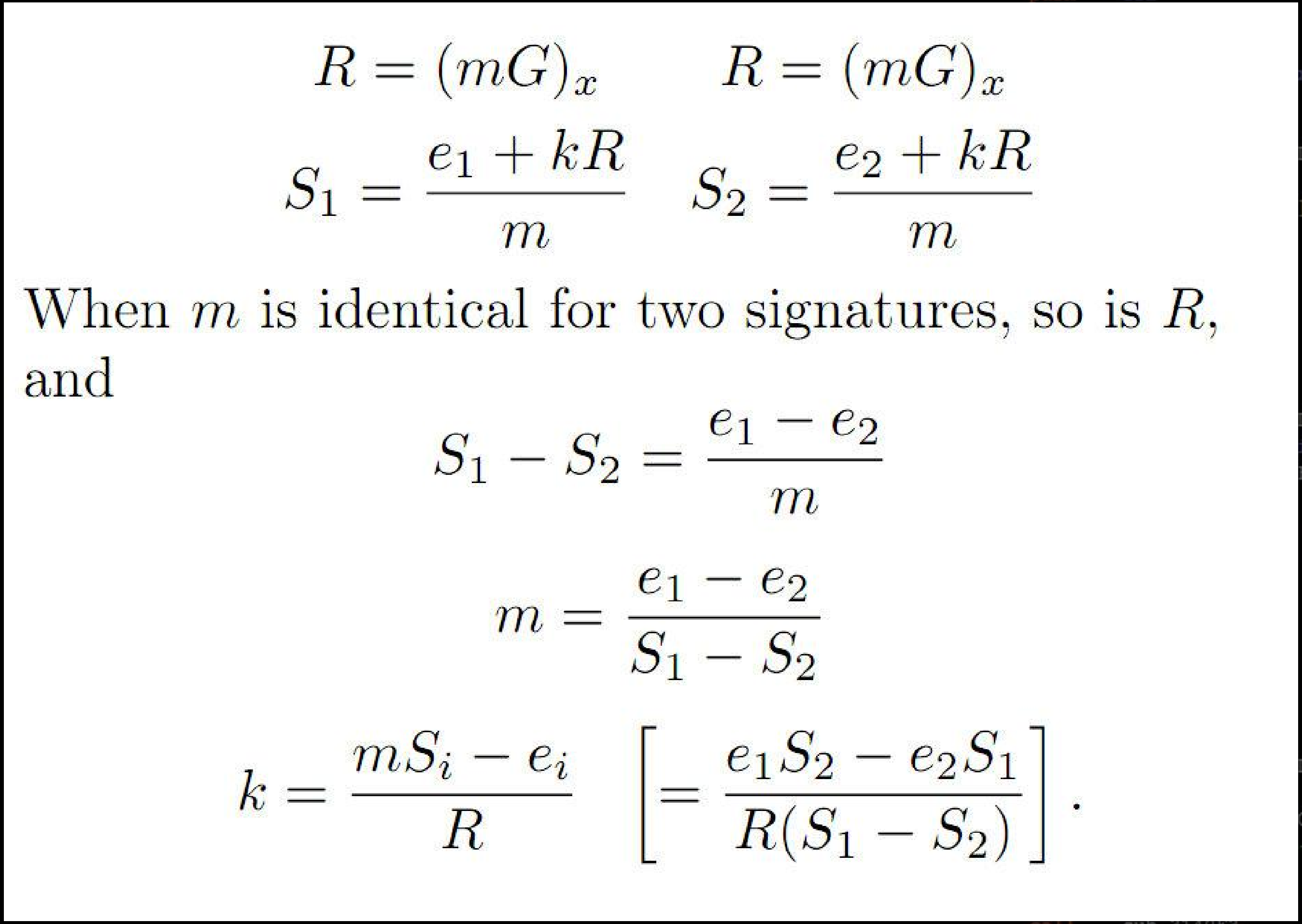

A signature is a pair of numbers R, S computed by the signer as

$$R = (mG)_x$$

$$S = \frac{e + kR}{m}.$$

It is imperative to have a $\underline{\text{random } m}$ for every signature: from a pair of signatures that use the same m, we can compute m and k.

Const Instead of Nonce



$$R = (mG)_x \qquad R = (mG)_x$$

$$S_1 = \frac{e_1 + kR}{m} \qquad S_2 = \frac{e_2 + kR}{m}$$

When m is identical for two signatures, so is R, and

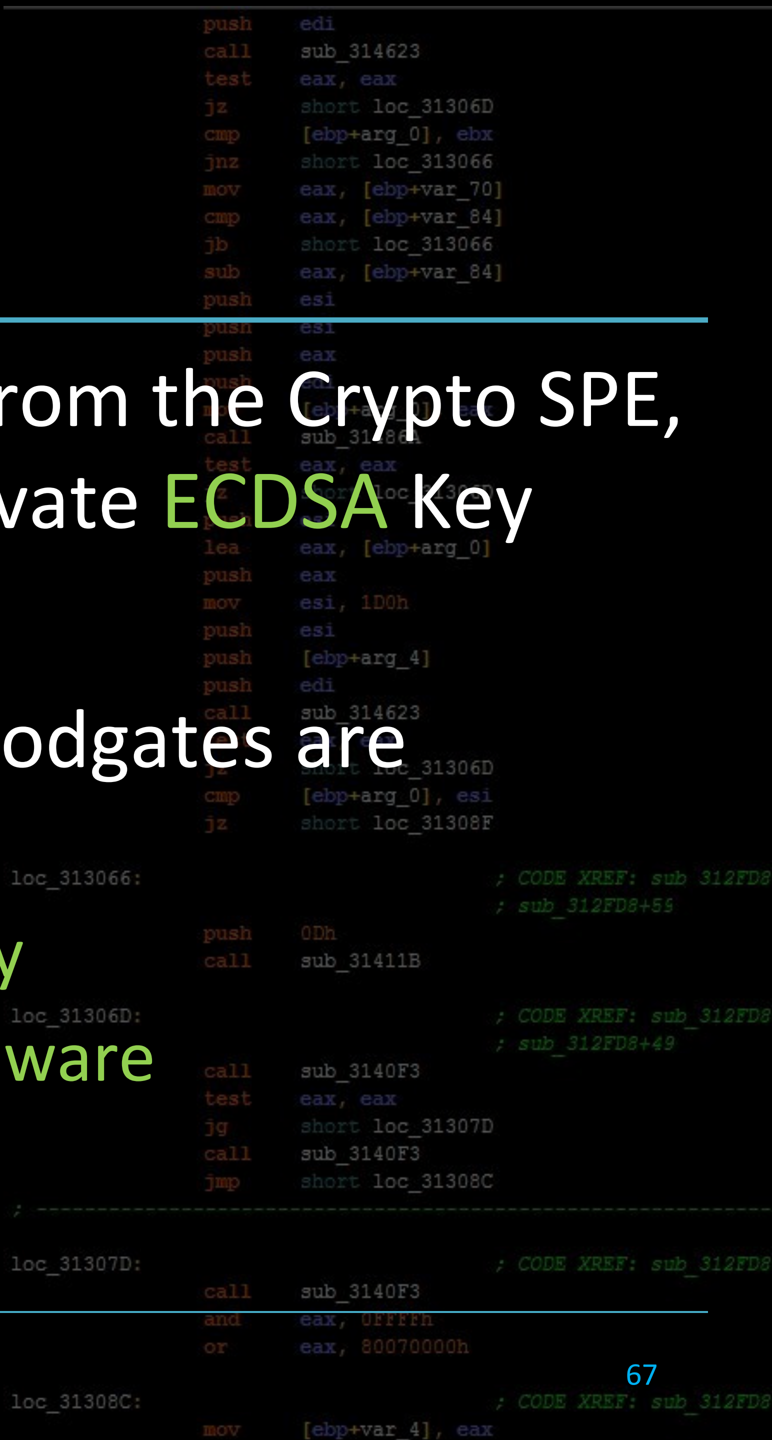
$$S_1 - S_2 = \frac{e_1 - e_2}{m}$$

$$m = \frac{e_1 - e_2}{S_1 - S_2}$$

$$k = \frac{mS_i - e_i}{R} \quad \left[= \frac{e_1 S_2 - e_2 S_1}{R(S_1 - S_2)}\right].$$

Effects of Missteps

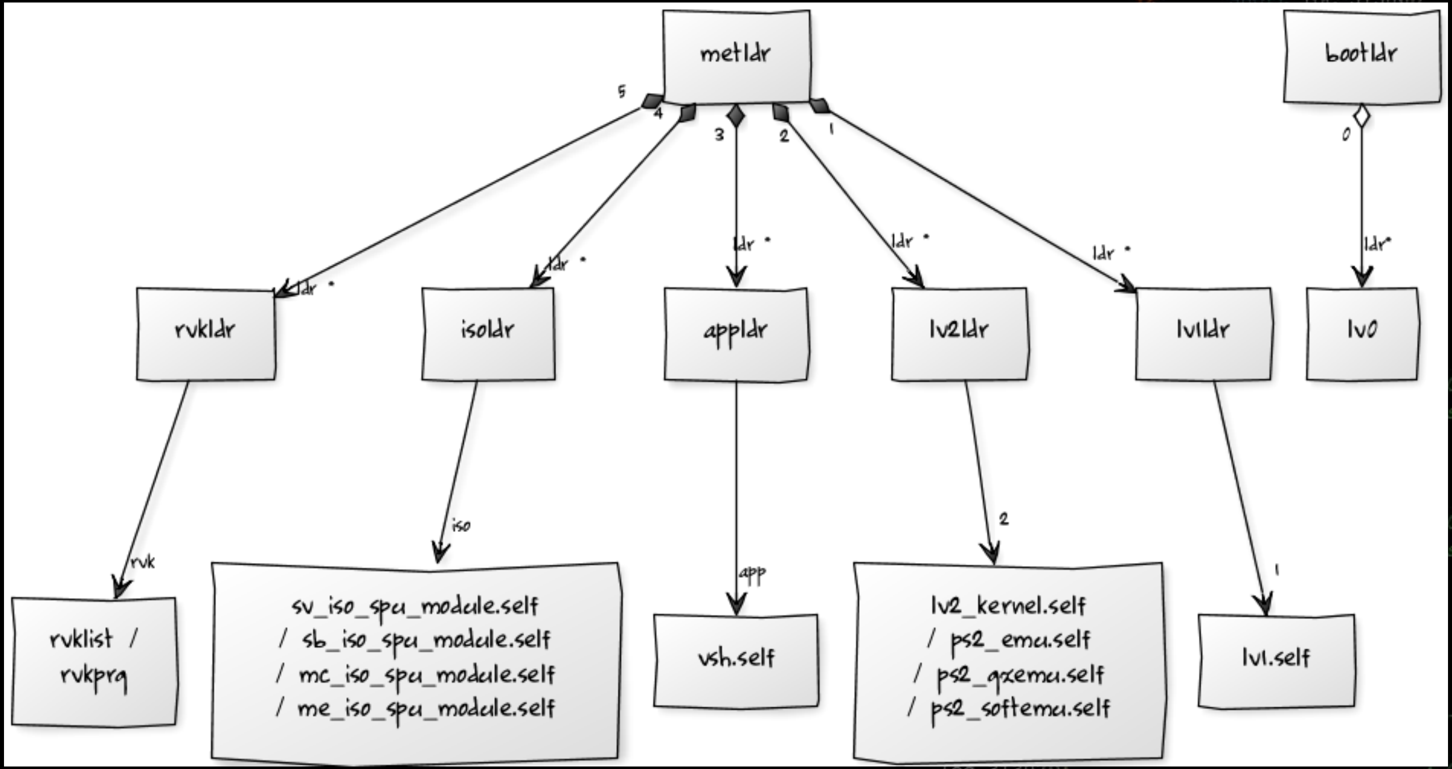
- With only TWO signatures from the Crypto SPE, you can compute Sony's Private ECDSA Key
- With the ECDSA Key, the floodgates are opened
  - You can sign anything as Sony
  - This key is embedded in hardware



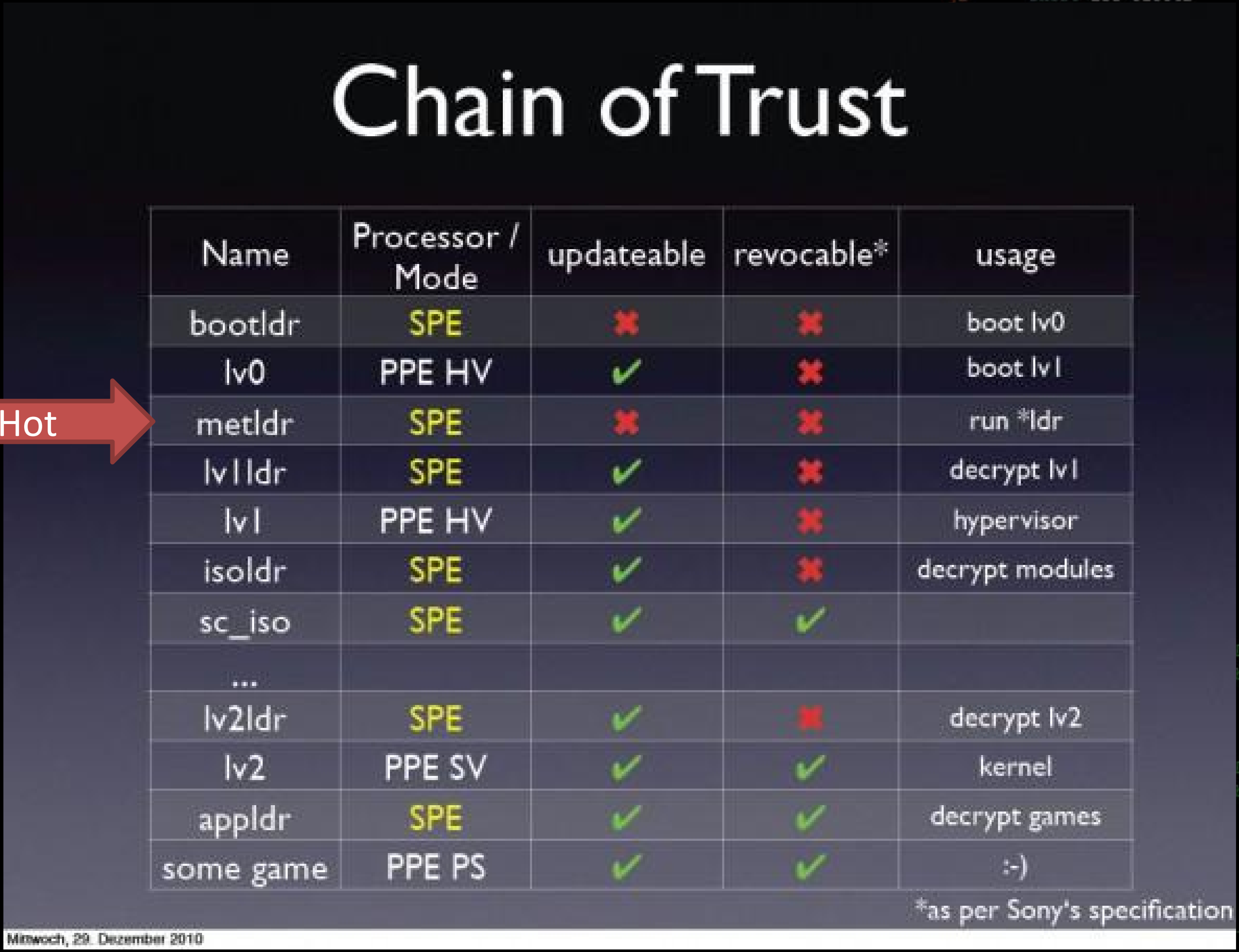
```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    [illegible]
                call    [illegible]
                test    eax, eax
                [illegible]
                [illegible]
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                [illegible]
                [illegible]    short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

metldr Owned

- Geohot releases metldr decryption keys



metldr
bootldr
5
4
3
2
1
0
ldr *
ldr *
ldr *
ldr *
ldr *
ldr*
rvkldr
isoldr
appldr
lv2ldr
lv1ldr
lv0
rvk
iso
app
2
1
rvklist / rvkprg
sv_iso_spu_module.self / sb_iso_spu_module.self / mc_iso_spu_module.self / me_iso_spu_module.self
vsh.self
lv2_kernel.self / ps2_emu.self / ps2_gxemu.self / ps2_softemu.self
lv1.self



Chain of Trust

Name	Processor / Mode	updateable	revocable*	usage
bootldr	SPE	✗	✗	boot lv0
lv0	PPE HV	✓	✗	boot lv1
metldr	SPE	✗	✗	run *ldr
lv1ldr	SPE	✓	✗	decrypt lv1
lv1	PPE HV	✓	✗	hypervisor
isoldr	SPE	✓	✗	decrypt modules
sc_iso	SPE	✓	✓	
...				
lv2ldr	SPE	✓	✗	decrypt lv2
lv2	PPE SV	✓	✓	kernel
appldr	SPE	✓	✓	decrypt games
some game	PPE PS	✓	✓	:-)

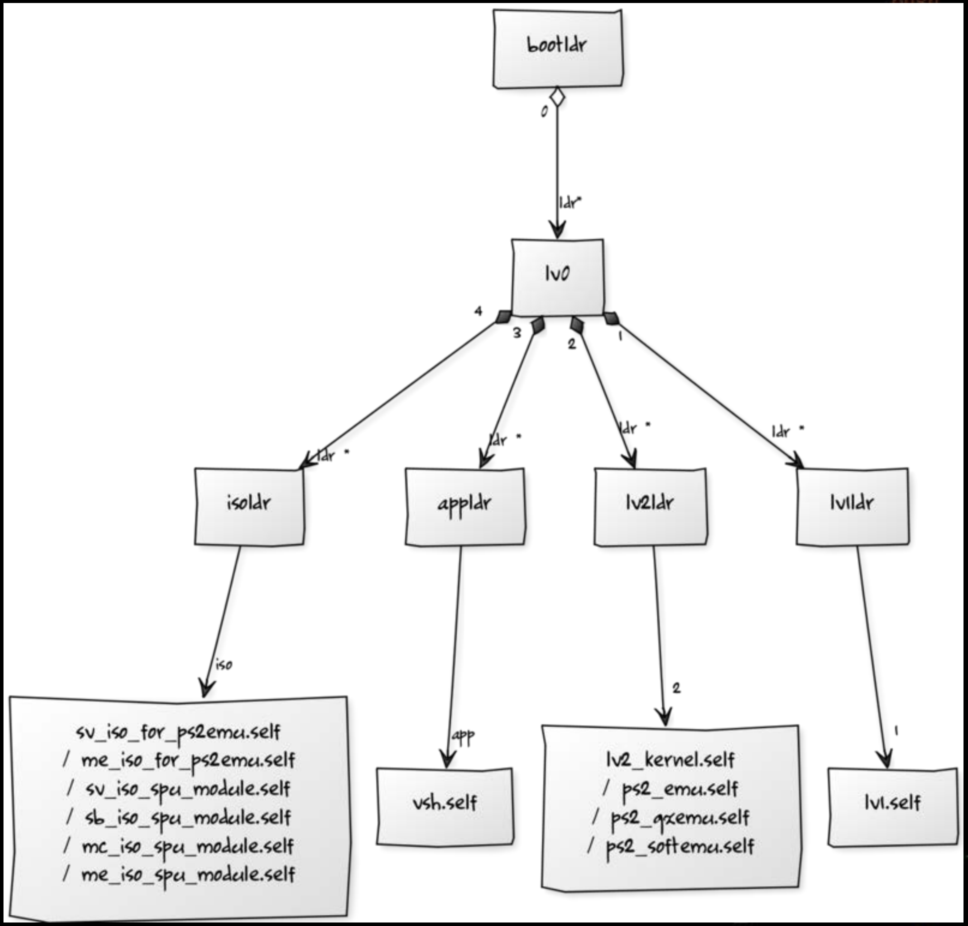
*as per Sony's specification

Mittwoch, 29. Dezember 2010

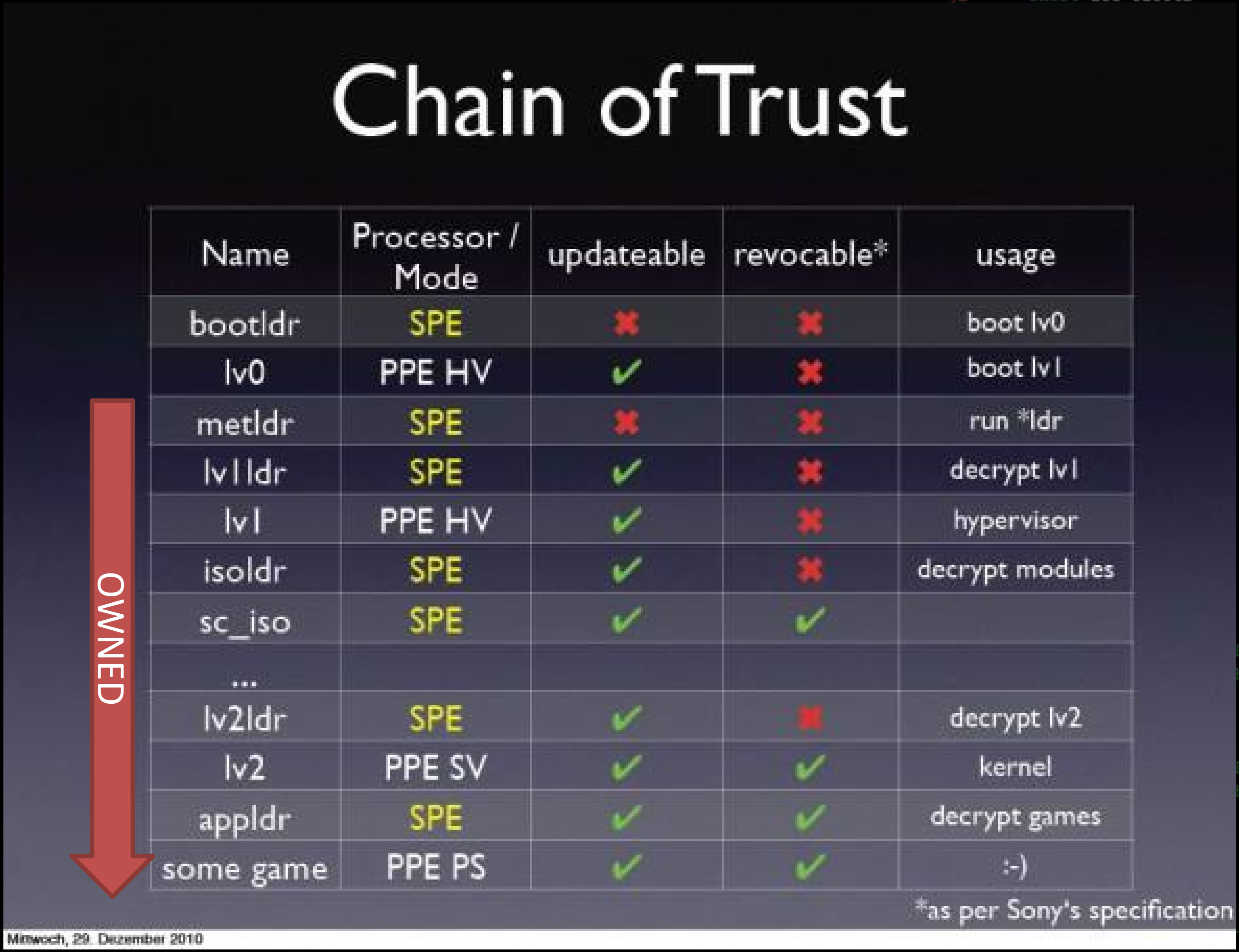
GeoHot

More Privileged

Sony Nukes metldr


bootldr
0
ldr*
lv0
4
3
2
1
ldr *
ldr *
ldr *
ldr *
isoldr
appldr
lv2ldr
lv1ldr
iso
app
2
1
sv_iso_for_ps2emu.self / me_iso_for_ps2emu.self / sv_iso_spu_module.self / sb_iso_spu_module.self / mc_iso_spu_module.self / me_iso_spu_module.self
vsh.self
lv2_kernel.self / ps2_emu.self / ps2_gxemu.self / ps2_softemu.self
lv1.self

Sony Sues Geohot – Jan. 2011



Chain of Trust

Name	Processor / Mode	updateable	revocable*	usage
bootldr	SPE	✖	✖	boot lv0
lv0	PPE HV	✓	✖	boot lv1
metldr	SPE	✖	✖	run *ldr
lv1ldr	SPE	✓	✖	decrypt lv1
lv1	PPE HV	✓	✖	hypervisor
isoldr	SPE	✓	✖	decrypt modules
sc_iso	SPE	✓	✓	
...				
lv2ldr	SPE	✓	✖	decrypt lv2
lv2	PPE SV	✓	✓	kernel
appldr	SPE	✓	✓	decrypt games
some game	PPE PS	✓	✓	:-)

*as per Sony's specification

Mittwoch, 29. Dezember 2010

OWNED

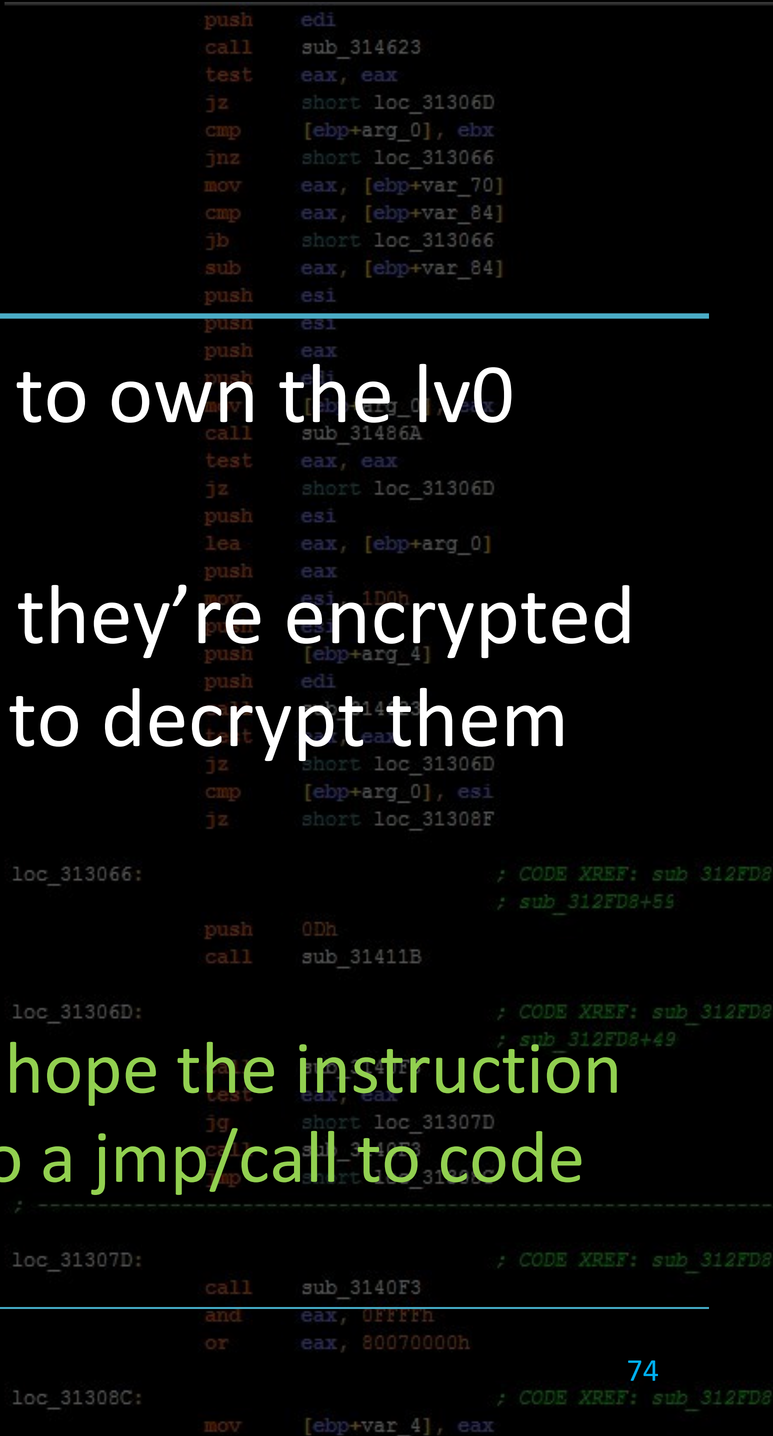
More Privileged

Owning the lv0

- metldr is gone, so you need to own the lv0
- lv0 blobs can be signed, but they’re encrypted and we don’t have the keys to decrypt them
- What do you do?????

Owning the lv0

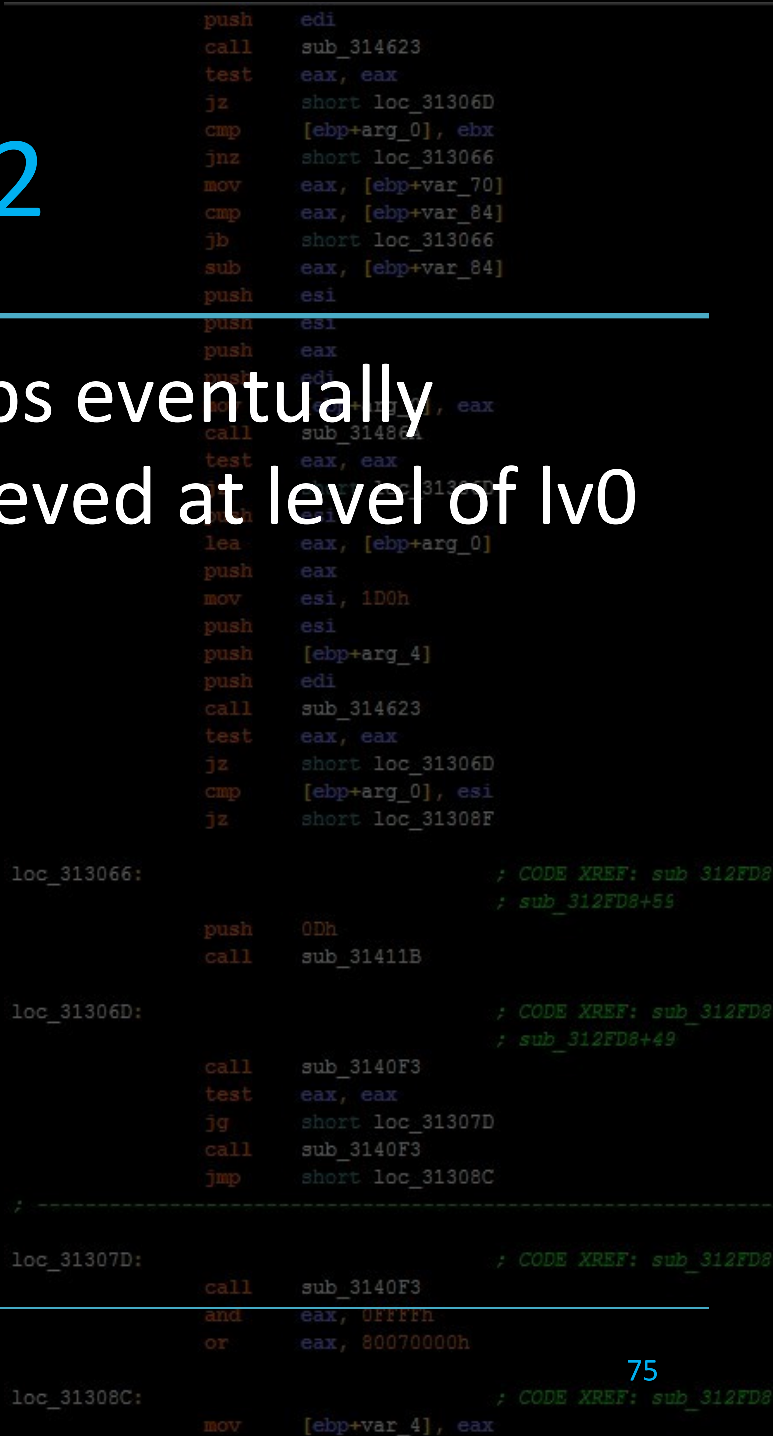
- metldr is gone, so you need to own the lv0
- lv0 blobs can be signed, but they're encrypted and we don't have the keys to decrypt them
- What do you do?????
  - Sign random data blobs, and hope the instruction at the entry point 'decrypt' to a jmp/call to code that you control



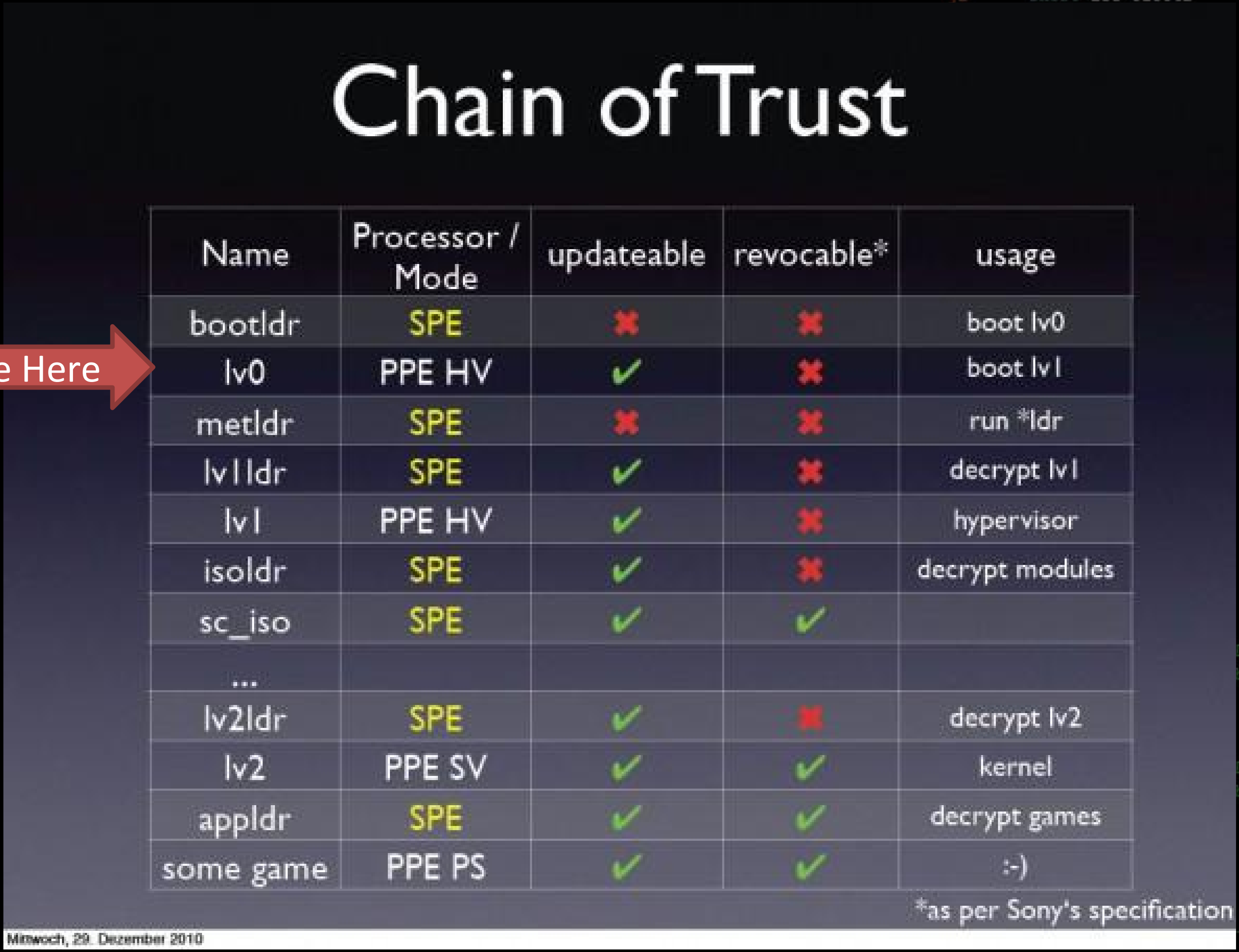
```
          push    edi
          call    sub_314623
          test    eax, eax
          jz      short loc_31306D
          cmp     [ebp+arg_0], ebx
          jnz     short loc_313066
          mov     eax, [ebp+var_70]
          cmp     eax, [ebp+var_84]
          jb      short loc_313066
          sub     eax, [ebp+var_84]
          push    esi
          push    esi
          push    eax
          [illegible]
          [illegible]
          call    sub_31486A
          test    eax, eax
          jz      short loc_31306D
          push    esi
          lea     eax, [ebp+arg_0]
          push    eax
          mov     esi, 1D0h
          [illegible]
          push    [ebp+arg_4]
          push    edi
          [illegible]
          [illegible]
          jz      short loc_31306D
          cmp     [ebp+arg_0], esi
          jz      short loc_31308F

loc_313066:                            ; CODE XREF: sub_312FD8
                                       ; sub_312FD8+55
          push    0Dh
          call    sub_31411B

loc_31306D:                            ; CODE XREF: sub_312FD8
                                       ; sub_312FD8+49
          [illegible]
          test    eax, eax
          jg      short loc_31307D
          call    sub_3140F3
          jmp     short [illegible]
;  ---------------------------------------------------------------

loc_31307D:                            ; CODE XREF: sub_312FD8
          call    sub_3140F3
          and     eax, 0FFFFh
          or      eax, 80070000h

loc_31308C:                            ; CODE XREF: sub_312FD8
          mov     [ebp+var_4], eax
```

lv0 Owned – Oct. 2012

- Trying randomly signed blobs eventually works and execution is achieved at level of lv0



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                [illegible] [illegible], eax
                call    sub_3148[illegible]
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```



Chain of Trust

Name	Processor / Mode	updateable	revocable*	usage
bootldr	SPE	✖	✖	boot lv0
lv0	PPE HV	✓	✖	boot lv1
metldr	SPE	✖	✖	run *ldr
lv1ldr	SPE	✓	✖	decrypt lv1
lv1	PPE HV	✓	✖	hypervisor
isoldr	SPE	✓	✖	decrypt modules
sc_iso	SPE	✓	✓	
...				
lv2ldr	SPE	✓	✖	decrypt lv2
lv2	PPE SV	✓	✓	kernel
appldr	SPE	✓	✓	decrypt games
some game	PPE PS	✓	✓	:-)

*as per Sony's specification

Mittwoch, 29. Dezember 2010

You are Here

More Privileged

lv0 Owned – Oct. 2012

- Decryption keys are retrieved as lv0. Now you can create meaningful lv0 blobs, encrypt them, and sign them

- bootldr also exploited and dumped for fun
  - Not updateable anyway, so it doesn't matter much

GAME OVER
Thank you for playing

PS3 Aftermath

- Sony drops lawsuit against Geohot
  - Must never hack Sony products again
- No more updateable seeds of trust exist on the PS3 that Sony can utilize
  - PS3 totally broken

Lecture Overview

- Secure Systems & Patch Sets
  - OpenBSD
  - SELinux
  - Grsecurity
- Owning Game Consoles
  - Xbox 360
  - Nintendo 3DS
  - PS3
- Current Generation

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

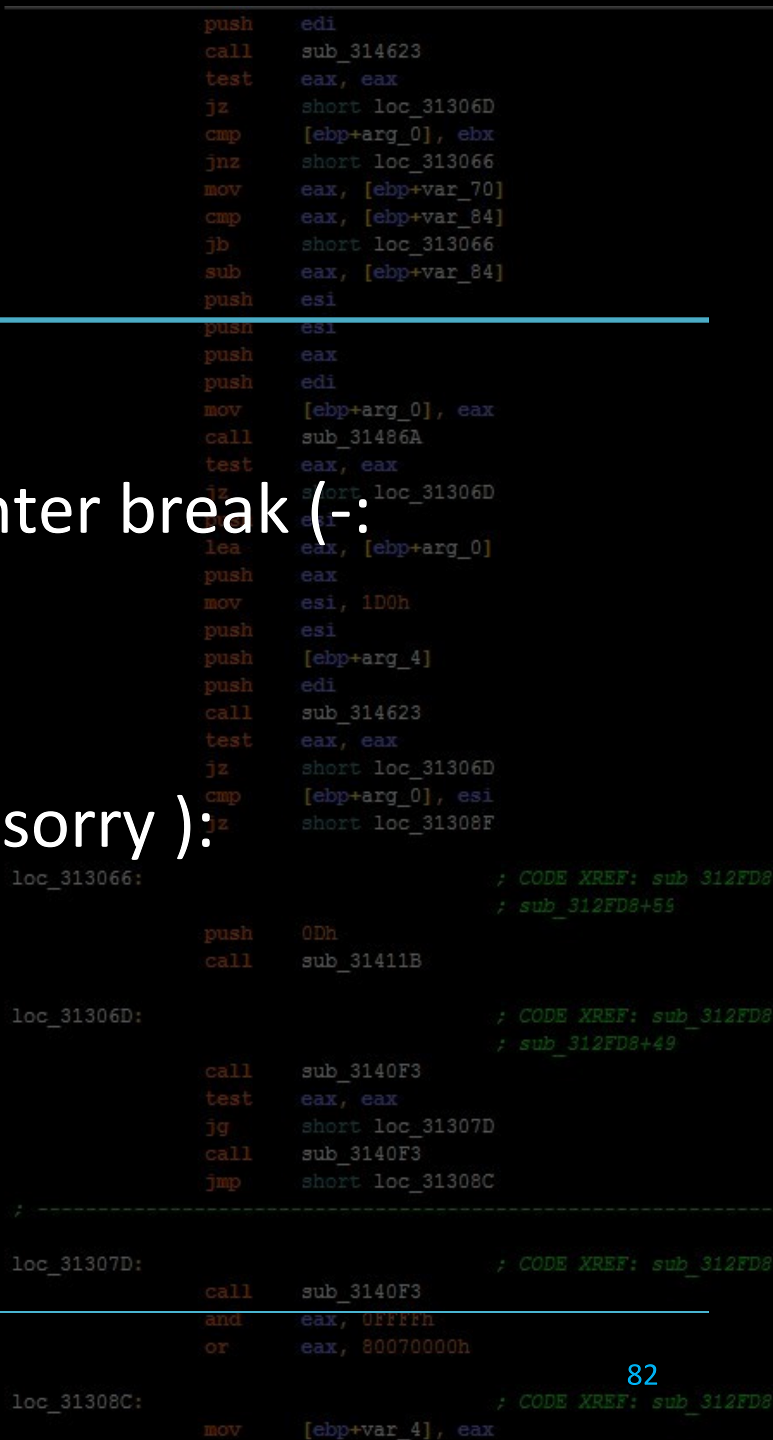
CURRENT GENERATION

A peek at the current generation of consoles

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Current Generation

- Xbox One
  - I did some reversing over winter break (-:
- PS4
  - I don't know as much about, sorry):



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                mov     [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                [illegible]
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Xbox One – Nov. 2013

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

Xbox One OS

- Ditches the 360’s custom operating system
  – Xbox OS (XOS) is forked from Windows 8(ish)
- Spins up minimal windows VM instances to run your games, apps, etc
- Windows 8/8.1 core OS bugs likely apply!

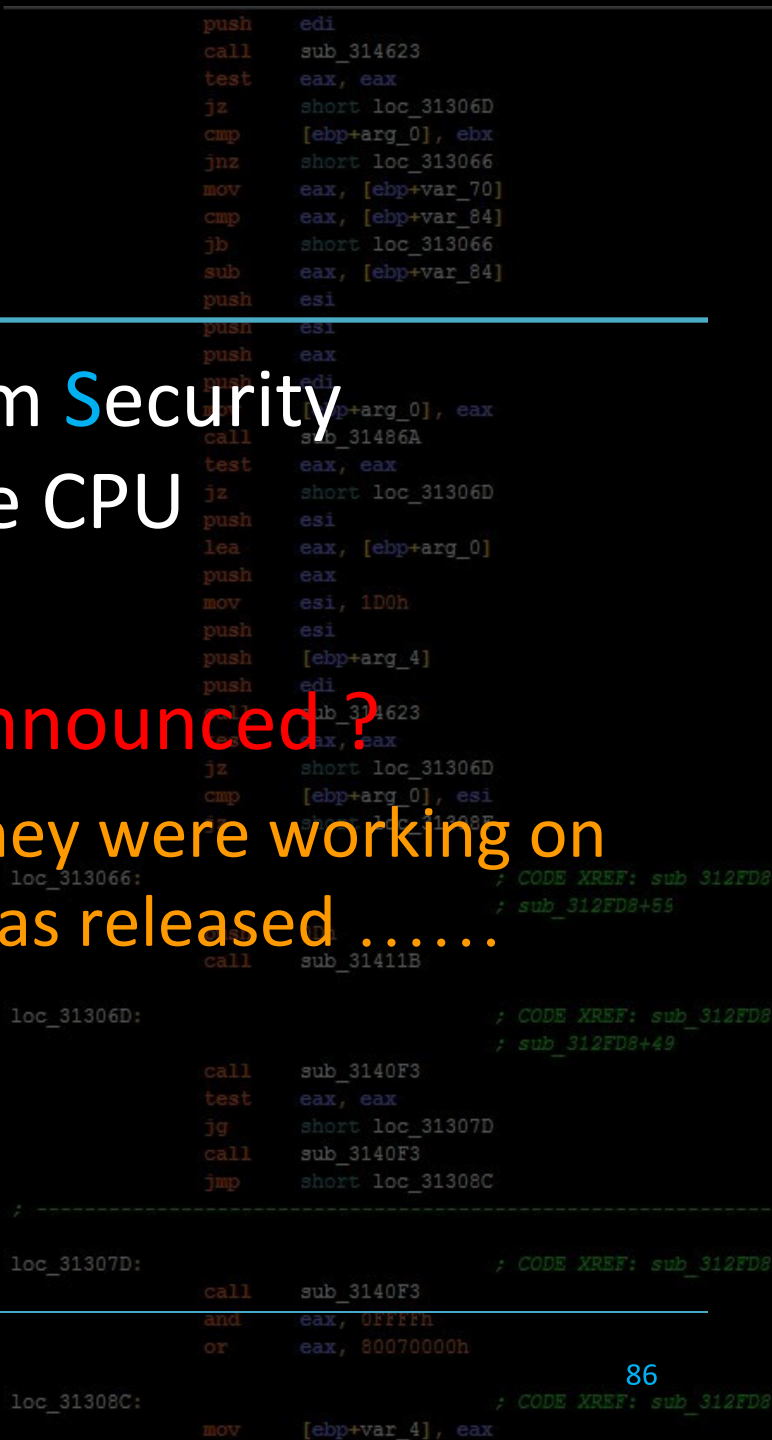
Xbox Virtual Disks

- Starting out, most things are in AES128 encrypted containers known as XVDs & XVCs
- Just like an encrypted virtual disk or zip, contains .exe’s, assets, directory structure, etc
- A lot of the security elements of the 360’s . XEX’s were inherited by the XVDs/XVCs

Xbox One PSP

- AMD snuck an ARM Platform Security Processor into the Xbox One CPU

- ? This was never formally announced ?
  - AMD only ever announced they were working on this technology, not that it was released ……



```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                        [ebp+arg_0], eax
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                mov     esi, 1D0h
                push    esi
                push    [ebp+arg_4]
                push    edi
                        sub_314623
                        eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Xbox One PSP

- Nobody can decrypt system files, updates, without the ‘green’ AES256 ODK

- Host OS queries the PSP for the green AES256 ODK key, PSP passes it to the Host OS for XVD decryption
  – It would be nice to get this key)-:

Xbox One Host OS

- Owning the system means owning the Host OS
- You can't effectively comb the Host OS for bugs if you can't decrypt its system files
- You need to own the Host OS to get access to the keys used to decrypt it
  - Chicken & the egg problem

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                [illegible]
                [illegible]
                call    sub_31486A
                test    eax, eax
                jz      short loc_31306D
                push    esi
                lea     eax, [ebp+arg_0]
                push    eax
                [illegible]
                [illegible]
                push    [ebp+arg_4]
                push    edi
                [illegible]
                [illegible]
                jz      short loc_31306D
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                [illegible]

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

PS4 – Nov. 2013

PS4 Details

- I really don’t know as much about the PS4 OS or its security features
- I do know that it has a very similar AMD CPU as the Xbox One
  - An ARM PSP is also present in the CPU

References, Readings, Talks

- https://www.youtube.com/watch?v=82vf0JQS1Sk
- http://www.securityfocus.com/archive/1/461489
- https://www.youtube.com/watch?v=XtDTNnEvlf8
- https://www.youtube.com/watch?v=uxjpmc8ZIxM
- http://beta.ivc.no/wiki/index.php/Xbox_360_King_Kong_Shader_Exploit
- http://free60.org/wiki/SMC_Hack
- http://pastebin.com/gDLyZ6DU
- http://www.ibm.com/developerworks/power/library/pa-cellsecurity/
- http://www.eurasia.nu/wiki/index.php/PS3_Glitch_Hack
- http://rdist.root.org/2010/01/27/how-the-ps3-hypervisor-was-hacked/
- https://www.youtube.com/watch?v=4loZGYqaZ7I
- http://www.ps3news.com/PS3-Hacks/Fail0verflow-27C3-PS3-Exploit-Hacker-Conference-2010-Highlights/
- http://www.ps3news.com/PS3-Dev/ps-jailbreak-ps3-exploit-reverse-engineering-is-detailed/
- http://www.ps3devwiki.com/ps3/Boot_Order
- http://3dbrew.org/