Enter SandMan

Nicolas RUFF / EADS-IW SE/CS
nicolas.ruff [à] eads.net

Matthieu SUICHE
msuiche [à] gmail.com - http://www.msuiche.net/

What is this all about?

- We analyzed the details of Windows “suspend to disk” feature
  - a.k.a. “hibernation”
- We wrote a C library to read *and* write the hibernation file
- We are going to show some applications in:
  - Offensive computing
  - Defensive computing
  - Forensics

What is hibernation?

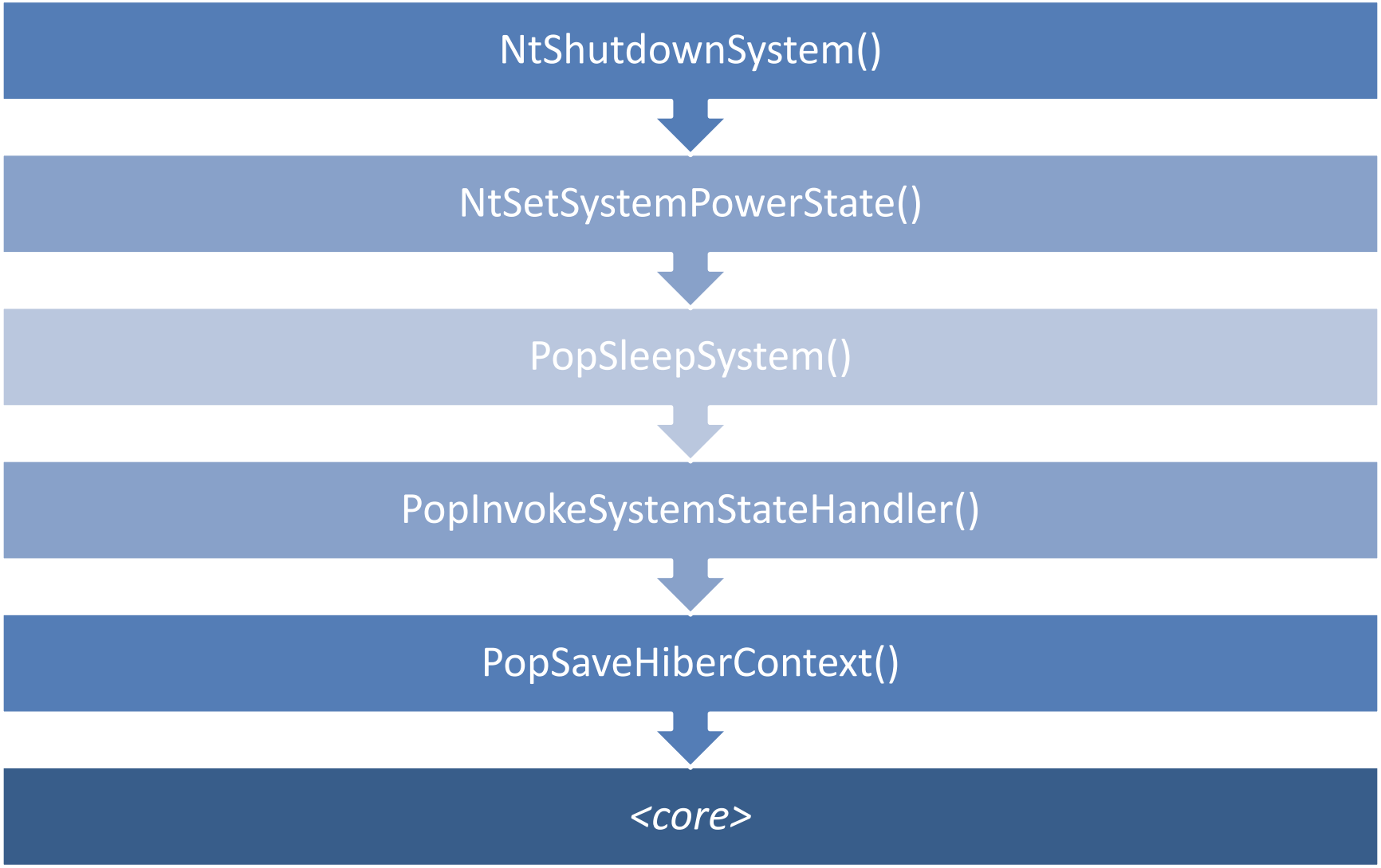
- Microsoft name for “suspend to disk” feature
  - Available on all modern OS
- System state is fully backed up on disk
  - Includes memory and processor state
  - “Zero power” sleep mode
- Available since Windows 2000
- Command-line controls:
  - POWERCFG /HIBERNATE
  - SHUTDOWN /H

The challenges of debugging

- Power transitions are challenging to debug
  - Entering hibernation state is not debugger-friendly
    - KdDeleteAllBreakpoints() and the like
  - Resuming from hibernation state is done by NTLDR
    - *Before* debugger attach hook (see [1])

- Fortunately, the whole thing is quite small
  - “Dead listing” analysis is possible
  - Call chain from entry point:
    - NtShutdownSystem()
    - NtSetSystemPowerState()
    - PopSleepSystem()
    - PopInvokeSystemStateHandler()
    - PopSaveHiberContext()
    - *<core processing>*

Call chain from entry point



File format

Field	Content
Header	PO_MEMORY_IMAGE structure
Page list	Not sure – might be a list of “free pages” for loader use
Processor State	CONTEXT + SPECIAL_REGISTERS structures
Memory Range Array #1	*Header:* list entries count + next list offset + checksum *List:* Up to 255 entries *List entry:* start page + end page + checksum
Xpress Blocks Array #1	*Magic:* “\x81\x81xpress” (Windows > 2000) *Header:* size + checksum + other *Content:* compressed data
Memory Range Array #2	(…)

File format - details

- Header
  - PO_MEMORY_IMAGE is exported in debugging symbols
    - However, this structure *does* change across Windows versions
  - Magic bytes can be:
    - hibr: hibernation file is valid, system shall be resumed on boot
      - Vista makes use of caps (HIBR)
    - wake: hibernation file is invalid, system shall be start anew
    - link: supported, but never seen to date

File format - details

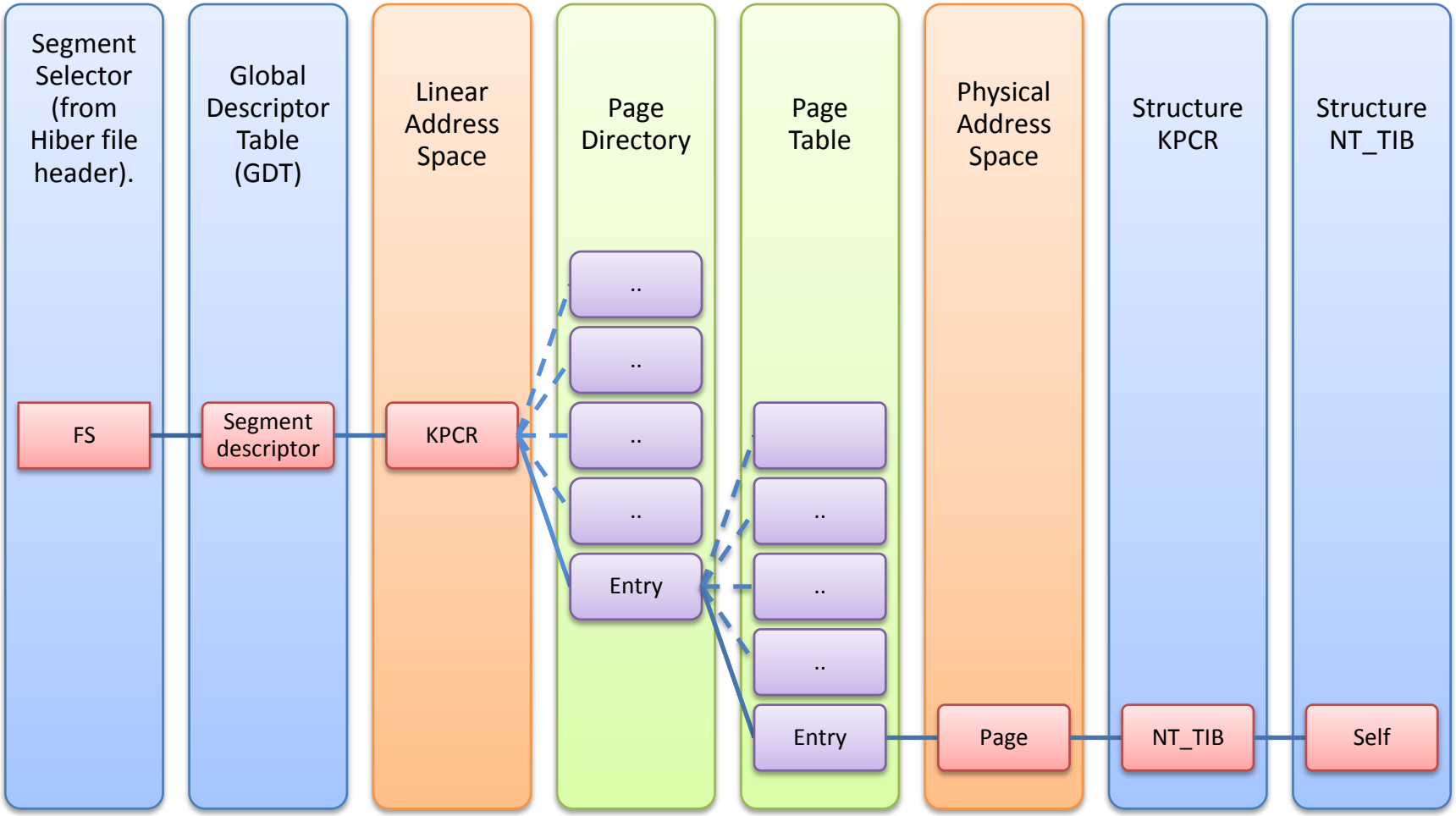
- Memory Range Array
  - List count is stored on 32 bits, but count is always 0xFF (except the last range)
  - Pages are *not* ordered
- Xpress blocks
  - 1 Xpress block = 0x10 physical pages (except the last block)
  - Windows 2000: compressed using RtlCompressBuffer()
    - Compression method called “LZNT1” internally
    - Other compression methods are available, but not used
      - We support them, though ☺
  - Windows > 2000: compressed using internal function XpressEncode()
    - RtlCompressBuffer(COMPRESSION_ENGINE_HIBER) has never been used

File format - details

- For each array:
  - ∑ (pages in Memory Range) == ∑ (pages in Xpress Blocks)
  - Xpress Blocks are holding 0x10 pages each
- Other random notes:
  - Most checksums are set to zero (Windows > 2000)
  - Everything is page-aligned
    - 1 page = 0x1000 bytes
  - OS fingerprinting is possible using slight variations
    - Required because of PO_MEMORY_IMAGE header changes
    - Required for selecting compression method (Windows 2000)
  - Special memory layout (/PAE) can be detected through configuration registers

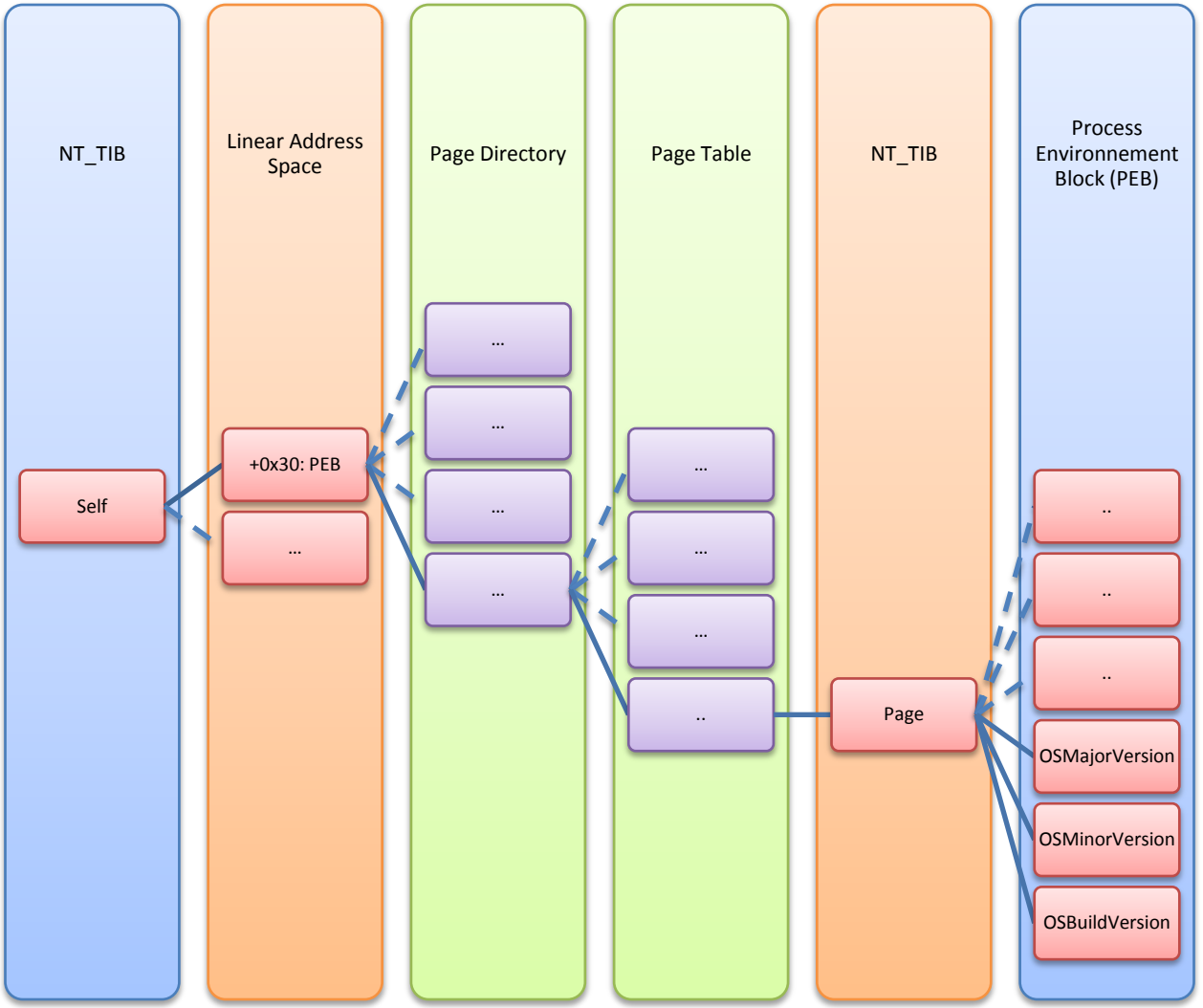
Method #1 : OS Finger-Printing (1/2)

(simplified – withouth PAE)



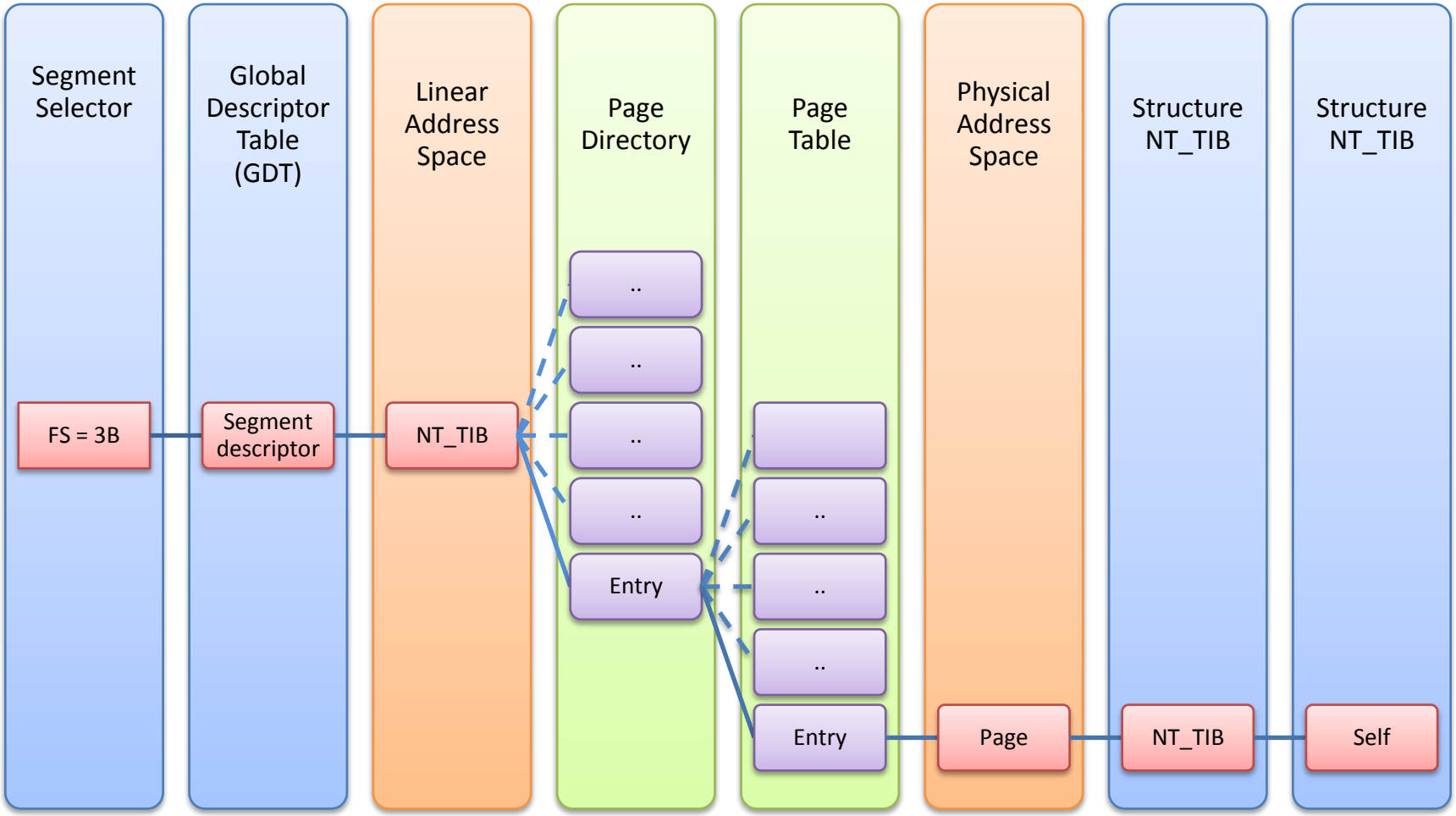
Method #1 : OS Finger-Printing (2/2)

(simplified)



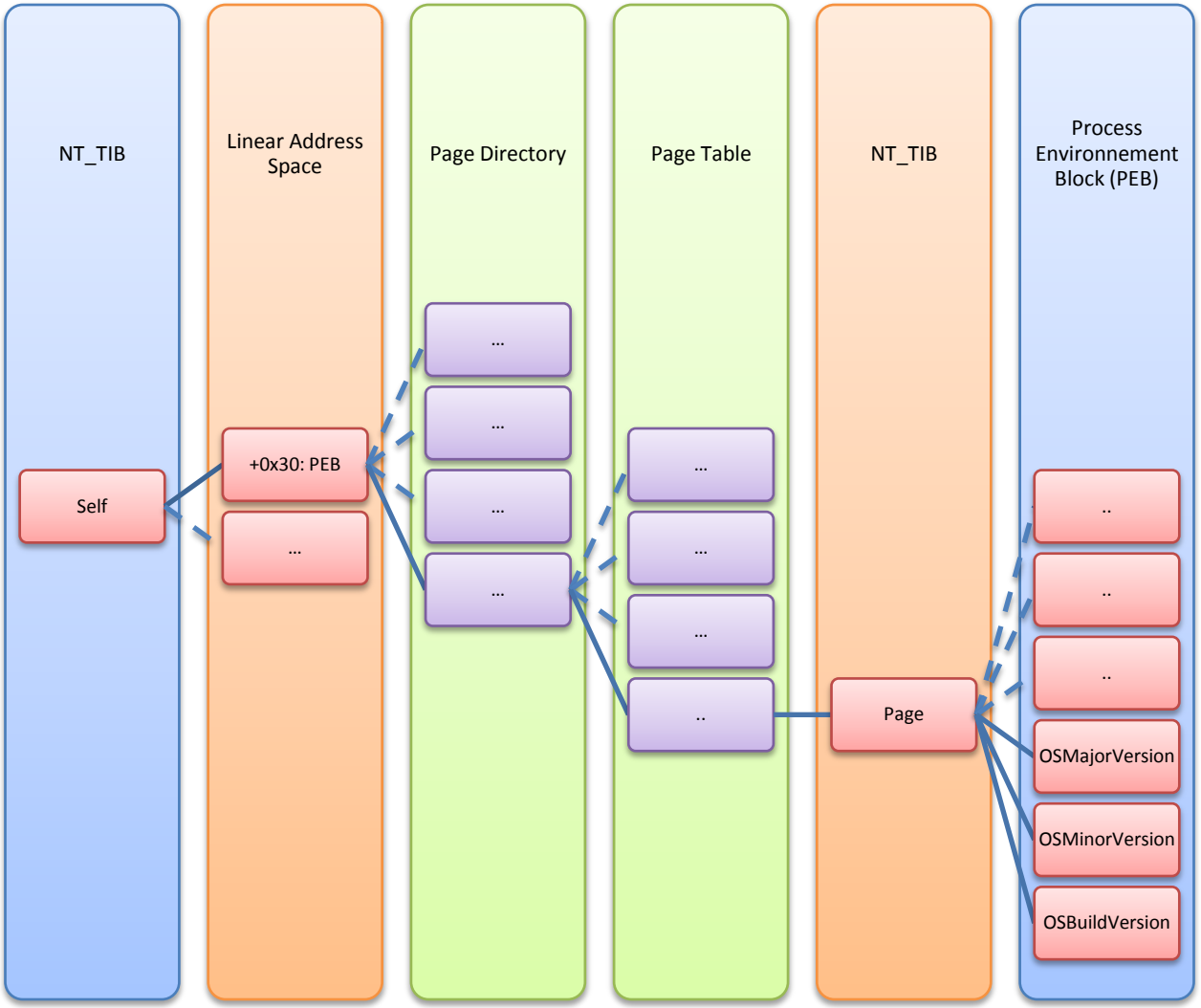
Method #2 : OS Finger-Printing (1/2)

(simplified – Windows > 200 only - without PAE)



Method #2 : OS Finger-Printing (2/2)

(simplified – Windows > 2000 only - without PAE)



SandMan library

- Wish list for the SandMan library:
  - Ability to parse any hibernation file, regardless of Windows version
    - Including 32/64-bit
  - Well-documented library
  - Python binding
  - Cool sample apps
    - Convert to “dd”-style file
    - Locate and patch pages (w/ fast lookup)
    - Append pages
    - Screenshot
    - Nice GUI

- Let’s see what we have today ☺
  - DEMO

Use cases

- Offensive computing
  - Patching a sleeping machine
    - Target #1 : nt!SeAccessCheck()
    - Target #2 : msv1_0!MsvpPasswordValidate
  - Data extraction
    - Everything is on disk, including non-paged pool
    - Removing PatchGuard or loading unsigned drivers, anyone ? ☺

Use cases

- Defensive computing
  - Malware detection
    - We could think of no code hiding technique
      - If it is not in the hibernation file, it will not resume execution
      - You have to trust your hardware, though
    - Bonus question: what happens to hypervisor pages during hibernation … ?
      - No, I won't use the "Blue Pill" buzzword in those slides ☺

Use cases

- Forensics through hibernation
  - Live memory analysis is of growing interest since DFRWS 2005
    - Sample offensive work:
      - Meterpreter (Metasploit project)
      - Syscall Proxying (CORE Impact)
    - Sample defensive work:
      - PTFinder (Andreas Schuster)
      - MemParser (Chris Betz)
      - Windows Memory Forensics Toolkit (Mariusz Burdach)
      - PMODump (TRUMAN project)
      - FATkit (4tphi)
      - Volatools (Komoku)
      - Volatility (Volatile Systems)
      - Oracle memory analysis (Black Hat 2007)
      - Etc.

Use cases

- Forensics through hibernation
  - Memory collection is still a challenge
    - Hardware techniques have strong prerequisites
      - Dedicated PCI hardware
      - IEEE 1394 bus
    - Software techniques are Windows version dependant
      - dd “\Device\PhysicalMemory”
      - ZwSystemDebugControl
      - Driver load
    - Most common technique “in the field” is: BSoD + full memory dump

Use cases

- Forensics through hibernation
  - Pros:
    - No hardware prerequisite
    - Coherent system state
      - Atomic hibernation + pagefile acquisition
    - Machine activity can be resumed seamlessly
    - Hibernation can be activated without reboot
    - Hibernation file can be converted to “dd-style” memory dump
    - Processor context (including CR3 register) is readily available
  - Cons:
    - No guarantee that 100% of physical memory has been saved

Use cases

- Forensics on hibernation file
  - Hibernation file is never wiped out
    - First page is filled with 0's
      - » It does not prevent full memory reconstruction
    - Can hold sensitive data (passwords and keys)
  - There is slack space in hibernation files
    - Hibernation file is preallocated to physical memory size
    - Effective use depends on physical memory use
    - Xpress Blocks can be extracted, given the “xpress” header

Conclusion

- Hibernation file is really a low-hanging fruit
  - Available since Windows 2000
  - Not really hard to understand
- It has a potential for (ab)use
  - “The Ultimate LiveKD”
  - Rootkit detection
  - Live memory forensics
- I hope you enjoyed the show

(Few) references

- “How Windows Starts Up” (part 2)
  - http://blogs.msdn.com/ntdebugging/archive/2007/06/28/how-windows-starts-up-part-the-second.aspx
- Nirsoft: Vista Kernel Structures
  - http://www.nirsoft.net/kernel_struct/vista/
- Processes, Threads, VirtualMemory
  - http://www.i.u-tokyo.ac.jp/edu/training/ss/msprojects/data/07-ProcessesThreadsVM.ppt
- Intel - Volume 3 SYSTEM PROGRAMMING GUIDE
  - Chapter 3 - Protected Mode Memory Management
- Microsoft Vista SP1 Overview
  - http://download.microsoft.com/download/9/0/d/90da9663-815a-4ce8-88c0-2b9f54c69efe/windows%20vista%20service%20pack%201%20beta%20overview.pdf