Protection & The Intel Pentium Architecture

By Sherri Sparks

Presentation Overview

- Hardware Protection
  - The Intel Pentium Processor Architecture
- Software Protection On The Intel Platform
  - The Windows Operating System
- Case Studies: Vulnerabilities
  - The F00F Bug
  - Windows 2000 Privilege Escalation Using Callgates
  - FU: A Kernel Rootkit

Types Of Protection Supported By The Pentium Architecture

- Segment Level Protection
- Page Level Protection
- Combined Segment & Page Level Protections

Memory Segmentation (1)

- Segmentation model may be either Flat or Multi-segmented.
  - Flat: Does not partition the address space. Segment size is the size of the entire linear address space.
  - Multi-segmented: Partitions the address space with segments of different sizes.
- Segments are defined in descriptors within the global descriptor table (GDT).
  - Code Segments
  - Data Segments
  - System Segments (LDT, TSS)

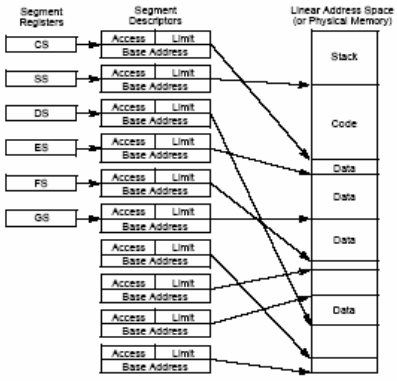
Memory Segmentation (2)

- **selector: offset** addressing format.
- 6 segment registers are provided to store segment selectors (cs, ds, ss, es, fs, gs)
- Example
  - mov eax, ds:80040000

Segment selector **Offset**

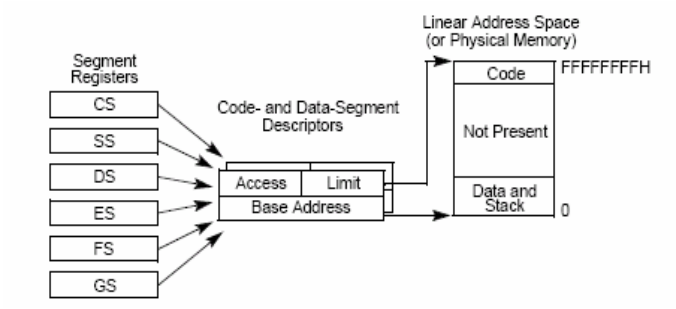
Memory Segmentation (3)

- Multi-segment model



Memory Segmentation (4)

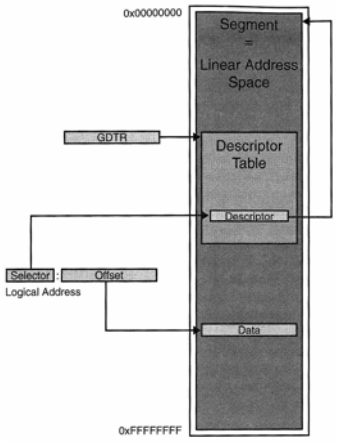
- Flat model



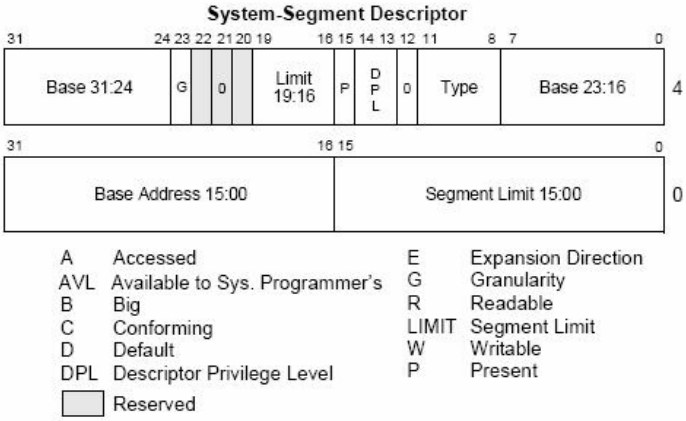
Memory Segmentation (5)

- Flat memory model addressing

1. Look up descriptor in GDT by using the selector as an index.
2. Use the segment base in the descriptor + the offset to locate the logical address.



Segment Descriptor



Segment Level Protection

- All memory references are verified in parallel during address translation.
- Protection violations result in an exception.
- 5 Checks
  - Type check
  - Limit check
  - Restriction of addressable domain
  - Restriction of procedure entry points
  - Restriction of instruction set

Type Check

- Controls access to a segment based upon its characteristics (readable, writable).
- The **Writable** bit in a data segment descriptor controls weather programs can write to the segment.
- The **Readable** bit in a code segment descriptor specifies weather programs can read from the segment.

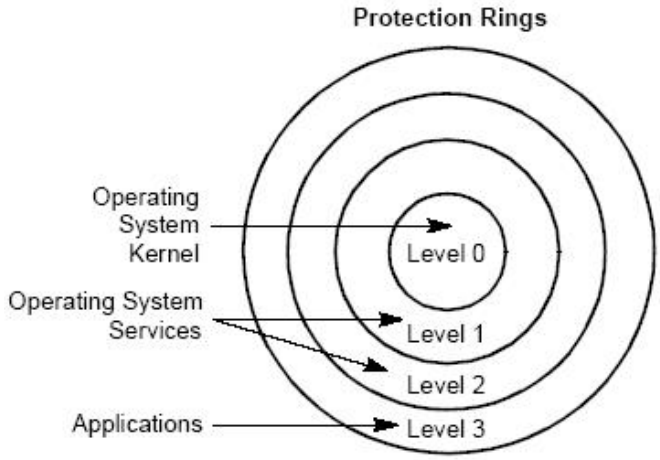
Limit Check

- A check on the **Limit** field in a segment descriptor prevents programs from addressing outside the segment.
- An attempt to access the memory outside the segment causes a general protection exception.
- Catches pointer errors so they don't overwrite memory in another module.

Restriction Of Addressable Domain: Privilege Levels

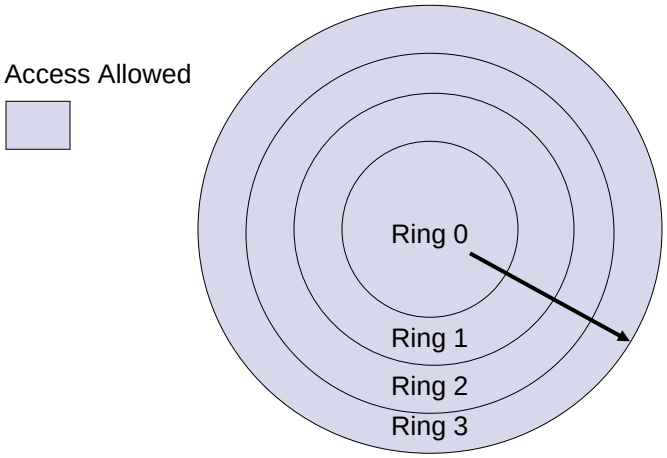
- 4 Processor privilege levels (0-3)
  - Level 0 – highest privilege
  - Level 3 – lowest privilege
- Code executing lower privilege levels (other than 0) are restricted with regards to memory and instruction set access.
- A general protection exception results if a program existing in a lesser privileged segment attempts to access memory in a more privileged segment.

Privilege Levels As Rings Of Protection



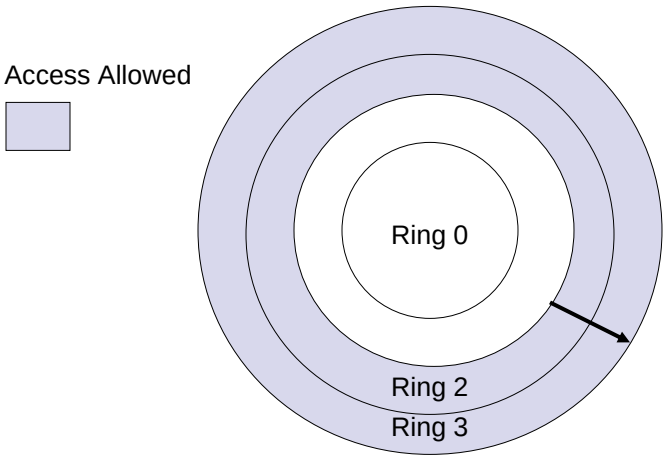
Access Restriction Using Rings: Ring 0

Access is allowed outward to higher numbered rings. Ring 0, therefore, can access rings 0 through 3



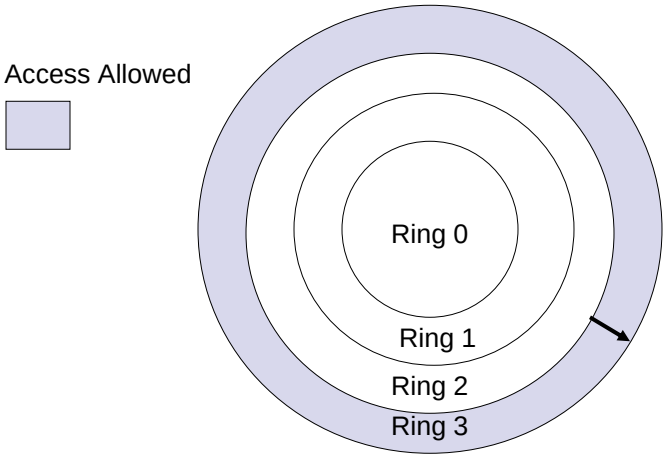
Access Restriction Using Rings: Ring 2

Access is allowed outward to higher numbered rings. Ring 2, therefore, can access rings 2 and 3.



Access Restriction Using Rings: Ring 3

Access is allowed outward to higher numbered rings. Ring 3, therefore, can access ring 3.

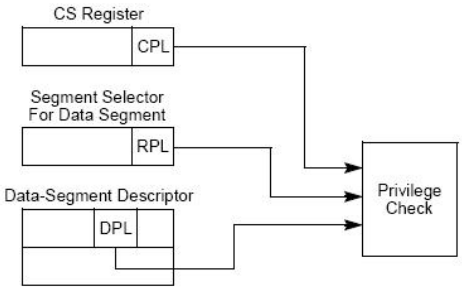


Some Terminology…

- **CPL = "Current Privilege Level"**
  - The privilege level of the program currently being executed.
  - Encoded in the code segment selector contained in the cs segment register.
- **DPL = "Descriptor Privilege Level"**
  - The privilege of a given memory segment).
  - Given by a field in the segment descriptor.
- **RPL = "Requestor Privilege Level"**
  - The privilege level of a memory request. RPL is encoded in the segment selector portion of memory operands.
  - Encoded in the selector portion of a selector:offset address.

Using Privilege To Restrict Access To Data

- An instruction may load a segment register only if the DPL of the segment is less than or equal than the CPL and the selector's RPL.
- Addressable domain changes based on CPL.
  - At CPL=0, all data segments are accessible regardless of privilege level.
  - At CPL=3, only privilege level 3 data segments are accessible.

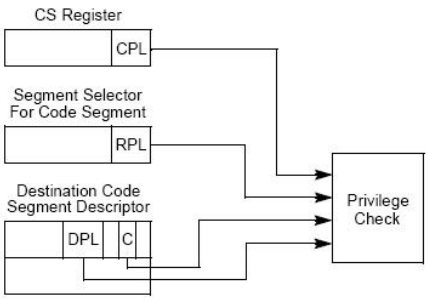


The Effect Of Privilege Level On Control Transfer Instructions

- Control transfers are provided by jmp and call instructions.
- Near Transfers
  - ***Intrasegment*** jmps and calls
  - Subject to limit checking with respect to the current code segment.
- Far Transfers
  - ***Intersegment*** jmps and calls
    - The operand selects the descriptor of another executable segment.
    - The operand selects a call gate descriptor.
  - Subject to privilege checking.

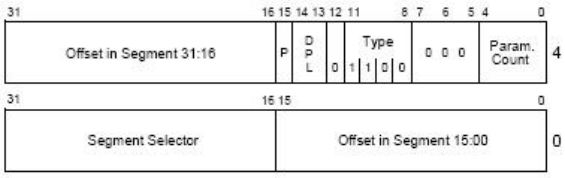
Restriction Of Procedure Entry Points (Non Call Gate)

- Control can only be transferred to other segments of the same privilege (CPL=DPL of the destination segment).
- If the segment is a conforming segment its DPL may be more privileged than the CPL (ie. the privilege it runs at).
- NOTE: a conforming segment runs at the privilege level of the calling procedure regardless of the DPL in its segment descriptor.



Restriction Of Procedure Entry Points (Call Gate)

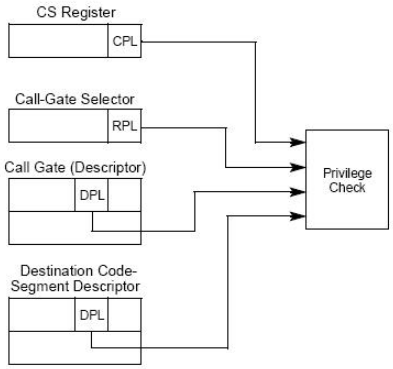
- Used to transfer control between different privilege levels.
- 2 Main functions
  - To define any entry point of a procedure.
  - To specify the privilege level required to enter a procedure.
- *Selector* & *Offset* fields form a pointer to a procedure entry point.



DPL Descriptor Privilege Level
P Gate Valid

Restriction Of Procedure Entry Points (Call Gate)

- 4 Privilege Checks for callgate control transfer.
  - Current privilege level (CPL).
  - The RPL of the segment selector used to specify the call gate.
  - The DPL of the gate descriptor (determines at which privilege levels the gate may be used).
  - The DPL of the segment descriptor of the destination code segment.



Restriction Of Instruction Set

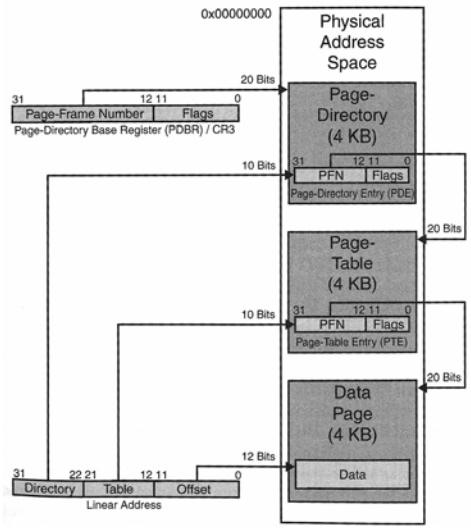
- Privileged instructions
  - Used for system control
  - Can only be executed at ring 0 (CPL = 0)
  - Generates a general protection exception if executed at CPL != 0.

- LGDT—Load GDT register.
- LLDT—Load LDT register.
- LTR—Load task register.
- LIDT—Load IDT register.
- MOV (control registers)—Load and store control registers.
- LMSW—Load machine status word.
- CLTS—Clear task-switched flag in register CR0.
- MOV (debug registers)—Load and store debug registers.
- INVD—Invalidate cache, without writeback.
- WBINVD—Invalidate cache, with writeback.
- INVLPG—Invalidate TLB entry.
- HLT—Halt processor.
- RDMSR—Read Model-Specific Registers.
- WRMSR—Write Model-Specific Registers.
- RDPMC—Read Performance-Monitoring Counter.
- RDTSC—Read Time-Stamp Counter.

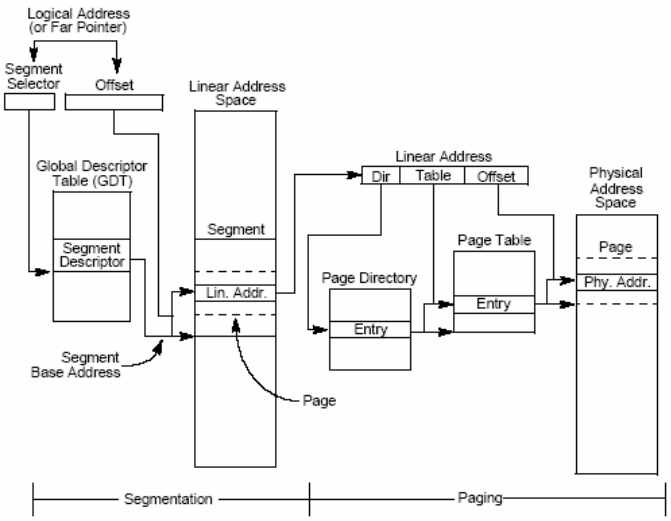
Memory Paging (1)

- 2 level paging architecture.
  - 4K page directory consisting of 32 bit page directory entries (PDEs) (aka pointers to page tables)
  - Page tables consist of 32-bit page-table entries (PTEs) (page frame mappings).
  - The address of the page directory is stored in control register CR3.

Memory Paging (2)

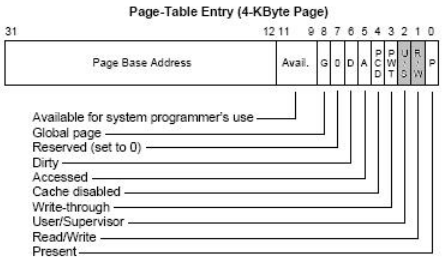


Segmentation With Paging



Page Level Protection

- 2 Page Level Protection Checks
  - Restriction of Addressable Domain (privilege check).
    - Supervisor Level: U/S = 0
    - User Level: U/S = 1
  - Type Checking
    - Read-only page access (R/W = 0)
    - Read / write access (R/W = 1)
- General exception protection generated on violation of checks.

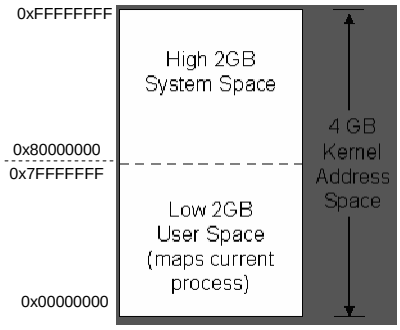


Combining Page & Segment Protection

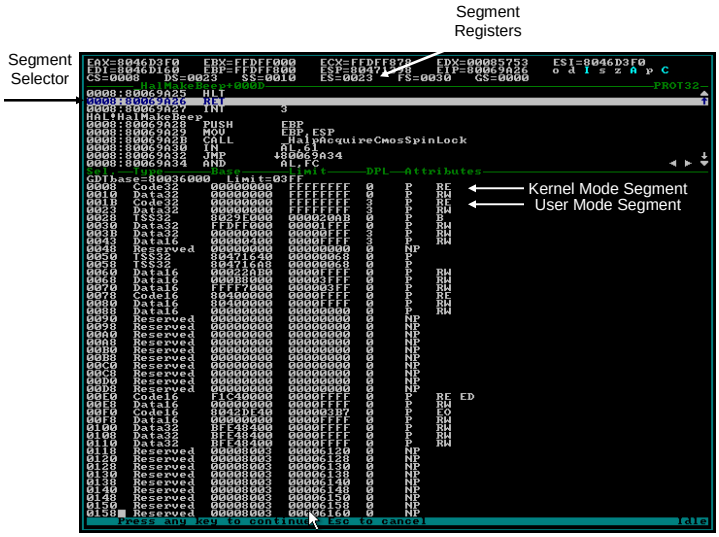
- When paging is enabled, processor first evaluates segment protection, then evaluates page protection.
- Possible to define a large flat memory space consisting of 1 segment with some portions that are read-only and other portions that are read-write. Intersegment partitioning of read / write protection is provided by page level protection (R/W bits in the page table entry).

Windows & Protection: User & Kernel Mode (1)

- Combines page and segment protection.
- Defines a “flat” address space where all code and data can be addressed by 32 bit pointers.
- Only uses 2 rings (0 & 3)
  - Kernel Mode - Pages from 0x80000000-FFFFFFFF have the supervisor bit set.
    - OS kernel
    - Drivers
  - User Mode – Pages from 0x00000000-7FFFFFFF have the supervisor bit cleared and execute at ring 3, or user mode.
    - OS support subsystems
    - User Applications



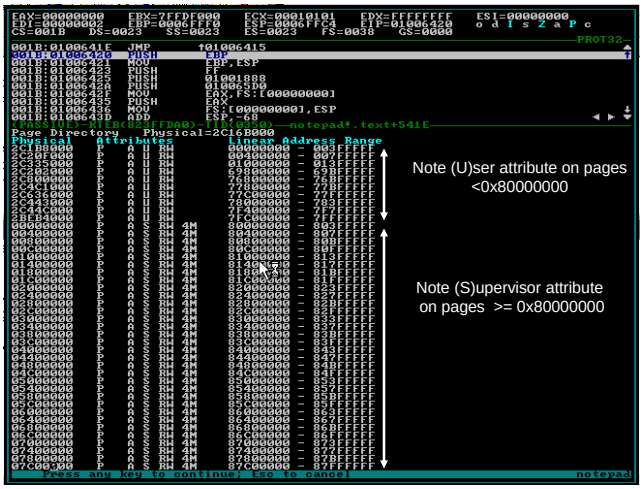
Windows & Protection: User & Kernel Mode (2) – The GDT



Windows & Protection: User & Kernel Mode (3) – The Page Directory

Process separation is provided by giving each process its own page directory entry and corresponding set of page tables that address the full 4 GB virtual address space.

The control register (CR3) maintains a pointer to the page directory of the currently executing process. It is reloaded by the OS during context switches.



Protection & Compromise

- Case Studies: Vulnerabilities
  - The F00F Bug
  - Windows 2000 Privilege Escalation Using Callgates
  - FU: A Kernel Rootkit

Case Study: The F00F Bug

- An invalid encoding of the LOCK CMPXCHG8B EAX instruction does not generate the expected “undefined opcode” exception and irrecoverably freezes the processor.
- Creates a denial of service vulnerability. Since this is not a privileged instruction, any user can create a program with this invalid encoding and bring down any Pentium based machine that executes it.

Case Study: Windows 2000 Privilege Escalation Using Call Gates

- Allows a ring 3 program to run code at ring 0.
- Takes advantage of the fact that the /Device/Physical memory object is accessible from ring 3.
- This makes it possible to locate the **physical** address of the GDT in memory and write a new call gate descriptor into **physical memory**.
- If the DPL of the new descriptor was set to 0, the function that it points to will run at ring 0 when called by a user application.

Case Study: FU – A Kernel Rootkit

- Exploits the lack of separation between the OS and user drivers.
  - Drivers have equal privilege to the OS kernel (ie. full access to memory and the instruction set).
  - Possible for a malicious driver to modify the OS itself.
    - FU Rootkit
    - Uses DKOM (Direct Kernel Object Manipulation).
    - Modifies kernel data structures in memory to avoid detection.
      - Example: Processes can be hidden by unlinking process objects from the kernel's internal doubly linked process list.

Protection Is A Joint Effort

- Hardware Level
  - The protection mechanisms provided by the processor architecture.
- Software Level
  - The Operating System implementation of hardware protection mechanisms.
- Vulnerabilities at either level may compromise system protection!!!

References

- The Intel Architecture Software Developer's Manual: Part 3 - Chapter 12
- Undocumented Windows 2000 Secrets by Sven B. Schriber
- Undocumented Windows NT by : Prasad Dabak, Milind Borate, and Sandeep Phadke
- FU Roootkit http://www.rootkit.com
- Dr. Dobbs Journal: The Pentium F00F Bug Description & Workaround http://www.x86.org/errata/dec97/f00fbug.htm

?