Computer Security at the Hardware/Memory/CPU Level

Nicolas T. Courtois
- University College London

Where Does Computer Security Come From?

Comes From

Trusted Computer Base???? Or TCB?

Do You Know…

Q1.

Can in Windows/Linux a process run by an administrator access the system/kernel memory?

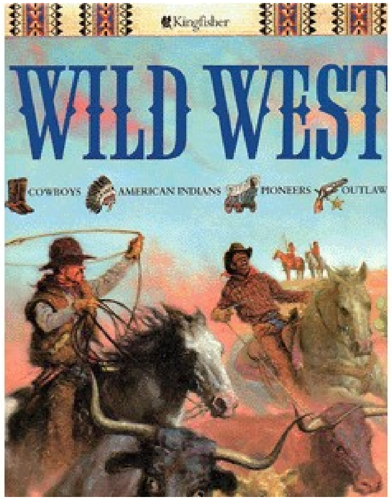
Q2.

Why do we must press Ctrl+Alt+Del when we log to a PC under many versions of Windows?

Q3.

Why more than half of large banks in London bought PCs with a pre-installed hardware Rootkit?

Software Security?



Bottom Line

Software can hardly be protected by software.

- TRUE.

=> <u>some</u> hardware mechanisms are needed.

Taxonomy of Software Threats [Microsoft]

Spoofing = pretending to be someone else

Tampering = altering data or settings

Repudiation = user denies it was him that did of didn’t do sth.

Information disclosure = leak of personal information

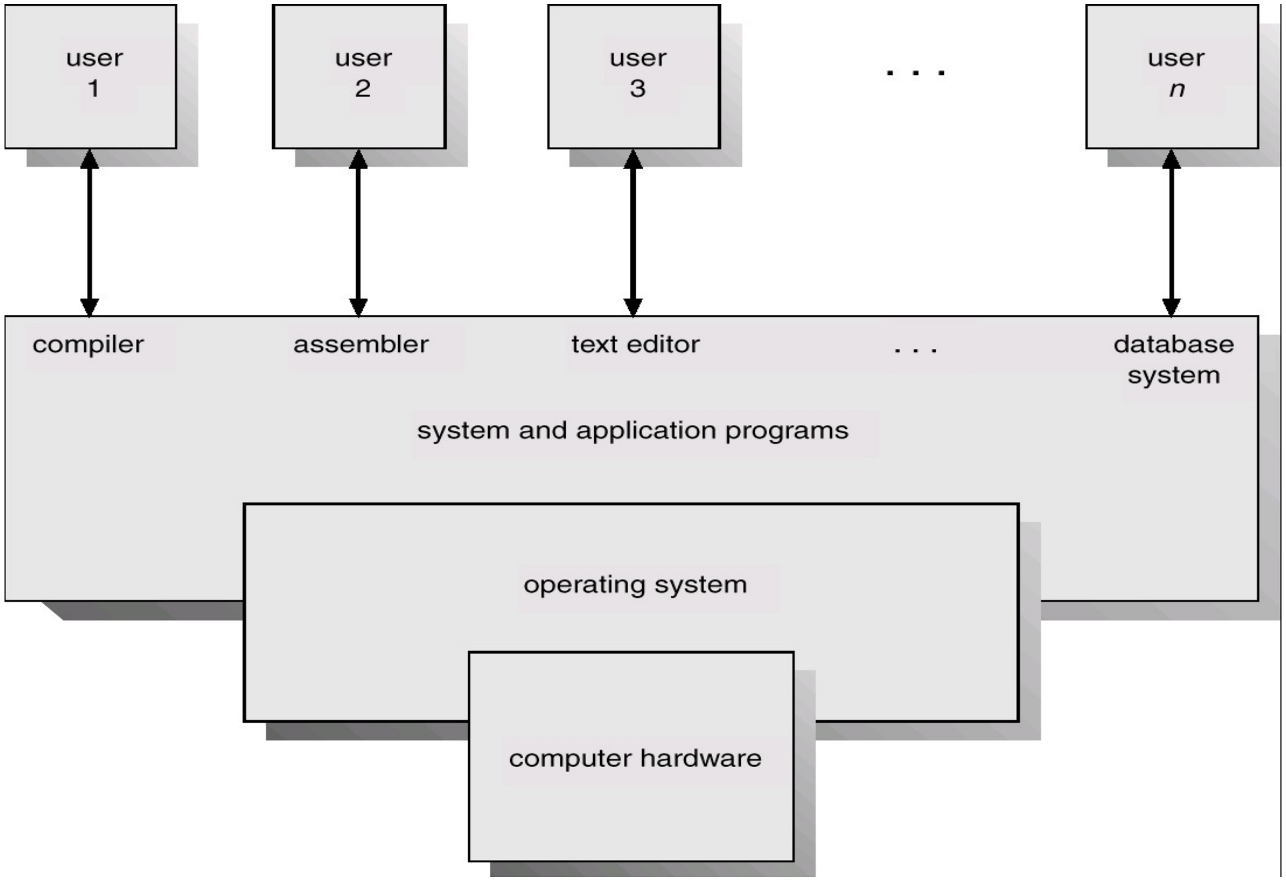
Denial of service = preventing normal operation

Elevation of privilege = e.g. gaining the powers of root

History

PCs are open source platform based on a set of open industry-wide standards. => Only very recently there is an evolution towards more closed source, and more fragmentation (competing standards)

Standard PC


user
1
user
2
user
3
. . .
user
n
compiler
assembler
text editor
. . .
database
system
system and application programs
operating system
computer hardware

Least Privilege [or Limitation] Principle

Every “module” (such as a process, a user or a program) should be able to access only such information and resources that are necessary to its legitimate purpose.

Main Security Goals For the OS+Hardware

(Goal 0.)

reliable operation and business

Goal 1A.

allowing multiple users securely share a computer.

Goal 1B.

allowing multiple processes securely share a computer.

(Goal 2.)

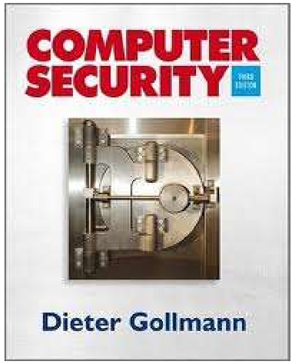
allow secure operation in a networked environment.

Goal 1 – Means to Achieve It

Goal 1A+B. multiple users / processes securely sharing a computer.

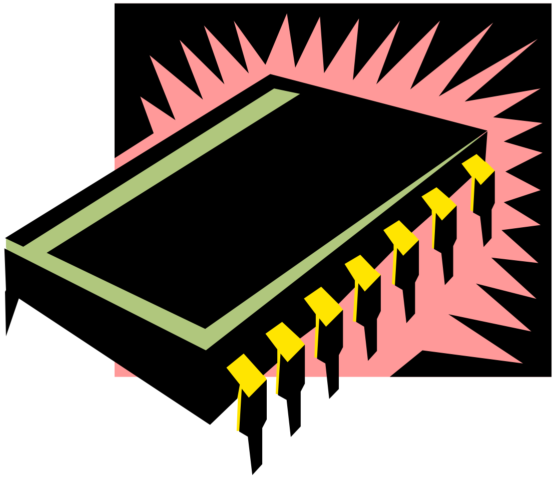
- authentication of users, cf. part 05
- file access control and (drive/file) encryption and auth.

- memory protection
  - Chapter 6.4.
  - (possibly memory encryption)
- processor modes
  - Cf. Chapter 6.3.5.

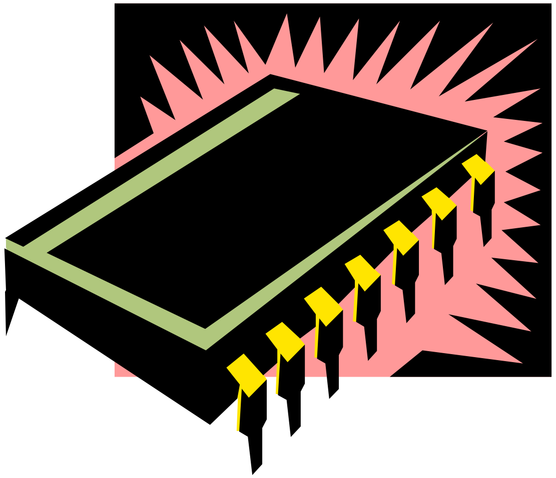


- logging & auditing

Basic Hardware Mechanisms



Memory Protection



Memory Protection

Allows to implement memory access rights for processes.

- Goal: One process should not access other's memory.

- Prerequisite: Operating system and user processes need to have different privileges

Kernel space vs. User space

- Kernel space: the OS kernel, some kernel extensions, some device drivers
  - they run in the most privileged CPU mode = system mode = ring 0.
  - typically cannot be swapped to disk
- User space, Userland: other parts of the OS that

  run as processes or services/daemons in the user mode.
  - I/O and components
  - manipulating the filesystem
  - shell

  Windows:
  - system processes will be running in the "user space" but as user=system, so user space is a confusing name!

  Depending on OS we can have system = root = super-user = administrator or all these will be distinct...

Memory Protection History

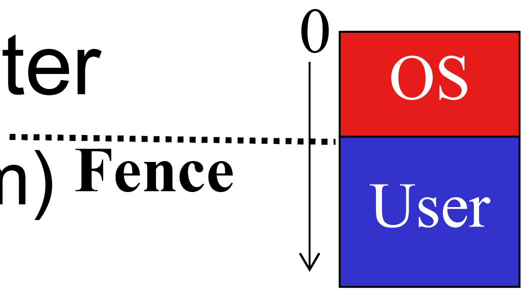
- In Windows 9x, user processes COULD access system/kernel memory.
    - 386 CPUs would allow the separation…
        - capacity not used!

- Protection exists in:
    - Most UNIX and Linux systems
    - Mac OS X [2001]
    - Windows NT since NT3.1.
    - Windows XP [2001] and ever since.

1) Pre-history of Memory Protection

1.1. Fence Register

– (OS+1 program)



Problems:

-multi-user

-multi-tasking

-flexibility

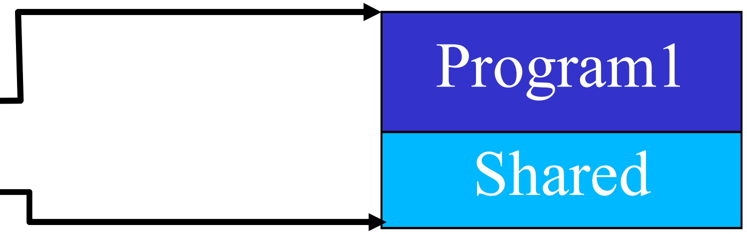
-sharing

Runtime checks by CPU

1.2. Base

\+ Bound

– (OS + several programs)



2 registers for each program

-flexibility

later base+bound for each segment but still not good enough

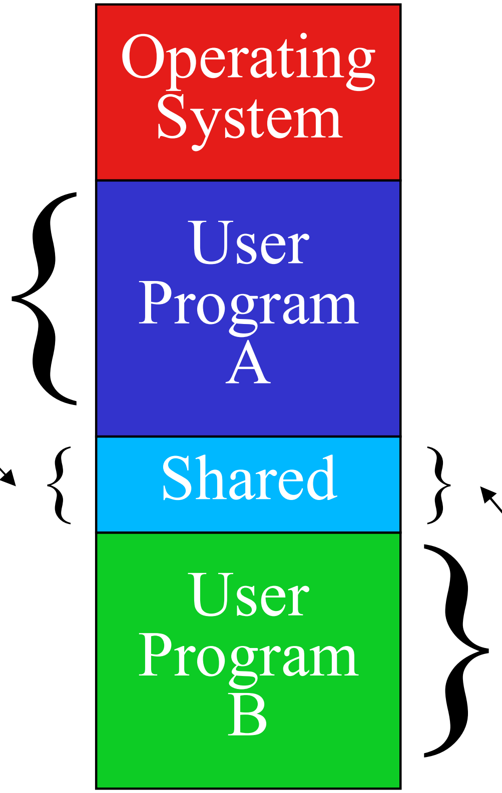
-performance

2.0.) Segments: Sharing & rwx

segment size >64 K

different table in each program

Seg	Base	Bnd	Prot
0	a	n	rwx
1	n+1	m	rw



Seg	Base	Bnd	Prot
0	m+1	high	rwx
1	n+1	m	r

Working With Segments?

Q1: Are we inside the segment?

Q2: Do we have the right to write?

– checked at runtime
– costly checks at each memory access

And how do we manage all this?

Memory Management

Methods further evolved into having:

- A hardware memory management unit (MMU) and a lot of special circuits in the CPU and chipset.
- A lot of support functions done by the OS.
- A more abstract view where the programmer and the compiler would see a simple linear address space. Will be achieved with paging…

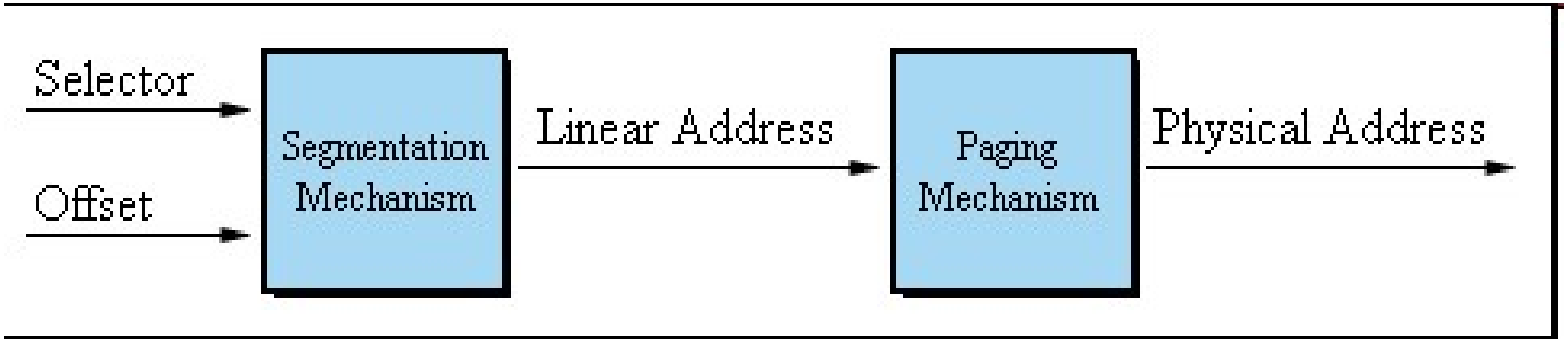
Modern Memory Protection Methods

1. Segmentation: used until recently, no longer used
2. Paging: the dominant method in 2012
3. Capability-based addressing
   - the closest to “least-privilege” ideal.

     But not used in Windows/Linux/MAC PCs.

2.1. Segmentation + Paging Combined

- Before 200X segmentation + paging worked together, e.g. Intel x32+early Windows XP



- However until very recently (even in say 2009) there was a big loophole in cheap commercial PCs:
  – no x protection at page level, only at segment level

Virtual Memory History

- This architecture exists since i386.
- Later CPUs added more performance and more complexity

  +special modes for legacy code…

Segments - Security

- Each segment had permissions R/W/X.
- If the program uses pointer such that
  - it would jump to a segment for which has no X right
  - it would read memory for which he has no R right
  - the offset is outside the allowed range,
  - all this is checked by the CPU at runtime
    - with help of MMU = memory management unit

  ⇒ a HARDWARE exception 0xEh is raised

  ⇒ will be handled by OS Kernel

Paging

Virtual Memory, typically 2,3 or 4 Gbytes for a 32-bit process in Windows.

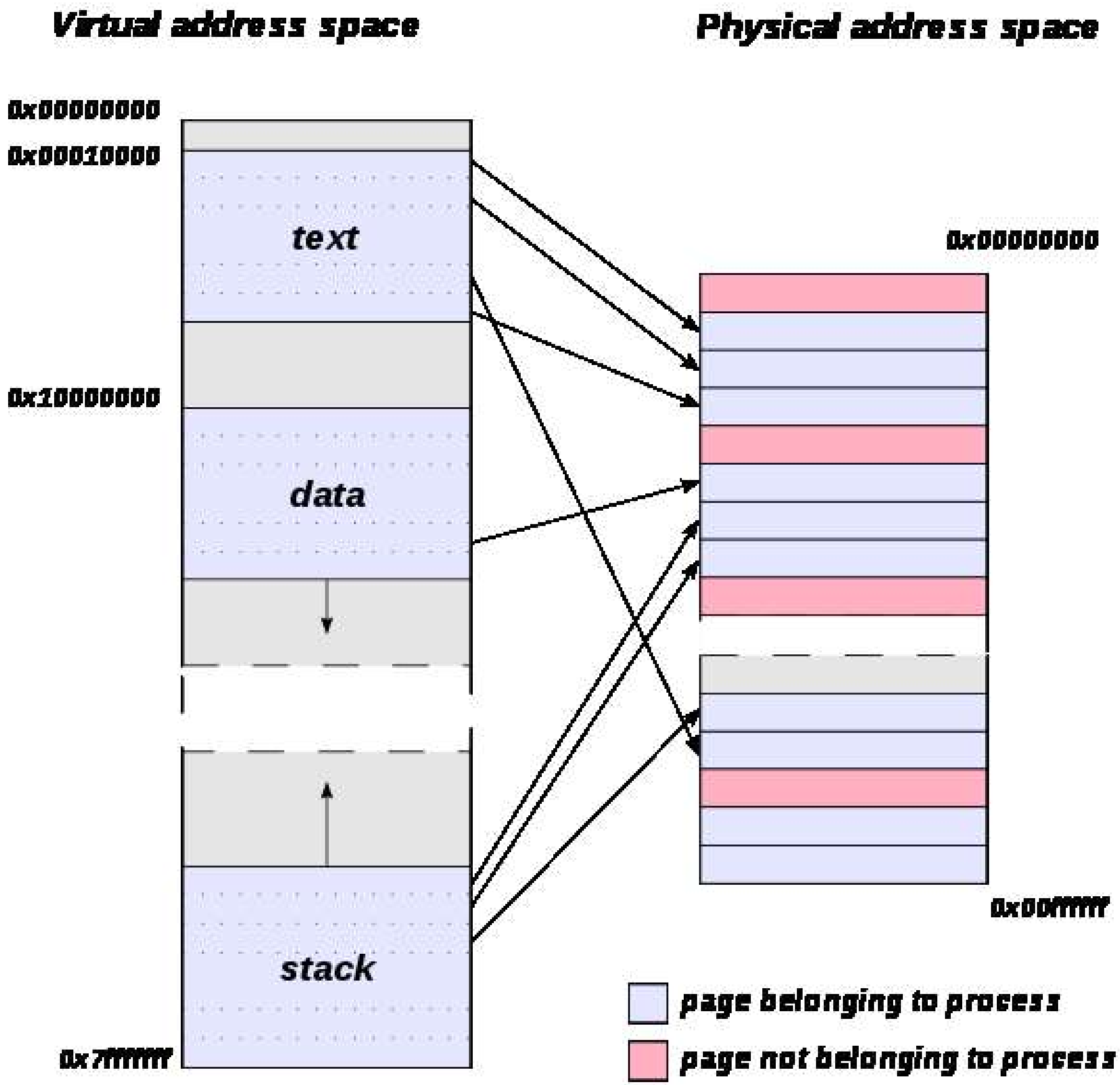
Each block is mapped either

- somewhere into memory
- or there is a page fault (OS handles it)
  - in the swap (security risk on its own!)
  - or not used.

  Some security: a page not previously used, can automatically generate a page fault error.

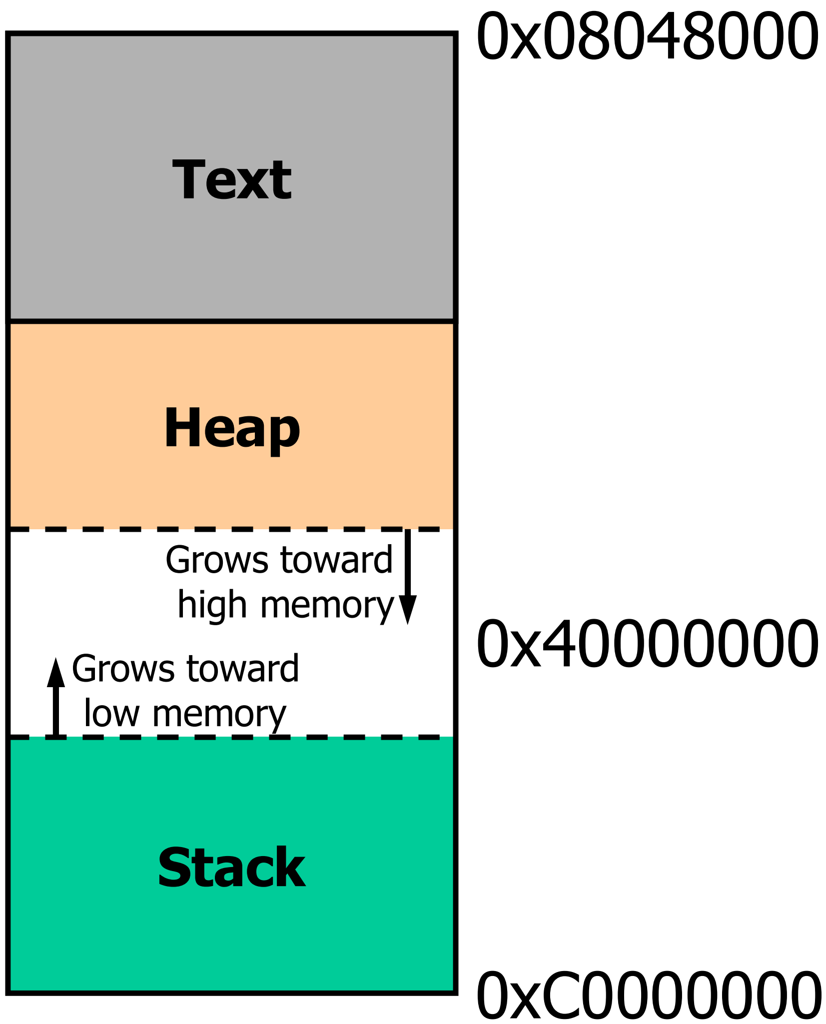
Paging is implemented in hardware+software (part of OS). Pages are typically 4K bytes.

Security: each block can be marked as protected. (Kernel/OS)



Virtual address space
Physical address space
0x00000000
0x00010000
0x10000000
0x7fffffff
text
data
stack
0x00000000
0x00ffffff
page belonging to process
page not belonging to process

***Process Memory Layout – see part 10



- Text: loaded from exec code and read-only data
- Heap: runtime allocated objects,
- Stack: LIFO, holds function arguments and local variables,

Basic Security:

Linux: Kernel pages are never swapped to disk.

Windows: similar but more complex.

General Protection Fault (GPF)

It is a hardware mechanism! Exception 0xD.

Occurs when the program does violate the CPU security rules

- using an instruction which can ONLY be used at ring 0 by the OS Kernel…
- accessing special types of registers and Descriptor Tables…
- etc…

- The OS is expected to catch it and close the process:
  - XP: XXX has encountered a problem and needs to close. We are sorry for the inconvenience.
  - Vista/7: XXX has stopped working.
- If not, if GPF occurs 3 times, (exception within exception handler routine) even the OS Kernel cannot recover from it.

=> Must reboot CPU and OS.

Segmentation Fault a.k.a. Bus Error

One example program in C:

```
char *s="abcd";
*s=0;//change 'a' to '\0'
```

This will cause segmentation fault,
both in Unix/Windows
because compilers allocate “abcd”
in a segment marked as “read only”.

Software mechanism with hardware detection by CPU.

Page Protections and Permissions

- Historically, in Intel 32-bit CPUs, permissions (R/W/X) exist at segment level, hard to use and wasteful.
- Since i386, W/R permissions exist at the page table entry level, 4 K pages typically
  - implemented in combination of hardware / OS kernel with the "page descriptors"
- ONLY since late Pentium 4 Prescott, X (execution) can also be disabled with DEP (later about it)

- BTW: No problem ever with x64. R/W/X at page level.
- BTW: with x64 segments are no longer used.
  - Most new PCs in 2012 are x64.

**Vista and Encrypted Paging

What about memory used for operations on protected audio/video content in Vista?

- For example a graphics card using RAM for its real-time working data?

Vista activates a special protection bit indicating that they must be encrypted before being paged out to the disk, and decrypted back again after being paged in.

*Kernel or Memory Dumps

If Windows XP crashes …

- it will write either a full memory dump, or just a Kernel dump into the page file (pagefile.sys).
  - temporary.
    - after reboot it will be copied to a separate file.

Who else is accessing my memory? Direct Memory Access (DMA)

- processes do not access “physical” memory.
  - HOWEVER physical memory also include hardware devices that are memory mapped by the MMU.

- What is DMA?
  - Mechanism for devices attached to the CPU to read / write directly to memory.
  - Avoids the CPU becoming a bottleneck, and frees it from spending most of its time doing I/O.
  - With DMA the CPU initiates a transfer, does something else, and receives an interrupt when it is done!
  - A good idea, performance wise.

- Used by, e.g., disk drive controllers, graphics cards, network cards and sound cards – all big users of I/O

*The RowHammer Attack

```
00000000|01111111|00000111|11111111|00000000
01010100|01010101|00010101|01010101|00101001
00000000|01111111|00000111|11111111|00000000
```

- Basic idea: when writing into a memory row, there is a very small chance that the write operation interferes with neighbouring memory rows.
- Vulnerable: DDR3 DRAM, Samsung, CPU-dependent.
- Mitigated by: refreshing, server ECC memory.

****ASLR = Address Space Layout Randomisation

will be covered later, see exploits / buffer overflow.

SOFTWARE, not hardware.

$W \oplus X$ Page Protections – in Linux

- What is $W \oplus X$?
  - Each page should be either writable or executable, but not both: Exclusive OR

Applications:

- Exe part of the program space (a.k.a. text) pages: X, not W
- Data pages (stack, heap): mark them as W, not X

Implementation of $W \oplus X$ in Linux:

- In Linux 32-bit, and with 64-bit CPU, in hardware, since Kernel 2.6.8.
- In other cases, mechanism can be implemented in software.
  - In OpenBSD since version 3.3. @2003.
  - In Linux PaX patch (optional), for 32-bit x86 processors

Windows DEP = Data Execution Prevention

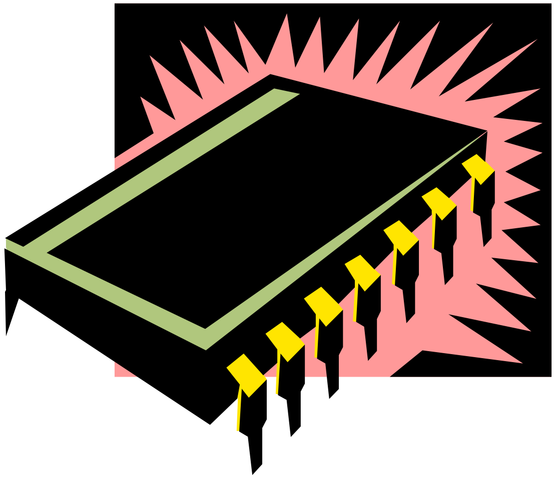
Old “X” idea: must allow explicitly, current OS+programs would stop working.

The “NX” idea: Never Xecute = can forbid. Easier to make compatible systems.

Hardware mechanism. Both Intel and AMD implement it but Intel was the last to deliver this benefit to large-public CPUs, since P4 Prescott.

- Windows - Since XP SP2.
  - Not active by default. Choice dictated by legacy programs…
  - PAE mode needed: 64-bit page tables. Bit 63 is used.
  - Compatibility problems with older processors and old motherboards
- Also active in Linux, mostly only for x64 CPUs even if you install 32-bit Linux on x64 CPU

CPU Security Features

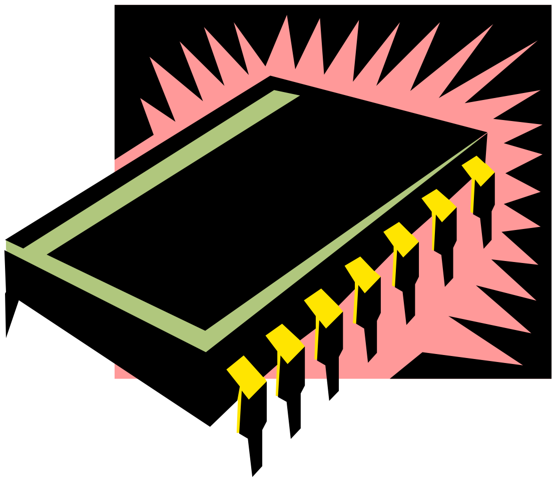


Unique Serial Number

- Routine mechanism in most industries.
  Unique serial number cannot be changed
  (fixed by the manufacturer)
  - Example:
    Oyster card, building passes block 0.

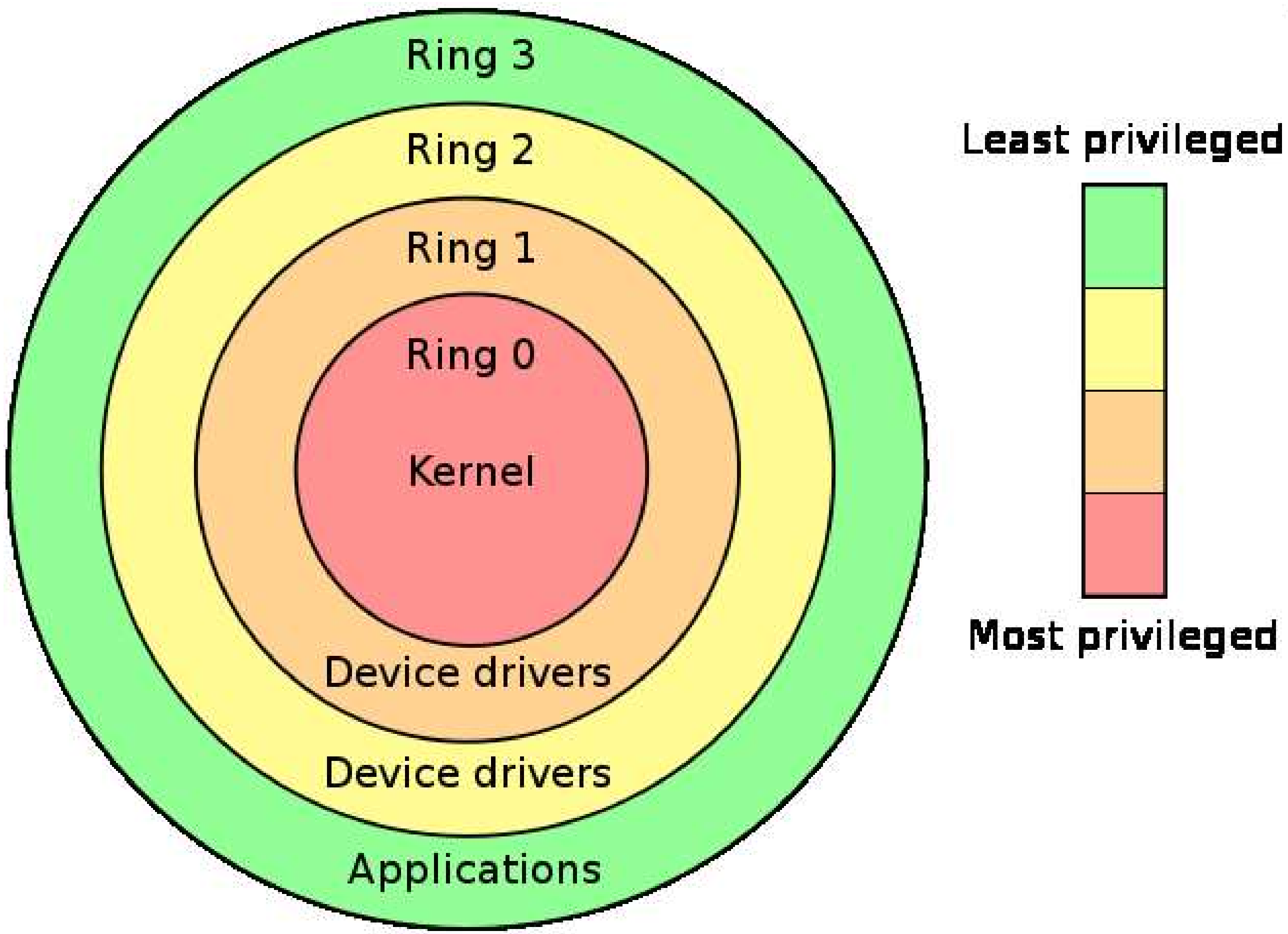
- Introduced by Intel since P3.
  – can be disabled too, due to privacy advocates outcry…

CPU Protection – Hardware Side



Rings – Hardware @ CPU

Different CPU architectures define several Rings.

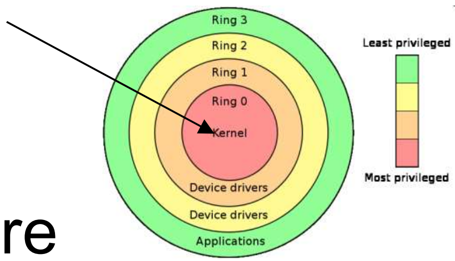


Transition Calls (Transition Gates)

- Transition only through special “system calls”
  - transfers control to a predefined entry point in more privileged code;
  - the more privileged code does specify and checks
    - where it can be entered,
    - in which prior processor state one can enter.
    - Privileged code, from the processor state and the stack left by the less privileged code, determines what is requested and allows it or not.

***Privileged instructions Only accessible to Ring 0

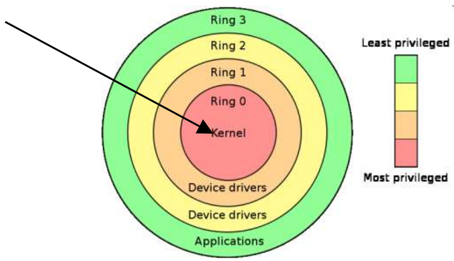
- LGDT — Load GDT register.
- LLDT — Load LDT register.
- LTR — Load task register.
- LIDT — Load IDT register.
- MOV (control registers) — Load and store control registers.
- LMSW — Load machine status word.
- CLTS — Clear task-switched flag in register CR0.
- MOV (debug registers) — Load and store 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.



How to Penetrate to Ring 0?

Whatever you do, it is always possible to get there through the boot loader.

- Critical and privileged access point in all PCs.
  - Would allow to disable some hardware securities such as DEP…
  - Could allow a virus to be so stealth that no anti-virus would detect it.
- Beware of boot sector viruses!
- Good news: most motherboards have a hardware mechanism that prevents the OS from writing the boot sector of the hard drive. No access from the O/S level.
  - Problem: this can be disabled in BIOS.
    - which is looking for trouble:
      - IF this mechanism is totally usable: like it makes sounds and asks the user to press Y on the keyboard, and there is no bug/problem
      - THEN it is a bad idea to allow people to disable it.



How to Still Penetrate to Ring 0?

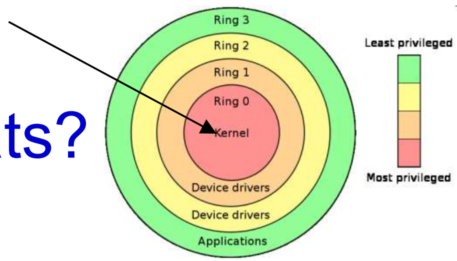
More HW mechanisms…

Furthermore, the BIOS has a password (and usually also an admin password). But all NVM can be reset by a jumper, so it is easy to hack…

Some computer motherboards designed for high security customers/applications and certified by the government will have better security… such as

- WORM* mechanisms = Write Once, Read Many
- unhackable BIOS… (more about BIOS sec later)

Nicolas T. Courtois, 2008-2018



Can We Defend Against Such Threats?

Yes, or partly so, through logging helped by hardware.

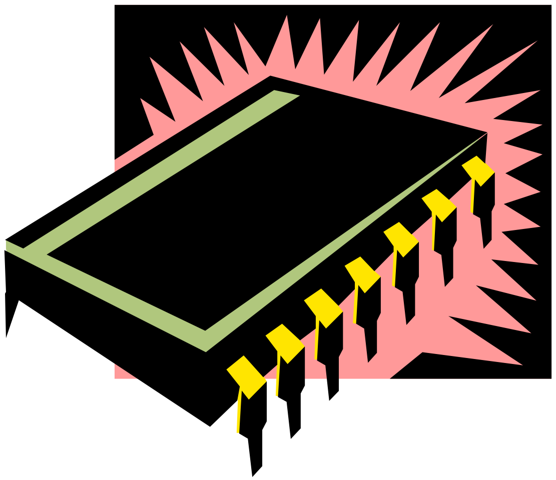
<u>Example 1:</u> a motherboard can have a log of events that cannot be erased (WORM).

- Sandisk recently started commercializing WORM memory cards (with very large capacity) – the data can be written but cannot be erased.

<u>Example 2:</u> Hard disk hardware can make it impossible to modify the file creation and modification dates of files.

Then the virus can be detected (removal is another story).

CPU Protection (3)
Hardware + OS



CPU Modes vs. Modern OS

- most current OS and Windows XP use only two rings
- ring 0 == kernel mode
- ring 3 == user mode
- only recently Microsoft have added some ring 1 code…

CPU Modes = Processor Modes = Privilege Levels

Hardware mechanisms that allow the OS to run with much more privileges than <u>any</u> process.

- System mode = privileged mode = master mode = supervisor mode = kernel mode = unrestricted mode.
- User mode
- Transition only through special "system calls" or privileged instructions or hardware interruptions

In theory, only highly-trusted kernel code should run in the unrestricted way.

In practice… Real time code such as drivers are allowed to also run in the system mode.

system mode = privileged mode = master mode = supervisor mode = kernel mode = unrestricted mode

- execute any instruction
- access any memory location (not anymore in recent CPUs)
- access hardware devices
- change a number of special processor features:
  - enable/disable interrupts,
  - special registers,
  - descriptor tables,
  - change privileged/not processor state,
  - access memory management units,

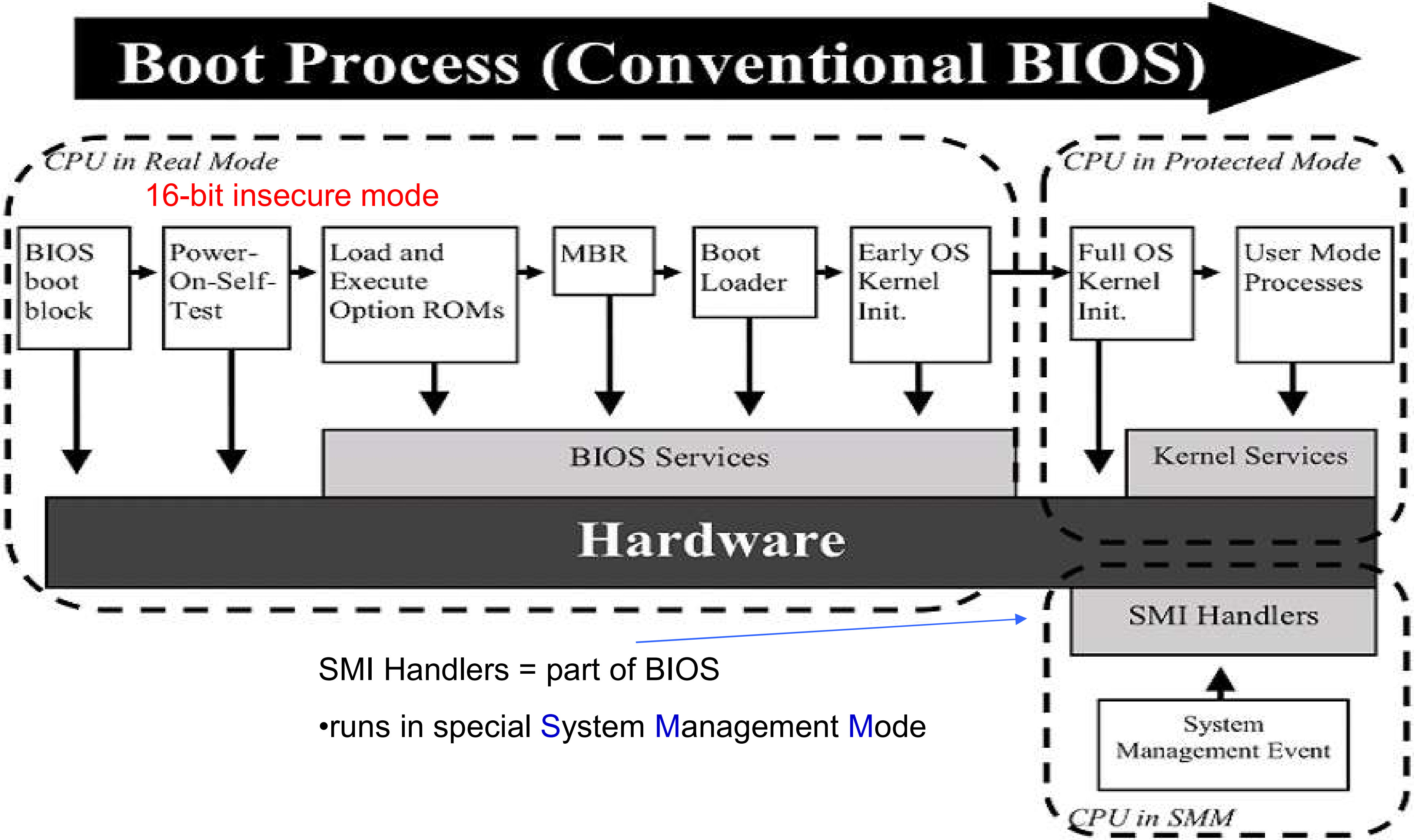
user mode

- access only the “usual” CPU resources: computations+registers
  - access to memory is limited,
    - cannot access MMUs
  - cannot execute certain special instructions,
  - cannot disable interrupts, go to privileged state, change special registers/tables, etc..

*-1=System Management Mode = SMM

- on every PC,
- like ring 0 or-1, invisible to OS @ring 0
- 16-bit powerful mode, hardware interrupts,
- used by the BIOS NOT the OS,
  - OEM extensions, dealing with hardware specifics
  - prevents CPU from overheat etc..
  - no one knows what the code does, or even when it is executed

SMM mode Viruses?



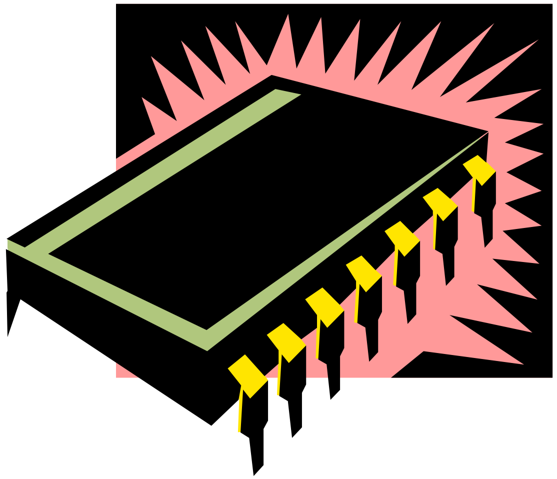
*-2=Micro-code Updates

- x86 architecture is CISC:
  - Complex instruction set.
  - Implemented inside the processor using microcode.
  - Facilities exist to update the microcode to fix bugs.

- It is not in the public domain:
  - (a) What exactly the microcode update does. For example it may add new instructions.
  - (b) What facilities protect the update mechanism.
    (expect that the code is digitally signed)

  - “No one” knows what it does, or how it works.

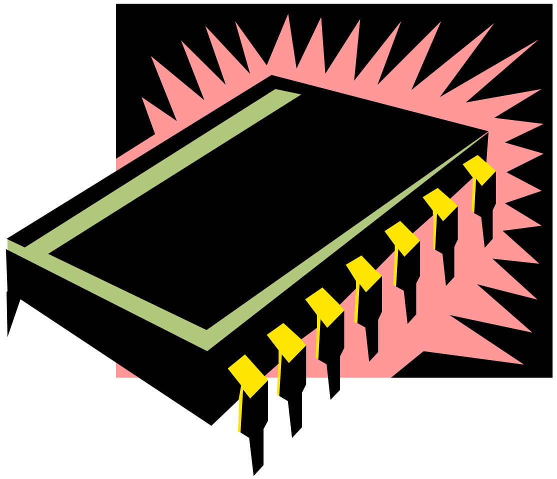
Virtual PCs and Hardware Memory Isolation



VirtualBox, VMWare etc

- most current OS:
  - ring 0 == kernel mode
  - ring 3 == user mode
- using Virtual Box in software mode
  - Hosted OS kernel runs at ring 1 replacing 0
    - has a real-time code recompiler which replaces some instructions
    - also does real-time code patching
    - cannot run VirtualBox or VMWare inside it because cannot create virtual machines
- Virtual Box in hardware –assisted mode:
  - ring 0 is run as ring 0 with isolated memory
    - possible only on very recent CPUs: Intel VT = AMD-V
    - a virtual PC+OS cannot detect it is a virtual PC
    - faster execution, the difference between real and virtual PCs tends to disappear,
    - run Windows, Linux and Mac-OS AT THE SAME TIME as native (!)

BIOS and BOOT security



Firmware

Def: a tiny ‘master’ program included in our PC

- runs first when you switch it on / or reset
  - can be updated

Traditional BIOS

BIOS def. = a firmware in your PC

Basic Input/Output System, 25 years old

Responsible for (picture next slide):

1. initialization of much of the system, including important components such as video, RAM, keyboards and mice.
   – POST = Power-On Self Test, (NOT hard drives or media)
2. responsible for finding and loading the OS Boot
   – from a number of different types of media, ranging from hard disks to USB and LAN devices [can load option ROMs]
3. cooperates with the OS load further parts of the operating system before the operating system completely takes over.

Threats and Attacks

Firmware update: if a virus does it, it can circumvent all the OS security… Pre-OS attacks = before the OS loads.

Malicious updates can enter as:

- user-initiated from a bootable disk
- runtime software exploits
- managed BIOS updates inside a company
- new vectors? - self-updating BIOS rootkits? Maybe.

Payload:

- Roll-back to old insecure version of BIOS (with attacks)
- Install a Firmware Rootkit
- Install a virtualization virus

Firmware Rootkits

Firmware Rootkit = def:

uses device or platform firmware to create a persistent malware image in hardware, such as a network card hard drive, or the system BIOS. The rootkit hides in firmware, because firmware is not usually inspected for code integrity.

Attacks

- At BlackHat 2006 Heasman demonstrated the viability of firmware rootkits in both ACPI firmware routines and in a PCI expansion card ROM.

- Virtualization viruses: e.g. Blue Pill,
  run whole OS as a virtual machine, some physical RAM is invisible, rootkit claimed impossible to detect
  - code released by Joanna Rutkowska c. 2010

Enterprise Remote IT Management And Anti-Theft

[since 2010]

Couple = AMT + vPro

- AMT = Active Management Technology – software part.
- works with HARDWARE Intel vPro support on CPU + motherboard + network adapter support.

A very impressive set of out-of-band techniques to remotely connect to PCs, even without knowledge or permission of the OS and the user that physically controls the PC.

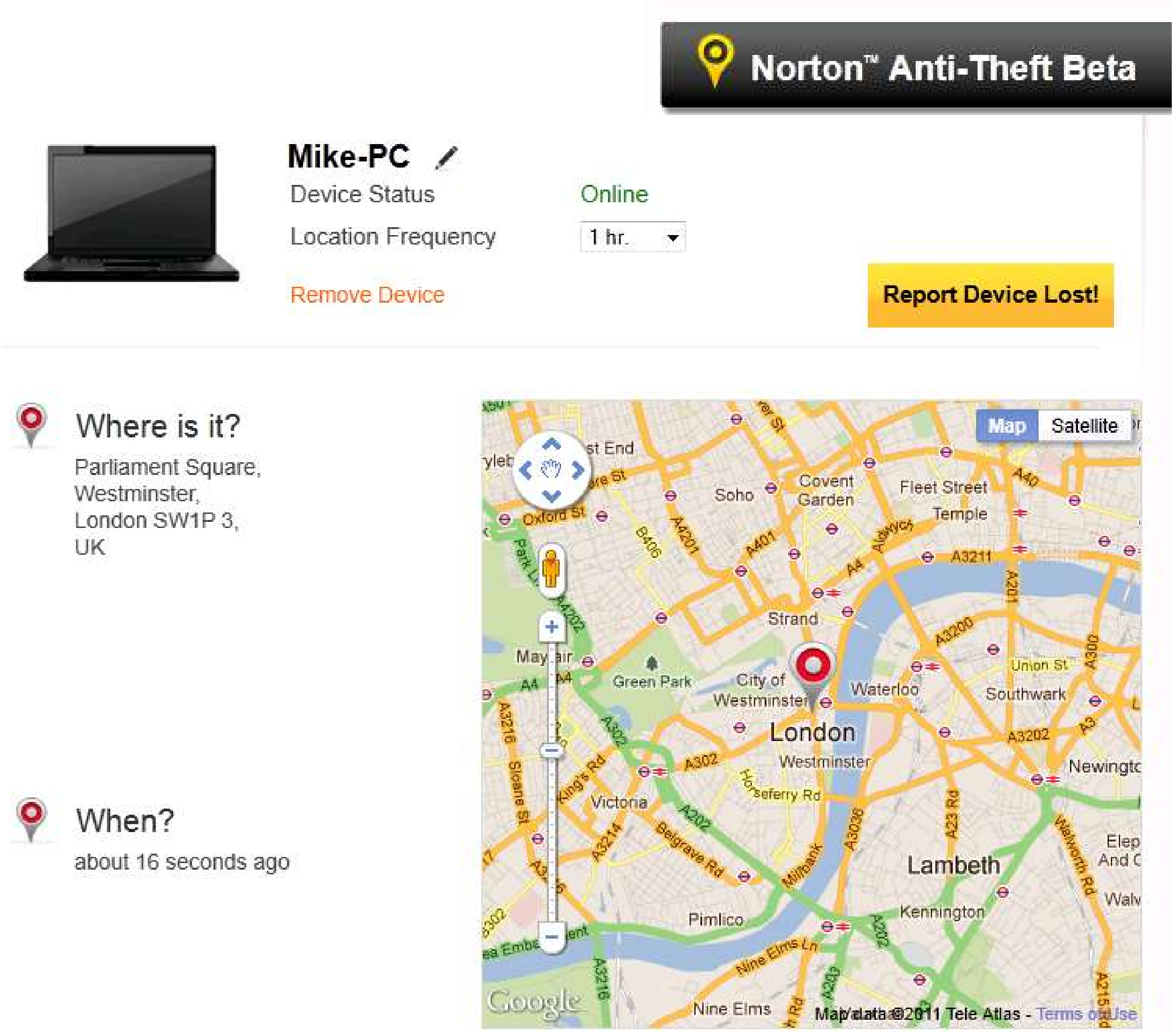
- Remote power up
- Remote configuration, including access to BIOS
- Encrypted network boot
- Programmable hardware-based network filters and alerts
- Remotely limit network traffic of infected PCs
- Persistent logs stored in protected hardware

*Intel Anti-Theft Technology [2010]

Tamper-resistant HARDWARE protections:

- Allows encryption solutions to store and manage cryptographic keys in CPU hardware.
- Ability to disable your PC with a local or remote poison pill if the system is lost or stolen. The poison pill can then delete essential cryptographic keys in CPU hardware.
- The PC will refuse to boot
  - works even if the OS is reimaged, the boot order is changed, a new hard-drive is installed, or the laptop is disconnected from the network.
  - supports outgoing SMS (alert) and incoming SMS (poison pill) through an optional 3G card built-in.
  - can display a message to the thief:
    - like laptop reported missing, 100 $ reward if you find it, call this number etc.
- Customize the policy to respond to events
  - invalid login attempts
  - failure to check-in to company network
  - tamper detection
- Has a reactivation capacity: restore to normal.
  - Secondary long pass phrase to unlock
  - Unlock code can be transmitted by phone by the company's IT service.

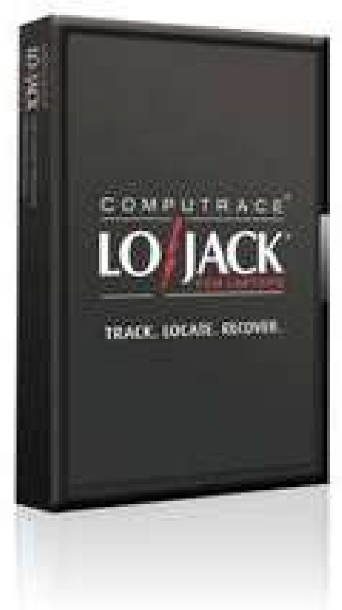
Location Tracing of a PC



CompuTrace:

Computrace® Track Locate Recover

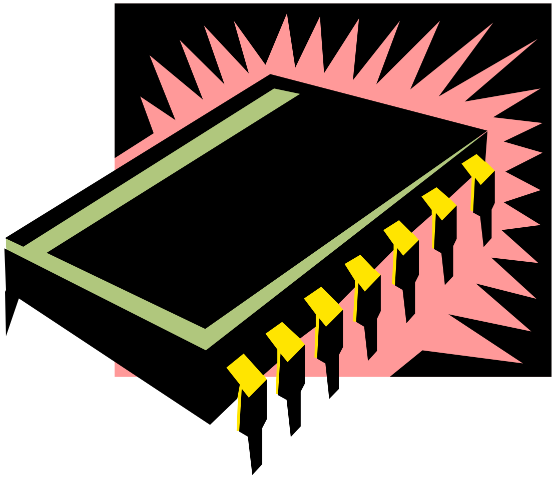
- embedded in laptops and more expensive Dell workstations.
- + paid subscription for servers
- intended to trace lost/stolen PCs without the knowledge of the thief



CompuTrace as a RootKit

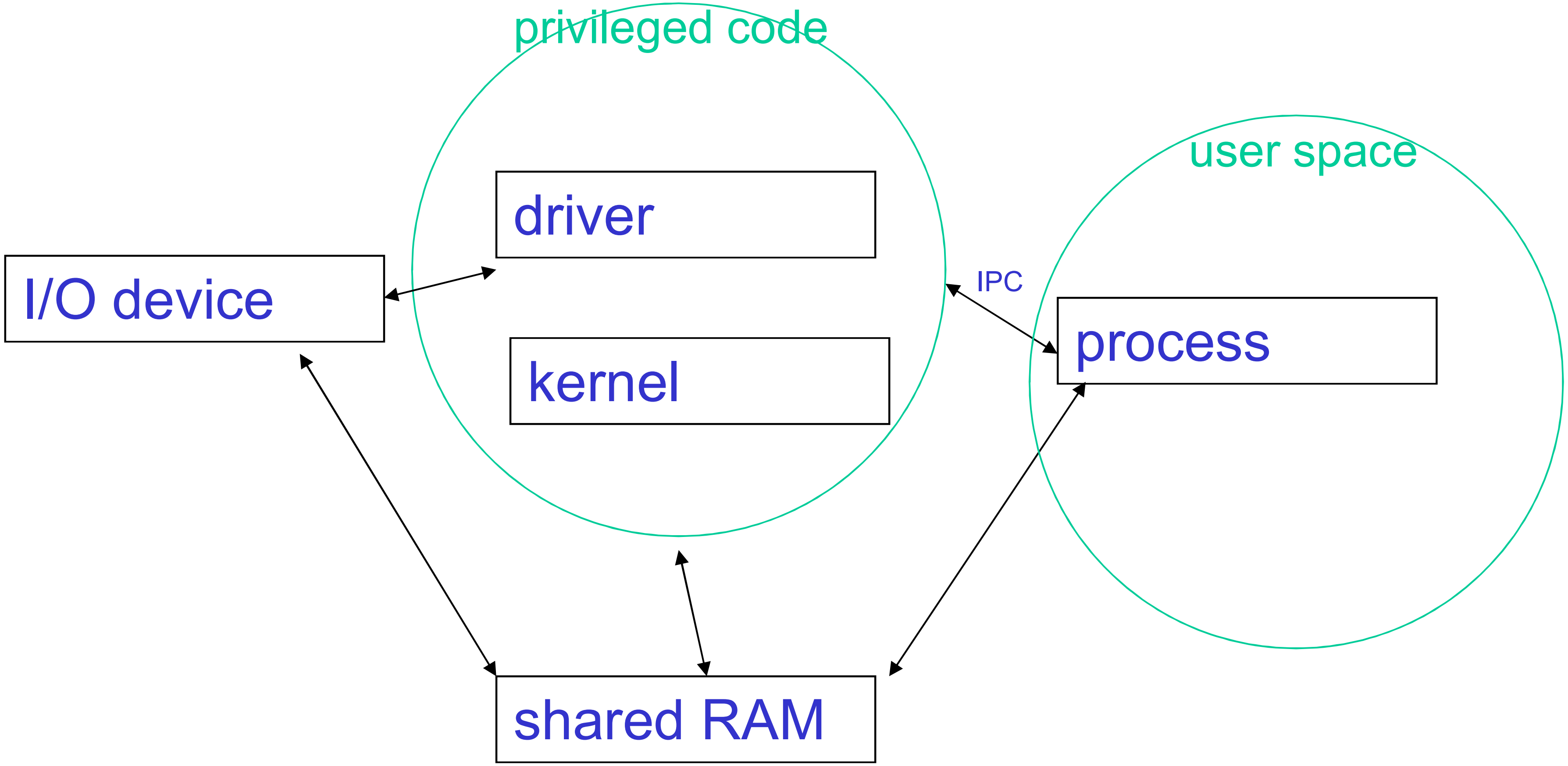
- also known as "a legitimate BIOS rootkit".
  - upon activation it will HACK/MODIFY the Windows partition:
    - add a new service!
    - modify several system files and the registry
    - modify self-healing mechanisms such as Autochk.exe so it CANNOT be detected or repaired(!)
- can be enabled or disabled or killed
  - in theory cannot be reactivated…
  - in the BIOS, appears as a PCI device 1917:1234
- can be hacked/subverted, cf. Sacco-Ortega attacks, BlackHat 2009,
    - redirection of communications… changing the URL/IP address
    - lack of authentication of code, could be replaced by malicious code...
    - Rootkit CAN be re-set to default settings and re-activated by software only
    - maybe can download unauthorized code during updates?

I/O Protection

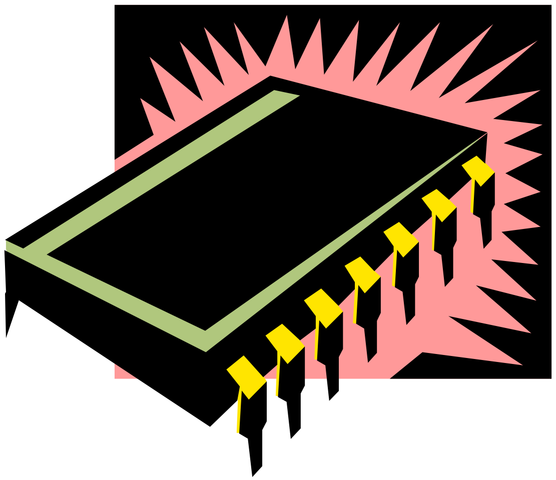


In Both Unix and Windows

- I/O devices are usually NOT accessible in user mode!



Trusted Path



Trusted Path (InWards)

One possible meaning:

- a mechanism that provides confidence that the user is communicating with the right program/process
  - attackers should not be able to
    - initiate the communication process
    - snoop on it / modify it
  - defense against fake login programs.
- In other words: something close to an “unspoofable” and “incorruptible” channel (for a process in question).

Trusted Path (Outside direction)

- Windows User Account Control (UAC)
  - Windows Vista (relaxed for W7-W10 now)
  - Aim: communicate the need of a security decision to the user.
  - Threat: A user application hijacks the screen buffer, to ask an innocuous question.
    - Do you Trust Adobe… Yes of course I do…
    - Update later.
  - The user clicks “yes” to the wrong question!

**Trusted Path and DRM

The dream of Hollywood studios:

A graphics card that decrypts video directly with AES-128, so that high-quality video cannot be captured…

Implemented in Windows Vista…

Secure Attention Key

Def: a special key combination to be entered before a login screen is presented.

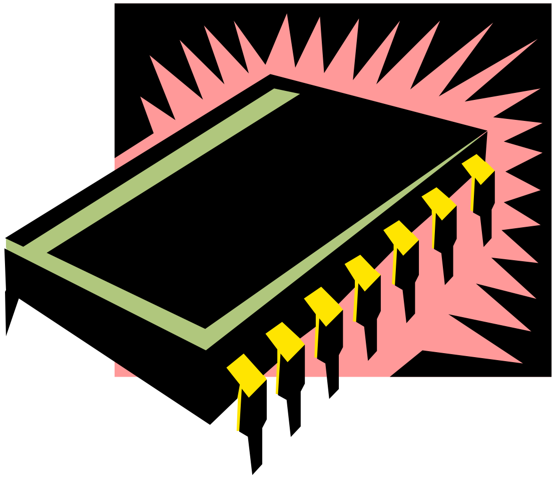
- Windows NT, XP and better: Ctrl+Alt+Del
- Linux: Ctrl-Alt-Pause or the SysRq-K

WinLogon Security

Windows NT is designed so that only the WinLogon process, a trusted system process, can receive notification of this keystroke combination.

Image Name	PID	User Name
winlogon.exe	804	SYSTEM

Security in the User Space



Security Mechanisms in the User Space

- User privileges (admin/not admin)
- Access Control
- Authentication
- Logging / Auditing
- Intrusion Detection
- …

Logging

- Both normal and suspicious events, e.g.
  - Every logon attempt
  - Every time permissions are changed
  - Network connection events
- Methods
  - – application logging,
  - – API hooking,
  - – system call interception,
  - – packet sniffing, …

Again, logging can be hidden
and use WORM mechanisms (forensic logging).

Disturbing Questions and Virusology

Disturbing Questions:

The OS does have some “file locks”:

- It does not allow one to change system files and things such as file meta-data easily.

Some software, such as real-time disk defrag or real-time partition tools do need to have higher privileges than normal software.

Disturbing Questions:

The anti-virus software works in the user space?

Not really (try to kill an anti_virus),
but even if it has Kernel-level drivers there is a process to install it…

Q: What prevents a virus from installing in the same way? With very high privileges the anti-virus needs to function?

Hacking Anti-virus Software

Could one install a slightly modified anti-virus software?

Defences: The process is in fact pre-approved by Microsoft, installation is usually allowed by checking digital signatures of its key component = a Kernel-level driver.

Jailing Anti-virus Software

Could we put the anti-virus software in jail?
Or just alter its communications with the central servers [updates, status/virus reporting, redirection etc]

Maybe.

Hardware Attacks on PCs

High-Level Categories

- Stealing data (removing hard drive, memory chips).
  - Now hard drives can be encrypted.
    - Memory still isn’t.
- Hardware keyboard sniffers.
  - Optical
  - Acoustic / mechanical vibrations
  - EM radiations. PS/2 vs. USB (two wires=).
- Intercepting screen output.
  - There are TEMPEST machines (shielded electromagnetically)
- Side channel attacks focusing on crypto.
  - Acoustic attack on AES; .
  - CPU cache attacks on crypto (AES)…
  - Side channel attacks on VM running on the same CPU

Use Case:
**Web Browser Design

(not covered in class,
study at home)

Big (Monolithic) Browsers

- Legacy de facto dominant situation,
  - since NCSA Mosaic program [1993]
- Monolithic architecture:
  - initially, the browser kernel and the rendering engine were just single image (one exe file)
  - later they became modular with dlls, plugins, JVM, etc.
  - But from the point of view of the security nothing changed: all code executed in one single protection domain.
    - Examples: IE7 under XP, Firefox 3, Safari 3.1.

One Single Domain

Everything is run in one single protection domain at the user's level of privilege, for example as admin.

- A single crash crashes everything
- Code that comes from the web runs locally at user's level of privilege
  - an un-patched vulnerability in the browser allows to run any code on the host machine, with the privilege level of the user.

Google Chrome Architecture

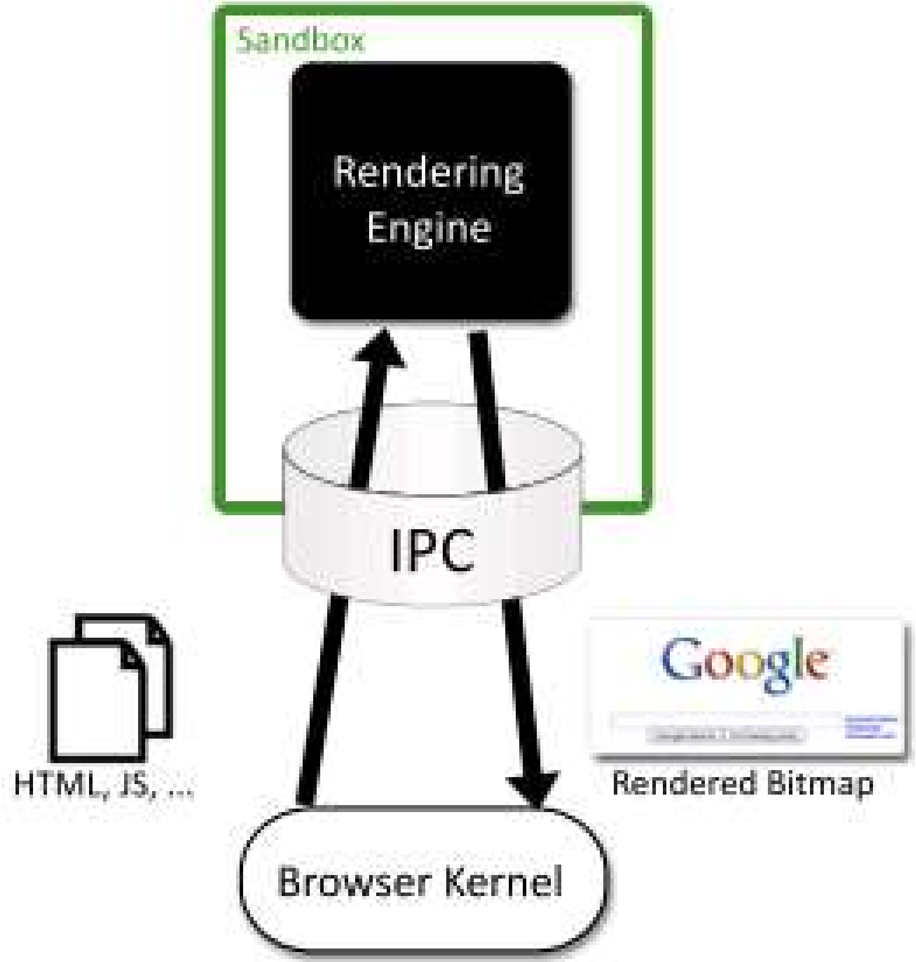
Divides the browser application into two protection domains:

- Browser kernel; runs with user's privileges,
- Sandboxed and isolated multiple instances of the rendering engine run at very low "web" privilege level,
- Chrome is open source.
  - And highly compatible with existing web sites, unlike many other existing modular "highly secure" browsers [DarpaBrowser, Tahoma].

Chromium Browser Kernel

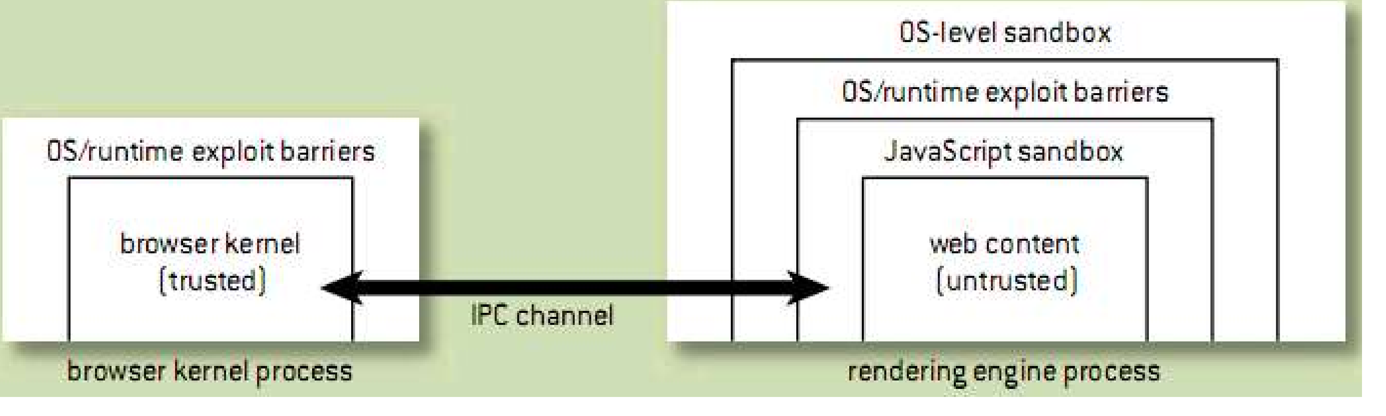
Browser kernel: responsible for

- mediating file and network access, like a firewall
- displaying bitmaps produced by rendering engine[s] seen as black boxes.



Chromium Browser Kernel Privileges

- run at user's privileges.
- run as a medium-integrity process under Vista,
  - as a result, it can be installed without an admin account (!).
    Maybe because it is not as dangerous as most other browsers…



Rendering Engines

- multiple instances
- sandboxed and isolated
- each running with DEP and ASLR
- all run at very low “web” privilege level,

- execute error and exploit-prone tasks of
  - web parsing,
  - Java script,
  - etc.

How Does It Protect Our Files!

An engine, accessing URLs outside, is just totally unable of accessing local files file:///

However, of course, one can open a local web page.

- But only in another sandboxed and isolated rendering engine (!).

***Small Technicality

Memory isolation – OK.

But many file system privilege / isolation features will be ineffective if the file system is FAT32, not NTFS.

Quiz

- Can in Windows/Linux a process run by an administrator access the system/kernel memory?
- Explain what is virtual memory and paging?
- How one can make a dump of kernel memory?
- What is DEP? Which OS has it?
- Explain the protected/Kernel mode and user mode for CPUs.
- How can the DEP and the protected/Kernel mode be circumvented or attacked?

Quiz (2)

- How can we at the BIOS level make it difficult to modify the boot sector?
- Optional: What is the architecture of Google's Chrome regarding the privileges to read and write files, access the network, and the screen?