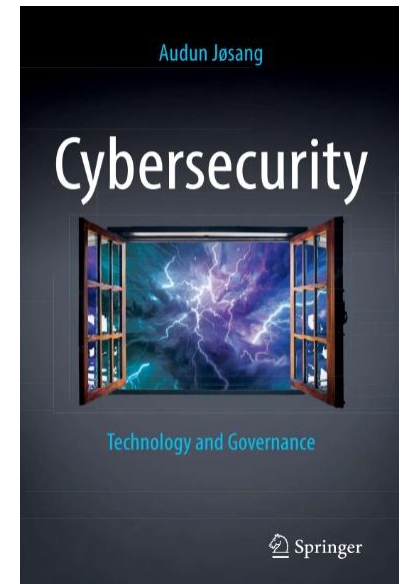


Chapter 3: System Security

Cybersecurity: Technology and Governance

Audun Jøsang,

1st Edition, Springer Nature, 2024.

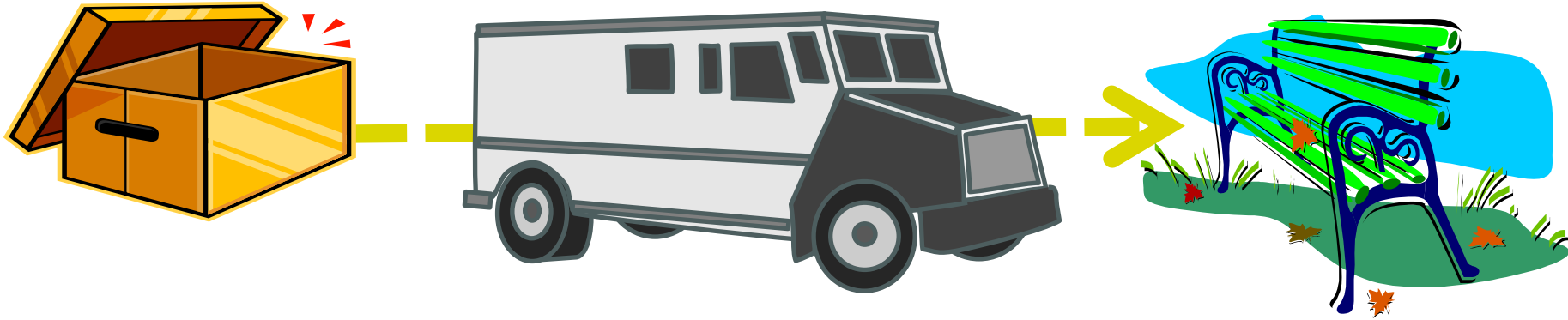


Overview

- System architecture
- Security in operating systems
- Virtualization architectures
- Trusted calculation
- Secure boot
- Side channels and hidden channels



The Importance of system security

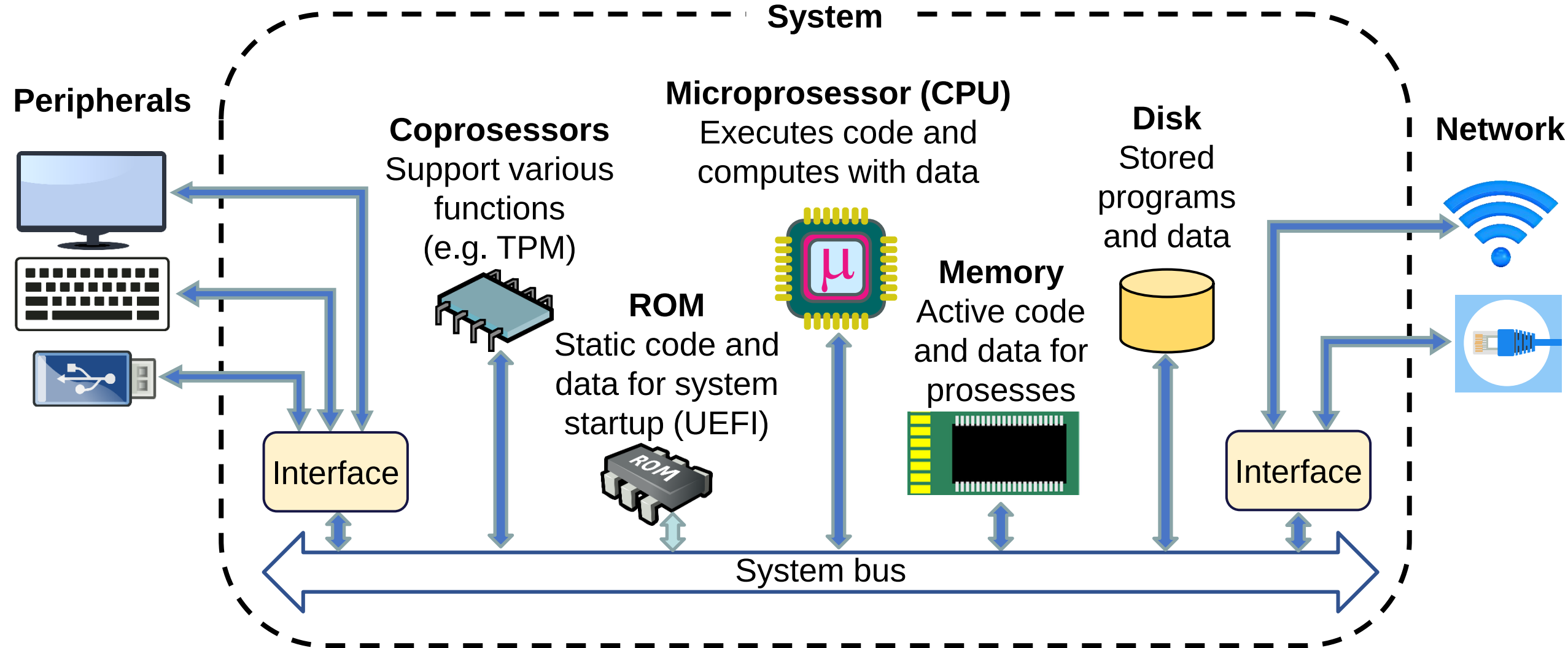


- *«Using encryption on the Internet is equivalent to using an armored car to deliver credit card information from someone who lives in a cardboard box to someone who lives on a park bench»*

Eugene Spafford

- The moral is that strong communication security is wasted if system security is weak.

System architecture



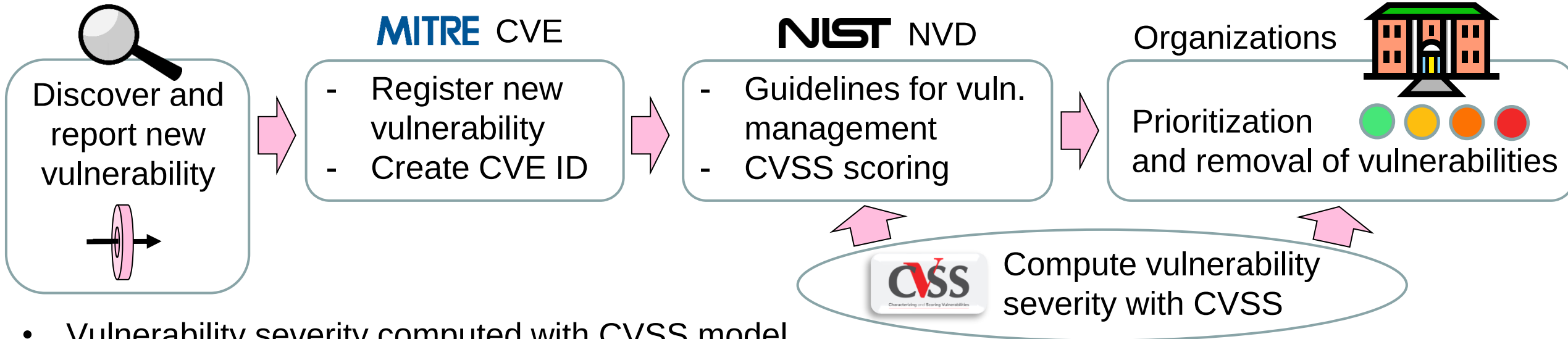
Continuous security update



- New vulnerabilities are continuously discovered
- Requires continuous online security update
- A zero-day vulnerability is only known to the attacker, so the software manufacturer has had zero days to remove the vulnerability
- Security patching can take time, so attackers often exploit known vulnerabilities

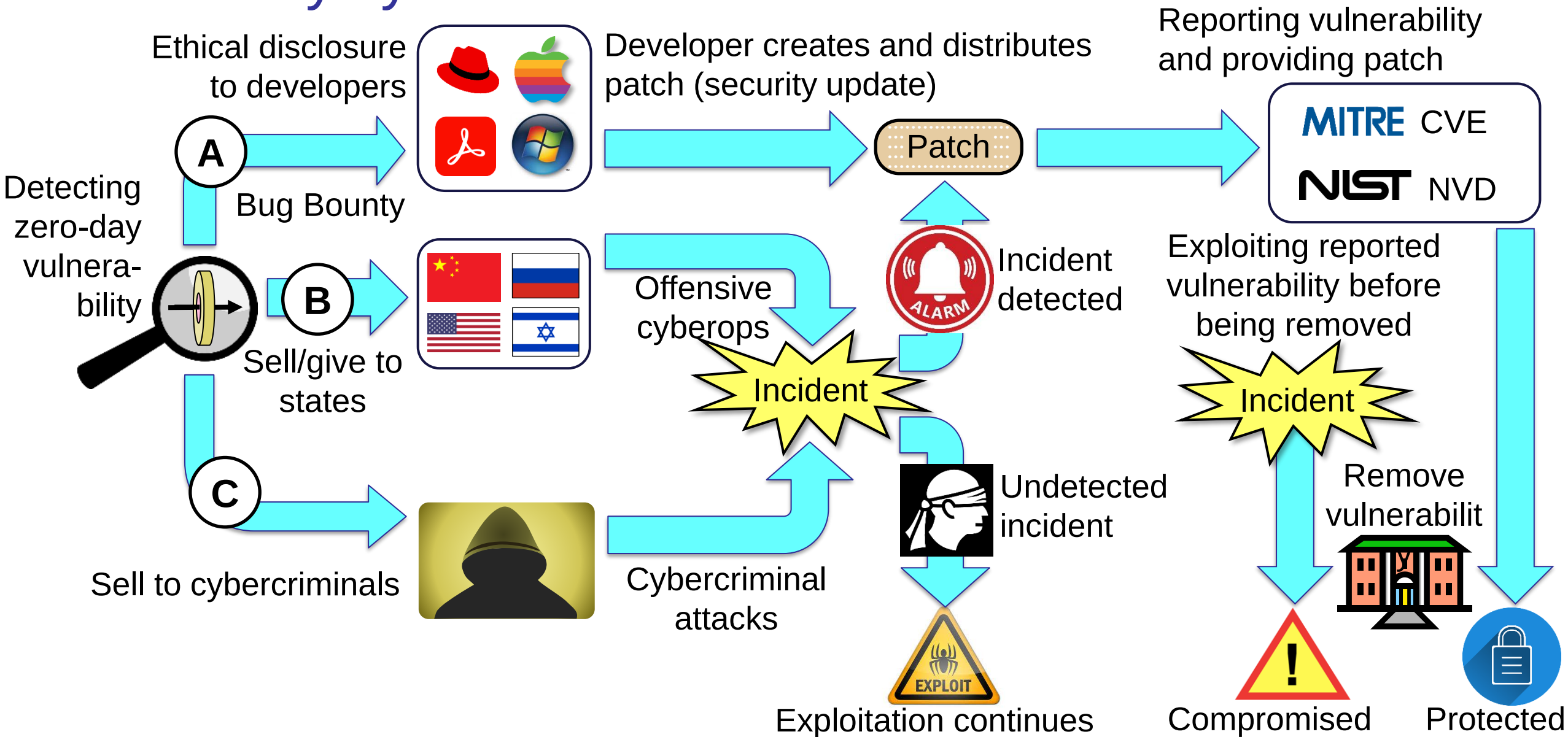
CVE (Common Vulnerability Enumeration)

CVSS (Common Vulnerability Scoring System)



- Vulnerability severity computed with CVSS model.
- Companies prioritise the removal/mitigation of vulnerabilities based on severity score:
 - Critical: 9,0–10
 - High: 7,0–8,9
 - Medium: 4,0–6,9
 - Low: 0,1–3,9
 - None: 0

Vulnerability cycle

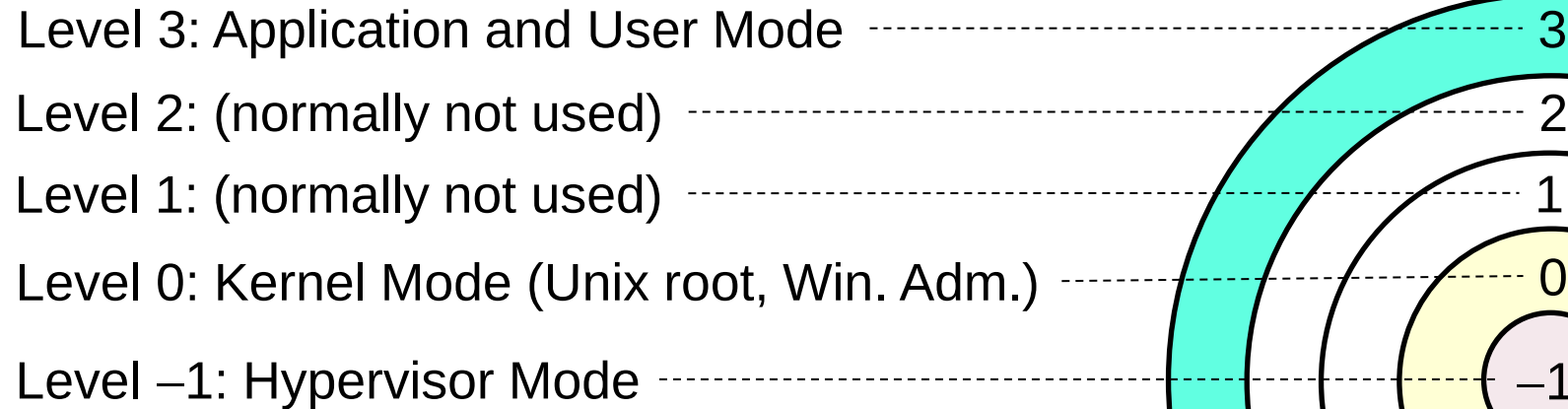


Approaches to strengthen system security

- Remove bugs and vulnerabilities in the OS (operating system)
 - New versions and security updates
- Add security features to the OS and CPU (Central Processing Unit)
 - Privilege levels, NX (No eXecute), ASLR (Address Space Layout Randomization), Shadow Stack
- System Security Monitoring
 - Antivirus tools
 - Firewall, intrusion detection
- Virtualization technology
 - Protect processes by separating virtual machines
- Trusted computing, e.g.
 - Secure boot with UEFI
 - Secure hardware on the platform, e.g. Trusted Platform Module (TPM)



Privilege levels for processes running in a system



- Hierarchical privilege levels were introduced in the CPU architecture of Intel (and AMD) in 1985 (Intel 80386), with 4 levels.
 - Level 0: most privileged (highest)
 - Level 3: least privileged (lowest)
- A new, most privileged "level -1" for hypervisors was introduced around 2006
- Illustrated metaphorically as "protection rings"

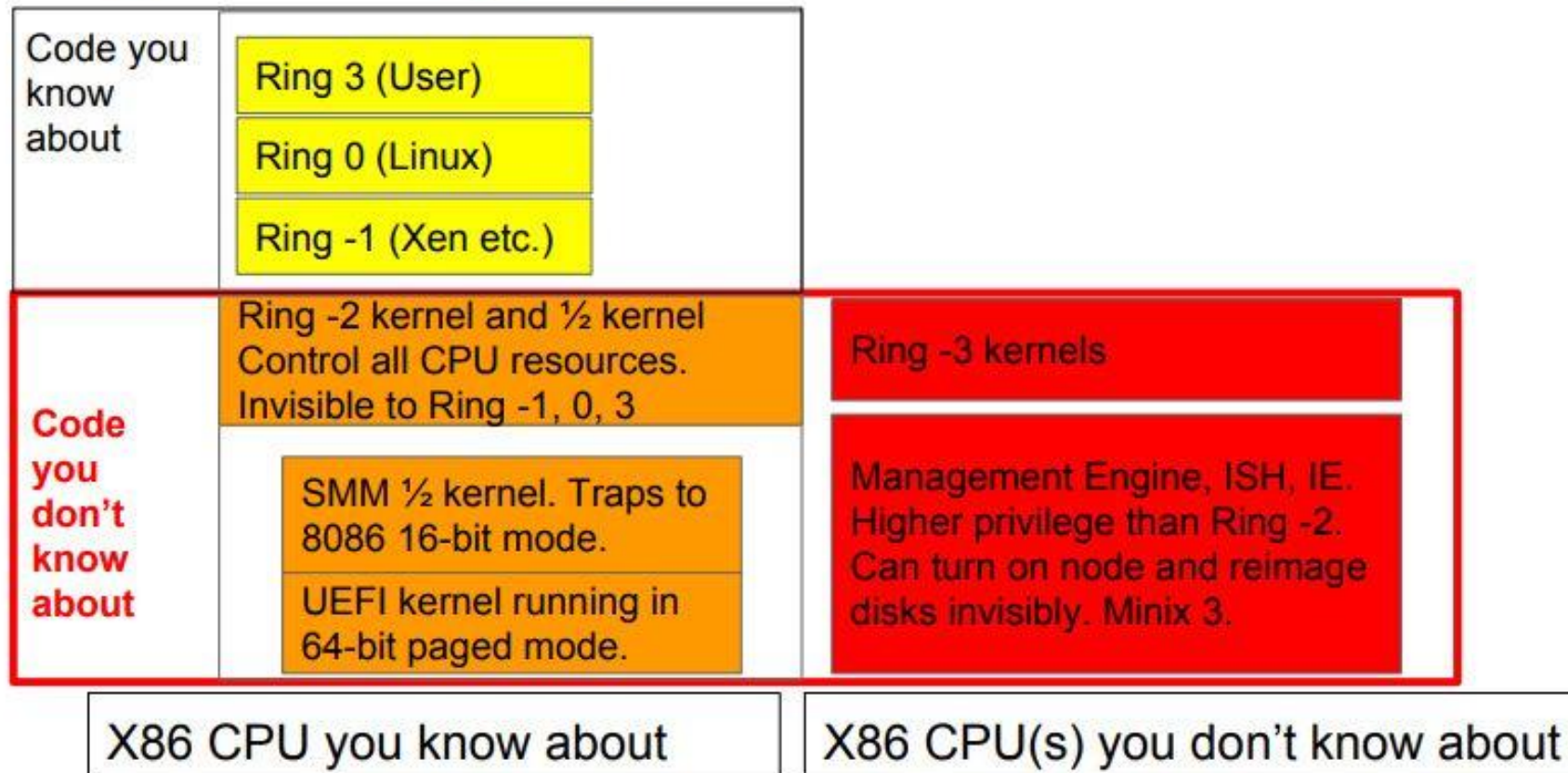
What happened to levels 1 and 2?

- ... it eventually became clear that the hierarchical protection that rings provided did not closely match the requirements of the system programmer and gave little or no improvement on the simple system of having two modes only. Rings of protection lent themselves to efficient implementation in hardware, but there was little else to be said for them.
- The attractiveness of fine-grained protection remained, even after it was seen that rings of protection did not provide the answer... This again proved a blind alley...

Maurice Wilkes (1994)

The hidden privilege levels on x86 CPUs

The operating systems



https://www.usenix.org/system/files/lisa21_slides_bautista.pdf

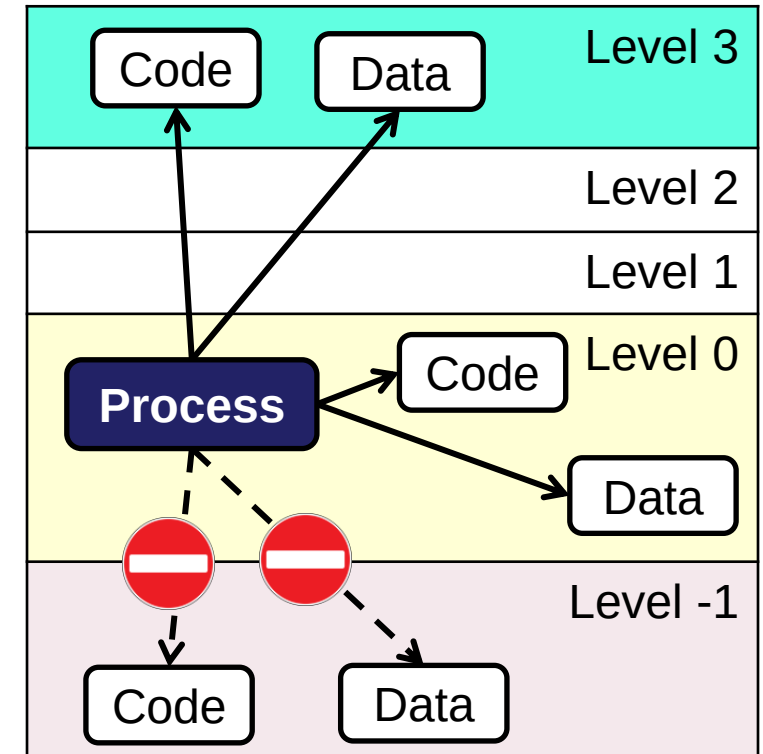
Intel ME (Management Engine)



- Intel ME (Management Engine) is a small, low-power subsystem that is integrated with Intel microprocessors.
- Intel ME runs MINIX-OS, which is active at all times when the microprocessor has power.
- Intel ME has full access to system hardware, including system memory, display, keyboard, camera, microphone, peripherals, and network.
- Intel AMT (Advanced Management Technology) is used for remote computer management. It can be for managing servers and client computers running Intel processors. Users of Intel AMT are typically large organizations, not home users.
- AMT must be enabled in UEFI/BIOS to be used, and is disabled by default when servers are delivered to the customer.
- AMT remote management can configure and reinstall the OS, manage or wipe software on computers with Intel processors. AMD has similar technology. This is called "wake-on-LAN".
- Unlike typical platform management solutions that require OS support, AMT works even if there is no operating system (Windows, Linux, MacOS, hypervisor) installed.

Access control based on the privilege levels

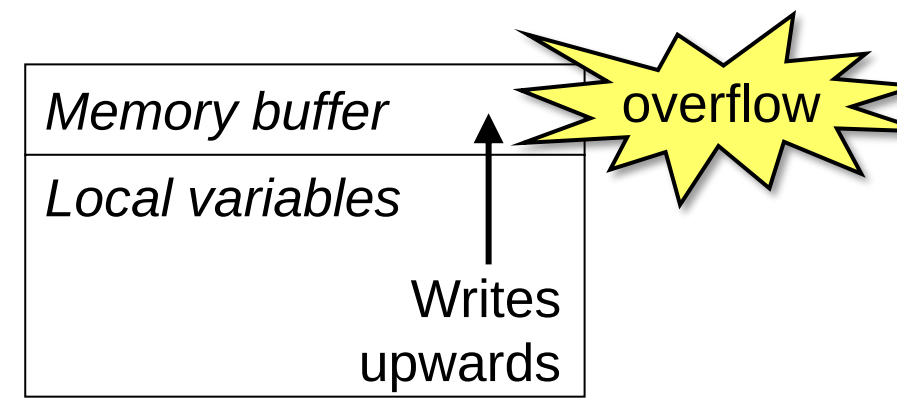
- A process can access and modify data and software at the same or lower privilege level as itself.
- A process running in kernel mode (level 0) can access data and SW at levels 0, 1, 2, and 3
 - but not at level -1
- The goal of the attacker is to gain access to kernel (level 0) or hypervisor mode (level -1), e.g.
 - through exploits
 - by tricking users into installing malware



Example

Buffer overflow

- Buffer overflow is a classic vulnerability
 - Exploited by e.g. Morris worm and SQL slammer
- In principle, the idea is that the memory is overwritten and corrupted, which e.g. allows the attacker to run (their own) malicious code
- The code will then run with the same privileges as the original program
 - If the vulnerability is in the root/admin program, the attacker will be able to run their code with those privileges
- There are several ways to achieve a buffer overflow, and we will focus on a stack-based version
- This requires some insight into how (virtual) memory for a process works



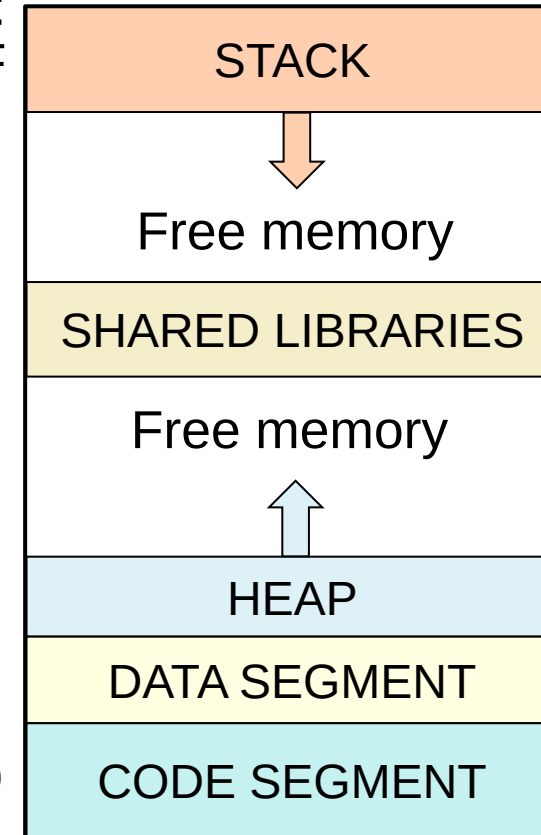
Virtual memory of a process

- Each process has its own 4GB of virtual memory space.
- Virtual memory addresses are translated by the OS into physical memory addresses before data is read and written into the physical system memory.
- Although all processes have the same virtual address space, in reality they have separate physical address spaces.
- Thus, a process does not have access to physical addresses for other processes, only to its own physical address space.
- It also means that virtual memory addresses will be the same every time the process runs. Attackers can exploit this, but this can be avoided with ASLR (see later).

Highest virtual
memory address:
FFFF FFFF

Available memory
for each process

0000 0000
Lowest virtual
memory address:



Data: read/write

Code: read/execute

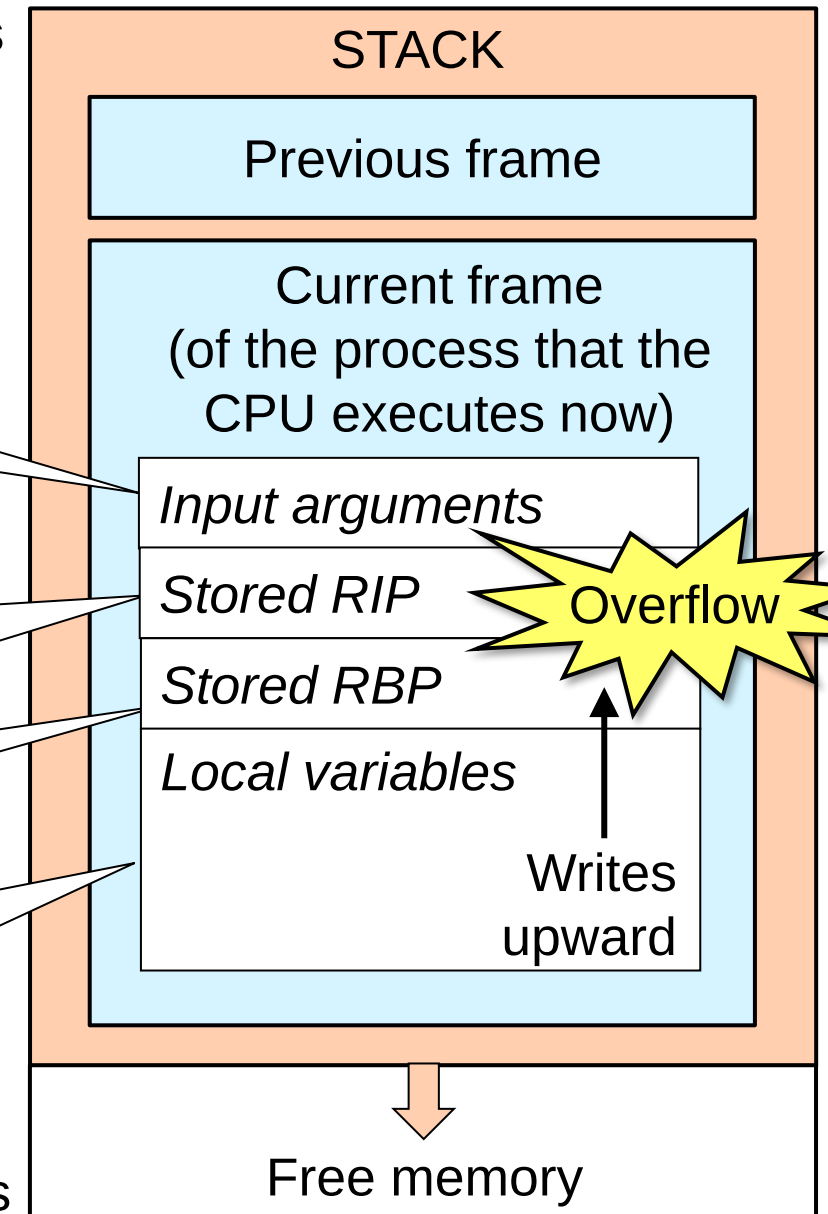
Data: read/write

Data: read/write

Code: read/execute

Buffer overflow

High memory address



Input parameters to the function being executed

RIP (Return Instruction Pointer): Return address to the code that the process will execute after this function is finished

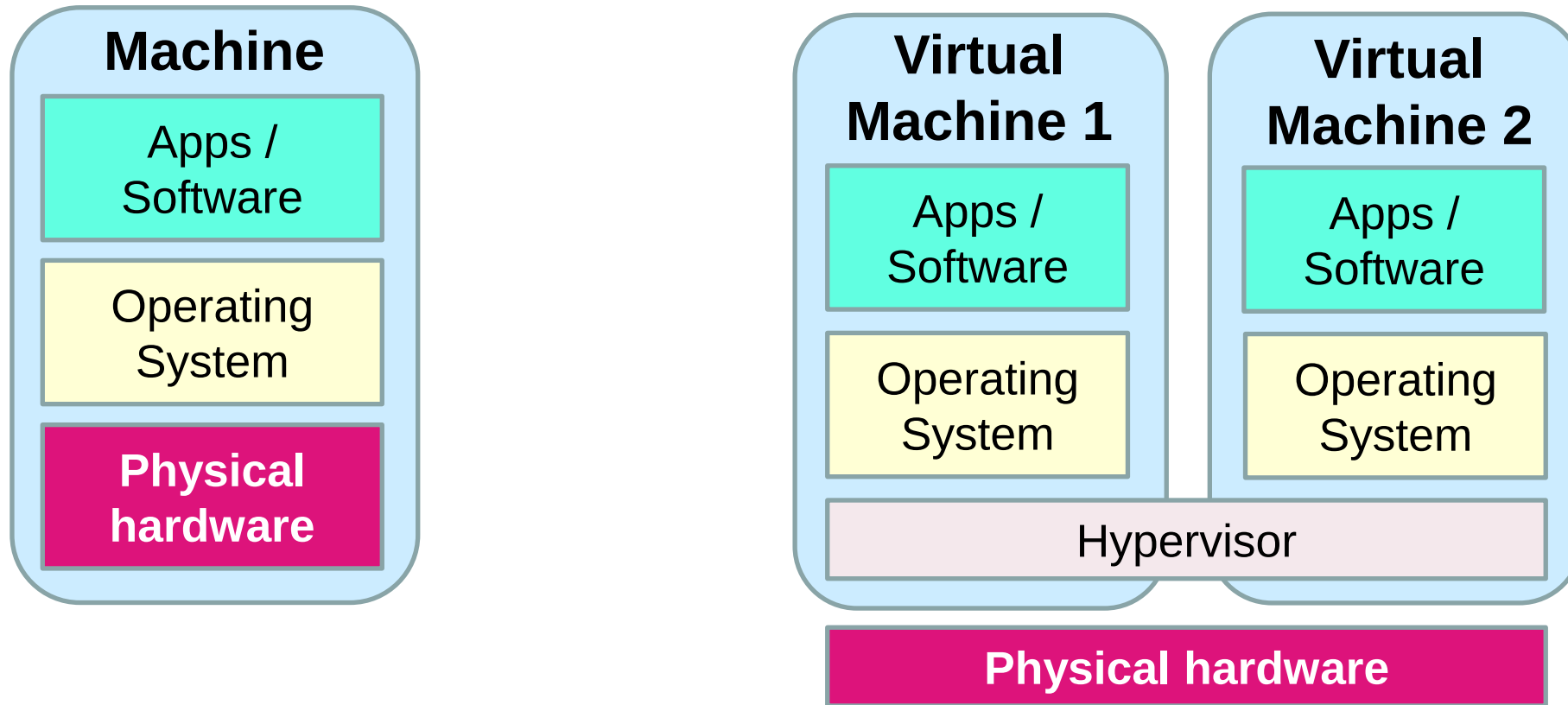
RBP (Return Base Pointer): pointer (address) to the previous frame

Local variables, where each variable has a reserved buffer space of a certain size

Buffer overflow – countermeasures

- No Execute (NX): OS does not allow code to be executed on the stack
- Stack canaries: random value stored elsewhere that checks if data has been overwritten
- Address Space Layout Randomization (ASLR): The attacker needs to know where in memory the attack code to execute is located. ASLR makes it harder to find this address
- Shadow Stack: The microprocessor keeps a parallel copy of pointers on a “shadow stack” which can not be written to by the process. This prevents corruption of pointers.
- Use of (more secure) programming language and static analysis that avoids buffer overflow vulnerability
- Write better code (secure by design / DevSecOps)

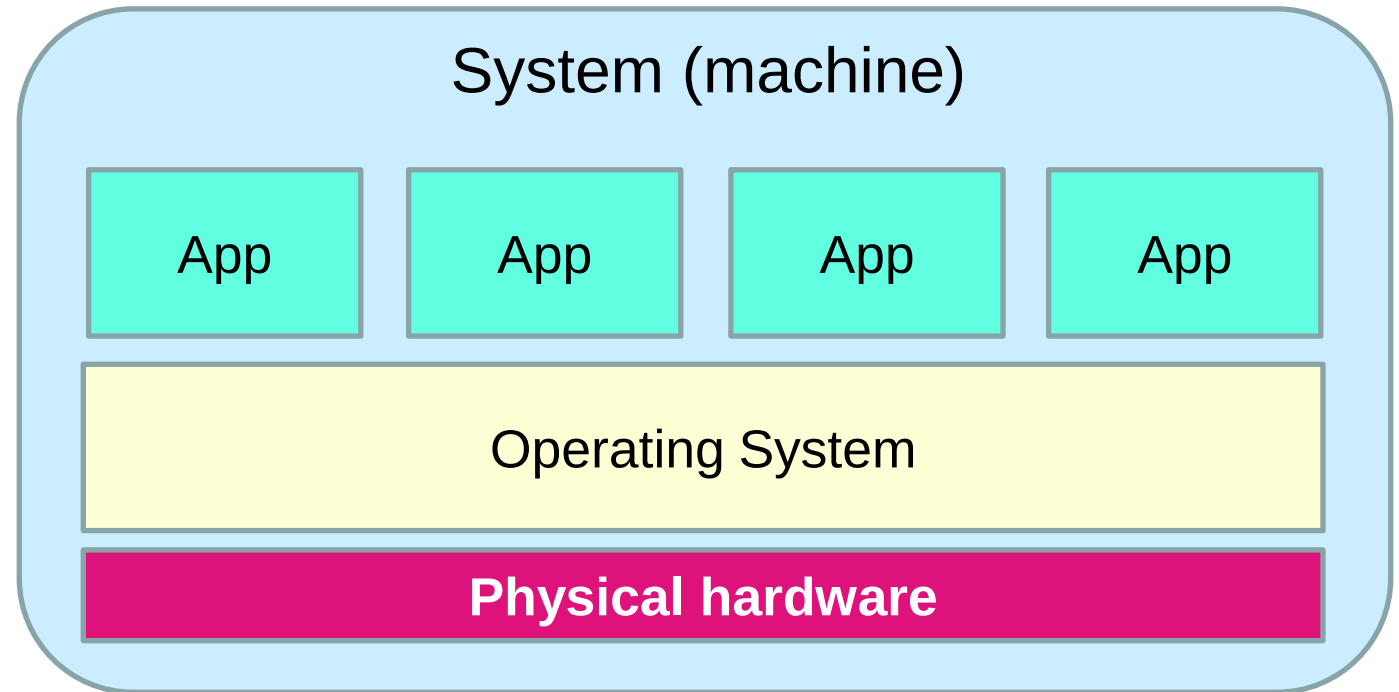
Machine and "virtual machine"



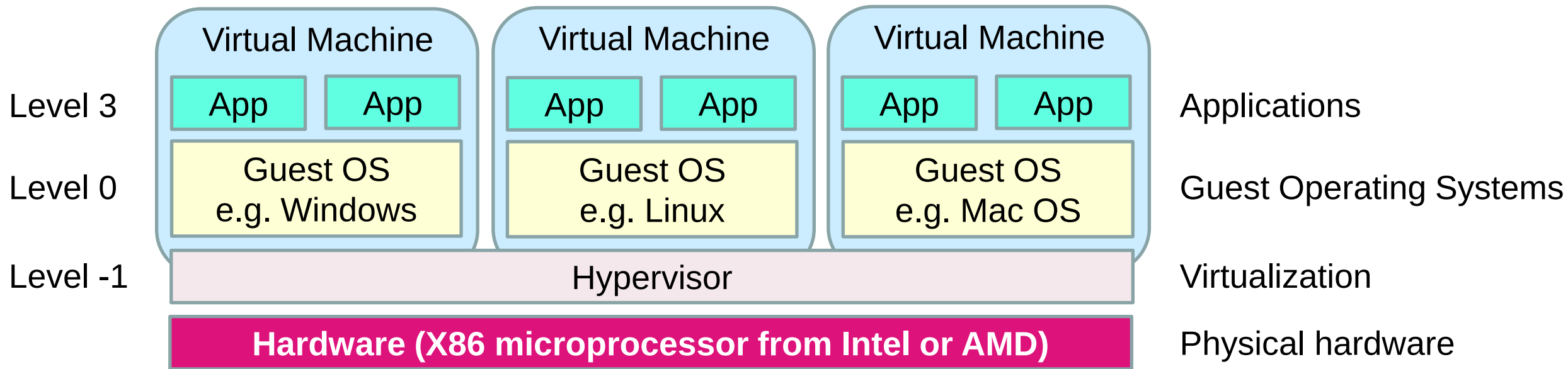
Traditional system stack

Level 3: User Applications

Level 0: OS, Linux root, Win. adm.

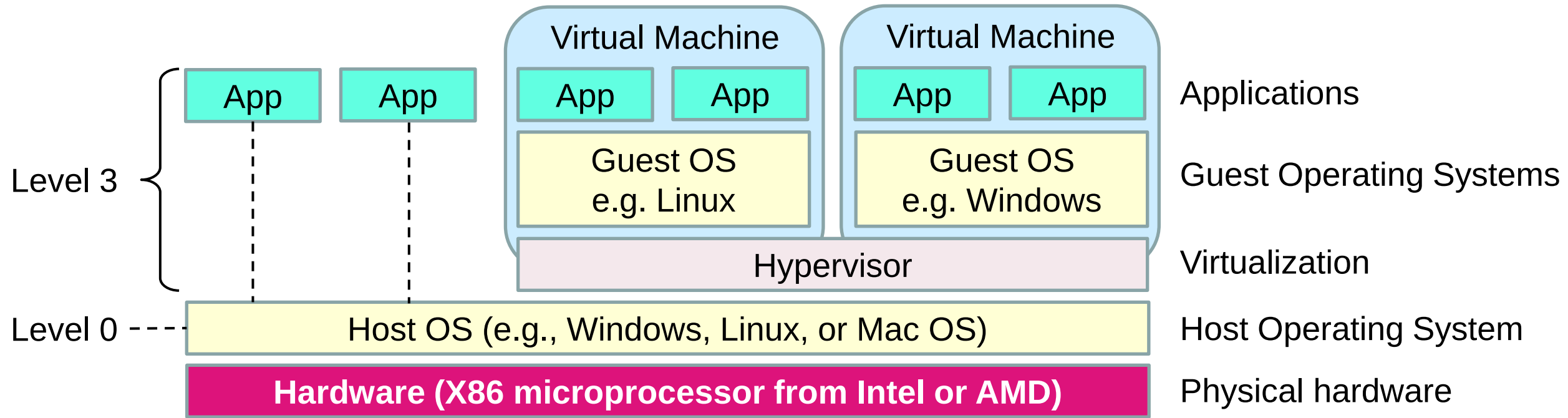


Type 1 virtualization (native)



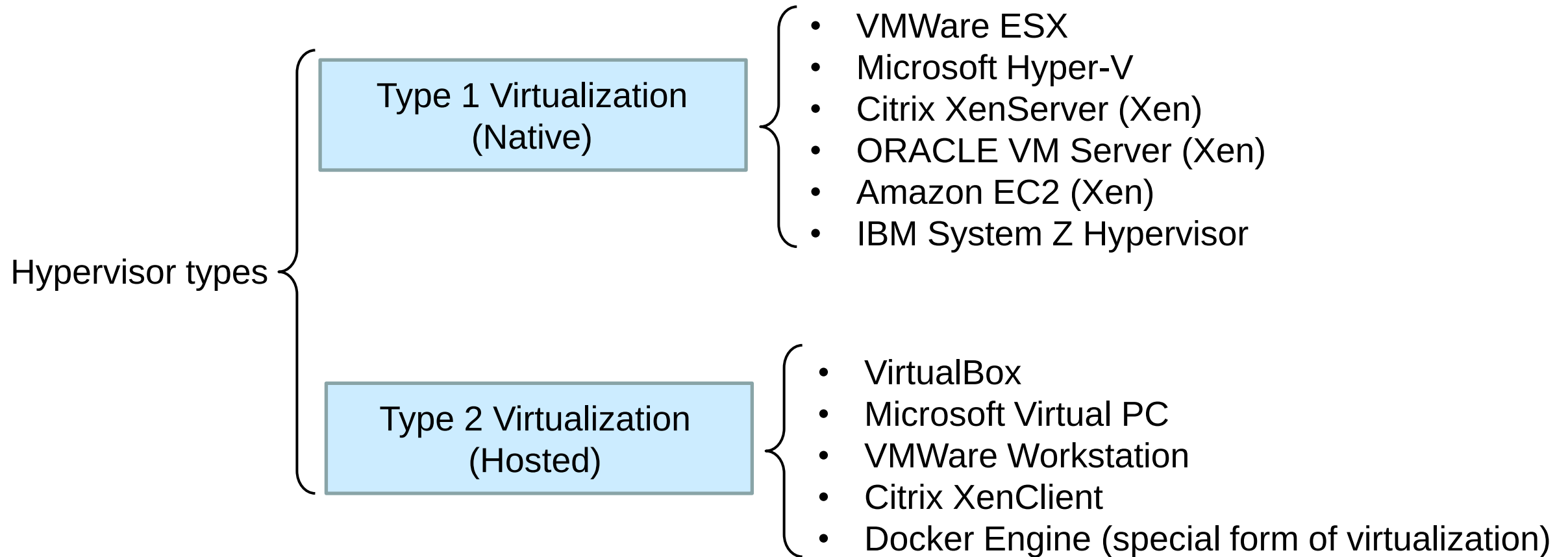
- Type 1 virtualization is the most direct and most efficient way to run virtual machines.
- Hypervisor is software that plays the role of physical hardware.
- Type 1 virtualization provides a logical structure at the privilege levels in that the hypervisor is more privileged than the guest operating systems that it governs.
- The Guest operating systems run at privilege level 0, for which they were precisely designed.

Type 2 Virtualization (Hosted)

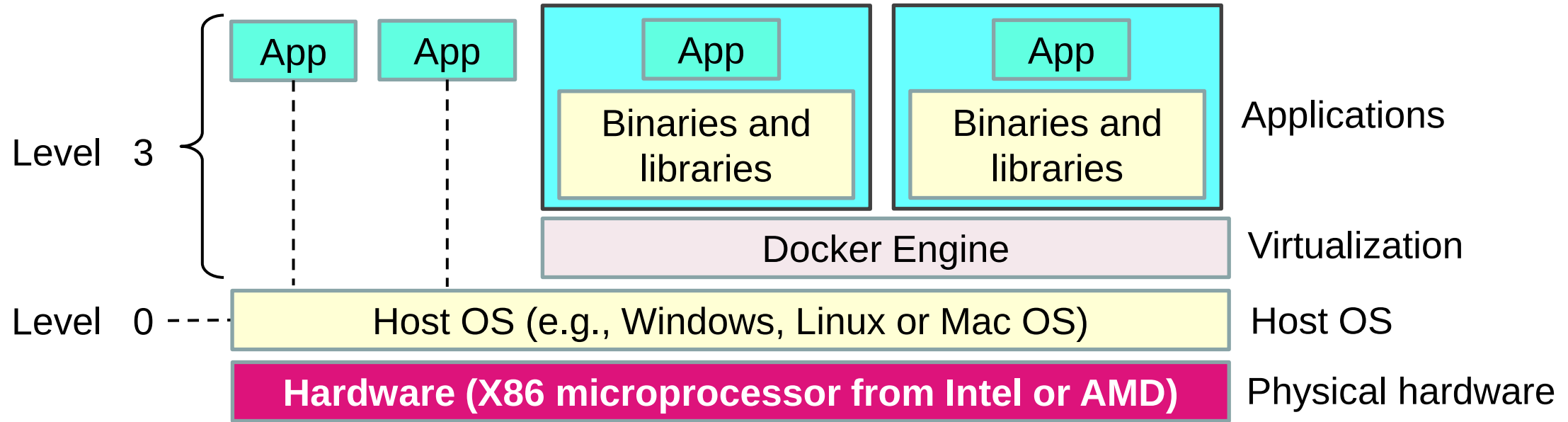


- The hypervisor program is installed as a regular application on the host operating system.
- Easy for regular users to install and use.
- Level -1 is not used: hypervisor and guest OS run as normal Level 3 applications.
- Operating systems need to perform privileged functions that require level 0 (or level -1).
- Privileged functions must be called through the host operating system, which causes additional delay in processing. Therefore, Type 2 is a relatively inefficient virtualization architecture.

Products for virtualization



Docker Engine and contains (hosted)



- Docker Engine enables containerized applications to run regardless of platform.
- Docker Engine is a form of virtualization that makes guest OS unnecessary.
- Eliminates dependencies on OS platforms, because the dependencies are built into the container.
- A container is a piece of software with code and all its dependencies, so that the application runs quickly and reliably, and can be easily ported between different platforms with a Docker Engine.

Use of virtualization

- Cloud providers operate large server parks
 - Each customer gets their own World Cup
 - Many customers share the same hardware
 - Easy to migrate VM between servers to increase/decrease capacity



Amazon EC2 Datasenter

- Testing and software analysis
 - Potentially harmful experiments can be safely conducted in isolated environments
 - Take a snapshot of the current state of the operating system
 - The system can be returned to the snapshot state at any time
 - Malware analysis



Virtualization and security goals

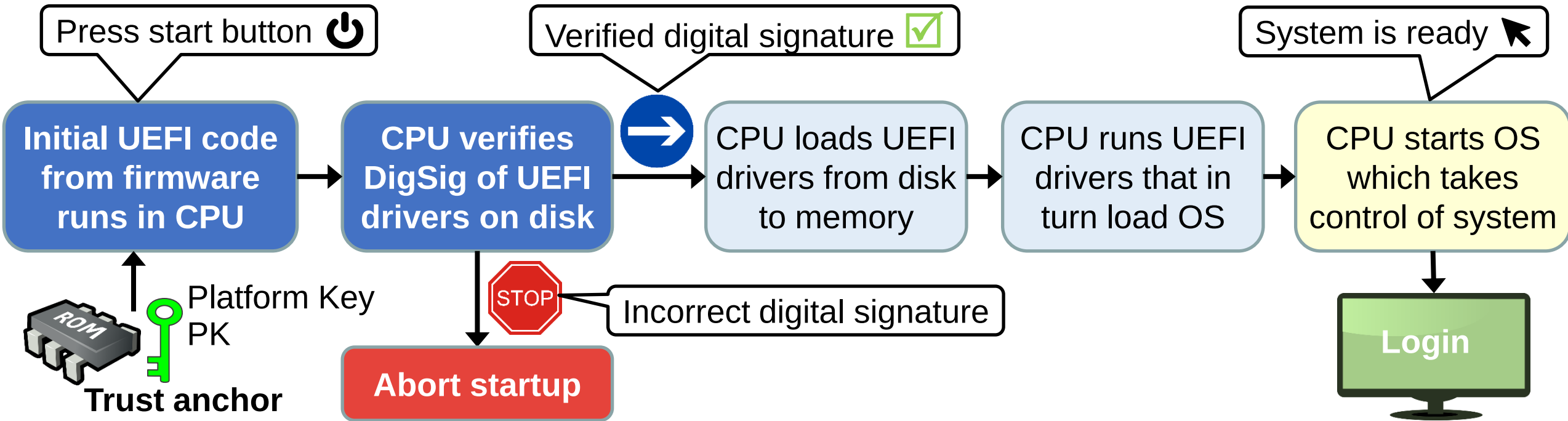
- The guest (operating) systems must not be able to access or be affected by each other
- The guest (operating) systems must not be able to affect the hypervisor program
- The guest (operating) systems must not be able to detect that they are virtualized

Trusted computing

- Trusted computing means that aspects of a system's security are anchored in hardware.
- Hardware is considered more robust against security threats than software.
- Examples:
 - *Secure boot* with UEFI.
 - *TPM (Trusted Platform Module)* is a coprocessor built into many different systems, TPM supports three specific security features: 1) secure boot, 2) remote attestation of secure state to third parties, 3) disk encryption (sealed storage). Secure boot with TPM is different from secure boot with UEFI.
 - *TEE (Trusted Execution Environment)* means that data and computations by a process are protected with hardware technology in the microprocessor, not just by security features in the operating system. Intel SGX (Software Guard Extensions) is an example of such technology.
 - *Tamper-resistant physical encapsulation* is difficult for an attacker to penetrate. An encapsulation is tamper-proof if it can also detect attempts at physical manipulation and, if necessary, automatically delete sensitive information such as cryptographic keys. An example is the IBM 4765 Secure Coprocessor

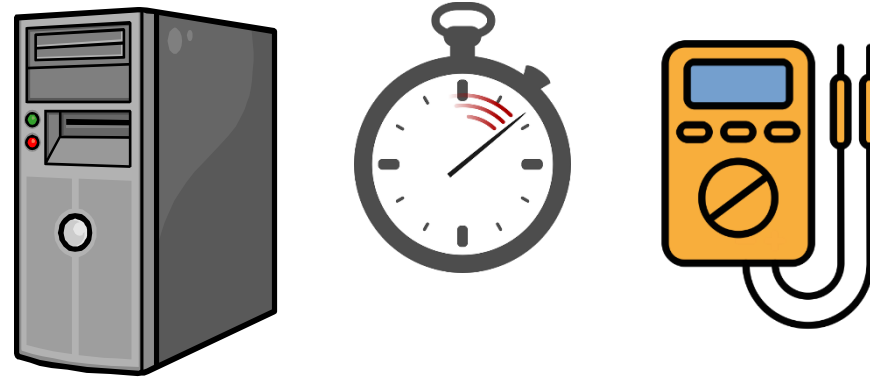


Secure boot with UEFI (*Unified Extensible Firmware Interface*)



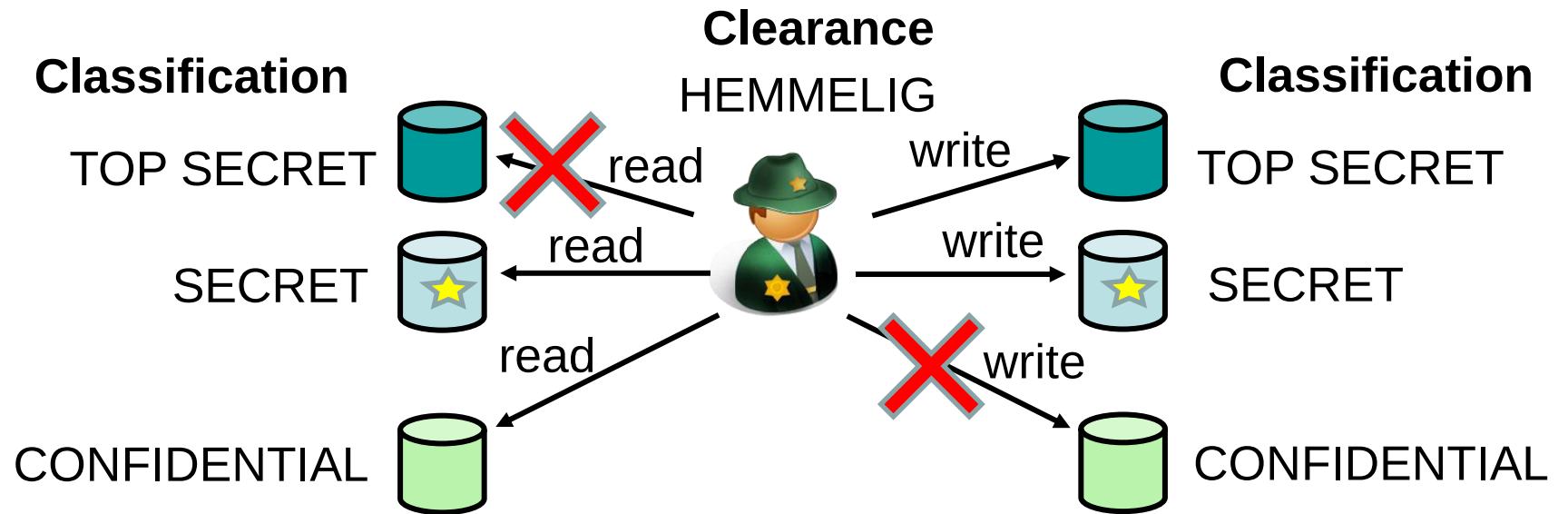
- UEFI replaces the BIOS in modern computers. Controls the boot sequence.
- Program modules for startup are digitally signed by the computer vendor.
- UEFI code in ROM is unsigned but assumed to be correct, initiating the boot sequence.
- Application modules that are loaded are checked for correct digital signatures with Platform Key.
- If an incorrect signature is detected, the boot sequence will be canceled.

Side Channels



- A *side channel* is an unintended channel outside of the regular interface, which is caused by the physical implementation of a system, that can emit sensitive information in a way that violates security policy.
- Examples of side channels are: time delay for instructions in the CPU can say something about value (e.g. of some bits in the encryption key), power consumption, sound, radiation
 - Different delays in processing data can cause information leakage of the data
- Meltdown and Spectre are examples of side channels in microprocessors
 - Modern processors "guess" (possible) next instructions (although they may violate security policies), and data still lives in cache even if not selected
 - The vulnerability can cause other processes to read its data
- The bandwidth (capacity) of hidden channels is typically very limited.

Covert Channels



- A covert channel is a mechanism that is part of the interface, which is not designed for communication, but that can be abused to transmit sensitive information in a way that violates security policy.
- Suppose a user with a SECRET clearance who thus does not have read access "no-read-up" to a file system classified TOP SECRET, but who has write access "write-up" to create new files and write in the same file system. This corresponds to the MAC (Mandatory Access Control) model. This means that the user should also not know which files are already in the file system. If the user tries to create a new TOP SECRET file, and the system gives an error message that "the file name already exists", this represents a covert channel leaking sensitive info.
- The bandwidth (capacity) of covert channels is typically very limited.

End of presentation

