Confidential Computing

- lots problems to address -

Prof. Dr. Christof Fetzer

How to protect data/code/secrets…

- First approximation, we need to ensure that
  - **app** is **correct**,
  - **app** is executed **correctly**,
  - **correct app** is executed,
  - our **assumptions** about attacks (and failures) are correct, and
  - correct user connects to app

Code Integrity

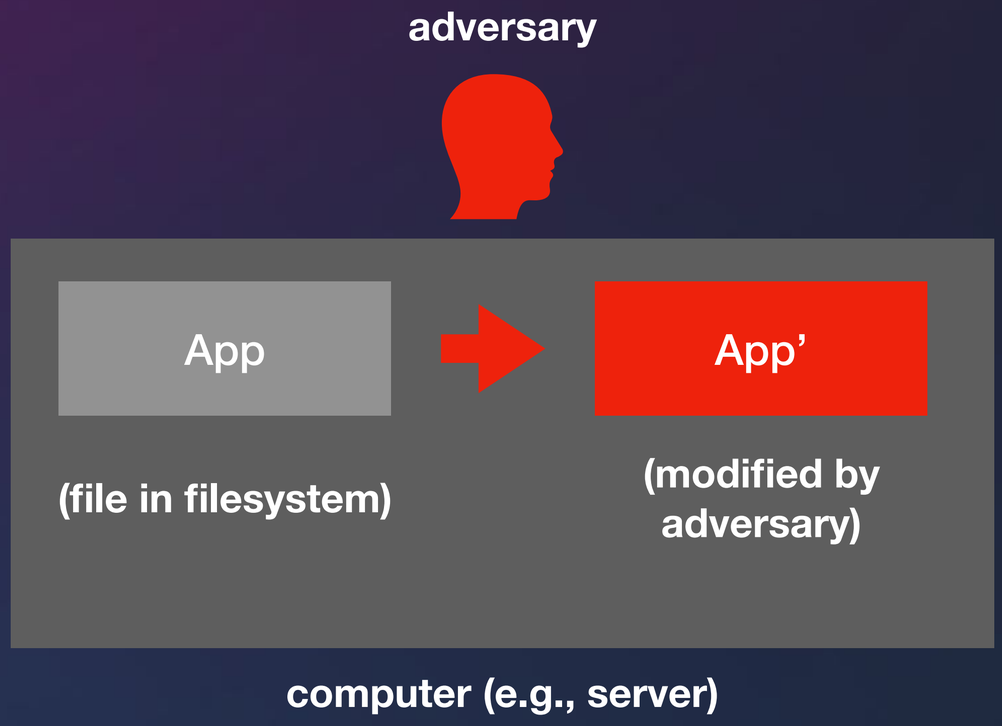
- First approximation, we need to ensure that
  - **app** is **correct**,
  - **app** is executed **correctly**,
  - **correct app** is executed (aka code integrity),
  - our **assumptions** about attacks (and failures) are correct, and
  - correct user connects to app

Code Integrity

- *Ensuring that the code running in a production environment is indeed the correct code created and published by the application developer.*
  - ***No code modification*** *is possible, i.e., even if a hacker has already gained root access to the system cannot modify the code before, during, or after the application has started.*
  - ***Secure code updates****. Attacks like the* ***Solarwind*** *attacks, in which the attacker gained access and altered the software update process to ensure that the backdoors stay open*

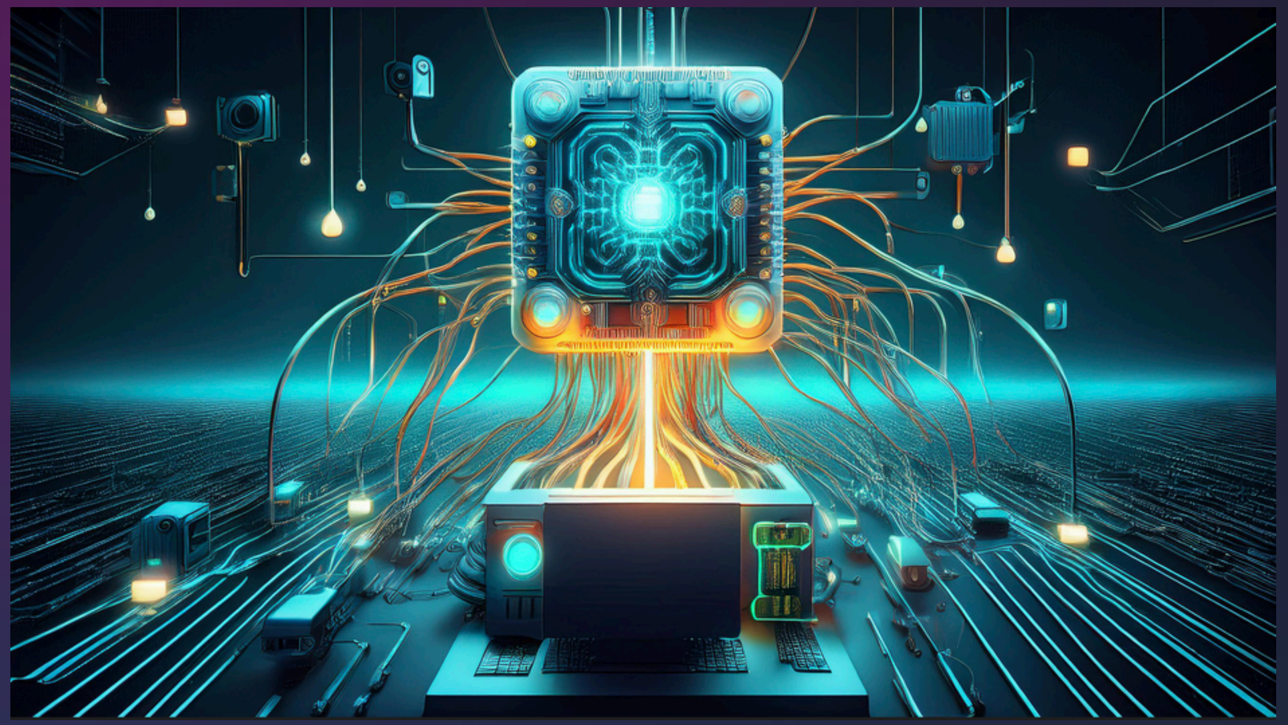
Attack: Replacing Binary

- An adversary
  - might gain access to a system, and
    - replacing/modifying the original program (App)
    - by a malicious variant App'



Virus: a **malware** that replicates itself by modifying other computer programs and inserting its own code into those programs.

Maleware



*„A malware attack is a common cyberattack where **malware** (**malicious software**) executes unauthorized actions on the victim's system. The malicious software encompasses many specific types of attacks such as **ransomware**, **spyware**, **command and control**, and more.“*

Types of Attack

- **Spyware**: gathers data from the device and user, and sends it to third parties without their consent.
- **Ransomware**: denies users access to files on their computer(s): encrypts the files and demanding a ransom payment for the decryption key.
- **Command and control**: a type of attack that involves tools to communicate with and control an infected machine or network.

Protecting the Durability of Data

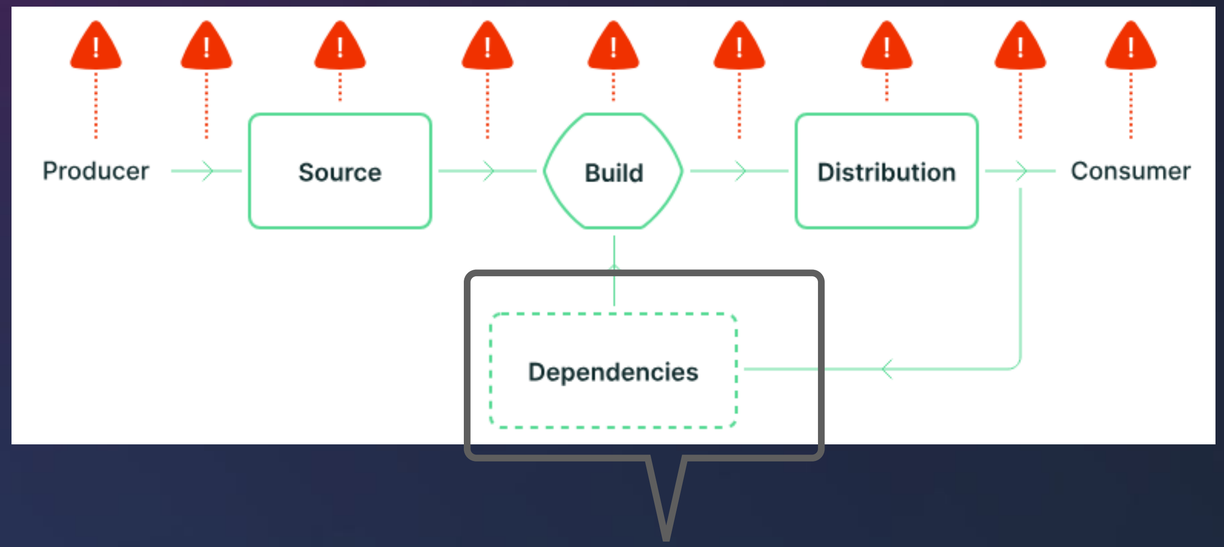
- Protection against **Ransomware**:
  - Some ransomware succeeds in encrypting backups by gaining access to the credentials of the backup system (e.g., object store credentials), or
  - by pushing already encrypted data to the backup.
- Confidential computing can (-> Later Lecture):
  - **protect the credentials** used to backup the data, and
  - **verify the integrity of files** protected by the SCONE file shield during the backup,

To protect integrity of apps

- **State of the art**: system checks integrity of apps during installation:
  - ensuring that software is signed by a trusted entity,
  - the signature matches the downloaded code
- **Problems**:
  - apps could be modified after installation (**runtime attack**)
  - apps could be modified before signing (**supply chain attack**)

Supply Chain Attack

- Supply chain attack:
  - adversary modifies the code of an app before it hits
- We will see in a later lecture:
  - many points at which an adversary could change the code of an app



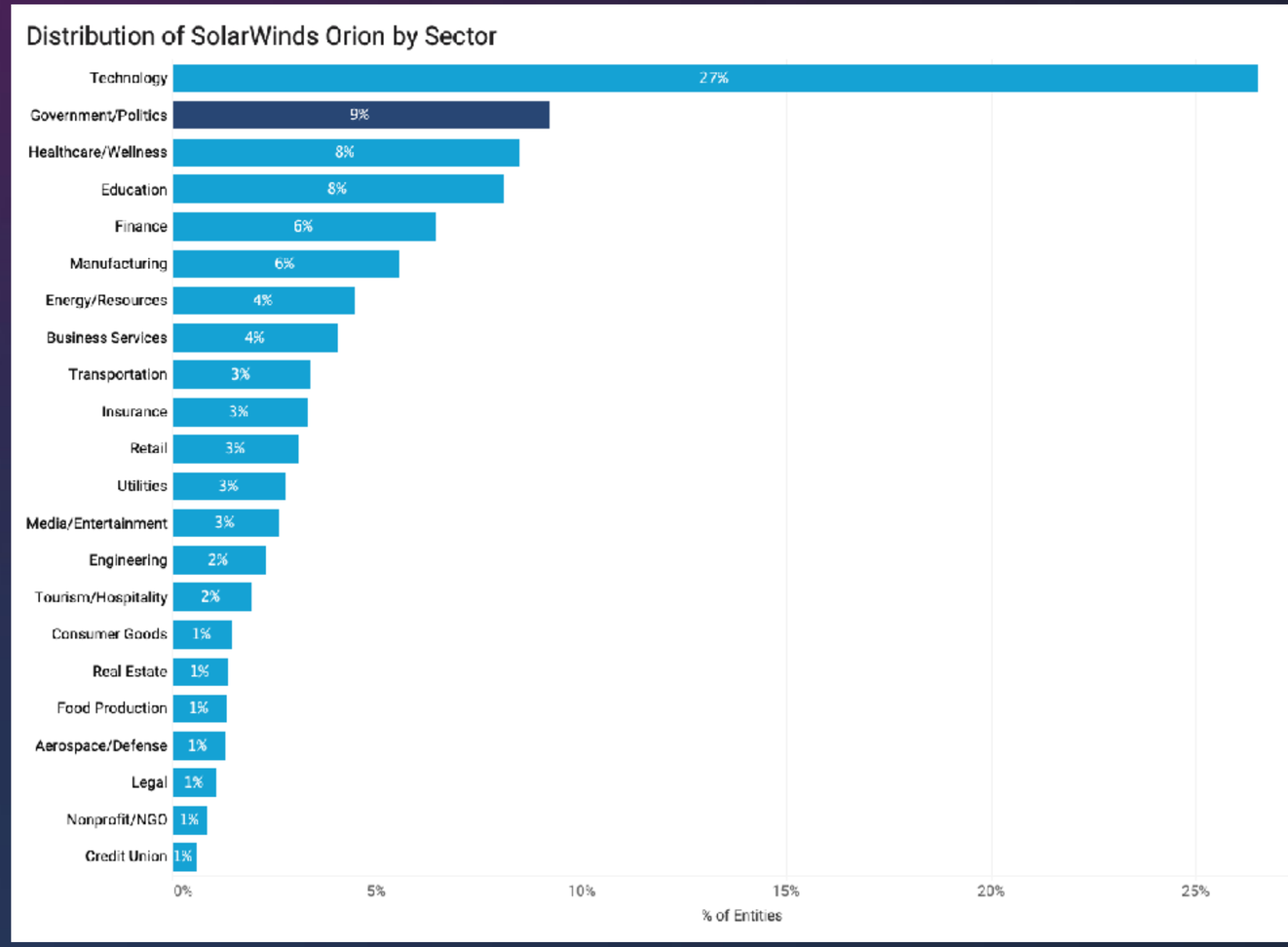
libraries are dependencies - used by virtually all apps

Example: Supply Chain Attack

- **SolarWinds (Orion) attack**:
  - adversaries changed software update system to keep access to the systems
  - adversary kept control of servers for a long time

SolarWinds (Orion) Attack

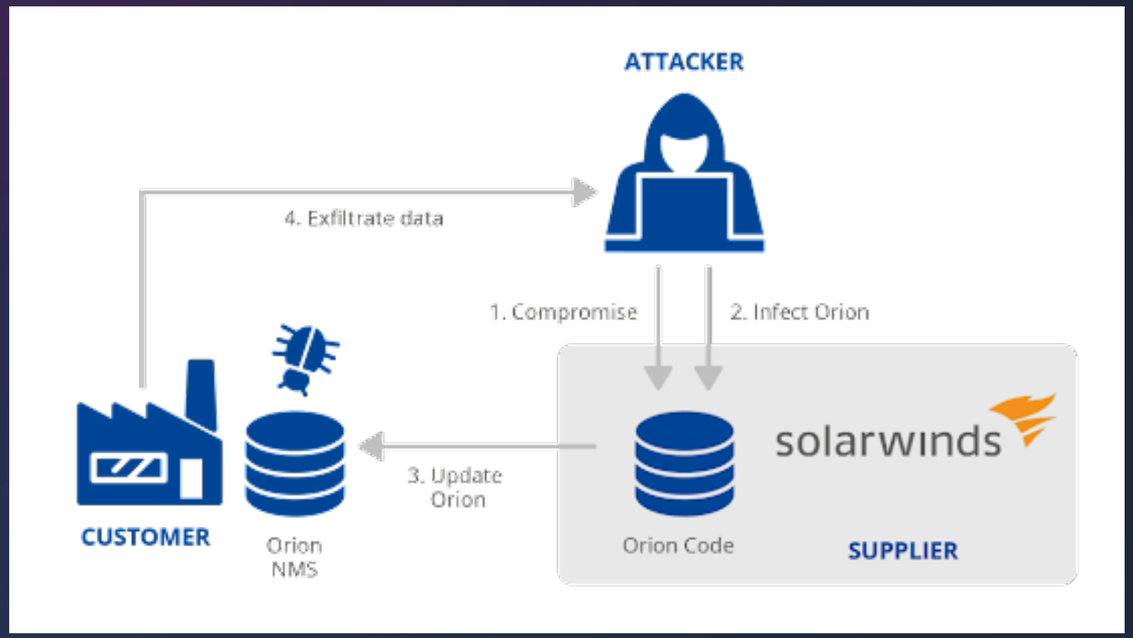
- SolarWinds Orion is an IT **monitoring system**
- **Orion**: **privileged access** to IT systems to obtain **log** and **system performance data**.
- **privileged access & wide deployment**:
  - e.g., used by **governments**
  - => lucrative target for attackers.
- SolarWinds attack is a **supply chain attack**



https://www.bitsight.com/blog/bitsight-analysis-of-solarwinds-orion-part-1-prevalence

Solarwinds Attack

- Attacker compromised the Orion Source Code
- Customers installed the updates of Orion
- Orion installed a backdoor
  - i.e., adversary gained access to data on infected systems
  - software updates kept that backdoor open



https://www.activestate.com/blog/how-to-avoid-becoming-the-next-solarwinds/

Agenda

- How can we detect modifications of application code?
  - modified during runtime?
    - -> **later lecture**: attestation of programs
    - -> **later lecture**: IMA (Linux Integrity Measurement Architecture)
  - modified in the build process?
    - -> **later lecture**: confidential build process

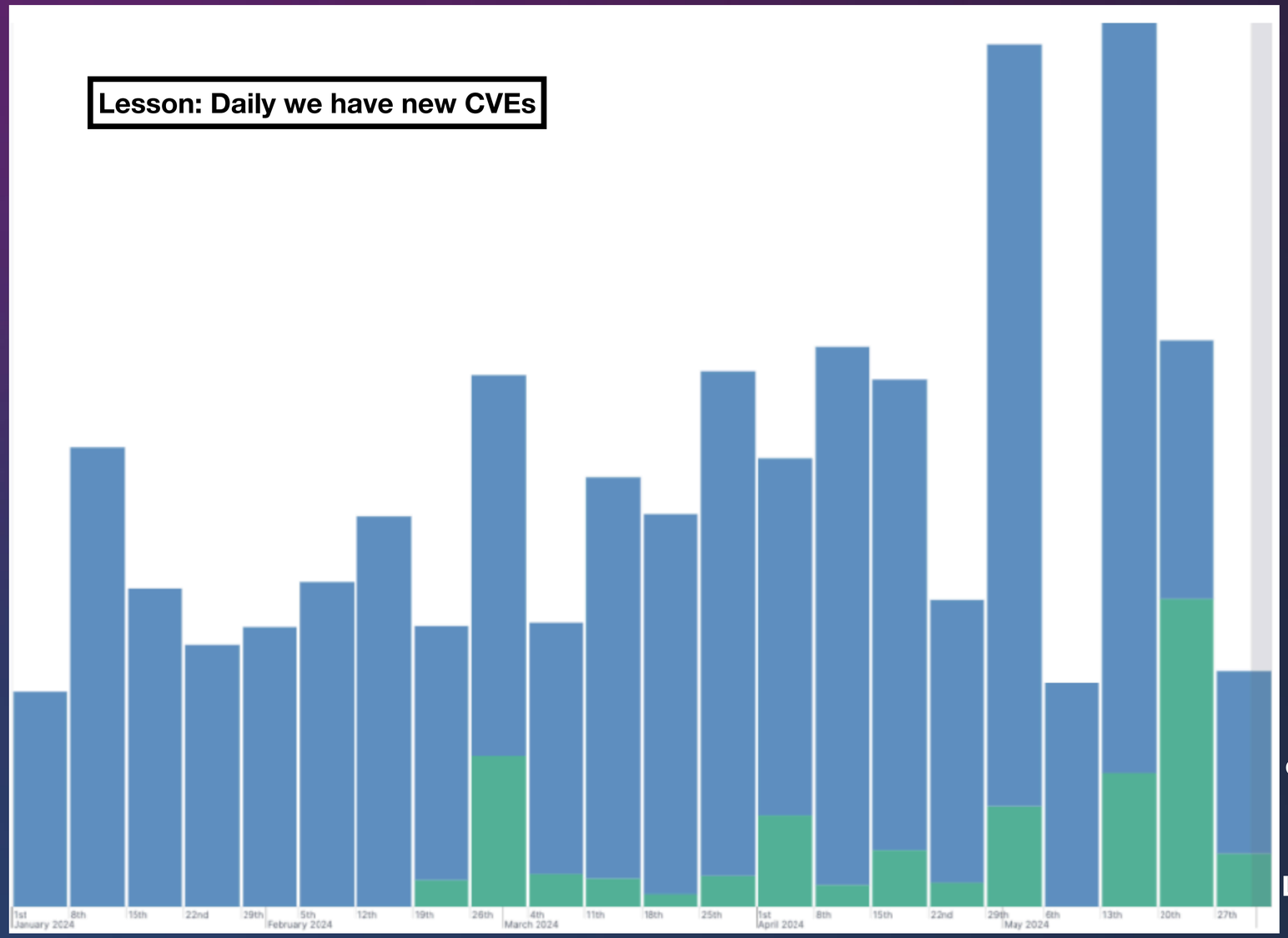
How to protect data/code/secrets…

- We need to ensure that
  - **app** is **correct**,
  - **app** is executed **correctly**,
  - **correct app** is executed (aka code integrity),
  - our **assumptions** about attacks (and failures) are correct, and
  - correct user connects to app

CVEs

- Most programs are NOT correct:
  - they have vulnerabilities
- CVE (Common Vulnerabilities and Exposures):
  - list of publicly disclosed computer security flaws
- CVE system developed in 1999:
  - a standard way for reporting and tracking software security bugs

Example: Linux Kernel CVEs



Linux kernel: about
55 CVEs per week in 2024

other software

Linux kernel

https://www.heise.de/news/Linux-Kritik-Gruende-und-Folgen-der-CVE-Schwemme-im-Kernel-9963793.html

https://opensourcesecurity.io/2024/06/03/why-are-vulnerabilities-out-of-control-in-2024/


Objective: Zero Vulnerabilities
- very difficult to achieve if we have lots of dependencies —

Agenda

- Later lecture:
  - we will look at CVEs and SBOMs (Software Bill of Materials)
- Later lecture:
  - we will look at attestation to detect vulnerabilities in the Trusted Computing Base (TCB), i.e., all parts that need to correct to protect the application
- Later lecture:
  - we will look at isolation using TEEs (Trusted Execution Environments) to exclude some vulnerabilities

Isolation: Air-Gapped Deployment

- We can ensure **virtually** or **physically air-gapped** deployment of applications
  - we can verify the security without needing Internet connectivity for
    - **deployment**, or
    - **attestation**
- Scenarios (discussed in later lecture)**:**
  - application owner runs app in an air-gapped environment, or
  - **customer** of application owner runs the app to prevent app owner to extract information

Protection against Insider Attacks

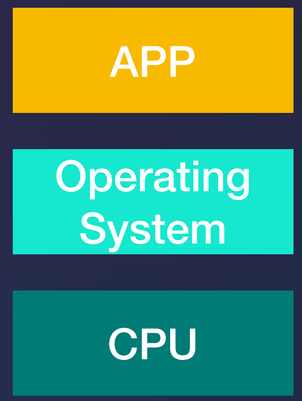
- **App** has configuration parameters:
  - determines the behavior (and hence, correctness) of App
- Later lecture: A **board of governors** can protect any **code**, **policy**, or **configuration** updates:
  - a single administrator cannot change policy
  - The board of governors protects against insider attacks and attackers that have gained root access to the production system.

How to protect data/code/secrets…

- We need to ensure that
  - **app** is correct,
  - **app** is executed **correctly**,
  - **correct app** is executed (aka code integrity),
  - our **assumptions** about attacks (and failures) are correct, and
  - **correct user** connects to app

Can we trust execution CPU?

- App is executed by CPU
- CPU consists of hardware and firmware
  - CPU is complex > 100 B transistors
  - CPU Hardware might be incorrect
  - CPU Firmware might be incorrect
- APP might not be directly executed by CPU
  - e.g., might be executed by a simulated CPU injected by operating system



Approach: Cyber-Hygiene

- Ensure that the **latest application code** and the latest version of the **trusted computing base** (TCB) are executing using remote attestation:
  - The firmware of the CPU is measured and verified,
  - All components participating in the attestation are attested and verified, and
  - All application components are measured and verified.
  - The Trusted Computing Base and all components of the confidential application are checked for CVEs (i.e., if there are any known vulnerabilities) when any process starts.

Agenda

- Later lecture:
  - we will look at DCAP (Data Center Attestation Protocol) to attest and verify the TCB of applications

How to protect data/code/secrets…

- We need to ensure that
  - **app** is correct,
  - **app** is executed **correctly**,
  - **correct app** is executed (aka code integrity),
  - our **assumptions** about attacks (and failures) are correct, and
  - **correct user** connects to app

Confidential Zero Trust Architecture

- We look at **zero-trust architecture** that includes policy engines and policy enforcement
  - trust is established via measurements and policy enforcement that user's device(s) can be trusted
  - isolation of components to maintain trust in components

Agenda

- Later lecture:
  - zero trust approach to authenticate user’s and their devices
- Later lecture:
  - zero trust approach to establish trust into the resources an application uses

Objective: Confidential Computing

- **Confidential computing** protects the services of an application
  - **at Rest**,
  - **in Transit**, and
  - **in Use**.
- Question: how strong is our adversary?
  - e.g., we need to enforce **confidentiality**, **integrity**, and **consistency** even if adversaries have gained root access to a system