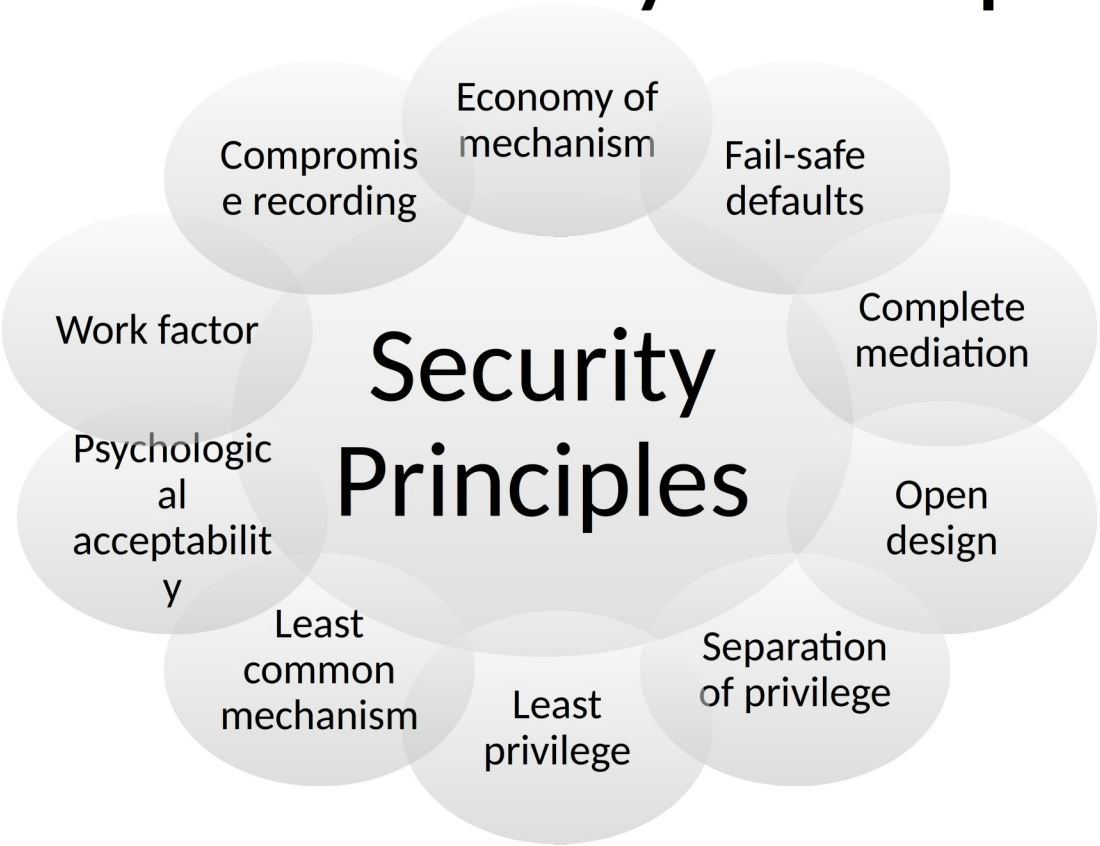
University of BRISTOL

# Computer System B -Security

Sanjay Rawat

sanjay.rawat@bristol.ac.uk

# The Ten Security Principles



# Economy of mechanism

- This principle stresses **simplicity** in the **design** and **implementation** of security measures.
  - While applicable to most engineering endeavors, the notion of simplicity is especially important in the security domain, since a simple security framework facilitates its understanding by developers and users and enables the efficient development and verification of enforcement methods for it.

# Fail-safe defaults

- This principle states that the default configuration of a system should have a **conservative protection scheme**.
  - For example, when adding a new user to an operating system, the default group of the user should have minimal access rights to files and services. Unfortunately, operating systems and applications often have default options that favor usability over security.
  - This has been historically the case for a number of popular applications, such as web browsers that allow the execution of code downloaded from the web server.

# Complete mediation

- The idea behind this principle is that every access to a resource must be checked for **compliance with a protection scheme**.
  - As a consequence, one should be wary of performance improvement techniques that save the results of previous authorization checks, since permissions can change over time.
  - For example, an online banking web site should require users to sign on again after a certain amount of time, say, 15 minutes, has elapsed.

# Open design

- According to this principle, the security architecture and **design** of a system should be made **publicly available**.
  - Security should rely only on keeping cryptographic keys secret.
  - Open design allows for a system to be scrutinized by multiple parties, which leads to the early discovery and correction of security vulnerabilities caused by design errors.
  - The open design principle is the opposite of the approach known as **security by obscurity,** which tries to achieve security by keeping cryptographic algorithms secret and which has been historically used without success by several organizations.

# Separation of privilege

- This principle dictates that **multiple conditions** should be required to achieve access to restricted resources or have a program perform some action.

# Least privilege

- Each program and user of a computer system should operate with the bare **minimum privileges necessary** to function properly.
  - If this principle is enforced, abuse of privileges is restricted, and the damage caused by the compromise of a particular application or user account is minimized.
  - The military concept of **need-to-know** information is an example of this principle.

# Least common mechanism

- In systems with multiple users, mechanisms allowing resources to be **shared by more than one user should be minimized**.
  - For example, if a file or application needs to be accessed by more than one user, then these users should have separate channels by which to access these resources, to prevent unforeseen consequences that could cause security problems.

# Psychological acceptability

- This principle states that user interfaces should be **well designed and intuitive**, and all security-related settings should adhere to what an ordinary user might expect.

# Work factor

- According to this principle, the **cost of circumventing** a security mechanism should be compared with the resources of an attacker when designing a security scheme.
  - A system developed to protect student grades in a university database, which may be attacked by snoopers or students trying to change their grades, probably needs less sophisticated security measures than a system built to protect military secrets, which may be attacked by government intelligence organizations.

# Compromise recording

- This principle states that sometimes it is more desirable to **record the details** of an intrusion than to adopt more sophisticated measures to prevent it.
  - Internet-connected surveillance cameras are a typical example of an effective compromise record system that can be deployed to protect a building in lieu of reinforcing doors and windows.
  - The servers in an office network may maintain logs for all accesses to files, all emails sent and received, and all web browsing sessions.

# Efficiency and Usability

- Bringing in security is not cheap, e.g.
  - Crypto involves extra computation
    - Symmetric vs asymmetric crypto (we'll talk about)
  - Boundary checks (implementation)
  - Sanitization
- Historically, security was compromised for performance.
- Modern view is to seek balance

# Conti...

- Security related features should not make applications hard to use, e.g.
  - Android Apps Permission, cookies consent, prompt on every action, …
  - Hard to use solutions tend to encourage users to either stop using the application or start finding some work-around (which may also result in a solution-- security by obscurity!)
- Single sign-on (SSO) is a solution that considers such aspects (but it also has its own trade-offs).

# Conti...

- Security related features should not make applications hard to use, e.g.
  - Android Apps Permission, cookies consent, prompt on every action, …
  - Hard to use solutions tend to encourage users to either stop using the application or start finding some work-around (which may also result in a solution-- security by obscurity!)
- Single sign-on (SSO) is a solution that considers such aspects (but it also has its own trade-offs).

Linus Torvalds “*Those security people are f*cking morons*.”

# Conti...

- Security related features should not make applications hard to use, e.g.
  - Android Apps Permission, cookies consent, prompt on every action, …
  - Hard to use solutions tend to encourage users to either stop using the application or start finding some work-around (which may also result in a solution-- security by obscurity!)
- Single sign-on (SSO) is a solution that considers such aspects (but it also has its own trade-offs).

Linus Torvalds "*Those security people are f*cking morons*."
(http://lkml.iu.edu/hypermail/linux/kernel/1711.2/01701.html)

# Social Engineering

- **Pretexting**: creating a story that convinces an administrator or operator into revealing secret information.
- **Baiting:** offering a kind of “gift” to get a user or agent to perform an insecure action. e.g. malware (specially Trojan horse)
- **Quid pro quo:** offering an action or service and then expecting something in return.
- Phishing is more recent term used to refer several of such activities (we will talk about them later in the course)

# Vulnerabilities from Programming Errors

- Secure and security features
  - Proper use of Crypto APIs
  - Programming bugs
- Design vs. programming flaws
  - Programming error , like buffer overflow (we will talk a lot about them later)
  - Design flaw, like insecure information flow or weak access control
- All lead to an insecure application!