COMP3028
Computer Security

Lecture 2

Foundations of Computer Security

This Lecture

- A more detailed look at computer security
- Security Strategies
- The Fundamental Dilemma
- Data vs. Information
- Principles of Computer Security Design
- Contemporary Computer Security

General Security

- As we mentioned last lecture, security is about the protection of assets
  - Assets might be physical, but they could also be data, information, even ideas
- Protection Measures
  - Prevention
  - Detection
  - Reaction – manual or automatic

Example – E-commerce

- Prevention: encrypt your orders, rely on the merchant to perform checks on the caller, don't use the Internet?
- Detection: an unauthorized transaction appears on your credit card statement.
- Reaction: complain, ask for a new card number, etc.
- Footnote: Your credit card number has not been stolen; your card can be stolen, but not the number

Managing Security

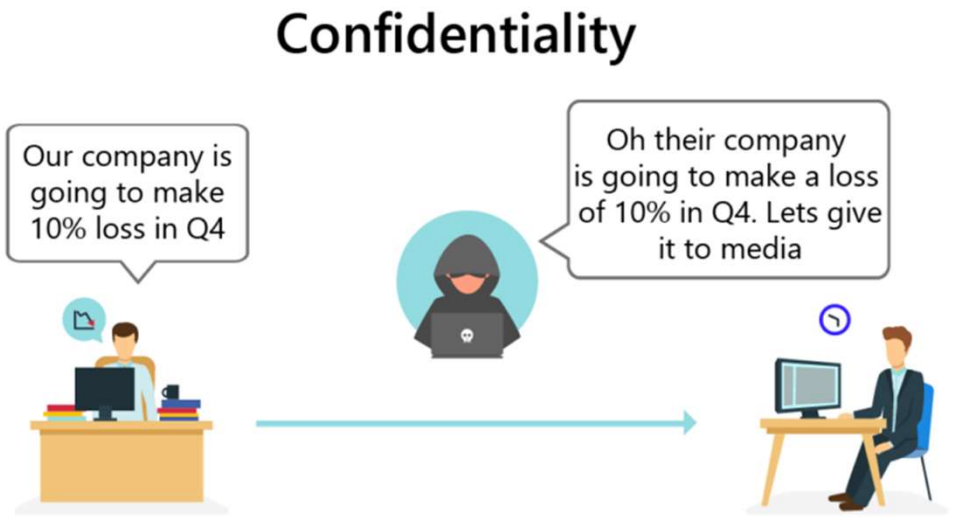
- Within organisations, management are responsible for defining security needs
- Developers implement these policies
- A concise document explaining the needs is called a ***Security Policy***
  - *What should be protected?*
  - *How should we protect it?*

Security Strategies

- Three key areas (**CIA**):
- **Confidentiality**: prevent unauthorised disclosure of information
- **Integrity**: prevent unauthorised modification of information
- **Availability**: prevent unauthorised withholding of information or resources

Confidentiality

- The prevention of unauthorised users reading private or secret information
- **Privacy** –The protection of personal data
- Examples:
  - Medical records
  - Transfer of credit card details



Integrity

- The prevention of unauthorised modification of data, and the assurance that data remains unmodified
- Examples:
  - Distributed bank transactions
  - Database records

Authenticity

- Just because we have integrity does not mean we have authenticity.
  - Can we verify the sender?
  - Can we have freshness?
- Authenticity = Integrity + freshness
- Freshness is pretty important in bank transactions!

"On the Internet, nobody knows you're a dog."

Availability

- The property of being accessible and useable upon demand by an authorised entity
- In other words, we want to prevent **denial of service** (DoS)
- Examples:
  - Redundant power supplies
  - Firewall packet filtering

Accountability

- Users should be held responsible for their actions
- The system should identify and authenticate users and ensure compliance
- Audit trails must be kept
- In distributed systems, cryptographic non-repudiation mechanisms can be used to achieve the same goal.

Non-repudiation

- Provides un-forgeable evidence that someone did something
- Evidence verifiable by a trusted third party
  - E.g. Notaries, Digital Certificates
- Applies to physical security: e.g. keycards
- Mostly a legal concept

Reliability

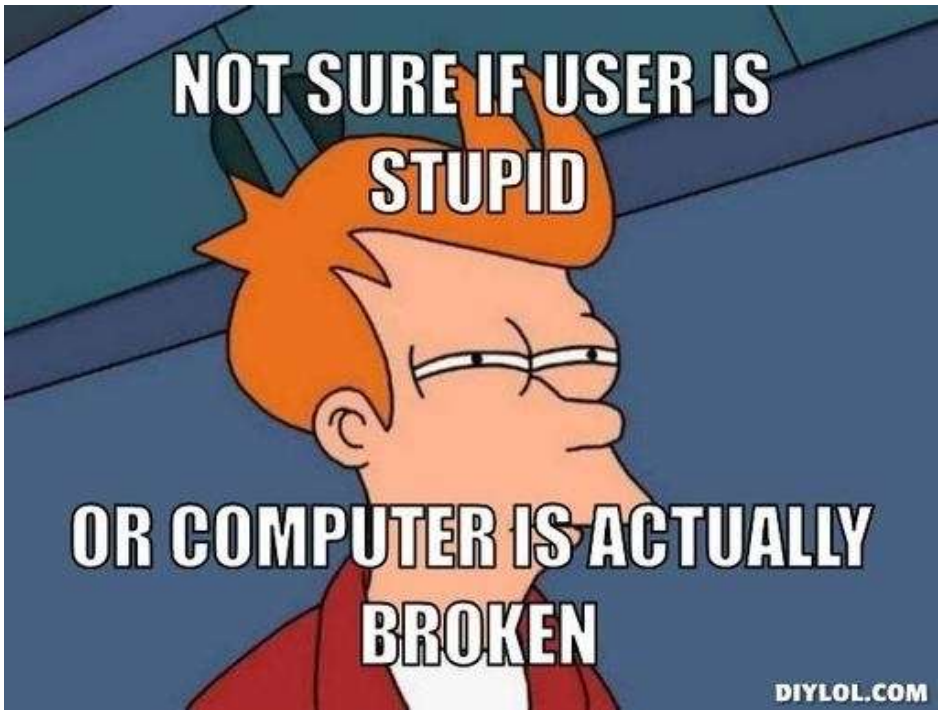
- Reliability – (accidental) failure
- Security is an aspect of reliability, and vice versa
- Safety – impact of the system failures on their environment
- Dependability

  "The property of a computer system such that reliance can justifiably be placed on its service"

Fundamental Dilemma

“Security-unaware users have specific security requirements but no security expertise”

- If you provide your customers with a standard solution it might not meet their requirements.
- If you want to tailor your solution to your customers’ needs, they may be unable to tell you what they require.



Data vs Information

- Security is generally seen as controlling access to information
- This is hard, we usually control access to data instead
- Data – A means to represent information
- Information – An interpretation of that data

Inference

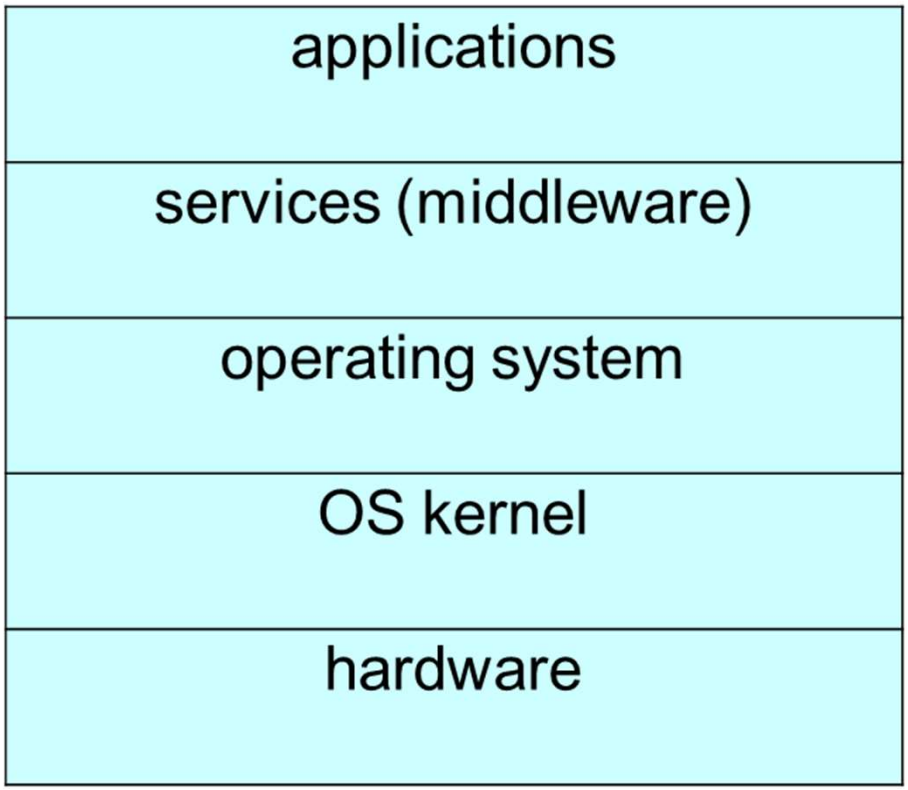
- Focusing on data can still leave information vulnerable
- Consider a medical database
  - Medical records cannot be queried
  - Aggregates like prescription totals can be
- Carefully chosen queries can narrow down who has what conditions

1st Fundamental Design Decision

- **Where to focus security controls?**
- The focus may be on **data – operations – users**; e.g. integrity requirements may refer to rules on
  - Format and content of **data items** (internal consistency): account balance is an integer.
  - **Operations** that may be performed on a data item: credit, debit, transfer, …
  - **Users** who are allowed to access a data item (authorised access): account holder and bank clerk have access to account.

2nd Fundamental Design Decision

- **Where to place security controls?**

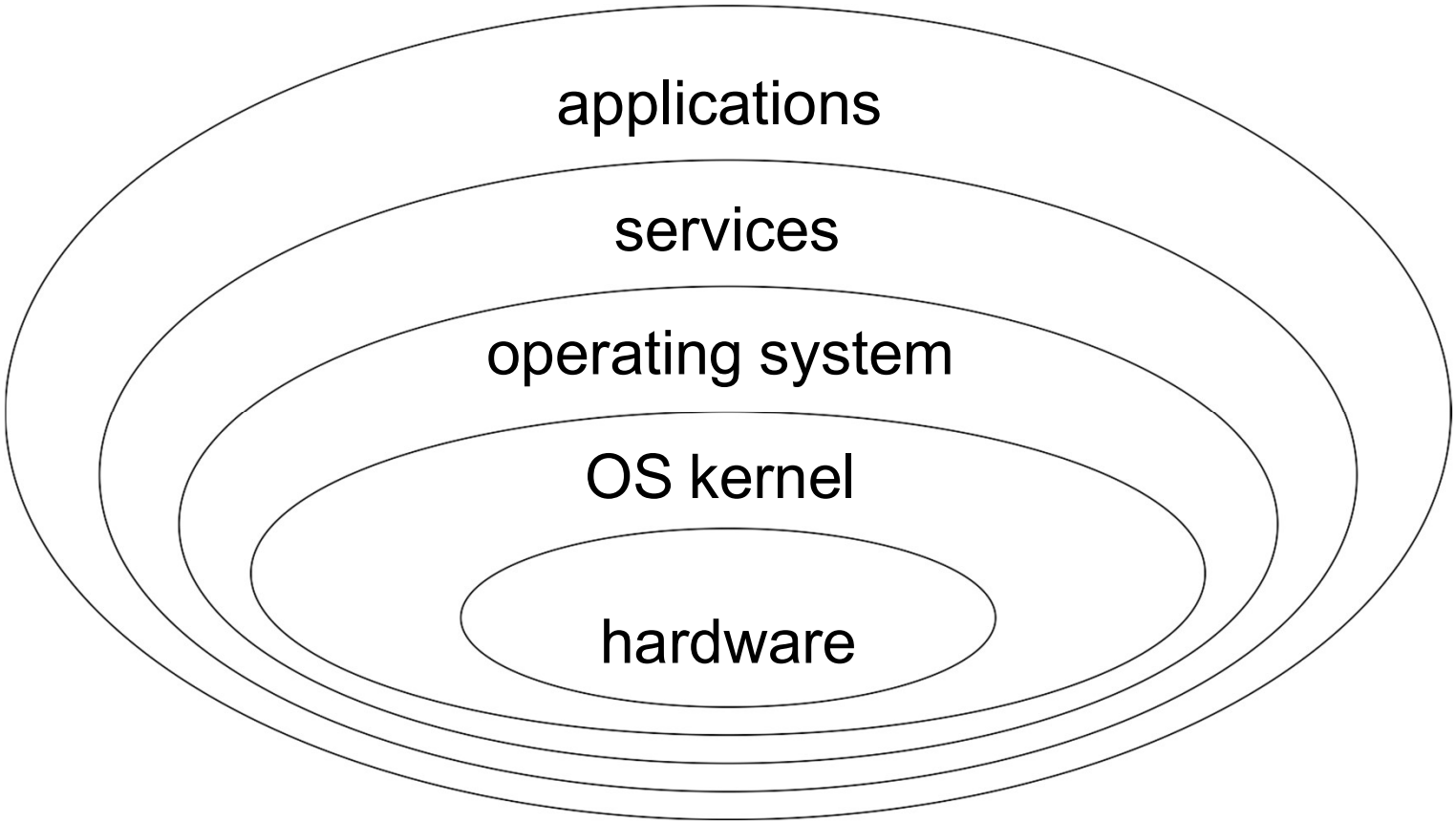


Man-Machine Scale

- Visualize security mechanisms as concentric **protection rings**, with hardware mechanisms in the centre and application mechanisms at the outside.
- Mechanisms towards the centre tend to be more generic while mechanisms at the outside are more likely to address individual user requirements.
- The **man-machine scale** for security mechanisms combines our first two design decisions.

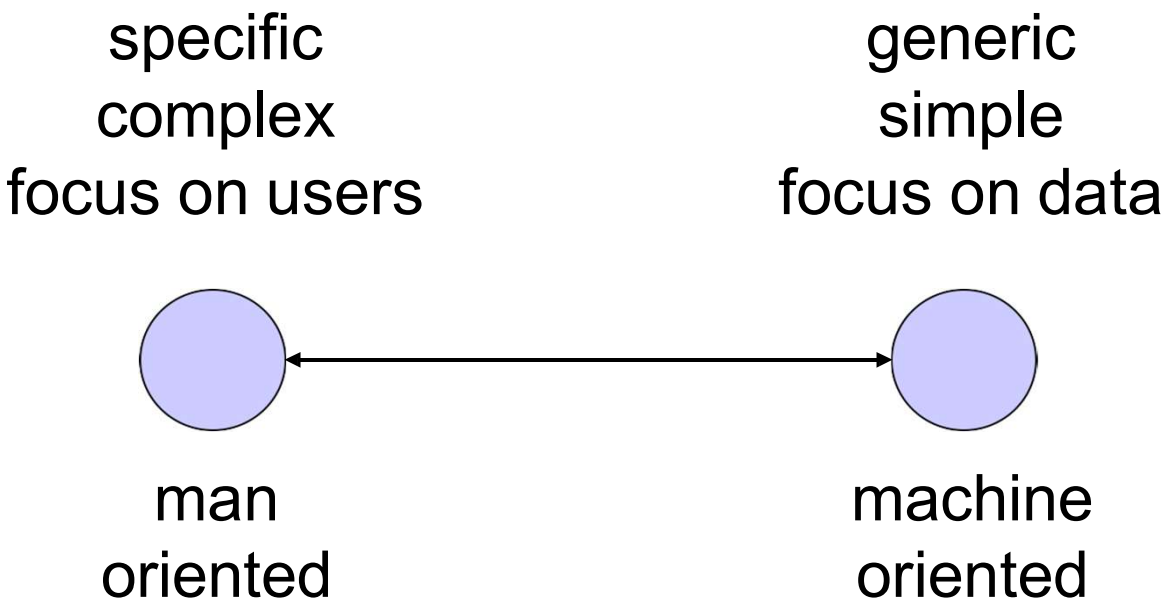
Onion Model of Protection

- Each layer protects a boundary, and relies on the security of the layers below



Man-Machine Scale

- By combining the first two design decisions:



3rd Fundamental Design Decision

- **Complexity or Assurance?**
- Often, the location of a security mechanism on the man-machine scale is related to its complexity.
- Generic mechanisms are simple, applications clamour for **feature-rich** security functions.
- **Do you prefer simplicity – and higher assurance – to a feature-rich security environment?**

3rd Fundamental Design Decision

- Fundamental dilemma:
- Simple generic mechanisms may not match specific security requirements.
- To choose the right features from a rich menu, you have to be a security expert.
- Security unaware users are in a no-win situation.
- **Feature-rich security and high assurance do not match easily.**

4th Fundamental Design Decision

- **Centralized or decentralized control?**
- Within the domain of a security policy, the same controls should be enforced.
- Having a single entity in charge of security makes it easy to achieve uniformity but this central entity may become a performance bottleneck.
- A distributed solution may be more efficient but you have to take added care to guarantee that different components enforce a consistent policy.
- **Should a central entity define and enforce security or should these tasks be left to individual components in a system?**

Security Perimeter

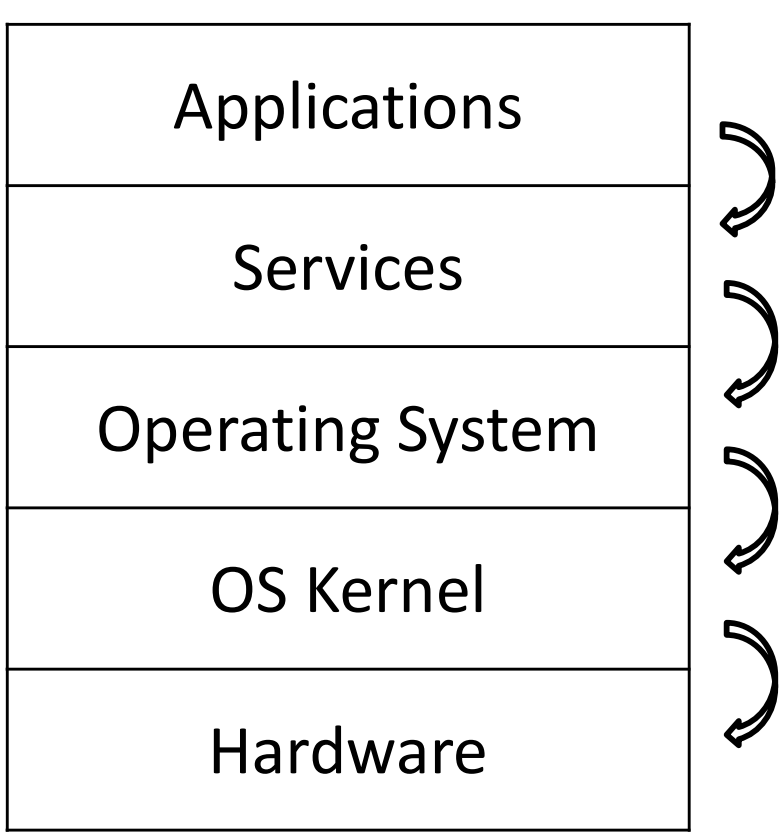
- Every protection mechanism defines a **security perimeter** (**security boundary**).
- The parts of the system that can malfunction without compromising the mechanism lie outside the perimeter.
- The parts of the system that can disable the mechanism lie within the perimeter.
- Note: Attacks from insiders are a major concern in security considerations.

5th Fundamental Design Decision

- Blocking access to the layer below
- Attackers try to bypass protection mechanisms.
- There is an immediate and important corollary to the second design decision:
- **How do you stop an attacker from getting access to a layer below your protection mechanism?**

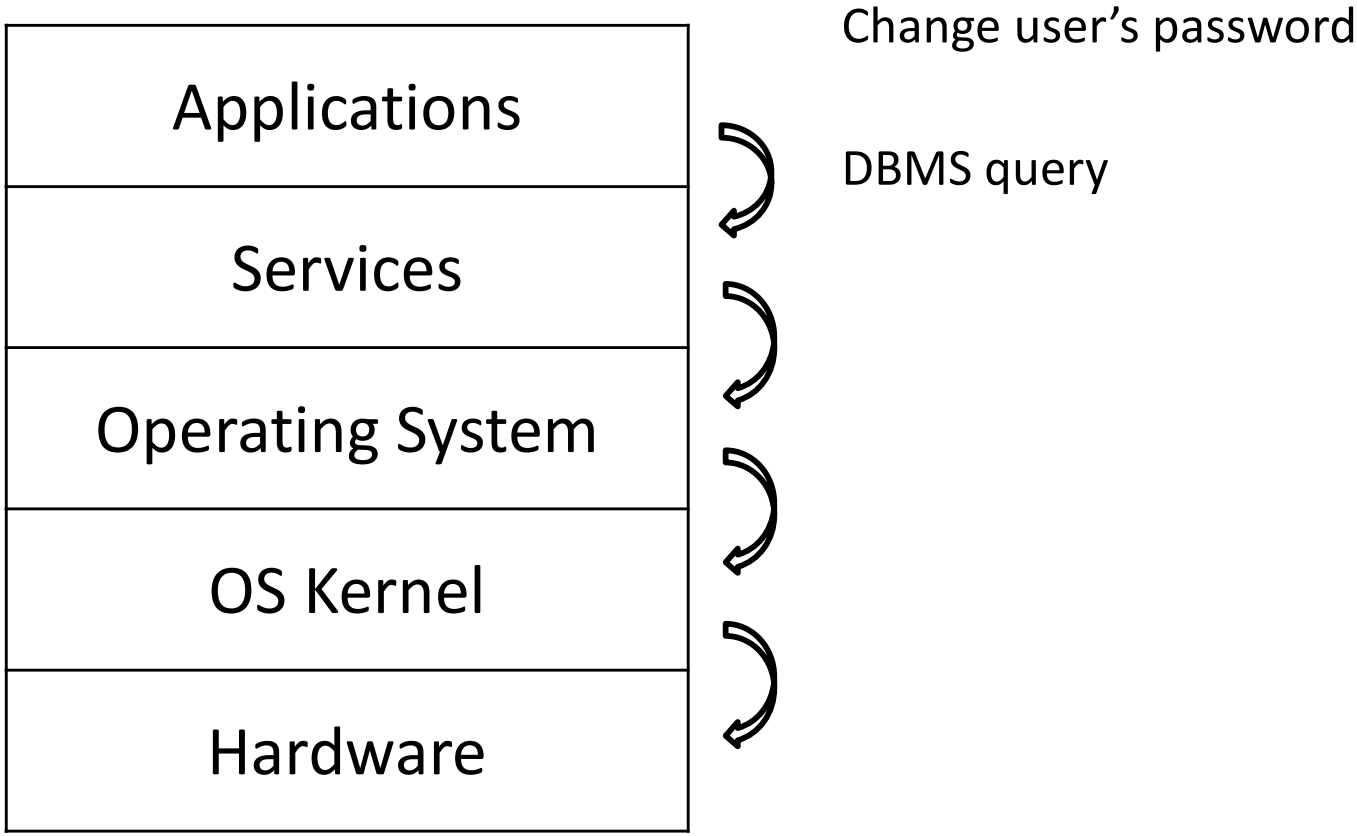
Attacking the layer below

- Consider an application using a database:



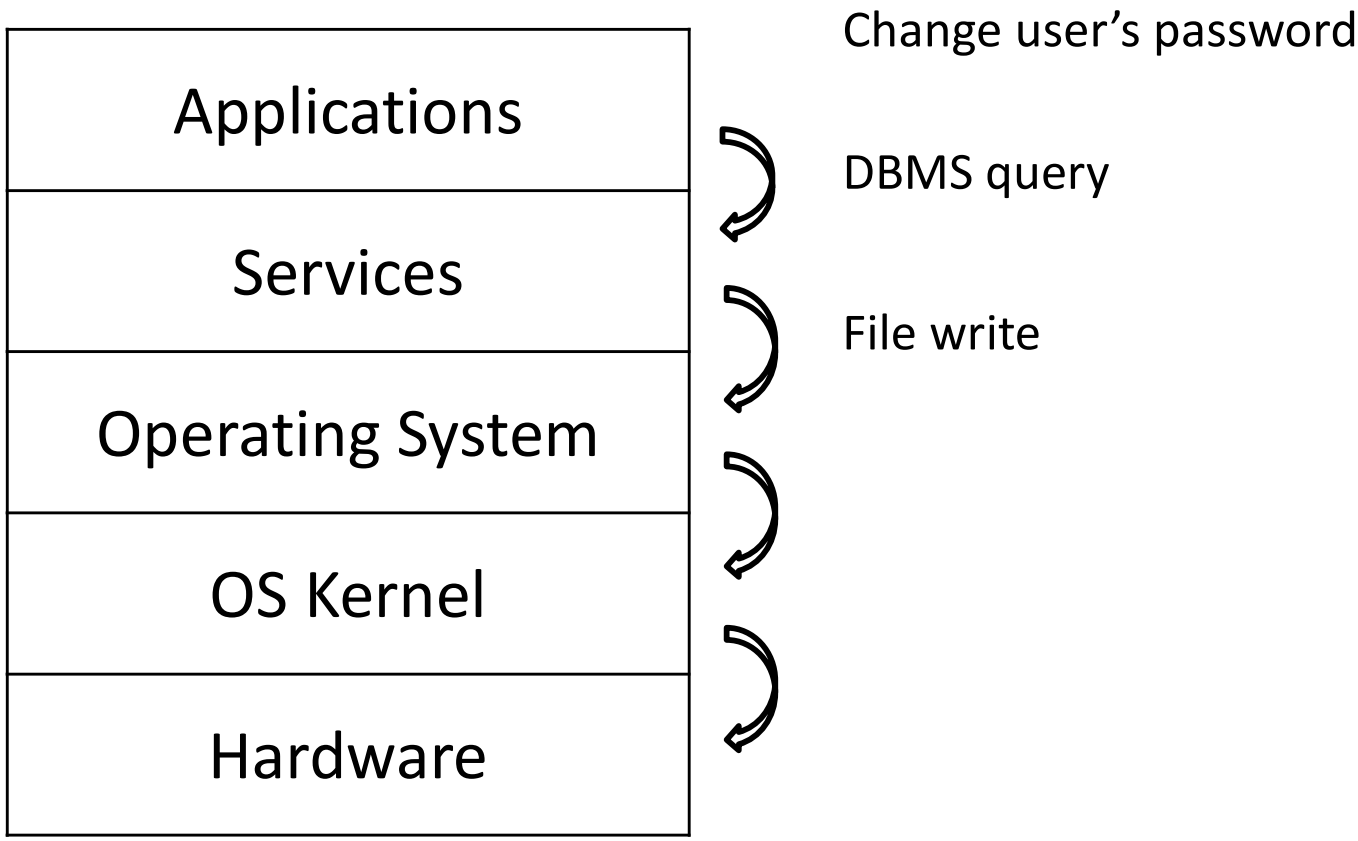
Attacking the layer below

- Consider an application using a database:



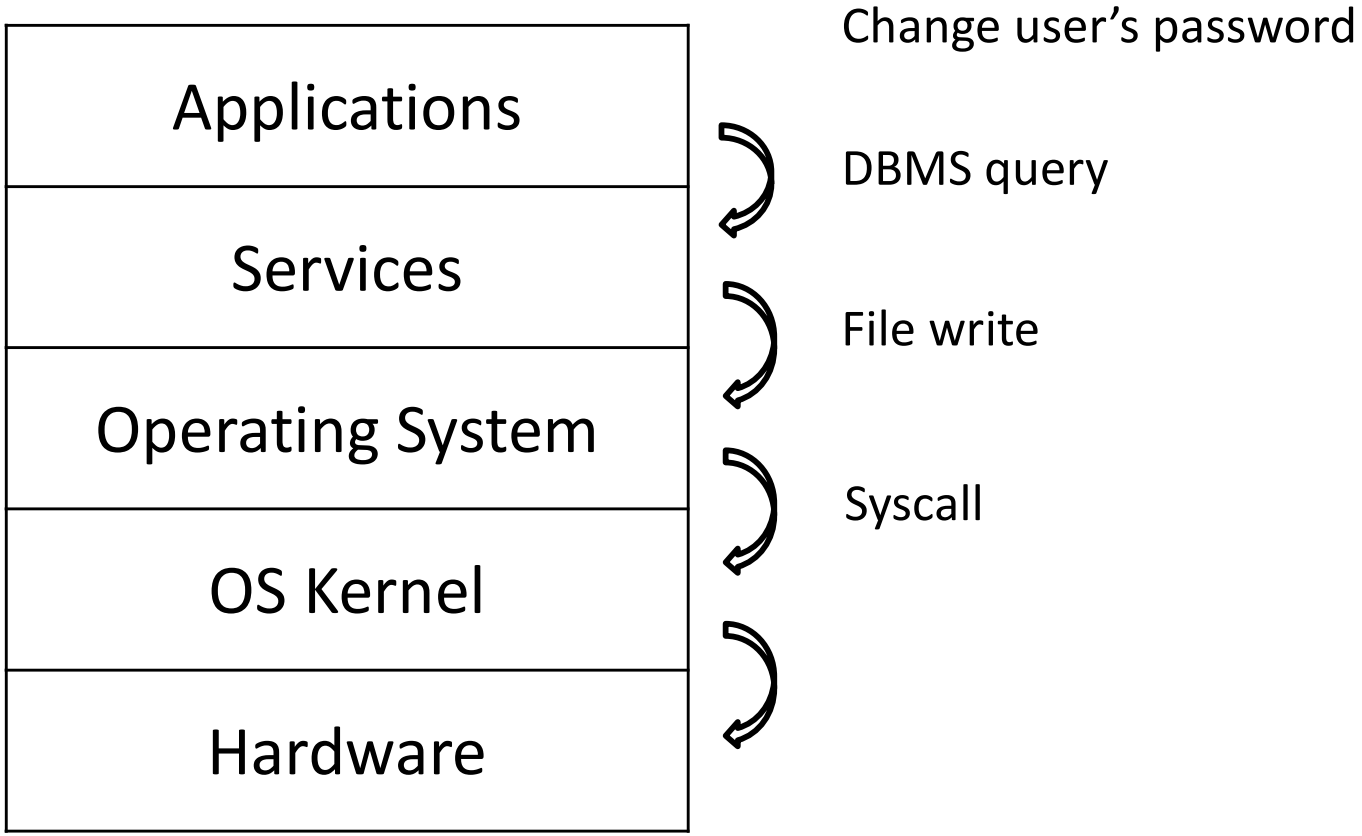
Attacking the layer below

- Consider an application using a database:



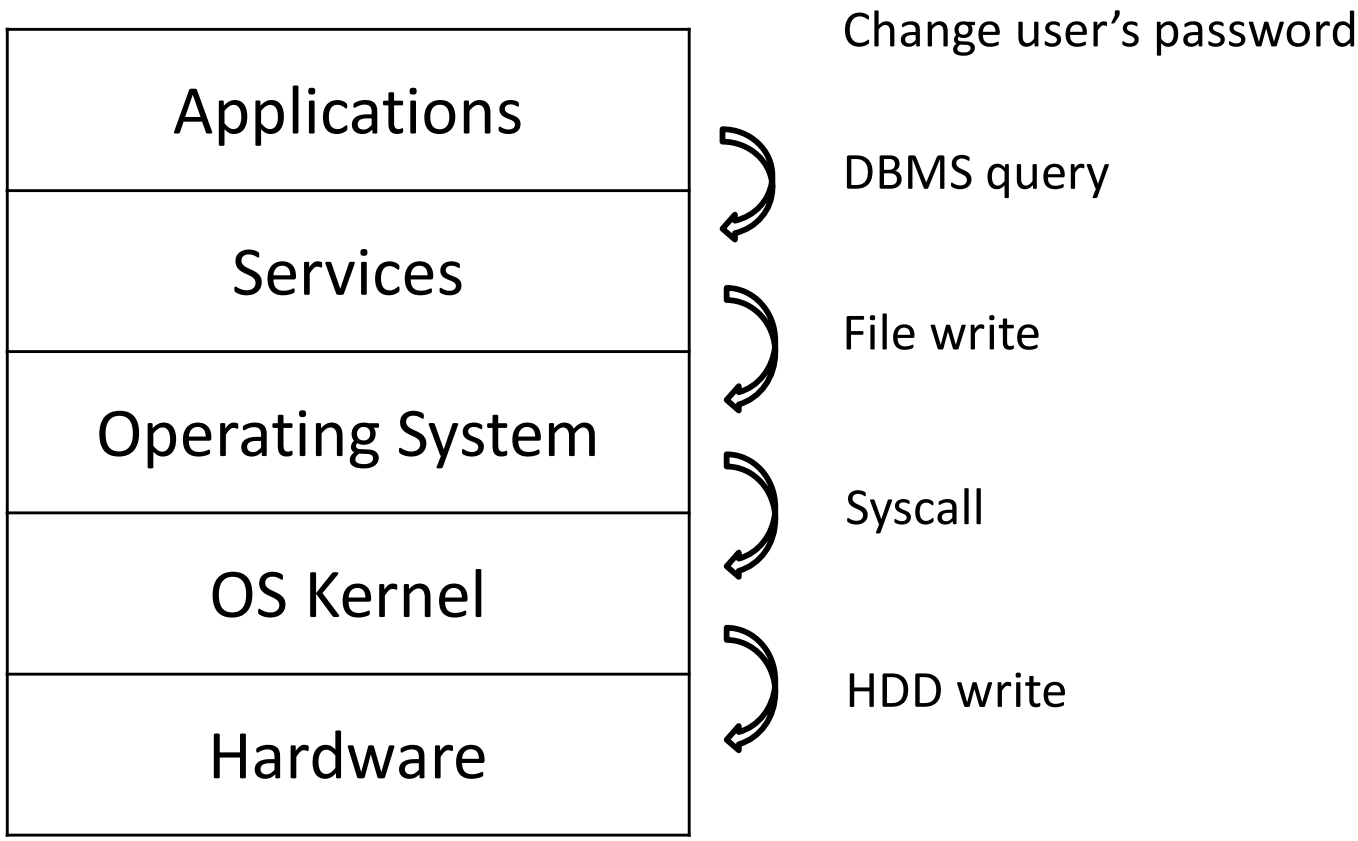
Attacking the layer below

- Consider an application using a database:



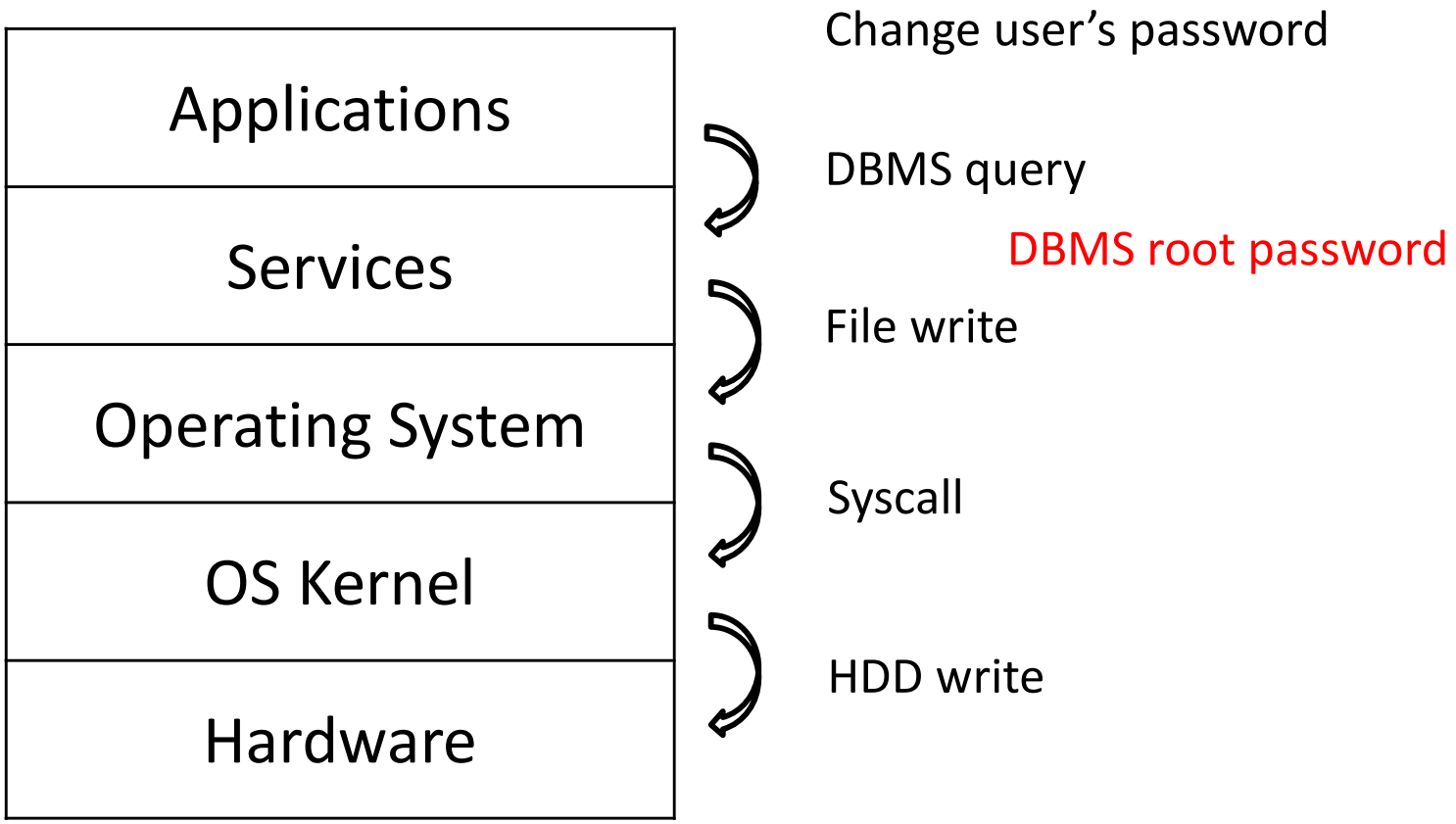
Attacking the layer below

- Consider an application using a database:



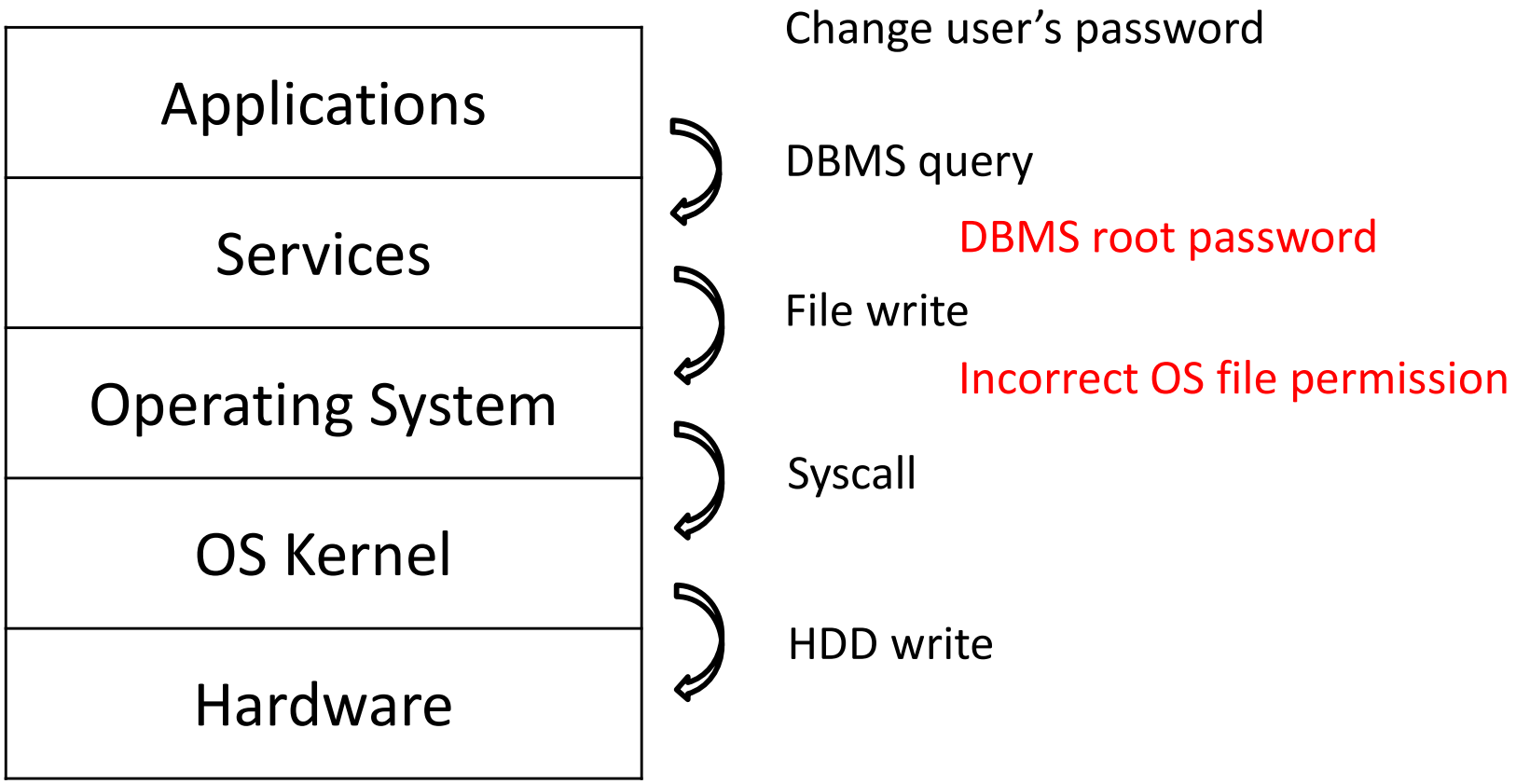
Attacking the layer below

- Consider an application using a database:



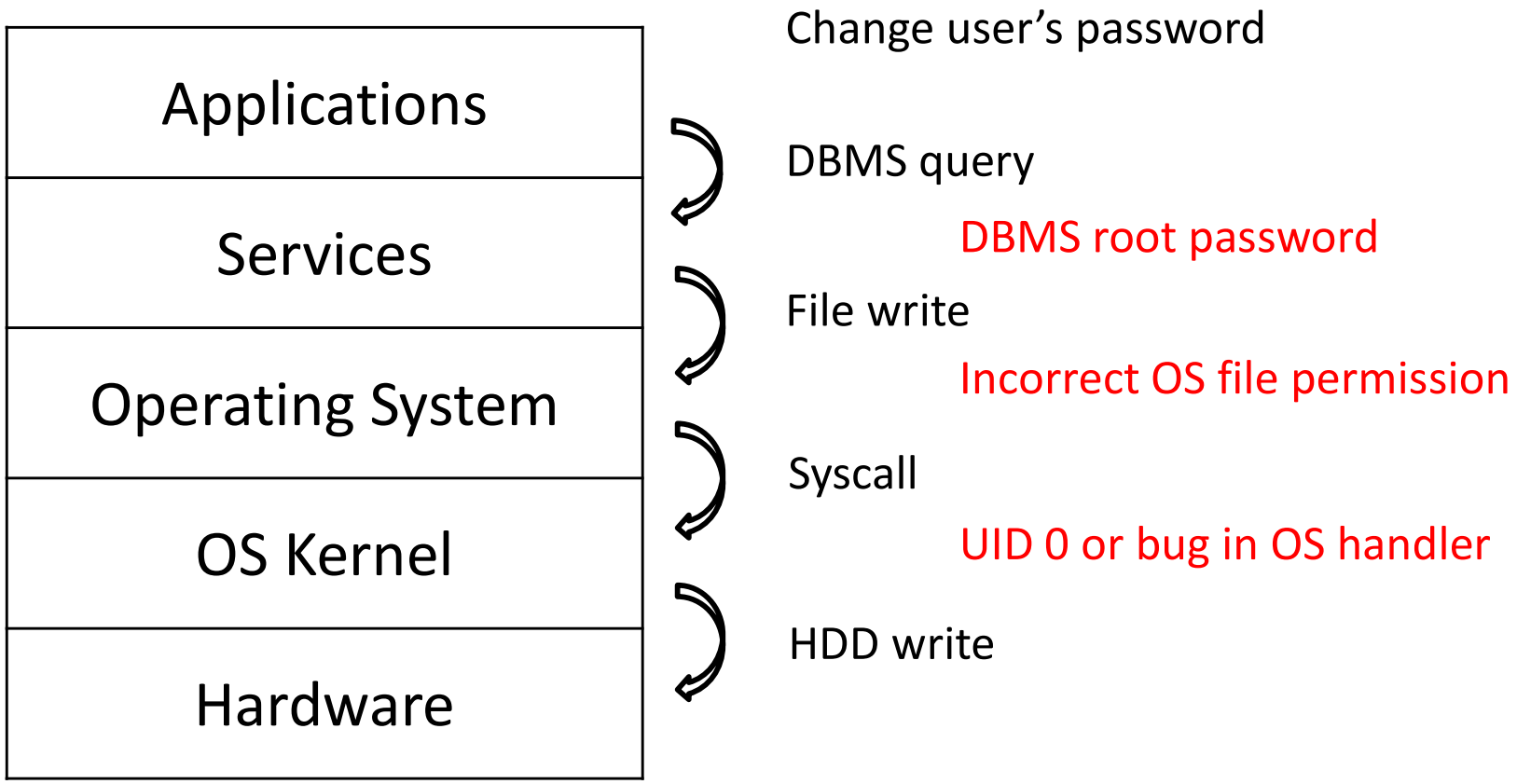
Attacking the layer below

- Consider an application using a database:



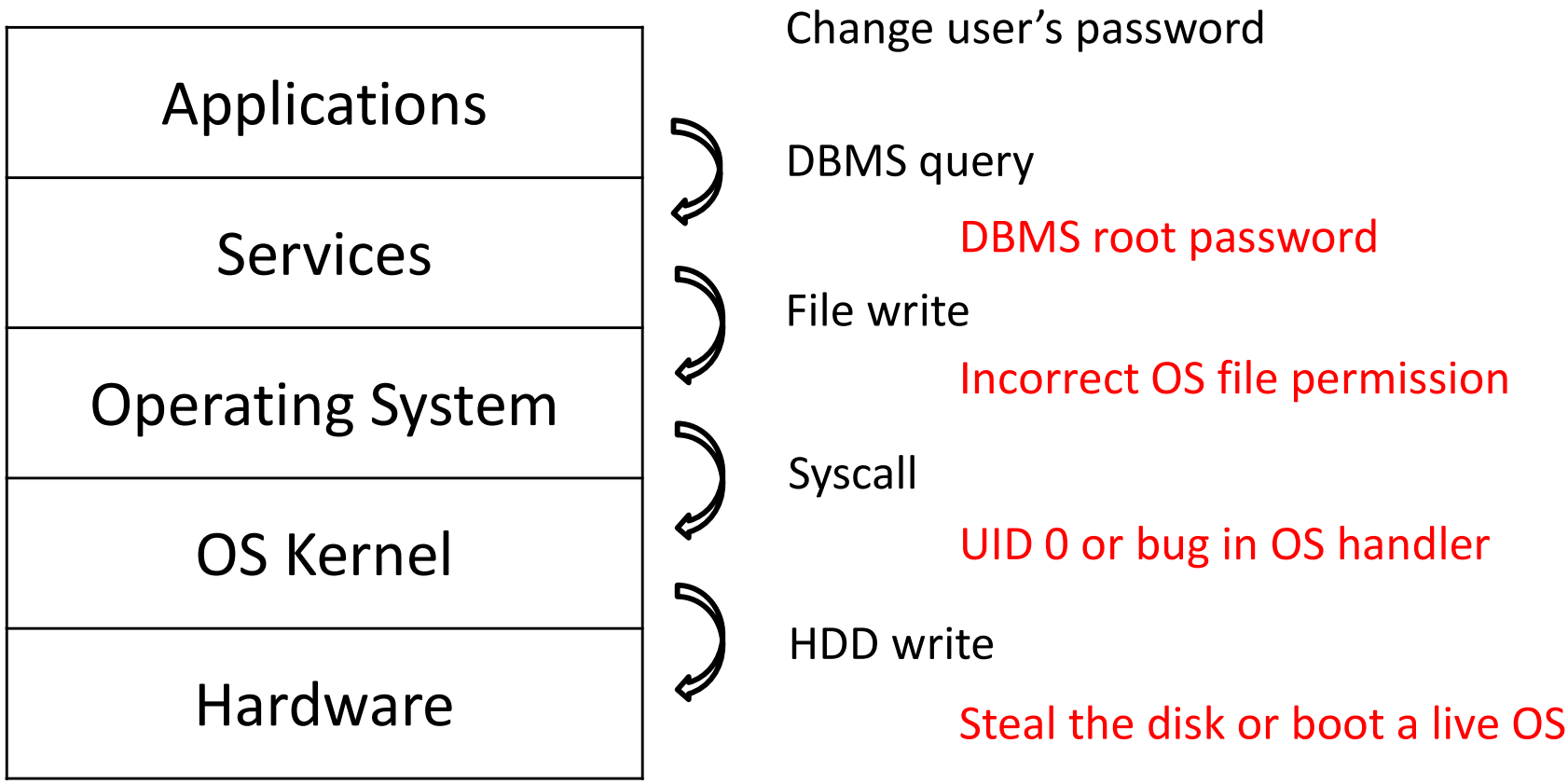
Attacking the layer below

- Consider an application using a database:



Attacking the layer below

- Consider an application using a database:



The Layer Below - Examples

- **Recovery tools** restore data by reading memory directly and then restoring the file structure. Such a tool can be used to circumvent logical access control as it does not care for the logical memory structure.
- Unix treats I/O devices and physical memory devices like files; with badly defined access permissions, e.g. if read access is given to a disk, an attacker can read the disk contents and reconstruct read protected files.
- **Buffer overruns**: a value assigned to a variable is too large for the memory buffer allocated to that variable; memory allocated to other variables is overwritten.

Summary

- Defining Security
- CIA
- Fundamental Dilemma
- Designing Secure Systems
- Principles of Computer security Design
- The Layer Below
- References: Anderson – Section 1.7, Gollmann – Chapter 3