Assign the least privilege possible

Why?	Broad privileges allow malicious or accidental access to protected resources
Principle	Limit privileges to the minimum for the context
Tradeoff	Less convenient; less efficient; more complexity
Example	Run server processes as their own users with exactly the set of privileges they require

Implement defence in depth

Why?	Systems do get attacked, breaches do happen, mistakes are made - need to minimise impact
Principle	Don't rely on single point of security, secure every level, stop failures at one level propagating
Tradeoff	Redundancy of policy; complex permissioning and troubleshooting; can make recovery difficult
Example	Access control in UI, services, database, OS

UI = User Interface

OS = Operating System

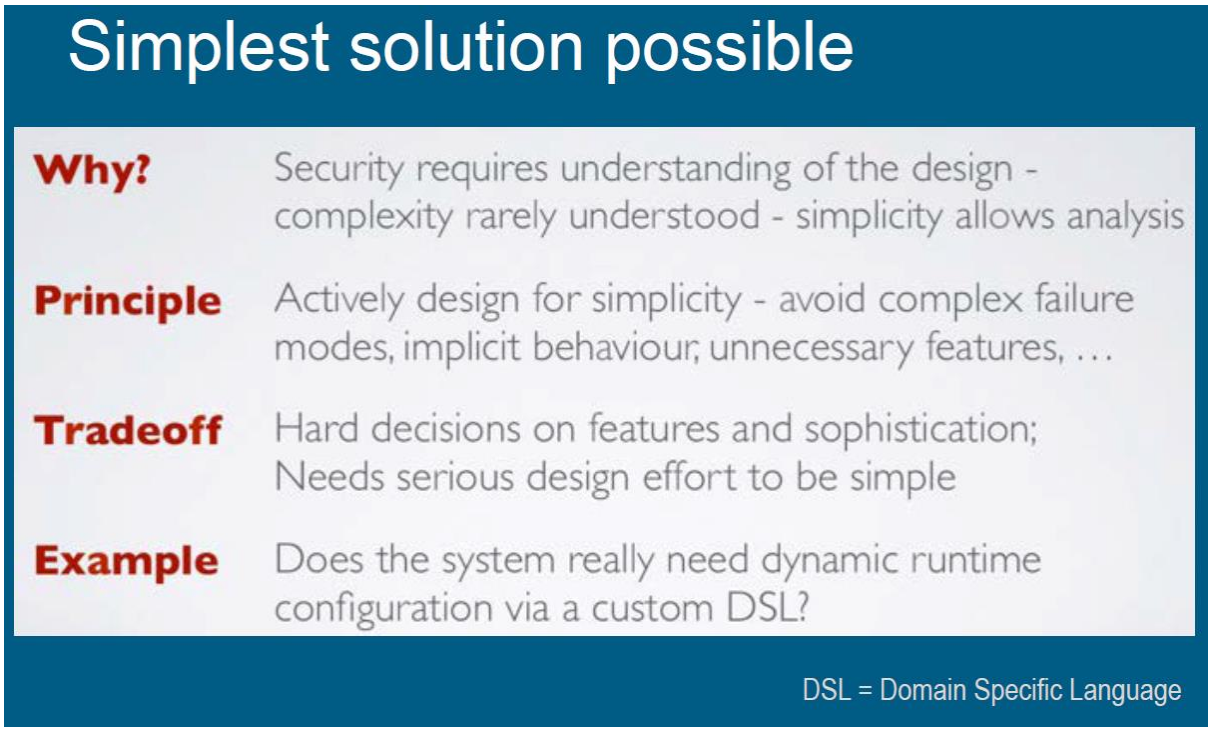
Fail securely & use secure defaults

Why?	Default passwords, ports & rules are "open doors" Failure and restart states often default to "insecure"
Principle	Force changes to security sensitive parameters Think through failures - to be secure but recoverable
Tradeoff	Convenience
Example	Don't allow "`SYSTEM/MANAGER`" after installation On failure don't disable or reset security controls

Separate responsibilities

Why?	Achieve control and accountability, limit the impact of successful attacks, make attacks less attractive
Principle	Separate and compartmentalise responsibilities and privileges
Tradeoff	Development and testing costs; operational complexity: troubleshooting more difficult
Example	"Payments" module administrators have no access to or control over "Orders" module features

Economy of Mechanism (Occam's razor)



Open Design:

The open design security principle states that the implementation details of the design should be independent of the design itself, allowing the design to remain open while the implementation can be kept secret. This is in contrast to security by obscurity where the security of the software is dependent upon the obscuring of the design itself.

When software is architected using the open design concept, the review of the design itself will not result in the compromise of the safeguards in the software.

Least Common Mechanism:

The security principle of least common mechanisms disallows the sharing of mechanisms that are common to more than one user or process if the users or processes are at different levels of privilege. This is important when defending against privilege escalation.

Psychological acceptability

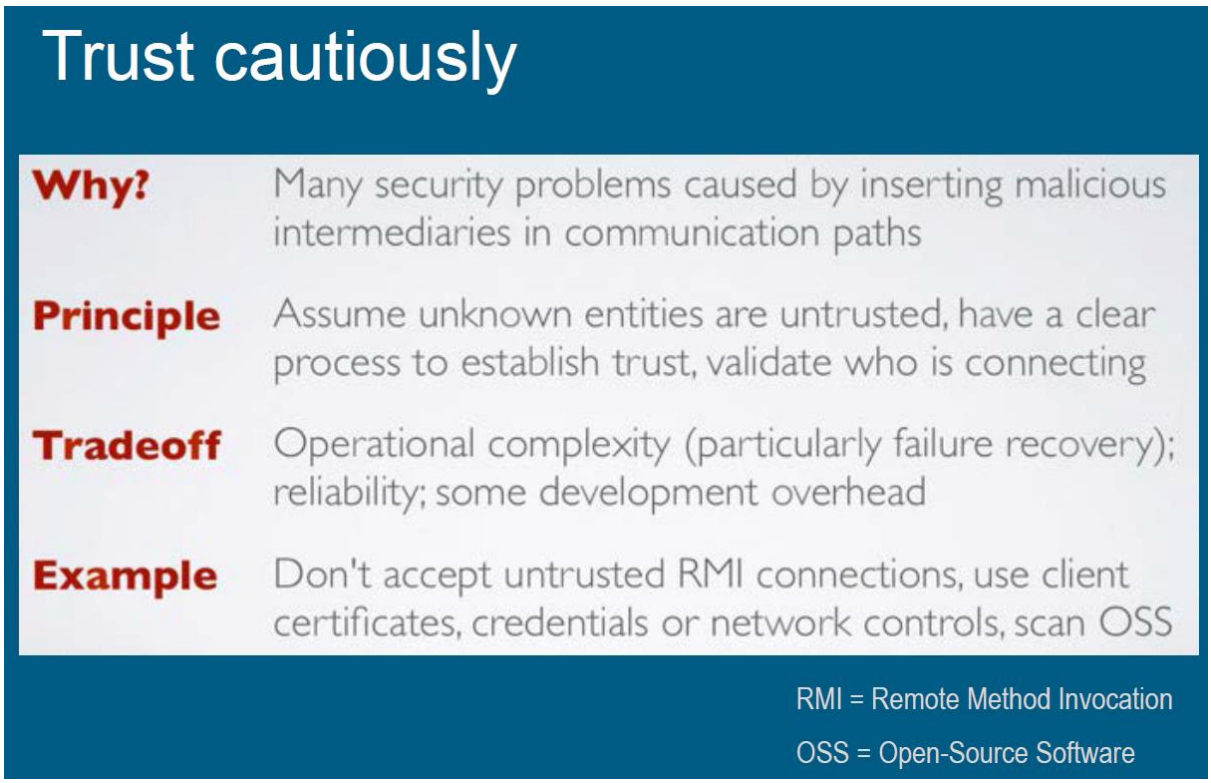
A security principle that aims at maximizing the usage and adoption of the security functionality in the software by ensuring that the security functionality is easy to use and at the same time transparent to the user. Ease of use and transparency are essential requirements for this security principle to be effective.

Security controls should not make the resource significantly more difficult to access than if the security control were not present. If a security control provides too much friction for the users then they may look for ways to defeat the control and “prop the doors open”.

Secure by Default

Secure by default means that the default configuration settings are the most secure settings possible. This is not necessarily the most user-friendly settings. Evaluate what the settings should be, based on both risk analysis and usability tests. As a result, the precise meaning is up to you to decide. Nevertheless, configure the system to only provide the least functionality and to specifically prohibit and/or restrict the use of all other functions, ports, protocols, and/or services. Also configure the defaults to be as restrictive as possible, according to best practices, without compromising the “Psychological acceptability” and “Usability and Manageability” of the system.

‘Older/other’ princlipes.....



Audit sensitive events

Why?	Provide record of activity, deter wrong doing, provide a log to reconstruct the past, provide a monitoring point
Principle	Record all security significant events in a tamper-resistant store
Tradeoff	Performance; operational complexity; dev cost
Example	Record changes to "core" business entities in an append-only store with (user, ip, timestamp, entity, event)

Never rely upon obscurity

Why?	Hiding things is difficult - someone is going to find them, accidentally if not on purpose
Principle	Assume attacker with perfect knowledge, this forces secure system design
Tradeoff	Designing a truly secure system takes time and effort
Example	Assume an attacker will guess a "port knock" network request sequence or a password obfuscation technique

Never invent security technology

Why?	Security technology is difficult to create - avoiding vulnerabilities is difficult
Principle	Don't create your own security technology - always use a proven component
Tradeoff	Time to assess security technology; effort to learn it; complexity
Example	Don't invent your own SSO mechanism, secret storage or crypto libraries … choose proven components

SSO = Single Sign-On

Find the weakest link

Why?	"Paper Wall" problem - common when focus is on technologies not threats
Principle	Find the weakest link in the security chain and strengthen it - repeat! (Threat modelling)
Tradeoff	Significant effort required; often reveals problems at the least convenient moment!
Example	Data privacy threat => encrypted communication but with unencrypted database storage and backups