Rules of Prudent Security Engineering

Jan Jürjens

Software & Systems Engineering, Informatics, TU Munich

juerjens@in.tum.de

http://www.jurjens.de/jan

Problems in Computer Security

Example (1997):
NSA hacker team breaks into U.S. Department of Defense computers and the U.S. electric power grid system, simulating power outages and 911 emergency telephone overloads in Washington, D.C..

Problems in Security Design

Attacks against system design limitations and implementation errors rather than direct attacks against security mechanisms (Anderson 1994).

Developers of security-critical systems don't always know much about security.

(And security is difficult anyway.)

Solution

Saltzer, Schroeder (1975):
design principles for security-critical systems

This talk introduces these principles.

(And mentions method that may help enforcing them.)

Security Design

Saltzer, Schroeder (1975):

- "Expansive view of problem most appropriate to ensure no gaps appear in strategy".

- "No complete method applicable to construction of large [security-critical] general-purpose systems exists yet".

Proposed method

Use formal core of Unified Modeling Language (UML)
to encapsulate knowledge on prudent security engineering.

Express security-relevant information in widely used notation.

Allows to evaluate specifications formally for security requirements.

Economy of mechanism

Keep the design as simple and small as possible.

Design and implementation errors resulting in unwanted access paths are not noticed during normal use or normal testing.

Small and simple systems can be inspected or verified.

(Is of course blissfully ignored by popular operating systems.)
(Would want trusted computing base (TCB) really.)

Economy of mechanism

Often systems made complicated to make them (look) secure.

Method for reassurance may reduce this temptation.

Payoffs from formal evaluation may increase incentive for following the rule.

Fail-safe defaults

Base access decisions on permission rather than exclusion.

Default situation should be lack of access; protection scheme identifies conditions under which access permitted.
(Conversely, could forget to enforce refusal situations.)

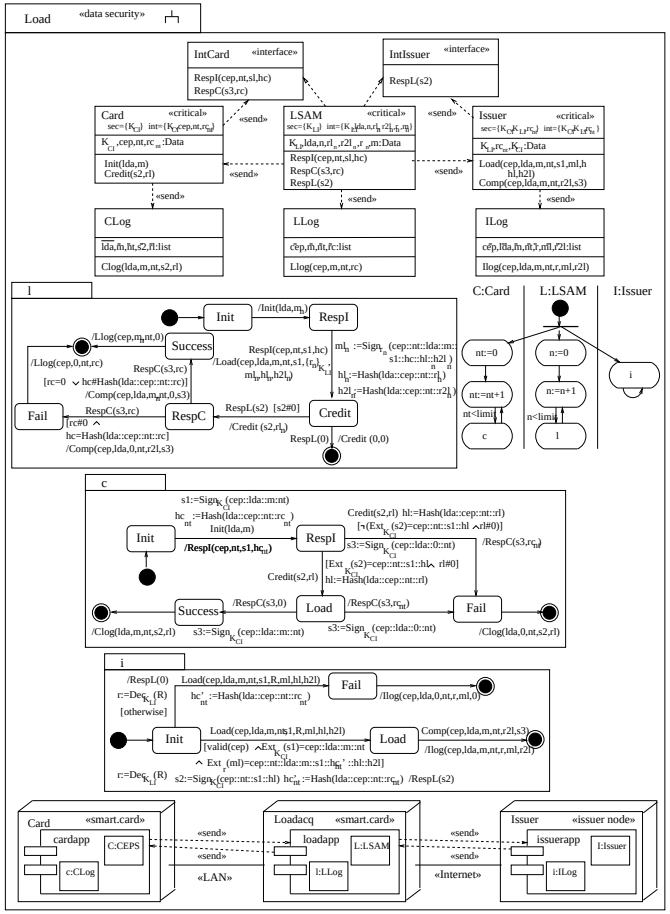
(Note this assumes that protection is more important than availability.)

(Often systems shipped with disabled security mechanisms. . .)

Fail-safe defaults

Security-relevant invariants ensured throughout execution.

Example: secure log-keeping for audit control in Common Electronic Purse Specifications (CEPS).



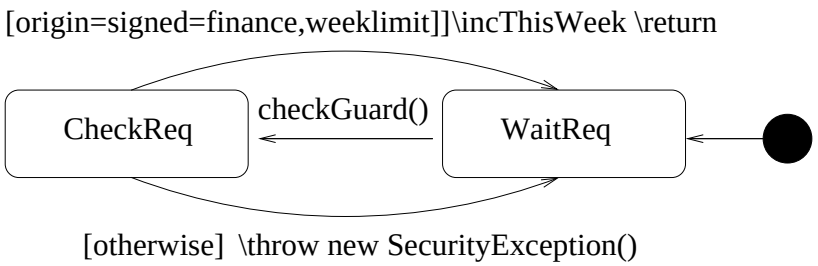
Complete mediation

Every access to every object must be checked for authority.

(Note this requires secure authentication and may be bad for performance.)
(Also: need to consider all layers !)

Complete mediation

Can enforce principle e.g. in Java using guarded objects.
Ensure proper use of guards (prevent forgotten access checks).



More feasibly, mediation wrt. a set of sensitive objects.

Open design

The design should not be secret.

Mechanisms should not depend on ignorance of potential attackers, but on possession of more easily protected keys or passwords.
Can review design for flaws without compromising security.
Not realistic to keep design of system with wide distribution secret.

(Note not universally agreed.)

Open design

Method of reassurance may help to develop systems whose security does not rely on the secrecy of its design.

Separation of privilege

A protection mechanism that requires two keys to unlock it is more robust and flexible than one that allows access to the presenter of only a single key (can be given to two different entities that have to collude).

Examples: banks, nuclear arms control.

(Note works only if users understand the point.)

Separation of privilege

Here specification satisfies separation of privilege wrt. privilege p if signature of two or more principals required to be granted p.

Formulate such requirements abstractly using activity diagrams. Verify behavioural specifications wrt. these requirements.

Least privilege

Every program and every user of the system should operate using the least set of privileges necessary to complete the job.

Minimizes possibility of misuse.

(Often ignored in UNIX ("root bloat").)

Least privilege

Least privilege: every proper diminishing of privileges gives system not satisfying functionality requirements.

Can formalise and verify this.

Least common mechanism

Minimize the amount of mechanism common to more than one user and depended on by all users.

Every shared mechanism represents a potential information path between users (prevent hidden flow of information).

Least common mechanism

Object-orientation:

- data encapsulation
- data sharing well-defined (keep at necessary minimum).

Psychological acceptability

Human interface must be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly.

(Needs to be simple, so mistakes will be minimized.)
(Also: "never underestimated ingenuity of engineers at bypassing obstacles that prevent them from getting their job done.")

Psychological acceptability

Wrt. development process:
ease of use in development of secure systems.

User side: e.g. performance evaluation
(acceptability of performance impact of security).

Two further principles: Work factor

Compare the cost of circumventing the mechanism with the resources of a potential attacker.

(Problem: work factor not always easy to calculate - there may be a shortcut.)

Compromise recording

Use mechanisms that reliably record that a compromise of information has occurred.

Example: unbreakable padlock on a flimsy file cabinet.

(These days: intrusion detection systems.)
(Problem: difficult to guarantee discovery once security is broken.)

Discussion

No absolute rules, but warnings.

Violation of rules symptom of potential trouble; review design to be sure that trouble accounted for or unimportant.

Design principles reduce number and seriousness of flaws.

Relevant material

"The Protection of Information in Computer Systems",
http://web.mit.edu/Saltzer/

Forum to discuss security patterns: www.security-patterns.de

R. Anderson 2001

Conclusion

"Expansive view of computer security."

Towards method for constructing secure systems.

Evaluate system design wrt. design guidelines.