Federal Office
for Information Security

Integration of Architectural Requirements into the CC Structure

Dr. Susanne Pingel

Federal Office for Information Security

Helmut Kurth

atsec information security corporation

9th ICCC Jeju (Korea) / September 23-25th, 2008

Agenda

- Description of the Problem & Justification of a Modification of the CC
- Description of the Approach
- Example
- CC Implementation
- Modifications of CC Sections
- Test Methodology
- Advantages of the Approach

Justification of a Modification of the CC

Description of the problem:

- The CC are intended to address **all** possible security requirements
- Security objectives are often expressed in terms of “what should not happen” (**negative requirements**) or “what should always be true” (**properties**)
- Examples:
  - The TOE shall not leak information about the keys used for encryption
  - The TOE shall provide continuous access to objects even in the case of failure of individual components

Justification of a Modification of the CC

Starting point:

General discussion on the following topics:

- Inclusion of **additional SFRs** into the CC ?
- Requirements for the **testability of SFRs** ?

→ **. . . Different perceptions . . .**

Justification of a Modification of the CC

. . . One view:

- Today's CC test methodology mainly oriented towards "functional requirements" in the narrow sense
- Focus on requirements, which can be tested by calls to external interfaces and the observation of their results
- Consequence: requirements on the overall architecture of the TOE, which cannot (or can only hardly) be mapped to specific external interfaces of the TOE, cannot be modelled by SFRs

Justification of a Modification of the CC

- Examples:
  - So-called “negative requirements” which state that certain actions or events must not be possible
  - Functional requirements which can be mapped to a specific part of the TOE design, but are not directly visible at external interfaces
- Idea: extension of CC part 2 by the specification of a new type of SFRs

Federal Office
for Information Security

Justification of a Modification of the CC

. . . Other view:

- Already the present CC allows to use the kind of requirements discussed here
- Present CC framework already allows a broader understanding of "testability" by other means
- To do:
  - No fundamental modifications of the existing CC concept necessary
  - Only necessary: amendment of existing CC components + appropriate guidance for developers and evaluators for a correct and complete implementation, description and assessment of security requirements

Justification of a Modification of the CC

Conclusion:

- Different views possible
- At present: proposed SFRs can be rejected partly with the argument that they were not testable

Necessity to improve the CC

Alternatives depending on the point of view:

- **Explicit extension of the possibilities of the CC**

OR

- **Clarifying the already existing possibilities of the CC**

Description of the Approach

- Goal: Find an **effective solution** under consideration of **migration, user friendliness and methodology aspects**
- **Main idea:** appropriate **extension of the existing SFR concept** in CC part 2
- Besides functional requirements in the narrow sense (i.e. requirements testable at external interfaces) coverage of other types of requirements
- Specification of
  - **functional requirements in the narrow sense**
  - **internal requirements**
  - **architectural requirements**

Federal Office
for Information Security

Description of the Approach

3 types of requirements:

- **"Positive" functional requirements**: directly visible and testable at the external interfaces
  Ex.: definition of rules for an access control policy, specification of externally visible behaviour of an authentication mechanism
- **Functional requirements related to TSF internals** with no directly visible effect at external interfaces
  Ex.: specification of internal functions and properties like the protection of data when transferred or stored internal within the TSF
- **"Negative" functional requirements**: expressed by stating a property to be enforced by the whole TOE
  Ex.: definition of information flow or fault tolerance

Description of the Approach

Simple relationship between the different SFR types and design aspects:

- SFRs in the narrow sense → FSP
- SFRs specifying internal behaviour of the design → ADV_TDS
- SFRs covering architectural aspects → ADV_ARC

To do:

- Coverage of all clarifications needed for the extended scope of SFRs within CC part 2
- Coverage of the different refinement paths in CC part 3 class ADV and ATE

Description of the Approach

Analysis shows:

"architectural parts" of an SFR usually implemented

- in part by some supporting (testable) functionality within the TSF and
- in part (potentially) by some design principles

Idea in our approach:

Supporting functionality

- **completely traceable down to the TOE design documentation**
- **testable appropriately using the requirements of ATE_DPT**

Example: Protection against Key Leakage

Security objective:

The TOE shall restrict the use of cryptographic keys to authorized users and not leak information about the content of the keys to any user

Implemented by:

- ❐ Testable functions
- ❐ Architecture
- ❐ Properties

Example: Protection against Key Leakage

- **Testable functions:**
  - Access control to keys
- **Architecture:**
  - Keys stored in specific storage areas that the HW protects from access by large parts of the TSF
  - Keys only accessed by defined parts of the TSF
  - Side effects of access to keys are taken care of such that they do not expose information about the key content at any interface: power consumption, caching, resource exhaustion
- **Properties:**
  - Functions using the keys are implemented in a way that the time they take is not dependent on the key value

Example: Protection against Key Leakage

→ Implementation uses a **combination of protection measures**

- ❐ Access control to keys (→ “positive” functional requirement)
- ❐ Dedicated key storage only accessible to defined parts of the TSF (→ functional requirement related to TSF internals)
- ❐ Implementation done such that the effects of the following don’t leak information about the key:
  Power consumption and time used for encryption / decryption (→ architectural property)
- ❐ Keys are excluded from being cached (→ architectural property)

Example: Protection against Key Leakage

Evaluation:

- **Access control function**
  - Interfaces defined in the functional specification, analyzed using the work units of ADV_FSP
  - Design defined in TOE design documents, analyzed using the work units of ADV_TDS
  - Implementation can be traced down to the code making the access decision, analyzed using the work units of ADV_IMP
  - Functional testing is easy, done using the work units of ATE_FUN, ATE_DPT and ATE_IND
  - Checking for other problems like buffer overflow, incorrect or insufficient parameter validation, race conditions etc. in AVA_VAN

Example: Protection against Key Leakage

- **Power analysis**
  - Based on the developer’s model of power consumption (→ should be described as part of the architecture)
  - Developer implements algorithms such that power consumption leakage is eliminated or minimized (→ should be described in architecture and design)
  - Validation usually tool based combined with large set of measurements of power consumption (→ need to analyze the effectiveness of the tools and process and interpret the data collected)
  - No “functional” test in the classical sense, but measurement based validation of the developer’s architectural model of power

Example: Protection against Key Leakage

- **Timing analysis**
  - Based on instruction timing (→ usually defined in the HW manuals)
  - Analyzes possible code paths and calculates timing (→ usually done tool based)
  - Take side effects like memory caching, instruction prefetch queues and branch prediction into account (→ requires architectural details and usually some manual analysis)
  - Larger measurement samples used to validate the model (→ tool based analysis of the samples)

Example: Protection against Key Leakage

- **Key caching**
  - Requires cache model described in the architecture and design (type of cache used, caching strategies, how to exclude elements from being cached)
  - Requires flow analysis for key material (show that no flow to cached objects exists)
  - Reduces flow problem to (hopefully) small amount of code
  - Usually validated using tools (manual source code analysis only for very small code pieces)

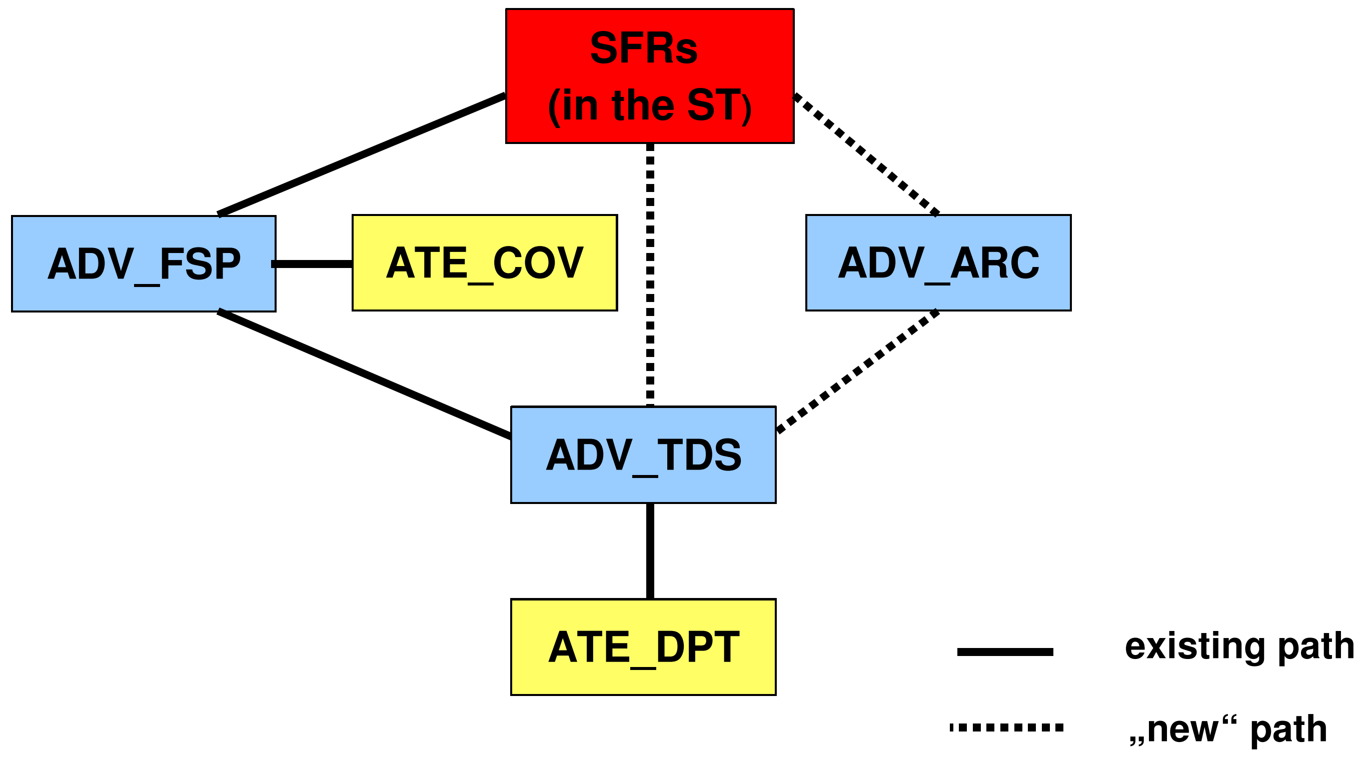
Example: Protection against Key Leakage

Summary:

- “Negative requirements” or “properties” usually can be broken down into testable functions and other (hopefully simpler) architectural properties
- Broken down until the remaining architectural property becomes “simple enough” to be verified
- Verification methods are usually dependent on the property and quite often use specific tools
  - Developer should use those tools as part of the development process
  - Evaluator needs to validate that the tools used are appropriate for the problem

CC Implementation

Refinement paths for tracing the SFR related aspects to design & testing



Federal Office
for Information Security

CC Implementation

I. Mapping of the SFRs in the ST/PP to the functional specification, TOE design and security architecture:

- Mapping from the level of SFRs in the ST/PP to the level of FSP and from the FSP to TDS: identical to the refinement mappings as in the CC today
- New mappings from the ST/PP to ADV_TDS respective ADV_ARC:
  - Consideration of all SFRs which can not (completely) be mapped to external interfaces and therefore to the FSP
  - Direct mapping of these SFRs to ADV_TDS or description of their realisation in the document Security Architecture Description (as required by ADV_ARC)

CC Implementation

- In principle four possible cases for each SFR:
  - Complete mapping to FSP
  - Complete mapping to ADV_TDS
  - Complete mapping to ADV_ARC or
  - Realisation in part by more than one of them
- Rules for mapping:
  - All effects of an SFR visible at the external interfaces of the TSF → mapping to the TSFI
  - All parts of an SFR that directly relate to testable functionality that is not directly visible at the external interfaces → mapping to the TOE design

CC Implementation

- Rules for mapping (ctd.):
  - All parts of an SFR related to properties or "negative" requirements → mapping to the security architecture, which explains how the property or negative requirement is split up into testable functions and refined properties or negative requirements
- Additional mapping between ADV_ARC and ADV_TDS: those aspects of properties (requirements of type 3) realised by mechanisms in the TOE design need to be traceable from ADV_ARC to the TDS

CC Implementation

II. Connection between ATE_COV/FSP and ATE_DPT/TDS:

- As already defined plus new work item in ATE_DPT: justification that not only the functions described traditionally in TDS but also the architectural properties described in ADV_ARC are tested adequately
- No additional work as all the cases, which are described as exceptions in today's CC, will become a normal case !
  - Refer to the already defined connection between ATE_DPT and ADV_ARC in CC V3.1
  - CEM, chap. 14.2.2: “Testing vs. alternate Approaches to verify the expected behaviour of functionality” (e.g. module testing, source code review, ...)

Modification of CC Sections

- **CC part 1:**
  - No relevant modifications necessary
  - Only section “Terms and Definitions” may need small modifications
- **CC part 2:**
  - Adaptation of the paradigm necessary
  - Addition of an explicit state that SFRs do not only cover security functionality related to the TSFI but can also cover requirements for the TOE`s overall architecture
  - Adaptation of testability requirements: functional tests in the narrow sense (triggering of external interfaces), methods like simulation or source code review, ...

Modification of CC Sections

- **CC part 2 (ctd.):**
  - Additional guidance for developers and evaluators for each SFR defined in CC part 2 how to address a potential "architectural aspect" of the SFR
    → Proposal: usage of existing appendices in CC part 2; basically extension of CC part 2 with additional guidance rather than restructuring the SFRs in CC part 2
  - Possibly addition of new security functional requirements to CC part 2 that so far have been rejected since they address "negative" requirements or properties which were known to be hard to handle in the existing CC framework

Modification of CC Sections

- **CC part 3:**
  - Some amendments and modifications necessary for ADV_FSP, ADV_ARC and ATE_DPT (see above)
  - ADV_ARC at present covers the aspect of a “developer security analysis” of the TOE's construction
    → For developers to understand the importance and the meaning of this analysis: addition of information clarifying that SFRs may not only require the implementation of specific functions at specific interfaces, but also the absence of undesired behaviour and the adherence to global security properties

Modification of CC Sections

- **CC part 3 (ctd.):**
  - Modification of the vulnerability analysis in AVA_VAN: analysis of all parts of the design and testing for violations of the Security Problem Definition
    Ex.: design decision, which doesn't formally violate the SFRs might nonetheless violate the Security Problem Definition

Federal Office
for Information Security

Test Methodology

- Needed: Suitable categories of tests to
  - cover a state-of-the-art security level
  - demonstrate the testability of SFRs
- Description of the various test methods allowed by the CC in a more systematic way and as methods of equal validity
- Proposal:
  - Description in more detail in the introduction for ATE (e.g. chapter 14.2.2 of the CEM)
  - In CC part 2 insertion of a reference to this section as a clarification what the general requirement “an SFR has to be testable” means

Test Methodology

Different test types:

- **Functional tests in the narrow sense**
  - Testing at external interfaces and testing the behaviour at these interfaces
  - Useful for those (parts of) SFRs which can be mapped to TSFI and their behaviour
- **Tests of subsystems and modules**
  - Using internal interfaces (e.g. using module-specific test software)
  - Useful for those (parts of) SFRs which are still mainly functional in the narrow sense, but not visible at external interfaces alone

Test Methodology

- **Tests using tools like debuggers, simulators, model checkers, etc.**
  - Allow e.g. inspection of memory contents at run-time
  - In parts similar to the preceding category, but many tests, which are usually seen as penetration tests, can also be in this category (e.g. use of root-kits)
- **Source code analysis**
  - Useful for verification of behaviour, which cannot be verified by one of the preceding methods
  - Tool-supported analysis: detection of potential buffer / stack overflow; check, where in the memory specific data elements are handled; check, that specific exceptions can not occur; check for illegal information flow; check for completeness of parameter checks, ...

Test Methodology

- **Penetration tests**
  - Simulation of the behaviour of an attacker, who e.g. tries to bypass specified external interfaces or tries to provoke undesired behaviour at specified interfaces

Advantages of the Approach

- **Migration**
  - No change in the construction and evaluation of STs/PPs necessary (as no changes in CC part 1, 3 and CEM for APE/ASE aspects necessary) → reuse of STs/PPs without problems
  - No introduction of separate categories for architectural and internal requirements necessary → no renaming of existing SFRs in CC part 2 → no problems with re-usage of existing STs/PPs

Advantages of the Approach

- **User friendliness**
  - Distiction between functional requirements in the narrow sense, internal and architectural requirements not important for general readers and potential authors of STs/ PPs (only relevant: definition and realisation of security objectives)
  - Categorisation of different types of SFRs: formal overhead, implementation issue
- **Methodology**
  - General question: Is a definitive distinction between different types of requirements possible in each and every case?

Contact

Federal Office for Information Security

Dr. Susanne Pingel
Godesberger Allee 185-189
53175 Bonn
Germany

Phone:	+49(0)22899-9582-5023
Telefax:	+49(0)22899-10-9582-5023

eMail: susanne.pingel@bsi.bund.de

www.bsi.bund.de
www.bsi-fuer-buerger.de

Contact

atsec information security corporation

Helmut Kurth
9130 Jollyville Road, Suite 260
Austin, TX 78759
USA

Phone:	+1-512-615-7300
Telefax:	+1-512-615-7301
eMail:	helmut@atsec.com