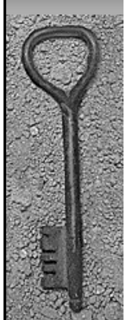
The System Security Realization

- ♦ Role of the System Security Realization
- ♦ Aspects of the System Security Realization
  - –ADV_HLD: The High Level (Security) Design
  - –ADV_LLD: The Low Level (Security) Design
  - –ADV_INT: TSF Internals
  - –ADV_IMP: The Implementation Representation
  - –ADV_RCR: Representation Correspondence
- ♦ Techniques for Developing the System Security Realization



Role of the System Security Realization

- ♦ Describe the TOE Security Functions (TSF) as Seen From the Inside
- ♦ Provide the Basis For Understanding How the TSF Actually Meet ST Security Function Requirements
- ♦ Provide the Information Needed to Analyze the Correctness of the TSF With Respect to the System Security Concept Specifications
- ♦ Provide the Basis For Developing “White Box” Tests of the Security Functions

Points to Remember

- System Security Realization Families Seem to Require Top-Down Development
  - NOT SO – they require that the realization be PRESENTED as a series of refinements
  - There are NO requirements that the TOE be developed in a strict top-down fashion
- System Security Realization Families Are Specified To Facilitate Evaluation
  - Challenge is to develop TOE so that all required information is captured

ADV_HLD – High Level Design

- Definition
- Aspects
- Components
  - Descriptive HLD
  - Security enforcing HLD
  - Semiformal HLD
  - Semiformal High-level Explanation
  - Formal HLD

Definition

- The High Level Design of a TOE provides a description of the TSF in terms of major structural units (i.e., subsystems) and relates these units to the functions that they provide
  - Provides support for argument that TOE architecture is appropriate for implementing the TOE security functional requirements

Aspects of the Family

- Define and describe “subsystems”
- Describe security functions provided by each subsystem
- Describe if and how each subsystem depends on the presences of security functions in the external IT environment
- Describe interactions between subsystems
- Externally visible subsystem interfaces

Subsystems

- Relatively small number of parts
- Structuring concept that doesn’t HAVE to be reflected in lower level decompositions
- Presentation must use something “like”subsystems
  - e.g., layers, domains, servers, etc

Subsystem Security Functionality

- Operations that each subsystem performs as part of TOE security function realization
- Relationship between subsystems and security functions may be many-to-many
  - One subsystem may wholly realize many security functions
  - Aspects of one security function may be realized through the cooperation of many subsystems
- “TSP-enforcing subsystem” directly or indirectly contibutes to TSP enforcement

External IT Environment Dependencies

- Identify the security functions or protection mechanisms that the TSF requires be present in underlying
  - hardware
  - firmware
  - software

Interactions Between Subsystems

- Identify all interfaces to each subsystem
  - Callable functions
  - Shared data structures
- Major data flows effected through the interfaces

Externally Visible Subsystem Interfaces

- Identify the subset of the interfaces to a subsystem that are directly exposed as external interfaces described in the functional specification

Descriptive HLD

- **High-level design shall**
  - **be informal.**
  - **be internally consistent.**
  - **describe the structure of the TSF in terms of subsystems.**
  - **describe the security functionality provided by each subsystem of the TSF.**
  - **identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.**
  - **identify all interfaces to the subsystems of the TSF.**
  - **identify which of the interfaces to the subsystems of the TSF are externally visible.**

Security enforcing HLD

- Same as Descriptive HLD plus
  - describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects, exceptions and error messages, as appropriate.
  - describe the separation of the TOE into TSP-enforcing and other subsystems.

Semiformal HLD

- Same as security enforcing, plus
  - The presentation of the high-level design shall be **semiforma**l.
  - The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing **complete** details of **all** effects, exceptions and error messages.

Semiformal High-level Explanation

- Same as semiformal HLD, plus
  - **The high-level design shall justify that the identified means of achieving separation, including any protection mechanisms, are sufficient to ensure a clear and effective separation of TSP-enforcing from non-TSP-enforcing functions.**
  - **The high-level design shall justify that the TSF mechanisms are sufficient to implement the security functions identified in the high-level design.**

Formal HLD

- Same as Semiformal High-level Explanation,, except that
  - The presentation of the high-level design shall be **formal**.

ADV_LLD – Low Level Design

- Definition
- Aspects
- Components
  - Descriptive Low Level Design
  - Semiformal Low Level Design
  - Formal Low Level Design

Definition

- "A … description of the internal workings of the TSF in terms of modules and their interrelationships and dependencies."
- The role of the low level design in Common Criteria is to increase confidence the TSF subsystems are correctly and effectively refined.

Aspects of the Family

- ♦ TSF Modules
- ♦ Module Interrelationships
- ♦ Module Interfaces & Behavior
- ♦ Externally Visible Interfaces
- ♦ TSP-Enforcing Functions

TSF Modules

- ♦ Unit Of Implementation, e.g.,
  - – SW: source files, packages, modules, objects
  - – HW: ASICs, VHDL specs, CPUs
- ♦ TSP-Enforcing Modules
  - – "…any module that must be relied upon for correct enforcement of the TSP."
  - – "The LLD shall describe the separation of the TOE into TSP-enforcing and other modules."

Module Interrelationships

- Security Functionality Provided
  - "…set of operations that a module performs in contribution to security functions implemented by the TOE."
  - Modules don't necessarily have to relate to specific security functions
  - Many-to-many relationship between security functions and modules
- Internal Consistency –Only Defined Interfaces Are Used For Inter-Module Interactions
- Dependencies On Other Modules
  - Security Functions from other modules that a module uses to provide its security functions

Module Interfaces & Behavior

- Purpose
  - How module fits into HLD
- Function
  - The set of operations performed by the module
- Interface(s)
  - Purpose: Operations to which access is provided
  - Method-of-use: Syntax for calling on the module operations
  - Effects: results of "standard" processing
  - Exceptions: results of "non-standard" processing
  - Error Messages: for faults and failure

Externally Visible Interfaces

- Module Interfaces That Are Exposed As Functional Specification Interfaces

TSP-Enforcing Functions

- "The low-level design shall describe how each TSP-enforcing function is provided."
  - Describe the approach being used to implement a module's security functionality
  - Experience shows that this description should also document other implementation approaches that were considered and the reasons they were not chosen

Descriptive Low Level Design

- **The low-level design shall**
  - **be presented informally.**
  - **be internally consistent.**
  - **describe the TSF in terms of modules.**
  - **describe the purpose of each module.**
  - **define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.**
  - **describe how each TSP-enforcing function is provided.**
  - **identify all interfaces to the modules of the TSF.**
  - **identify which of the interfaces to the modules of the TSF are externally visible.**
  - **describe the purpose and method of use of all interfaces to the modules of the TSF, providing details of effects, exceptions and error messages, as appropriate.**
  - **describe the separation of the TOE into TSP-enforcing and other modules.**

Semiformal Low Level Design

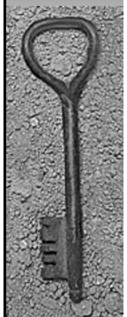
- The presentation of the low-level design shall be **<u>semiformal</u>**.
- The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing **<u>complete</u>** details of **<u>all</u>** effects, exceptions and error messages.

Formal Low Level Design

- The presentation of the low-level design shall be **formal**.

ADV_INT – TSF Internals

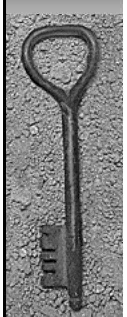
- Definition
- Aspects
- Components
  - Modularity
  - Reduction of Complexity
  - Minimization of Complexity



Definition

- (Design patterns for) the internal structure of the TSF (that) result in a TSF that is simple enough to be analyzed.

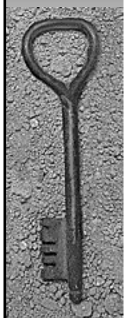
600.436 High-Assurance Systems Fall 01/System Security Realization 29



Aspects of the Family

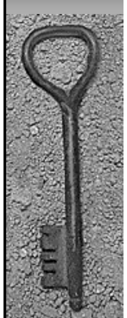
- Modularity
- Layering
- Complexity Minimization
- Functionality Minimization

600.436 High-Assurance Systems Fall 01/System Security Realization 30



Modularity

- Modular Design Reduces:
  - Interdependence between elements of TSF
  - Risk that change or error in one element will affect another element
- Modular Design Provides Basis For:
  - Determining scope of interaction between TSF elements
  - Assuring that unexpected effects don’t occur
  - Designing and evaluating test suites



Layering

- “The use of layering and of simpler designs for the TSP-enforcing functionality reduces the complexity of the TSF.”
- Resulting In:
  - Better understanding of the TSF
  - More assurance that the TOE security functional requirements are accurately and completely instantiated in the implementation.

Complexity Minimization

- Increases assurance (confidence) that code is understood
- Based on premise that if the code in the TSF is less complex, it is more likely that the design of the TSF is comprehensible.
- “Design complexity minimization is a key characteristic of a reference validation mechanism.”

600.436 High-Assurance Systems Fall 01/System Security Realization 33

Functionality Minimization

- Goal is to minimize the amount of functionality in the TSF that doesn’t enforce the TSP, thus:
  - reducing the possibility of flaws in the TSF, and
  - with modularity and layering, allowing the evaluator to focus only on that functionality which is necessary for TSP enforcement
- Also allows cleaner mapping from security policy model to implementation

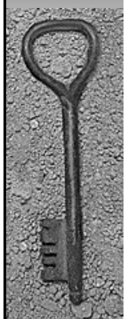
600.436 High-Assurance Systems Fall 01/System Security Realization 34

Modularity

- **The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.**
- **The developer shall provide an architectural description.**
- **The architectural description shall identify the modules of the TSF.**
- **The architectural description shall describe the purpose, interface, parameters, and effects of each module of the TSF.**
- **The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.**

Reduction of Complexity

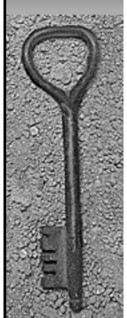
- **The developer shall design and structure the TSF in a layered fashion that minimizes mutual interactions between the layers of the design.**
- **The developer shall design and structure the TSF in such a way that minimizes the complexity of the portions of the TSF that enforce any access control and/or information flow control policies.**
- **The architectural description shall identify the modules of the TSF and shall specify which portions of the TSF enforce the access control and/or information flow control policies**.
- **The architectural description shall describe the layering architecture.**
- **The architectural description shall show that mutual interactions have been minimized, and justify those that remain.**
- **The architectural description shall describe how the portions of the TSF that enforce any access control and/or information flow control policies have been structured to minimize complexity.**



Minimization of Complexity

- The developer shall design and structure the TSF in such a way that minimizes the complexity of the **entire TSF**.
- **The developer shall design and structure the portions of the TSF that enforce any access control and/or information flow control policies such that they are simple enough to be analyzed**.
- **The developer shall ensure that functions whose objectives are not relevant for the TSF are excluded from the TSF modules**.
- The architectural description shall describe how the **entire TSF has** been structured to minimize complexity.
- **The architectural description shall justify the inclusion of any non-TSP-enforcing modules in the TSF.**

600.436 High-Assurance Systems Fall 01/System Security Realization 37



ADV_IMP – Implementation Representation

- Definition
- Aspects
- Components
  - Subset of TSF Implementation
  - TSF Implementation
  - Structured TSF Implementation

600.436 High-Assurance Systems Fall 01/System Security Realization 38

Definition

- Least abstract representation of the TSF
- The representation that is used without further refinement to create (or generate) the TSF
  - Software source code
  - Hardware drawings

Aspects of the Family

- Implementation Representation
- Implementation Structuring
- Internal Consistency

Implementation Representation

- Must be provided
  - Possibly for only a subset of the TSF
- Must "unambiguously define TSF to level of detail such that the TSF can be generated without further design decisions"

Implementation Structuring

- Describe the relationships between portions of implementation
- Language or coding style should ensure that Implementation Representation is structured into small and comprehensible sections

Internal Consistency

- All portions of the Implementation Representation
  - Conform to the rules of the representation format (programming language or drafting guidelines)
  - Match with each other
    - Function calls match function definition
    - Circuit traces have source and sink
    - Cross-checked by successful compilation or fabrication

Subset of TSF Implementation

- "Developer shall provide implementation representation for a selected subset of the TSF"
  - Negotiated between developer and evaluator
- "The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions."
- "The implementation representation shall be internally consistent"

TSF Implementation

- "…provide implementation representation for the entire TSF."
- "…**implementation representation shall describe the relationships between all portions of the implementation.**"

Structured TSF Implementation

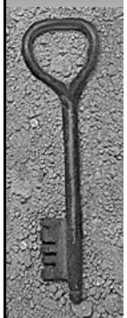
- "**implementation representation shall be structured into small and comprehensible sections.**"

ADV_RCR – Representation Correspondence

- Definition
- Aspects
- Components
  - Informal Correspondence Demonstration
  - Semiformal Correspondence Demonstration
  - Formal Correspondence Demonstration

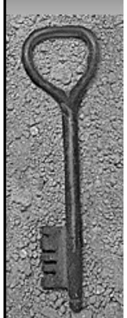
Definition

- Demonstration that, for all adjacent abstractions of representation, the less abstract representation correctly refines the more abstract representation
- Goal is to attain confidence that the least abstract representation of the TSF is a complete and correct realization of the ST security functional requirements



Aspects of the Family

- ♦ Pair-wise Correspondence
- ♦ All Relevant Security Functionality
- ♦ Exclusion of TSP or TSP Model



Pair-wise Correspondence

- ♦ Adjacent Representations
  - – TOE Summary Spec to Functional Spec
  - – Functional Spec to High Level Design
  - – High Level Design to Low Level Design
  - – Low Level Design to Implementation Representation
- ♦ Demonstration of Correspondence Should Be As Formal As Possible, Limited Only By the Formality Of The Representations

All Relevant Security Functionality

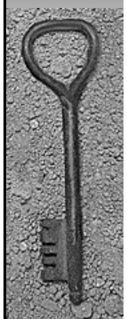
- Between TSS and FSP
  - Only TOE security functions: assurance measures not required
- Between LLD and Implementation
  - For subset implementation of TSF, required refinements limited to security functionality in the subset
- For all other cases, all parts of more abstract TSF representation must be refined into less abstract TSF representation

600.436 High-Assurance Systems Fall 01/System Security Realization 51

Exclusion of TSP or TSP Model

- Correspondence demonstrations related to TSP or TSP Model are explicitly addressed in other families or components and not repeated in ADV_RCR.

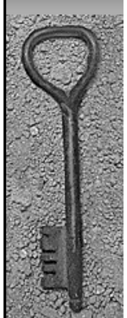
600.436 High-Assurance Systems Fall 01/System Security Realization 52



Informal Correspondence

- **The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided**.
- **For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation**.

600.436 High-Assurance Systems Fall 01/System Security Realization 53



Semiformal Correspondence

- **For each adjacent pair of provided TSF representations, where portions of both representations are at least semiformally specified, the demonstration of correspondence between those portions of the representations shall be semiformal**.

600.436 High-Assurance Systems Fall 01/System Security Realization 54

Formal Correspondence

- **For those corresponding portions of representations that are formally specified, the developer shall prove that correspondence**.
- For each adjacent pair of provided TSF representations, where portions of **one** representation are **semiformally specified and the other** at least semiformally specified, the demonstration of correspondence between those portions of the representations shall be semiformal.
- **For each adjacent pair of provided TSF representations, where portions of both representations are formally specified, the proof of correspondence between those portions of the representations shall be formal.**

Techniques

- Stepwise Refinement
- Reverse Engineering
- Interviews
- Hybrids

Stepwise Refinement

- Also Called Top-Down Development
- Goal Is To Develop System As A Series of Progressively Less Abstract Representations – With Source Code As the Least Abstract Representation
- Implicitly Assumed Approach In CC

Stepwise Refinement (2)

- Great In Theory, BUT…
  - Few people are able to establish and maintain separation among abstract representations
  - Most project managers are too impatient to tolerate a design process that doesn't result in immediate production of executable code
  - "The devil is in the details…"
    - Creating a less abstract representation almost always forces changes to the more abstract representation
    - The notation used for each abstract representation can introduce arbitrary restrictions and dependencies that aren't strictly part of the refinement process

Reverse Engineering

- ♦ Also Called Bottom-Up Development
- ♦ Develops a Least Abstract Representation From a Rough Notion Of Requirements
- ♦ More Abstract Representations Are Developed By Summarizing the Details In Less Abstract Representations

Reverse Engineering (2)

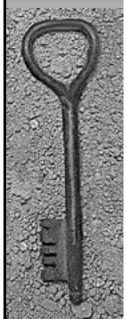
- ♦ VERY Common Practice, BUT…
  - – Labor intensive
  - – Doesn't directly result in running code and is therefore considered "busywork" by those most qualified to do it
    - More abstract representations usually not kept current with the least abstract representation
  - – Few people are able to establish and maintain separation among abstract representations
    - Worse here, since "out" is to refer to less (or usually least) abstract representation – "Read the Code!!"
  - – Natural focus is on what is built rather than what is needed

Interviews

- Data Mining On the Developers Or Maintainers
- Especially Useful When
  - HLD is least abstract representation required
  - System is horribly complex legacy code
  - Access to source code, and therefore use of reverse engineering, is hard
  - There exist one or more “gurus” who can answer all, or most, questions about the system after thinking for a short time

Interviews (2)

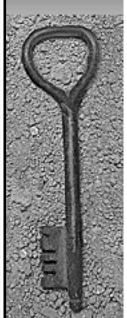
- Goal is to Get The Guru’s Internal Model of The System
- “Drink From Fire Hose” Since Gurus Usually
  - Are in short supply and high demand
  - Have a short attention span and low tolerance for those who don’t understand



Interviews (3)

- “Oracle” Mode
  - One guru seems to know everything
  - Frame interview as “training” that will reduce “stupid” questions later on
- “Inner Circle” Mode
  - Multiple gurus with interlocking and/or overlapping expertise in system realization
  - Use moderated discussion with easy whiteboard access
  - Use “same answer three times” rule

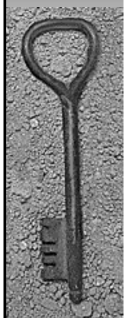
600.436 High-Assurance Systems   Fall 01/System Security Realization   63



Hybrids

- In Most Real Systems, A Combination Of Techniques Is Needed
- Plan How To Reconcile Equivalent Material Gained From Multiple Techniques
- Stepwise Refinement is Quite Effective For
  - Security Target development
  - Security Policy Model development
  - Functional Spec and High-Level Design in new systems development

600.436 High-Assurance Systems   Fall 01/System Security Realization   64



Hybrids (2)

- Reverse Engineering Is Obviously Most Useful For
  - Implementation Representation
  - TSF Internals
  - Low Level Design
  - FSP and HLD in lower assurance systems
- Interviews Are Most Effective For
  - HLD and LLD in any existing system
  - FSP, as a starting point in existing systems