PURDUE UNIVERSITY

CS52600: Information Security

Formal Verification

1 November, 2010

Prof. Chris Clifton

Formal Verification: Components

- Formal Specification defined in unambiguous (mathematical) language
  - Example: security policy models
- Implementation Language
  - Generally somewhat constrained
- Formal Semantics relating the two
- Methodology to ensure implementation ensures specifications met

CS526, Spring 2003 2

Specification Languages

- Specify WHAT, not HOW
  - Valid states of system
  - Postconditions of operations
- Non-Procedural
- Typical Examples:
  - Propositional / Predicate Logic (see Chapter 34)
  - Temporal Logic (supports before/after conditions)
  - Set-based models (e.g., formal Bell-LaPadula model of 5.2.3)

Specification Languages

- Must support machine processing
  - Strong typing
  - Model input/output/errors
- Example: SPECIAL
  - First order logic base
  - Strongly typed
  - VFUN: describes variables (state)
  - OFUN: describe state transitions

Example: SPECIAL

if (r $\notin \Delta(\rho_6)$)
then $\rho_6(r,v) = (\mathbf{i}, v)$
else if ([$o \neq root(o)$ **and** $parent(o) \neq root(o)$ **and** $parent(o) \in b(s_1: \mathbf{w})$] **or** [$parent(o) = root(o)$ **and** $canallow(s_1, o, v)$] **or** [$o = root(o)$ **and** $canallow(s_1, root(o), v)$])
then $\rho_6(r,v) = (\mathbf{y}, (b, m + m[s_2,o] \leftarrow \mathbf{r}, f, h))$
else $\rho_6(r,v) = (\mathbf{n}, v)$

```
MODULE Bell_LaPadula_Model Give_read
Types
Subject_ID:  DESIGNATOR;
Object_ID:  DESIGNATOR;
Access_Model:  {READ, APPEND, WRITE};
Access:  STRUCT_OF(Subject_ID subject;
    Object_ID object; Access_Mode mode);
Functions
VFUN active (Object_ID object) -> BOOLEAN
    active:  HIDDEN; INITIALLY TRUE;
VFUN access_matrix() -> Access accesses:
    HIDDEN;
    INITIALLY FORALL Access a: a
    INSERT accesses => active(a.object);
OFUN give_access(Subject_ID giver; Access
    access);
    ASSERTIONS active(access.object) =
    TRUE;
    EFFECTS `access_matrix() =
    access_matrix() UNION (access);
END_MODULE
```


Verification Methodologies

- Proof based vs. model based
  - Proof: Formula define premises / conclusions
    - Proof shows how to reach conclusions from premises
  - Model-based: Premises and conclusions have compatible truth tables
- Full vs. property verification
  - Does methodology model full system?
  - Or just prove certain key properties?
- Automation – may be manual or have tool support

Example: Enhanced Hierarchical Development Methodology

- Proof-based method
  - Uses Boyer-Moore Theorem Prover
- Hierarchical approach
  - *Abstract Machines* defined at each level
    - specification written in SPECIAL
  - *Mapping Specifications* define functionality in terms of machines at higher layers
  - *Consistency Checker* validates mappings "match"
- Compiler that maps a program into a theorem-prover understood form
- Successfully used on MLS systems
  - Few formal policy specifications outside MLS domain

Alternate Approach: Combine Specifications and Language

- Specifications defined on procedures
  - Entry conditions
  - Exit conditions
  - Assertions
- Proof techniques ensure exit conditions / assertions met given entry conditions
  - Also run-time checking
- Examples:
  - Gypsy (in book) – uses theorem prover
  - CLU
  - Eiffel (and derivatives) – run-time checks

Other Examples

- Prototype Verification System (PVS)
  - Based on EHDM
  - Interactive theorem-prover
- Symbolic Model Verifier
  - Temporal logic based
  - Notion of "path" – program represented as tree
  - Statements that condition must hold at *a* future state, *all* future states, all states on one path, etc.

Is this Real?

- Formal verification of protocols
  - Key management
  - Protocol development
- Verification of libraries
  - Entire system not verified
  - But components known okay
- High risk subsystems

PURDUE UNIVERSITY

CS52600: Information Security

Formal Evaluation

1 November, 2010

Prof. Chris Clifton

What is Formal Evaluation?

- Method to achieve *Trust*
  - Not a guarantee of security
- Evaluation methodology includes:
  - Security requirements
  - Assurance requirements showing how to establish security requirements met
  - Procedures to demonstrate system meets requirements
  - *Metrics for results*
- Examples: TCSEC (Orange Book), ITSEC, CC

CS526, Spring 2003 14

Formal Evaluation: Why?

- Organizations require assurance
  - Defense
  - Telephone / Utilities
  - "Mission Critical" systems
- Formal verification of entire systems not feasible
- Instead, organizations develop formal evaluation methodologies
  - Products passing evaluation are trusted
  - Required to do business with the organization

TCSEC: The Original

- Trusted Computer System Evaluation Criteria
  - U.S. Government security evaluation criteria
  - Used for evaluating commercial products
- Policy model based on Bell-LaPadula
- Enforcement: Reference Validation Mechanism
  - Every reference checked by compact, analyzable body of code
- Emphasis on Confidentiality
- Metric: Seven trust levels:
  - D, C1, C2, B1, B2, B3, A1
  - D is "tried but failed"

TCSEC Class Assurances

- C1: Discretionary Protection
  - Identification
  - Authentication
  - Discretionary access control
- C2: Controlled Access Protection
  - Object reuse and auditing
- B1: Labeled security protection
  - Mandatory access control on limited set of objects
  - Informal model of the security policy

CS526, Spring 2003 17

TCSEC Class Assurances *(continued)*

- B2: Structured Protections
  - Trusted path for login
  - Principle of Least Privilege
  - Formal model of Security Policy
  - Covert channel analysis
  - Configuration management
- B3: Security Domains
  - Full reference validation mechanism
  - Constraints on code development process
  - Documentation, testing requirements
- A1: Verified Protection
  - Formal methods for analysis, verification
  - Trusted distribution

CS526, Spring 2003 18

How is Evaluation Done?

- Government-sponsored independent evaluators
  - Application: Determine if government cares
- Preliminary Technical Review
  - Discussion of process, schedules
  - Development Process
  - Technical Content, Requirements
- Evaluation Phase

TCSEC: Evaluation Phase

- Three phases
  - Design analysis
    - Review of design based on documentation
  - Test analysis
  - Final Review
- Trained independent evaluation
  - Results presented to Technical Review Board
  - Must approve before next phase starts
- Ratings Maintenance Program
  - Determines when updates trigger new evaluation

TCSEC: Problems

- Based heavily on confidentiality
- Tied security and functionality
- Base TCSEC geared to operating systems
  - TNI: Trusted Network Interpretation
  - TDI: Trusted Database management System Interpretation

CS526, Spring 2003 21

Later Standards

- CTCPEC – Canada
- ITSEC – European Standard
  - Did not define criteria
  - Levels correspond to strength of evaluation
  - Includes code evaluation, development methodology requirements
  - Known vulnerability analysis
- CISR: Commercial outgrowth of TCSEC
- FC: Modernization of TCSEC
- FIPS 140: Cryptographic module validation
- Common Criteria: International Standard
- SSE-CMM: Evaluates developer, not product

CS526, Spring 2003 23

ITSEC: Levels

- E1: Security target defined, tested
  - Must have informal architecture description
- E2: Informal description of design
  - Configuration control, distribution control
- E3: Correspondence between code and security target
- E4: Formal model of security policy
  - Structured approach to design
  - Design level vulnerability analysis
- E5: Correspondence between design and code
  - Source code vulnerability analysis
- E6: Formal methods for architecture
  - Formal mapping of design to security policy
  - Mapping of executable to source code

CS526, Spring 2003 24

ITSEC Problems:

- No validation that security requirements made sense
  - Product meets goals
  - But does this meet user expectations?
- Inconsistency in evaluations
  - Not as formally defined as TCSEC

CS526, Spring 2003 25

What is Formal Evaluation?

- Method to achieve *Trust*
  - Not a guarantee of security
- Evaluation methodology includes:
  - Security requirements
  - Assurance requirements showing how to establish security requirements met
  - Procedures to demonstrate system meets requirements
  - *Metrics for results*
- Examples: TCSEC (Orange Book), ITSEC, CC

CS526, Spring 2003 28

- Replaced TCSEC, ITSEC
- CC Documents
  - Functional requirements
  - Assurance requirements
  - Evaluation Assurance Levels
- CC Evaluation Methodology
  - Detailed process model for each level
- National Scheme

CS526, Spring 2003 29

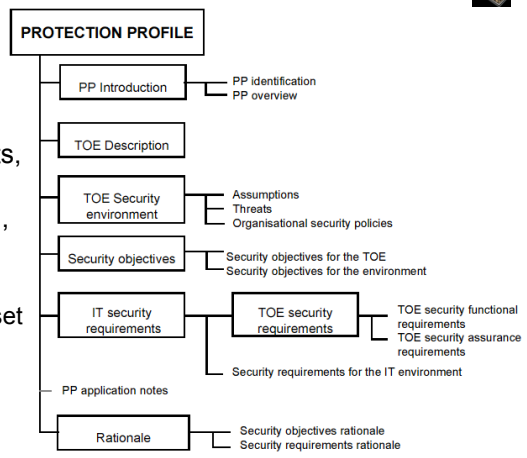
Common Criteria: Origin

CS526, Spring 2003 30

Common Criteria: Protection Profile

Domain-specific set of security requirements

- Narrative Overview
- Domain description
- Security Environment (threats, overall policies)
- Security Objectives: System, Environment
- IT Security Requirements
  - Functional drawn from CC set
  - Assurance level
- Rationale for objectives and requirements

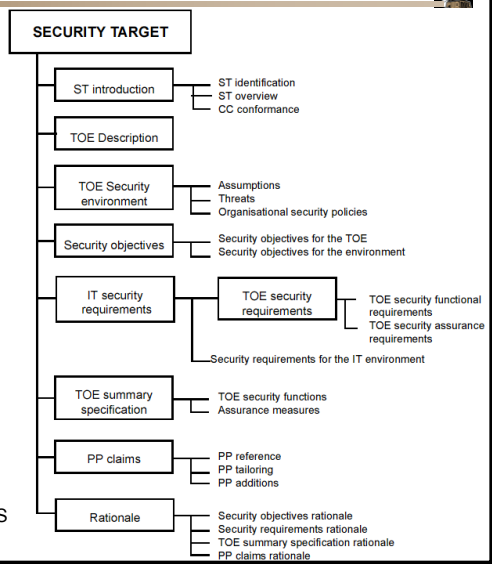


CS526, Spring 2003 31

Common Criteria: Security Target

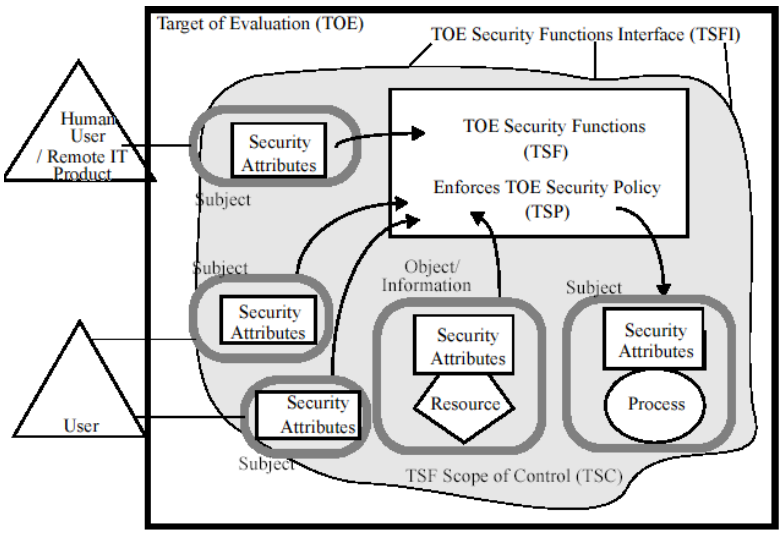
Specific requirements used to evaluate system

- Narrative introduction
- Environment
- Security Objectives
  - How met
- Security Requirements
  - Environment and system
  - Drawn from CC set
- Mapping of Function to Requirements
- Claims of Conformance to Protection Profile



CS526, S

Security Paradigm

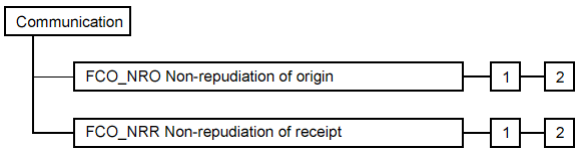


33

Common Criteria: Functional Requirements

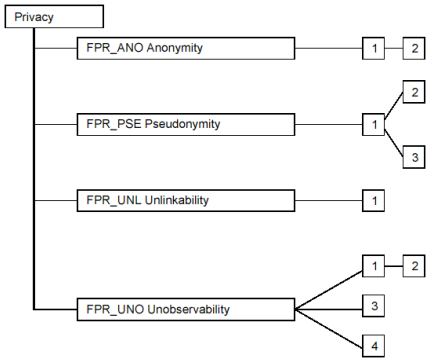
- 362 page document
- 17 Classes
  - Audit, Communication, Cryptography, User data protection, ID/authentication, Management, Privacy, Protection of Security Functions, Resource Utilization, Access, Trusted paths
- Several families per class
- Lattice of components in family

Class Example: Communication



- Non-repudiation of origin
  1. Selective Proof. Capability to request verification of origin
  2. Enforced Proof. All communication includes verifiable origin

Class Example: Privacy



1. Pseudonymity
   1. The TSF shall ensure that [assignment: *set of users and/or subjects*] are unable to determine the real user name bound to [assignment: *list of subjects and/or operations and/or objects*]
   2. The TSF shall be able to provide [assignment: *number of aliases*] aliases of the real user name to [assignment: *list of subjects*]
   3. The TSF shall [selection: *determine an alias for a user, accept the alias from the user*] and verify that it conforms to the [assignment: *alias metric*]
2. Reversible Pseudonimity
   1. …
3. Alias Pseudonimity
   1. …

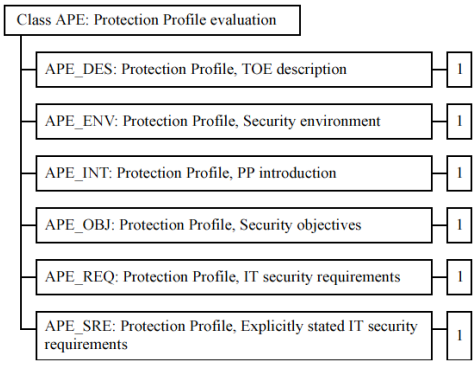
CS526, Spring 2003 36

Common Criteria: Assurance Requirements

- 216 page document
- 10 Classes
  - Protection Profile Evaluation, Security Target Evaluation
  - Configuration management, Delivery and operation, Development, Guidance, Life cycle, Tests, Vulnerability assessment
  - Maintenance
- Several families per class
- Lattice of components in family

CS526, Spring 2003 37

Example: Protection Profile Evaluation



Security environment

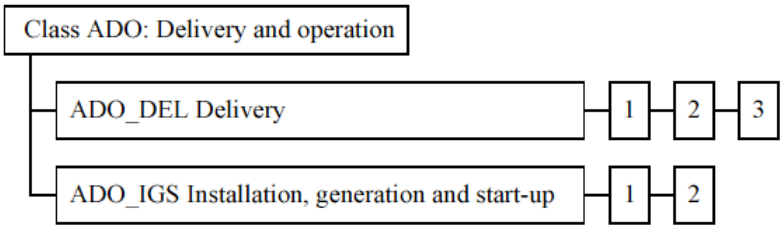
- In order to determine whether the IT security requirements in the PP are sufficient, it is important that the security problem to be solved is clearly understood by all parties to the evaluation.

1. Protection Profile, Security environment, Evaluation requirements
   - Dependencies: No dependencies.
   - Developer action elements:

- The PP developer shall provide a statement of TOE security environment as part of the PP.
  - Content and presentation of evidence elements:
- The statement of TOE security environment shall identify and explain any assumptions about the intended usage of the TOE and the environment of use of the TOE.
- The statement of TOE security environment shall identify and explain any known or presumed threats to the assets against which protection will be required, either by the TOE or by its environment.
- The statement of TOE security environment shall identify and explain any organisational security policies with which the TOE must comply.
  - Evaluator action elements:
- The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
- The evaluator shall confirm that the statement of TOE security environment is coherent and internally consistent.

CS526, Spring 2003 38

Example: Delivery and Operation



Installation, generation and start-up

A. Installation, generation, and start-up procedures
   - Dependencies: AGD_ADM.1 Administrator guidance

B. Developer action elements:
   - The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

C. Content and presentation of evidence elements:
   - The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

D. Evaluator action elements:
   - The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
   - The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

Generation Log

CS526, Spring 2003 39

Common Criteria: Evaluation Assurance Levels

1. Functionally tested
2. Structurally tested
3. Methodically tested and checked
4. Methodically designed, tested, and reviewed
5. Semiformally designed and tested
6. Semiformally verified design and tested
7. Formally verified design and tested

Common Criteria: Evaluation Process

- National Authority authorizes evaluators
  - U.S.: NIST accredits commercial organizations
  - Fee charged for evaluation
- Team of four to six evaluators
  - Develop work plan and clear with NIST
  - Evaluate Protection Profile first
  - If successful, can evaluate Security Target

Common Criteria: Status

- About 80 registered products
  - Only one at level 5 (Java Smart Card)
  - Several OS at 4
  - Likely many more not registered
- New versions appearing on regular basis

CS526, Spring 2003 42